Autómatas Finitos

Juan Mendivelso

Universidad Nacional de Colombia
Facultad de Ciencias
Departamento de Matemáticas

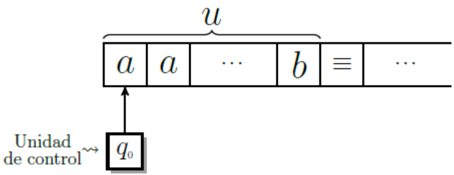
Presentación basada en las Notas de Clase del Profesor Rodrigo De Castro.

Outline

Outline

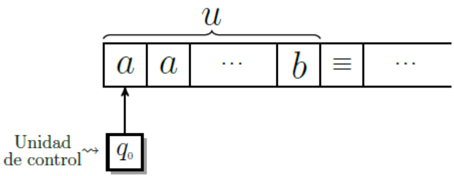
Autómatas Finitos Deterministas (AFD)

- En 1936, Alan Turing introdujo las Máquinas de Turing.
- En los 1940's y 1950's se introdujeron máquinas de Turing con capacidad restringida.
- Los autómatas finitos son máquinas abstractas que tienen la capacidad de reconocer lenguajes.



Autómatas Finitos Deterministas (AFD)

- Aceptan o rechazan cadenas de texto.
- Cinta semi-infinita dividida en celdas.
- Unidad de control, cabeza lectora, control finito o unidad de memoria.
- Estados del autómata.
- Estado inicial.
- Estado final.



Autómatas Finitos Deterministas (AFD)

Un Autómata Finito Determinista (AFD) $M = (\Sigma, Q, q_0, F, \delta)$ tiene cinco componentes:

1. Alfabeto Σ.
2. Estados $Q = \{q_0, q_1, \ldots, q_n\}$.
3. Estado inicial $q_0 \in Q$.
4. Estados finales o de aceptación $F \subseteq Q$. Se tiene que $F \neq \emptyset$.
5. Función de transición (o dinámica del autómata): δ.

Un AFD acepta la cadena u si está en un estado de aceptación al leer la primera casilla vacía.

Función de Transición (o Dinámica del Autómata)

- Lista de instrucciones de $M = (\Sigma, Q, q_0, F, \delta)$ para procesar toda $u \in \Sigma^*$:

$$\begin{aligned} \delta : Q \times \Sigma &\to Q \\ (q, s) &\mapsto \delta(q, s) \end{aligned}$$

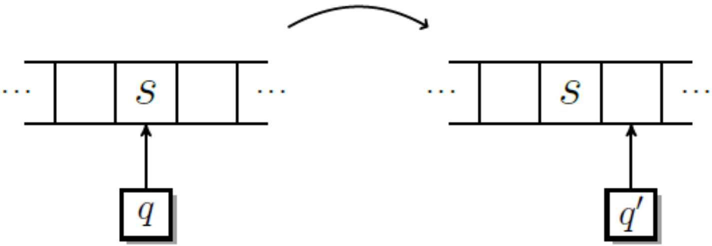
- La función δ está definida para toda pareja (q, s), $\forall q \in Q$ y $\forall s \in \Sigma$.

Paso Computacional

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD y $u \in \Sigma^*$.
- La unidad de control apunta al primer caracter u inicialmente.
- Cada procesamiento $\delta(q, s) = q'$, para $q, q' \in Q$ y $s \in \Sigma$ es un **paso computacional**.
- Al final de cada paso computacional, la unidad de control se mueve una celda a la derecha.
- M acepta la cadena u si está en un estado de aceptación al leer la primera casilla vacía.
- La cadena u se procesa de manera única.

Paso Computacional

$$\delta(q, s) = q'$$



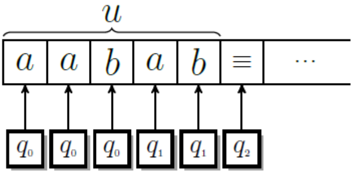
Ejemplo de AFD

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

1. $u = aabab$.



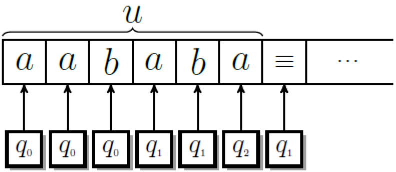
Ejemplo de AFD

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

2. $v = aababa$.



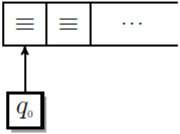
Ejemplo de AFD

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

3. Caso especial: la cadena λ es la cadena de entrada.



Función de Transición Extendida

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- La función de transición δ, $\delta : Q \times \Sigma \to Q$ se extiende a la **función de transición extendida** $\hat{\delta} : Q \times \Sigma^* \to Q$ definida así:

$$\begin{cases} \hat{\delta}(q, \lambda) = q & q \in Q \\ \hat{\delta}(q, a) = \delta(q, a) & q \in Q, a \in \Sigma \\ \hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a) & q \in Q, a \in \Sigma, q \in \Sigma^* \end{cases}$$

- $\hat{\delta}(q, u)$: Estado en el que queda la unidad de control después de procesar la cadena $u \in \Sigma^*$ partiendo del estado $q \in Q$.
- $\hat{\delta}(q, u)$ se puede denotar como $\delta(q, u)$ sin lugar a ambigüedad.

Lenguaje Aceptado (Reconocido) por un AFD

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- La unidad de control apunta al primer caracter u inicialmente.
- M acepta o rechaza las cadenas en Σ^*.
- Se puede decir que M clasifica las cadenas de Σ^* en dos clases disyuntas: las aceptadas y las rechazadas.
- El conjunto de las cadenas aceptadas por M se denomina **lenguaje aceptado (reconocido)** por M:

$$\begin{aligned} L(M) &= \{u \in \Sigma^* : u \text{ es aceptada por } M\} \\ &= \{u \in \Sigma^* : M \text{ termina el procesamiento de } u \text{ en un } q \in F\} \\ &= \{u \in \Sigma^* : \hat{\delta}(q_0, u) \in F\}. \end{aligned}$$

Estados Limbo

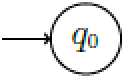
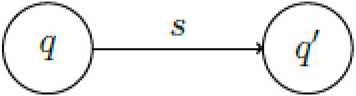
- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- Un estado $q \in Q$ es **limbo** si no existe ninguna cadena $u \in \Sigma^*$ tal que $\hat{\delta}(q, u) \in F$.
- En otras palabras, un estado $q \in Q$ es **limbo** si una vez se llega a q ya no se puede llegar a un estado de aceptación.
- Se requieren para rechazar cadenas.
- Hacen parte del autómata y deben considerarse en la función de transición.

Estados Accesibles

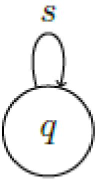
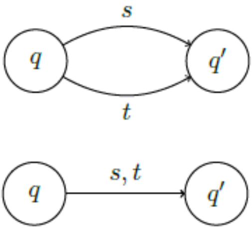
- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- Un estado $q \in Q$ es **accesible** si existe una cadena de entrada $u \in \Sigma^*$ tal que $\hat{\delta}(q_0, u) = q$.
- Los estados inaccesibles son inútiles ya que no serán accedidos por la unidad de control.
- Los estados inaccesibles se pueden eliminar sin alterar $L(M)$.
- Los estados inaccesibles son diferentes a los estados limbo, pues los primeros son inútiles mientras los últimos son necesarios para rechazar cadenas.

Grafo de un AFD

Un AFD se puede representar por un grafo dirigido y etiquetado:

Estado $q_0 \in Q$	
Estado $q \in Q - F$	
Estado $q \in F$	
Transición $\delta(q, s) = q'$	

Grafo de un AFD

Bucle $\delta(q,s) = q$	
Transiciones donde $\delta(q,s) = \delta(q,t) = q'$	

Grafo de un AFD

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- El grafo de M es útil para hacer seguimiento del procesamiento de una cadena $u \in \Sigma^*$.
- Si u es aceptada, existe una trayectoria desde q_0 hasta un estado $q \in F$ etiquetada por los caracteres de u.
- Como δ está definida para toda pareja (q, s), $\forall q \in Q$ y $\forall s \in \Sigma$, desde cada nodo del grafo salen $|\Sigma|$ arcos.
- Un estado $q \in Q$ es **limbo** si desde q no parten trayectorias que conduzcan a estados de aceptación.
- Los estados limbo hacen parte integrante del autómata. Sin embargo, en el grafo a veces se suprimen.

Ejemplo de Grafo de un AFD

Ejercicio

Dibuje el grafo para el siguiente autómata:

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1.$$

Diseño de un AFD

- Se aborda el siguiente problema: Dado un lenguaje L, diseñar un AFD M que acepte L, i.e. $L(M) = L$.
- Se debe cumplir lo siguiente:
  1. $u \in L(M) \Longrightarrow u \in L$.
  2. $u \in L \Longrightarrow u \in L(M)$.

Diseño

Ejercicio

Dado el alfabeto $\Sigma = \{a, b\}$ diseñe AFDs que acepten los siguientes lenguajes. Para cada uno muestre el grafo con y sin estados limbo.

1. $L = a^*$
2. $L = a^+$
3. $L = \{u \in \Sigma^* : u \text{ contiene exactamente dos aes}\}$
4. $L = \{u \in \Sigma^* : u \text{ tiene un número par de símbolos } \geq 0\}$
5. $L = \{u \in \Sigma^* : u \text{ tiene un número par de aes } \geq 0\}$
6. $L = (b \cup ab)^*$
7. $L = (a \cup b)^*$
8. $L = \emptyset$

Outline

1. Autómatas Finitos Deterministas
   - Definiciones Básicas
   - Complemento
   - Producto Cartesiano
   - Minimización
2. Autómatas Finitos No Deterministas
   - Definición y Representación
   - Equivalencia Computacional entre los AFD y los AFN
3. Autómatas con Transiciones λ
   - Definición y Representación
   - Equivalencia Computacional entre los AFN y los AFN-λ
4. Lenguajes Regulares & Autómatas Finitos
   - Teorema de Kleene
   - Teorema de Myhill-Nerode
   - Pruebas de Regularidad

Complemento

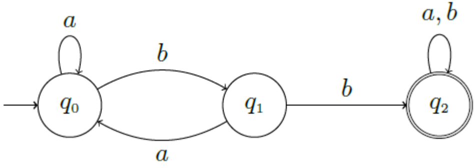
- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- El **complemento** de M es $\bar{M} = (\Sigma, Q, q_0, \bar{F}, \delta)$ donde $\bar{F} = Q - F$.
- Se intercambian los estados de aceptación y no aceptación.
- Si $L(M) = L$, $L(\bar{M}) = \bar{L} = \Sigma^* - L$.
- Son útiles cuando el lenguaje está definido por medio de una condición negativa.

Complemento

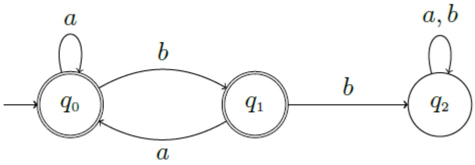
Ejercicio

- Sea $\Sigma = \{a, b\}$. Encontrar un AFD que acepte el lenguaje de todas las cadenas que no tienen dos *b*es consecutivas.

M:



$\overline{M}$:



Complemento

Ejercicio

Sea $\Sigma = \{a, b, c\}$. Encontrar un AFD que acepte el lenguaje de todas las cadenas que ...

- No contienen la subcadena bc.
- No terminan en bb.

Outline

Producto Cartesiano

- Sean $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ y $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ dos AFDs definidos sobre el alfabeto Σ, tales que $L(M_1) = L_1$ y $L(M_2) = L_2$.
- El **producto cartesiano** de M_1 y M_2 es

$$M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$$

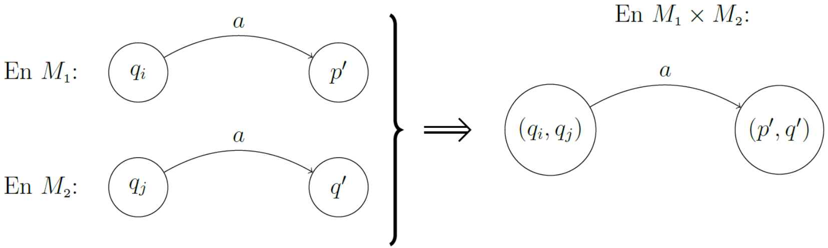
  donde la función de transición δ está dada por $\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2$, definida por $\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$, y el conjunto de estados de aceptación se escoge a conveniencia según el lenguaje a aceptar:

  1. Para $L(M_1 \times M_2) = L_1 \cup L_2$, escoger $F = \{(q_i, q_j) : q_i \in F_1 \vee q_j \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
  2. Para $L(M_1 \times M_2) = L_1 \cap L_2$, escoger $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \in F_2\} = F_1 \times F_2$.
  3. Para $L(M_1 \times M_2) = L_1 - L_2$, escoger $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \notin F_2\} = F_1 \times (Q_2 - F_2)$.

Función de Transición en el Producto Cartesiano

- Sean $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ y $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ dos AFDs definidos sobre el alfabeto Σ, tales que $L(M_1) = L_1$ y $L(M_2) = L_2$.
- Sea $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$ su producto cartesiano.
- La función de transición δ está dada por

$$\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2$$
$$\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$$



Función de Transición Extendida en el Producto Cartesiano

Lema

- Sean $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ y $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ dos AFDs definidos sobre el alfabeto Σ, tales que $L(M_1) = L_1$ y $L(M_2) = L_2$.
- Sea $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$ su producto cartesiano donde $\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$.
- La función de transición δ se puede extender a cadenas arbitrarias:

$$\hat{\delta}((q_i, q_j), u) = (\hat{\delta}_1(q_i, u), \hat{\delta}_2(q_j, u)), \forall u \in \Sigma^*, q_i \in Q_1, q_j \in Q_2.$$

- Para la demostración (inductiva), recordar que para un AFD $M = (\Sigma, Q, q_0, F, \delta)$, la función de transición extendida se define:
  1. $\hat{\delta}(q, \lambda) = q$ para $q \in Q$.
  2. $\hat{\delta}(q, a) = \delta(q, a)$ para $q \in Q$ y $a \in \Sigma$.
  3. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a)$ para $q \in Q$, $u \in \Sigma^*$ y $a \in \Sigma$.

Estados de Aceptación del Producto Cartesiano

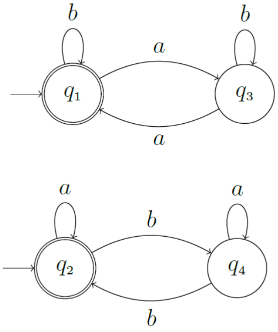
Teorema

- Sean $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ y $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ dos AFDs definidos sobre el alfabeto Σ, tales que $L(M_1) = L_1$ y $L(M_2) = L_2$.
- Sea $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$ el producto cartesiano de estos.
- Entonces,
  1. $F = \{(q_i, q_j) : q_i \in F_1 \vee q_j \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$, si y solo si $L(M_1 \times M_2) = L_1 \cup L_2$.
  2. $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \in F_2\} = F_1 \times F_2$, si y solo si $L(M_1 \times M_2) = L_1 \cap L_2$.
  3. $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \notin F_2\} = F_1 \times (Q_2 - F_2)$, si y solo si $L(M_1 \times M_2) = L_1 - L_2$.

Ejemplos de Producto Cartesiano de AFDs

Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre $\Sigma = \{a, b\}$ que tienen un número par de $a$es y un número par de $b$es.

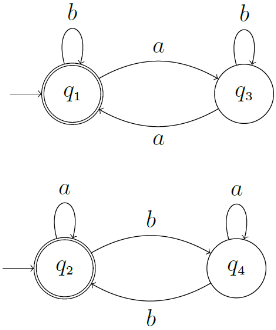


$$\begin{cases} \delta((q_1,q_2),a) = (\delta_1(q_1,a),\delta_2(q_2,a)) = (q_3,q_2), \\ \delta((q_1,q_2),b) = (\delta_1(q_1,b),\delta_2(q_2,b)) = (q_1,q_4), \\ \delta((q_1,q_4),a) = (\delta_1(q_1,a),\delta_2(q_4,a)) = (q_3,q_4), \\ \delta((q_1,q_4),b) = (\delta_1(q_1,b),\delta_2(q_4,b)) = (q_1,q_2), \\ \delta((q_3,q_2),a) = (\delta_1(q_3,a),\delta_2(q_2,a)) = (q_1,q_2), \\ \delta((q_3,q_2),b) = (\delta_1(q_3,b),\delta_2(q_2,b)) = (q_3,q_4), \\ \delta((q_3,q_4),a) = (\delta_1(q_3,a),\delta_2(q_4,a)) = (q_1,q_4), \\ \delta((q_3,q_4),b) = (\delta_1(q_3,b),\delta_2(q_4,b)) = (q_3,q_2). \end{cases}$$

Ejemplos de Producto Cartesiano de AFDs

Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre $\Sigma = \{a, b\}$ que tienen un número par de *a*es y un número par de *b*es.

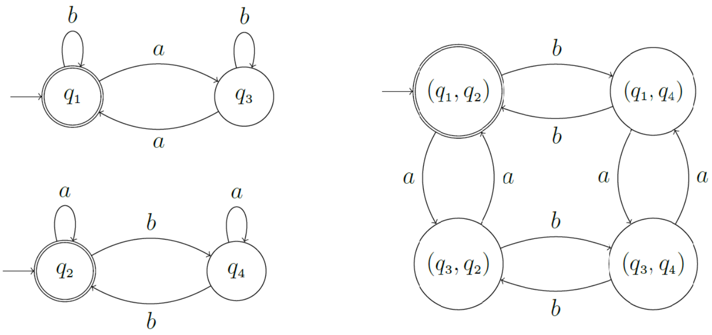


$$\begin{cases} \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) = (q_3, q_2), \\ \delta((q_1, q_2), b) = (\delta_1(q_1, b), \delta_2(q_2, b)) = (q_1, q_4), \\ \delta((q_1, q_4), a) = (\delta_1(q_1, a), \delta_2(q_4, a)) = (q_3, q_4), \\ \delta((q_1, q_4), b) = (\delta_1(q_1, b), \delta_2(q_4, b)) = (q_1, q_2), \\ \delta((q_3, q_2), a) = (\delta_1(q_3, a), \delta_2(q_2, a)) = (q_1, q_2), \\ \delta((q_3, q_2), b) = (\delta_1(q_3, b), \delta_2(q_2, b)) = (q_3, q_4), \\ \delta((q_3, q_4), a) = (\delta_1(q_3, a), \delta_2(q_4, a)) = (q_1, q_4), \\ \delta((q_3, q_4), b) = (\delta_1(q_3, b), \delta_2(q_4, b)) = (q_3, q_2). \end{cases}$$

Ejemplos de Producto Cartesiano de AFDs

Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre $\Sigma = \{a, b\}$ que tienen un número par de $a$es y un número par de $b$es.



Ejemplos de Producto Cartesiano de AFDs

Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre $\Sigma = \{a, b\}$ que ...

1. tienen longitud impar y que no contienen dos *b*es consecutivas.
2. tienen longitud impar o que no contienen dos *b*es consecutivas.

¿Cuándo se pueden omitir los estados limbo?

Outline

Minimización de AFDs

Problema

Dado un AFD M, encontrar un AFD M' con el mínimo número de estados posible tal que $L(M') = L(M)$.

Estados Equivalentes

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD y $p, q \in Q$. Se dice que p y q son **equivalentes**, notado $p \approx q$, si y solo si

$$(\forall u \in \Sigma^*)[\hat{\delta}(p, u) \in F \iff \hat{\delta}(q, u) \in F].$$

- Dados $p, q, r \in Q$, la relación $\approx$ cumple con las siguientes propiedades:
  - Reflexividad: $p \approx p$.
  - Simetría: Si $p \approx q$, entonces $q \approx p$.
  - Transitividad: Si $p \approx q$ y $q \approx r$, entonces $p \approx r$.
- Entonces, $\approx$ es una relación de equivalencia sobre Q.
- La **clase de equivalencia** de $p \in Q$, respecto a $\approx$, es:

$$[p]_\approx := \{q \in Q : p \approx q\}.$$

Autómata Cociente

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD.
- El autómata cociente de M es $M' = (\Sigma, Q', q_0', F', \delta')$ donde
  - $Q' = \{[p]_\approx : p \in Q\}$
  - $q_0' = [q_0]_\approx$
  - $F' = \{[p]_\approx : p \in F\}$
  - $\delta'([p]_\approx, a) = [\delta(p, a)]_\approx$ para todo $a \in \Sigma$ y para todo $p \in Q$.
- Se requiere verificar que tanto F' como δ' están bien definidos: no dependen del representante escogido de cada clase de equivalencia.

Validez de F' y δ'

Proposición

1. δ' está bien definida: si $[p]_{\approx} = [q]_{\approx}$, entonces $\delta(p, a) \approx \delta(q, a)$.
2. F' está bien definido: si $q \in F$ y $p \approx q$, entonces $p \in F$.
3. $p \in F \iff [p]_{\approx} \in F'$.
4. $\hat{\delta}'([p]_{\approx}, u) = [\hat{\delta}(p, u)]_{\approx}$ para toda cadena $u \in \Sigma^*$.

Algoritmo de Minimización

Algoritmo por llenado de tabla para determinar la equivalencia de estados en un AFD

ENTRADA:

AFD $M = (\Sigma, Q, q_0, F, \delta)$ completo (incluyendo estados limbo) cuyos estados son todos accesibles y tabla triangular que muestra todos los pares $\{p, q\}$ de estados p, $q \in Q$.

INICIALIZAR:

$i := 1$. Se marca con $\times$ la casilla $\{p, q\}$ si $p \in F$ y $q \notin F$ (o viceversa).

REPETIR:

$i := i + 1$. Para cada casilla no marcada $\{p, q\}$ y cada $a \in \Sigma$, hallar $\{\delta(p, a), \delta(q, a)\}$. Si para algún $a \in \Sigma$, la casilla $\{\delta(p, a), \delta(q, a)\}$ ha sido marcada previamente, entonces se marca la casilla $\{p, q\}$ con $\times$.

HASTA:

No se puedan marcar más casillas en la tabla.

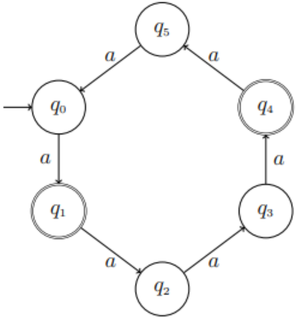
SALIDA:

Si la casilla $\{p, q\}$ está marcada con $\times$, entonces $p \not\approx q$. Si la casilla $\{p, q\}$ no está marcada, entonces $p \approx q$.

Algoritmo de Minimización

Ejercicio

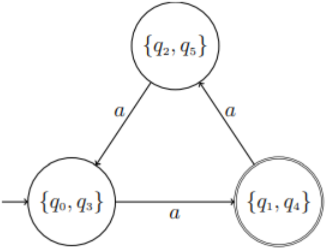
Minimizar el siguiente AFD, donde $\Sigma = \{a\}$.



	q_0	q_1	q_2	q_3	q_4
q_1	×				
q_2		×			
q_3		×			
q_4	×		×	×	
q_5		×			×

	q_0	q_1	q_2	q_3	q_4
q_1	×				
q_2	×	×			
q_3		×	×		
q_4	×		×	×	
q_5	×	×		×	×

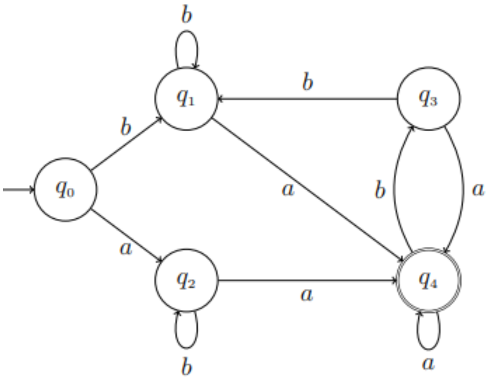
$\{p, q\}$	$\{\delta(p,a), \delta(q,a)\}$
$\{q_0, q_2\}$	$\{q_1, q_3\}$ ×
$\{q_0, q_3\}$	$\{q_1, q_4\}$
$\{q_0, q_5\}$	$\{q_1, q_0\}$ ×
$\{q_1, q_4\}$	$\{q_2, q_5\}$
$\{q_2, q_3\}$	$\{q_3, q_4\}$ ×
$\{q_2, q_5\}$	$\{q_3, q_0\}$
$\{q_3, q_5\}$	$\{q_4, q_0\}$ ×



Algoritmo de Minimización

Ejercicio

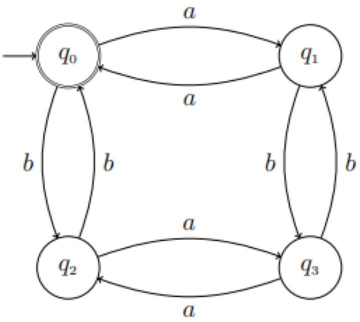
Minimizar el siguiente AFD, donde $\Sigma = \{a, b\}$.



Algoritmo de Minimización

Ejercicio

Mostrar que el siguiente AFD, donde $\Sigma = \{a, b\}$, no se puede simplificar.

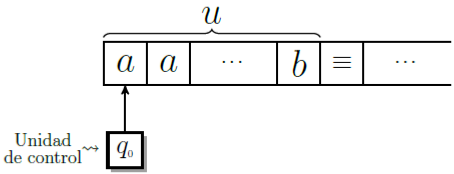


Outline

Outline

Autómatas Finitos No Deterministas (AFN)

- Aceptan o rechazan cadenas de texto.
- Cinta semi-infinita dividida en celdas.
- Unidad de control, cabeza lectora, control finito o unidad de memoria.
- Estados del autómata, incluyendo inicial y final(es).
- Es no determinista porque una cadena se puede procesar de varias formas y puede haber procesamientos abortados.
- La función de transición para determinado estado y símbolo puede conducir a varios estados o a ningún estado.



Autómatas Finitos No Deterministas (AFN)

Un Autómata Finito No Determinista (AFN) $M = (\Sigma, Q, q_0, F, \Delta)$ tiene cinco componentes:

1. Alfabeto Σ.
2. Estados $Q = \{q_0, q_1, \ldots, q_n\}$.
3. Estado inicial $q_0 \in Q$.
4. Estados finales o de aceptación $F \subseteq Q$. Se tiene que $F \neq \emptyset$.
5. Función de transición (o dinámica del autómata): Δ.

Un AFN acepta la cadena u si existe algún procesamiento completo de u que termine en un estado de aceptación.

Función de Transición (o Dinámica del Autómata)

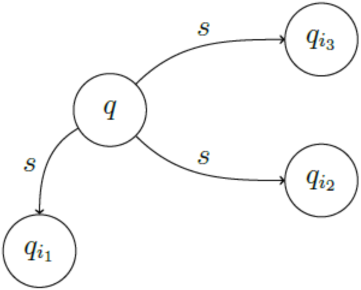
- Lista de instrucciones de $M = (\Sigma, Q, q_0, F, \delta)$ para procesar toda $u \in \Sigma^*$:

$$\Delta : Q \times \Sigma \to \mathcal{P}(Q)$$
$$(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$$

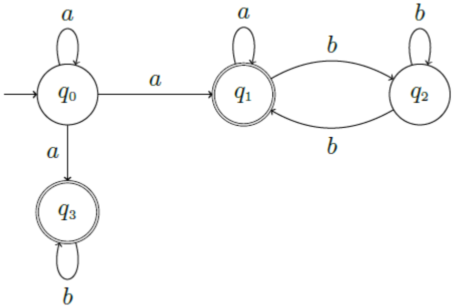
- Esto quiere decir que estando en el estado q, en presencia del símbolo s, la unidad de control puede pasar aleatoriamente a uno cualquiera de los estados $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$, después de lo cual se desplaza a la derecha.
- Puede suceder que $\Delta(q, s) = \emptyset$: si al procesar la cadena u, la cabeza lectora de M ingresa al estado q, leyendo sobre la cinta el símbolo s, el procesamiento se aborta.

Función de Transición (o Dinámica del Autómata)

- Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN con función de transición $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, donde $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$.
- La representación del AFN como digrafo es similar a la de AFD.
- Si $\Delta(q, s) = \{q_{i_1}, q_{i_2}, q_{i_3}\}$, esto se puede representar en el autómata de la siguiente manera.

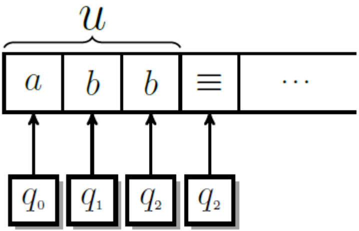


Ejemplo de AFN

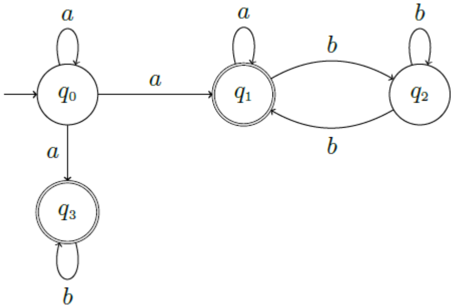


Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Procesamiento de rechazo:

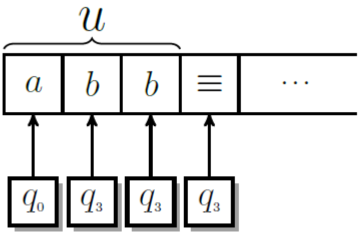


Ejemplo de AFN

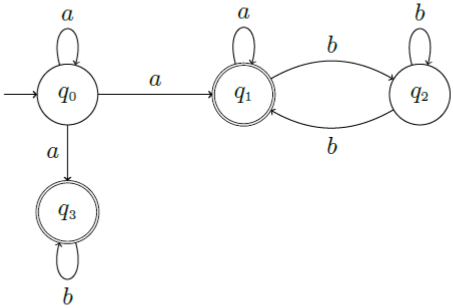


Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Procesamiento de aceptación:

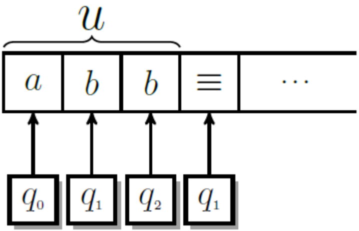


Ejemplo de AFN

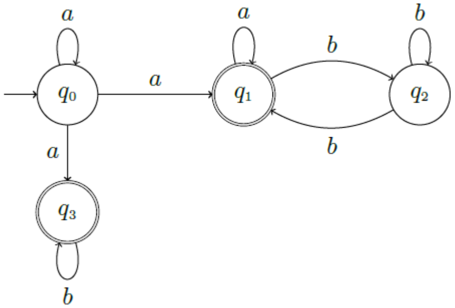


Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Otro procesamiento de aceptación:

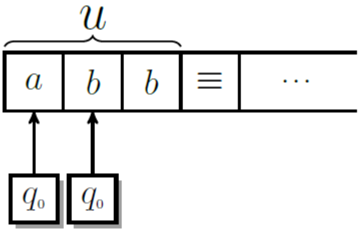


Ejemplo de AFN

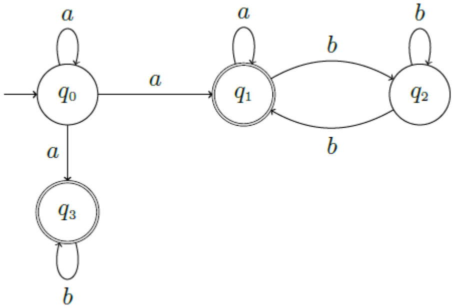


Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Procesamiento abortado:

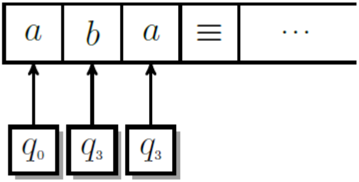


Ejemplo de AFN



Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Procesamiento abortado:



Las cadenas que empiezan por b no pueden ser aceptadas. Tampoco *aba*.

Función de Transición para Conjuntos de Estados

- Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN con función de transición $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, donde $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$.
- La función de transición Δ se extiende a la **función de transición para conjuntos de estados** $\Delta : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ definida así: Para $a \in \Sigma$ y $S \subseteq Q$,

$$\Delta(S, a) := \bigcup_{q \in S} \Delta(q, a).$$

- $\Delta(S, a)$ es el conjunto de estados a los que se puede llegar al procesar a partiendo de alguno de los estados en S.
- $\Delta(S, a) = \emptyset$ si $\Delta(q, a) = \emptyset$ para todo $q \in S$.

Función de Transición Extendida

- Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN con función de transición $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$, donde $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$.
- La función de transición Δ se extiende a conjuntos de estados definida así: Para $a \in \Sigma$ y $S \subseteq Q$, $\Delta(S, a) := \bigcup_{q \in S} \Delta(q, a)$.
- La función de transición Δ se extiende a la **función de transición extendida** $\hat{\Delta} : Q \times \Sigma^* \to Q$ definida así:

$$\begin{cases} \hat{\Delta}(q, \lambda) = q & q \in Q \\ \hat{\Delta}(q, a) = \Delta(q, a) & q \in Q, a \in \Sigma \\ \hat{\Delta}(q, ua) = \Delta(\hat{\Delta}(q, u), a) & q \in Q, a \in \Sigma, q \in \Sigma^*. \end{cases}$$

- $\hat{\delta}(q, u)$: Estados a los que puede llegar la unidad de control con *procesamientos completos* de la cadena $u \in \Sigma^*$ partiendo de $q \in Q$.
- $\hat{\Delta}(q, u)$ se puede denotar como $\Delta(q, u)$ sin lugar a ambigüedad.

Lenguaje Aceptado (Reconocido) por un AFN

- Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN.
- La unidad de control apunta al primer caracter u inicialmente.
- M acepta o rechaza las cadenas en Σ^*, i.e. M clasifica las cadenas de Σ^* en dos clases disyuntas: las aceptadas y las rechazadas.
- El conjunto de las cadenas aceptadas por M se denomina **lenguaje aceptado (reconocido)** por M:

$$\begin{aligned}
L(M) &= \{u \in \Sigma^* : u \text{ es aceptada por } M\} \\
&= \{u \in \Sigma^* : M \text{ tiene al menos un procesamiento completo de} \\
&\qquad\qquad u \text{ que parte de } q_0 \text{ y termina en } q \in F\} \\
&\quad\ \{u \in \Sigma^* : \hat{\Delta}(q_0, u) \text{ contiene algún estado } q \in F\} \\
&= \{u \in \Sigma^* : \hat{\Delta}(q_0, u) \cap F \neq \emptyset\}.
\end{aligned}$$

Diseño de un AFN

- Se aborda el siguiente problema: Dado un lenguaje L, diseñar un AFN M que acepte L, i.e. $L(M) = L$.
- Se debe cumplir lo siguiente:
  1. $u \in L(M) \Longrightarrow u \in L$.
  2. $u \in L \Longrightarrow u \in L(M)$.

Diseño

Ejercicio

Dado el alfabeto $\Sigma = \{a, b\}$ diseñe AFNs que acepten los siguientes lenguajes. Dibuje AFDs completos que acepten los mismos lenguajes.

1. $L = ab^* \cup a^+$
2. $L = (ab \cup aba)^*$
3. $L = a(a \cup ab)^*$
4. $L = a^+ b^* a$

Outline

Equivalencia Computacional entre los AFD y los AFN

- Los modelos AFD Y AFN son computacionalmente equivalentes: aceptan los mismos lenguajes.
- Un AFD $M = (\Sigma, Q, q_0, F, \delta)$ puede ser considerado como un AFN $M' = (\Sigma, Q, q_0, F, \Delta)$ definiendo $\Delta(q, a) = \{\delta(q, a)\}$ para cada $q \in Q$ y cada $a \in \Sigma$.
- La afirmación recíproca también es correcta.

Equivalencia Computacional entre los AFD y los AFN

Teorema

Dado un AFN $M = (\Sigma, Q, q_0, F, \Delta)$, se puede construir un AFD $M' = (\Sigma, \mathcal{P}(Q), \{q_0\}, F', \delta\})$ equivalente a M, i.e. tal que $L(M) = L(M')$, donde

$$
\begin{aligned}
\delta : \mathcal{P}(Q) \times \Sigma &\to \mathcal{P}(Q) \\
(S, a) &\mapsto \delta(S, a) := \Delta(S, a) \\
F' &= \{S \subseteq Q : S \cap F \neq \emptyset\}.
\end{aligned}
$$

.

Algoritmo para encontrar un AFD equivalente a un AFN

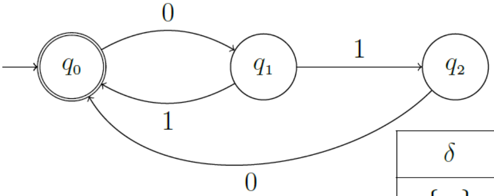
Dado un AFN $M = (\Sigma, Q, q_0, F, \Delta)$, encontrar un AFD $M' = (\Sigma, Q', q_0, F', \delta)$ tal que $L(M) = L(M')$:

1. Para cada estado $q \in Q$, incluir el estado $\{q\}$ en Q'.
2. Establecer como estado inicial a $\{q_0\}$.
3. Para cada $q \in Q$ y cada $a \in \Sigma$, incluir el conjunto de estados $\Delta(q, a) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\} = S$ como un único estado en Q'. Incluir $\delta(\{q\}, a) = \Delta(q, a)$ en la función δ.
4. Para cada estado nuevo $S \in Q'$ y para cada $a \in \Sigma$, hallar $\delta(S, a) = \Delta(S, a) = S'$ y, si $S' \notin Q'$, incluir S' en Q' como un único estado.
5. Repetir el Paso 4 hasta que no se puedan añadir más estados a Q'.
6. Eliminar de Q' los estados inalcanzables.
7. Incluir en F' aquellos elementos de $S \in Q'$ tales que $S \cap F \neq \emptyset$.

Ejemplo

Ejercicio

Hallar un AFD que acepte el lenguaje $L = (01 \cup 010)^*$ sobre $\Sigma = \{0, 1\}$.



Δ	0	1
q_0	$\{q_1\}$	$\varnothing$
q_1	$\varnothing$	$\{q_0, q_2\}$
q_2	$\{q_0\}$	$\varnothing$

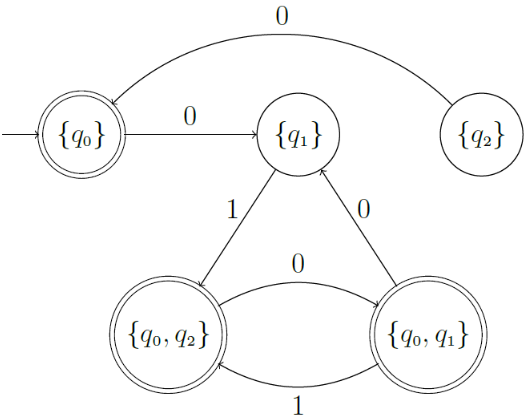
δ	0	1
$\{q_0\}$	$\{q_1\}$	$\varnothing$
$\{q_1\}$	$\varnothing$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\varnothing$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\varnothing$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$

Ejemplo

Ejercicio

Hallar un AFD que acepte el lenguaje $L = (01 \cup 010)^*$ sobre $\Sigma = \{0, 1\}$.

δ	0	1
$\{q_0\}$	$\{q_1\}$	$\varnothing$
$\{q_1\}$	$\varnothing$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\varnothing$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\varnothing$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$

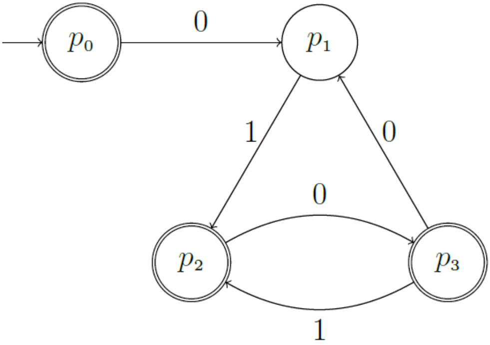


Ejemplo

Ejercicio

Hallar un AFD que acepte el lenguaje $L = (01 \cup 010)^*$ sobre $\Sigma = \{0, 1\}$.

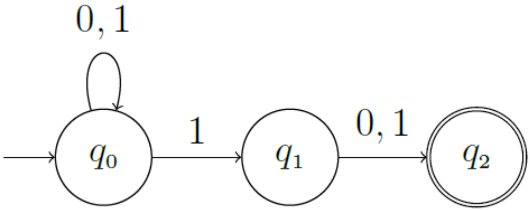
δ	0	1
$\{q_0\}$	$\{q_1\}$	$\varnothing$
$\{q_1\}$	$\varnothing$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\varnothing$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\varnothing$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$



Ejercicio

Ejercicio

Hallar un AFD que acepte el lenguaje de todas las cadenas de longitud ≥ 2 en las que el penúltimo símbolo es uno, sobre $\Sigma = \{0, 1\}$.



Outline

Outline

Autómatas con Transiciones λ (AFN-λ)

- Un **autómata finito con transiciones λ (AFN-λ)** es un autómata finito no determinista $M = (\Sigma, Q, q_0, F, \Delta)$ en el que la función de transición está definida como:

$$\Delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$$

- Además de las transiciones no deterministas usuales, permite hacer **transiciones nulas** o **transiciones espontáneas**:

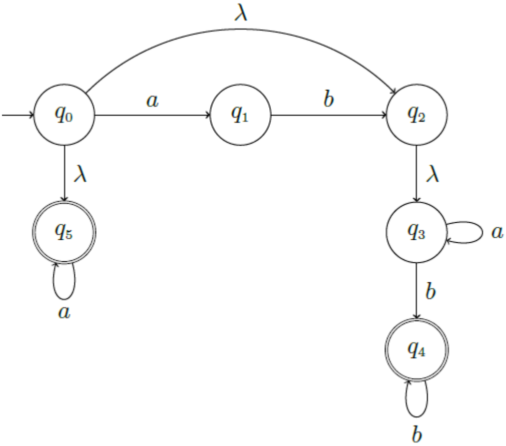
$$\Delta(q, \lambda) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}.$$

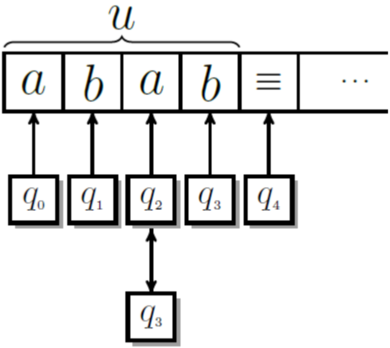
- Esta transición significa: estando en el estado q, el autómata puede cambiar aleatoriamente a cualquiera de los estados $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$, independientemente del símbolo leído y sin mover la unidad de control a la derecha.

Autómatas con Transiciones λ (AFN-λ)

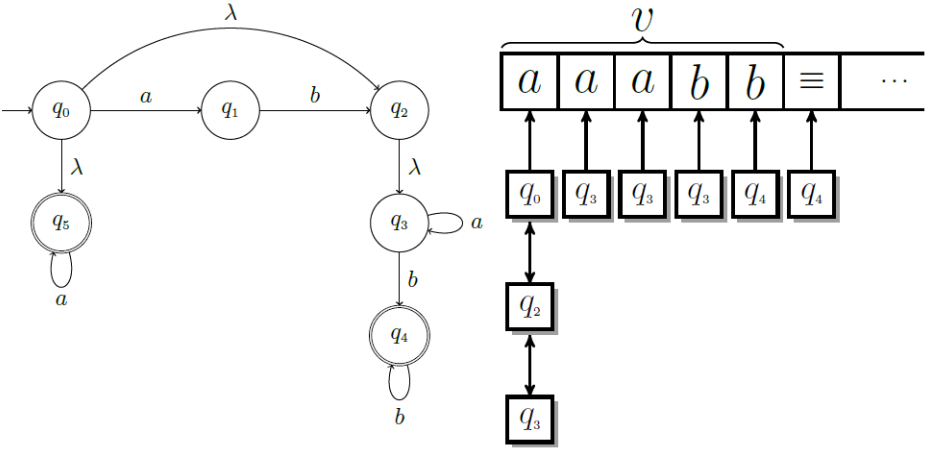
- Un AFN-λ $M = (\Sigma, Q, q_0, F, \Delta)$ puede cambiar de estado sin consumir ningún símbolo de la cinta.
- En el grafo las transiciones pueden tener la etiqueta λ.
- Puede tener múltiples procesamientos de una cadena incluyendo procesamientos abortados, de aceptación y de rechazo.
- Una cadena $u \in \Sigma^*$ es aceptada si existe por lo menos un procesamiento completo de u, desde q_0, que termina en un estado de aceptación, i.e. $\hat{\Delta}(q_0, u) \cap F \neq \emptyset$.
- En el grafo, se dice que $u \in \Sigma^*$ es aceptada si existe por lo menos una trayectoria desde q_0 hasta un estado de aceptación, cuyas etiquetas son los símbolos de u intecalados con cero, uno o más λs.

Ejemplo


λ
q_0
a
q_1
b
q_2
λ
λ
q_5
a
q_3
a
b
q_4
b


u
a b a b $\equiv$ $\cdots$
q_0 q_1 q_2 q_3 q_4
q_3

Ejemplo



Ejercicio

Ejercicio

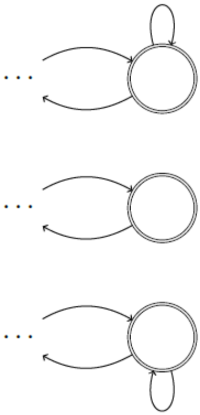
Dado $\Sigma = \{a, b\}$, diseñar AFN-λ, diseñar autómatas que acepten los siguientes lenguajes:

1. $a^* \cup (ab \cup ba)^* \cup b^+$
2. $a^*(ab \cup ba)^* b^+$

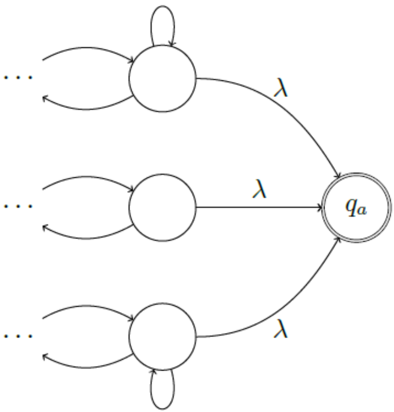
Estado de Aceptación Único

- Un autómata cualquiera M se puede modificar, sin alterar el lenguaje aceptado, de manera que tenga un único estado de aceptación.
- Se agrega un estado q_a que será el único estado de aceptación.
- Se trazan transiciones λ desde los estados de aceptación originales hasta q_a.
- El resto de transiciones se dejan iguales.
- Los estados de aceptación originales se convierten en estados de no aceptación.

Estado de Aceptación Único



Tres estados de aceptación



Único estado de aceptación q_a

Outline

Equivalencia Computacional entre los AFN y los AFN-λ

- Los modelos AFN Y AFN-λ son computacionalmente equivalentes: aceptan los mismos lenguajes.
- Un AFN puede ser considerado como un AFN-λ en el que simplemente hay cero transiciones λ.
- Dado un AFN-λ, también se puede construir un AFN que acepte el mismo lenguaje.
- Este proceso se basa en añadir transiciones que simulen las transiciones λ.
- Dichas simulaciones están basadas en el concepto de λ-clausura.

λ-Clausura

- Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN-λ.
- La **λ-clausura de un estado** $q \in Q$, denotada como $\lambda[q]$ es el conjunto de estados de Q a los que se puede llegar desde q mediante 0, 1 o más transiciones λ.
- La **λ-clausura de un conjunto de estados** $\{q_1, \ldots, q_k\}$, donde $q_i \in Q$ para $i \in \{1, 2, \ldots, k\}$, es:

$$\lambda[\{q_1, \ldots, q_k\}] = \lambda[q_1] \cup \lambda[q_2] \cup \cdots \cup \lambda[q_k].$$

- $\lambda[\emptyset] := \emptyset$.

Equivalencia Computacional entre los AFN y los AFN-λ

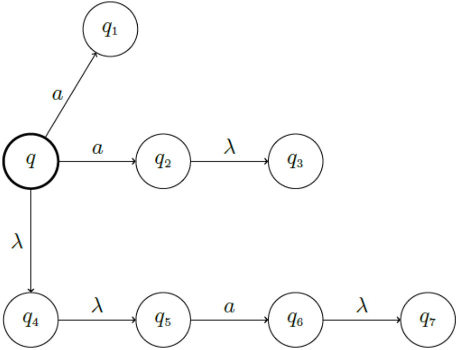
Teorema

Dado un AFN-λ $M = (\Sigma, Q, q_0, F, \Delta)$, se puede construir un AFN $M' = (\Sigma, Q, q_0, F', \Delta')$ sin transiciones λ equivalente a M, i.e. tal que $L(M) = L(M')$, donde

$$\begin{aligned} \Delta' : Q \times \Sigma &\to \mathcal{P}(Q) \\ (q, a) &\mapsto \Delta'(q, a) := \lambda[\Delta(\lambda[q], a)] \\ F' &= \{q \in Q : \lambda[q] \cap F \neq \emptyset\}. \end{aligned}$$

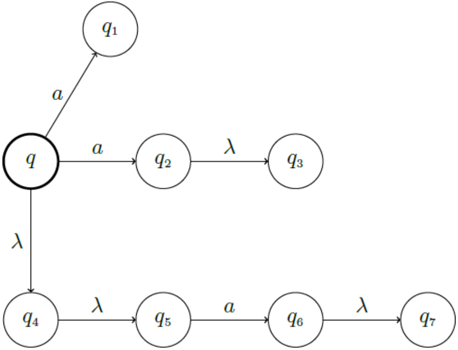
.

Función de Transición Δ'



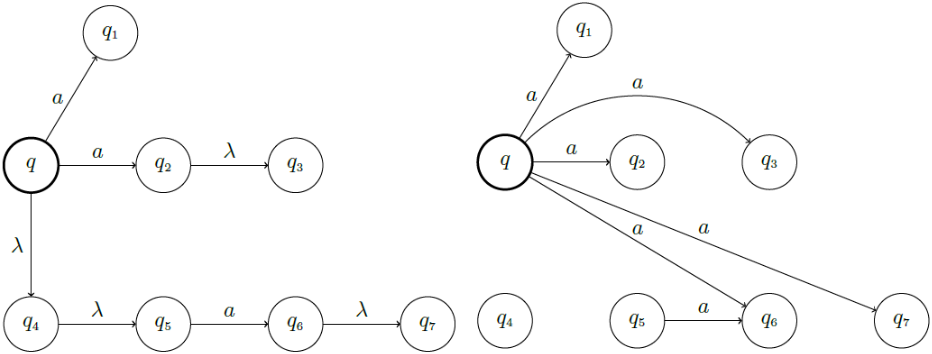
- Una vez procesada a, el autómata puede pasar desde el estado q a cualquierar de $\{q_1, q_2, q_3, q_6, q_7\}$.
- Se tienen en cuenta todas las transiciones λ que preceden o prosiguen el procesamiento del símbolo a desde el estado q.

Función de Transición Δ'



$$\begin{aligned}\Delta'(q,a) &= \lambda[\Delta(\lambda[q],a)]\\ &= \lambda[\Delta(\{q,q_4,q_5\},a)]\\ &= \lambda[\{q_1,q_2,q_6\}]\\ &= \lambda[q_1]\cup\lambda[q_2]\cup\lambda[q_6]\\ &= \{q_1\}\cup\{q_2,q_3\}\cup\{q_6,q_7\}\\ &= \{q_1,q_2,q_3,q_6,q_7\}\end{aligned}$$

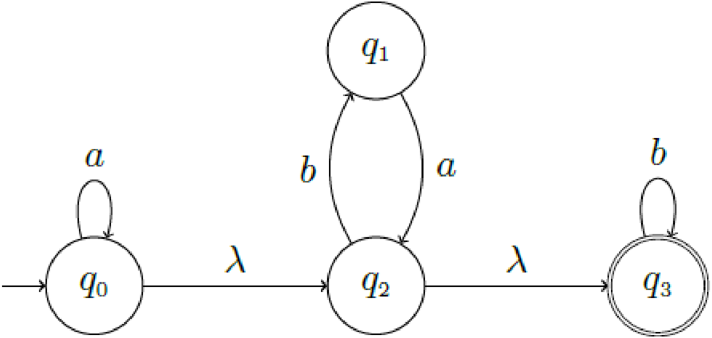
Función de Transición Δ'



Ejercicio

Ejercicio

Encontrar un AFN equivalente al siguiente AFN-λ.



Outline

Outline

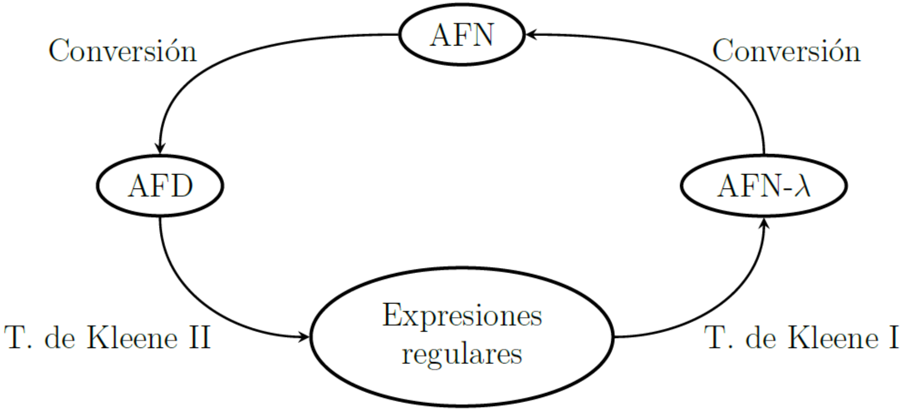
Teorema de Kleene

- Se ha mostrado la equivalencia computacional entre los modelos AFD, AFN y AFN-λ
- Es decir, aceptan la misma colección de lenguajes.
- El Teorema de Kleene establece que dicha colección corresponde a los lenguajes regulares.

Teorema de Kleene

Sea Σ un alfabeto dado. Un lenguaje sobre Σ es regular si y sólo si es aceptado por un autómata finito (AFD, AFN o AFN-λ) con alfabeto de entrada Σ.

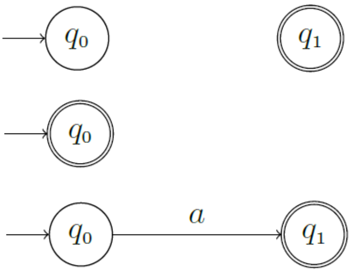
Teorema de Kleene



Teorema de Kleene - Parte I

Para un lenguaje regular, representado por una expresión regular R dada, se puede construir un AFN-λ M tal que $L(M) = R$.

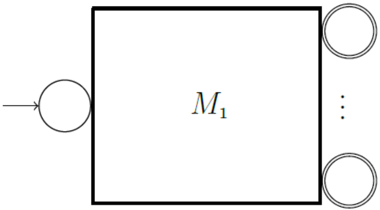
Se prueba para los lenguajes regulares básicos: $\emptyset$, $\{\lambda\}$, $a \in \Sigma$.



Teorema de Kleene - Parte I

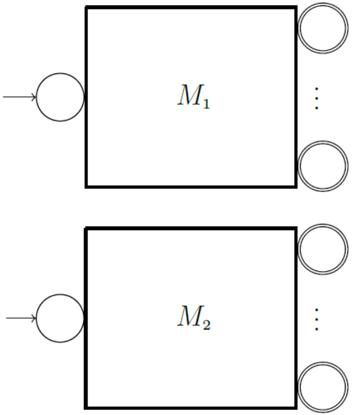
Para un lenguaje regular, representado por una expresión regular R dada, se puede construir un AFN-λ M tal que $L(M) = R$.

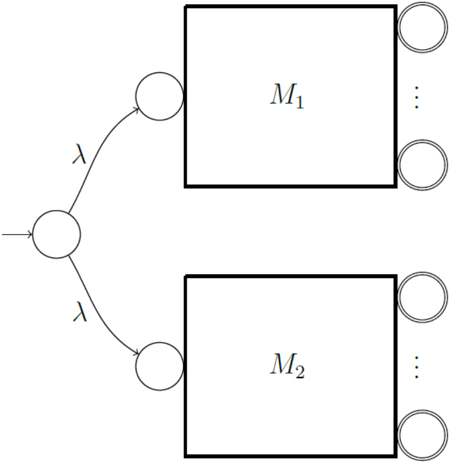
- Razonando recursivamente, supóngase que para las expresiones regulares R_1 y R_2 se tienen autómatas AFN-λ M_1 y M_2 tales que $L(M_1) = R_1$ y $L(M_2) = R_2$.
- Probar que también se pueden construir autómatas AFN-λ para aceptar $R_1 \cup R_2$ y $R_1 R_2$.



Teorema de Kleene - Parte I

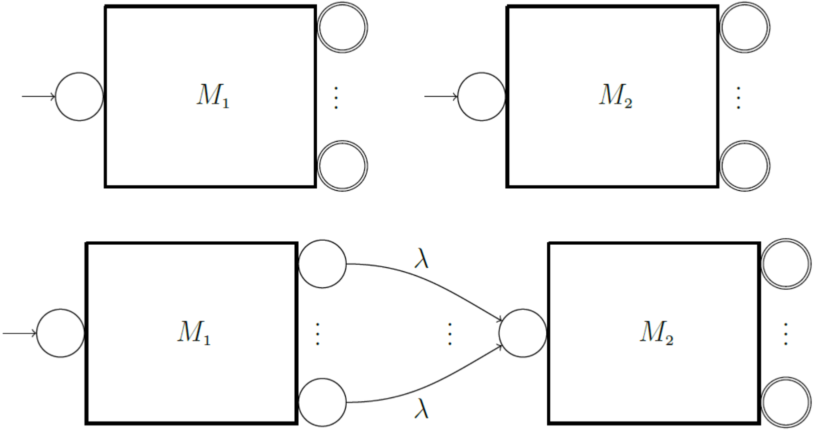
Para aceptar $R_1 \cup R_2$, se realiza una **conexión en paralelo** a M_1 y M_2. Se agrega un estado inicial y se mantienen estados de aceptación.





Teorema de Kleene - Parte I

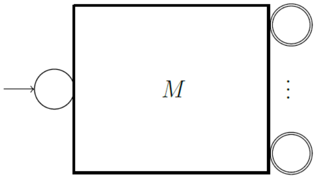
Para aceptar $R_1 R_2$, se realiza una **conexión en serie** a M_1 y M_2. Se mantiene el estado de inicial de M_1 y los estados de aceptación de M_2.

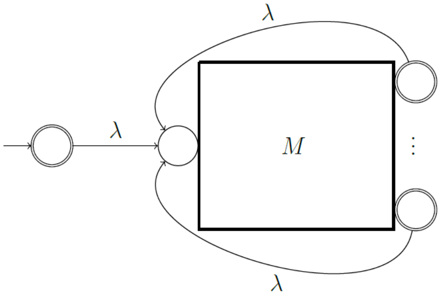


Teorema de Kleene - Parte I

Para un lenguaje regular, representado por una expresión regular R dada, se puede construir un AFN-λ M tal que $L(M) = R$.

- Razonando recursivamente, supóngase que para la expresión regular R se tiene un autómata AFN-λ M tal que $L(M) = R$.
- Probar que también se pueden construir un autómata AFN-λ para aceptar R^*.

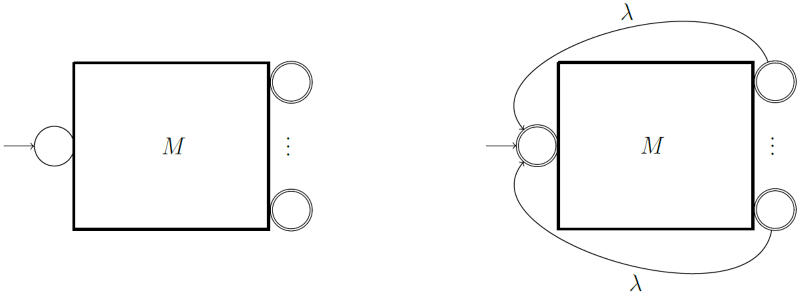




Teorema de Kleene - Parte I

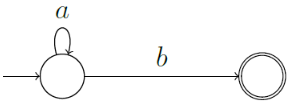
Para un lenguaje regular, representado por una expresión regular R dada, se puede construir un AFN-λ M tal que $L(M) = R$.

- Se podría pensar que el siguiente autómata acepta R^*, pero no es así.
- En la siguiente diapositiva, se muestra un contraejemplo.

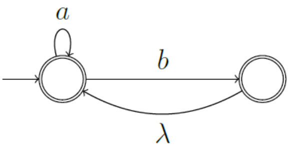


Teorema de Kleene - Parte I

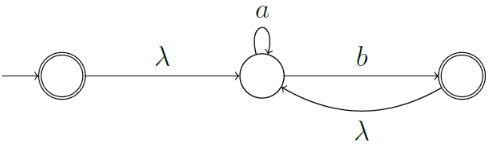
M:



M':



M'':

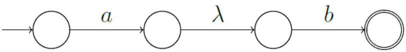


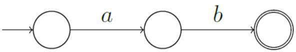
Simplificaciones

Según el procedimiento:

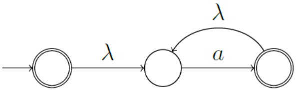
Simplificación:

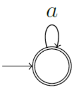
Para aceptar ab:



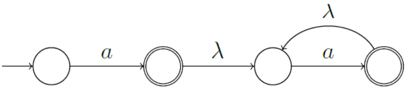


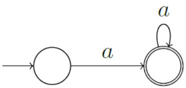
Para aceptar a^*:

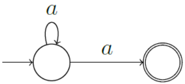




Para aceptar a^+:







Ejercicio

Ejercicio

Construir un AFN-λ que acepte el lenguaje $(bc \cup cb)^* a^* b \cup (b^* ca)^* c^+$.

Teorema de Kleene - Parte II

Teorema de Kleene - Parte II

Dado un autómata M, ya sea un AFD, AFN o AFN-λ, se puede encontrar una expresión regular R tal que $L(M) = R$.

- La demostración también es constructiva.
- Se basa en la noción de Grafo Etiquetado Generalizado.

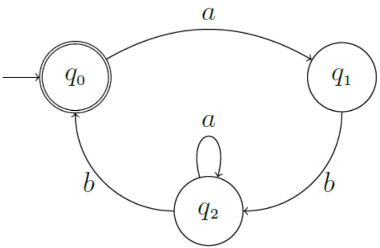
Grafo Etiquetado Generalizado (GEG)

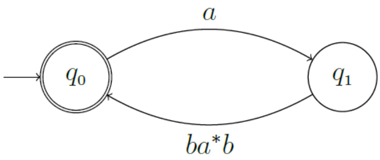
Un **Grafo Etiquetado Generalizado (GEG)** es un grafo como el del autómata excepto que las etiquetas de los arcos entre estados pueden ser expresiones regulares y no simplmente símbolos de un alfabeto.

Teorema de Kleene - Parte II

La parte II del Teorema de Kleene se puede demostrar constructivamente mediante el siguiente procedimiento:

1. Eliminar uno a uno los estados del autómata original M, obteniendo en cada paso un GEG cuyo lenguaje aceptado es $L(M)$.
2. Cuando el grafo se reduce a dos estados (uno de ellos debe ser el estado inicial), el lenguaje aceptado se puede obtener por simple inspección.



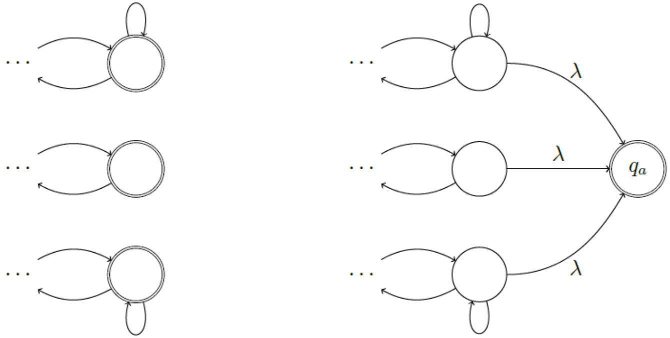


Procedimiento para encontrar una R tal que $L(M) = R$

1. Reemplazar múltiples arcos entre dos estados por uno:

$$q_i \xrightarrow{a_1, a_2, \ldots, a_k} q_j \quad \text{se transforma en} \quad q_i \xrightarrow{a_1 \cup a_2 \cup \cdots \cup a_k} q_j$$

2. Modificar el GEG G para que tenga un único estado de aceptación.



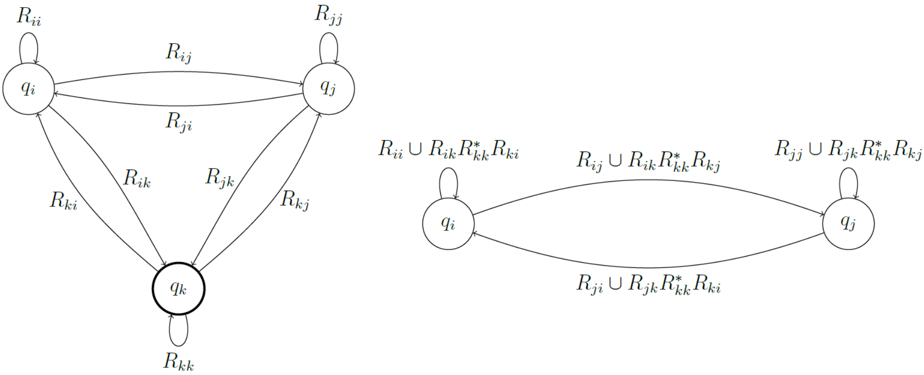
Tres estados de aceptación

Único estado de aceptación q_a

Procedimiento para encontrar una R tal que $L(M) = R$

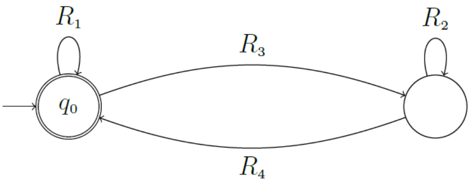
3. Ciclo iterativo por medio del cual se van eliminando uno a uno los estados de G hasta que permanezcan únicamente dos estados (uno de ellos debe ser q_0 y uno de ellos debe ser el estado de aceptación).
   - R_{ij}: etiqueta entre q_i y q_j.
   - $R_{ij} = \emptyset$ si no existe arco entre q_i y q_j.
   - Escoger un estado cualquiera q_k diferente de q_0 y que no sea de aceptación.
   - Eliminar q_k añadiendo transiciones entre los estados restantes de manera que el lenguaje aceptado no se altere.
   - Para q_i y q_j diferentes de q_k, q_k sirve de puente entre q_i y q_j y entre q_j y q_i.
   - También q_k sirve de puente tanto para q_i como para q_j para conectarse consigo mismo.
   - Al eliminar q_k se deben tener en cuenta todas estas trayectorias en los arcos que queden.
   - Realizar esto para todas las parejas de q_i y q_j diferentes a q_k.

Procedimiento para encontrar una R tal que $L(M) = R$

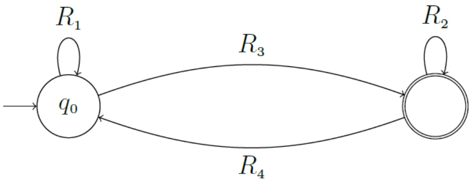


Procedimiento para encontrar una R tal que $L(M) = R$

4. Cuando haya solo dos estados, obtener la expresión regular de acuerdo a los siguientes casos:



$$L(M) = (R_1 \cup R_3 R_2^* R_4)^*.$$



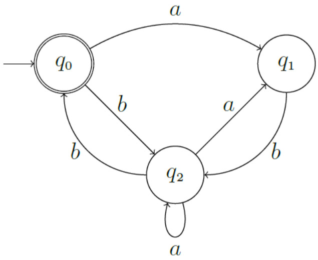
$$L(M) = R_1^* R_3 (R_2 \cup R_4 R_1^* R_3)^*.$$

Procedimiento para encontrar una R tal que $L(M) = R$

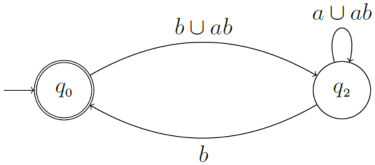
Observaciones

- El procedimiento es flexible. Por ejemplo se puede dejar un estado de aceptación único después de haber eliminado algunos estados.
- No se pueden eliminar estados de aceptación ni el estado inicial.
- No es necesario memorizar las expresiones de los Pasos 3 y 4. Se pueden deducir por simple inspección.

Ejemplo

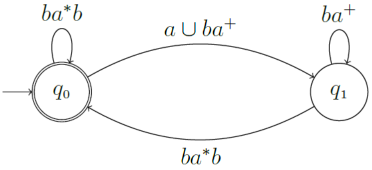


Eliminación del estado q_1:



$L(M) = [(b \cup ab)(a \cup ab)^*b]^*.$

Eliminación del estado q_2:

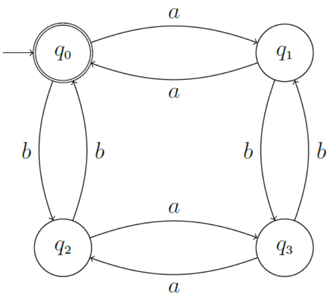


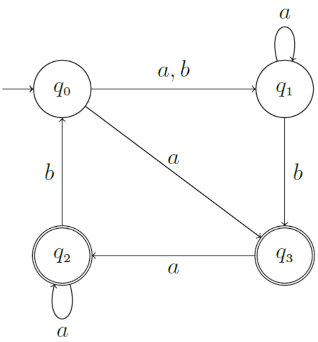
$L(M) = [ba^*b \cup (a \cup ba^+)(ba^+)^*ba^*b]^*.$

Ejercicio

Ejercicio

Encontrar una expresión regular para representar los lenguajes aceptados por los siguientes autómatas.





Propiedades de Clausura de los Lenguajes Regulares

- La regularidad es preservada por ciertas operaciones.
- Los lenguajes regulares son cerrados bajo dichas operaciones.

Teorema

Sean M, M_1 y M_2 autómatas finitos defidos sobre el alfabeto Σ tales que $L(M) = L$, $L(M_1) = L_1$ y $L(M_2) = L_2$. Se pueden construir autómatas finitos que acepten:

1. $L_1 \cup L_2$
2. $L_1 L_2$
3. L^*
4. L^+
5. $\bar{L} = \Sigma^* - L$
6. $L_1 \cap L_2$
7. $L_1 - L_2$
8. $L_1 \triangle L_2$

Propiedades de Clausura de los Lenguajes Regulares

- Por el Teorema de Kleene, los lenguajes aceptados por los autómatas finitos son los regulares.

Teorema

Si L, L_1 y L_2 son lenguajes regulares definidos sobre el alfabeto Σ, también son regulares los siguientes lenguajes:

1. $L_1 \cup L_2$
2. $L_1 L_2$
3. L^*
4. L^+
5. $\bar{L} = \Sigma^* - L$
6. $L_1 \cap L_2$
7. $L_1 - L_2$
8. $L_1 \triangle L_2$

Outline

Repaso de Relaciones

Relación Binaria

- Una **relación binaria** R definida sobre los conjuntos A y B es un subconjunto del producto Cartesiano $A \times B$.
- Si $(a, b) \in R$, donde $a \in A$ y $b \in B$, se denota como $a\,R\,b$.
- Una **relación binaria sobre el conjunto** A es aquella donde R es un subconjunto $A \times A$. Ejemplo: $<$ sobre $\mathbb{Z}$.
- Sea A un conjunto y $a, b, c \in A$. Una relación $R \subseteq A \times A$ es:
  - **Reflexiva:** si $a\,R\,a$.
  - **Simétrica:** $a\,R\,b \Rightarrow b\,R\,a$.
  - **Transitiva:** $(a\,R\,b \wedge b\,R\,c) \Rightarrow a\,R\,c$.

Repaso de Relaciones

Relación de Equivalencia

- Una **relación de equivalencia** es una relación reflexiva, simétrica y transitiva.
- Una relación de equivalencia R definida sobre un conjunto A, particiona a A en clases disyuntas llamadas clases de equivalencia.
- La **clase de equivalencia** a la que pertenece $a \in A$ con respecto a R es $[a]_R = \{b \in A : a\, R\, b\}$.
- El conjunto de todas las clases de equivalencia se denomina **conjunto cociente inducido por la relación** R y se nota como A/R.
- $|A/R|$: índice de la relación R.

Relaciones Importantes en esta Sección

Relación de Estados Equivalentes

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD con todos sus estados accesibles y $p, q \in Q$. Se dice que p y q son **equivalentes**, notado $p \approx q$, si

$$(\forall u \in \Sigma^*)[\hat{\delta}(p, u) \in F \iff \hat{\delta}(q, u) \in F].$$

- Dados $p, q, r \in Q$, la relación $\approx$ cumple con las siguientes propiedades:
  - Reflexividad: $p \approx p$.
  - Simetría: Si $p \approx q$, entonces $q \approx p$.
  - Transitividad: Si $p \approx q$ y $q \approx r$, entonces $p \approx r$.
- Entonces, $\approx$ es una relación de equivalencia sobre Q.
- La **clase de equivalencia** de $p \in Q$, respecto a $\approx$, es:

$$[p]_{\approx} := \{q \in Q : p \approx q\}.$$

Relaciones Importantes en esta Sección

Cadenas cuyo procesamiento termina en el mismo estado

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD con todos sus estados accesibles y $p, q \in Q$.
- Se define la relación R_M sobre Σ^*: $uR_Mv \iff \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$.
- R_M es de equivalencia ya que, dados $u, v, w \in \Sigma^*$, cumple con las siguientes propiedades:
  - Reflexividad: $u\,R_M\,u$.
  - Simetría: Si $u\,R_M\,v$, entonces $v\,R_M\,u$.
  - Transitividad: Si $u\,R_M\,v$ y $v\,R_M\,w$, entonces $u\,R_M\,w$.
- La **clase de equivalencia** de $u \in \Sigma^*$, respecto a R_M, es:

$$[u]_{R_M} := \{v \in \Sigma^* : uR_Mv\}.$$

- $|\Sigma^*/R_M| = |Q|$. Entonces R_M es de índice finito.

Repaso de Particiones

Partición

- Una partición P de un conjunto A es una colección $\{A_1, A_2, \ldots\}$ de subconjuntos de A tales que:
  1. $A_i \neq \emptyset$.
  2. $A_i \cap A_j = \emptyset$ si $i \neq j$.
  3. $\bigcup_i A_i = S$.
- Cada elemento de A está exactamente en una partición.
- Dada una relación de equivalencia R sobre A, A/R constituye una partición de A.
- También, cada partición crea una relación de equivalencia: "está en la misma partición de".

Repaso de Particiones

Refinamiento de Particiones

- Sean $P = \{A_1, A_2, \ldots\}$ y $P' = \{A'_1, A'_2, \ldots\}$ dos particiones de A.
- P es un refinamiento de P' si cada A_i es un subconjunto de algún A'_j.
- Dadas las relaciones de equivalencia R_1 y R_2 sobre A, se dice que R_1 es un refinamiento de R_2 si, dado $a \in A$, $[a]_{R_1} \subseteq [a]_{R_2}$.
- Toda clase R_2 es la unión de clases R_1. Hay menos clases de equivalencia de R_2 que de R_1.
- R_1 es más fina que R_2, i.e. R_1 refina R_2, porque es una clasificación más específica (hay más clases en R_1).

Cadenas L-Indistinguibles

- Sea Σ un alfabeto dado, $L \subseteq \Sigma^*$ y $u, v \in \Sigma^*$.
- u y v son **indistinguibles con respecto a L (L-indistinguibles)**, notado como $u \, I_L \, v$, si

$$(\forall x \in \Sigma^*)[ux \in L \iff vx \in L].$$

- Si u y v no son L-indistinguibles, se dice que son **L-distinguibles**

$$(\exists x \in \Sigma^*)[(ux \in L \wedge vx \notin L) \vee (ux \notin L \wedge vx \in L)].$$

Relación de Indistinguibilidad

- La relación de indistiguibilidad I_L también se llama **relación de Myhill-Nerode**.
- Sean $u, v, w \in \Sigma^*$. I_L es una relación de equivalencia porque es:
  - Reflexiva: $u \, I_L \, u$
  - Simétrica: $u \, I_L \, v \Rightarrow v \, I_L \, u$
  - Transitiva: $u \, I_L \, v \wedge v \, I_L \, w \Rightarrow u \, I_L \, w$
- Para $u \in \Sigma^*$, la **clase de equivalencia de** u es

$$[u]_{I_L} = \{v \in \Sigma^* : u \, I_L \, v\}.$$

Proposición 1

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD con todos sus estados accesibles tal que $L(M) = L$. Sean $u, v \in \Sigma^*$.
- Si $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$, entonces $u\, I_L\, v$, i.e. $u\, R_M\, v \Rightarrow u\, I_L\, v$.
- Es decir, dos cadenas cuyo procesamiento con M termina en el mismo estado son L-indistinguibles.
- *Demostración:* Dado que $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$, se debe probar que $(\forall x \in \Sigma^*)[ux \in L \iff vx \in L]$.

- La Proposición 1 establece que, dada $u \in \Sigma^*$, $[u]_{R_M} \subseteq [u]_{I_L}$.
- Entonces, R_M es un refinamiento de I_L: I_L es la unión de clases R_M.
- Por lo anterior, $|\Sigma^*/I_L| \leq |\Sigma^*/R_M|$.
- R_M es de índice finito porque $|\Sigma^*/R_M| = |Q|$. Por tanto, I_L también.

Teorema de Myhill-Nerode

Teorema

- Sea Σ un alfabeto dado y $L \subseteq \Sigma^*$.
- L es regular si y sólo si I_L es de índice finito, i.e. I_L determina un número finito de clases de equivalencia sobre Σ^*.

Demostración ($\Rightarrow$)

- Si L es regular, existe un AFD $M = (\Sigma, Q, q_0, F, \delta)$ tal que $L(M) = L$ con todos sus estados accesibles.
- Se define la relación R_M sobre Σ^*: $u \, R_M \, v \iff \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$.
- R_M es de equivalencia: reflexiva, simétrica y transitiva.
- $|\Sigma^*/R_M| = |Q|$. Entonces R_M es de índice finito.
- Para todo $u \in \Sigma^*$, $[u]_{R_M} \subseteq [u]_{I_L}$ porque $u \, R_M \, v \Rightarrow u \, I_L \, v$ (Prop. 1).
- $|\Sigma^*/I_L| \leq |\Sigma^*/R_M| = |Q|$, i.e. I_L es de índice finito.

Teorema de Myhill-Nerode

Demostración ($\Leftarrow$)

- Si I_L es de índice finito se puede construir un AFD $M_L = (\Sigma, Q_L, q_i, F_L, \delta_L)$ que acepte a L donde

$$\begin{aligned} Q_L &= \{[u]_{I_L} : u \in \Sigma^*\} \\ q_i &= [\lambda]_{I_L} \\ F_L &= \{[u]_{I_L} : u \in L\} \\ \delta_L([u]_{I_L}, a) &= [ua]_{I_L}, (\forall u \in \Sigma^*)(\forall a \in \Sigma) \end{aligned}$$

- Demostrar que δ_L está bien definida, i.e. $u\, I_L\, v \Rightarrow ua\, I_L\, va (\forall a \in \Sigma)$.
- Demostrar que F_L está bien definida, i.e $u \in L \wedge v \in [u]_{I_L} \Rightarrow v \in L$.
- Demostrar que $\hat{\delta}_L([\lambda]_{I_L}, u) = [u]_{I_L}, (\forall u \in \Sigma^*)$.
- Demostrar que $L(M_L) = L$.

Autómatas Isomorfos

- Sean $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ y $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ dos AFDs.
- M_1 y M_2 son **autómatas isomorfos** si existe una biyección $f : Q_1 \to Q_2$ tal que:
  1. $f(q_1) = q_2$.
  2. $q \in F_1 \iff f(q) \in F_2$.
  3. $\delta_2(f(q), a) = f(\delta_1(q, a))$ para todo $q \in Q_1$ y todo $a \in \Sigma$.

Teorema 1

Teorema

- Sea L un lenguaje regular sobre Σ.
- Sea $M_L = (\Sigma, Q_L, q_i, F_L, \delta_L)$ un AFD que acepta L donde

$$\begin{aligned} Q_L &= \{[u]_{I_L} : u \in \Sigma^*\} \\ q_i &= [\lambda]_{I_L} \\ F_L &= \{[u]_{I_L} : u \in L\} \\ \delta_L([u]_{I_L}, a) &= [ua]_{I_L}, (\forall u \in \Sigma^*)(\forall a \in \Sigma) \end{aligned}$$

- M_L es el AFD con el mínimo número de estados posible para aceptar a L. Además, M_L es único salvo isomorfismo.

Teorema 1

Demostración de que M_L es mínimo

- En la demostración del Teorema de Myhill-Nerode se considera un AFD M *cualquiera* que acepta a L.
- Se define la relación R_M sobre Σ^*: $u \, R_M \, v \iff \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$.
- R_M es de equivalencia: reflexiva, simétrica y transitiva.
- $|\Sigma^*/R_M| = |Q|$. Entonces R_M es de índice finito.
- Para todo $u \in \Sigma^*$, $[u]_{R_M} \subseteq [u]_{I_L}$ porque $u \, R_M \, v \Rightarrow u \, I_L \, v$ (Prop. 1).
- Se encuentra que $|\Sigma^*/I_L| \leq |\Sigma^*/R_M| = |Q|$.
- Esto demuestra que M_L es un AFD con el mínimo número de estados para aceptar L.

Teorema 1

Demostración de que M_L es único

- Considerar $M = (\Sigma, Q, q_0, F, \delta)$ tal que $L(M) = L$ y $|Q| = |Q_L|$.
- Se define la función $f : Q \to Q_L$ así: $f(q) = [u]_{I_L}$ donde $\hat{\delta}(q_0, u) = q$ para alguna $u \in \Sigma^*$.
- Para todo $q \in Q$ existe una cadena $u \in \Sigma^*$ tal que $\hat{\delta}(q_0, u) = q$ porque todos los estados son accesibles.
- f no depende del representante: $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v) \Rightarrow u\, I_L\, v$ (Prop 1), por lo que $[u]_{I_L} = [v]_{I_L}$.
- f es sobreyectiva y como $|Q| = |Q_L|$, se concluye que f es biyectiva.
- Probar que f cumple las propiedades 1 y 2 de autómatas isomorfos.
- f cumple la propiedad 3: $\delta_L(f(q), a) = f(\delta(q, a))$ con $q \in Q$ y $a \in \Sigma$. Se prueba tomando $u \in \Sigma^*$ tal que $\hat{\delta}(q_0, u) = q$. Entonces, $\hat{\delta}(q_0, ua) = \delta(\hat{\delta}(q_0, u), a) = \delta(q, a)$. Por esto, $f(\delta(q, a)) = [ua]_{I_L}$.

Proposición 2

Proposición

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD y $u, v \in \Sigma^*$.
- Si $u\ I_L\ v$, entonces $\hat{\delta}(q_0, u) \approx \hat{\delta}(q_0, v)$.

Demostración

- Sean $p = \hat{\delta}(q_0, u)$ y $q = \hat{\delta}(q_0, v)$.
- Demostrar que $(\forall w \in \Sigma^*)[\hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F]$.
- $\hat{\delta}(p, w) = \hat{\delta}(\hat{\delta}(q_0, u), w) = \hat{\delta}(q_0, uw)$.
- $\hat{\delta}(q, w) = \hat{\delta}(\hat{\delta}(q_0, v), w) = \hat{\delta}(q_0, vw)$.
- Entonces,

$$\begin{aligned}\hat{\delta}(p, w) \in F &\iff \hat{\delta}(q_0, uw) \in F \iff uw \in L(M) = L \\ &\iff vw \in L \iff \hat{\delta}(q_0, vw) \in F \iff \hat{\delta}(q, w) \in F.\end{aligned}$$

Teorema 2

Teorema

- Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD tal que $L(M) = L$, con todos sus estados accesibles.
- El autómata cociente M' y el autómata M_L son isomorfos.
- Por tanto, M' es un AFD con el mínimo número de estados posibles para aceptar L y es único salvo isomorfismo.

Teorema 2

Recordar

El autómata cociente de $M = (\Sigma, Q, q_0, F, \delta)$ es $M' = (\Sigma, Q', q_0', F', \delta')$, donde:

$$\begin{aligned} Q' &= \{[p]_{\approx} : p \in Q\} \\ q_0' &= [q_0]_{\approx} \\ F' &= \{[p]_{\approx} : p \in F\} \\ \delta'([p]_{\approx}, a) &= [\delta(p, a)]_{\approx} \\ &\quad (\forall a \in \Sigma)(\forall p \in Q). \end{aligned}$$

Los componentes de $M_L = (\Sigma, Q_L, q_i, F_L, \delta_L)$ son:

$$\begin{aligned} Q_L &= \{[u]_{I_L} : u \in \Sigma^*\} \\ q_i &= [\lambda]_{I_L} \\ F_L &= \{[u]_{I_L} : u \in L\} \\ \delta_L([u]_{I_L}, a) &= [ua]_{I_L}, \\ &\quad (\forall u \in \Sigma^*)(\forall a \in \Sigma). \end{aligned}$$

Teorema 2

Demostración

- Como $L(M') = L$, basta demostrar que $|Q'| = |Q_L|$ y usar el Trm 1.
- Como M_L es mínimo, $|Q_L| \leq |Q'|$, demostrar que $|Q'| \leq |Q_L|$.
- Definir $g : Q_L \to Q'$ así: $g([u]_{I_L}) = [q]_{\approx}$, donde $q = \hat{\delta}(q_0, u)$, $u \in \Sigma^*$.
- g está bien definida: si $[u]_{I_L} = [v]_{I_L}$, entonces $u \, I_L \, v$ y por tanto $\hat{\delta}(q_0, u) \approx \hat{\delta}(q_0, v)$ (Prop 2).
- Como los estados de M' son accesibles, g es sobreyectiva. Entonces $|Q'| \leq |Q_M|$.

Outline

Pruebas de Regularidad

Para determinar si un lenguaje L es regular o no, se puede emplear alguna de las siguientes pruebas:

1. Argumento por contradicción.
2. Criterio de Regularidad.
3. Lema del bombeo.

Las dos primeras se basan en una de las versiones del Principio de Palomar.

Principio de Palomar

Si infinitos objetos de distribuyen en un número finito de compartimientos, entonces hay un compartimiento que contiene por lo menos dos objetos.

Argumento por Contradicción

- Suponer que L es regular. Por el Teorema de Kleene, existe un AFD $M = (\Sigma, Q, q_0, F, \delta)$ tal que $L(M) = L$.
- Considerar infinitas cadenas que sean prefijos de cadenas de L.
- Como hay infinitas de estas cadenas y solo un número finito de estados, debe haber dos de estas cadenas u y v, tales que $u \neq v$, cuyo procesamiento completo termine en el mismo estado: $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$ (Principio de Palomar).
- Como el autómata es determinista, para toda $x \in \Sigma^*$, se tiene que $\hat{\delta}(q_0, ux) = \hat{\delta}(\hat{\delta}(q_0, u), x) = \hat{\delta}(\hat{\delta}(q_0, v), x) = \hat{\delta}(q_0, vx)$.
- Lo anterior implica que $ux \in L \iff vx \in L$.
- Encontrar una cadena x tal que solo una entre ux y vx sea aceptada y la otra rechazada. Esta contradicción demuestra que L no es regular.

Argumento por Contradicción

Ejercicio

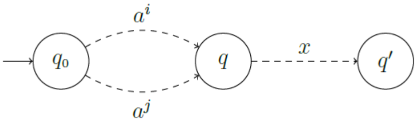
Demostrar por medio de un argumento por contradicción que el lenguaje $L = \{a^n b^n : n \geq 0\} = \{\lambda, ab, a^2b^2, a^3b^3, \ldots\}$ sobre $\Sigma = \{a, b\}$ no es regular.

Solución

- Por el Teorema de Kleene, un lenguaje L es regular si y solo si puede ser aceptado por un AFD $M = (\Sigma, Q, q_0, F, \delta)$.
- Para aceptar L, M requiere almacenar la información sobre el número de *a*es y compararla con el número de *b*es.
- Con un número finito de estados no se puede almacenar un valor $n \geq 0$.

Argumento por Contradicción

- Si L es regular, es aceptado por un AFD $M = (\Sigma, Q, q_0, F, \delta)$.
- Considerar a, a^2, a^3, ... Como hay infinitas de estas cadenas y solo un número finito de estados, debe haber dos de estas cadenas a^i y a^j, tales que $i \neq j$, cuyo procesamiento completo termine en el mismo estado: $\hat{\delta}(q_0, a^i) = \hat{\delta}(q_0, a^j)$ (Principio de Palomar).
- Como M es determinista, para toda $x \in \Sigma^*$, se tiene que $\hat{\delta}(q_0, a^i x) = \hat{\delta}(\hat{\delta}(q_0, a^i), x) = \hat{\delta}(\hat{\delta}(q_0, a^j), x) = \hat{\delta}(q_0, a^j x)$.
- Lo anterior implica que $a^i x \in L \iff a^j x \in L$.
- Para $x = b^i$, se tiene que $a^i b^i \in L$ y $a^j b^i \notin L$. Esta contradicción demuestra que L no es regular.



Argumento por Contradicción

Ejercicio

Demostrar por medio de un argumento por contradicción que el lenguaje de los palíndromes sobre $\Sigma = \{a, b\}$ no es regular.

Criterio de Regularidad

Criterio de Regularidad

- Sea Σ un alfabeto y L un lenguaje sobre Σ.
- Si en Σ^* hay infinitas cadenas L-distinguibles dos a dos, entonces L no es regular.
- Con este criterio, no es necesario utilizar explícitamente los autómatas.

Demostración 1 (por el Teorema de Myhill-Nerode)

- Si hay infinitas cadenas L-distinguibles entre sí, entonces la relación de L-indistinguibilidad I_L determina infinitas clases de equivalencia sobre Σ^*, i.e. I_L tiene índice infinito.
- Por el Teorema de Myhill-Nerode, L no es regular.

Criterio de Regularidad

Demostración 2 (por contradicción)

- Suponer que L es regular. Por el Teorema de Kleene, existe un AFD $M = (\Sigma, Q, q_0, F, \delta)$ tal que $L(M) = L$.
- Por hipótesis, existen infinitas cadenas L-distinguibles dos a dos: u_1, u_2, u_3, ...
- Como solo hay un número finito de estados, debe haber dos de estas cadenas u_i y u_j, tales que $i \neq j$, cuyo procesamiento completo termine en el mismo estado: $\hat{\delta}(q_0, u_i) = \hat{\delta}(q_0, u_j)$ (Principio de Palomar).
- Como el autómata es determinista, para toda $x \in \Sigma^*$, se tiene que $\hat{\delta}(q_0, u_i x) = \hat{\delta}(\hat{\delta}(q_0, u_i), x) = \hat{\delta}(\hat{\delta}(q_0, u_j), x) = \hat{\delta}(q_0, u_j x)$.
- Lo anterior implica que $u_i x \in L \iff u_j x \in L$ para toda $x \in \Sigma^*$.
- Esto contradice que u_i y u_j son L-distinguibles. Esta contradicción demuestra que L no es regular.

Criterio de Regularidad

Ejercicio

Demostrar por medio del criterio de regularidad que el lenguaje $L = \{a^n b^n : n \geq 0\}$ sobre $\Sigma = \{a, b\}$ no es regular.

Solución

- Comprobar que las cadenas a, a^2, a^3, $\ldots$ son infinitas cadenas L-distinguibles dos a dos.
- Sean $i, j \geq 1$, con $i \neq j$. Mostrar que a^i y a^j son L-distingibles.
- Para $x = b^i$, se tiene que $a^i b^i \in L$ y $a^j b^i \notin L$, i.e. a^i y a^j son L-distinguibles.
- Por ende, L no es regular.

Criterio de Regularidad

Ejercicio

Demostrar por medio del criterio de regularidad que el lenguaje de los palíndromes sobre $\Sigma = \{a, b\}$ no es regular.

Criterio de Regularidad

Ejercicio

Demostrar por medio del criterio de regularidad que el lenguaje $L = \{a^{n^2}\}$ sobre $\Sigma = \{a\}$ no es regular.

Problemas solubles con autómatas finitos

- La existencia de lenguajes no regulares implica que hay ciertos problemas que no pueden ser resueltos por medio de autómatas finitos.
- Un autómata es un mecanismo que produce dos salidas para cada entrada $u \in \Sigma^*$:
  - SÍ, cuando u es aceptada.
  - NO, cuando u es rechazada.
- Los problemas que podría resolver un autómata finito son los **problemas de decisión**: problemas para los cuales hay dos únicas salidas: SÍ o NO.

Problemas solubles con autómatas finitos

Dado el alfabeto Σ, considérese una propiedad $\mathcal{P}$ referente a las cadenas de Σ^*. El problema de decisión para $\mathcal{P}$ es el siguiente:

Dada $u \in \Sigma^*$, ¿Satisface u la propiedad $\mathcal{P}$?

- El lenguaje de las cadenas que satisfacen la propiedad $\mathcal{P}$ es: $L_{\mathcal{P}} = \{u \in \Sigma^* : u \text{ satisface } \mathcal{P}\} = \{u \in \Sigma^* : \mathcal{P}(u)\}$.
- El problema puede ser resuelto usando autómatas finitos si existe un AFD M tal que $L(M) = L_{\mathcal{P}}$.
- Por el Teorema de Kleene, se puede resolver con autómatas finitos si y solo si $L_{\mathcal{P}}$ es regular.
- En tal caso, si u es aceptada quiene decir que u cumple con $\mathcal{P}$; si es rechazada es porque u no cumple con $\mathcal{P}$.

Problemas solubles con autómatas finitos

Ejercicio

Sea $\Sigma = \{0, 1\}$, ¿El problema siguiente puede ser resulto con autómatas finitos?

- Dada $u \in \Sigma^*$, ¿tiene u un número par de ceros?

Problemas solubles con autómatas finitos

Ejercicio

Sea $\Sigma = \{0, 1\}$, ¿El problema siguiente puede ser resulto con autómatas finitos?

- Dada $u \in \Sigma^*$, ¿es u un palíndrome?

Problemas solubles con autómatas finitos

Ejercicio

Sea $\Sigma = \{0, 1\}$, ¿El problema siguiente puede ser resulto con autómatas finitos?

- Dada $u \in \Sigma^*$, ¿tiene u el mismo número de ceros que de unos?

Bibliografía

1. Rodrigo De Castro Korgi. **Notas de Clase de Introducción a la Teoría de la Computación**. 2023. Contenidos e imágenes de estas notas fueron incluidas en esta presentación.
2. Harry R. Lewis, Christos H. Papadimitriou. **Elements of the Theory of Computation**. Prentice Hall. 1998.
3. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. **Introduction to Automata Theory, Languages and Computation, Third Edition**. Pearson. 2006.