Autómatas Finitos y Expresiones Regulares

Pablo Verdes

LCC

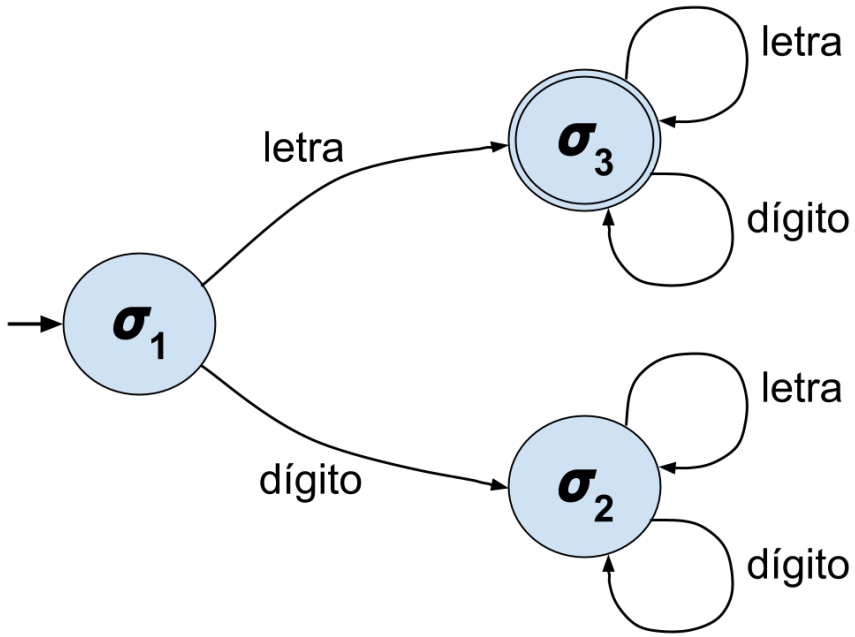
21 de mayo de 2018

Autómatas Finitos: introducción

- En esta unidad definimos y estudiamos la clase de máquinas teóricas conocida como **autómatas finitos**.
- Veremos que se pueden identificar con la clase de los lenguajes regulares.
- Resultan fundamentales para la mayoría de las aplicaciones que requieren reconocimiento de patrones.
- Ejemplo: construcción de compiladores. El analizador léxico debe reconocer cuáles son las cadenas del programa fuente que representan objetos individuales tales como variables, constantes numéricas y palabras reservadas.
- En un lenguaje de programación típico, los nombres de variables comienzan con una letra, seguida por una combinación arbitraria (finita) de letras y dígitos. Por ej., son nombres de variables:
  - aceptables: X25, PepeRosas, x2y3z
  - no aceptables: 25, Pepe-Rosas, x.h

Autómatas Finitos: introducción

- Para reconocer las ocurrencias de nombres de variables podríamos utilizar un diagrama de transiciones como el de la figura:



- Diremos que una cadena es aceptada si sus símbolos corresponden a una secuencia de arcos que conducen del círculo inicial a uno doble.
- Un diagrama de transiciones puede ser usado como herramienta de diseño para producir rutinas de análisis léxico.

Autómatas de estado finito (AEF)

Definición: Un autómata de estado finito (determinista) es una tupla

$$A = (\Sigma, S, f, Ac, \sigma)$$

donde:

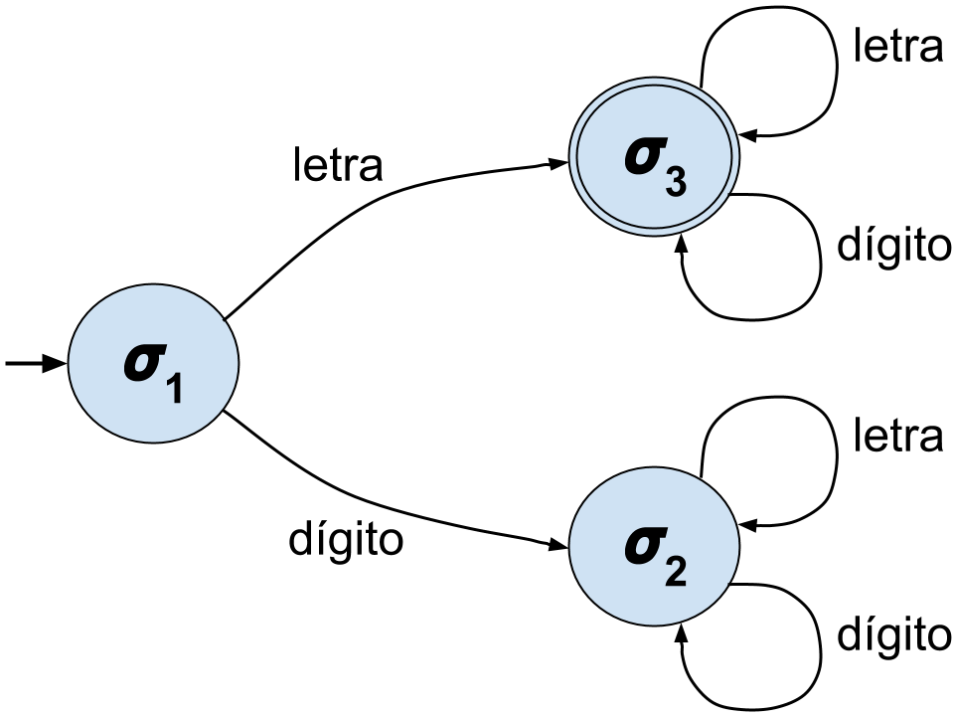
- Σ es un conjunto finito de **símbolos de entrada**
- S es un conjunto finito de **estados**
- $f : S \times \Sigma \to S$ es una **función de transición**
- $Ac \subseteq S$ es un conjunto de **estados de aceptación**
- $\sigma \in S$ es el **estado inicial**

Observaciones:

1. La interpretación de la función de transición es que $f(p, x) = q$ sii la máquina pasa de un estado p a un estado q al leer el símbolo x.
2. Cada estado del diagrama de transiciones de un AEF debe tener un y sólo un arco saliente para cada símbolo del alfabeto.

Autómatas de estado finito

- Para el ejemplo:

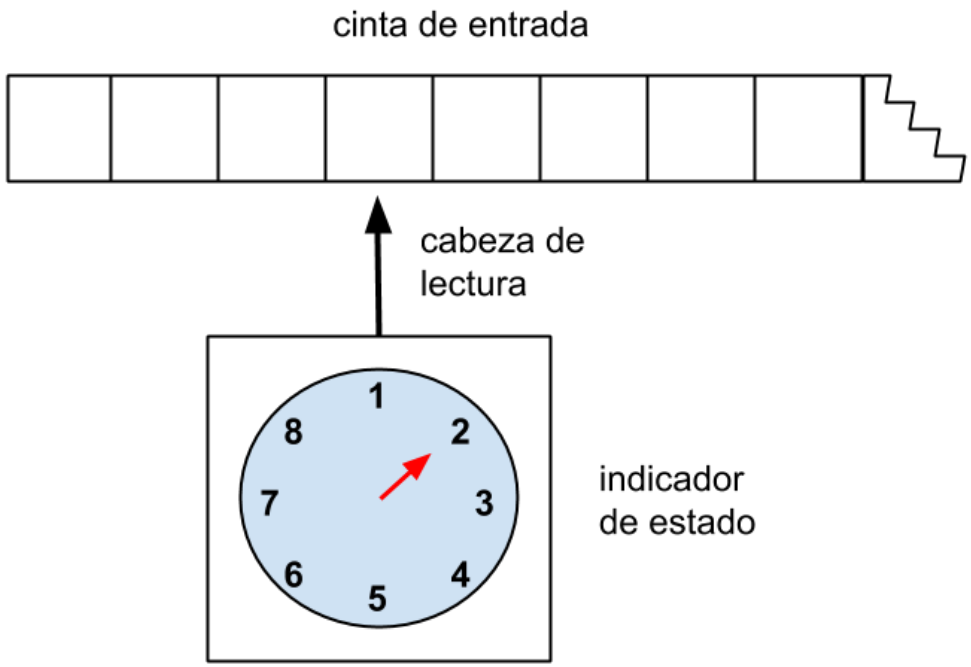


la función de transición está definida de la siguiente forma:

$$f(\sigma_1, \text{dígito}) = \sigma_2 \qquad f(\sigma_2, \text{dígito}) = \sigma_2 \qquad f(\sigma_3, \text{dígito}) = \sigma_3$$
$$f(\sigma_1, \text{letra}) = \sigma_3 \qquad f(\sigma_2, \text{letra}) = \sigma_2 \qquad f(\sigma_3, \text{letra}) = \sigma_3$$

Autómatas de estado finito

- Se suele representar a un AEF de la siguiente manera:



donde el mecanismo interno del indicador de estado está gobernado por la función de transición.

Autómatas de estado finito

- **Definición:**

  Sean $A = (\Sigma, S, f, Ac, \sigma)$ un AEF y $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ una palabra sobre Σ. Si existen $\sigma_0, \sigma_1, \ldots, \sigma_n \in S$ tales que

  $$\sigma_0 = \sigma$$
  $$\sigma_i = f(\sigma_{i-1}, \alpha_i) \quad \text{para} \quad i = 1, \ldots, n$$

  y se cumple que $\sigma_n \in Ac$, entonces diremos que la palabra α es aceptada por el autómata A.

  λ es aceptada $\Leftrightarrow \sigma \in Ac$
  (la palabra vacía es aceptada sii el estado inicial es de aceptación)

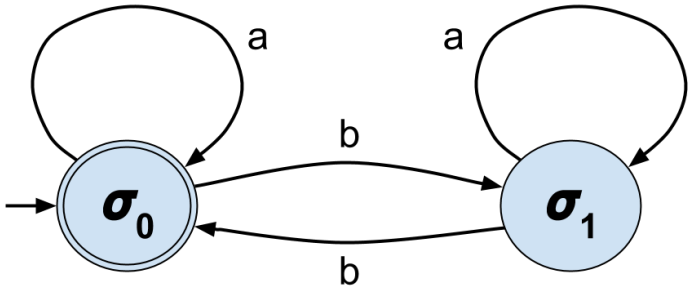
- **Definición:** Lenguaje aceptado por un autómata A:

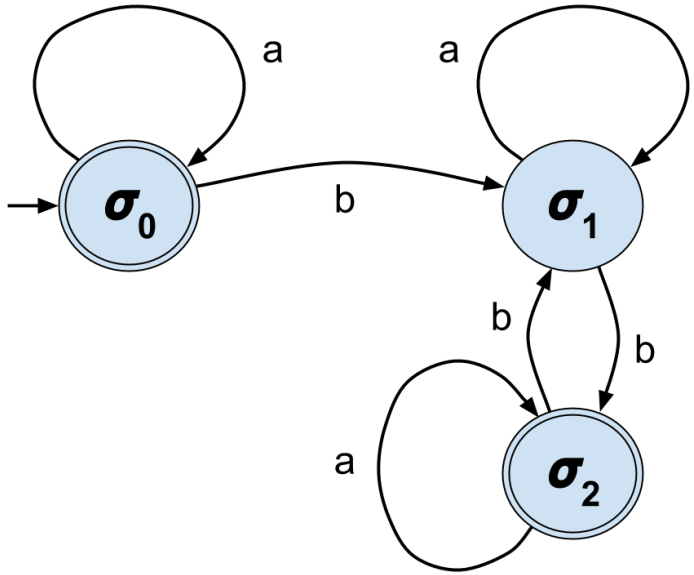
  $$\mathcal{AC}(A) = \{\alpha \in \Sigma^* \mid \alpha \text{ es aceptado por } A\}$$

Autómatas de estado finito

- Las definiciones anteriores permiten definir un lenguaje a partir de un autómata.
- Sin embargo, es más frecuente el proceso inverso: dado un lenguaje, construir un autómata que lo reconozca. Ejemplo:

$$L = \{\alpha \in \{a, b\}^* \mid N_b(\alpha) = 2k,\ k \in \mathbb{N}_0\}$$





Autómatas de estado finito

f(\sigma_{i-1}, \alpha_i) = \sigm

- **Definición:**

  Sean A_1, A_2 AEF. Diremos que A_1 es equivalente a A_2 ($A_1 \equiv A_2$) sii

  $$\mathcal{AC}(A_1) = \mathcal{AC}(A_2),$$

  es decir, sii aceptan el mismo lenguaje.

- **Definición:**

  Clausura de la función de transición o función de transición extendida

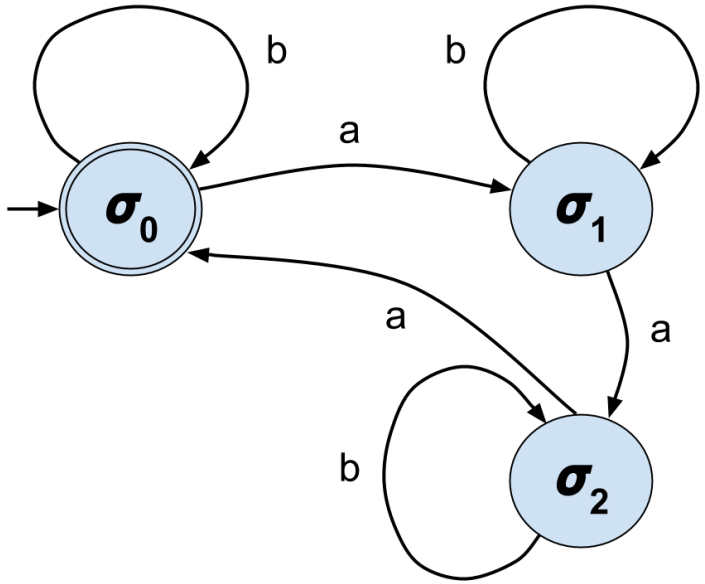
  - Describe lo que ocurre cuando un AEF parte de un estado arbitrario s y procesa una palabra w.
  - Dado $A = (\Sigma, S, f, Ac, \sigma)$ AEF, se define $F : S \times \Sigma^* \to S$ por inducción sobre la longitud de la cadena:

    $F(s, \lambda) = s$

    $F(s, wa) = f(F(s, w), a)$ donde w es una palabra y a un caracter

Autómatas de estado finito

- **Otro ejemplo:** $L = \{\alpha \in \{a, b\}^* \mid N_a(\alpha) = 3k, k \in \mathbb{N}_0\}$



- Comparemos con la gramática regular $G = (N, T, P, \sigma)$ donde $N = \{\sigma_0, \sigma_1, \sigma_2\}$, $T = \{a, b\}, \sigma = \sigma_0$, y P está dado por:

$$\sigma_0 \to a\sigma_1 \mid b\sigma_0$$
$$\sigma_1 \to a\sigma_2 \mid b\sigma_1$$
$$\sigma_2 \to a\sigma_0 \mid b\sigma_2$$
$$\sigma_0 \to \lambda$$

Autómatas de estado finito

- A continuación veremos que dado un lenguaje aceptado por un AEF, siempre podremos escribir una gramática regular que lo genere.

- **Teorema:** *Todo lenguaje aceptado por un AEF es regular:*

$$\{L \in \mathcal{L} \mid L = \mathcal{AC}(A) \text{ para algún } A \text{ AEF}\} \subseteq \mathcal{L}_3$$

D/ Sea $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF. Definimos $G = (N, T, P, \sigma)$ donde

$$N = S, \qquad T = \Sigma, \qquad \sigma = \sigma_0$$

y reglas de producción

$$U \to xV \quad \text{sii} \quad f(U, x) = V$$
$$U \to \lambda \quad \text{sii} \quad U \in Ac$$

Debemos probar que $L(G) = \mathcal{AC}(A)$, es decir, que

$$\sigma \Rightarrow^* \alpha_1\alpha_2 \ldots \alpha_n \qquad \Leftrightarrow \qquad F(\sigma_0, \alpha_1\alpha_2 \ldots \alpha_n) \in Ac$$

Queda como ejercicio completar la demostración. □

Autómatas de estado finito

- Acabamos de ver que todo lenguaje aceptado por un AEF es regular.
- Por lo tanto, los AEF no sirven para reconocer lenguajes independientes de contexto.
- Ahora veremos un ejemplo de un lenguaje para el cual no existe un AEF que lo reconozca.
- **Teorema:** No existe un AEF A tal que

$$\mathcal{AC}(A) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

- Para demostrar este teorema necesitamos probar primero un resultado conocido como *Lema del Bombeo* para AEF, también llamado *Pumping Lemma*.

Autómatas de estado finito

Lema del Bombeo para AEF:

Sean $A = (\Sigma, S, f, Ac, \sigma_0)$ un AEF, $L = \mathcal{AC}(A)$, $x, y \in \Sigma$.

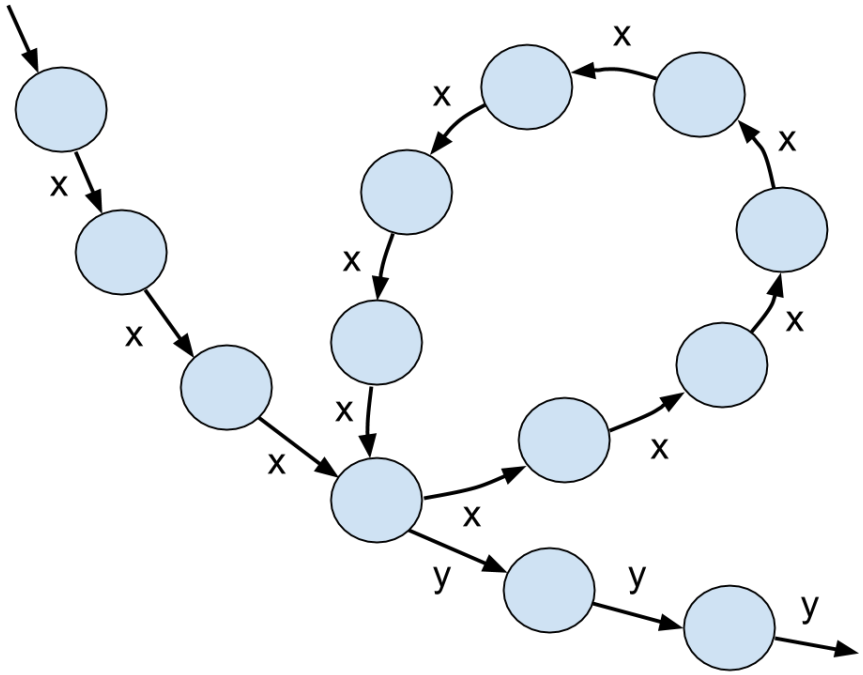
Si L contiene infinitas cadenas de la forma $x^n y^n$, entonces también contiene infinitas cadenas de la forma $x^m y^n$ con $m \neq n$.

D/

- El número de estados de A, $|S|$, es finito. Sea $n > |S|$ tal que $x^n y^n \in L$. (¿Existe tal n?)
- Observemos que durante el proceso de aceptación de la palabra $x^n y^n$ (en particular, de la subcadena x^n) el autómata A deberá pasar sucesivamente por n estados conectados por el caracter x.
- Los estados visitados en dicha secuencia no tienen por qué ser, en general, necesariamente distintos. Sin embargo, como el número de transiciones (n) es mayor que el de estados disponibles ($|S|$), al menos un estado debe aparecer repetido.

Lema del Bombeo para AEF

- Se debe dar, entonces, la situación de la figura:



- De aquí concluimos que A acepta también la palabra $x^{n+m}y^n$, donde m denota a la longitud del bucle.
- Más generalmente, A aceptará las palabras de la forma $x^{n+km}y^n$, con $k \in \mathbb{N}_0$. $\square$

Autómatas de estado finito

Ahora estamos en condiciones de demostrar el siguiente:

Teorema: No existe un AEF A tal que

$$\mathcal{AC}(A) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

D/

- Por el absurdo, sup. que $\exists A$ AEF $\mid \mathcal{AC}(A) = \{a^n b^n \mid n \in \mathbb{N}_0\}$.
- Dado que $\mathcal{AC}(A)$ contiene infinitas cadenas de la forma $a^n b^n$, por el Lema del Bombeo concluimos que $\mathcal{AC}(A)$ también contiene cadenas de la forma $a^m b^n$ con $m \neq n$.
- Contradicción. Luego $\nexists A$ AEF $\mid \mathcal{AC}(A) = \{a^n b^n \mid n \in \mathbb{N}_0\}$. $\square$

Autómatas de estado finito

- Una consecuencia del teorema anterior es que los AEF no sirven como analizadores léxicos de expresiones aritméticas que contienen paréntesis. Veamos por qué.
- Si un AEF aceptara tales expresiones, debería aceptar cadenas de la forma $(^n)^n$ para n arbitrariamente grande.
- Entonces, por el teorema anterior, el AEF aceptará también expresiones de la forma $(^m)^n$ con $m \neq n$.
- Esto significa que el AEF aceptaría expresiones aritméticas tanto correctas como incorrectas.
- Vemos que los AEF no son muy poderosos. Para ampliar su expresividad se propuso introducirles **no determinismo**, como veremos a continuación.

Autómatas de estado finito no deterministas (AEFND)

Definición: Un AEF no determinista es una tupla

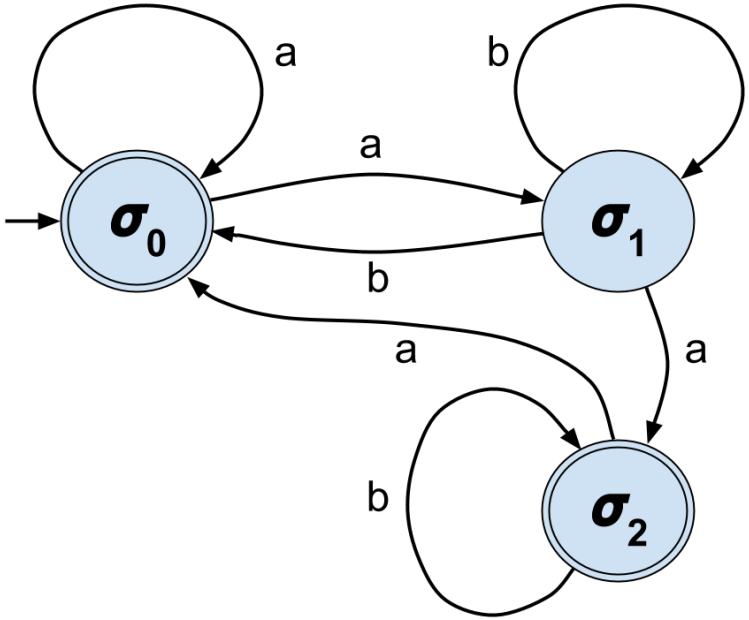
$$A = (\Sigma, S, R, Ac, \sigma)$$

donde:

- Σ es un conjunto finito de **símbolos de entrada**
- S es un conjunto finito de **estados**
- $R \subseteq S \times \Sigma \times S$ es una **relación de transición**
- $Ac \subseteq S$ es un conjunto de **estados de aceptación**
- $\sigma \in S$ es el **estado inicial**

AEF no deterministas

Ejemplo:



$$A = (\Sigma, S, R, Ac, \sigma), \text{ donde:}$$

$$\Sigma = \{a, b\}, \qquad S = \{\sigma_0, \sigma_1, \sigma_2\}, \qquad AC = \{\sigma_0, \sigma_2\}, \qquad \sigma = \sigma_0$$

$$\begin{aligned} R = \{&(\sigma_0, a, \sigma_0), (\sigma_0, a, \sigma_1), \\ &(\sigma_1, a, \sigma_2), (\sigma_1, b, \sigma_0), (\sigma_1, b, \sigma_1), \\ &(\sigma_2, a, \sigma_0), (\sigma_2, b, \sigma_2)\} \end{aligned}$$

AEF no deterministas

- Alternativamente, podríamos pensar a R como una función

$$
\begin{aligned}
g : S \times \Sigma &\to \mathcal{P}(S) \\
(\sigma_0, a) &\mapsto \{\sigma_0, \sigma_1\} \\
(\sigma_0, b) &\mapsto \emptyset \\
(\sigma_1, a) &\mapsto \{\sigma_2\} \\
(\sigma_1, b) &\mapsto \{\sigma_0, \sigma_1\} \\
(\sigma_2, a) &\mapsto \{\sigma_0\} \\
(\sigma_2, b) &\mapsto \{\sigma_2\},
\end{aligned}
$$

  es decir,

$$g(\sigma, x) = \{\sigma' \in S \mid (\sigma, x, \sigma') \in R\}$$

- Si existe más de un camino posible, ¿cómo determinamos si una palabra es aceptada por un AEFND?

AEF no deterministas

Definición: Lenguaje aceptado por un AEFND

Sea $A = (\Sigma, S, R, Ac, \sigma)$ un AEFND y $\alpha \in \Sigma^*$.

- Si $\alpha = \lambda$: $\alpha \in \mathcal{AC}(A) \Leftrightarrow \sigma \in Ac$
- Si $\alpha = \alpha_1\alpha_2 \ldots \alpha_n \neq \lambda$:

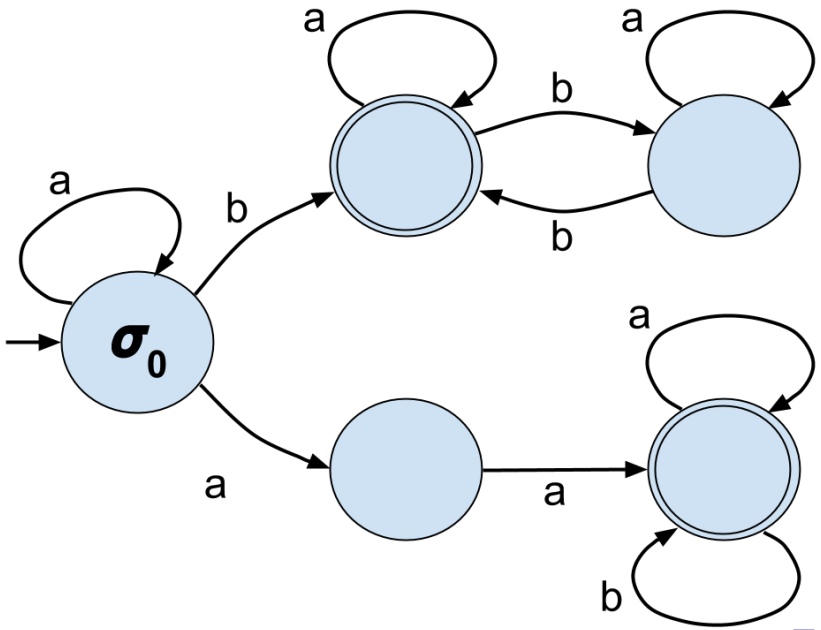
  $\alpha \in \mathcal{AC}(A) \Leftrightarrow$ existen $\sigma_0, \sigma_1, \ldots, \sigma_n$ tales que:

  1. $\sigma_o = \sigma$
  2. $(\sigma_{i-1}, \alpha_i, \sigma_i) \in R \qquad \forall i = 1, \ldots, n$
  3. $\sigma_n \in Ac$

A toda secuencia $(\sigma_0, \sigma_1, \ldots, \sigma_n)$ que cumple los items (1) y (2), pero no necesariamente el (3), se la llama secuencia que representa a α.

AEF no deterministas

- En otras palabras, en un AEFND una cadena α será:
  - **aceptada** ($\alpha \in \mathcal{AC}(A)$) si existe un camino que la represente que termine en Ac
  - **no aceptada** ($\alpha \notin \mathcal{AC}(A)$) si:
    - ★ no existe camino que la represente, o bien
    - ★ todo camino que la representa termina en $s \notin Ac$
- **Ejemplo:** $L = \{\alpha \in \{a, b\}^* \mid N_b(\alpha)$ es impar o α comienza con $aa\}$



Relación entre AEF deterministas y no deterministas

- Dado que toda función es relación, todo AEF es AEFND.
- Por lo tanto

$$\{L \mid \exists A \text{ AEF} : \mathcal{AC}(A) = L\} \quad \subseteq \quad \{L \mid \exists A \text{ AEFND} : \mathcal{AC}(A) = L\}$$

- Más precisamente, sea $A = (\Sigma, S, f, Ac, \sigma)$ un AEF.

  Definamos el AEFND $A' = (\Sigma, S, R, Ac, \sigma)$ con R dado por

$$(P, x, Q) \in R \quad \Leftrightarrow \quad f(P, x) = Q$$

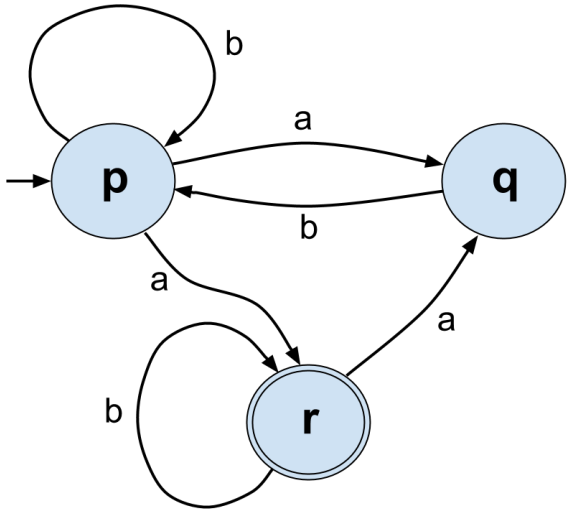
  Entonces A' es AEFND y $\mathcal{AC}(A') = \mathcal{AC}(A)$.

- Recordemos que el no determinismo fue introducido para aumentar la expresividad de los AEF. Cabe entonces hacerse la siguiente pregunta:

  ¿existe L reconocido por un AEFND pero no reconocido por AEF?

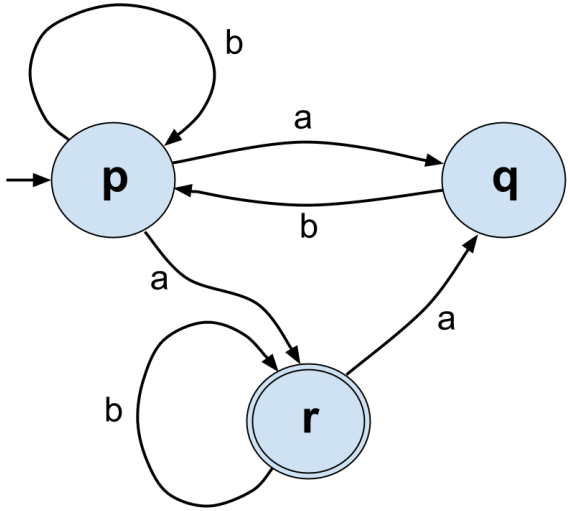
Relación entre AEF deterministas y no deterministas

- Para responder dicha pregunta, consideremos primero esta otra: ¿Cómo verificamos si el AEFND de la figura efectivamente acepta la palabra $w = abbaabb$?



- Una alternativa es examinar todas las secuencias que representan a la palabra w y ver si alguna termina en el estado de aceptación r.
- Sin embargo, el número de caminos puede resultar muy grande y crecer exponencialmente con la longitud de la palabra w.
- Una opción mejor es ir calculando el conjunto de los estados posibles del autómata a medida que se lee, por única vez, la palabra w.

Relación entre AEF deterministas y no deterministas

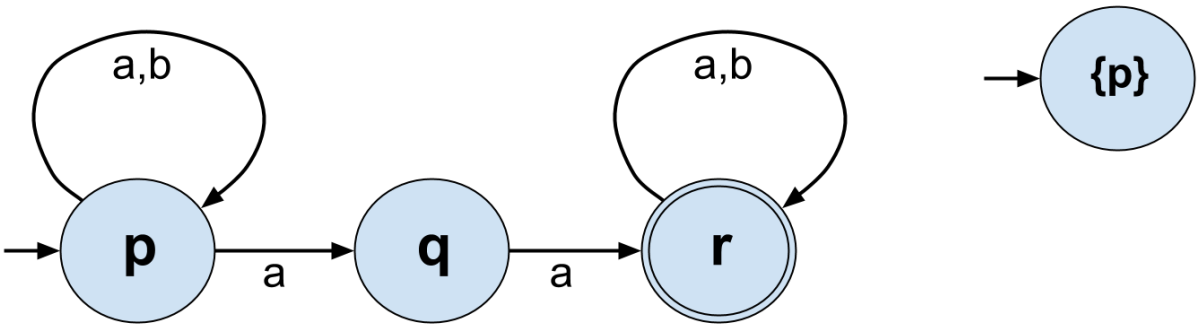


- Así, para la palabra $w = abbaabb$ tendremos, por ejemplo:

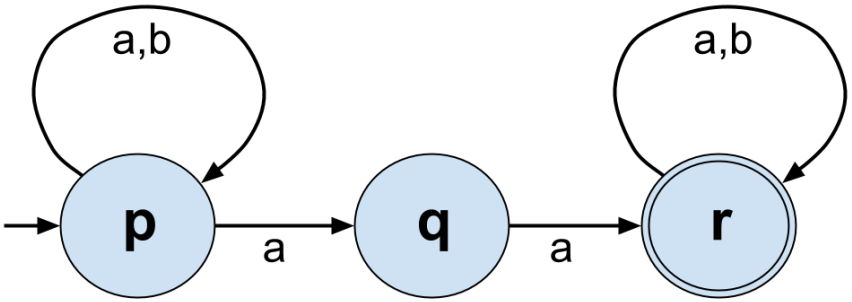
$$\{p\} \xrightarrow{a} \{q,r\} \xrightarrow{b} \{p,r\} \xrightarrow{b} \{p,r\} \xrightarrow{a} \{q,r\} \xrightarrow{a} \{q\} \xrightarrow{b} \{p\} \xrightarrow{b} \{p\}$$

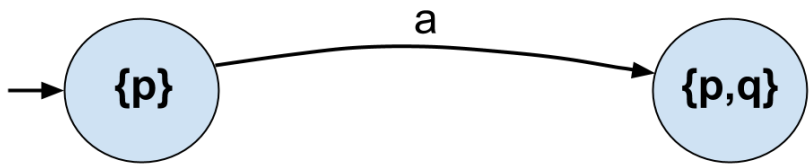
- Vemos que esta manera de procesar w es de tipo determinista, sólo que los estados son ahora elementos de $\mathcal{P}(S)$.
- Esta observación es crucial y constituye la llave para demostrar que para todo AEFND existe un AEF equivalente. Esto probará que los AEFND sólo reconocen lenguajes regulares.

Construcción de un AEF a partir de un AEFND: ejemplo

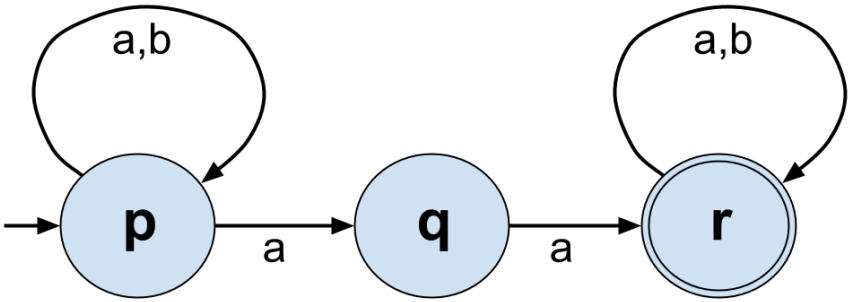


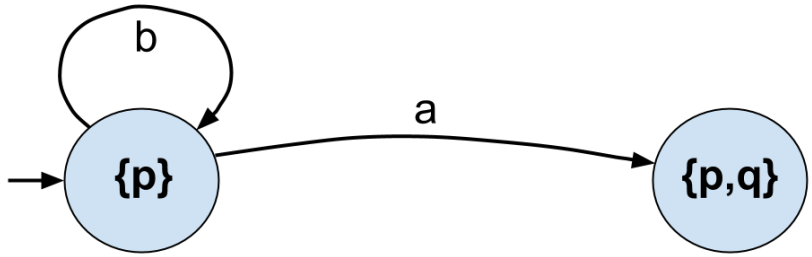
Construcción de un AEF a partir de un AEFND: ejemplo



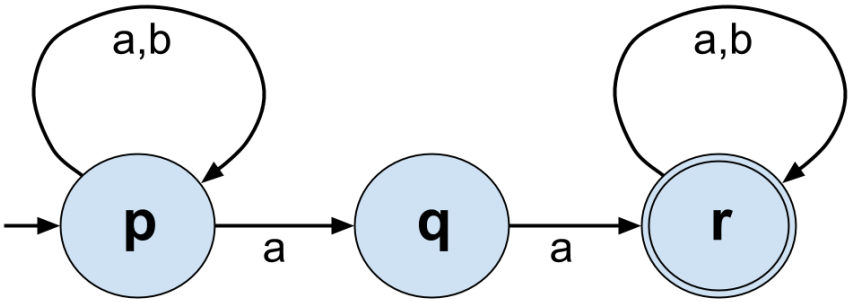


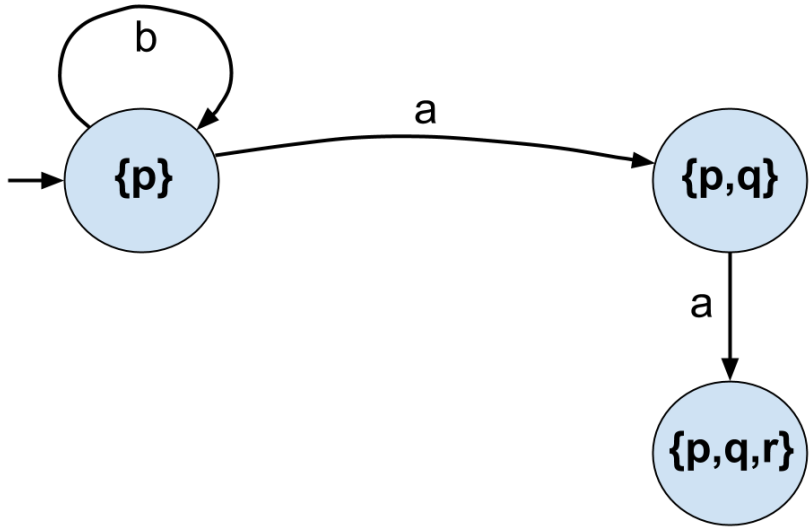
Construcción de un AEF a partir de un AEFND: ejemplo



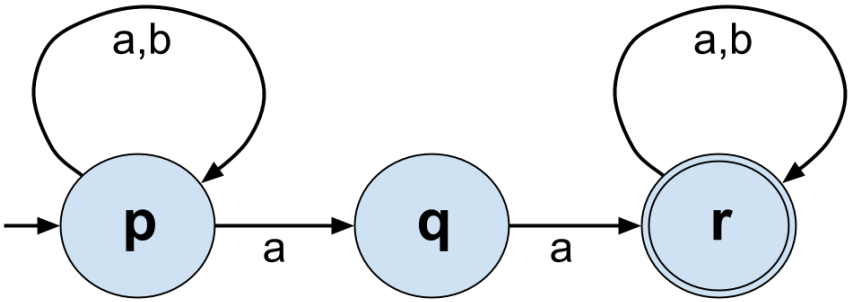


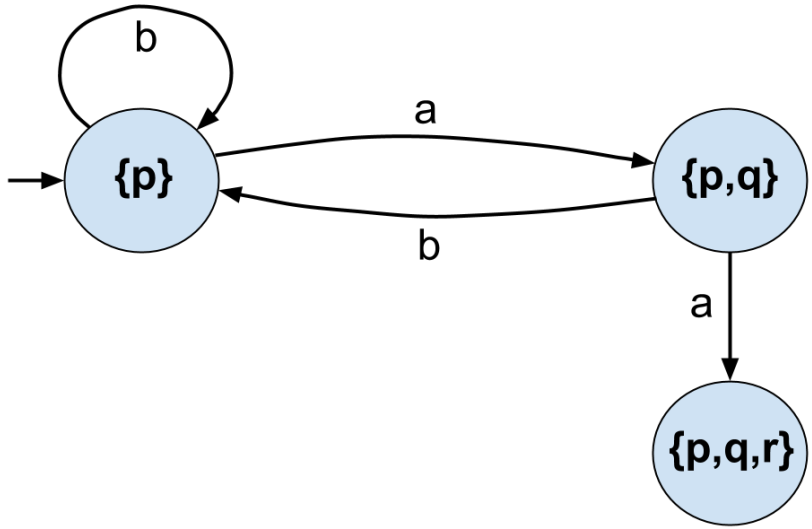
Construcción de un AEF a partir de un AEFND: ejemplo



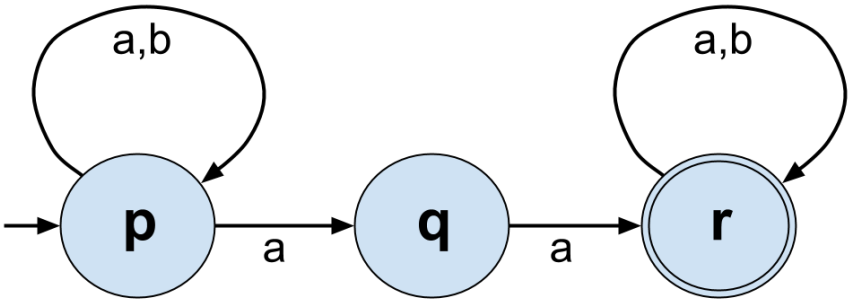


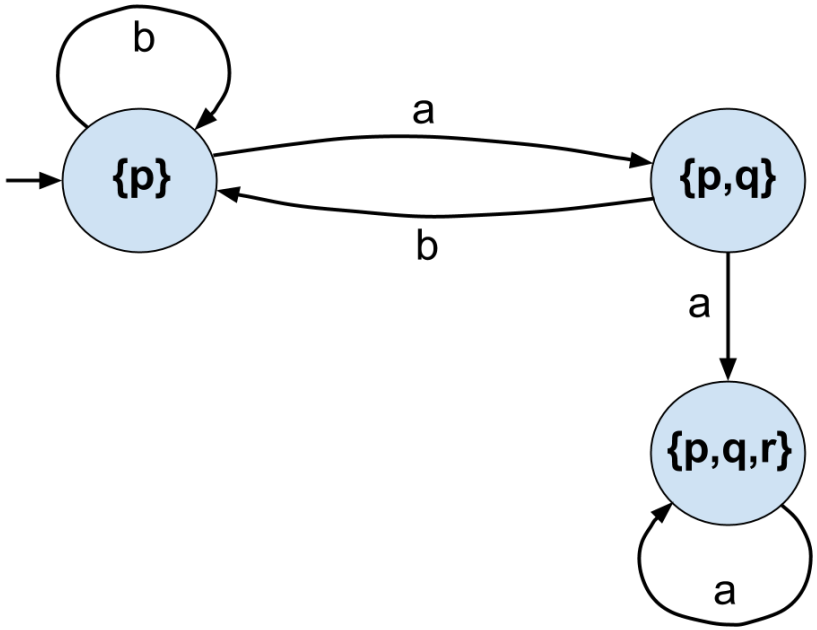
Construcción de un AEF a partir de un AEFND: ejemplo



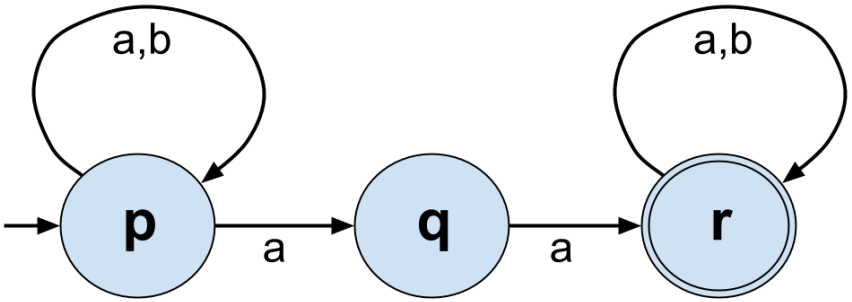


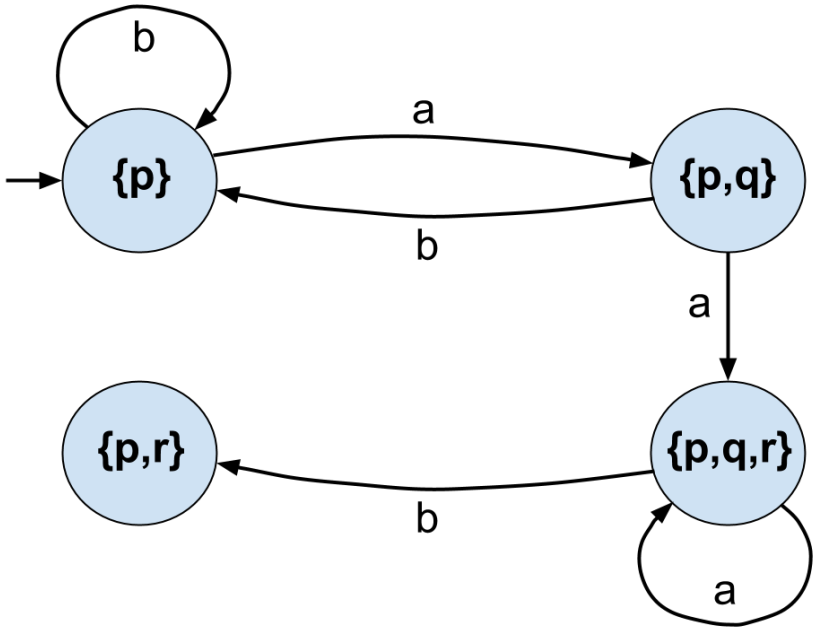
Construcción de un AEF a partir de un AEFND: ejemplo



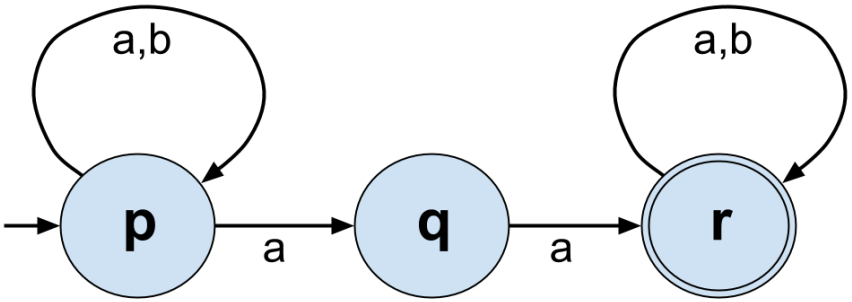


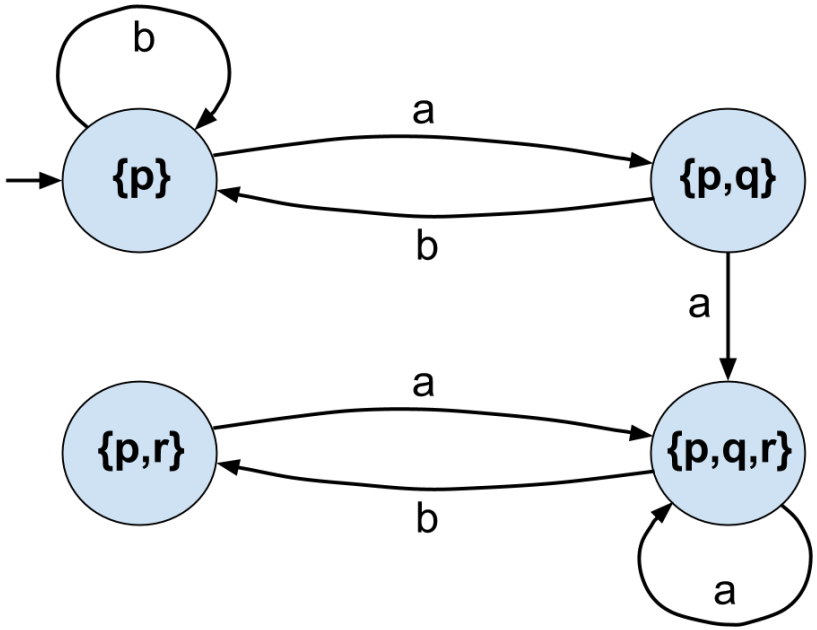
Construcción de un AEF a partir de un AEFND: ejemplo



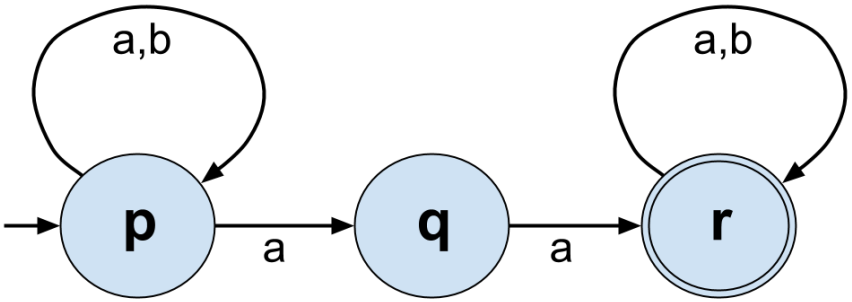


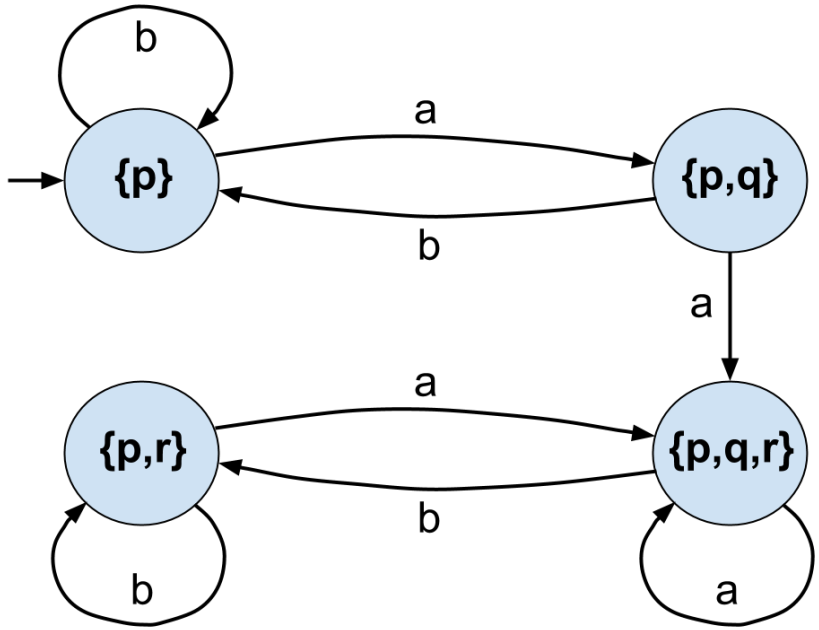
Construcción de un AEF a partir de un AEFND: ejemplo



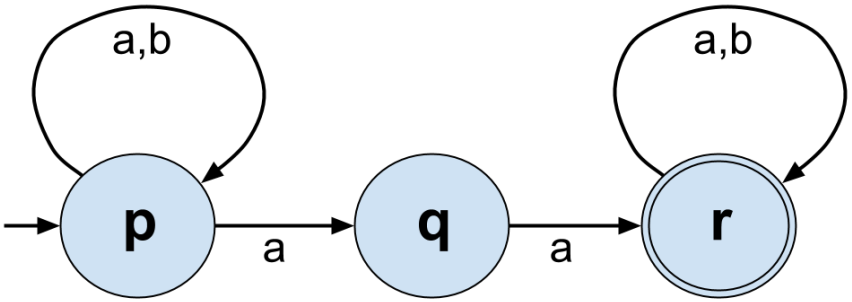


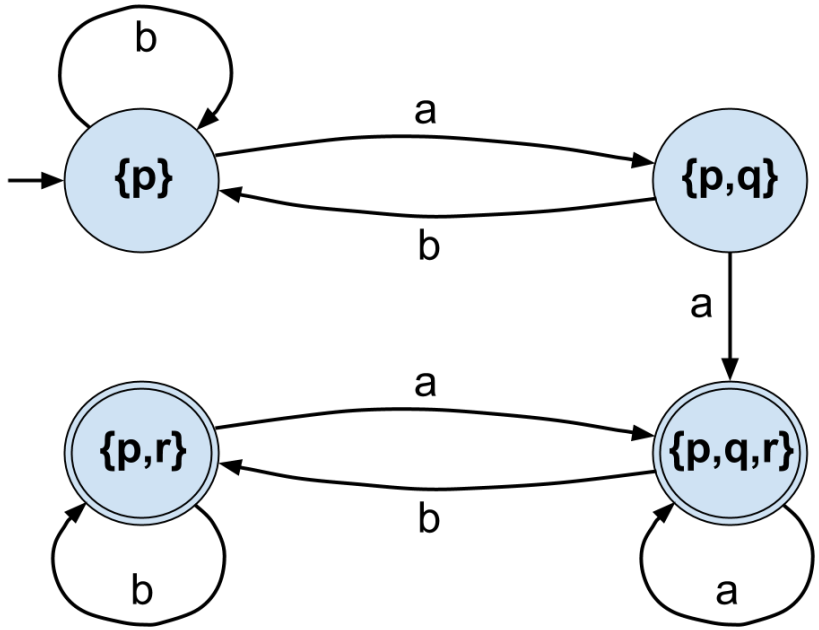
Construcción de un AEF a partir de un AEFND: ejemplo





Construcción de un AEF a partir de un AEFND: ejemplo





Construcción de un AEF a partir de un AEFND

Teorema: (de Kleene-Rabin-Scott)

Para cada AEFND existe un AEF que acepta el mismo lenguaje.

D/ Sea $A = (\Sigma, S, R, Ac, s_0)$ un AEFND y $A' = (\Sigma, S', f, Ac', s_0')$, donde:

- $S' = \mathcal{P}(S)$
- $Ac' = \{s' \in \mathcal{P}(S) \mid s' \cap Ac \neq \emptyset\}$
- $s_0' = \{s_0\}$
- $f : S' \times \Sigma \to S'$ tal que

$$f(s', x) = \{s \in S \mid \exists u \in s', (u, x, s) \in R\}$$

$f(s', x)$ es el conjunto de todos los estados de S a los que se puede llegar desde un estado de s' siguiendo un arco con etiqueta x.

Construcción de un AEF a partir de un AEFND

Veamos que $\mathcal{AC}(A) = \mathcal{AC}(A')$. Para mostrar que $w \in \mathcal{AC}(A)$ sii $w \in \mathcal{AC}(A')$ usaremos inducción sobre $|w|$. Más precisamente, mostraremos que $\forall n \in \mathbb{N}_0$ vale el siguiente enunciado E_n:

Para cada ruta en A que va de su estado inicial s_0 a un estado s_n, existe una ruta en A' que va de su estado inicial $s'_0 = \{s_0\}$ a un estado s'_n tal que $s_n \in s'_n$. Recíprocamente, para cada ruta en A' que va de s'_0 a un estado s'_n, y para cada $s_n \in s'_n$, existe una ruta en A que va de s_0 a s_n.

- **Caso base:** $n = 0$ trivial
  En A: $s_n = s_0$, corresponde al caso en que la entrada es λ.
  En A': $s'_n = s'_0$, idem.
- **Paso inductivo:** supongamos que vale E_n y veamos que vale E_{n+1}. Consideremos una ruta en A, de la forma $s_0, \ldots, s_n, s_{n+1}$, que recorre los arcos rotulados $x_1, \ldots, x_n, x_{n+1}$. Para todas las transiciones sobre esta ruta tenemos que $(s_i, x_{i+1}, s_{i+1}) \in R$; en particular, para la última de ellas $(s_n, x_{n+1}, s_{n+1}) \in R$.

Construcción de un AEF a partir de un AEFND

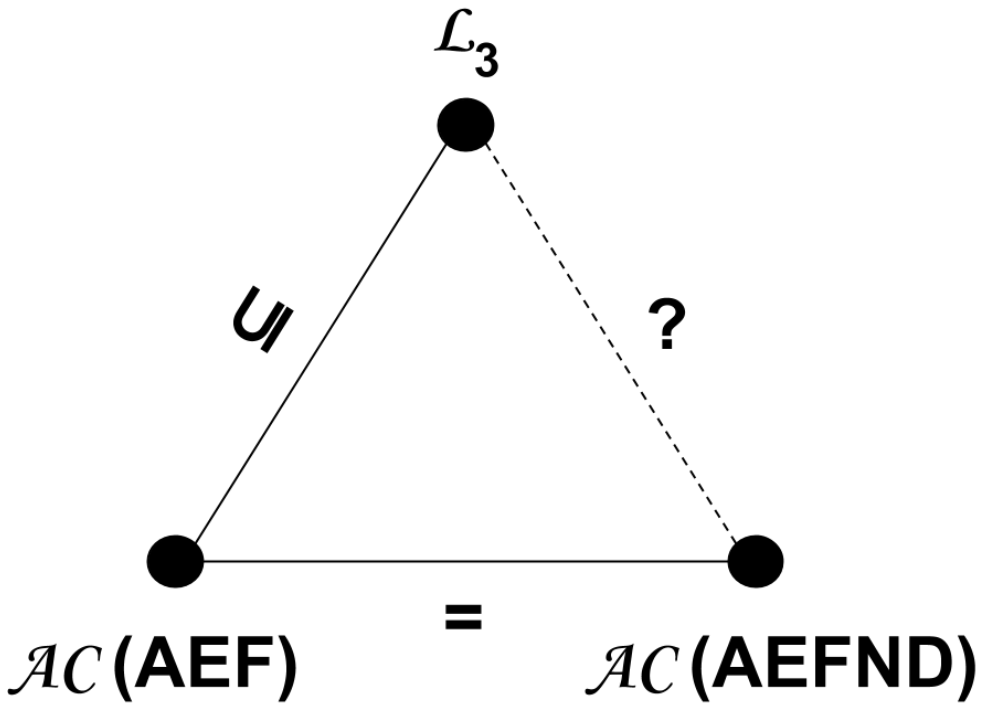
- **Paso inductivo (cont.):**

  Por H.I. existe una ruta en A' de la forma $s'_0, \ldots, s'_n$ tal que $s_n \in s'_n$. Dado que $(s_n, x_{n+1}, s_{n+1}) \in R$, existe un arco en A' rotulado x_{n+1} de s'_n a un estado que contiene a s_{n+1}. Llamémoslo s'_{n+1}. Por lo tanto existe una ruta en A', de s'_0 a s'_{n+1}, tal que $s_{n+1} \in s'_{n+1}$.

  Recíprocamente, consideremos una ruta en A' de la forma $s'_0, \ldots, s'_n, s'_{n+1}$ que recorre los arcos rotulados $x_1, \ldots, x_n, x_{n+1}$. Para todas las transiciones sobre esta ruta tenemos que $f(s'_i, x_{i+1}) = s'_{i+1}$; en particular, para la última de ellas $f(s'_n, x_{n+1}) = s'_{n+1}$.
  Por H.I., para cada $s_n \in s'_n$ debe existir una ruta en A que va de s_0 a s_n. Dado que $f(s'_n, x_{n+1}) = s'_{n+1}$, por definición de f tenemos que s'_{n+1} es el conjunto de estados $s \in S$ a los que se puede llegar desde un estado de s'_n siguiendo un arco con etiqueta x_{n+1}. Por lo tanto, para cada s en s'_{n+1}, existe una ruta en A que va desde s_0 a s. □

Relación entre AEF, AEFND y GR

- Teníamos que $\mathcal{AC}(AEF) \subseteq \mathcal{AC}(AEFND)$ (pág. 22).
  Acabamos de probar que $\mathcal{AC}(AEF) \supseteq \mathcal{AC}(AEFND)$.
  Por lo tanto $\mathcal{AC}(AEF) = \mathcal{AC}(AEFND)$.
- Por otro lado, teníamos que $\mathcal{AC}(AEF) \subseteq \mathcal{L}_3$ (pág. 11).
- Gráficamente:



Relación entre AEF, AEFND y GR

- Resulta fácil probar que $\mathcal{L}_3 \subseteq \mathcal{AC}(AEFND)$.
- **Ejemplo:**

  La gramática $G = (\{A, B, C, D\}, \{a, b\}, P, A)$, con P dada por:

$$A \to bA \mid aB \mid bC$$
$$B \to aB \mid bD \mid aA$$
$$C \to \lambda \mid aB \mid bD$$
$$D \to \lambda \mid aC \mid b$$

  se puede transformar en $G' = (\{A, B, C, D, X\}, \{a, b\}, P', A)$ con

$$P' = P - \{D \to b\} \cup \{D \to bX,\ X \to \lambda\},$$

  lo cual nos permite construir un AEFND como en un ejemplo ya visto.

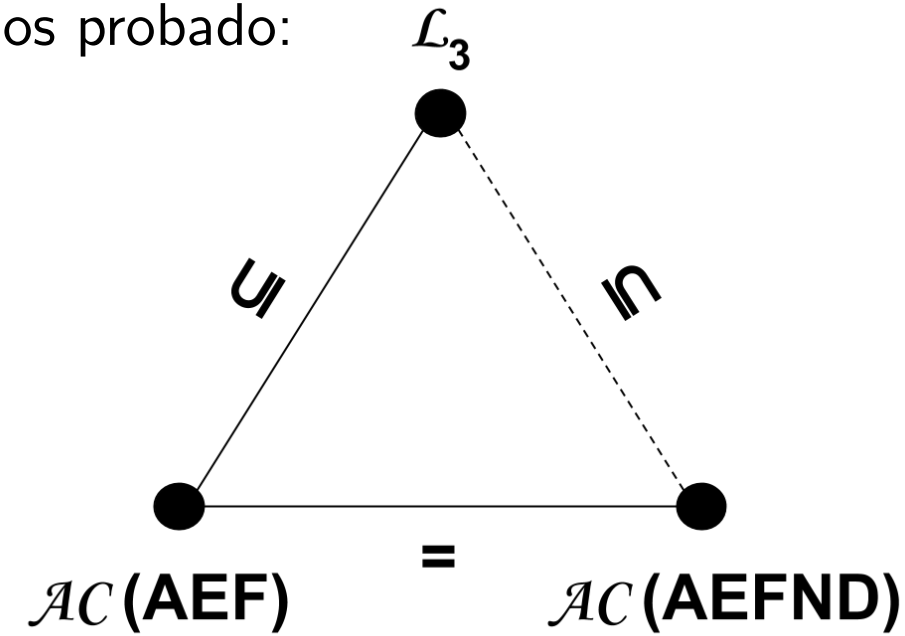
- Esta idea se usa para demostrar el siguiente:

Relación entre AEF, AEFND y GR

- **Teorema:** $\mathcal{L}_3 \subseteq \{L \mid \exists A \text{ AEFND} : \mathcal{AC}(A) = L\}$

  D/ No la hacemos, queda como ejercicio opcional.

- En resumen, hemos probado:



- Juntando los resultados anteriores, concluimos que:

$$\mathcal{L}_3 = \{L \mid \exists A \text{ AEF} : \mathcal{AC}(A) = L\} = \{L \mid \exists A \text{ AEFND} : \mathcal{AC}(A) = L\}$$

Expresiones regulares: introducción

- Al igual que los AEF, las expresiones regulares constituyen una caracterización de los lenguajes regulares.
- Sin embargo, proveen una descripción más detallada que los AEF sobre la composición de $\mathcal{L}_3$.
- El enfoque de las expresiones regulares es el de conjunto definido inductivamente.
- La idea es mostrar cómo se pueden construir los lenguajes regulares a partir de pequeños bloques de construcción:
  - Se comienza considerando los lenguajes más sencillos que pueden formarse con un alfabeto Σ: $\emptyset$, $\{x\}$ con $x \in \Sigma$.
  - Se utilizan formas precisas de combinación de estos lenguajes básicos: unión, concatenación y estrella de Kleene. Dichas operaciones son cerradas en $\mathcal{L}_3$.

Expresiones regulares

- **Definición:**

  Una expresión regular (ER) sobre un alfabeto Σ es una cadena sobre el alfabeto $\Sigma \cup \{ (,\ ),\ \circ,\ \cup,\ ^*,\ \emptyset \}$ definida inductivamente como:

  1. $\emptyset \in ER$
  2. Si $x \in \Sigma$, entonces $x \in ER$.
  3. Si $p, q \in ER$, entonces $(p \cup q) \in ER$.
  4. Si $p, q \in ER$, entonces $(p \circ q) \in ER$.
  5. Si $p \in ER$, entonces $(p^*) \in ER$.

- Para reducir el número de paréntesis, convenimos el siguiente orden de precedencia: *, $\circ$, $\cup$. Ejemplo: $((x^*) \cup (y \circ z)) = x^* \cup y \circ z$
- **Ejemplo:** $a^* \circ b \cup (a \cup c)^* \in ER$
- Obs. que una expresión regular es una cadena de símbolos, no es un lenguaje. ¿Qué lenguaje describe una expresión regular?

Expresiones regulares

- **Definición:**

  El lenguaje asociado a una expresión regular está dado por la función $L : ER \to \mathcal{L}$, que tiene las siguientes propiedades:

  1. $L(\emptyset) = \emptyset$
  2. Si $x \in ER, x \in \Sigma$, entonces $L(x) = \{x\}$
  3. Si $p, q \in ER$, entonces $L(p \cup q) = L(p) \cup L(q)$
  4. Si $p, q \in ER$, entonces $L(p \circ q) = L(p) \circ L(q)$
  5. Si $p \in ER$, entonces $L(p^*) = L(p)^*$

- **Ejemplos:**

  - $L(\emptyset^*) = L(\emptyset)^* = \emptyset^* = \{\lambda\}$
  - $L(a \cup b \circ b) = L(a) \cup L(b \circ b) = \{a\} \cup L(b) \circ L(b) =$
    $= \{a\} \cup \{b\} \circ \{b\} = \{a\} \cup \{bb\} = \{a, bb\}$
  - $L((x \circ y)^* \cup z^*) = L((x \circ y)^*) \cup L(z^*) = L(x \circ y)^* \cup L(z)^* =$
    $= (L(x) \circ L(y))^* \cup \{z\}^* = (\{x\} \circ \{y\})^* \cup \{z\}^* = \{xy\}^* \cup \{z\}^*$

Expresiones regulares

- **Definición:** $L_{ER} = \{L \in \mathcal{L} \mid \exists e \in ER : L(e) = L\}$

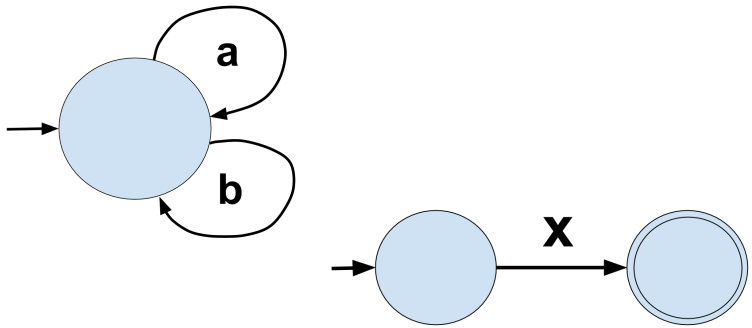
  es el conjunto de todos los lenguajes asociados a expresiones regulares.

- **Teorema:** $L_{ER} = \mathcal{L}_3$

  D/

  1) $L_{ER} \subseteq \mathcal{L}_3$: por inducción sobre ER.

  1. $e = \emptyset$: $L(e) = \emptyset \in \mathcal{L}_3$

  2. $e = x, \ x \in \Sigma$: $L(e) = \{x\} \in \mathcal{L}_3$

  3. $e = p \cup q, \ p, q \in ER$:
     Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \cup q) \in \mathcal{L}_3$.
     Pero $L(p \cup q) = L(p) \cup L(q)$ y la unión es cerrada en $\mathcal{L}_3$ (ejercicio).

Expresiones regulares

4. $e = p \circ q, \quad p, q \in ER$:

   Debemos probar que $L(p), L(q) \in \mathcal{L}_3 \Rightarrow L(p \circ q) \in \mathcal{L}_3$.

   Pero $L(p \circ q) = L(p) \circ L(q)$ y la concatenación es cerrada en $\mathcal{L}_3$ (ejercicio).

5. $e = p^*, \quad p \in ER$:

   Debemos probar que $L(p) \in \mathcal{L}_3 \Rightarrow L(p^*) \in \mathcal{L}_3$.

   Pero $L(p^*) = L(p)^*$ y la *-clausura de Kleene es cerrada en $\mathcal{L}_3$ (ejercicio).

2) $L_{ER} \supseteq \mathcal{L}_3$:

Sea $L \in \mathcal{L}_3$. Sabemos que $\exists A$ AEFND $\mid \mathcal{AC}(A) = L$. Se demuestra que existe una ER que genera el lenguaje aceptado por este autómata por inducción en el número de estados que no son ni el inicial ni de aceptación. Leerlo de Teoría de la Computación de J. Glenn Brookshear, pág. 63. □