Análisis Léxico

Análisis Léxico

Cálculo Discriminante de una Ecuación Cuadrática

Errores Léxicos

```
C:\Users\jdmar\Documents>disc.py
  File "C:\Users\jdmar\Documents\disc.py", line 4
    disc = b**2 - 4a*c
                  ^
SyntaxError: invalid syntax

C:\Users\jdmar\Documents>
```

disc.py | disc.java

```
1  a= int(input("Digite coeficiente cuadrático "))
2  b= int(input("Digite coeficiente lineal "))
3  c= int(input("Digite término independiente "))
4  disc = b**2 - 4a*c
5  print (disc)
6
```

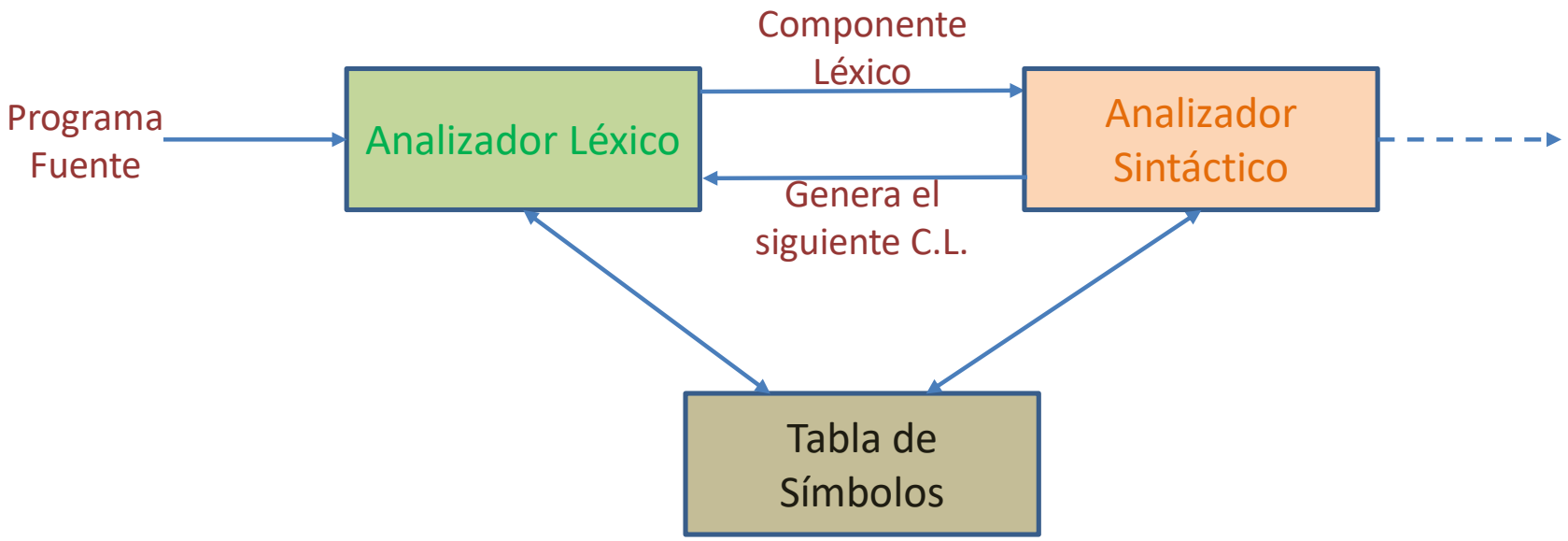
Python

```
C:\Users\Uninorte\Desktop>javac disc.java
disc.java:7: error: ';' expected
                disc = b*b - 4a*c;
                             ^

disc.java:7: error: not a statement
                disc = b*b - 4a*c;
                             ^

2 errors

C:\Users\Uninorte\Desktop>
```

```
1  public class disc{
2    public static void main(String[] args){
3      double a = Double.parseDouble(args[0]);
4      double b = Double.parseDouble(args[1]);
5      double c = Double.parseDouble(args[2]);
6      double disc;
7      disc = b*b - 4a*c;
8      System.out.println(disc);
9    }
10 }
```

Java

Análisis Léxico



Analizador Léxico: Primera fase de un compilador. Lee los caracteres de entrada suministrados por un programa fuente y elabora una secuencia de componentes léxicos que toma el Analizador Sintáctico para hacer el análisis.

Componentes Léxico, Patrones y Lexemas

Componente Léxico	Lexemas de Ejemplo	Descripción del Patrón o Regla
	x4tu, v3, saldo38	
	56, 3432, 4	
	+	
	-	
	If	
	while	
	>	

Componentes Léxico, Patrones y Lexemas

Componente Léxico	Lexemas de Ejemplo	Descripción del Patrón o Regla
id	x4tu, v3, saldo38	Inician por letra y siguen cero o más letras y/o dígitos
num	56, 3432, 4	Inician por un dígito y siguen cero o más dígitos
suma	+	+
resta	-	-
if	if	if
while	while	while
mayorque	>	>

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Alfabeto o clase de caracter:

Conjunto finito de símbolos. Se denota por Σ.

Ejemplos:

$$\Sigma_1 = \{a, b, c, \dots, z\}$$

$$\Sigma_2 = \{0,1,2, \dots, 9\}$$

$$\Sigma_3 = \{0,1\}$$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Secuencia finita de símbolos tomados de un alfabeto Σ.

Ejemplos:

De $\Sigma_1 = \{a, b, c, \ldots, z\}$, s_1="casa", s_2= "pesado"

De $\Sigma_2 = \{0,1,2, \ldots, 9\}$, t_1=3429, t_2= 87632

De $\Sigma_3 = \{0,1\}$, u_1=1011101, u_2= 11101

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Longitud de una cadena *s*, escrita como $|s|$, es el número de apariciones de símbolos en *s*.

Ejemplos:

s_1="casa", $|s_1|=4$; s_2= "pesado", $|s_2|=6$

$t_1=3429$, $|t_1|=4$; $t_2= 87632$, $|t_2|=5$

$u_1=1011101$, $|u_1|=7$; $u_2= 11101$, $|u_2|=5$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

La cadena *vacía*, representada por $\in$, es una cadena especial de longitud cero. Es decir, $|\in|=0$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Concatenación de cadenas: Si x e y son cadenas definidas en los alfabetos Σ_1 y Σ_2 o en el mismo alfabeto Σ; la concatenación xy, es la cadena resultante de agregar y a x.

Ejemplo:

Si *x*="casa" e *y*="quinta", *xy* = "casaquinta"

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Elemento Identidad de Concatenación: La cadena vacía es el elemento identidad de la concatenación. Es decir, $s\in = \in s = s$

Pregunta:

Si x es una cadena definida en Σ_1 e y es una cadena definida en Σ_2, o ambas definidas en Σ, a qué será igual $|xy|$?

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Concatenación como producto: La concatenación se puede considerar el producto de cadenas.

Exponenciación de cadenas: Si s es una cadena definida en Σ, luego ss, sss, $sssss$, etc. están definida en Σ. Se podría resumir su escritura como s^2, s^3, s^5, respectivamente.

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Cadena, frase, palabra o *string*:

Exponenciación de cadenas: La exponenciación se define como:

$s^0 = \in$ y si $i > 0$, $s^i = s^{i-1}s = ss^{i-1}$

Luego,

$s^0 = \in$

$s^1 = s^{1-1}s = ss^{1-1} = s^0 s = ss^0 = s$

$s^2 = s^{2-1}s = ss^{2-1} = s^1 s = ss^1 = ss$

$s^3 = s^{3-1}s = ss^{3-1} = s^2 s = ss^2 = sss$

Pregunta:

Si x es una cadena definida en Σ_1 e y es una cadena definida en Σ_2, o ambas definidas en Σ, a qué será igual $|x^i y^j|$, donde i, j son números naturales ?

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Lenguaje:

Cualquier conjunto de cadenas de un alfabeto Σ.

Ejemplos:

$L_1 = \{\quad\} = \emptyset$

$L_2 = \{\in\}$

$L_3 = \{00,10,11,01\}$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Partes de una cadena:

Prefijo de *s*: Cadena que se obtiene al eliminar cero o más símbolos desde la derecha de la cadena *s*.

Ejemplos:

cam es prefijo de camino

camino es prefijo de camino

$\in$ es prefijo de camino

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Partes de una cadena:

Sufijo de *s*: Cadena que se obtiene al eliminar cero o más símbolos desde la izquierda de la cadena *s*.

Ejemplos:

mino es sufijo de camino

camino es sufijo de camino

$\in$ es sufijo de camino

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Partes de una cadena:

Subcadena de *s*: Cadena que se obtiene al eliminar un prefijo y un sufijo de la cadena *s*.

Todo prefijo y sufijo de s es una subcadena de s, pero no toda subcadena de s es un prefijo o sufijo de s.

Ejemplos:

ami es subcadena de camino

camino es subcadena de camino

$\in$ es subcadena de camino

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Partes de una cadena:

Prefijo, Sufijo o Subcadena propios de *s*:
Cualquier cadena *x*, donde $x \neq \in$, y respectivamente *x* es un prefijo, sufijo o subcadena de *s* tal que $s \neq x$.

Ejemplos:

cam es prefijo propio de camino

mino es sufijo propio de camino

ami es subcadena propia de camino

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Partes de una cadena:

Subsecuencia de *s*: Cualquier cadena formada por la eliminación de cero o más símbolos no necesariamente contiguos de ***s***.

Ejemplos:

can es subsecuencia de camino

mio es subsecuencia de camino

ami es subsecuencia de camino

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Unión de *L* y *M*:

$$\boldsymbol{L \cup M = \{s / s \text{ está en } L \text{ o } s \text{ está en } M\}}$$

Ejemplo:

Sea $L = \{a, b, c, \dots, z\}$ y $M = \{0,1, \dots, 9\}$

$L \cup M = \{a, b, c, \dots, z, 0,1, \dots, 9\}$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Concatenación de *L* y *M*:

$$\boldsymbol{LM} = \{\boldsymbol{st}/\boldsymbol{s} \text{ está en } \boldsymbol{L} \text{ y } \boldsymbol{t} \text{ está en } \boldsymbol{M}\}$$

Ejemplo:

Sea $L = \{a, b, c, \dots, z\}$ y $M = \{0,1, \dots, 9\}$

$LM = \{a0, a1, \dots, a9, b0, b1, \dots, b9, \dots, z0, z1, \dots, z9\}$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Exponenciación de *L*:

$$\boldsymbol{LL} = \{\boldsymbol{st}/\boldsymbol{s} \text{ está en } \boldsymbol{L} \text{ y } \boldsymbol{t} \text{ está en } \boldsymbol{L}\}$$

Se define:

$L^0 = \{\in\}$

$L^i = L^{i-1}L = LL^{i-1}, i > 0$

Luego:

$L^1 = L^{1-1}L = LL^{1-1} = L^0L = LL^0 = \{\in\}L = L\{\in\} = L$

$L^2 = L^{2-1}L = LL^{2-1} = L^1L = LL^1 = LL$

Ejemplo:

Sea $L = \{a, b, c, \dots, z\}$

$L^2 = \{aa, ab, \dots, az, ba, bb, \dots, bz, \dots, za, zb, \dots, zz\}$

$= \{\boldsymbol{a^2}, ab, \dots, az, ba, \boldsymbol{b^2}, \dots, bz, \dots, za, zb, \dots, \boldsymbol{z^2}\}$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Cerradura de Kleene de *L*:

$$\boldsymbol{L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \cdots \cup L^\infty}$$

Ejemplo:

Sea $L = \{a, b, c, \ldots, z\}$

$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots \cup L^\infty = \{\in\} \cup L \cup L^2 \cup \ldots \cup L^\infty$

$L^* = \{\in, a, b, c, \ldots, z, \boldsymbol{a^2}, ab, \ldots, az, ba, \boldsymbol{b^2}, \ldots, bz, \ldots, za, zb, \ldots, \boldsymbol{z^2}, \ldots\}$

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Cerradura Positiva de *L*:

$$\boldsymbol{L^{+} = \bigcup_{i=1}^{\infty} L^{i} = L^{1} \cup L^{2} \cup \cdots \cup L^{\infty}}$$

Ejemplo:

Sea $L = \{a, b, c, \ldots, z\}$

$L^{+} = L^{1} \cup L^{2} \cup \cdots \cup L^{\infty} = L \cup L^{2} \cup \ldots \cup L^{\infty}$

$L^{+} = \{a, b, c, \ldots, z, \boldsymbol{a^{2}}, ab, \ldots, az, ba, \boldsymbol{b^{2}}, \ldots, bz, \ldots, za, zb, \ldots, \boldsymbol{z^{2}}, \ldots\}$

Pregunta: ¿A que equivalen L^{*} en función de L^{+} y L^{+} en función de L^{*}?

Especificación de Componentes Léxicos

Alfabeto, Cadenas y Lenguajes

Operaciones sobre lenguajes

Ejercicios:

Dado los lenguajes $L = \{a, b, c, \dots, z\}$ y $M = \{0,1, \dots, 9\}$, obtener o describir los siguientes lenguajes:

a) LM^2 c) ML^* e) $L(L \cup M)^*$

b) $L(L \cup M)$ d) $M(L \cup M)^3$ f) M^+

Especificación de Componentes Léxicos

Expresiones Regulares

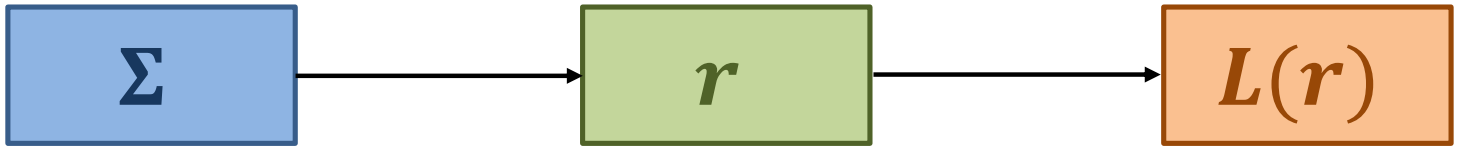
https://xkcd.com/208/

Especificación de Componentes Léxicos

Expresiones Regulares

Es una norma o regla definida sobre un alfabeto Σ y a partir de la cual se generan las cadenas de un lenguaje L.

Si r es una expresión regular (e.r.) definida sobre Σ, luego $L(r)$ es el lenguaje generado a partir de r.



Especificación de Componentes Léxicos

Expresiones Regulares

Construcción

Se construyen a partir de expresiones regulares más simples usando un método inductivo:

Caso Base:

1. $\in$ es una e.r. que designa al lenguaje $\{\in\}$.
2. Si a es un símbolo de Σ, entonces a es una e.r. que designa al lenguaje $\{a\}$.

Especificación de Componentes Léxicos

Expresiones Regulares

Construcción

Suposición:

Si $\boldsymbol{r}$ y $\boldsymbol{s}$ son e.r. definidas en los alfabetos Σ_1 y Σ_2, respectivamente, o definidas en el mismo Σ; y además, generan los lenguajes $L(r)$ y $L(s)$, se induce que:

Paso Inductivo:

3. $r|s$ es una e.r. que genera al lenguaje $L(r|s) = L(r) \cup L(s)$.
4. rs es una e.r. que genera al lenguaje $L(rs) = L(r)L(s)$.
5. r^*es una e.r. que genera al lenguaje $L(r^*) = (L(r))^* = L^*(r)$.

Especificación de Componentes Léxicos

Expresiones Regulares

Construcción

Ejemplos

1. Construir una e.r. en $\Sigma = \{a, b\}$, que genere las cadenas que finalicen en abb, no importa el prefijo de inicio. Ej: abb, $aababb$.
2. Construir una e.r. que genere los número binarios.
3. Construir una e.r. que genere los binarios pares.
4. Construir una e.r. que genere los binarios impares.
5. Construir una e.r. que genere los binarios con un número impar de 1's, no importa la cantidad de 0's. Ej: 1, 010011, 10010110010, 00111001001000011100.

Leer: Fig. 3.9

Especificación de Componentes Léxicos

Definiciones Regulares

Por notación, es conveniente dar nombres a las e.r. y definir más e.r. utilizando estos nombres como si fueran símbolos.

Dado un alfabeto Σ, una *definición regular* (d.r.) es una secuencia de definiciones de la forma:

$$d_1 \rightarrow r_1$$
$$d_2 \rightarrow r_2$$
$$\dots$$
$$d_n \rightarrow r_n$$

donde cada d_i es un nombre distinto, y cada r_i es una e.r. sobre los símbolos de $\Sigma \cup \{d_1, d_2, \dots, d_{i-1}\}$.

No se utiliza un d_j con $j > i$ para construir r_i.

Especificación de Componentes Léxicos

Definiciones Regulares

Ejemplos:

Construir d.r. para:

1. Las variables de un programa.
2. Los números reales positivos.
3. La identidad de archivos en DOS, donde un archivo consta de un nombre alfanumérico de cualquier longitud; seguido por una extensión que contiene un punto(.), y máximo tres caracteres alfanuméricos.
4. La ruta de un directorio en DOS.

Especificación de Componentes Léxicos

Definiciones Regulares

Abreviaturas en la notación:

1. **Uno o más casos:**

$$a^*a = aa^* = a^+$$

2. **Cero o un caso:**

$$a| \in = \in |a = a?$$

3. **Clases de carácter:** Selección de un carácter del conjunto.

$$a|b|c| \dots |z = [a - z]$$
$$0|1| \dots |9 = [0 - 9]$$
$$a|b|c| \dots |z|A|B|C| \dots |Z = [a - zA - Z]$$
$$a|b|c| \dots |z|0|1|2| \dots |9 = [a - z0 - 9]$$

Autómatas Finitos

Reconocedor de un lenguaje: Programa que toma como entrada una cadena *x* y responde "si" si *x* es una frase que el programa reconoce para un lenguaje, y "no", si no lo es.

A partir de una e.r., se construye un reconocedor utilizando un diagrama de transiciones llamado autómata finito

Autómatas Finitos

Tipos de Autómatas

Tipo	Memoria	Complejidad	Reconocimiento
Determinístico	Menos	Más Complejos	Más rápido
No Determinístico	Más	Menos Complejos	Menos rápido

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Modelo matemático formado por una quíntupla

$$N = \{S, \Sigma, s_0, F, mueve(\delta)\}$$

1. Un conjunto de *estados* S
2. Un conjunto de símbolos de entrada Σ (alfabeto de símbolos de entrada)
3. Un estado s_0 denominado el *estado de inicio o inicial* (s_0 está en S)
4. Un conjunto de *estados de finalización o de reconocimiento* F ($F \subseteq S$)
5. Una función de transición $mueve\ o\ \delta$ que transforma pares estado-símbolos en conjuntos de estados.

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Función de transición $\boldsymbol{mueve}$:

$$mueve: S \times \Sigma \rightarrow \wp(S)$$

$$mueve(s, a) = S', S' está\ en\ \wp(S)$$

Nota: Dado un conjunto A, el conjunto $\wp(A)$ se denomina *Potencia de* A y es el conjunto formado por todos los subconjuntos de A.

$$\wp(A) = \{X / X \subseteq A\}$$

Ejemplo:

$$A = \{a, e, o\}, \text{luego}$$

$$\wp(A) = \{\emptyset, A, \{a\}, \{e\}, \{o\}, \{a, e\}, \{a, o\}, \{e, o\}\}$$

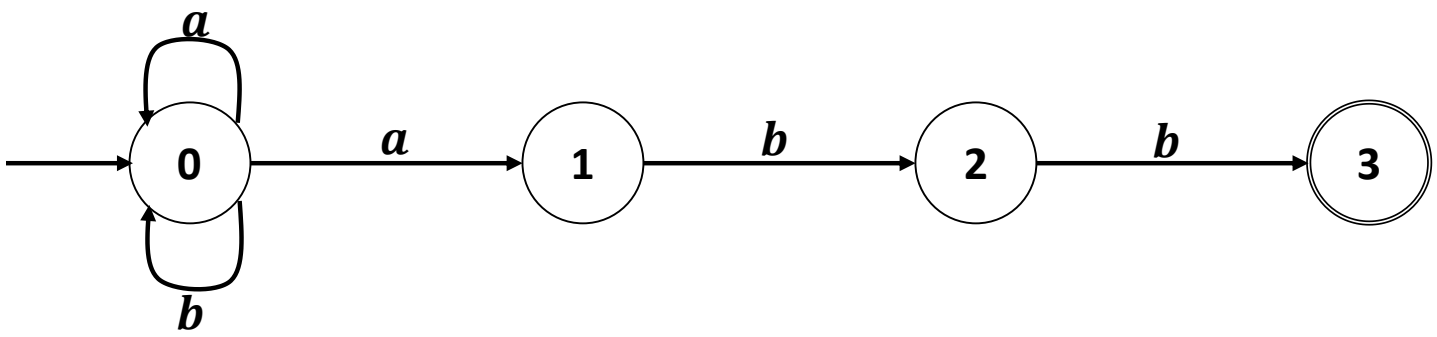
$$|A| = 3 \text{ y } |\wp(A)| = 2^{|A|} = 8$$

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Se puede representar mediante un *grafo dirigido etiquetado*, llamado *grafo de transiciones.*

Ejemplo:



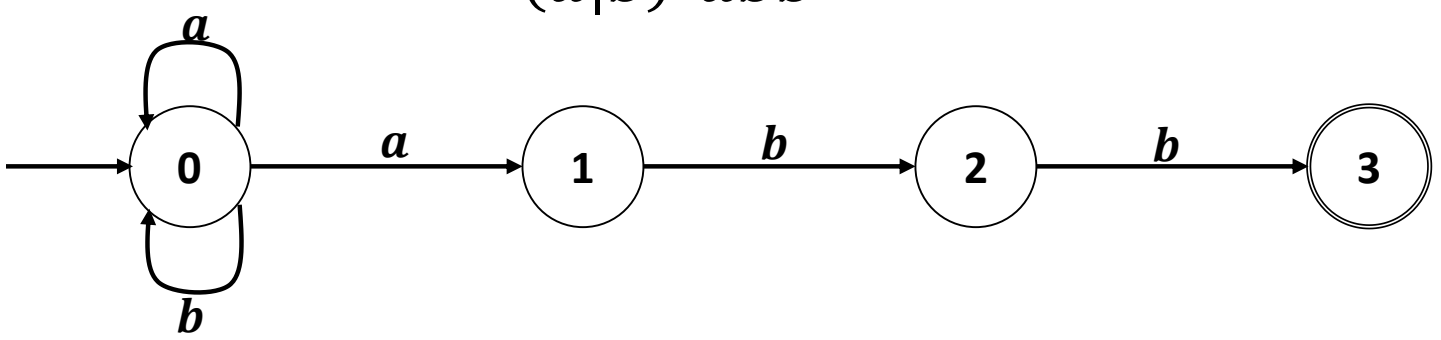
$$(a|b)^*abb$$

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Dado el AFN:

$$(a|b)^*abb$$



Componentes:

$S = \{0,1,2,3\}$

$\Sigma = \{a, b\}$

$s_0 = 0$

$F = \{3\}$

$mueve(0, a) = \{0,1\}$

$mueve(0, b) = \{0\}$

$mueve(1, a) = \{\}$

$mueve(1, b) = \{2\}$

$mueve(2, a) = \{\}$

$mueve(2, b) = \{3\}$

$mueve(3, a) = \{\}$

$mueve(3, b) = \{\}$

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Dado el AFN:

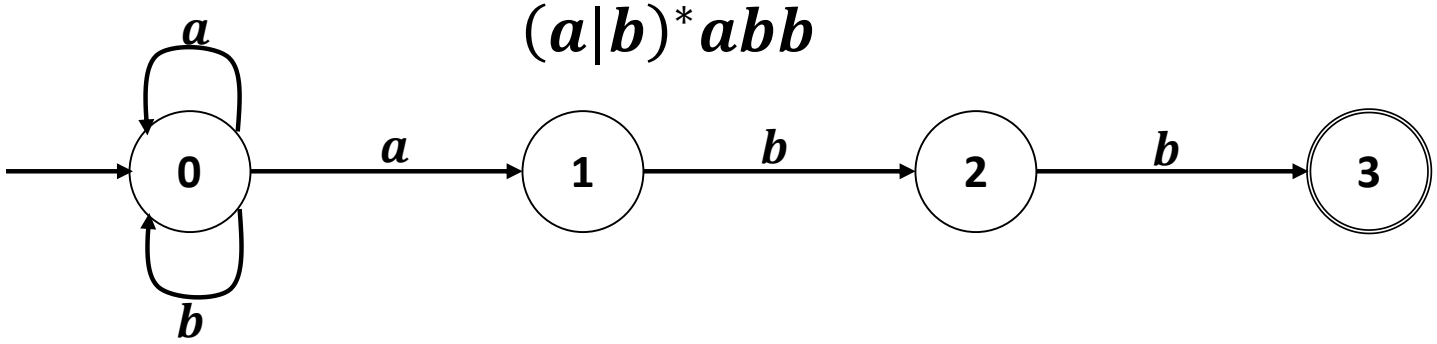


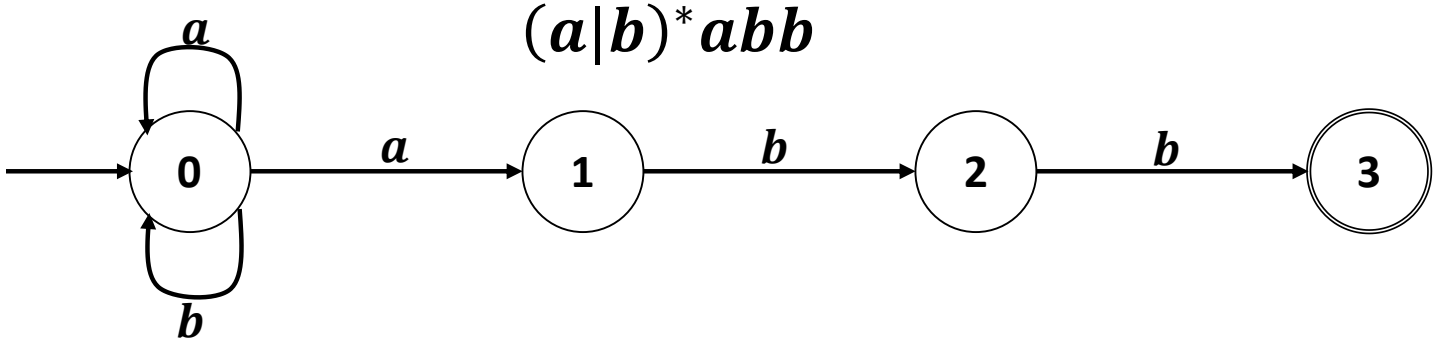
Tabla de Transiciones

Estado	a	b
→0	{0,1}	{0}
1	{ }	{2}
2	{ }	{3}
*3	{ }	{ }

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Dado el AFN:



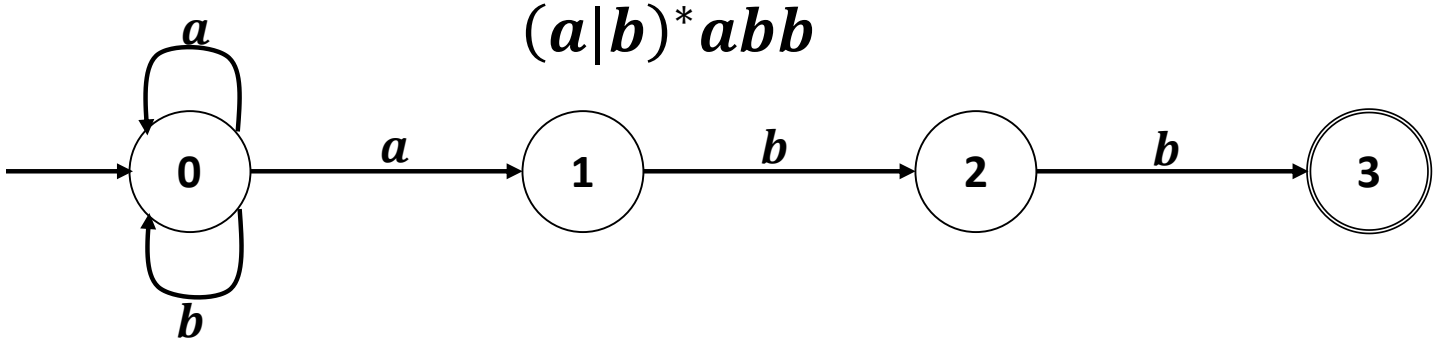
Reconocimiento o Aceptación

Un AFN **acepta** una cadena de entrada *x*, si y solo si, hay algún camino en el grafo de transiciones desde el estado de inicio a algún estado de aceptación, de tal forma que las etiquetas de las aristas a lo largo del camino deletreen a *x*.

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Dado el AFN:



Reconocimiento o Aceptación

El camino se realiza mediante transiciones llamadas *movimientos*

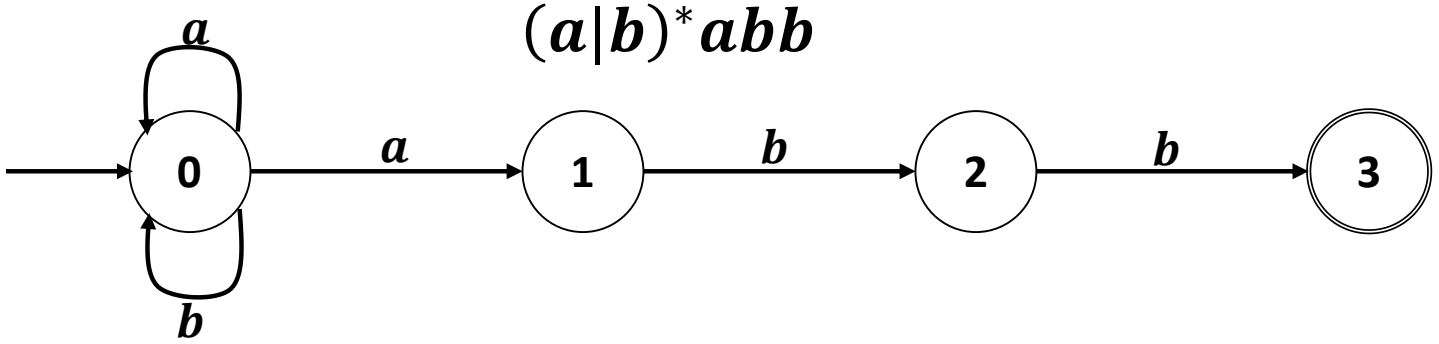
Ejemplo:

Reconocer $abb, ababb, ababa$

Autómatas Finitos

Autómatas Finitos No Determinístico (AFN)

Dado el AFN:



Reconocimiento o Aceptación

Ejemplo: abb

0 $\xrightarrow{a}$ 0 $\xrightarrow{b}$ 0 $\xrightarrow{b}$ **0** *No Reconoce*

0 $\xrightarrow{a}$ 1 $\xrightarrow{b}$ 2 $\xrightarrow{b}$ **3** *Si Reconoce*

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Modelo matemático formado por una quíntupla

$$D = \{S, \Sigma, s_0, F, mueve(\delta)\}$$

1. Un conjunto de *estados* S
2. Un conjunto de símbolos de entrada Σ (alfabeto de símbolos de entrada)
3. Un estado s_0 denominado el *estado de inicio o inicial* (s_0 está en S)
4. Un conjunto de *estados de finalización o de reconocimiento* F ($F \subseteq S$)
5. Una función de transición $mueve\ o\ \delta$ que transforma pares estado-símbolos en estados del AFD.

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Función de transición $\boldsymbol{mueve}$:

$$mueve: S \times \Sigma \rightarrow S$$

$$mueve(s, a) = s', s' está\ en\ S$$

Nota: A diferencia del AFN, en el AFD, la función de transición $mueve$ genera una estado de S, a partir del par (s, a).

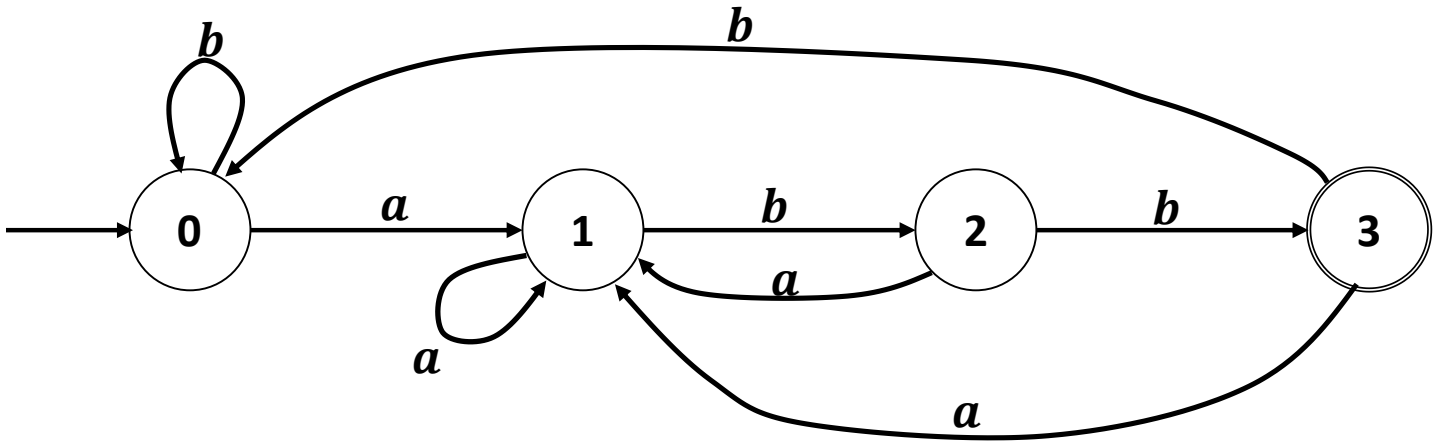
Al AFD se le considera un caso especial del AFN.

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Se representa también con el *grafo de transiciones.*

Ejemplo:

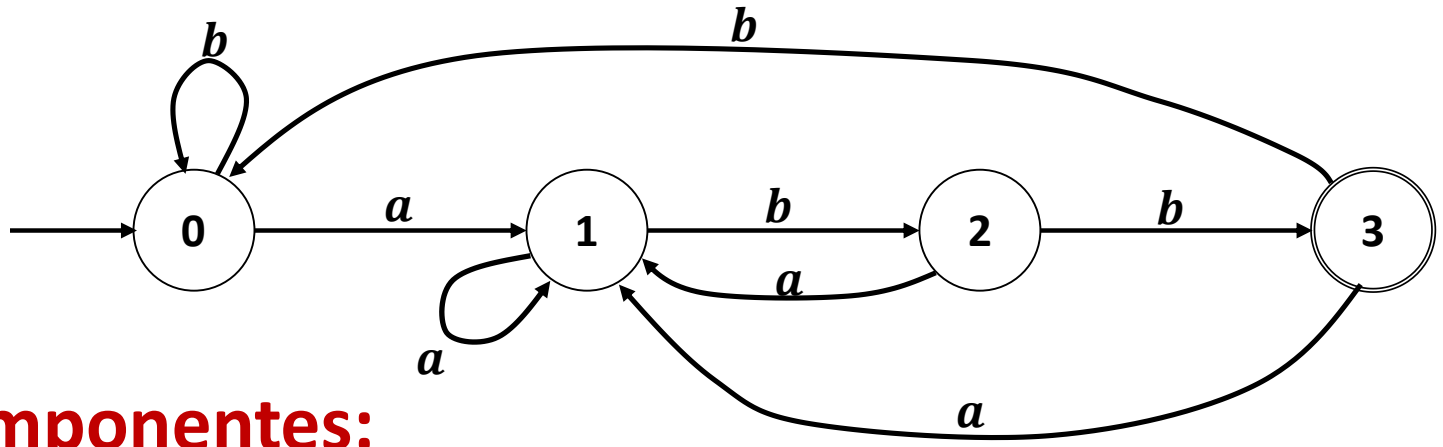


$$(a|b)^*abb$$

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Dado el AFD: $(a|b)^*abb$



Componentes:

$S = \{0,1,2,3\}$

$\Sigma = \{a, b\}$

$s_0 = 0$

$F = \{3\}$

$mueve(0, a) = 1$
$mueve(0, b) = 0$
$mueve(1, a) = 1$
$mueve(1, b) = 2$

$mueve(2, a) = 1$
$mueve(2, b) = 3$
$mueve(3, a) = 1$
$mueve(3, b) = 0$

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Dado el AFN: $(a|b)^*abb$

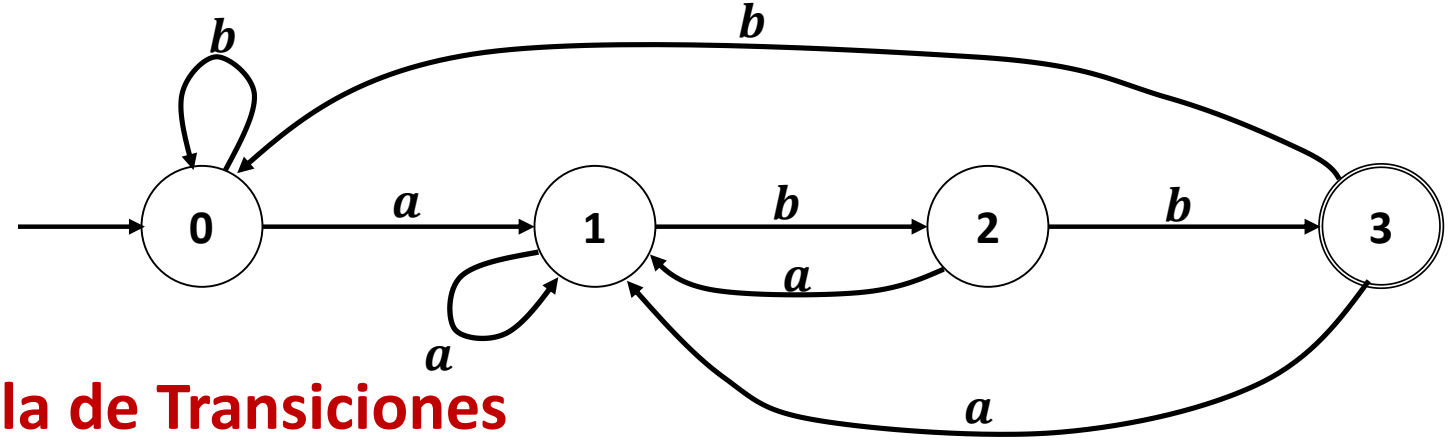


Tabla de Transiciones

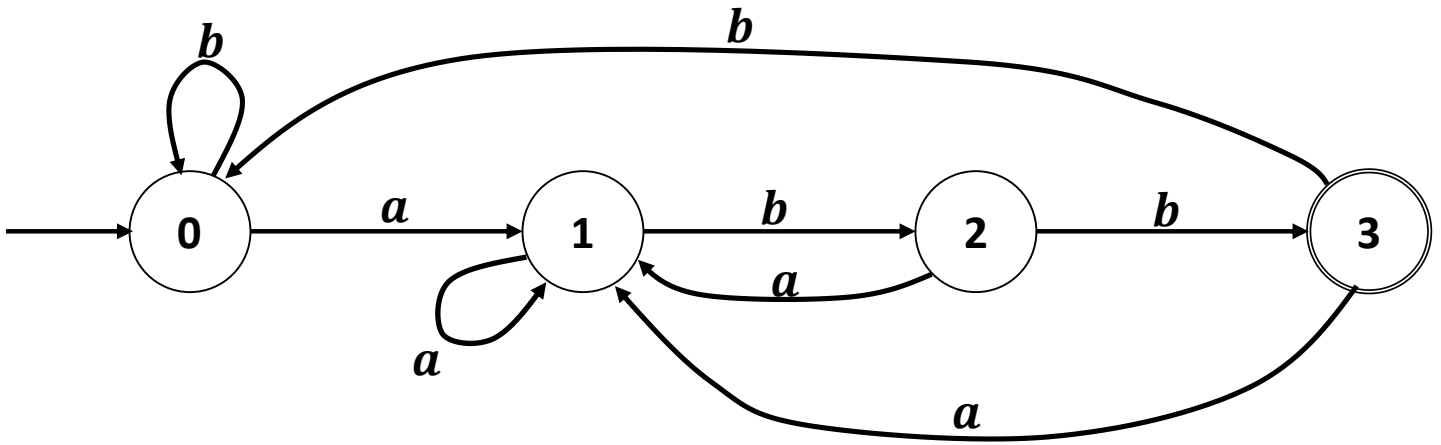
Estado	a	b
→**0**	1	0
1	1	2
2	1	3
***3**	1	0

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Dado el AFN:

$$(a|b)^*abb$$



Reconocimiento o Aceptación

El camino se realiza mediante transiciones llamadas *movimientos*

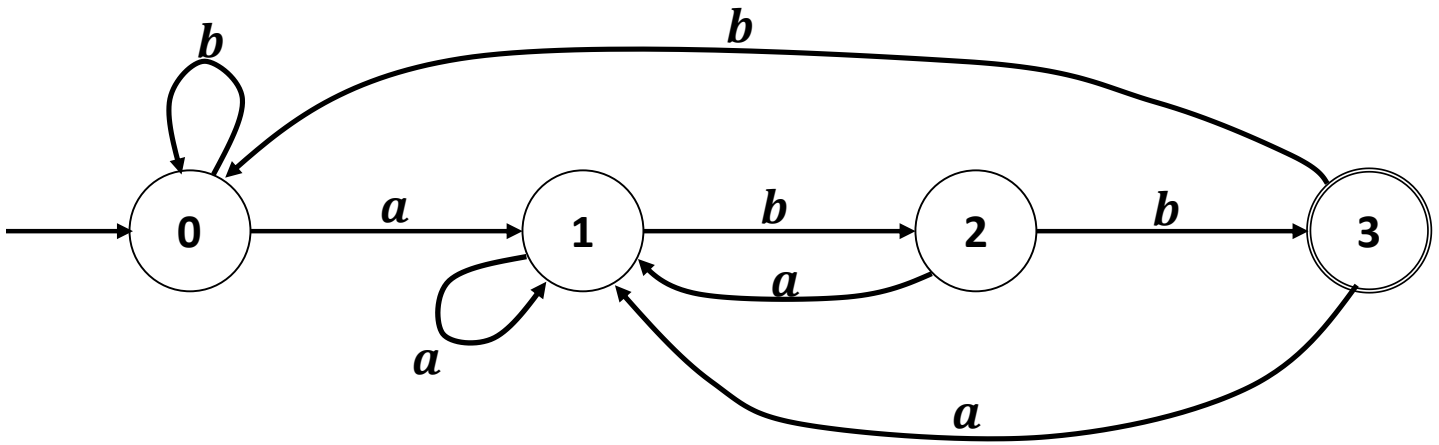
Ejemplo:

Reconocer $abb, ababb, ababa$

Autómatas Finitos

Autómatas Finitos Determinístico (AFD)

Dado el AFN: $(a|b)^*abb$



Reconocimiento o Aceptación

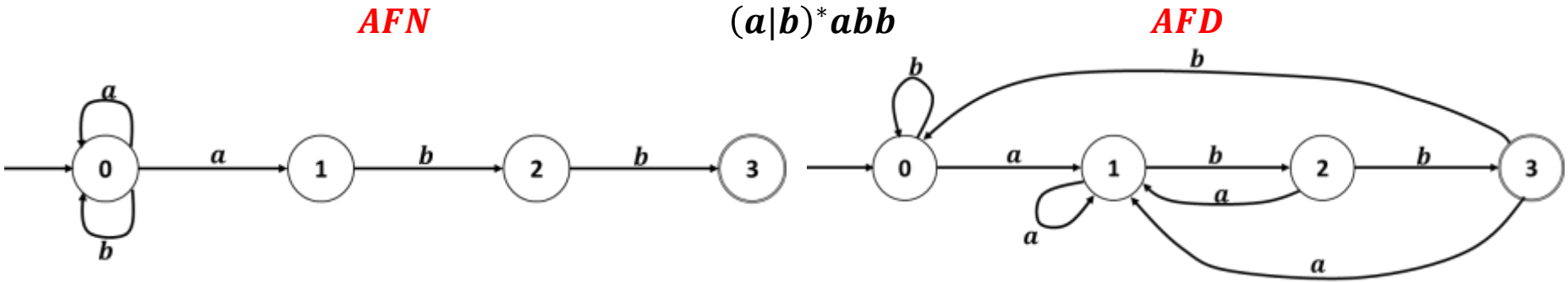
Ejemplo: abb, $ababa$

0 $\xrightarrow{a}$ 1 $\xrightarrow{b}$ 2 $\xrightarrow{b}$ **3** *Si Reconoce* abb

0 $\xrightarrow{a}$ 1 $\xrightarrow{b}$ 2 $\xrightarrow{a}$ 1 $\xrightarrow{b}$ 2 $\xrightarrow{a}$ **1** *No Reconoce* $ababa$

Autómatas Finitos

Diferencias entre AFN y AFD



1. En un AFD, para cada estado s y cada símbolo de entrada a, hay como máximo una arista etiquetada con a que sale de s.

2. En un AFN puede haber aristas etiquetadas con la entrada $\in$.

Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Método Inductivo

Entrada: Una e.r. r definida en un alfabeto Σ.

Salida: Un AFN N que acepte el lenguaje $L(r) = L(N)$.

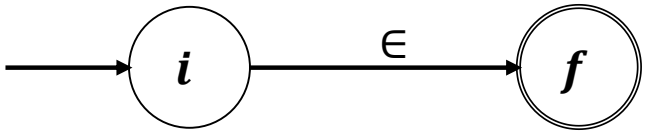
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

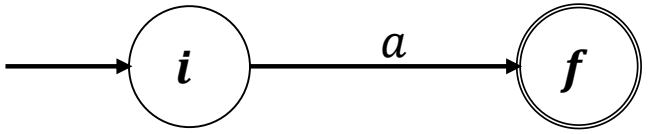
Caso Base:

1. Para la e.r. ∈, el AFN es el siguiente:



Este AFN reconoce {∈}

2. Para la e.r. a, el AFN es el siguiente:



Este AFN reconoce $\{a\}$

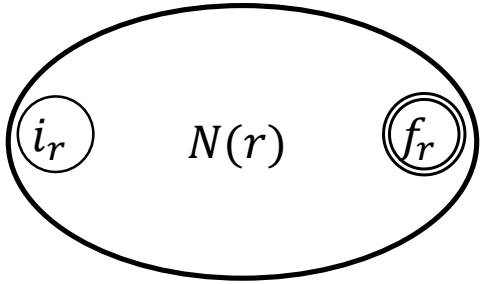
Autómatas Finitos

Paso de una e.r. a un AFN

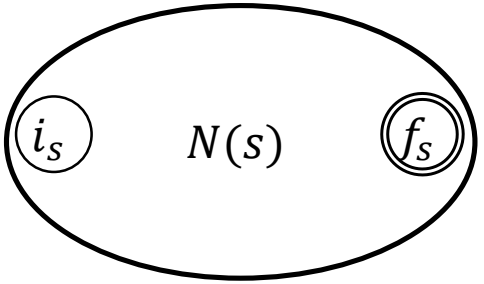
Método de Thompson

Suposición:

Si $\boldsymbol{N(r)}$ y $\boldsymbol{N(s)}$ son los AFNs para las e.r. $\boldsymbol{r}$ y $\boldsymbol{s}$, respectivamente; además $\boldsymbol{L(r)}$ y $\boldsymbol{L(s)}$ son los lenguajes que reconocen respectivamente estas e.r.



Reconoce a $L(r)$



Reconoce a $L(s)$

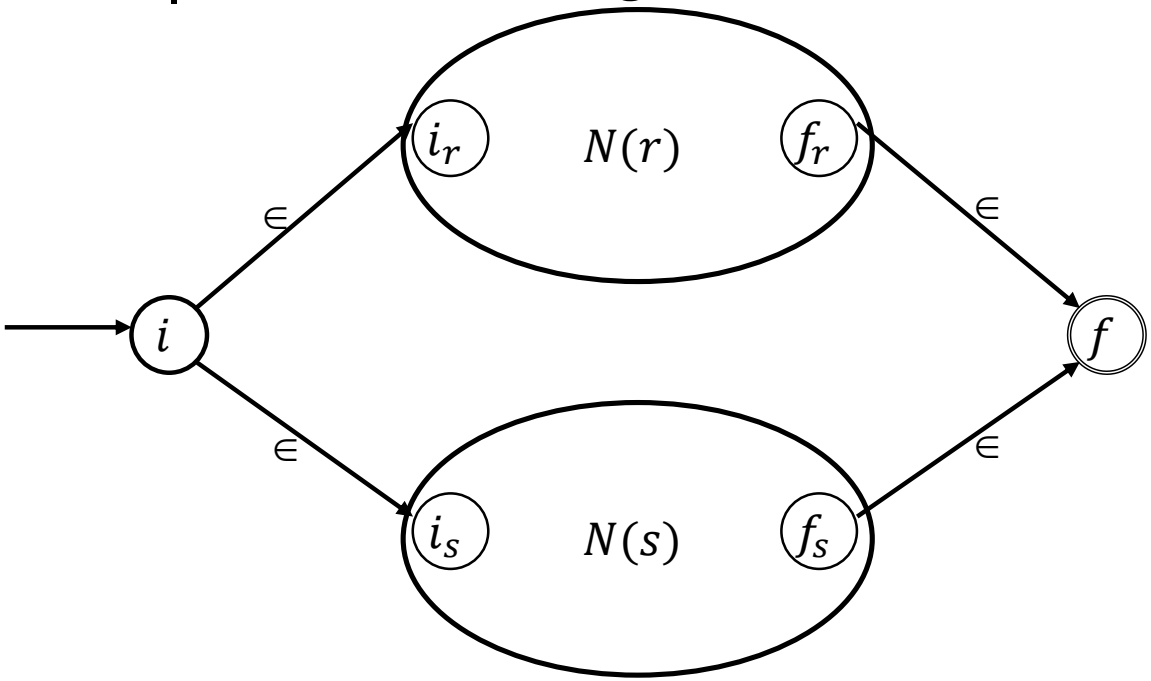
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Paso Inductivo:

3. Para la e.r. $r|s$, el AFN es el siguiente:



Este AFN reconoce $L(r|s) = L(r) \cup L(s)$

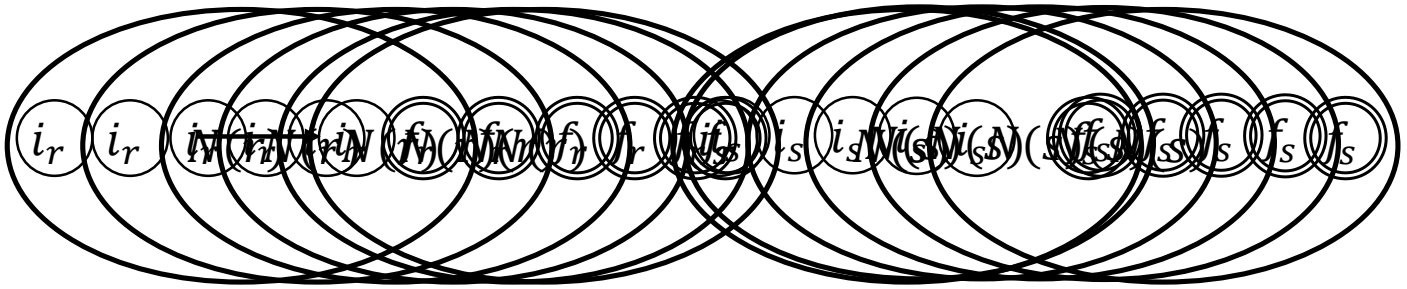
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Paso Inductivo:

4. Para la e.r. rs, el AFN es el siguiente:



Este AFN reconoce $L(rs) = L(r)L(s)$

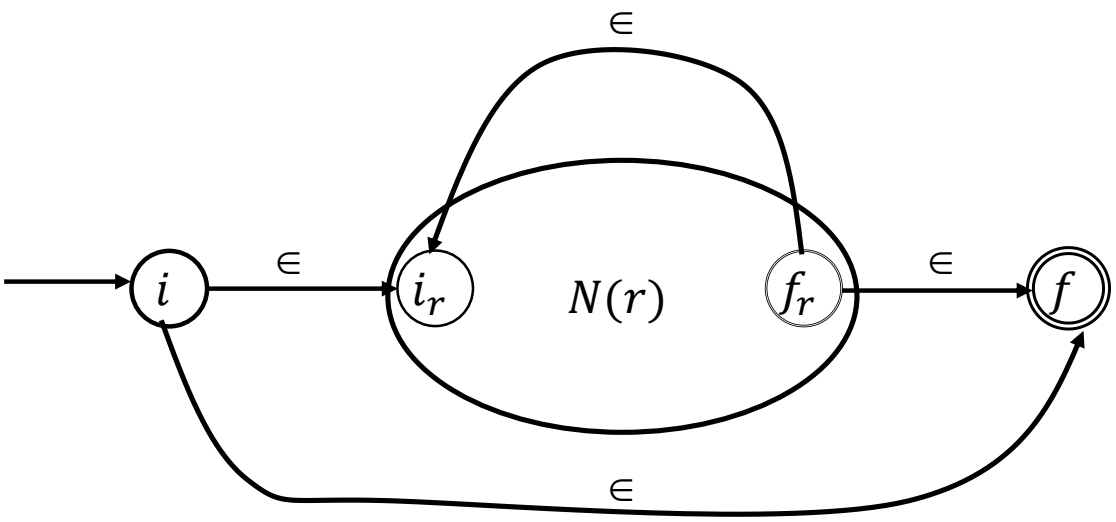
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Paso Inductivo:

5. Para la e.r. r^*, el AFN es el siguiente:



Este AFN reconoce $L(r^*) = (L(r))^* = L^*\ (r)$

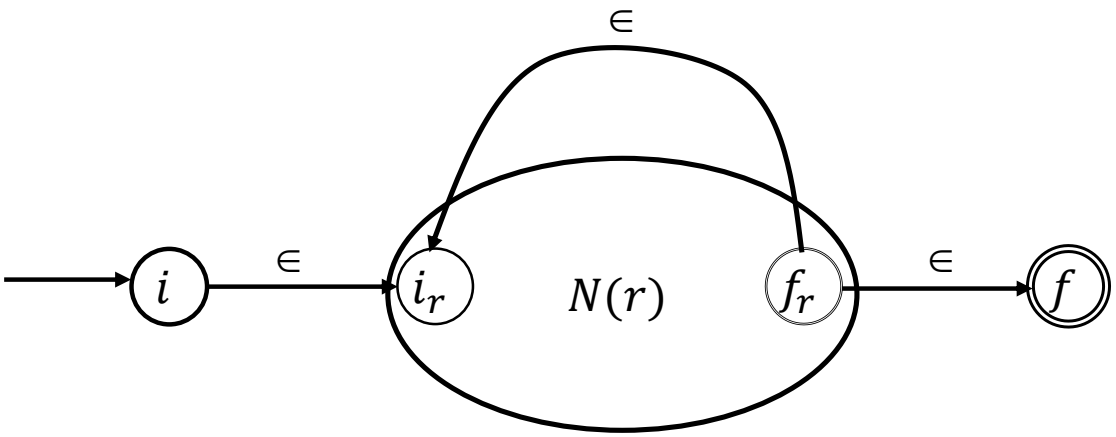
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Paso Inductivo:

6. Para la e.r. r^+, el AFN es el siguiente:



Este AFN reconoce $L(r^+) = (L(r))^+ = L^+\ (r)$

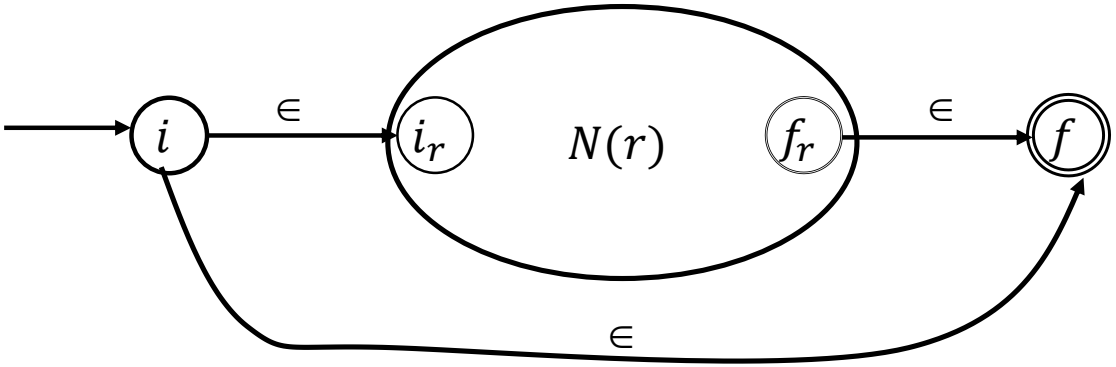
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Paso Inductivo:

7. Para la e.r. $r?$, el AFN es el siguiente:



Este AFN reconoce $L(r?) = L(r)?$

Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Ejemplos:

Construir el AFN para las e.r.

1. $(\boldsymbol{a}|\boldsymbol{b})^*\boldsymbol{abb}$
2. $\boldsymbol{a}(\boldsymbol{a}|\boldsymbol{b})^*\boldsymbol{bb}$

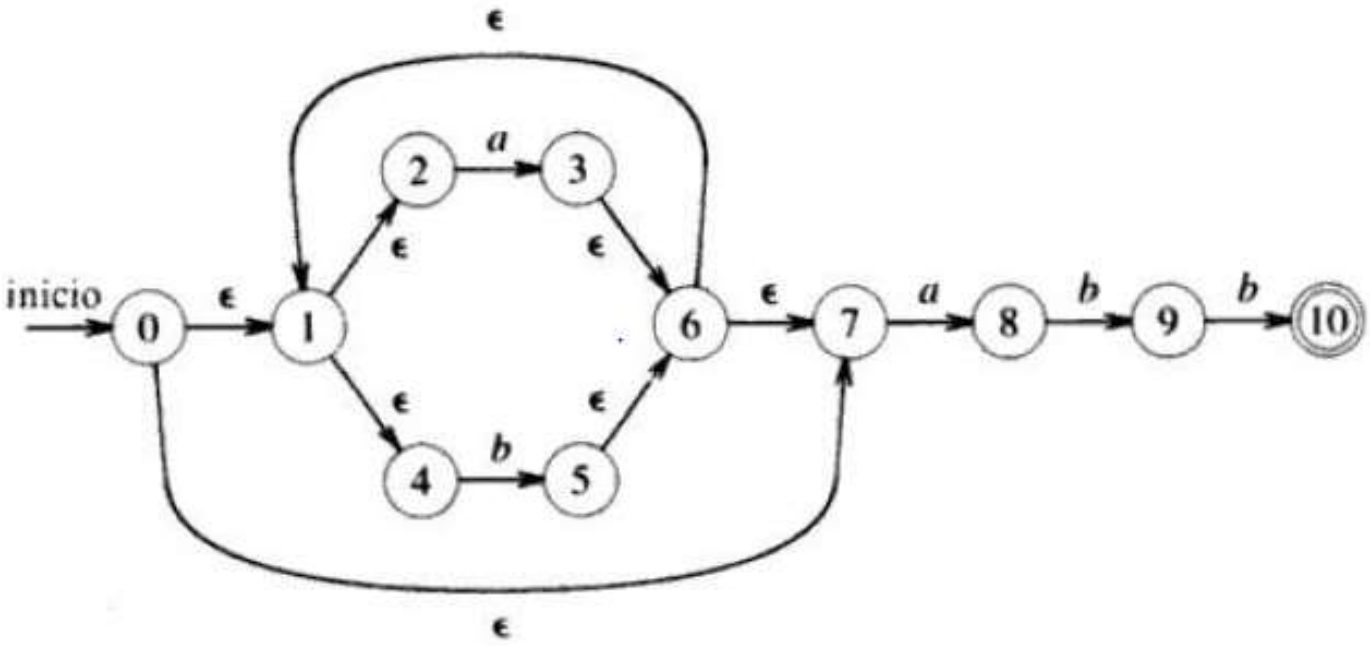
Autómatas Finitos

Paso de una e.r. a un AFN

Método de Thompson

Ejemplos:

1. AFN para $(a|b)^*abb$



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Entrada: Un AFN N que acepta el lenguaje $L(N)$.

Salida: Un AFD D que acepte el mismo lenguaje.

Autómatas Finitos

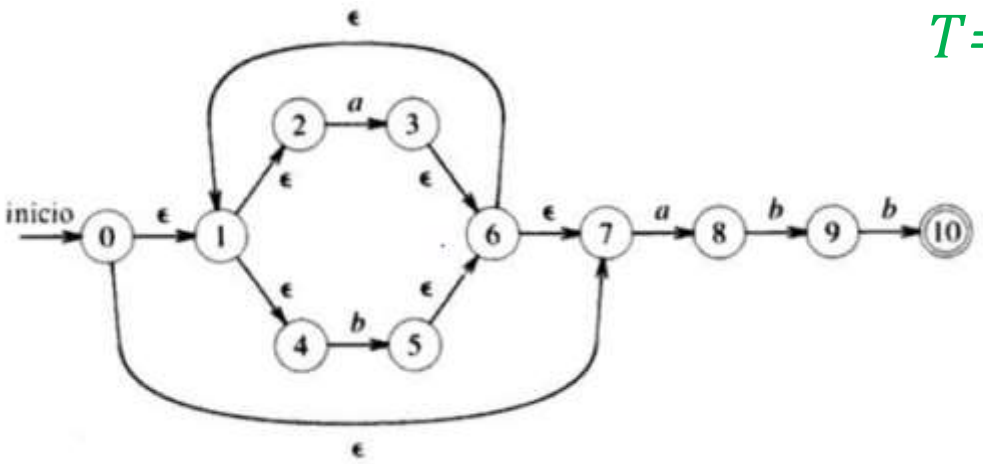
Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Operaciones sobre los estados del AFN.

cerradura-$\in(s)$: Conjunto de estados del AFN alcanzables desde el estado s del AFN con transiciones $\in$ solamente.

Ejemplo:



T=*cerradura-*$\in(0)=\{0,1,2,4,7\}$

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Operaciones sobre los estados del AFN.

$mueve(T, a)$: Conjunto de estados del AFN hacia los cuales hay una transición con el símbolo de entrada a desde algún estado s en T del AFN.

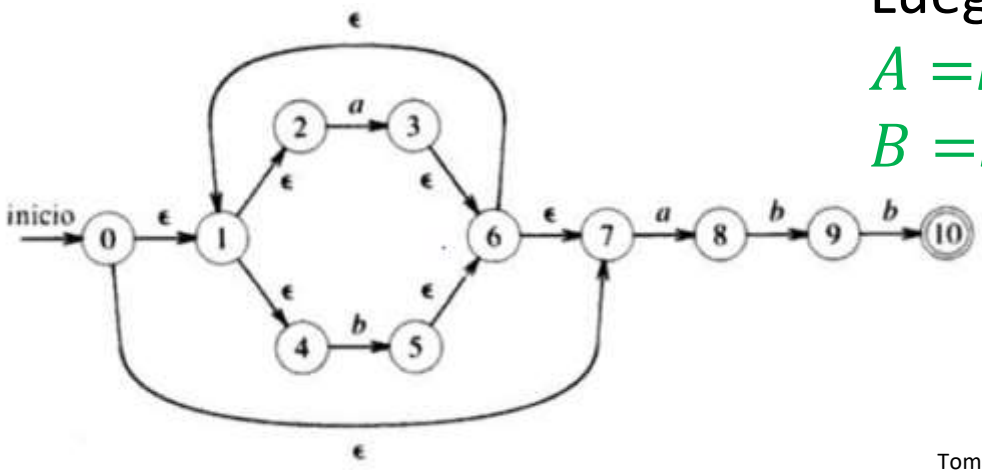
Ejemplo:

Si T=cerradura-$\in(0)$={0,1,2,4,7}

Luego

$A = mueve(T, a)$ = {3,8}

$B = mueve(T, b)$ = {5}



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Operaciones sobre los estados del AFN.

cerradura-$\in(T)$: Conjunto de estados del AFN alcanzables desde algún estado s en T con transiciones $\in$ solamente.

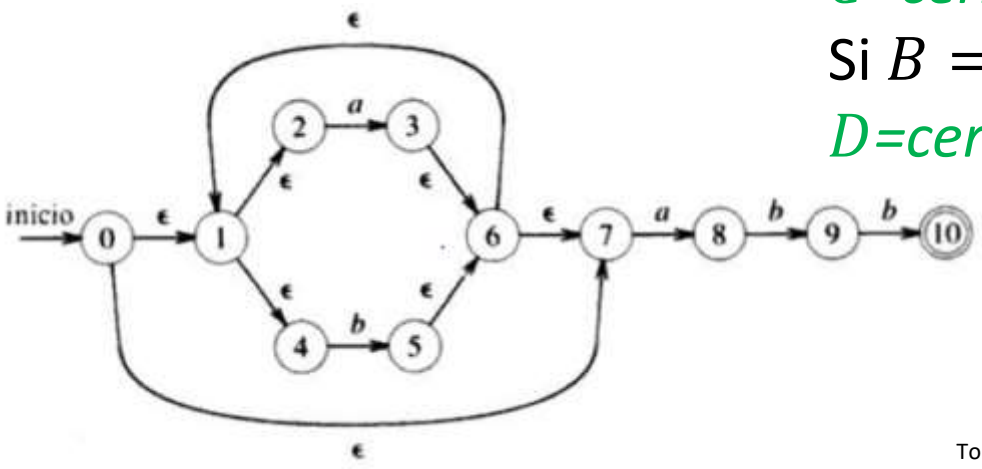
Ejemplo:

Si A =*mueve*(T, a) = {3,8}

C=cerradura-$\in(A)$={3,6,1,7,2,4,8}

Si B =*mueve*(T, b) = {5}

D=cerradura-$\in(B)$={5,6,1,7,2,4}



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

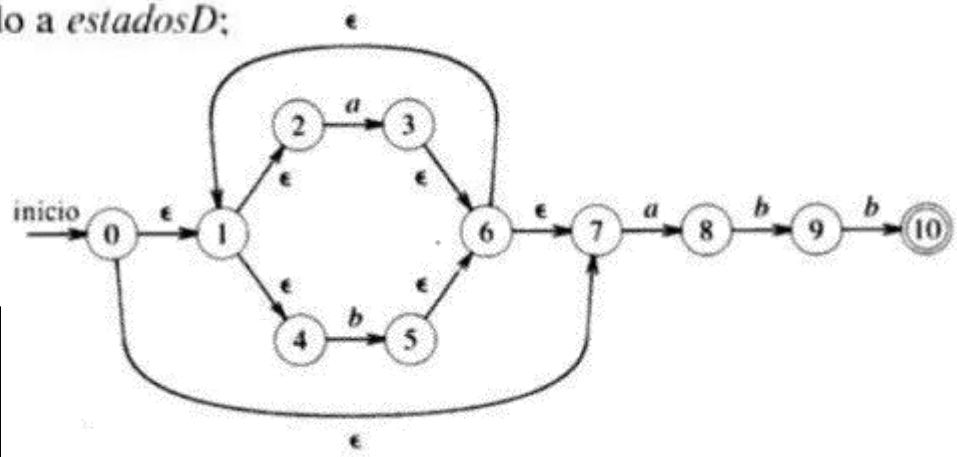
Autómatas Finitos

Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Algoritmo de Subconjuntos

al inicio, *cerradura-ϵ* (s_0) es el único estado dentro de *estadosD* y no está marcado;
while haya un estado no marcado T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        $U :=$ *cerradura-ϵ* (*mueve* (T, a));
        if U no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD [T, a] := U
    end
end



Estado	Símbolos	
	a	*b*

tranD

Estado		
Marca		

estadosD

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

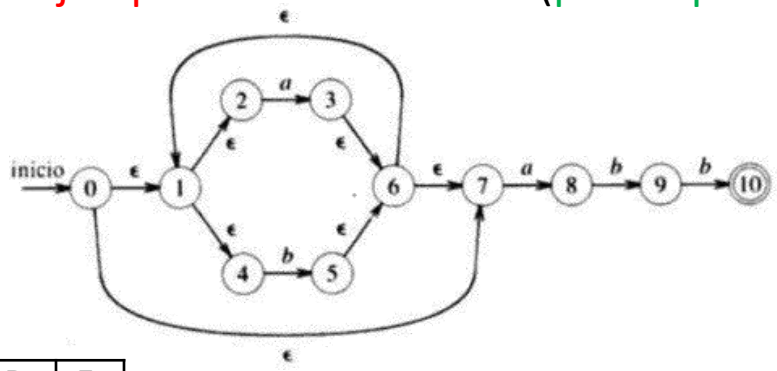
Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Algoritmo de Subconjuntos

al inicio, *cerradura-ϵ* (s_0) es el único estado dentro de *estadosD* y no está marcado;
while haya un estado no marcado T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        $U :=$ *cerradura-ϵ* (*mueve* (T, a));
        if U no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD [T, a] := U
    end
end

Ejemplo: Hallar el AFD de (paso a paso):



Estado	A	B	C	D	E
Marca	x	x	x	x	x

estadosD

***A*=*cerradura*-∈(0)={0,1,2,4,7}**

***T*=*A*, *a*=*a*, *U*=*B* =*cerradura*-∈(mueve(*A*,*a*))=*cerradura*-∈({3,8})={1,2,3,4,6,7,8}**
***T*=*A*, *a*=*b*, *U*=*C* =*cerradura*-∈(mueve(*A*,*b*))=*cerradura*-∈({5})= {1,2,4,5,6,7}**
***T*=*B*, *a*=*a*, *U*=*B* =*cerradura*-∈(mueve(*B*,*a*))=*cerradura*-∈({3,8})**
***T*=*B*, *a*=*b*, *U*=*D* =*cerradura*-∈(mueve(*B*,*b*))=*cerradura*-∈({5,9})={1,2,4,5,6,7,9}**
***T*=*C*, *a*=*a*, *U*=*B* =*cerradura*-∈(mueve(*C*,*a*))=*cerradura*-∈({3,8})**
***T*=*C*, *a*=*b*, *U*=*C* =*cerradura*-∈(mueve(*C*,*b*))=*cerradura*-∈({5})**
***T*=*D*, *a*=*a*, *U*=*B* =*cerradura*-∈(mueve(*D*,*a*))=*cerradura*-∈({3,8})**
***T*=*D*, *a*=*b*, *U*=*E* =*cerradura*-∈(mueve(*D*,*b*))=*cerradura*-∈({5,10})={1,2,4,5,6,7,10}**
***T*=*E*, *a*=*a*, *U*=*B* =*cerradura*-∈(mueve(*E*,*a*))=*cerradura*-∈({3,8})**
***T*=*E*, *a*=*b*, *U*=*C* =*cerradura*-∈(mueve(*E*,*b*))=*cerradura*-∈({5})**

Estado	Símbolos	
	a	*b*
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

tranD

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

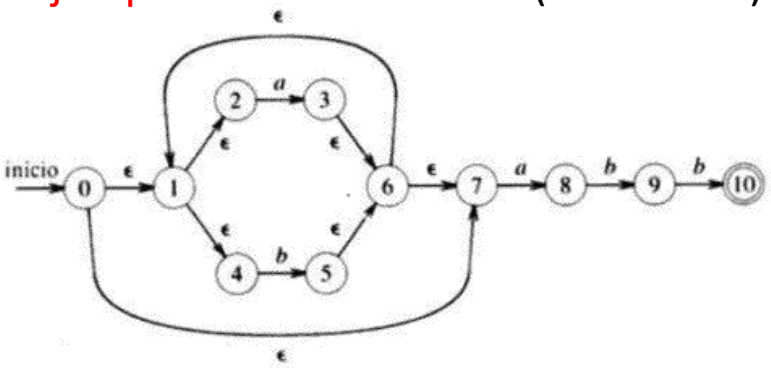
Paso de una AFN a un AFD-No Óptimo

Método de Subconjuntos

Algoritmo de Subconjuntos

al inicio, *cerradura-ϵ* (s_0) es el único estado dentro de *estadosD* y no está marcado;
while haya un estado no marcado T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        $U :=$ *cerradura-ϵ* (*mueve* (T, a));
        if U no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD [T, a] := U
    end
end

Ejemplo: Hallar el AFD de (resultados):



A=cerradura-∈(0)={0,1,2,4,7}
B=cerradura-∈(mueve(A,a))=cerradura-∈({3,8})={1,2,3,4,6,7,8}
C=cerradura-∈(mueve(A,b))=cerradura-∈({5})={1,2,4,5,6,7}
B=cerradura-∈(mueve(B,a))=cerradura-∈({3,8})
D=cerradura-∈(mueve(B,b))=cerradura-∈({5,9})={1,2,4,5,6,7,9}
B=cerradura-∈(mueve(C,a))=cerradura-∈({3,8})
C=cerradura-∈(mueve(C,b))=cerradura-∈({5})
B=cerradura-∈(mueve(D,a))=cerradura-∈({3,8})
E=cerradura-∈(mueve(D,b))=cerradura-∈({5,10})={1,2,4,5,6,7,10}
B=cerradura-∈(mueve(E,a))=cerradura-∈({3,8})
C=cerradura-∈(mueve(E,b))=cerradura-∈({5})

Estado	Símbolos	
	a	*b*
→ **A**	B	C
B	B	D
C	B	C
D	B	E
***E**	B	C

tranD

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFN a un AFD-No Óptimo

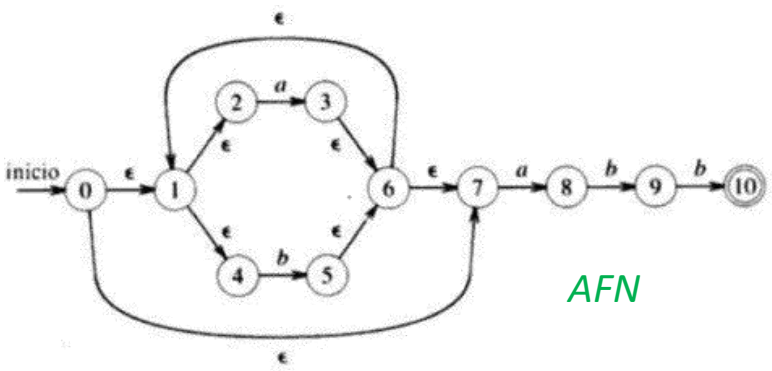
Método de Subconjuntos

Algoritmo de Subconjuntos

***A*=*cerradura*-∈(0)={0,1,2,4,7}**
***B*=*cerradura*-∈(mueve(*A*,*a*))=*cerradura*-∈({3,8})={1,2,3,4,6,7,8}**
***C*=*cerradura*-∈(mueve(*A*,*b*))=*cerradura*-∈({5})={1,2,4,5,6,7}**
***D*=*cerradura*-∈(mueve(*B*,*b*))=*cerradura*-∈({5,9})={1,2,4,5,6,7,9}**
***E*=*cerradura*-∈(mueve(*D*,*b*))=*cerradura*-∈({5,10})={1,2,4,5,6,7,10}**

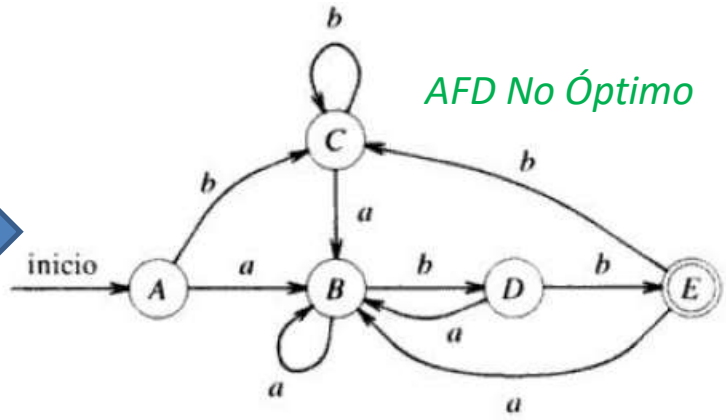
Estado	Símbolos	
	a	*b*
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

tranD



AFN

Subconjuntos



AFD No Óptimo

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFD No Óptimo a un AFD Óptimo

Método de Estados Significativos

Entrada: Un AFD D No Óptimo que acepta el lenguaje $L(D)$.

Salida: Un AFD D Óptimo que acepte el mismo lenguaje $L(D)$.

Autómatas Finitos

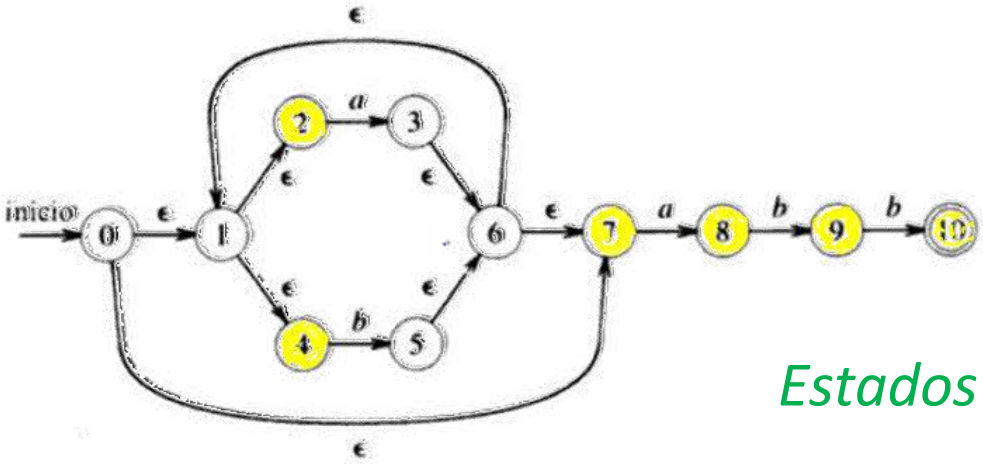
Paso de una AFD No Óptimo a un AFD Óptimo

Método de Estados Significativos

Estado Significativo: Es un estado de un AFN que tiene transiciones de salida diferentes de $\in$.

El *mueve*$(s, a) \neq \emptyset$, si s es significativo para algún a en Σ.

Ejemplo: En el *AFN,* cuáles son los estados significativos?



Estados Significativos={2,4,7,8,9,10}

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una AFD No Óptimo a un AFD Óptimo

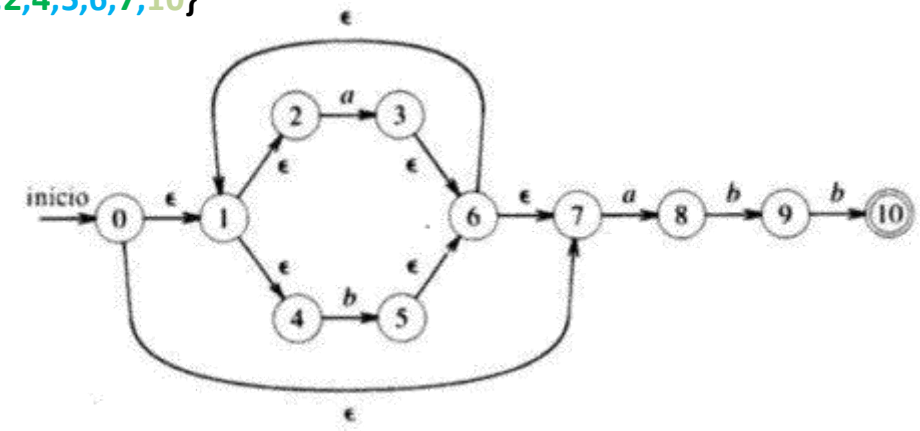
Método de Estados Significativos

En el algoritmo de subconjuntos, el *mueve*(*T*,*a*) utiliza los estados significativos en el subconjunto *T* para hallar los estados alcanzables desde *T*.

***A*=*cerradura*-∈(0)={0,1,2,4,7}**
***B*=*cerradura*-∈(mueve(*A*,*a*))=*cerradura*-∈({3,8})={1,2,3,4,6,7,8}**
***C*=*cerradura*-∈(mueve(*A*,*b*))=*cerradura*-∈({5})={1,2,4,5,6,7}**
***D*=*cerradura*-∈(mueve(*B*,*b*))=*cerradura*-∈({5,9})={1,2,4,5,6,7,9}**
***E*=*cerradura*-∈(mueve(*D*,*b*))=*cerradura*-∈({5,10})={1,2,4,5,6,7,10}**

Ejemplo:

mueve(*A*,*a*)=mueve({0,1,2,4,7},*a*)
mueve(*A*,*b*)=mueve({0,1,2,4,7},*b*)



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

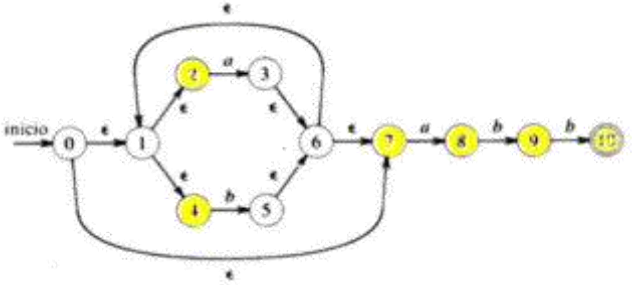
Autómatas Finitos

Paso de una AFD No Óptimo a un AFD Óptimo

Método de Estados Significativos

Durante la construcción, si dos subconjuntos tienen los mismos estados significativos, se identifican.

A=cerradura-∈(0)={0,1,2,4,7}
B=cerradura-∈(mueve(*A,a*))=*cerradura-*∈({3,8})={1,2,3,4,6,7,8}
C=cerradura-∈(mueve(*A,b*))=*cerradura-*∈({5})={1,2,4,5,6,7}
D=cerradura-∈(mueve(*B,b*))=*cerradura-*∈({5,9})={1,2,4,5,6,7,9}
E=cerradura-∈(mueve(*D,b*))=*cerradura-*∈({5,10})={1,2,4,5,6,7,10}



Ejemplo:

Est_Sig(*A*)={2,4,7}
Est_Sig(*B*)={2,4,7,8}
Est_Sig(*C*)={2,4,7}
Est_Sig(*D*)={2,4,7,9}
Est_Sig(*E*)={2,4,7,10}

Luego, los estados ***A*** y ***C*** se identifican

Estado	Símbolos	
	a	*b*
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Significativos

Estado	Símbolos	
	a	*b*
→A	B	A
B	B	D
D	B	E
*E	B	A

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

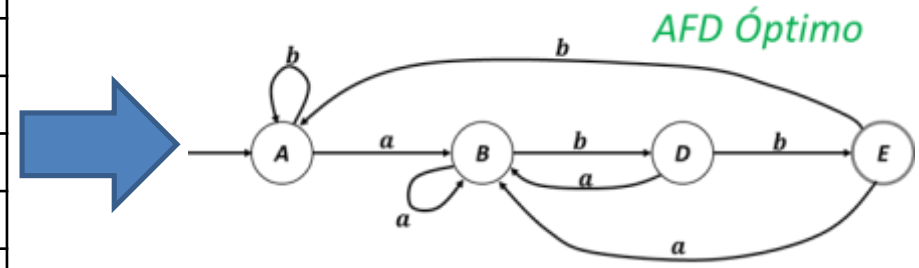
Paso de una AFD No Óptimo a un AFD Óptimo

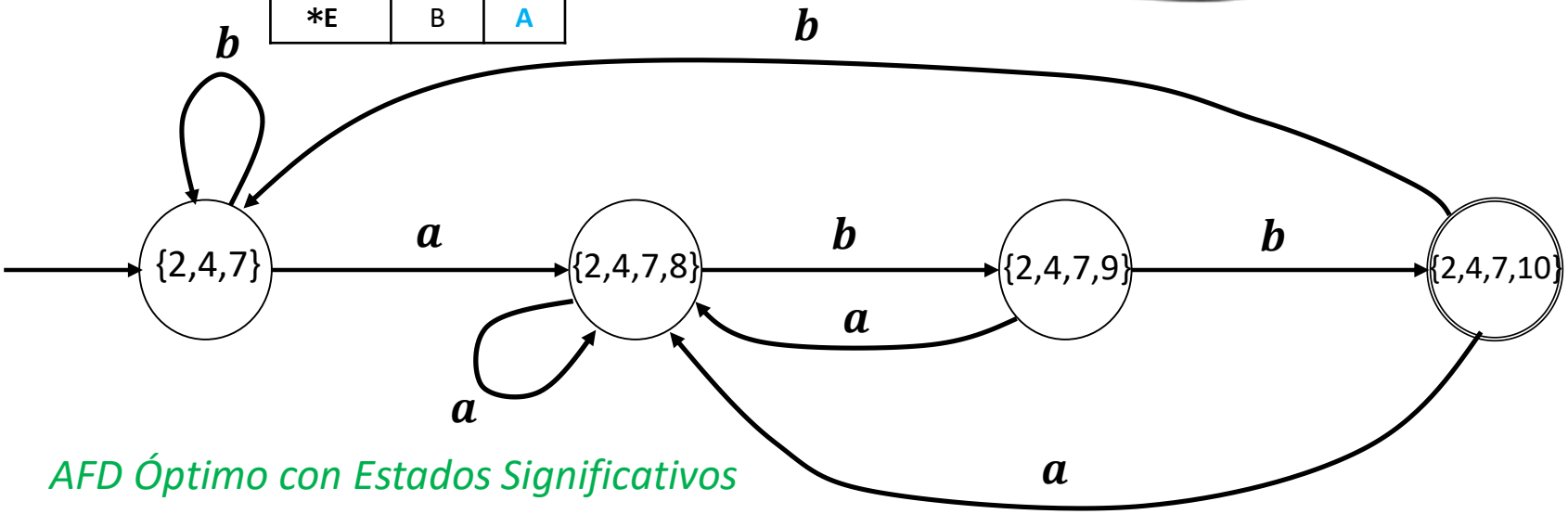
Método de Estados Significativos

AFD Óptimo con Estados Significativos

tranD

Estado	Símbolos	
	a	*b*
→A	B	A
B	B	D
D	B	E
*E	B	A



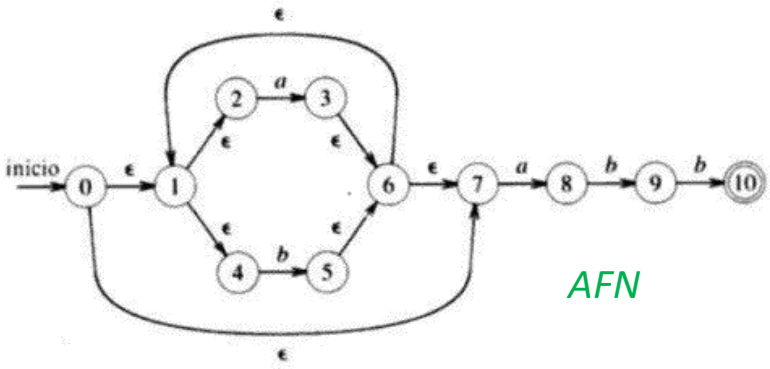


AFD Óptimo con Estados Significativos

Autómatas Finitos

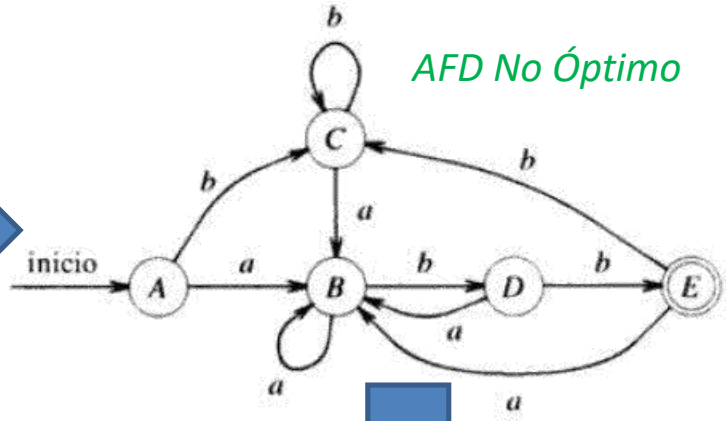
Resumen de Métodos

Métodos de Thompson, Subconjutos y Estados Significativos



AFN

Subconjuntos



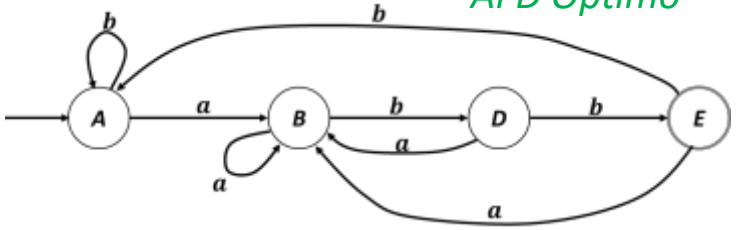
AFD No Óptimo

Sig nifi cat ivo

Estado	Símbolos	
	a	*b*
→**A**	B	C
B	B	D
C	B	C
D	B	E
***E**	B	C

Significativos

Estado	Símbolos	
	a	*b*
→**A**	B	A
B	B	D
D	B	E
***E**	B	A



AFD Óptimo

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Entrada: Una e.r. r que acepta el lenguaje $L(r)$.

Salida: Un AFD D Óptimo que acepte el mismo lenguaje.

Método:

1. Construir un árbol sintáctico $\boldsymbol{T}$ para la e.r. aumentada $r\#$.

2. Construir las funciones $anulable, primerapos, ultimapos$ y $siguientepos$ haciendo recorridos de abajo-arriba en $\boldsymbol{T}$.

3. Construir el conjunto de estados de D, $estadosD$, y la tabla de transiciones para D, $tranD$.

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

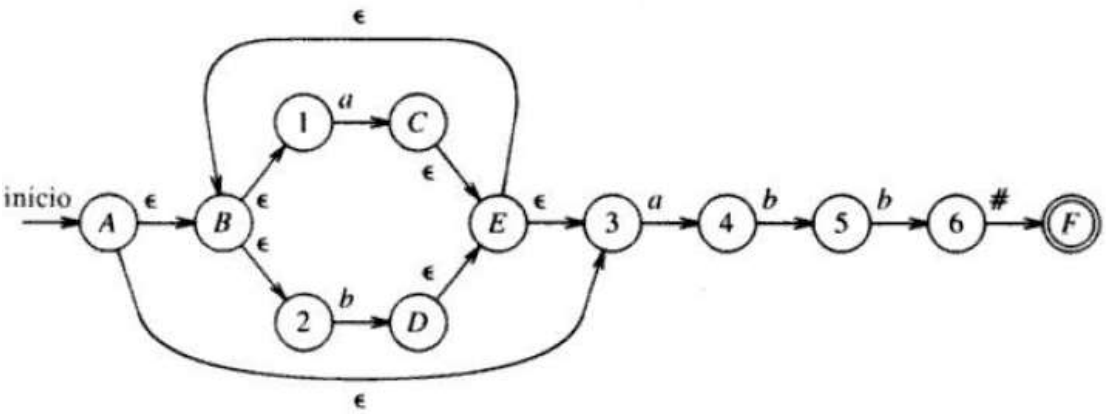
La construcción de Thompson genera un estado significativo cuando un símbolo del alfabeto aparece en la e.r.

La construcción relaciona los estados significativos del AFN con los símbolos de la e.r.

El AFN tiene un solo estado de aceptación, pero no es significativo porque no tiene transiciones de salida.

Se concatena el marcador # a la e.r. r, y se da al estado de aceptación de r una transición en #, convirtiéndolo en estado significativo del AFN para $r\#$.

Se genera una e.r. aumentada r#. Cuando se realiza el reconocimiento, y la construcción está completa, cualquier estado del AFN con una transición en # debe ser estado de aceptación.



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Una e.r. aumentada $r\#$, se representa mediante un árbol sintáctico con los siguientes tipos de nodos:

Tipos de nodos	**Clase**	**Etiqueta**	**Posición**
Hojas	Símbolo del Σ	Símbolo	$i \; está \; en \; \mathbb{N}$
	$\in$	$\in$	-
Interiores	nodo-o	$\mid$	-
	nodo-cat	$\cdot$	-
	nodo-ast	$*$	-

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

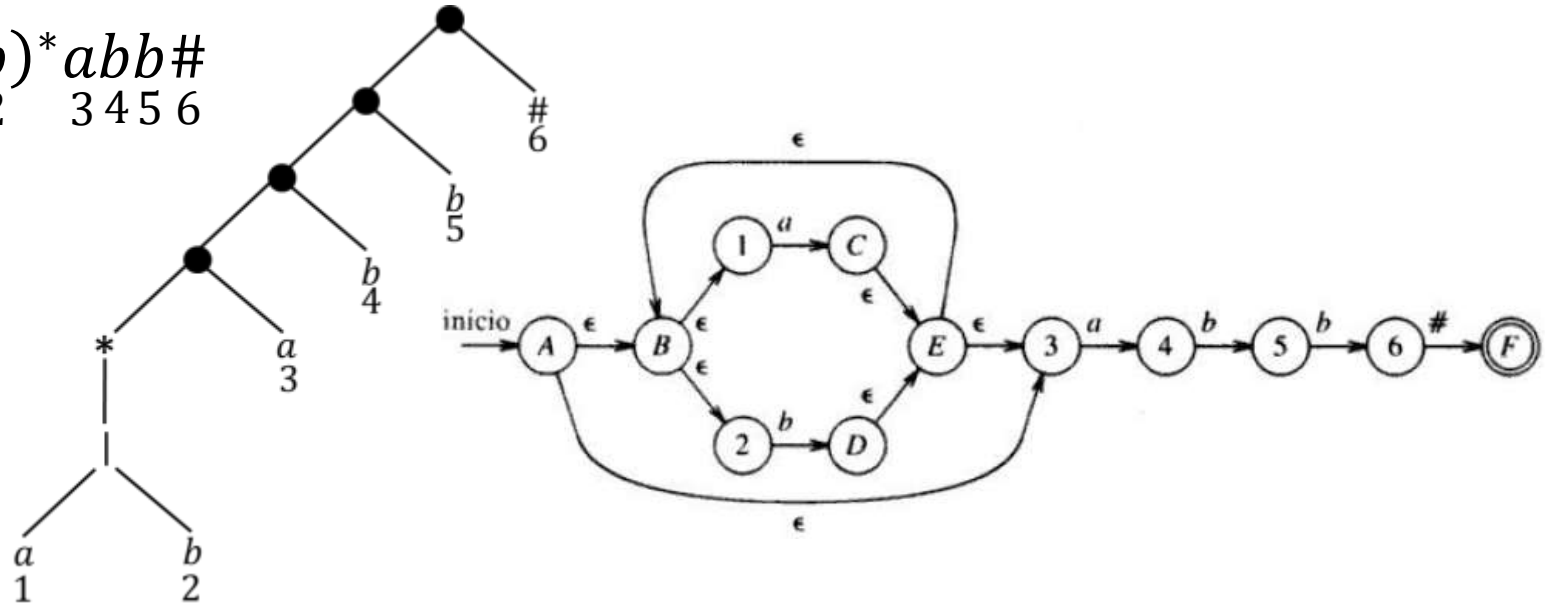
Método del Árbol Sintáctico

A cada hoja no etiquetada con $\in$ se le asocia una posición, que es un entero.

Un símbolo repetido en una e.r. tiene varias posiciones.

La posición de cada símbolo se puede identificar con la etiqueta del estado significativo en el AFN, como se observa en la figura.

$$r\# = \underset{1}{(}\underset{1}{a}\underset{2}{|}\underset{2}{b}\underset{3}{)^*}\underset{3}{a}\underset{4}{b}\underset{5}{b}\underset{6}{\#}$$



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

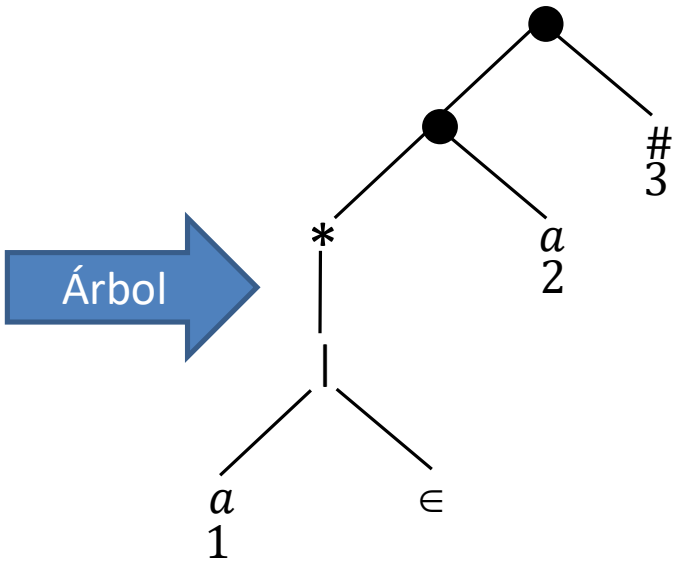
Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Ejemplo:

Para la e.r. $(a|\in)^*a$, construir el árbol sintáctico, y señalar las posiciones tanto en la e.r. como en el árbol:

$$r\# = \underset{1}{(a}|\in)^*\underset{2}{a}\underset{3}{\#}$$



Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Funciones:

$anulable(n)$: Función booleana sobre un nodo n. El anulable será Verdadero si al aplicarlo sobre el nodo n, existe la posibilidad de generar la cadena vacía $\in$ en algún momento; o Falso si nunca la genera.

Ejemplo:

El *anulable* de las siguientes e.r. es:

$(a|\in)^+ b?$

$a?\, b^+$

$b^* b?\, a^*$

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Funciones:

$primerapos(n)$: Función sobre un nodo n que devuelve el conjunto de posiciones que pueden concordar con el primer símbolo de una cadena generada por la sub-expresión regular con raíz en n.

Ejemplo:

El *primerapos* en las siguientes e.r. es:

$(a|b)^* abb$

$a?\, b^+ ab^*$

$b^* b?\, a^*$

$ab^+(a|b)^*$

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Funciones:

$ultimapos(n)$: Función sobre un nodo n que devuelve el conjunto de posiciones que pueden concordar con el último símbolo de una cadena generada por la sub-expresión regular con raíz en n.

Ejemplo:

El *ultimapos* en las siguientes e.r. es:

$(a|b)^* abb$

$a? \, b^+ ab^*$

$b^* b? \, a^*$

$ab^+ (a|b)^*$

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Funciones:

$siguientepos(i)$: Si i es una posición, $siguientepos(i)$ es el conjunto de posiciones j tales que hay alguna cadena de entrada … cd … tal que i corresponde a la aparición de c y j a la aparición de d.

Ejemplo:

El *siguientepos* en las siguientes e.r. es:

$(a|b)^*abb$

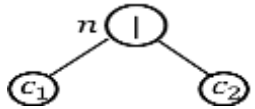
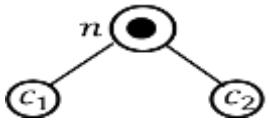
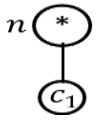
$ab^+(a|b)^*$

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Reglas para calcular las funciones en los nodos

Nodo n	$\boldsymbol{anulable}(n)$	$\boldsymbol{primerapos}(n)$	$\boldsymbol{ultimapos}(n)$
n es un nodo hoja etiquetado con $\in$	**true**	$\emptyset$	$\emptyset$
n es una hoja etiquetada con posición i	**false**	$\{i\}$	$\{i\}$
	$anulable(c_1)$ **or** $anulable(c_2)$	$primerapos(c_1) \cup primerapos(c_2)$	$ultimapos(c_1) \cup ultimapos(c_2)$
	$anulable(c_1)$ **and** $anulable(c_2)$	$if\ anulable(c_1)\ then\ ppos(c_1) \cup ppos(c_2)\ else\ ppos(c_1)$	$if\ anulable(c_2)\ then\ upos(c_1) \cup upos(c_2)\ else\ upos(c_2)$
	**true**	$primerapos(c_1)$	$ultimapos(c_1)$

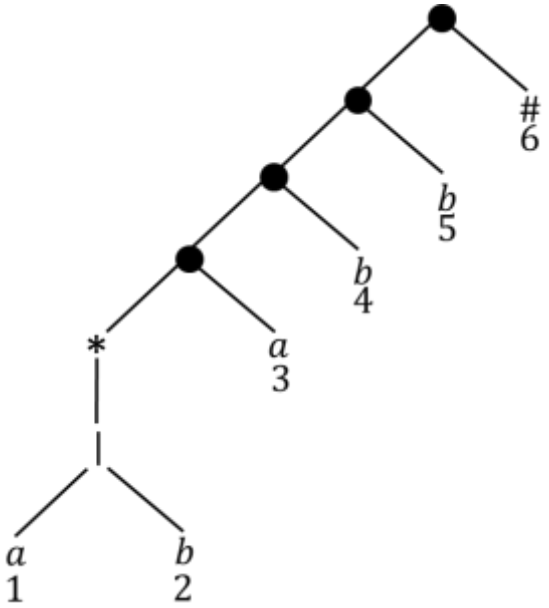
Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

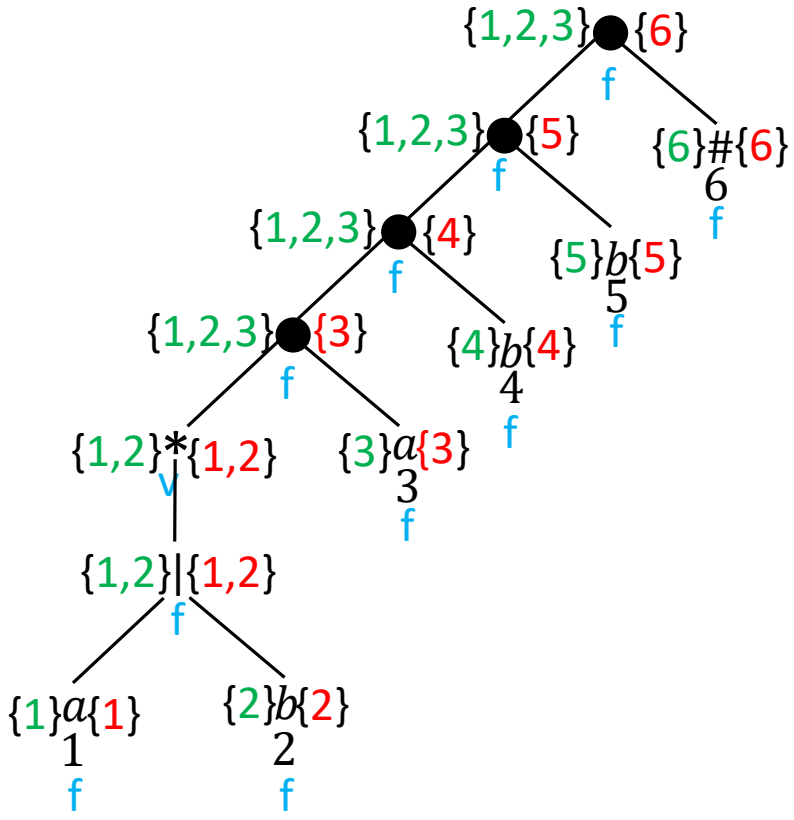
Método del Árbol Sintáctico

Ejemplo:

$anulable, primerapos\ y\ ultimapos$ para $(a|b)^*abb\#$



Árbol Sintáctico Inicial



anulable, *primerapos*, *ultimapos*

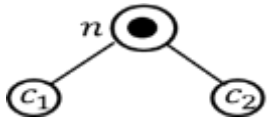
Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Cálculo del $siguientepos$:

1. Si n es un nodo-cat con hijo izquierdo c_1e hijo derecho c_2, e i es una posición dentro de $ultimapos(c_1)$, entonces todas las posiciones de $primerapos(c_2)$ están en el $siguientepos(i)$.



$$c_1 \cdot c_2$$

$pp(c_1)$ $up(c_1)$ $pp(c_2)$ $up(c_2)$

$\{i, j, \dots\}$ $\{k, l, \dots\}$

$spos(i)$ *contiene a* $\{k, l, \dots\}$

$spos(j)$ *contiene a* $\{k, l, \dots\}$

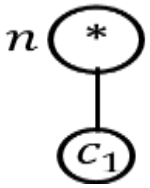
Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Cálculo del $siguientepos$:

2. Si n es un nodo-ast con hijo c_1, e i es una posición dentro de $ultimapos(n)$, entonces todas las posiciones de $primerapos(n)$ están en el $siguientepos(i)$.



$$\bullet \; c_1 \bullet c_1 \bullet$$

$pp(c_1)$ $up(c_1)$ $pp(c_1)$ $up(c_1)$

$up(c_1)$: $\{i, j, \dots\}$

$pp(c_1)$: $\{k, l, \dots\}$

$spos(i)$ *contiene a* $\{k, l, \dots\}$

$spos(j)$ *contiene a* $\{k, l, \dots\}$

Autómatas Finitos

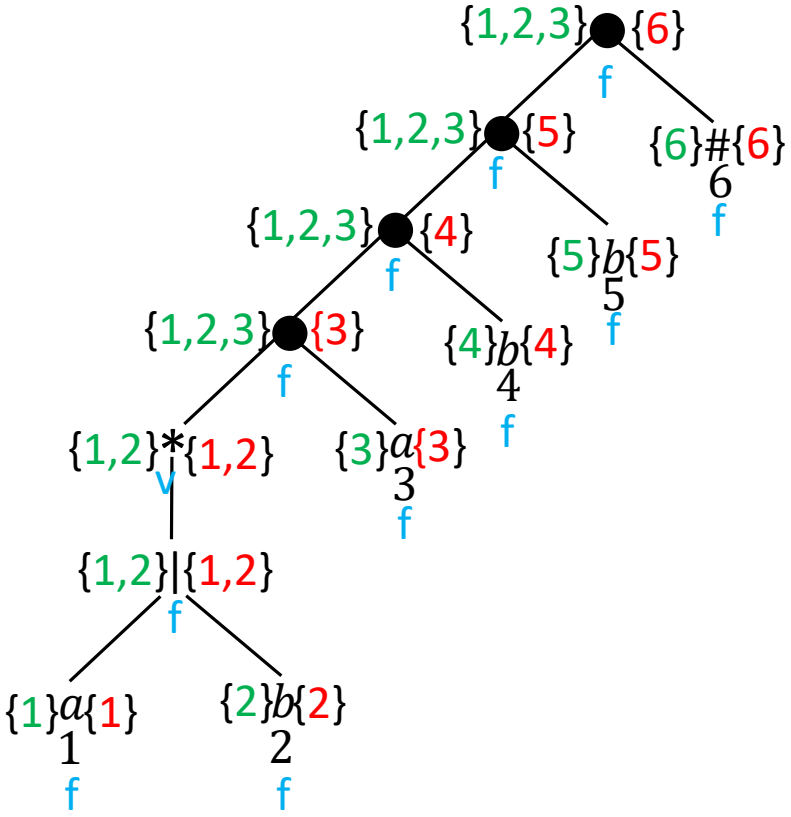
Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

Cálculo del $siguientepos$:

Dado que ya se calcularon el $primerapos$ y $ultimapos$ en el árbol sintáctico para cada nodo, se calcula $siguientepos$ haciendo un recorrido de abajo hacia arriba en el árbol.

Posición i	$siguientepos(i)$
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}



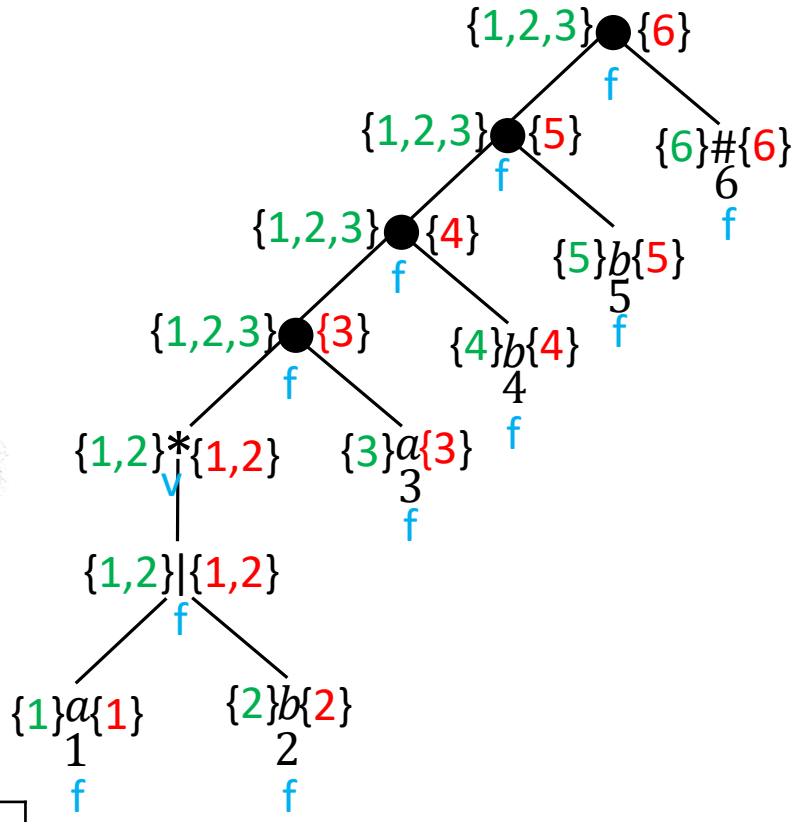
$$r\# = \underset{1}{(}\underset{1}{a}|\underset{2}{b}\underset{2}{)^*}\underset{3}{a}\underset{4}{b}\underset{5}{b}\underset{6}{\#}$$

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

al principio, el único estado no marcado en *estadosD* es
    primerapos (*raíz*), donde *raíz* es la raíz del árbol
    de sintaxis para $(r)\#$;
while hay un estado sin marcar T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        sea U el conjunto de posiciones que están en
            siguientepos (p) para alguna posición p en T, tal
            que el símbolo en la posición p es a;
        if U no está vacío y no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD $[T, a] := U$
    end
end



Estado	Símbolos	
	a	*b*

tranD

Estado		
Marca		

estadosD

$$r\# = \underset{1}{(}\underset{1}{a}\underset{2}{|}\underset{2}{b}\underset{1}{)^*}\underset{3}{a}\underset{4}{b}\underset{5}{b}\underset{6}{\#}$$

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

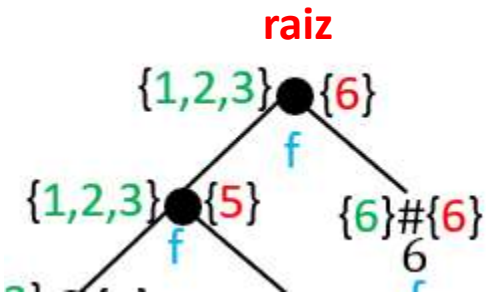
Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

$$r\# = (a|b)^* abb\#$$
$$\quad\;\; 1\;\; 2 \quad\; 3\,4\,5\;6$$

al principio, el único estado no marcado en *estadosD* es
    primerapos (*raíz*), donde *raíz* es la raíz del árbol
    de sintaxis para $(r)\#$;
while hay un estado sin marcar T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        sea U el conjunto de posiciones que están en
            siguientepos (p) para alguna posición p en T, tal
            que el símbolo en la posición p es a;
        if U no está vacío y no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD $[T, a] := U$
    end
end



Posición i	$siguientepos(i)$
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Estado	A	B	C	D	
Marca	x	x	x	x	

estadosD

El estado de inicio del AFD es que corresponde con la $ppos(raíz)$.
El estado de finalización del AFD es el que contiene la posición de #.

***A*=*ppos*(raiz)={1,2,3}**
***T*=A, *a*=a, *U*=*B* =*spos*(1)∪spos(3)={1,2,3,4}**
***T*=A, *a*=b, *U*=*A* =*spos*(2) = {1,2,3}**
***T*=B, *a*=a, *U*=*B* =*spos*(1)∪spos(3)**
***T*=B, *a*=b, *U*=*C* =*spos*(2)∪spos(4)={1,2,3,5}**
***T*=C, *a*=a, *U*=*B* =*spos*(1)∪spos(3)**
***T*=C, *a*=b, *U*=*D* =*spos*(2)∪spos(5)={1,2,3,6}**
***T*=D, *a*=a, *U*=*B* =*spos*(1)∪spos(3)**
***T*=D, *a*=b, *U*=*A* =*spos*(2)**

Estado	Símbolos	
	a	b
A	B	A
B	B	C
C	B	D
D	B	A

tranD

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

Método del Árbol Sintáctico

$$r\# = \underset{1}{(}\underset{2}{a}|\underset{3}{b})^{*}\underset{4}{a}\underset{5}{b}\underset{6}{b}\#$$

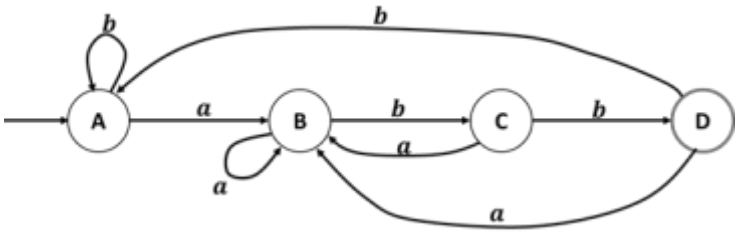
Posición i	$siguientepos(i)$
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

al principio, el único estado no marcado en *estadosD* es
    primerapos (*raíz*), donde *raíz* es la raíz del árbol
    de sintaxis para $(r)\#$;
while hay un estado sin marcar T en *estadosD* **do begin**
    marcar T;
    for cada símbolo de entrada a **do begin**
        sea U el conjunto de posiciones que están en
            siguientepos (p) para alguna posición p en T, tal
            que el símbolo en la posición p es a;
        if U no está vacío y no está en *estadosD* **then**
            añadir U como estado no marcado a *estadosD*;
        tranD $[T, a] := U$
    end
end

Estado	Símbolos	
	a	b
→A	B	A
B	B	C
C	B	D
*D	B	A

tranD

El estado de inicio del AFD es que corresponde con la $ppos(raíz)$.
El estado de finalización del AFD es el que contiene la posición de #.



***A*=*ppos*(raiz)={1,2,3}**
***T*=*A*;*a*=*a*;*U*=*B*=*spos*(1)∪spos(3)={1,2,3,4}**
***T*=*A*;*a*=*b*;*U*=*A*=*spos*(2)={1,2,3}**
***T*=*B*;*a*=*a*;*U*=*B*=*spos*(1)∪spos(3)**
***T*=*B*;*a*=*b*;*U*=*C*=*spos*(2)∪spos(4)={1,2,3,5}**
***T*=*C*;*a*=*a*;*U*=*B*=*spos*(1)∪spos(3)**
***T*=*C*;*a*=*b*;*U*=*D*=*spos*(2)∪spos(5)={1,2,3,6}**
***T*=*D*;*a*=*a*;*U*=*B*=*spos*(1)∪spos(3)**
***T*=*D*;*a*=*b*;*U*=*A*=*spos*(2)**

Tomado de Compiladores. Principios, Técnicas y Herramientas, Aho, Sethi, Ullman. Pearson

Autómatas Finitos

Paso de una e.r. a un AFD Óptimo

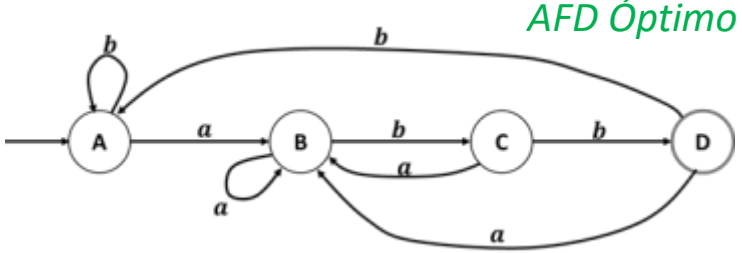
Método del Árbol Sintáctico

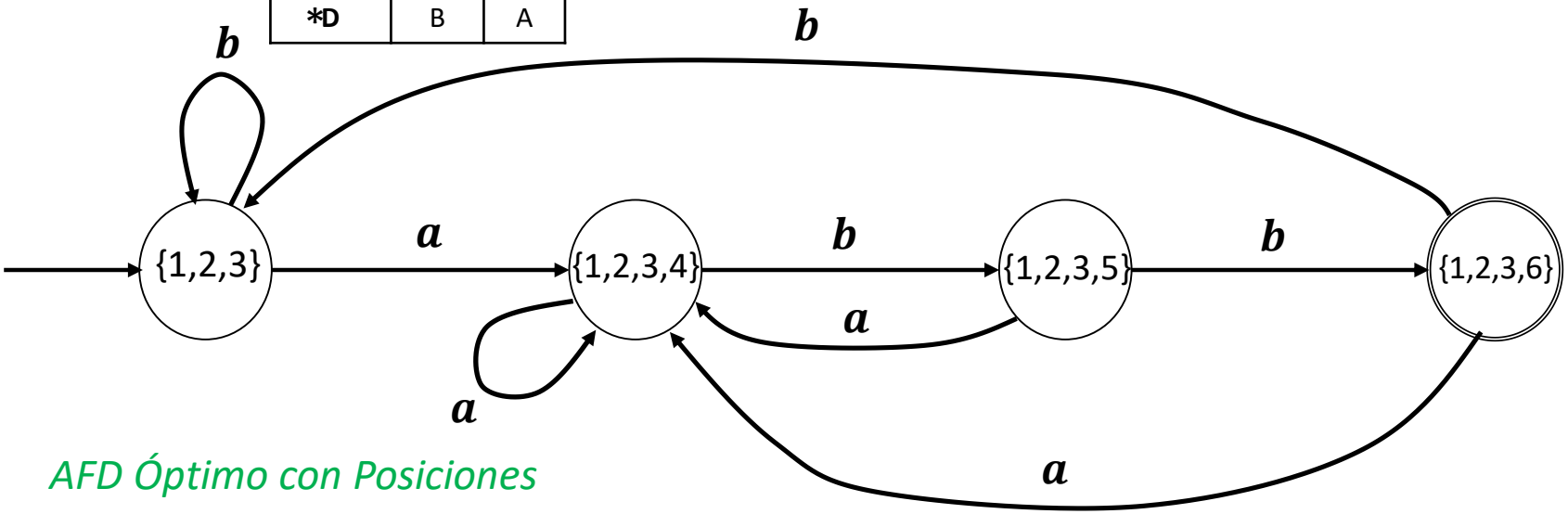
AFD Óptimo con Posiciones(Anterior Estados Significativos)

tranD

Estado	Símbolos	
	a	*b*
→A	B	A
B	B	C
C	B	D
*D	B	A

AFD Óptimo





AFD Óptimo con Posiciones

Autómatas Finitos

Conversión de un AFD a una E.R.

Lema de Ardem

Sistemas de Ecuaciones en E.R.

Ecuación en expresiones regulares: Ecuación lineal donde variables y coeficientes toman la forma de expresiones regulares.

$$X = rX \mid s$$

Entrada: Autómata finito $D = \{\Sigma, S, s_0, F, \delta\}$, con $S = \{s_0, s_1, s_2, \dots, s_n\}$

Salida: Sistema de ecuaciones donde en s_0 determina el lenguaje del AFD

Método:

1. Por cada estado s_i , introducir una variable s_i.

2. Si s_i está en F, luego añadir en la parte derecha de la i-ésima ecuación el término $\in$; es decir, $s_i = \in$.

3. Si $\delta(s_i, a) = s_j$ entonces en la parte derecha de la i-ésima ecuación aparece el término as_j; es decir, $s_i = as_j$, con a en $\Sigma \cup \{\in\}$.

Autómatas Finitos

Conversión de un AFD a una E.R.

Lema de Ardem

- Sea $\boldsymbol{X = rX|s}$ una ecuación en expresiones regulares, luego $\boldsymbol{X = r^{*}s}$ es una solución para la ecuación. Es única si $\in$ es generada por r.

Sistema de Ecuaciones en E.R.

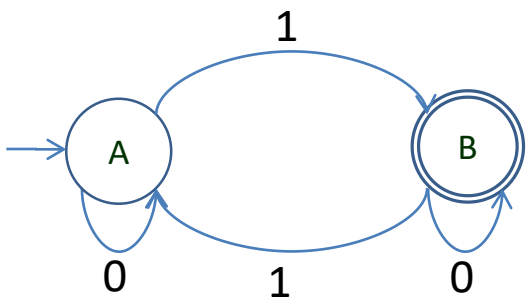
- Dado un sistema de ecuaciones en expresiones regulares, la resolución viene tras aplicar el método de Gauss utilizando el Lemma de Arden para reducir.
- Útil para obtener una e.r. a partir de un A.F. o una gramática regular.

Autómatas Finitos

Conversión de un AFD a una E.R.

Lema de Ardem

Ejemplo 1:



A=1B | 0A (1)

B=1A | 0B | ∈ (2)

Resolviendo

De (1): Aplicando Ardem

A=0*1B (1')

Rempl. (1') en (2)

B=10*1B | 0B | ∈

B= (10*1| 0)B | ∈

B= (10*1 | 0)* ∈ Aplicando Ardem

B= (10*1 | 0)* (2')

Rempl. (2') en (1')

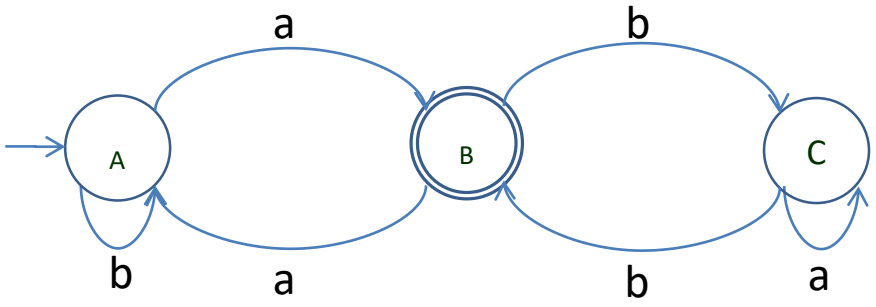
A=0*1 (10*1 | 0)* Expr. Regular buscada

Autómatas Finitos

Conversión de un AFD a una E.R.

Lema de Ardem

Ejemplo 2:



A= bA | aB (1)
B= bC | aA | ∈ (2)
C=aC | bB (3)
Resolviendo
De (1): Por Ardem
A=b*aB (1')
De (3): Por Ardem
C=a*bB (3')
Rempl (1') y (3') en (2)
B=ba*bB | ab*aB | ∈
B=(ba*b | ab*a) B | ∈
B= (ba*b | ab*a)* ∈ Por Ardem
B= (ba*b | ab*a)* (2')
Rempl en (1')
A=b*a (ba*b | ab*a)* Exp. Regular