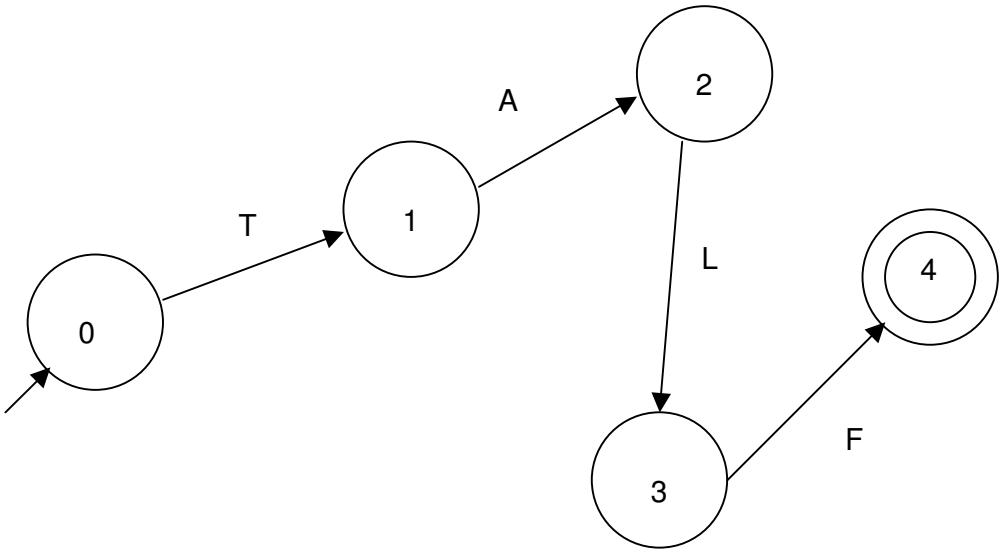
TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES



Breve comentario introductorio

Estos apuntes no quieren (ni mucho menos) sustituir los que impartirá el profesor de la asignatura. Sirven únicamente como una especie de "bastón" a la hora de estudiar la asignatura. Fueron tomados en el Curso 2003/2004 y los he intentado organizar de alguna forma para poder pasarlos a ordenador y posteriormente ponerlo a disposición de la gente. Es muy posible que encuentren más de una errata, estos apuntes fueron tomados por mí en clase, por lo que me he podido equivocar hasta copiando de la pizarra. De ser así ruego me lo comuniquen para corregirlo lo antes posible.

He obviado pasar el tema 0 del curso, que es una introducción al C++, porque en mi opinión no tienen nada que ver con el contenido real de la asignatura, y quiero centrarme estrictamente en el contenido de la materia.

Los títulos en **rojo** anuncian el comienzo de un nuevo tema, que explicaré hasta que aparezca otro título en rojo o se termine el temario.

Los títulos en **azul** corresponden a subapartados de un determinado tema, que también explico a continuación.

Def: corresponde a una definición de algún término o términos, que aparecen subrayados.

Teor: corresponde al enunciado de algún teorema.

Dem: implica el inicio de una demostración.

Agradecimientos a: David Taima Hernández por la notable mejora de los dibujos.

Espero que estos apuntes les sirvan de algo... un saludo.

Cualquier cosa que me quieran comunicar, sean erratas, comentarios, etc., lo pueden hacer a la siguiente dirección de correo: **NKNeumann@gmail.com**

TEMARIO DE LA ASIGNATURA

Tema 1
Conceptos Previos.

Lógica Proposicional.

- Equivalencia $P \equiv Q$ Significa que P es equivalente a Q, siempre que P es cierto $\Rightarrow$ Q es cierto, y siempre que P es falso $\Rightarrow$ Q es falso.

  P. ej: "$3/5$ es entero" $\equiv$ "N es finito" (ambas falsas)

- Negación ($p \rightarrow \neg p$)

Tabla de verdad

p	¬p
F	T
T	F

(F = False, T = True)

- Conjunción ($p \wedge q$)

Tabla de verdad

p	q	p ∧ q
F	F	F
F	T	F
T	F	F
T	T	T

- Disyunción ($p \vee q$)

Tabla de verdad

p	q	p ∨ q
F	F	F
F	T	T
T	F	T
T	T	T

- Condicional ($p \rightarrow q$)

  Cuando hablamos de una proposición condicional, de la forma $p \rightarrow q$, a "p" se le llama hipótesis o antecedente, y a "q" se le llama consecuente o conclusión.

Tabla de verdad

p	q	p → q
F	F	T
F	T	T
T	F	F
T	T	T

- Recíproca $(q \rightarrow p)$

  La tabla de verdad se saca fácilmente viendo el ejemplo anterior

- Contrapuesta $(\neg q \rightarrow \neg p)$

  Tabla de verdad

p	q	$\neg q \rightarrow \neg p$
F	F	T
F	T	T
T	F	F
T	T	T

  Como se puede apreciar, la Contrapuesta es equivalente a la Condicional.

  bicondicional

  Si $(p \rightarrow q) \wedge (q \rightarrow p) = p \leftrightarrow q$
  Es equivalente a decir, p sii (si y sólo si) q.

  Si p y q son equivalentes, entonces $p \leftrightarrow q$ es una tautología.

Def: Una tautología es una proposición siempre cierta.
Def: Una contradicción es una proposición siempre falsa.

Si $p \rightarrow q$ es una tautología, se escribe $p \Rightarrow q$
Si $p \leftrightarrow q$ es una tautología, se escribe $p \Leftrightarrow q$

Def: Se llama $p(x)$ a una frase abierta o función proposicional, donde "x" es la variable proposicional.

P. ej.: $x^2 - 1 = 0$. Esta ecuación depende de la variable "x".

Def: Se llama "Conjunto de significados de una variable" al conjunto de entidades que pueden sustituir a las variables.

P. ej.: Sea A = {1, 2, 3}.
¿Existe algún elemento de A que haga cierta la ecuación del ejemplo anterior? Vemos que el 1 la hace cierta, luego A es un conjunto.

Def: Se llama "Conjunto de verdad" a aquel subconjunto del conjunto de significados que hace cierta la función.

P. ej:

La función	Conjunto de significados	Conjunto de verdades
$x^2 - 1 = 0$	{1, 2, 3}	{ 1 }
	{ N }	{ 1 }
	{ P }	{ 1, -1 }

- Cuantificadores

Son cuantificadores los siguientes símbolos:

Universal
Para todo x, P(x) es verdadera
$\forall x$, P(x)

Existencial
Existe un x del conjunto de significados para el que P(x) es cierta

$\exists x$ P(x), $\dot{\exists} x$ P(x) ($\dot{\exists}$ = "Existe un único")

Sea | (tal que)

Teor: $\neg (\forall x \, P(x)) \equiv \exists x \mid \neg P(x)$.

Teoría de conjuntos.

Se denotarán los conjuntos con letras mayúsculas, y los elementos con letras minúsculas.

Sea el conjunto de las vocales, A = {a, e, i, o, u}.
Sea el conjunto de los números naturales, N = {0, 1, 2, 3, ...}.
Sea el conjunto de las letras, L = {a, b, c, d, e, f ...}.

$a \in$ A (a pertenece al conjunto de las vocales).
$b \notin$ A (b no pertenece al conjunto de las vocales) .

$a, b \notin$ N (a, b no pertenecen al conjunto de los números).
$a, b \in$ L (a, b pertenecen al conjunto de las letras).

Teor: Dos conjuntos son iguales si contienen los mismos elementos.

Def: Se denomina cardinal de un conjunto, y se representa por |A|, siendo A el nombre del conjunto, al número de elementos de dicho conjunto.

Algunas cuestiones de notación:

$a \neq \{a\}$; el ELEMENTO "a" es distinto al CONJUNTO que contiene "a".
$a \in \{a\}$; el elemento a es subconjunto del conjunto que lo contiene.
$A \subseteq B$; se dice que A está contenido en B.
$\{a, b\} \neq \{\{a, b\}\}$; el conjunto $\{a, b\}$ es distinto del conjunto que lo contiene.

Teor: $(A \subseteq B) \wedge (B \subseteq A) \Leftrightarrow A = B$.

Teor: Si $(A \subseteq B) \wedge (B \subseteq C) \Rightarrow A \subseteq B$.

Def: Existe el llamado Conjunto Vacío ($\varnothing$), que es el que no tiene ningún elemento. $\varnothing \subseteq A, \forall A$ (el vacío es subconjunto de todo conjunto).

Def: Se denomina Conjunto Potencia al conjunto de todos los subconjuntos de A.
Se denota por $\wp(A)$. Este conjunto debe tener 2^A elementos.

P. ej.: $A = \{0, 1\}$
$2^A = \wp(A) = \{\{0\}, \{1\}, \{0, 1\}, \{\varnothing\}\}$

$|2^A| = 2^{|A|}$ ⟶ $\{\varnothing\} \in 2^A, \forall A$
⟶ $A \in, 2^A \ \forall A$

Def: Definición de Familia Indexada de Conjuntos.
Sea I conjunto.
Si $\forall \alpha \in I$, $A\alpha$ es un conjunto, $\{A\alpha \mid \alpha \in I\}$ se llama F.I.C.

P.ej.:
Si $\forall n > 0$, $An = [-\frac{1}{n}, \frac{1}{n}]$, $\{ An \mid n \in N, n > 0 \}$ es F.I.C.
$An = \{ [-1, 1], [-\frac{1}{2}, \frac{1}{2}], ... \}$

Operaciones con conjuntos.

- Unión $(A \cup B) = \{x \mid x \in A \ \vee \ x \in B\}$
- Intersección $(A \cap B) = \{x \mid x \in A \ \wedge \ x \in B\}$
- Complemento de B relativo a A $(A - B) = \{x \mid x \in A \ \wedge \ x \notin B\}$

Def: A y B son disjuntos si no tienen elementos comunes, es decir, si $A \cap B = \{\varnothing\}$

Def: Se llama U al conjunto universal

Ejemplos:

$U - A = \{x \mid x \in U \wedge x \notin A\}$, también se denomina conjunto complementario de A, se denota también: $\bar{A}$

$\bar{U} = \varnothing$

$\bar{\varnothing} = U$

Teor:

1) $\overline{A \cup B} = \bar{A} \cap \bar{B}$
2) $\overline{A \cap B} = \bar{A} \cup \bar{B}$
3) $\bar{\bar{A}} = A$ (doble complementación)
4) $A - B = A \cap \bar{B}$

Def: Se define el Producto Cartesiano como:
$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$
$(a, b) = (a', b') \Leftrightarrow a = a' \ \wedge \ b = b'$

Def: Se define una relación como:
$A \Re B$, si $(a, b) \in \Re \ \Rightarrow \ a \Re b$

Ejemplos:
$\Re \subseteq A \times A$ (relación sobre A)
$\Re \subseteq N \times N$
$(x, y) \in \Re$ si $x \leq y$

Def: Se define el Dominio = $Dom(R) = \{x \in A \mid \exists y \in B, (x, y) \in R\}$

Def: Se define la Imagen = $Im(R) = \{y \in B \mid \exists x \in A, (x, y) \in R\}$

Def: Se define la partición de un conjunto como:
Sea $A \neq \varnothing$,
El conjunto Ψ de subconjuntos de A es una partición de A si

verifica:

1) $\forall$ A, B$\in \Psi$, A$\cap$B=$\varnothing$, ó bien A=B
2) $\bigcup_{B\in\Psi} B = A$

P.ej.:
Sea A={0,1,2,3,4,5,6,7,8,9,10}
Ψ = {{0,2,4,6},{1,3,5,7},{8,10},{9}}

La unión de todos los subconjuntos de Ψ da como resultado "A", con lo cual se cumple la regla 2) antes especificada.

Def: Se define Relación de Equivalencia del siguiente modo:
Sea Ψ una partición de un conjunto X$\neq \varnothing$.

P.ej.:
Relación sobre X: $\Re \subseteq X\times X$
$\Re$ ={(x, y) | x e y están en el mismo conjunto de Ψ }
X={1,2,3}
Ψ ={{1},{2,3}}
$\Re$ ={(2,3),(2,2),(3,3),(1,1),(3,2)}

Esta relación tiene 3 propiedades:
1) **Reflexiva** $\rightarrow$ $\forall$ a$\in$ X, (a, a)$\in \Re$
2) **Simétrica** $\rightarrow$ $\forall x, y\in$ X, $x \Re y \Rightarrow y \Re x$
3) **Transitiva** $\rightarrow$ $\forall x, y, z\in$ X, $x \Re y \wedge y \Re z \Rightarrow x \Re z$

Si cumple las tres propiedades, estamos hablando de una relación de equivalencia.

Def: Se define Clase de equivalencia, como:
Sea $\Re$ relación de equivalencia sobre X
Sea $a\in$ X,
[a] = clase de equivalencia de a = { $y\in$ X | (a, y)$\in \Re$ }

Teor: Las clases de equivalencia de una relación de equivalencia forman una partición del conjunto sobre el que está definida la relación.

Teor: Cualquier partición de un conjunto describe una relación de equivalencia sobre el conjunto.

Def: Se define Función Total de la siguiente manera:
Sea f $\subseteq$A$\times$B
1) Dom(f) = A
2) $\left.\begin{array}{l}(x, y)\in f\\(x, z)\in f\end{array}\right\} \Rightarrow y = z$

Dicho de otro modo, todos los elementos de A tienen que tener imagen en B.

Def: Se define Función Parcial de la siguiente manera:
Sea $f \subseteq A \times B$
1) $Dom(f) \subseteq A$
2) $\left.\begin{matrix}(x, y) \in f \\ (x, z) \in f\end{matrix}\right\} \Rightarrow y = z$
En este caso, sin embargo, no todos los elementos deben tener imagen en B.

Def: Se define Función Inversa como:
Sea $y \subseteq B$, $f^{-1}(y) = \{x \in A \mid f(x) = y$, para algún $y \in Y\}$

Def: Se define Función Inyectiva como:
Sea f: A $\longrightarrow$ B
f es inyectiva si,
$\left.\begin{matrix}(x, y) \in f \\ (z, y) \in f\end{matrix}\right\} \Rightarrow x = z$
$f(x) = f(z) \Rightarrow x = z$

Def: Se define Función Sobreyectiva como:
Sea f: A $\longrightarrow$ B
f es sobreyectiva si, $\forall y \in B, \exists x \in A \mid f(x) = y$

Def: Una Función es Biyectiva si y sólo si es inyectiva y sobre.

Ejemplos:

f: N $\longrightarrow$ N Función inyectiva y sobreyectiva
$x \quad \alpha \quad f(x) = x$

g: N $\longrightarrow$ N
p $\quad \alpha \quad g(p) = p+1$ Es inyectiva, pero no sobre ($\exists f^{-1}(0)$)

h: P $\longrightarrow$ P Hay elementos de y que no son imágenes de x, luego no es inyectiva (por ej, los nºs negativos), y tampoco es sobre porque por ejemplo el -1 y el 1 tienen la misma imagen.
$q \quad \alpha \quad h(q) = q^2$

Def: Se define la Composición de funciones como:
Sean $\Re \subseteq A \times B, \quad \Im \subseteq B \times C$

$\Im \circ \Re = \{(a, c) \mid \exists b \in B \mid (a, b) \in \Re \wedge (b, c) \in \Im\}$

Ejemplo:
$\Re = \{(0,1),(1,3),(1,2),(0,2)\}$

$\Im = \{(1,a),(1,b)\}$
$\Im \circ \Re = \{(0,a), (0,b)\}$
$\Re \circ \Im = \varnothing$

Sean f, g: A $\longrightarrow$ A
$g \circ f = \{(a, b) \mid$ Para algún $x \in$ A, $f(a) = x \wedge g(x) = b)\}$

Ejemplo:

Sean $f(x) = x^3 - 7$ y $g(x) = x + 5$
$f \circ g(x) = f(g(x)) = f(x + 5) = (x + 5)^3 - 7$
$g \circ f(x) = g(f(x)) = g(x^3 - 7) = x^3 - 7 + 5 = x^3 - 2$

Def: Se define Inducción de la siguiente manera:

Sea A $\subseteq$ N,
A es inductivo si:
$\forall x \in$ A, $x + 1 \in$ A

Por ejemplo:
{10,11,12,13,14,15...} es inductivo
{1,3,5,7,...} no es inductivo

Si es un conjunto inductivo, y además $0 \in$ Conjunto, Conjunto = N.
Dicho de otro modo:
$\left.\begin{array}{l} 0 \in A \\ n \Rightarrow n+1 \end{array}\right\}$ es cierta para todo n$\in$ N.

Principio de Inducción Matemática.

Dado A $\subseteq$ N,
1) $0 \in$ A
2) A es inductivo

Si se cumple 1) y 2), se puede afirmar que A = N

Dem:

Sea $n + 2 < 3(n + 1)$

n	$n + 2$	$3(n + 1)$
0	2	3
1	3	6
2	4	9
3	5	12

Nota: Lo anterior NO es una demostración, lo posterior, sí

Sea $B = \{n \in N \mid n + 2 < 3(n + 1)\}$

Es B = N? Lo intentaremos demostrar usando la inducción matemática.

$B \stackrel{?}{=} N$

1) $n = 0$; $\left.\begin{array}{l} n+2=2 \\ 3(n+1)=3 \end{array}\right\} 2 < 3$, luego la función es cierta para n=0,

luego ya tenemos base de inducción.

2) Sup. $n + 2 < 3(n + 1)$ (Hipótesis de inducción)

$(n + 1) + 2 \stackrel{?}{<} (3((n + 1) + 1)$ (hay que demostrar si se cumple $\forall n$)

$(n + 1) + 2 = n + 3 = (n + 2) + 1 < 3(n + 1) + 1 = 3n + 4 < 3n + 6 = 3((n + 1) + 1)$

3) Concl.: B = N, $n + 2 < 3(n + 1)$, $\forall n \in N$

Dem P(n): $1 + 3 + 5 + ... + (2n - 1) = n^2$

n	$\sum$	n^2
0	1	1
1	4	4
2	9	9
3	16	16

Como vemos, para $n = 0$ no vale, sin embargo, sí vale para N - {0}

1) Si $n = 1$, $1 = 1^2$

2) Sup. $1 + 3 + 5 + ... + (2n - 1) = n^2$

Hip. de inducción: $1 + 3 + 5 + ... + (2q - 1) = q^2$

$1 + 3 + 5 + ... + (2q - 1) + (2(q+1) - 1) \stackrel{?}{=} (q + 1)^2$

$1 + 3 + 5 + ... + (w1 - 1) + (2(q + 1) -1) = q^2 + (2q + 1) = (q + 1)^2$

3) Luego vemos que P(n) es cierta $\forall n \in N - \{0\}$

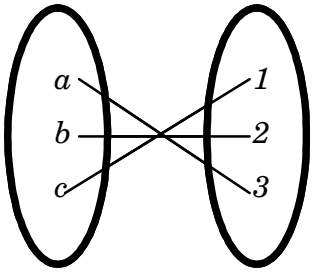
Def: Definición de Cardinalidad:

A y B son equipotentes ($A \cong B$) (que tienen el mismo cardinal) si

$\exists f: A \longrightarrow B$, con f biyectiva.

P. ej.:

$\{a, b, c\} \cong \{1, 2, 3\}$ Como vemos, tienen el mismo cardinal.



Teor: $\left.\begin{array}{l} A \cong C \\ B \cong D \\ A \cap B = C \cap D = \varnothing \end{array}\right\} A \cup B \cong C \cup D$

Def: Se definen Conjuntos N_K del siguiente modo:
$\forall K \in N, N_K = \{1..K\}$

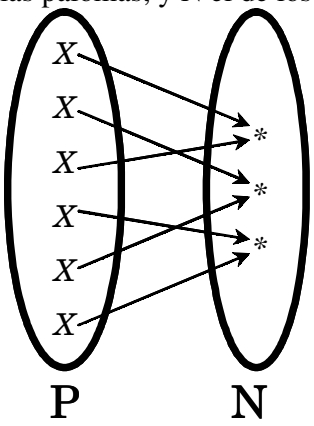
A es finito si:
a) $A = \varnothing$, en este caso $card|A| = |\varnothing| = 0$
b) $A \cong \bigcup_{B \in \varnothing} B = A_K$, en este caso $card|A| = |N_K| = K$

Teor: $\left.\begin{array}{l} A \cap B = \varnothing \\ A, B, finitos \end{array}\right\} |A \cup B| = |A| + |B|$

a) $A = \varnothing$, $|A \cup B| = |B| = 0 + |B| = |A| + |B|$
b) $B = \varnothing$, $|A \cup B| = |A| = 0 + |A| = |A| + |B|$
c) $A \neq \varnothing \wedge B \neq \varnothing$
$|A| = m \Rightarrow \exists f: A \longrightarrow N_m$ biyectiva
$|B| = n \Rightarrow \exists g: B \longrightarrow N_n$ biyectiva

Teor: Teorema del palomar

Sea el conjunto P el de las palomas, y N el de los nidos



Vemos que hay más palomas que nidos, luego al menos en algún nido deberá haber más de una paloma, con lo cual:

$|P| > |N|$

$f: P \longrightarrow N$, f no es inyectiva.

Corolario (consecuencia menor del teorema):

Si A es finito y $B \subset A$,

$$B \ncong A$$

Def: Un conjunto es enumerable si $A \cong N$.

Def: Un conjunto es numerable si es enumerable o finito.

Teor: Sea A enumerable,

Si $B \subseteq A$, B es infinito

$\Downarrow$

B es enumerable

Como A es enumerable $\Rightarrow \exists f: A \longrightarrow N \ \wedge \ \exists f: N \longrightarrow A$ es biyectiva.

$f(n) = a_n$

$A=\{a_0, a_1, a_2 .... a_n\}$

Sea h_0 el primer subíndice tal que $a_{n0} \in B$

Sea h_1 el primer subíndice tal que $a_{n1} \in B - \{a_{n0}\}$

$\aleph$

Sea h_k el primer subíndice tal que $a_{nk} \in B - \{a_{n0}, a_{n1}, a_{nk-1}\}$

$\Downarrow$

B es enumerable

Teor: Todo conjunto infinito contiene un subconjunto enumerable.

Sea $X \neq \varnothing$, $\exists x_0 \in X$, $\exists x_1 \in X$, ...

$|X| \leq |Y|$ si $\exists f: X \longrightarrow Y$ inyectiva.

$|X| < |Y|$ si $|X| \leq |Y| \ \wedge \ X \neq Y$

Teor: Teorema de Cantor.

$|X| < |\wp(X)|$

Tema 2
Conceptos Básicos. Lenguajes.

Alfabetos, cadenas y Lenguajes.

Def: Se denomina Alfabeto (Σ) a un conjunto finito y no vacío de símbolos.
Ejemplos:
$\Sigma = \{0, 1\}$ $\Sigma = \{a\}$
$\Sigma = \{\cdot\ , -\}$ $\Sigma = \{©, ®\}$

Si Σ_1 y Σ_2 son alfabetos, $\Sigma_1 \cup \Sigma_2$ es un alfabeto.
Si Σ_1 y Σ_2 son alfabetos, $\Sigma_1 \cap \Sigma_2$ es un alfabeto, siempre que dicha intersección sea distinta del vacío,

Sea $\Sigma = \{a, b, c\}$
$a \in \Sigma$
$€ \notin \Sigma$

Def: Se denomina Cadena, palabra o frase a una secuencia finita de símbolos de un alfabeto.

Ejemplos:
w=0101 w=*aaaaa*
w=·--· · -· w=©©

Teor: $\forall a \in \Sigma$, a es cadena sobre Σ.

¡Ojo!, Sea $\Sigma = \{a, b\}$, $ab \neq ba$

Def: $|w|$ = Longitud de la cadena w
Ejemplos:
w = *awañac*, $|w| = 6$
w = a, $|w| = 1$

Def: Se define cadena vacía, y se denota por ε (épsilon), aquella cuya longitud sea 0.
ε = Cadena vacía, $|\varepsilon| = 0$.

Def: Se define lenguaje formal al conjunto de cadenas sobre un alfabeto.

Ejemplos:
Sea $\Sigma = \{0, 1\}$

$L_1 = \{0, 1, 11, 1111\}$ $L_2 = \{0,1\} = \Sigma$

L_3={0, 00, 000, 0000.....

L_4={ ε, 1, 10, 110,...} $L_5=\varnothing$

Teor: Un lenguaje SÍ puede ser vacío, como es el caso de L_5.

Aclaración: $\{\varepsilon\} \neq \varnothing$

Más ejemplos:
Sea $\sum$ = {x, *}

L_1={x} L_2={x****,x}
L_3={*,**,***,...
$L_4=\varnothing$ L_5={ ε}

Si tenemos una cadena sobre $\sum$ → w∈ L (pertenece al lenguaje) ó w∉ L (no pertenece al lenguaje)

Def: Dado $\sum$, se llama lenguaje universal a $\sum$*, o cierre de $\sum$. Es el lenguaje de todas las cadenas que puedo formar con ese alfabeto.

Ejemplo:
$\sum$ = {*a*}, $\sum$*={ ε, *a*, *aa*, *aaa*.....}

Teor: Sobre cualquier $\sum$, $\sum$* es infinito.

Operaciones con cadenas.

Sean *x*, *w*∈ $\sum$*,

- *x*· *w* = **concatenación**
  Sea $\sum$ = {*a*, *b*, *c*}
  a, *b*∈ $\sum$, *c*∈ $\sum$, *ab*· *c* = *abc*

  |*wx*| = |*w*| + |*x*|
  w ε= ε*w*=*w*, ∀ *w*∈ $\sum$*

- w^n = **potencia**

$$w^n = \begin{cases} \varepsilon, & si\ n = 0 \\ w \cdot w^{n-1} & si\ n \geq 1 \end{cases}$$

  Sea $\sum$ = {1,0}
  w=010∈ $\sum$*

$(010)^0 = \varepsilon$
$(010)^1 = 010\cdot\ (010)^0 = 010\cdot\ \varepsilon = 010 = w$
$(010)^2 = 010\cdot\ 010\cdot\ (010)^0 = 010\cdot\ 010\cdot\ \varepsilon = 010\cdot\ 010 = 010010$

Def: Una cadena es igual a otra sii tienen la misma longitud y los símbolos en la misma posición.

Def: Se define un prefijo de la siguiente manera:
Sean $x, w \in \Sigma^*$,
x es prefijo de w si $w = x\cdot\ y$, si $\exists y \in \Sigma^* \mid w = x\cdot\ y$

Teor: Toda cadena es prefijo de sí misma

Ejemplos:
Sea w=123,
Prefijos(w) = { ö, 1, 12, 123}

Def: Se llaman prefijos propios aquellos que no son iguales a la cadena

Def: Se define subcadena de la siguiente manera:
Sea $w \in \Sigma^*$,
$x \in \Sigma^*$ es subcadena de w si $w = zxy$, $\exists z, y \in \Sigma^* \mid w = zxy$

Teor: ε es subcadena de toda cadena.

Ejemplos de subcadenas:
Sea w = 123,
Subcadenas sobre w: 1, 2, 3, 12, 23

¿Cuántas subcadenas se pueden formar con una cadena?

$s = 1 + 2 + 3 + ... + m$
$s = m + m\text{-}1 + ... + 1$
$2s = (m + 1) + (m + 1) + ... + (m + 1)$

$2s = (m + 1)\cdot\ m \Rightarrow s = \frac{m(m+1)}{2} + 1$ ← Porque ε es subcadena de toda cadena

Def: Se define una subsecuencia de una cadena como una combinación de símbolos de dicha cadena en orden, pero no tienen que ser consecutivos.

P.ej.:
w = *abcre*; Subsecuencias: "*ace*", "*abre*", "*acre*", "*bre*", "*are*"...

El número de subsecuencias que se pueden formar con una cadena es de 2^n

Def: Se define la inversa o traspuesta de una cadena como:

$$w^i = \begin{cases} \varepsilon & \text{si } w = \varepsilon \\ y^i a & \text{si } w = ay \ (a \in \Sigma, y \in \Sigma^*) \end{cases}$$

P.ej.:

$(arroz)^i = (rroz)^i a = (roz)^i r = (oz)^i rra = (z)^i orra = \varepsilon^i zorra = zorra$

Operaciones con lenguajes.

Sean L_1, L_2 lenguajes sobre Σ.

$L_1, L_2 \subseteq \Sigma^*$

Concatenación: $L_1 \cdot L_2 = \{ xy \in \Sigma^* | x \in L_1 \wedge y \in L_2 \}$

P.ej.:

$L_1 = \{0, 1\}$

$L_2 = \{a, b, cd\}$

$L_1 \cdot L_2 = \{0a, 0b, 0cd, 1a, 1b, 1cd\}$

Potencia:

Sea $L \subseteq \Sigma^*$,

$$L^n = \begin{cases} \text{å} & \text{si } n = 0 \\ L \cdot L^{n-1} & \text{si } n \geq 1 \end{cases}$$

P. ej.:

$L = \{0, 1\}$

$L^0 = \{ \varepsilon \} \neq \varnothing$

$L^1 = L \cdot L^0 = L \cdot \{\varepsilon\} = L = \{0, 1\}$

$L^2 = L \cdot L^1 = LL = \{00, 01, 10, 11\}$

L^5 = Números binarios de 5 bits

Teor: Todo alfabeto es un lenguaje

Teor: $\varnothing^0 = \{ \varepsilon \}$

Teor: $\varnothing^x = \varnothing, \forall x \geq 1$

Sean $L_1, L_2 \subseteq \Sigma^*$,

$L_1 \cup L_2 = \{x \in \Sigma^* | x \in L_1 \vee x \in L_2 \}$

$L_1 \cap L_2 = \{x \in \Sigma^* | x \in L_1 \wedge x \in L_2 \}$

Def: L_1 es sublenguaje de L_2 si $L_1 \subseteq L_2$

Teor: Todo lenguaje L sobre Σ es $L \subseteq \Sigma^*$

Teor: Dos lenguajes son iguales si tienen todos sus elementos iguales

Teor: Sean $L_1, L_2, L_3 \subseteq \Sigma^*$,
$L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$
$(L_1 \cup L_2)L_3 = L_1L_3 \cup L_2L_3$

Def: Se define el Cierre de Kleene (o cierre estrella) de este modo:

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$$ Por eso existe Σ^*

Def: Se define el Cierre Positivo de este modo:

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup \ldots$$

Teor: $L^+ = L^* - \{\varepsilon\}$

Complementación:
Sean $L_1, L_2 \subseteq \Sigma^*$,
$L_1 - L_2 = \{ x \in \Sigma^* \mid x \in L_1 \wedge x \notin L_2 \}$

$\Sigma^* - L = \overline{L} = \{ x \in \Sigma^* \mid x \in \Sigma^* \wedge x \notin L \}$

Teor: $A^+ = AA^* = A^*A$

Teor: $(A^*)^* = A^*$

Teor: $(A^+)^+ = A^+$

Expresiones regulares.

1) $\varnothing$ es Expresión regular
2) ε es Expresión regular
3) $\forall a \in \Sigma$, a es Expresión regular
4) Si r_1 y r_2 son expresiones regulares $\Rightarrow r_1 \cdot r_2, r_1 \mid r_2, r^*$ son E.R.
5) Ninguna otra secuencia de métodos es Expresión regular.

Teor: Toda expresión regular representa un lenguaje regular

r	$L(r)$
$\varnothing$	$\varnothing$
ε	ε
a	{a}
$r_1 \cdot r_2$	$L(r_1) \cdot L(r_2)$

← Lenguaje representado

$r_1 \mid r_2$	$L(r_1) \cup L(r_2)$
r_1^*	$L(r_1)^*$

Inversa:

Sea $L \subseteq \sum^*$,

$L^i = \{w \in \sum^* \mid w^i \in L\}$

P.ej.:

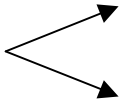
$L_1 = \{abra, cadabra\}$

$L^i = \{arba, arbadac\}$

Tema 3

Lenguajes Regulares y Autómatas Finitos.

Expresiones Regulares.

$L \subseteq \Sigma^*$ 

Tiene que ser finito

ó

Tiene que ser infinito enumerable ($|L|=|N|$)

P.ej.:

$\Sigma = \{d, c, z\}$

$\Sigma^* = \{ \varepsilon, d, c, z, dd, dc, dz, cd, cc, cz, zd, zc, zz \ldots\ldots$

$\Sigma^* \cong N$

Teor:

$\left.\begin{array}{l} \wp(\Sigma^*) \cong 2^N \\ |N| < 2^N \end{array}\right\}$ Ningún método de especificación determinará TODOS los lenguajes sobre un alfabeto.

Ejemplo de expresión no regular:

$\{b^i, i \geq 0\} = \{ \varepsilon, b, bb, bbb, bbbb \ldots\ldots \} = b^*$

Sin embargo, p.ej.:

$\{a^i b^j \mid i, j \geq 0\} = \{ \varepsilon, a, b, ab, aab, abb, \ldots\ldots \}$

Sí es regular, porque se puede hacer concatenando algunos lenguajes.

En este caso sería por ej:

$L_1 = \{a\}$, $L_2 = \{b\}$, $L_1^* \cdot L_2^* = a^* b^*$, que demuestra que es regular.

Teor: Sean r, s expresiones regulares.

$r = s \Leftrightarrow L(r) = L(s)$

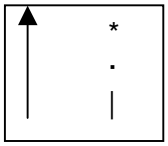
Teor: Puede que hayan varias expresiones regulares que representan el mismo lenguaje.

P. ej.:

$(a^*b)^* = \varepsilon | (a|b)^* b$

$L(\varepsilon | (a|b)^* b) = \{ \varepsilon, ab, b, aab, ababab \ldots\ldots \}$

Teor: Órden de precedencia entre operadores



Notación: $r|\emptyset = r$? El signo de interrogación significa que puede que r esté o no

Equivalencias entre expresiones regulares.

$r \mid s = s \mid r$	$r(s\|t)=rs\|rt$
$r \mid \emptyset = \emptyset \mid r = r$	$(r\|s)t=rt\|st$
$r \mid r = r$	$r^*=r^{**}=r^*r^*=(\varepsilon\|r)^*=r^*(r\|\varepsilon)=\varepsilon\|rr^*$
$r \mid (s \mid t) = (r \mid s) \mid t$	$(r\|s)^*=(r^*\|s^*)^*=(r^*s^*)^*$
$r\varepsilon = \varepsilon r = r$	$r(sr)^*=(rs)^*r$
$r\emptyset = \emptyset r = \emptyset$	$(r^*s)^* = \emptyset\|(r\|s)^*s$
$(rs)t=r(st)$	

Ejercicios:
Describir con palabras qué lenguaje representan las siguientes expresiones regulares:

Expresión regular	Descripción
0*	Cadenas de ceros de cualquier longitud
(0|1)*	Cadenas binarias de 0 y 1 de cualquier longitud
(10)*	Cadenas binarias donde cada 1 va seguido de un 0, ó es ε
(0|1)*1(1|0)*	Cadenas binarias con al menos un 1
(1|10)*	Cadenas binarias que no tienen 2 ceros consecutivos, ó es ε
1(0|1)*1	Cadenas binarias que empiezan y terminan en 1
1*01*01*	Cadenas binarias que tienen dos ceros
(0|1)*00(1|0)*	Cadenas binarias que tienen la subsecuencia 00

Dada la descripción del lenguaje, encontrar una expresión regular equivalente.
(Se supone $\sum = \{0, 1\}$)

Cadenas binarias que acaben en 0	(1|0)*0
Cadenas con sólo un 0	1*01*
Cadenas binarias que no contienen "000"	((1*|0)(1*|0)1)*
Cadenas que si tienen un 1 va precedido y seguido de un 0	0*(010)*0*

Def: Un conjunto es cerrado con respecto a la operación # si:
$\forall x, y \in C$, $x\#y \in C$
Ejemplos:
Z* este conjunto es cerrado respecto a la multiplicación
Z/ este conjunto no es cerrado respecto a la división

Conclusión de la parte: El conjunto de los lenguajes regulares es el menor conjunto cerrado respecto a las operaciones: Unión, intersección, concatenación, complementación y cierre de Kleene, y que contiene el $\emptyset$ y los lenguajes unitarios. Aparecen en analizadores léxicos y en búsqueda de patrones.

Autómatas Finitos Deterministas (AFD, DFA).

Se define un Autómata Finito Determinista como una quíntupla del siguiente modo:

$M \equiv (\Sigma, Q, F, q_0, \delta)$

Descripción de cada uno de los elementos del autómata:

$\Sigma \rightarrow$ Como ya sabemos, es el alfabeto, en este caso el alfabeto del autómata

$Q \rightarrow$ Es el conjunto de estados del autómata. $Q \neq \varnothing$, ha de ser finito y $Q \cap \Sigma = \varnothing$

$F \rightarrow F \subseteq Q$, es el conjunto de estados de aceptación, o estados finales

$q_0 \rightarrow q_0 \in Q$, es el estado de arranque o inicial del autómata

$\delta \rightarrow$ Es la función de transición del autómata. Viene definida como:

$\delta: Q \times \Sigma \longrightarrow Q$

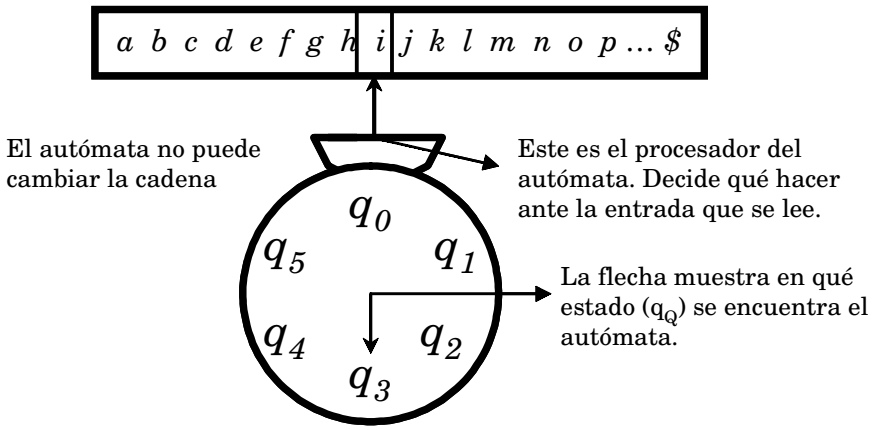
$\delta(q, a) = p$

$p, q \in Q$

$a \in \Sigma$

Aclaración: δ es una función, está definida para todos los pares $Q \times \Sigma$

Estructura de un autómata.



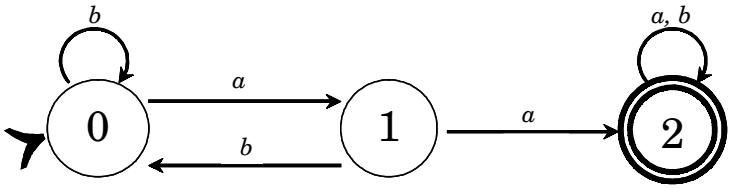
Nota: El apuntador del autómata devuelve el estado en el que se encuentra, que puede ser de aceptación o no. Si es de aceptación "acepto la cadena", si no, "no acepto o rechazo la cadena".

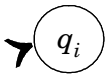
Diagrama de transiciones de un autómata.

Sea $\sum=\{a, b\}$, Q = {0, 1, 2}, $q_0=0$, F = {2}, y δ viene definida así:

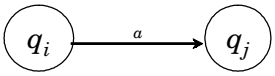
q_i / δ	*a*	*b*
0	1	0
1	2	0
2	2	2

Se define el diagrama de transiciones de dicho autómata como un grafo dirigido, en el que los estados se representan por nodos, las transiciones por flechas, de tal manera que dicho grafo satisface la definición de la función de transición, δ.
En este caso, en el autómata la función δ sería:
$\delta(0, a) = 1$, $\delta(0, b) = 0$, $\delta(1, a) = 2$, $\delta(1, b)=0$, $\delta(2, a) = 2$, $\delta(2, b)=2$
Nótese que para todos los símbolos del alfabeto, existe una transición de algún estado.
Sabiendo esto, el anterior autómata quedaría representado así



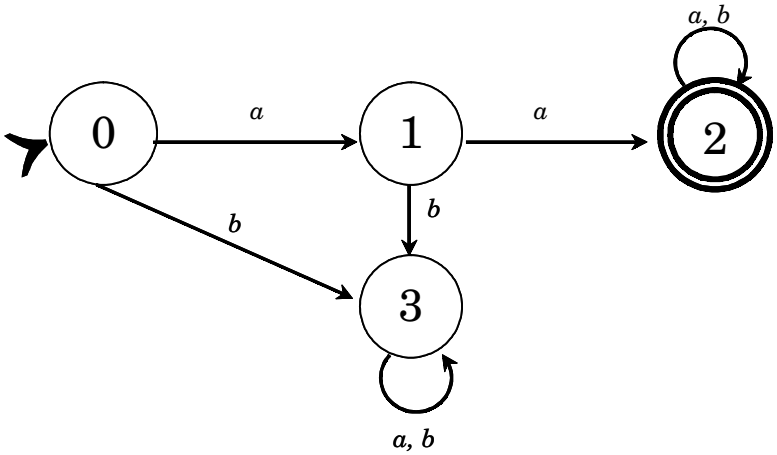
El estado inicial se representa por 

Los estados finales se representarán por 

Las transiciones lo harán como: $\delta(q_i, a) = q_j \Leftrightarrow$ 

Def Se define un estado de absorción o muerte como aquel estado q∈ Q, y q∉ F, que no tiene ninguna transición hacia ningún otro estado (opcionalmente, a sí mismo puede tenerlos), únicamente hay transiciones que inciden en él. Es decir, si $\delta(q, a) = \varnothing$, *ó*, $\delta(q, a) = q$, $\forall a \in \sum$.

P. ej.:



Como vemos, el estado 3 no tiene ninguna transición, únicamente hay transiciones que inciden en él. Además, $3 \notin F$, luego 3 es un estado de muerte.

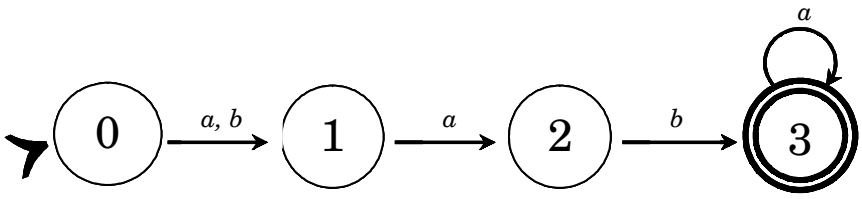
Cuando tenemos estados de muerte, se toma el convenio de no dibujarlos.

En este último autómata,

Si $w = aaab \Rightarrow$ El autómata acepta la cadena, puesto que para en 2, que es estado de aceptación.
Si $w = bbba \Rightarrow$ El autómata rechaza la cadena, puesto que para en 3, que no es un estado de aceptación.

Ejercicio

Hacer el autómata que reconozca este lenguaje: $(a|b)aba^*$



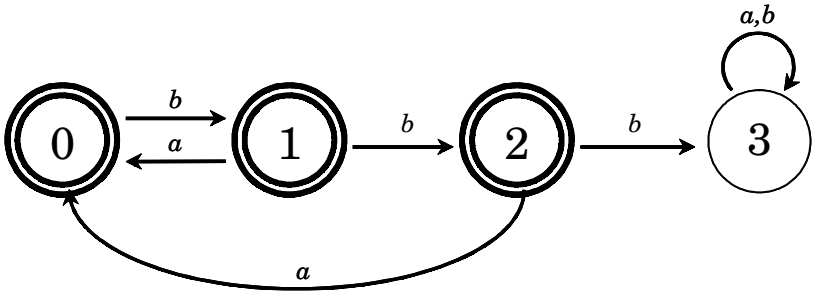
En este diagrama de transiciones, se ve que se ha omitido un estado de muerte, porque por ejemplo el estado 1 no tiene transición con el símbolo "b", y va a parar a dicho estado de muerte. Igual pasa con el estado 2 y el símbolo "a", y el estado 3 con el símbolo "b"

Ejercicio

Hacer el diagrama de transiciones con esta definición del autómata:

$Q=\{0,1,2,3\}$
$\Sigma = \{a, b\}$
$q_0 = 0$
$F=\{0, 1, 2\}$

q_i / δ	a	b
0	0	1
1	0	2
2	0	3
3	3	3



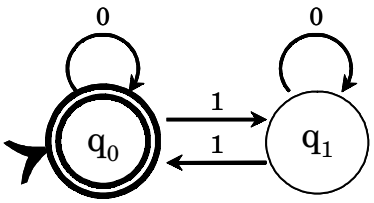
Como vemos, el estado 3 es un estado de muerte y podía haberse omitido.

Este autómata, por ejemplo, acepta combinaciones de cadenas que no tengan 3 "b" seguidas.

Ejercicio

$Q=\{q_0, q_1\}$
$\sum=\{0, 1\}$
$F=\{q_0\}$
$q_0=q_0$

q_i/δ	*0*	*1*
q_0	q_0	q_1
q_1	q_1	q_0

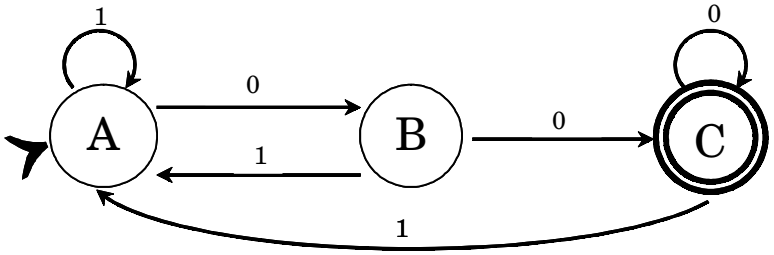


Teor: Los lenguajes reconocidos por los DFA son los **lenguajes regulares**

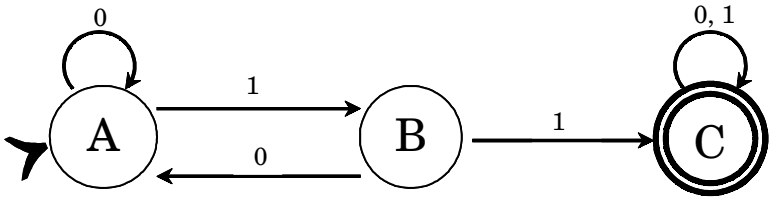
Ejercicio.
Suponiendo $\sum=\{0, 1\}$, dibujar los diagramas de transición que reconozcan:

a) Cadenas terminadas en 00
b) Cadenas con dos "unos" consecutivos
c) Cadenas que no contengan dos "unos" consecutivos
d) Cadenas con dos "ceros" consecutivos o dos "unos" consecutivos
e) Cadenas con dos "ceros" consecutivos y dos "unos" consecutivos
f) Cadenas acabadas en 00 o 11
g) Cadenas con un "uno" en la antepenúltima posición
h) Cadenas de longitud 4.

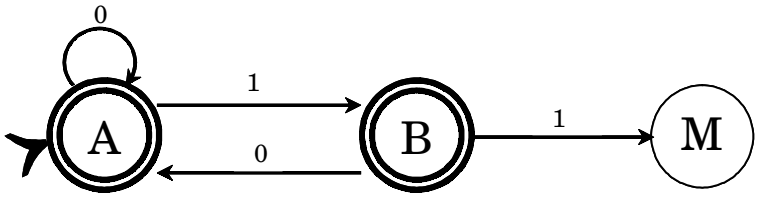
a)


1
0
0
A
B
C
1
1

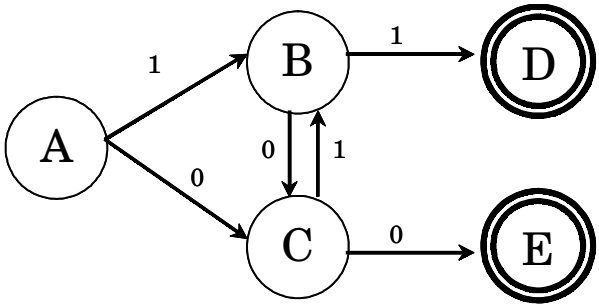
b)


0
0, 1
1
A
B
C
0
1

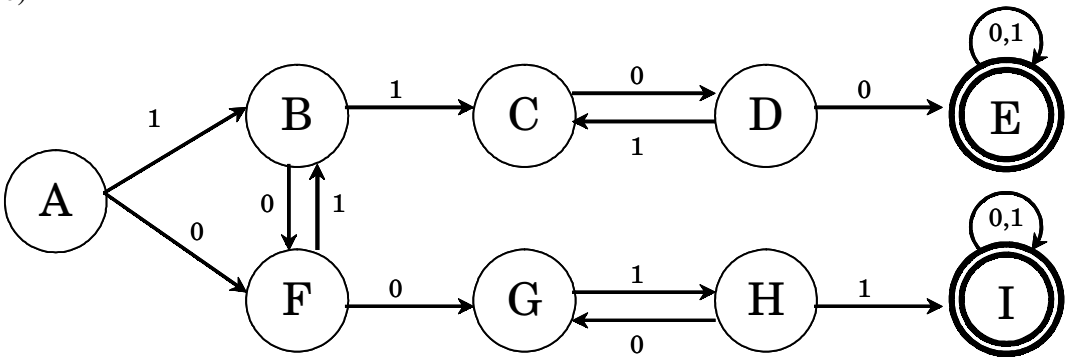
c)


0
1
A
B
1
M
0

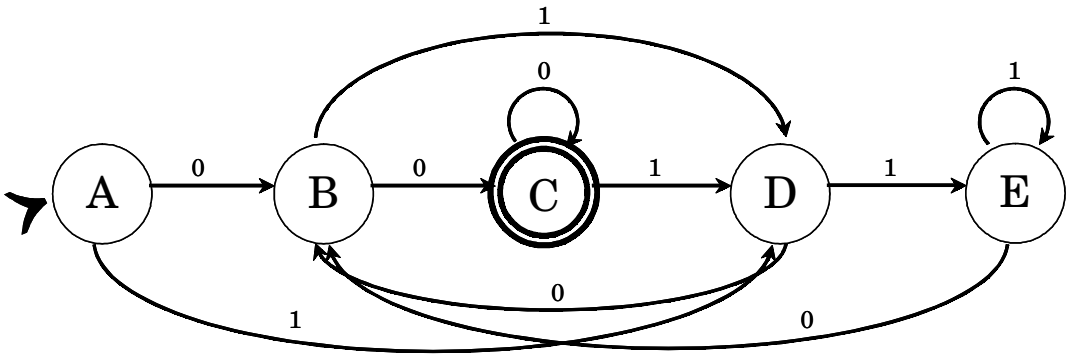
d)


1
B
1
D
A
0
1
0
C
0
E

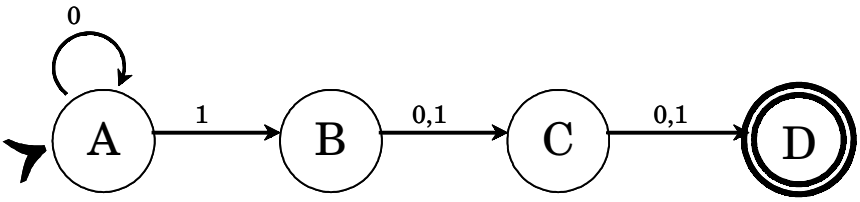
e)


0,1
1
B
1
C
0
D
0
E
1
A
0
1
0
0,1
F
0
G
1
H
1
I
0

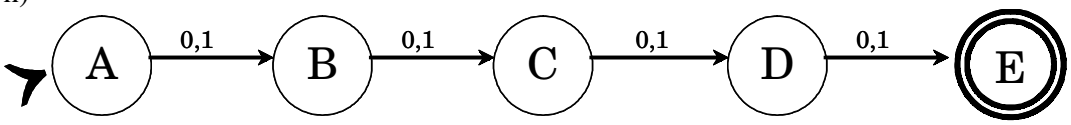
f)



g)



h)



Def: Sea $M \equiv (\sum, Q, F, q_0, \delta)$, con p, q $\in$ Q, se definen estados equivalentes, y se denota por $p \equiv q \Leftrightarrow \forall w \in \sum^*, \delta(p, w) \cap F \neq \varnothing \Leftrightarrow \delta(q, w) \cap F \neq \varnothing$, es decir,si los dos estados se comportan igual ante todas las cadenas de $\sum^*$.

Def: Se dice que $r, s \in Q$ son distinguibles, si $\exists w \in \sum^* | \delta(r, w) \in F \wedge \delta(s, w) \notin F$.

Algoritmo de Reducción de estados de un DFA.

Este algoritmo nos permitirá minimizar el número de estados de un DFA de tal manera que el funcionamiento antes y después de minimizar sea el mismo. Se parte de un conjunto llamado π_0, que contendrá a su vez dos conjuntos, uno que será Q - F y otro que será F, es decir:

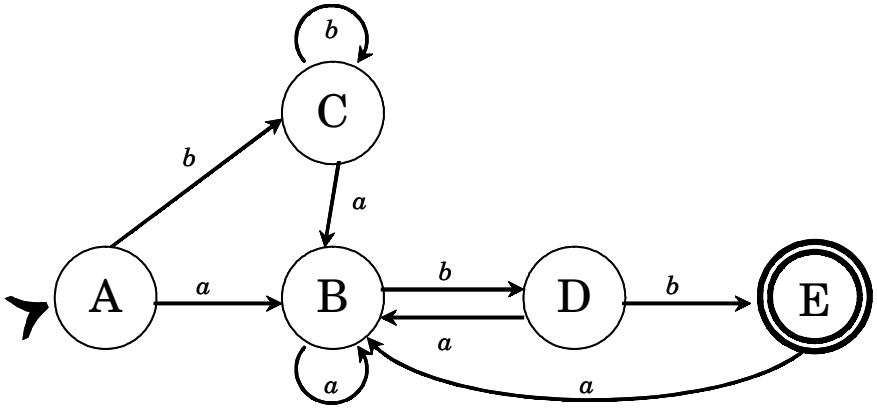
$\pi_0 = \{F, Q-F\}$

Los sucesivos pasos π_i, con i>0, se conseguirán de la siguiente manera:

Hay que coger los distintos símbolos pertenecientes a$\in \sum$, y mirar a dónde transita cada estado de cada cada uno de los conjuntos. Si algún estado $q \in Q$ transitase con dicho símbolo a otro conjunto, éste se separa en un conjunto distinto, y se vuelve a aplicar

dicho algoritmo. Si hubiera un conjunto cuyo cardinal fuera uno (es decir, tiene un elemento), no hace falta comprobar con él las transiciones, puesto que éste no puede cambiar. Si en algún paso del algoritmo, dos o más estados de un mismo conjunto transitasen a un mismo conjunto distinto del que están, esos estados se separan en un conjunto nuevo. El orden en que se aplican los $a \in \sum$ puede ser arbitrario. El algoritmo para una vez que no hayan cambios con NINGUNO de los $a \in \sum$, es decir, con ningún símbolo del alfabeto. Los estados equivalentes serán aquellos que pertenezcan a un mismo subconjunto de π_n, siendo π_n el último paso.

Ejemplo:



Se supone $\sum = \{a, b\}$.

$\pi_0 = \{F, Q\text{-}F\} = \{\{E\}, \{A, B, C, D\}\}$

Si elegimos "*a*" como símbolo del alfabeto, no hay cambios, puesto que no hay ningún elemento de Q que transite a uno de F, ni viceversa. Probemos ahora con la "*b*".

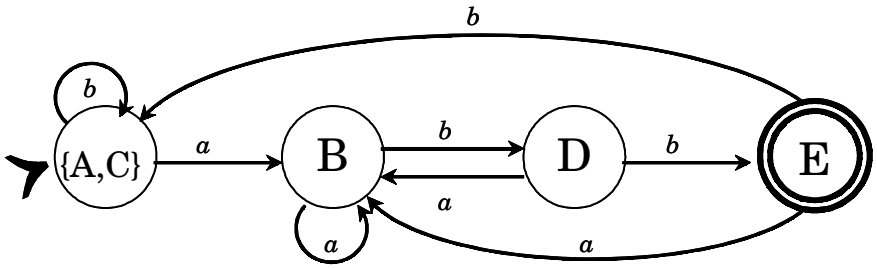
Con la "*b*" vemos que el estado D, transita a E, que es de aceptación y está en un conjunto distinto al de los Q. Separamos dicho estado en uno distinto.

$\pi_1 = \{\{E\}, \{A, B, C\}, \{D\}\}$

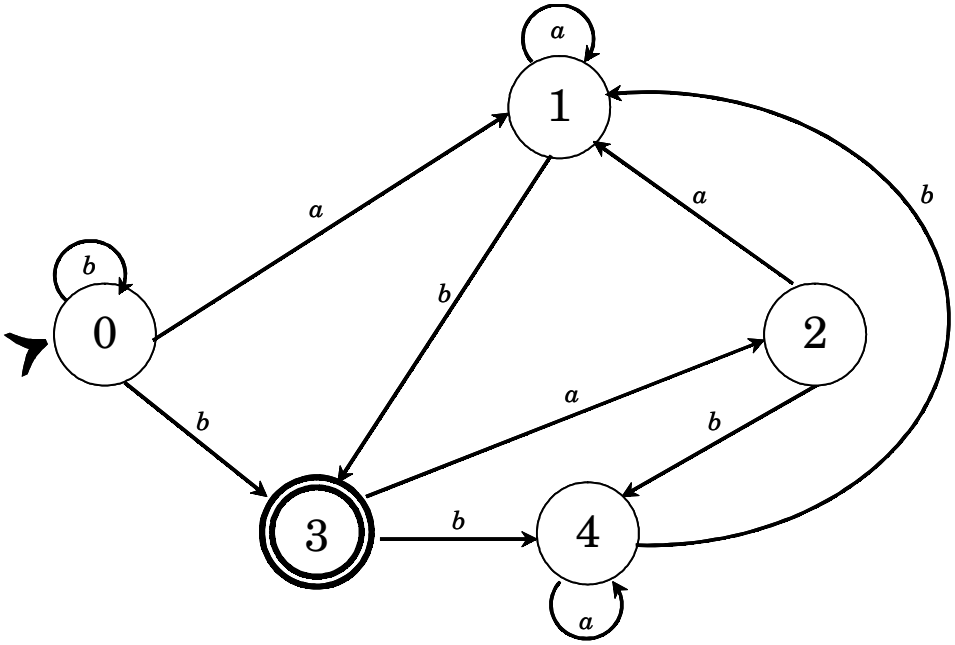
Igualmente con el símbolo "*a*" ahora no hay cambios, sin embargo con la "*b*", vemos que el estado B transita a D, que está en otro conjunto. Lo separamos.

$\pi_2 = \{\{E\}, \{A, C\}, \{B\}, \{D\}\}$

Ahora ni probando con la "*a*" ni con la "*b*" vemos que haya cambio, con lo cual el algoritmo termina. Llegamos a la conclusión de que los estados A y C son estados equivalentes. El diagrama de transiciones queda pues así:



Otro ejemplo:



π_0={{0, 1, 2, 4}, {3}}

El estado 0 con la "*b*", transita a 3, y el 1 igual, luego los separamos en un conjunto distinto.

π_1={{0,1}, {2,4}, {3}}

Si cogemos la "*a*" como símbolo, vemos que el estado 2 transita al estado 1, mientras que 4 con el mismo símbolo transita a 4, luego los separamos.

π_2={{0,1},{2},{4},{3}}

Vemos que el estado 0 con la "*a*" va al estado 1, que está en el mismo conjunto, y el estado 1 hace lo mismo; y el estado 0 con la "*b*" va al estado 3, y el estado 1 hace lo mismo. Por tanto, no hay cambios y el autómata mínimo se obtiene uniendo los estados 0 y 1, y nos queda un autómata de 4 estados.

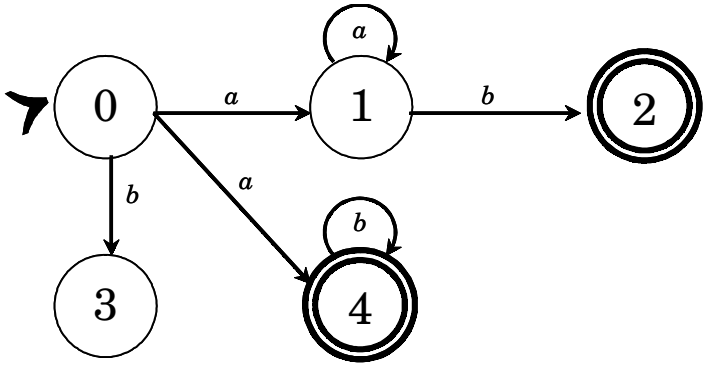
Teor: Si tenemos 2 DFAs, M_1 y M_2, $M_1 \equiv M_2 \Leftrightarrow L(M_1)=L(M_2)$

Autómatas Finitos No Deterministas (FND, NFA).

Los NFAs son un superconjunto de los DFAs, es decir, la relación que hay entre DFA y NFA es que los DFAs son un caso particular de los NFAs.

Cuando tenemos más de una transición con un mismo símbolo del alfabeto desde un estado, hablamos de NFA.

Dibujo ilustrativo:



El estado 0, para el símbolo "*a*" tiene dos transiciones, una al estado 1 y otra al estado 4, es decir, $\delta(0, a) = \{1, 4\}$

Teor: En los NFAs no existen los estados de muerte o absorción.

Def: Se define un NFA de la siguiente manera:
$M \equiv (\Sigma, Q, F, q_0, \delta)$

Los elementos Σ, Q, F, q_0, se definen igual que en los DFAs, lo que cambia es la función de transición, δ.

$$\delta : Q \times \Sigma \longrightarrow \wp(Q)$$ ← Partes de Q
$$\delta(q, a) \subseteq Q$$

Aclaración: $\begin{cases} \varnothing \in \wp(Q) \\ Q \in \wp(Q) \end{cases}$

En el ejemplo ilustrativo de antes:
Q={0,1,2,3,4}
F={2,3,4}
Σ={*a*, *b*}
q_0=0

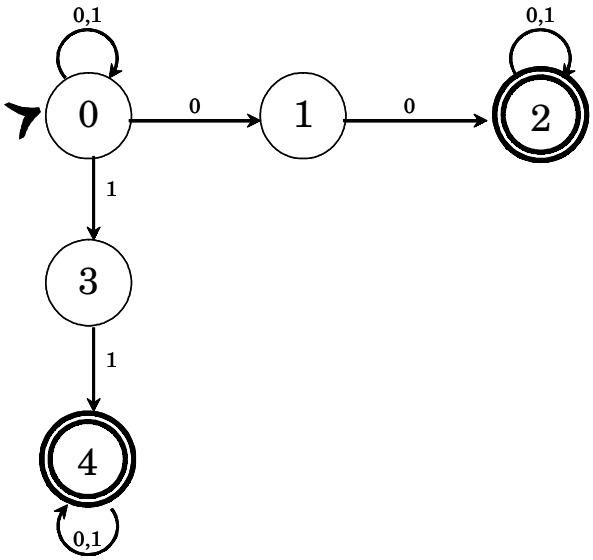
q_i / δ	a	B
0	{1, 4}	{3}
1	{1}	{2}
2	{∅}	{∅}
3	{∅}	{∅}
4	{∅}	{4}

Teor: Si tenemos un NFA $M \equiv (\sum, Q, F, q_0, \delta)$,

$L(M)=\{w \in \sum^* \mid \delta(q_0, w) \cap F \neq \varnothing\}$

Es decir, el autómata acepta aquellos lenguajes en los que al procesar la cadena se obtenga un conjunto en el que haya al menos un estado de aceptación.

Ejemplo:



Sea por ejemplo w = 01001, suponiendo que $\sum = \{0, 1\}$

$\delta(A, 0) = \{A, D\}$
$\delta(\{A, D\}, 1) = \delta(A, 1) \cup \delta(D, 1) = \{A, B\} \cup \varnothing = \{A, B\}$
$\delta(\{A, B\}, 0) = \{A, D\} \cup \varnothing = \{A, D\}$
$\delta(\{A, D\}, 0) = \{A, D, E\}$
$\delta(\{A, D, E\}, 1) = \{A, B, E\}$

¿Se acepta la cadena? El último conjunto, $\{A, B, E\} \cap F = \{E\}$, luego como $\{E\} \neq \varnothing$, esta cadena se acepta.

Teor: Sean M_1 y M_2 dos NFAs, $M_1 \equiv M_2 \Leftrightarrow L(M_1)=L(M_2)$

Teor: Siempre se puede convertir un NFA a DFA

Teor: Los NFAs reconocen los **lenguajes regulares**

Algoritmo de Conversión de un NFA a un DFA.

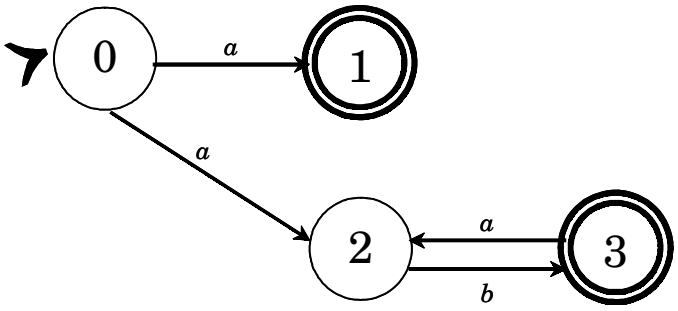
Teor: Dado un NFA $M \equiv (\Sigma, Q, F, q_0, \delta)$,
$\exists$ DFA $M' \equiv (\Sigma', Q', F', q_0', \delta') \mid L(M) = L(M')$

1) $Q' = \wp(Q)$
2) $\Sigma = \Sigma'$
3) $q_0' = \{q_0\}$ (conjunto de estados que contiene el de arranque del NFA)
4) $F' \subseteq Q'$ (aquellos subconjuntos de Q que contengan algún $p \in F$)
5) $\delta'(\{q_1, q_2, \ldots q_k\}, a) = \{p_1, p_2 \ldots p_m\} \Leftrightarrow \delta(\{q_1, q_2, \ldots q_k\}, a) = \{p_1, p_2 \ldots p_m\}$

Para construír un DFA a partir de un NFA, se procede del siguiente modo:
Se toman todos los símbolos de Σ, y se aplica δ sobre el símbolo inicial, q_0. Al hacer eso, darán como resultado un conjunto $A \in \wp(Q)$, que añadiremos a una lista de Estados nuevos. Se denomina marcar un estado a aplicar la función δ sobre todos los símbolos del alfabeto. A partir de ahí hay que marcar todos los estados nuevos que nos salgan. El algoritmo para una vez estén marcados todos los estados nuevos. Serán estados de aceptación aquellos estados que contengan algún estado de aceptación del NFA. Por último se dibuja el diagrama de transiciones.

Ejemplo:

Pasar a DFA este NFA

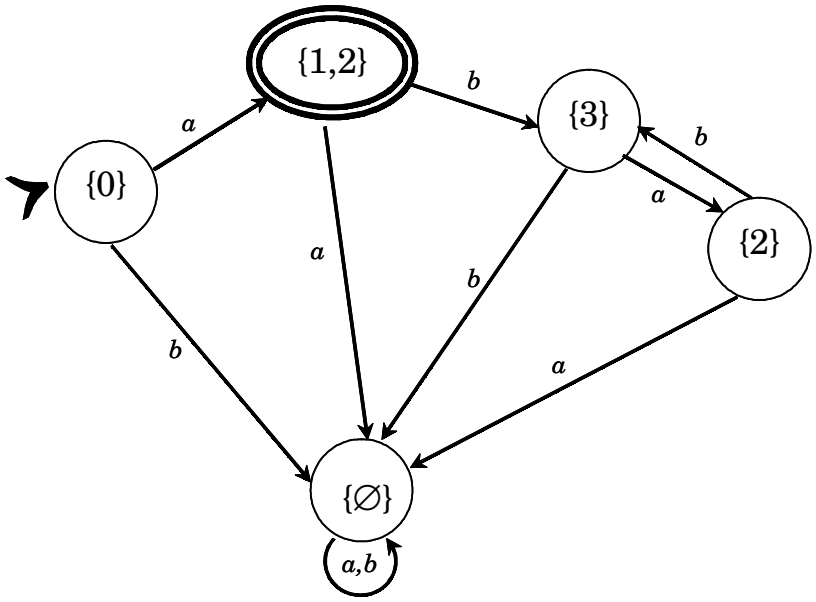


$\Sigma = \{a, b\}$, $q_0 = 0$

Empezamos aplicando δ sobre el símbolo inicial
$\delta(\{0\}, a) = \{1, 2\}$ Este conjunto no lo teníamos, se añade a la lista $\rightarrow$
$\delta(\{0\}, b) = \{\varnothing\}$
Ahora hay que marcar estos estados, y ver si salen estados nuevos.
$\delta(\{1, 2\}, a) = \{\varnothing\}$
$\delta(\{1, 2\}, b) = \{3\}$ No estaba, lo añadimos
$\delta(\{\varnothing\}, a) = \delta(\{\varnothing\}, b) = \{\varnothing\}$
$\delta(\{3\}, a) = \{2\}$
$\delta(\{3\}, b) = \{\varnothing\}$
$\delta(\{2\}, a) = \{\varnothing\}$
$\delta(\{2\}, b) = \{3\}$

Estados nuevos
{1, 2}
{∅}
{3}
{2}

Ya tenemos todos los estados marcados, ahora ya sólo queda pintar el diagrama de transiciones, luego el DFA quedaría así:



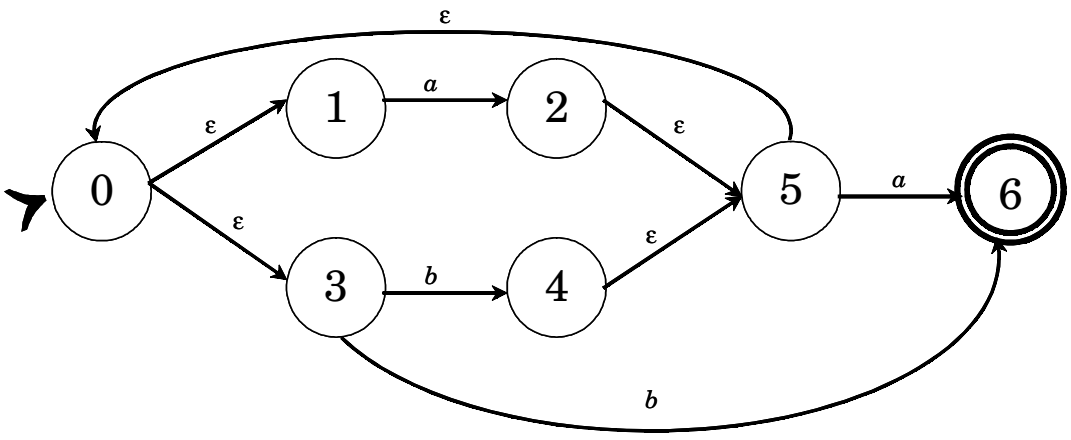
Como podemos apreciar, $\{\varnothing\}$ es un estado de muerte, con lo cual en una reducción de estados posterior se podría quitar.

NFAs con épsilon-transiciones.

Se puede definir un NFA con épsilon-transiciones (NFA-ε) como:
$M \equiv (\Sigma, Q, F, q_0, \delta)$, donde únicamente varía la función δ con respecto de los NFAs, estando definida del siguiente modo:

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow \wp(Q)$$

Ejemplo ilustrativo



Como ya sabemos, ε representa una cadena vacía, es decir $|w|=0$, pero también se puede definir como un símbolo que nos permite transitar a otro estado sin consumir símbolos. A raíz de esto, se puede definir la ε-Clausura.

Def: Se define la ε-Clausura (ε-cl) como un conjunto de todos los estados a los que se puede acceder desde un estado $q \in Q$ sin consumir símbolos de la entrada.

ε-cl={$p \in Q \mid p$ es accesible o alcanzable desde q sin consumir símbolos de la entrada}

P.ej.:
En el anterior diagrama de transiciones,
ε-cl(0) = {0, 1, 3}
ε-cl(2)={2, 5, 0, 1, 3} = {0, 1, 2, 3, 5}
ε-cl({1, 4})= ε-cl(1) ∪ ε-cl(4) = {1} ∪ {4, 5, 0, 1, 3} = {0, 1, 3, 4, 5}

Teor: $q \in$ ε-cl(q), $\forall q$. Es decir, la ε-cl nunca puede ser vacía, contendrá al menos el estado sobre el que se aplica la ε-cl, a menos que sea la ε-cl({∅}), porque ε-cl({∅})={∅}

Sea $r \in Q$,

$$\delta(r, a) = \bigcup_{q \in R} \delta(q, a)$$

$\hat{\delta}(q, \varepsilon) =$ ε-cl(q)

$\hat{\delta}(q, aw) = \cup r \in \delta$ (ε-cl(q), a)

$a \in \Sigma$, $w \in \Sigma^*$

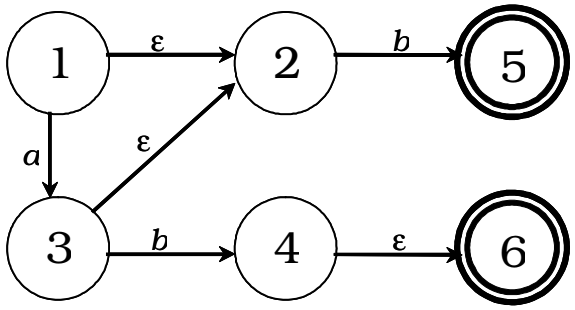
$$\hat{\delta}(r, a) = \bigcup_{q \in R} \delta(q, a)$$

En general, $\delta \neq \hat{\delta}$.

Teor: Si tenemos un NFA-ε, $M \equiv (\Sigma, Q, F, q_0, \delta)$

$L(M)=\{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

Ejemplo:



Para saber si w∈ L(M), se procede haciendo la ε-cl del símbolo inicial. Posteriormente al conjunto conseguido al aplicar dicha clausura, se aplica la función δ sobre el primer símbolo de la cadena, posteriormente se aplica la ε-cl y se vuelve a aplicar δ sobre el siguiente símbolo de la cadena. Estos dos últimos pasos se repiten hasta que se evalúen todos los símbolos de la cadena. Si en el conjunto obtenido haciendo la última ε-cl hubiera algún estado de aceptación, la cadena se acepta, si no, se rechaza.

Sea $w = a$, $|w| = 1$
$\hat{\delta}(0, w) = \delta(0, a)$

ε-cl(0) = {0, 1} (ε-cl del símbolo inicial)
$\delta(\{0, 1\}, a) = \{3\}$
ε-cl({3})={1, 3}

$\{1, 3\} \cap F = \varnothing$, luego la cadena se rechaza.

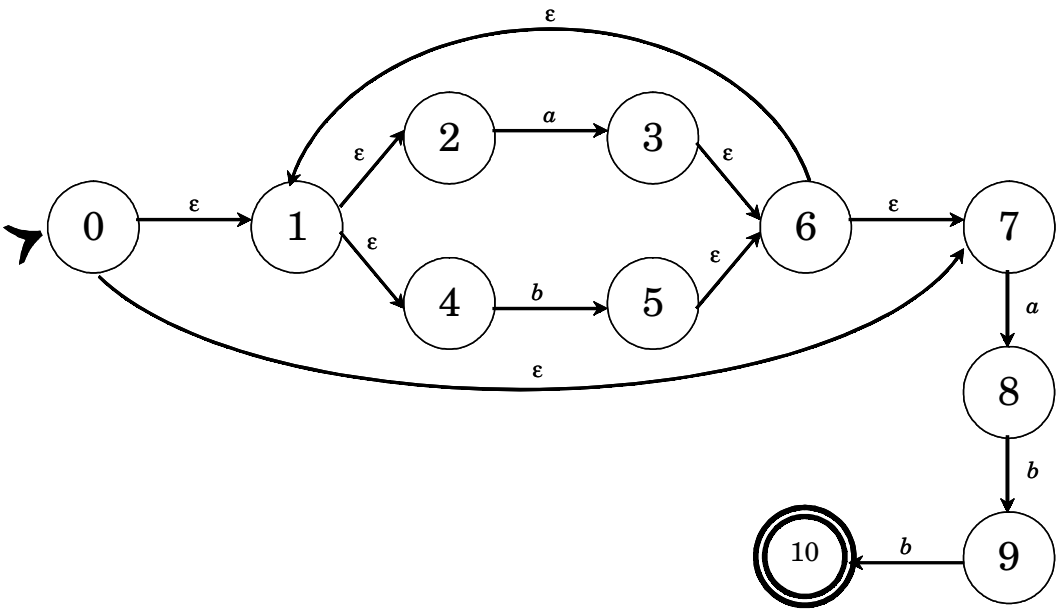
Sea ahora $w=ab$, $|w|=2$
ε-cl(0) = {0, 1} (ε-cl del símbolo inicial)
$\delta(\{0, 1\}, a) = \{3\}$
ε-cl({3})={1, 3}
$\delta(\{1, 3\}, b) = \{2, 4\}$
ε-cl({2, 4})={2, 4, 5}

$\{2, 4, 5\} \cap F \neq \varnothing$, luego acepta la cadena.

Algoritmo de paso de NFA con ε-transiciones a DFA.

Para pasar un NFA- ε a DFA, se sigue el siguiente algoritmo:

Para empezar, se hace la ε-cl del símbolo inicial, esto nos dará un conjunto, al que llamaremos, por ejemplo, S. Hay que marcar (véase def. en pág. 34) ese estado con los símbolos del alfabeto. A partir de aquí, cada vez que al marcar cada uno de los símbolos nos salga un conjunto que aun no tiene nombre, se aplica la ε-cl sobre éste, se le da un nombre, y se vuelve a marcar. Se considerarán estados de aceptación aquellos conjuntos que contengan algún estado de aceptación del NFA- ε inicial. El algoritmo para una vez estén marcados todos los estados nuevos de dicho autómata.

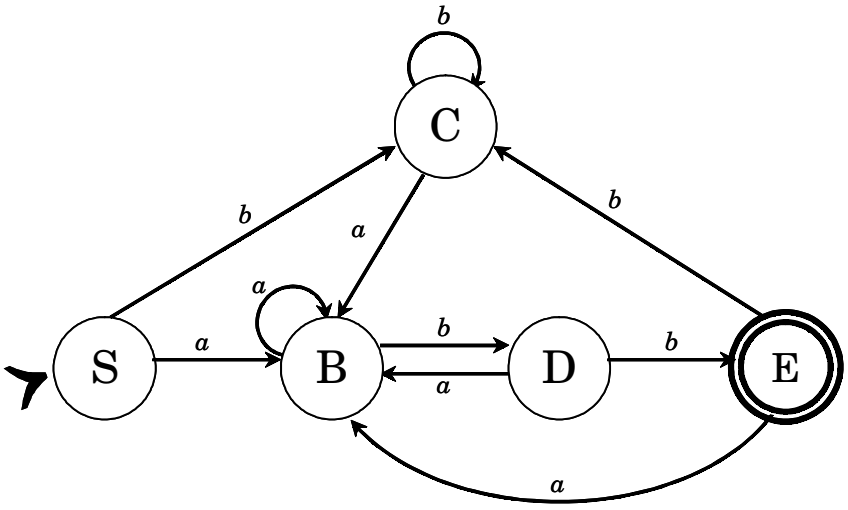


$\Sigma = \{a, b\}$

ε-cl(0)={0, 1, 2, 4, 7} = S (ε-cl del símbolo inicial)
Ahora marcamos este nuevo estado
δ (S, *a*)={3, 8}
δ (S, *b*)={5}
Ambos son conjuntos que aún no tienen nombre, hacemos sus ε-cl
ε-cl({3, 8})={1, 2, 3, 4, 6, 7, 8} = B
ε-cl({5})={1, 2, 4, 5, 6, 7} = C
Marcamos dichos conjuntos
δ (B, *a*)={3, 8} Este ya lo teníamos, es B, luego no hace falta volver a hacer su ε-cl.
δ (B, *b*)={5, 9} No lo tenemos, haremos su clausura.
δ (C, *a*)={3, 8}=B
δ (C, *b*)={5}=C
ε-cl({5, 9})={1, 2, 4, 5, 6, 7, 9}=D
δ (D, *a*)={3, 8}=B
δ (D, *b*)={5, 10} No está, hay que hacer su clausura.
ε-cl({5, 10})={1, 2, 4, 5, 6, 7, 10}=E
δ (E, *a*)={3, 8}=B
δ (E, *b*)={5}=C

Hemos terminado, ya que todos los estados están marcados.
E será el único estado de aceptación ya que es el único que contiene el estado 10, que era el estado de aceptación del NFA-ε.

Ahora ya sólo queda hacer el diagrama de transiciones:



Def: Se dice que un estado $q \in Q$ es importante si se da al menos una de de estas dos reglas:
1) $q \in F$
2) $\exists a \in \sum \mid \delta(q, a) \neq \varnothing$

P. ej., estados importantes del ejemplo anterior={2, 4, 7, 8, 9, 10}

Algoritmo de paso de NFA con ε-transiciones a NFA sin ε-transiciones.

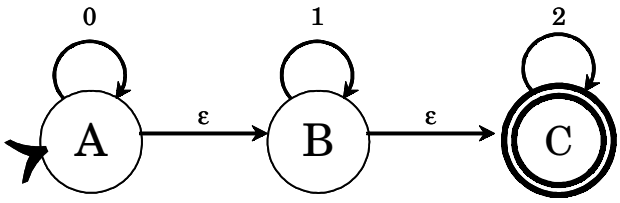
Teor: Dado un M NFA-ε $M \equiv (\sum, Q, F, q_0, \delta)$, $\exists$ NFA $M' \equiv (\sum, Q, F', q_0, \delta')$ | L(M) = L(M')

$F' = F \cup \{q \in Q \mid \varepsilon\text{-cl}(q) \cap F \neq \varnothing\}$
$\delta', \forall a \in \sum, \delta'(q, a) = \hat{\delta}(q, a)$

En este algoritmo se trata de aplicar la función $\hat{\delta}$ a cada estado con todos los símbolos $a \in \sum$. Cuando al aplicarla obtengamos un conjunto con n estados, significa que desde el estado al que estamos haciendo la $\hat{\delta}$, debemos poner una transición a cada uno de esos n estados que hemos obtenido al final.

Nota: $\hat{\delta}(q, a) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(q), a))$
$q \in Q$
$a \in \sum$

Ejemplo



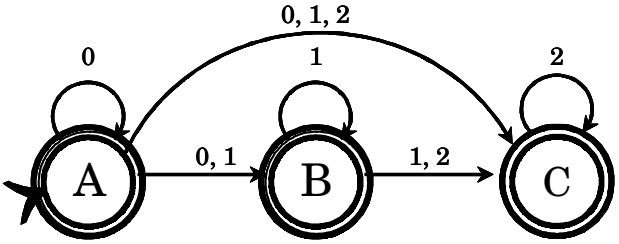
$\sum = \{0, 1, 2\}$

- $\hat{\delta}(A, 0) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(A), 0)) = \varepsilon\text{-cl}(\delta(\{A, B, C\}, 0)) = \varepsilon\text{-cl}(A) = \{A, B, C\}$

  Nota: Esto significa que desde el estado A, con el símbolo 0, hemos de añadir en el diagrama nuevo una transición a A, otra a B y otra a C

- $\hat{\delta}(A, 1) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(A), 1)) = \varepsilon\text{-cl}(\delta(\{A, B, C\}, 1)) = \varepsilon\text{-cl}(A) = \{B, C\}$
- $\hat{\delta}(A, 2) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(A), 2)) = \varepsilon\text{-cl}(\delta(\{A, B, C\}, 2)) = \varepsilon\text{-cl}(A) = \{C\}$
- $\hat{\delta}(B, 0) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(B), 0)) = \varepsilon\text{-cl}(\delta(\{B, C\}, 0)) = \varepsilon\text{-cl}(\varnothing) = \varnothing$
- $\hat{\delta}(B, 1) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(B), 1)) = \varepsilon\text{-cl}(\delta(\{B, C\}, 1)) = \varepsilon\text{-cl}(B) = \{B, C\}$
- $\hat{\delta}(B, 2) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(B), 2)) = \varepsilon\text{-cl}(\delta(\{B, C\}, 2)) = \varepsilon\text{-cl}(C) = \{C\}$
- $\hat{\delta}(C, 0) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(C), 0)) = \varepsilon\text{-cl}(\delta(\{C\}, 0)) = \varepsilon\text{-cl}(\varnothing) = \varnothing$
- $\hat{\delta}(C, 1) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(C), 1)) = \varepsilon\text{-cl}(\delta(\{C\}, 1)) = \varepsilon\text{-cl}(\varnothing) = \varnothing$
- $\hat{\delta}(C, 2) = \varepsilon\text{-cl}(\delta(\varepsilon\text{-cl}(C), 2)) = \varepsilon\text{-cl}(\delta(\{C\}, 2)) = \varepsilon\text{-cl}(C) = \{C\}$

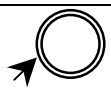
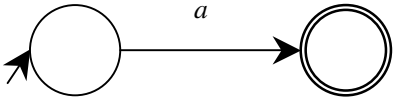
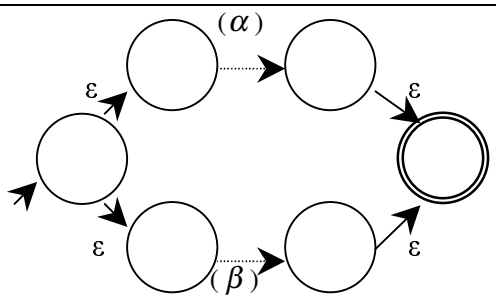
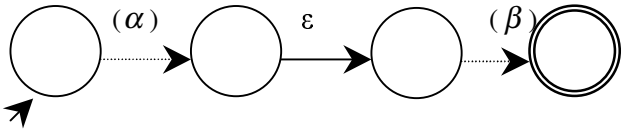
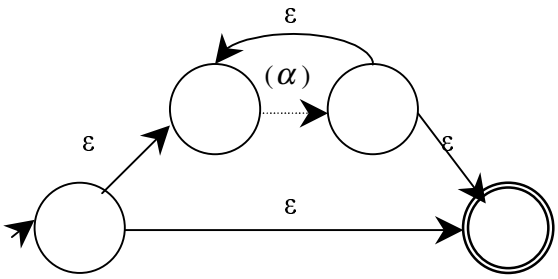
Serán estados de aceptación aquellos estados que lo fueron en el NFA-ε, y además todos aquellos cuya ε-cl contenga algún estado de aceptación de ese NFA-ε, en este caso los tres estados serán de aceptación. El diagrama del NFA sin ε-transiciones queda como sigue:



Construcción de Thompson.

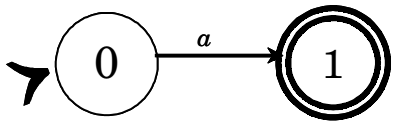
Este algoritmo nos permite convertir una expresión regular en un NFA-ε. Para ello, se siguen las siguientes reglas:

Sean α, β expresiones regulares.

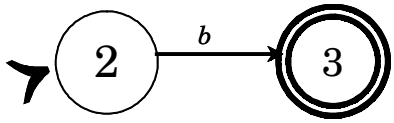
Expresión Regular	Autómata finito
$\varnothing$	
ε	
$a \in \Sigma$	
$\alpha \mid \beta$	
$\alpha \cdot \beta$	
α^*	

Ejemplo:
Sea $\alpha = ba^*$

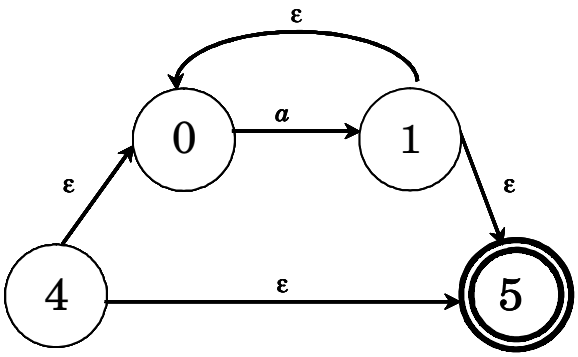
a



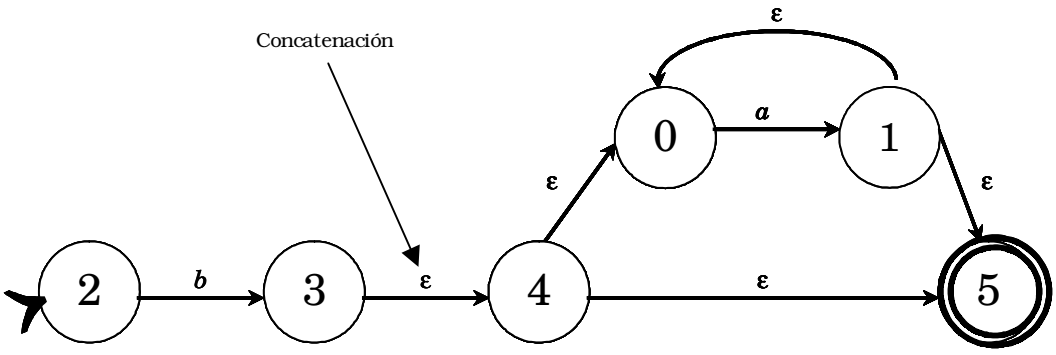
b



*a** (nótese que en este paso el estado 1 deja de ser de aceptación)



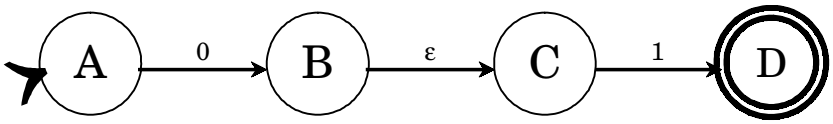
*b· a**



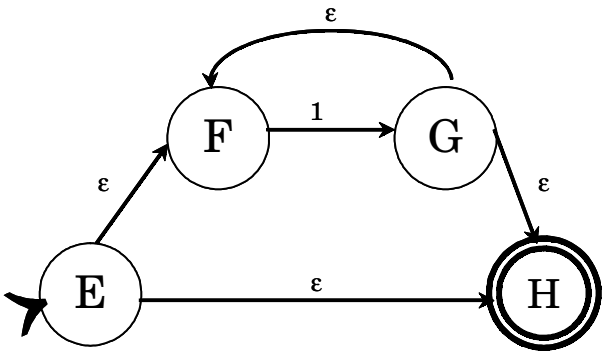
Otro ejemplo:

$\alpha = 01|1^*$

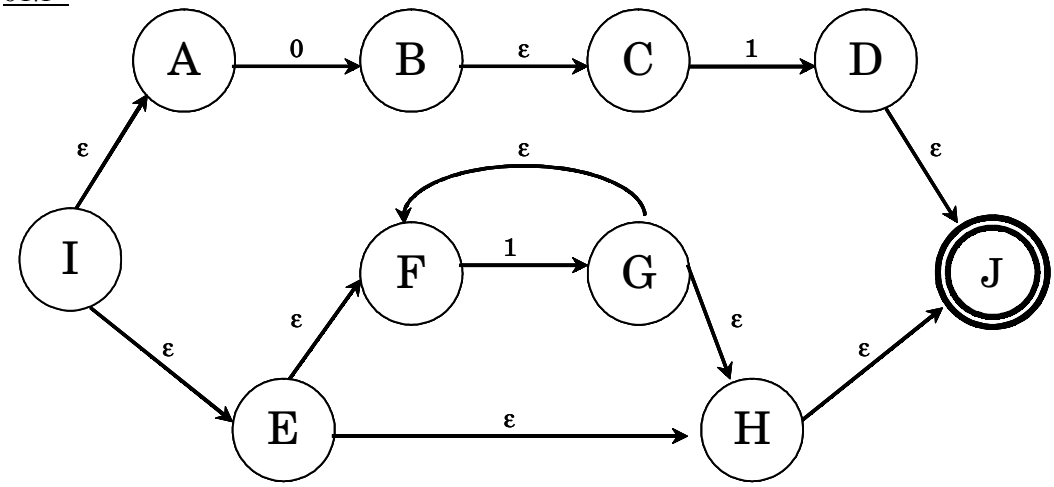
0· 1 (En este paso haremos directamente la concatenación, que es entre los estados B y C)



1*



01|1*



Paso de NFA a Expresión Regular (Lema de Arden).

Al igual que es posible pasar de expresión regular a NFA, como hemos visto con el algoritmo de Thompson, es posible hacer el paso contrario, es decir, pasar de NFA a Expresión Regular. La descripción es así:

Sea un NFA $M \equiv (\sum, Q, F, q_0, \delta)$. Sea $q_i \in Q$,

$A_i = \{w \in \sum^* | \delta(q_i, w) \cap F \neq \varnothing\}$ (se definen así las cadenas aceptadas por q_i)

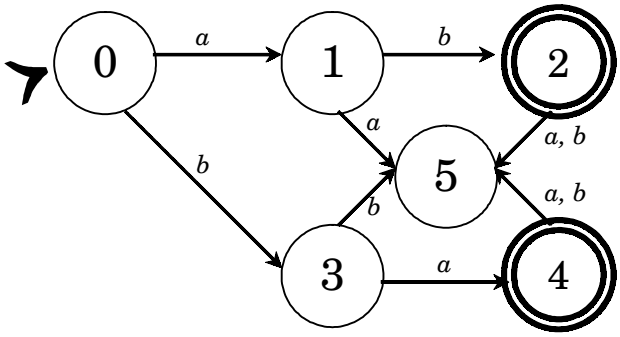
Hay 3 reglas:

1) $A_0 = L(M)$

2) Es posible que $A_i = \varnothing$

3) Si $q_i \in F$, $\varepsilon \in A_i$

Ejemplo:



$A_0 = aA_1 \cup bA_3$
$A_1 = aA_5 \cup bA_2$
$A_2 = aA_5 \cup bA_5 \cup \varepsilon$
$A_3 = aA_4 \cup bA_5$
$A_4 = aA_5 \cup bA_5 \cup \varepsilon$
$A_5 = \varnothing$

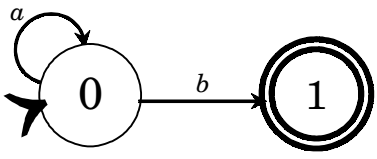
Regla 3, Si $q_i \in F$, $\varepsilon \in A_i$, es decir, como es de aceptación, se añade ε

Como se dice en la Regla 1, el objetivo de estos sistemas es despejar A_0, lo que nos dará la expresión regular del NFA.

Como $A_5 = \varnothing$,
$A_4 = aA_5 \cup bA_5 \cup \varepsilon = a\varnothing \cup b\varnothing \cup \varepsilon = \varnothing \cup \varnothing \cup \varepsilon = \varepsilon$
$A_3 = aA_4 \cup bA_5 = a\varepsilon \cup b\varnothing = a$
$A_2 = aA_5 \cup bA_5 \cup \varepsilon = A_4$
$A_1 = aA_5 \cup bA_2 = a\varnothing \cup b\varepsilon = b$
$A_0 = aA_1 \cup bA_3 = ab \cup ba.$

Luego, como vemos, efectivamente L(M)=(*ab*|*ba*)

Otros ejemplos de esta técnica



En este ejemplo,
$A_0 = aA_0 \cup bA_1$
$A_1 = \varepsilon$

Luego, $A_0 = aA_0 \cup b\,\varepsilon = aA_0 \cup b$???

En principio este sistema nos ha dado una expresión sin solución, ya que la función A_0 depende de sí misma. Estos casos tienen una única solución, y se aplica el llamado Lema de Arden.

Teor: Lema de Arden

$X = AX \cup B$, donde $\varepsilon \notin A$, tiene como única solución $X = A^*B$

Con este lema, podemos afirmar que en el anterior ejemplo $A_0 = aA_0 \cup b$, la solución es: $A_0 = a^*b$, luego $L(M) = L(a^*b)$

Dem: Demostración del Lema de Arden.

$$X = AX \cup B \overset{!}{\Leftrightarrow} X = A^*B$$

$\Leftarrow$) $X = A^*B = (A^+ \cup \varepsilon)B = A^+B \cup B = AA^*B \cup B = A(A^*B) \cup B$, con lo cual $X = A^*B$ □

$\Rightarrow$) La solución es única. Haremos esta parte por reducción al absurdo.

Sup. que $X = A^*B \cup C$ es solución con $A^*B \cap C = \varnothing$

$X = AX \cup B$

$A^*B \cup C = A(A^*B \cup C) \cup B = AA^*B \cup C \cup B = A^+B \cup AC \cup B = A^+B \cup B \cup AC = (A^+ \cup \varepsilon)B \cup AC = A^*B \cup AC$.

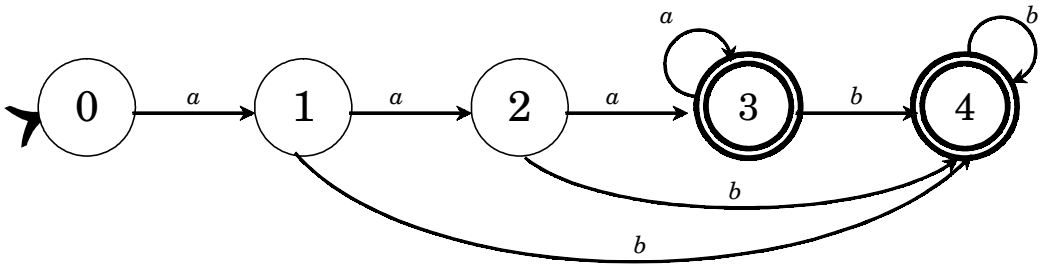
Con lo cual vemos que $A^*B \cup C = A^*B \cup AC$

$C \cap (A^*B \cup C) = C \cap (A^*B \cup AC)$

$C = C \cap AC \Rightarrow C \subseteq AC$

$\varepsilon \notin A$. Es absurdo, la cadena más corta de AC ha de ser más larga que la cadena más corta de C. Esto sólo es posible si $C = \varnothing$, con lo cual hemos llegado al absurdo por suponer que había otra solución que no fuese $X = A^*B$, con lo cual ésta es única □

Otro ejemplo



$A_0 = aA_1$

$A_1 = aA_2 \cup bA_4$

$A_2 = aA_3 \cup bA_4$
$A_3 = aA_3 \cup bA_4 \cup \varepsilon$
$A_4 = bA_4 \cup \varepsilon$

Solución:

$A_4 = bA_4 \cup \varepsilon \overset{Arden}{=} b^* \varepsilon = b^*$
$A_3 = aA_3 \cup bA_4 \cup \varepsilon = aA_3 \cup (bb^* \cup \varepsilon) = aA_3 \cup b^* = a^*b^*$
$A_2 = aA_3 \cup bA_4 = a(a^*b^*) \cup bb^* = a^+b^* \cup b^+$
$A_1 = aA_2 \cup bA_4 = a(a^+b^* \cup b^+) \cup bb^* = aa^+b^* \cup ab+ \cup b^+$
$A_0 = aA_1 = a(aa^+b^* \cup ab^+ \cup b^+) = aaa^+b^* \cup aab^+ \cup ab^+$

Luego, $L(M) = L(aaa^+b^* \cup aab^+ \cup ab^+)$

Lema: Sea M un Autómata Finito, $\exists$ una E.R. $r \mid L(M)=L(r)$.

Teor: Teorema de Kleene
Un lenguaje es regular sii es aceptado por el autómata finito.

Lema del Bombeo (Pumping Lemma).

Este lema nos permitirá demostrar que un lenguaje no es regular.

$$\forall L \subseteq \sum^*, \exists n \in N \mid \text{si } w \in \sum^*, w \in L \text{ con } |w| \geq n \Rightarrow \begin{cases} 1) w = xyz \\ 2) \mid xy \mid \leq n \\ 3) \mid y \mid \geq 1 \\ 4) xy^y z \in L, \forall k \in N \end{cases}$$

Teor: Si un lenguaje cumple el lema del bombeo, PUEDE que sea regular y puede que no. Si un lenguaje no lo cumple, entonces es únicamente cuando podemos afirmar que no es regular.

Ejemplo de lenguaje no regular

Dem:

$L=\{a^m b^m \mid m \geq 0\}$

Sea $n \in N$
(Nota: ¡n no se debe concretar!)

Sea $w = a^n b^n$.

(Nota: Hay que tener mucho cuidado con elegir la cadena)

$|w| = 2n \geq n$.

Si $w \in$ L, ha de ocurrir:
1) $w = xyz$
2) $|y| \geq 1$
3) $|xy| \leq n$
4) $xy^k z \in$ L, $\forall k \in$ N

Si $|xy| \leq n \Rightarrow$ y sólo contiene símbolos "a", sea entonces $y = a^s$, si $|y| \geq 1 \Rightarrow s \geq 1$

$w = a^n b^n = \underbrace{aaa...a}_{n} \underbrace{bbb...b}_{n} = xyz$
Entonces sea $x = a^r$, $y = a^s$, y por tanto $z = a^{n-(r+s)} b^n$

Sea $k = 2$, luego $xy^2z = a^r a^{2s} a^{n-(r+s)} b^n = a^{n+s} b^n$. Como $s \geq 1 \Rightarrow xy^2z \notin$ L

Con lo cual, L no es regular.□

Otro ejemplo.

Dem:

Sea L = $\{a^{i^2} \mid i \geq 1\}$

L = {aa, $aaaa$, $aaaaaaaaa$, $aaaaaaaaaaaaaaaa$, ...}

Sea $n \in$ N
Sea $z \in$ L; $z = a^{n^2}$; $|z| = n^2 \geq n$, $\forall n$.

1) $z = uvw$
2) $|v| \geq 1$
3) $|uv| \leq n$
4) $uv^r w \in$ L, $\forall r \in$ N

Si $|uv| \leq n \Rightarrow |v| \leq n$, con lo cual si $|v| \geq 1$, $\quad 1 \leq |v| \leq n$

$n \leq n^2 = |uvw| < |uv^2w| \leq n^2 + n < n^2 + 2n + 1 = (n+1)^2$

Si $r = 2$, $n^2 < |uv^2w| < (n+1)^2 \Rightarrow uv^2w \notin$ L, al estar entre dos cuadrados perfectos

Luego, L no es regular□

Teor: Sea M un autómata finito con k estados. L(M) es distinto de vacío si y sólo si M acepta una cadena de longitud menor que k.

Teor: Sea M un autómata finito con k estados. L(M) es infinito si y sólo si M

acepta una cadena de longitud m con $k \leq m < 2k$

Teor: Sea $L \subseteq \sum^*$, si L es regular $\Rightarrow \overline{L}$ es regular.

Teor: Si L_1 y L_2 son regulares $\Rightarrow L_1 \cup L_2$ es regular

Teor: Si L_1 y L_2 son regulares $\Rightarrow L_1 \cap L_2$ es regular

Teor: Si L_1 y L_2 son regulares $\Rightarrow L_1 \cdot L_2$ es regular

Teor: Si L es regular $\Rightarrow L^*$ es regular

Tema 4

Gramáticas. Lenguajes independientes del contexto.

Gramáticas.

Def: Una gramática es una cuaterna formada de la siguiente manera:
$G \equiv (V, \sum, S, P)$

$\sum$ es el alfabeto, o también llamado conjunto de símbolos terminales, se representan por letras minúsculas

$V \neq \varnothing$, $V \cap \sum = \varnothing$ es el conjunto de símbolos no terminales, se representan por letras mayúsculas

S es el símbolo de arranque, $S \in V$

P es el conjunto de producciones de la gramática

$P \subseteq V \times (\sum \cup V)^*$

Estas producciones se representan del siguiente modo

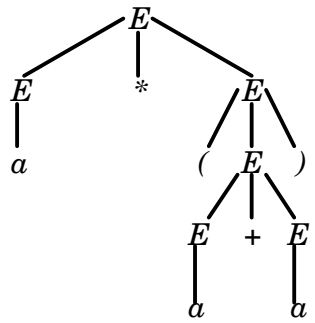
$X \rightarrow a$, con $X \in V$, $a \in \sum$, y se lee, "X produce a"

Ejemplo:

1) $E \rightarrow E + E$
2) $E \rightarrow E * E$
3) $E \rightarrow (E)$
4) $E \rightarrow a$

S=E
V={E}
P={1), 2), 3), 4)}
$\sum$={+, *, (,), *a*}

Ej. Encontrar alguna manera de llegar a *w*=*a**(*a*+*a*) con la anterior gramática.



Vemos que los nodos hoja corresponden a lo que nos pedía el ejercicio, luego eso sería una manera de llegar a la cadena *w*.

La anterior gramática también se puede escribir del siguiente modo:

$E \rightarrow E+E \mid E*E \mid (E) \mid a$

Teor: Las producciones son reglas de reescritura, secuencias de derivaciones

Def: Se define una derivación del siguiente modo:
Sea $\alpha A \beta$

Sea por ejemplo, $\alpha = ab\mathrm{B}c\mathrm{D}d\mathrm{E}$, y $\beta = abbc\mathrm{DDE}$
$a \in (\sum \cup V)^*$

Si $(A \rightarrow \gamma) \in P$, $\alpha A \beta \overset{derivaen}{\Rightarrow} \alpha \gamma \beta$

P.ej:
$\alpha \overset{n}{\Rightarrow} \beta \longrightarrow \alpha \overset{1}{\Rightarrow} \gamma \overset{n-1}{\Rightarrow} \beta$

Notación: $\alpha \overset{*}{\Rightarrow} \beta$ se lee: α en una serie de derivaciones deriva en β.

Notación: Se entiende CFG como Context Free Grammar, con su equivalente en castellano como Gramática independiente del contexto.

Teor: Dada una CFG, G

$G \equiv (V, \sum, S, P)$, $L(G)=\{w \in \sum^* \mid S \overset{*}{\Rightarrow} w\}$

P. ej.:
En la gramática de antes:
$E \Rightarrow a$
$E \Rightarrow (E) \Rightarrow (a)$
$E \Rightarrow (E) \Rightarrow ((E)) \Rightarrow ((a))$ $L(G)=\{a, (a), ((a)), a+a, ...\}$
$E \Rightarrow E+E \overset{*}{\Rightarrow} a+a$

Ejemplos de gramáticas finitas

$S \rightarrow t$
$L(G)=\{t\}$

$S \rightarrow Sa$
$S \Rightarrow Sa \Rightarrow Saa \Rightarrow Saaa \Rightarrow Saaaa$ ...
$L(G)=\varnothing$, ya que esta gramática nunca "para".

$S \rightarrow Sa \mid e$
$L(G)=ea^*$

Def: Una gramática es lineal por la derecha si es de la forma:
$A \rightarrow uB \mid v$
$A, B \in V$
$u, v \in \Sigma$

Def: Una gramática es lineal por la izquierda si es de la forma:
$A \rightarrow Bu \mid v$
$A, B \in V$
$u, v \in \Sigma$

Def: Una gramática es regular si es lineal por la izquierda o por la derecha.

P.ej.:
$S \rightarrow 0A$
$A \rightarrow 10A \mid \varepsilon$

L(G)=L(0(10)*)

Otro ej.:
$S \rightarrow S10 \mid 0$
$S \Rightarrow 0$
$S \Rightarrow S10 \Rightarrow 010$
$S \Rightarrow S1010 \Rightarrow 01010$

L(G)=L(0(10)*)

Teor: Dos gramáticas son equivalentes si los lenguajes que generan son el mismo, es decir, $G_1 \equiv G_2 \Leftrightarrow L(G_1) = L(G_2)$. Vemos que esto ocurre en los dos ejemplos anteriores.

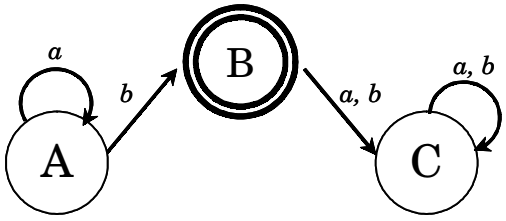
Aclaración: Una autómata RECONOCE lenguajes, una gramática los GENERA.

Equivalencia entre gramáticas regulares y lenguajes regulares.

Teor: Sea L regular $\Rightarrow \exists G \mid L(G) = L$
Si L regular $\Rightarrow \exists$ DFA $M \mid L = L(M)$
$M \equiv (\Sigma, Q, F, q_0, \delta)$

$V = Q$
$\Sigma = \Sigma$
$S = q_0$
$P = \{(q, ap) \mid \delta(q, a)=p\} \cup \{(q, \check{\varepsilon}) \mid q \in F\}$

Ejemplo:



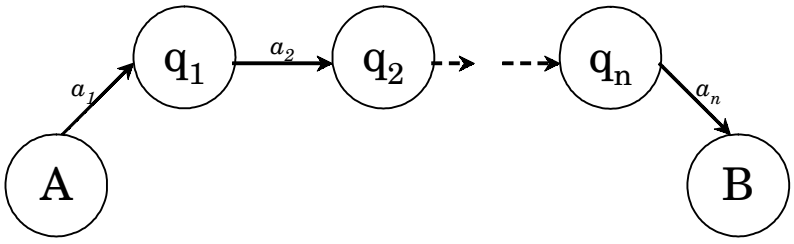
El paso a gramática sería así:
V={A, B, C}
$\sum = \{a, b\}$
S=A
P:
$A \rightarrow aA \mid bB$
$B \rightarrow aC \mid bC \mid \varepsilon$ (por ser de aceptación)
$C \rightarrow aC \mid bC$

Teor: Dada una gramática regular, también se puede construir el autómata.
Dada G regular $\Rightarrow \exists$ NFA M | L(M)=L(G)

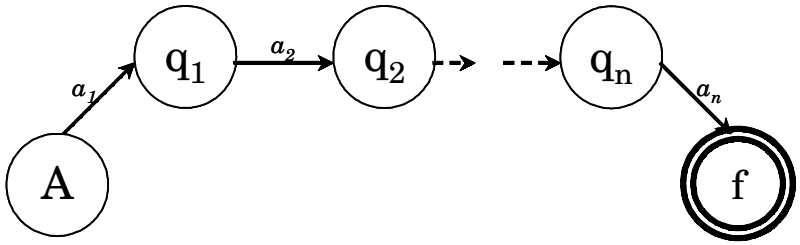
$G \equiv (V, \sum, S, P)$
NFA, $M \equiv (\sum, Q, F, q_0, \delta)$

$Q = V \cup \{f\}$, donde "f" es un estado nuevo
$\sum = \sum$
F={f}
q_0=S

La función δ se construye del siguiente modo:
Si $A \rightarrow (a_1 a_2 ... a_n B) \in P \Rightarrow Q = Q \cup \{q_1, q_2 ... q_{n-1}\}$
Y construímos las transiciones:
$\delta(A, a_1 a_2 ... a_n) = \delta(q_1, a_2 a_3 ... a_n) = ... = \delta(q_{n-1}, a_n) = B$
Interpretación gráfica:



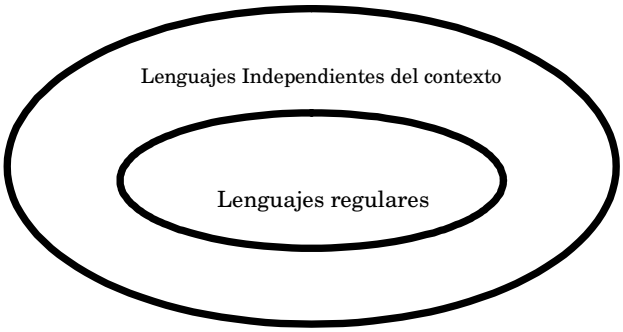
Si $A \rightarrow (a_1 a_2 ... a_n) \in P \Rightarrow Q=Q \cup \{ q_1, q_2 ... q_{n-1} \}$, y construimos transiciones:
δ (A, $a_1 a_2 ... a_n$)= δ ($q_1, a_2 a_3 ... a_n$)= ... = δ (q_{n-1}, a_n)=f
Interpretación gráfica:



Def: Las gramáticas independientes del contexto son un caso particular de las anteriores. Se definen así:

G≡(V, $\sum$, S, P)
A $\rightarrow \alpha$
A$\in$ V, $\alpha \in (\sum \cup V)^*$

P. ej.:
S $\rightarrow a S b \mid \varepsilon$ L(G)=$\{a^n b^n \mid n \geq 0\}$
Como vimos en el tema pasado, por el lema del bombeo demostramos que no es regular, sin embargo ahora hemos dado una gramática que genera este lenguaje, de lo cual concluimos que el conjunto de los lenguajes independientes del contexto es un superconjunto del de los regulares.



Árboles de Análisis Sintáctico (AAS).

Los árboles de análisis sintáctico (AAS) son una representación gráfica de una derivación.

Propiedades:

1) Nodo raíz = S 3) Nodos intermedios A$\in$ V

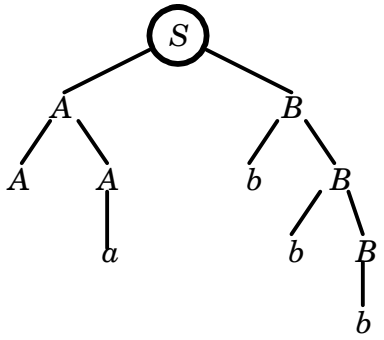
2) Hojas: $\sum \cup \{ \varepsilon \}$
P. ej.:

$S \rightarrow AB$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

Sea por ejemplo $w = aabbb$.
Con derivaciones podríamos llegar, por ejemplo, de este modo:

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow aabB \Rightarrow aabbB \Rightarrow aabbb$

Con el AAS sería del siguiente modo:



Derivaciones más a la izquierda y derivaciones más a la derecha.

En una gramática, se denomina "Derivaciones más a la izquierda" o "más a la derecha" a susttuir el símbolo no terminal que está más a la izquierda, o derecha, respectivamente.

Las derivaciones más a la izquierda se denominan Canónicas

No es situación normal que dada una cadena tenga siempre el mismo AAS.

P.ej.:

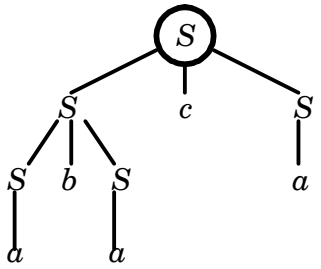
$S \rightarrow SbS \mid ScS \mid a$

Sea $w = abaca$, vamos a hacerlo mediante derivaciones más a la derecha.

$S \overset{m.d.}{\Rightarrow} ScS \overset{m.d.}{\Rightarrow} Sca \overset{m.d.}{\Rightarrow} SbSca \overset{m.d.}{\Rightarrow} Sbaca \overset{m.d.}{\Rightarrow} abaca$

Como vemos, siempre se sustituye el símbolo que está más a la derecha hasta formar la cadena, que es w.

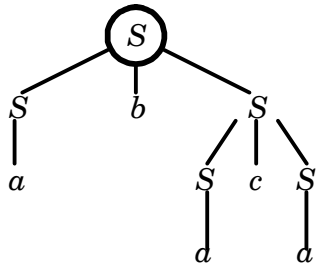
El AAS sería del siguiente modo:



Vamos a hacerlo ahora con derivaciones más a la izquierda

$$S \overset{m.i.}{\Rightarrow} Sb\overset{m.i.}{S} \Rightarrow a\overset{m.i.}{bS} \Rightarrow ab\overset{m.i.}{S}cS \Rightarrow abac\overset{m.i.}{S} \Rightarrow abaca$$

El AAS queda así:



Como vemos, los AAS no son iguales.

Def: Se dice que una gramática es ambigua si una cadena tiene más de un árbol de análisis sintáctico distinto. Si existe una cadena con más de una derivación a la izquierda o a la derecha, también es ambigua.

Def: Un lenguaje es intrínsecamente ambiguo si no existe ninguna gramática no ambigua que lo genere.

Ejercicio
Ver si la siguiente gramática es ambigua o no, si $w = a{:=}b{+}c{*}a$
$\sum = \{a, b, c, +, *, (,), :=\}$
Los símbolos de este alfabeto son llamados tokens.

P
$S \rightarrow I{:=}E$
$I \rightarrow a \mid b \mid c$
$E \rightarrow E + E \mid E * E \mid (E) \mid I$

$V=\{S, I, E\}$

Hagamos derivaciones más a la izquierda para obtener la cadena w

$S \overset{m.i.}{\Rightarrow} I:=E \overset{m.i.}{\Rightarrow} a:=E \overset{m.i.}{\Rightarrow} a:=E+E \overset{m.i.}{\Rightarrow} a:=b+E \overset{m.i.}{\Rightarrow} a:=b+E*E \overset{*}{\Rightarrow} a:=b+c*a$

Hagamos ahora otras derivaciones,

$S \Rightarrow I:=E \Rightarrow a:=E \Rightarrow a:=E*E \Rightarrow a:=E+E*E \Rightarrow a:=I+E*E \overset{*}{\Rightarrow} a:=b+c*a$

Hemos encontrado dos distintas derivaciones, con lo cual queda demostrado que dicha gramática es ambigua.

Simplificación de gramáticas.

Ejemplos de gramáticas mal escritas

$S \rightarrow A$
$A \rightarrow B$
$B \rightarrow C$ $\equiv$ $S \rightarrow a|S$
$C \rightarrow D$
$D \rightarrow a|A$

Eliminación de símbolos inútiles

Este algoritmo se divide en dos etapas, que hay que aplicar en ese orden.

Ej
$S \rightarrow Aa \mid B \mid D$
$B \rightarrow b$
$A \rightarrow aA \mid bA \mid B$
$C \rightarrow abd$

1ª etapa
Creamos un nuevo conjunto llamado V' , que en principio es vacío.
$V' = \{ \varnothing \}$

Como primer paso, metemos en el conjunto todos aquellos no terminales que producen SÓLO terminales, en alguna de sus producciones. Vemos que es el caso de de B y C, luego,
$V' = \{B, C\}$

Como segundo paso, metemos todos aquellos no terminales que en alguna de sus producciones tengan algún terminal. Es el caso de S ($S \rightarrow aA$) y A ($A \rightarrow aA \mid bA$).

$V' = \{B, C, S, A\}$

Como último paso, hay que eliminar todas aquellas producciones que no estén en el conjunto V', vemos que en este caso únicamente es D, que está en la producción S. Esto ocurre ya que desde S se puede llegar a D, pero la producción D no está, luego sobra. De momento la gramática nos queda:

$S \rightarrow Aa \mid B$
$B \rightarrow b$
$A \rightarrow aA \mid bA \mid B$
$C \rightarrow abd$

2ª etapa
Definimos un conjunto J que será el de variables a analizar. Inicialmente contiene el símbolo S. Definimos el conjunto V'', que también contiene inicialmente el símbolo S. Y sea T' el conjunto de símbolos terminales.

$J=V''=\{S\}$
$T'=\{\varnothing\}$

Paso 1:
Sacamos un no terminal de J, en este caso únicamente podemos sacar S.
Luego vamos a analizar S, $S \rightarrow Aa \mid B$.
Para cada producción de S, añadimos a V'' todos los no terminales de dicha producción, y también añadimos a J dichos no terminales, ya que los analizaremos. En este caso:
$V''=\{S, A, B\}$, $J=\{A, B\}$
También añadimos a T' los terminales que haya en dicha producción, en este caso:
$T'=\{a\}$

Paso 2:
Repetir paso 1. Sacamos de J un no terminal, A por ejemplo. $A \rightarrow aA \mid bA \mid B$
Vemos que todos los no terminales de A, ya están en V'', luego no hacemos cambios. Y también están o estuvieron en J, pero vemos que para el conjunto T', aparece el terminal "b", que no estaba. Lo añadimos.
$V''=\{S, A, B\}$; $J=\{B\}$; $T'=\{a, b\}$
Paso 3:
Sacamos de J otro no terminal, el único que nos queda, B. $B \rightarrow b$
Evidentemente, como no tiene producciones de no terminales, J se queda vacío, y V'' no sufre cambios. Y tampoco T', ya que la "b" ya estaba.

En este momento, como tenemos $J=\{\varnothing\}$, paramos. Hemos de eliminar todas las variables que no estén en V'' y todos los terminales que no estén en T', y aquellas producciones que contengan alguna variable que no esté en V'' o T'.
En esta caso, Como la C no está en V'', la eliminamos. Y como la "d" no está en T', eliminamos dicha producción. La gramática queda:

$S \rightarrow Aa \mid B$
$B \rightarrow b$
$A \rightarrow aA \mid bA \mid B$

Otro ejemplo

$S \rightarrow AB \mid a$
$A \rightarrow a$

1ª etapa
$V'=\{\varnothing\}$
Metemos las variables que generen algún terminal.
$V'=\{S, A\}$
Vemos que B no está, eliminamos toda la producción donde se encuentra esa variable.
La gramática queda:
$S \rightarrow a$
$A \rightarrow a$

2ª etapa
$J=V''=\{S\}$
$T'=\{\varnothing\}$

Paso 1:
Analizando S. $S \rightarrow a$
$J=\{\varnothing\}$
$V''=\{S\}$
$T'=\{a\}$

Como $J=\{\varnothing\}$, paramos. Vemos que $A \notin V''$, eliminamos la producción.
La gramática queda:

$S \rightarrow a$

Eliminación de producciones vacías.
Antes de empezar a explicar este algoritmo, definamos dos términos.

Def: Se llama producción nula a aquella producción del tipo $A \rightarrow \varepsilon$.

Def: Un símbolo no terminal es anulable si $A \overset{*}{\Rightarrow} \varepsilon$

Ejemplo del algoritmo
$S \rightarrow aA$
$A \rightarrow aA \mid \varepsilon$

Sea H el conjunto de símbolos anulables. Siguiendo las definiciones anteriores, A es anulable, ya que $A \rightarrow \varepsilon$. Luego $H=\{A\}$

Luego para cada elemento de H, deberemos quitar ε, esto se hace del siguiente modo:

$S \rightarrow aA$ (se queda igual)
$A \rightarrow aA \mid a$

Otro ejemplo
$S \rightarrow Abb \mid ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$
$C \rightarrow abC \mid AB$

El conjunto de símbolos anulables en este caso es:

H={A, B, C, S}

Nota: Si S es anulable, $\varepsilon \in L(G)$

Luego, la gramática nos queda:

$S \rightarrow ABb \mid Ab \mid Bb \mid b$
$S \rightarrow ABC \mid AB \mid AC \mid BC \mid A \mid B \mid C$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$
$C \rightarrow abC \mid ab$
$C \rightarrow AB \mid A \mid B$
$S \rightarrow \varepsilon$ (ya que S es anulable)

Eliminación de producciones unitarias

Una producción es unitaria si es del tipo:

$A \rightarrow B$
$B \rightarrow w|c$

En realidad debería ser: $A \rightarrow w|c$. Veamos cómo eliminar eso.

Ejemplo:

$S \rightarrow A|Aa$
$A \rightarrow B$
$B \rightarrow C|b$
$C \rightarrow D|ab$
$D \rightarrow b$

Llamaremos a H al conjunto de las parejas que están compuestas por un no terminal que produce y otro producido, es decir, si por ejemplo $A \rightarrow B$, en el conjunto deberíamos añadir el par (A, B). Por esta regla, tenemos:

H={(S, A), (A, B), (B, C), (C, D)}

Ahora tenemos que tener en cuenta todos los pares de la forma (A, B) y (B, C), entonces debemos añadir el par (A, C), y así sucesivamente. Luego,

H={(S, A), (A, B), (B, C), (C, D), (S, B), (S, C), (S, D), (A, C), (A, D), (B, D)}

Como último paso tengamos en cuenta la gramática que nos dan.
Debemos escribir una segunda gramática que tan sólo contenga aquellas producciones donde aparecen terminales, es decir:

$S \rightarrow Aa$
$B \rightarrow b$
$C \rightarrow ab$
$D \rightarrow b$

Y ahora, con cada uno de los pares de H, formamos la gramática final. Lo hacemos mirando los dos símbolos del par, es decir, si tenemos (S, B), deberemos añadir una producción $S \rightarrow B$, mirando antes en la anterior gramática qué es lo que produce B, y vemos que en este caso, $B \rightarrow b$, luego la producción sería $S \rightarrow b$. Al final, la gramática queda:

$S \rightarrow Aa \mid b \mid ab$
$B \rightarrow b \mid ab$
$C \rightarrow ab \mid b$
$D \rightarrow b$
$A \rightarrow b$ (no sufre modificación)

Teor: Cuando haya que limpiar una gramática, el orden a seguir será siempre éste:

1) Eliminar producciones inútiles
2) Eliminar producciones vacías
3) Eliminar producciones unitarias
4) Eliminar producciones inútiles

Algoritmo de Eliminación de recursividad indirecta.

Ejemplo
$S \rightarrow Aa \mid b$
$A \rightarrow Ac \mid Sd \mid \varepsilon$

Como vemos $S \rightarrow A$, y $A \rightarrow S$, luego tenemos recursividad indirecta.

En este caso habría que sustituír la producción $A \rightarrow Sd$ de tal manera que se elimine la recursividad y a la vez no se modifique el lenguaje que genera dicha gramática. En principio dejamos la producción S igual, y convertimos la producción Sd a lo siguiente:

Si $S \rightarrow Aa \mid b$,
$Sd = Aad \mid bd$ (se sustituye la S por todas las producciones de S, concatenadas con la d que ya estaba).

Luego, la gramática queda:
$S \rightarrow Aa \mid b$
$A \rightarrow Ac \mid Aad \mid bd \mid \varepsilon$

Equivalente a eliminación de recursividad por la derecha:

Sabiendo que:
$A \rightarrow \beta T \mid \beta$
$T \rightarrow \alpha T \mid \alpha$

$S \rightarrow Aa \mid b$
$A \rightarrow bdT \mid \varepsilon T \mid bd \mid \varepsilon$
$T \rightarrow cT \mid adT \mid c \mid ad$

Forma normal de Greibach.

Si todas las producciones de la gramática son del tipo
$A \rightarrow a\alpha$, con $a \in \sum$, $A \in V$, $\alpha \in V^*$.

- No puede ser recursiva por la izquierda
- No puede generar lenguajes vacíos

La forma de conversión es la siguiente:
Si tenemos,

$A \rightarrow BB \mid a$
$B \rightarrow AA \mid b$

Se convierte en:

$A_1 \rightarrow A_2 A_2 \mid a$
$A_2 \rightarrow A_1 A_1 \mid b$

$A_1 \rightarrow A_2 A_2 \mid a$
$A_2 \rightarrow A_2 A_2 A_1 \mid aA_1 \mid b$

$A_1 \rightarrow A_2 A_2 \mid a$
$A_2 \rightarrow aA_1 A_3 \mid bA_3 \mid aA_1 \mid b$
$A_3 \rightarrow A_2 A_1 A_3 \mid A_2 A_1$

Ahora hay que poner A_3 en términos de A_2

$A_1 \rightarrow A_2 A_2 \mid a$
$A_2 \rightarrow aA_1 A_3 \mid bA_3 \mid aA_1 \mid b$
$A_3 \rightarrow aA_1 A_3 A_1 A_3 \mid bA_3 A_1 A_3 \mid aA_1 A_1 A_3 \mid bA_1 A_3 \mid aA_1 A_3 A_1 \mid bA_3 A_1 \mid aA_1 A_1 \mid bA_1$

Y por último, reescribir A_1.

$A_1 \rightarrow a \mid aA_2 A_3 A_2 \mid bA_3 A_2 \mid aA_2 A_2 \mid bA_2$

$A_2 \rightarrow aA_1A_3 \mid bA_3 \mid aA_1 \mid b$

$A_3 \rightarrow aA_1A_3A_1A_3 \mid bA_3A_1A_3 \mid aA_1A_1A_3 \mid bA_1A_3 \mid aA_1A_3A_1 \mid bA_3A_1 \mid aA_1A_1 \mid bA_1$

Def: Una gramática es recursiva si desde un símbolo no terminal A puedo volver a llegar a A, es decir:

$\exists A \in V \mid A \overset{*}{\Rightarrow} \alpha A \beta \ , \ \alpha, \beta \in (\sum \cup V)^*$

Teor: La recursividad es la que permite generar lenguajes infinitos.

Forma normal de Chomsky.

Hay que suponer que se parte con una gramática "limpia"
Ej.:
$S \rightarrow bA \mid aB$
$A \rightarrow bAA \mid aS \mid a$
$B \rightarrow aBB \mid bS \mid b$

Se procede del siguiente modo:
$S \rightarrow CA \mid DB$
$A \rightarrow CAA \mid DS \mid a$
$B \rightarrow DBB \mid CS \mid b$
$C \rightarrow b$
$D \rightarrow a$

$S \rightarrow CA \mid DB$
$A \rightarrow CE \mid DS \mid a$
$B \rightarrow DF \mid CS \mid b$
$C \rightarrow b$
$D \rightarrow a$
$E \rightarrow AA$
$F \rightarrow BB$

Al final todas las producciones tienen que ser del tipo
$A \rightarrow BC$ ó
$A \rightarrow a$

$B, C \in V, a \in \sum$

Lema del Bombeo para CFL.

Al igual que para los lenguajes regulares, existe el lema del bombeo para los lenguajes independientes del contexto. Análogamente al caso anterior, en este un lenguaje ha de cumplir el lema del bombeo si es CFL, si no lo cumple, no lo es. Y también es posible que si lo cumple sea CFL o no lo sea.

Sea L CFL | ∅ ∉ L
$\exists k \in \mathbb{N}$ | si $z \in L$, $|z| > k$:

1) $z = uvwxy$
2) $|vwx| \leq k$
3) $uv^i wx^i y \in L, \ \forall i \geq 0$
4) $|vx| \geq 1$ (es decir, v y x no pueden ser ε a la vez)

Dem:

$L=\{a^i b^j \mid j = i^2\}$
Sea k la constante del lema del bombeo.

Sea $z = a^k b^{k^2} \in L$
Si $|z| > k \Rightarrow$

1) $z = uvwxy$
2) $|vwx| \leq k$
3) $|vx| \geq 1$
4) $uv^i wx^i y \in L, \ \forall i \geq 0$

Si $z = a^k b^{k^2} = uvwxy$
Supongamos $v = a^r b^s \Rightarrow v^i = (a^r b^s)^i \Rightarrow uv^i wx^i y \notin L$, porque habría símbolos b antes que a, por ejemplo: $v^2 = a^r b^s a^r b^s$.

Lo mismo ocurre con la "x"

Casos que quedan:
a) $v = a^r \quad x = a^s$
$|vx| \geq 1 \Rightarrow r + s \geq 1$ (3)

$z^2 = uv^2wx^2y = a^{k^2+r+s} b^k \notin L$, ya que $k^2 \neq (k + r + s)^2$

b) $v = b^r \quad x = b^s$
$|vx| \geq 1 \Rightarrow r + s \geq 1$ (3)

$z^i = uv^i wx^i y = a^k \ b^{k^2+r+s} \notin L$, ya que $k^2 + r + s \neq k^2$

c) $v = a^r \quad x = b^s$
$|vx| \geq 1 \Rightarrow r + s \geq 1$ (3)

$z^i = uv^i wx^i y = a^{k-r+ri} \ b^{k^2-s+si} = a^{k+(i-1)r} \ b^{k^2+(i-1)s} \notin L$, ya que $(k+(i-1)r)^2 \neq k^2+(i-1)s$

Luego, vemos que el lenguaje no cumple el lema del bombeo, con lo cual no es CFL.

Otro ejemplo

$L=\{a^i b^i c^i \mid i \geq 1\} = \{abc, aabbcc, aaabbbccc, aaaabbbbcccc...\}$

Sea n la constante del lema del bombeo
Sea $z=a^n b^n c^n \in L$

$|z| > n \Rightarrow$ 1) $z = xyrst$
2) $|yrs| \leq n$
3) $|ys| \geq 1$
4) $xy^i rs^i t \in L, \ \forall i \geq 0$

si $z = a^n b^n c^n = xyrst$

Como ocurre en 2), $|yrs| \leq n$, las cadenas $|ys|$ no pueden contener simultáneamente símbolos a y c, porque entre la última a y la primera c tiene que haber al menos una b.

Casos que quedan:

a) $y = a^p \quad s = a^q$
(3) $|ys| \geq 1 \Rightarrow p + q \geq 1$
$z^2 = xy^2rs^2t = a^{n+2p+2q} b^n c^n \notin L$, ya que el número de "$a$" es mayor que el número de "b" o "c"

b) $y = b^p \quad s = b^q$
Lo mismo

c) $y = c^p \quad s = c^q$
Lo mismo

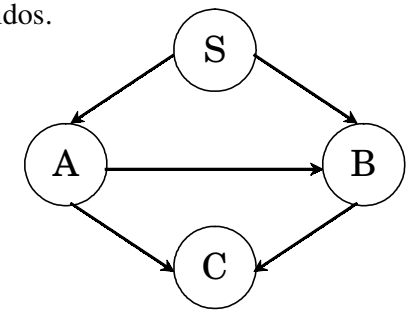
d) $y = a^p \quad s = b^q$
$z^2 = xy^2rs^2t = a^{n+p} b^{n+q} c^n \notin L$, salvo que $p+q = 0$, pero $p+q \geq 1$

e) $y = b^p \quad s = c^q$
sea $z^0 = xy^0rs^0t = a^n b^{n-p} c^{n-q} \notin L$

Luego, no es CFL al no cumplir el lema del bombeo.

Teor: Existen algoritmos para determinar si una CFG es finita, infinita o vacía.
Teor: Una CFG es vacía si en la eliminación de símbolos inútiles resulta eliminada S
Teor: Una CFG es finita si no tiene ciclos dirigidos
Teor: Una CFG es infinita si tiene ciclos dirigidos.

Ejemplos
$S \rightarrow AB$
$A \rightarrow BC \mid a$
$B \rightarrow CC \mid b$
$C \rightarrow a$



Vemos que no tiene ciclos dirigidos, luego es finito

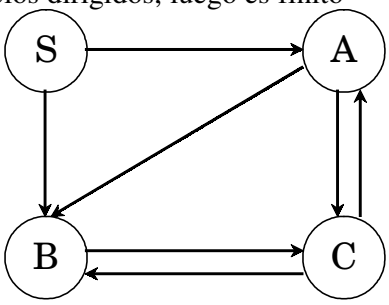
Sea ahora:

$S \rightarrow AB$

$A \rightarrow BC \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow a \mid AB$



Esta gramática sin embargo tiene varios ciclos dirigidos, por ejemplo A-B-C-A, lo que hará que sea infinita.

Análisis sintáctico: Algoritmo CYK (Cocke-Younger-Kasami).

Dada $G \equiv (V, \sum, S, P)$, y dada $w \in \sum^*$, ¿$w \in L(G)$? Dicho problema se conoce como problema de decisión.

Dada una CFG $G \equiv (V, \sum, S, P)$ escrita en Forma Normal de Chomsky y sin producciones vacías. Sea $x \in \sum^*$, $\forall A \in V$ y $\forall w$ subcadena de x, es posible determinar si A puede derivar en un nº indeterminado de pasos en w.

Ejemplo del algoritmo CYK

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

Sea por ej, *w = baaba*

	b	***a***	***a***	***b***	***a***
1	B	A, C	A, C	B	A, C
2	A, S	B	S, C	A, S	
3	∅	B	B		
4	∅	S, C, A			
5	S...				

Para sacar la anterior tabla se siguen los siguientes pasos:

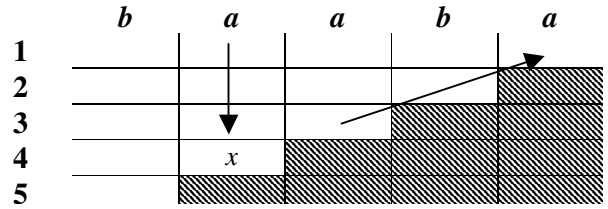
Se construye dicha tabla poniendo en la primera fila un símbolo de la cadena por cada celda (en este caso *baaba*).

En la fila número 1, tenemos que mirar el símbolo que tenemos encima. En la casilla (1, 1) es la "*b*". Entonces miramos la gramática, y buscamos todas aquellas variables que produzcan directamente la "*b*". Vemos que en este caso, únicamente es $B \rightarrow CC \mid b$, luego en la casilla (1, 1) va una B. Lo mismo con los elementos de la fila 1, en caso de la "*a*", las gramáticas que generan directamente este símbolo terminal son A y C.

Para formar la fila 2, debemos mirar los resultados obtenidos en el anterior paso, es decir, la fila que tenemos encima. Si queremos formar por ejemplo la casilla (2, 1), hemos de mirar la casilla justo encima de ella, es decir, la (1, 1), y la que está inmediatamente a su derecha, es decir, la (1, 2). Vemos que tenemos en una casilla B, y en otra A, C. Entonces entre las dos casillas formamos las combinaciones posibles, es decir, en este caso tenemos BA y BC. Una vez tenidas las combinaciones, miramos la gramática y vemos si existe alguna producción que las genere. Vemos que: $S \rightarrow AB|BC$, y $A \rightarrow BA \mid a$, con lo cual vemos que S, y A generan esas dos combinaciones. Luego en la casilla (2, 1), irá una S y una A. Las demás casillas de esa fila se forman exactamente

igual, mirando primero la casilla de arriba, la inmediatamente a su derecha, formando combinaciones y mirando la gramática.
A partir de ahí, la cosa cambia. Las demás filas se deben hallar del siguiente modo: Supondremos que tenemos dos punteros. Uno se coloca en la primera casilla de la columna cuyo hueco queremos hallar. Por ejemplo, si fuera la casilla (3, 1), colocaríamos el puntero en la posición (1, 1). El segundo puntero lo colocamos exactamente en la casilla de encima de la celda que queremos hallar, pero una columna desplazada a la derecha. Tenemos ya las dos primeras celdas de las que tenemos que hacer las combinaciones mencionadas antes, y evidentemente miramos la gramática y apuntamos en la celda aquellas producciones que las generen. Ahora movemos el primer puntero en dirección vertical descendente una posición, mientras que el otro se moverá en sentido diagonal hacia arriba-derecha una casilla. Ya tenemos otras dos casillas de las que hacer combinaciones. Seguimos moviendo del mismo modo hasta que el segundo puntero llegue hasta una casilla de la primera fila y el primero llegue a una posición por encima de la casilla que estamos hallando.

Supongamos que queremos hallar la casilla marcada con "*x*". El sentido de los punteros entonces es el siguiente:



$w \in L(G) \Leftrightarrow$ En la última casilla de la primera fila aparece el símbolo inicial, S.
En este momento podemos parar de calcular más combinaciones, ya que la cadena será aceptada. Si S no aparece en dicha casilla, la cadena no es generada por la gramática. En este caso, $w \in L(G)$.

Supongamos que tenemos la misma gramática que antes, pero la cadena es ahora
w = bbab

$S \rightarrow AB \mid BC$
$A \rightarrow BA \mid a$
$B \rightarrow CC \mid b$
$C \rightarrow AB \mid a$

w = bbab

	b	*b*	*a*	*b*
1	B	B	A, C	B
2	∅	A, S	S, C	
3	A	S, C		
4	S, C			

Vemos que S está en la última casilla de la primera columna, luego $w \in L(G)$ también.

Teor: Si L_1 y L_2 son CFL, $L_1 \cup L_2$ es CFL.

Teor: Si L_1 y L_2 son CFL, $L_1 \cdot L_2$ es CFL.

Teor: Si L es CFL, L* es CFL.

Teor: Los CFL no son cerrados respecto a la intersección, ni a la complementación.

Autómatas a Pila No Deterministas (APND).

Son un nuevo tipo de autómata que evidentemente funciona una pila. Un APND se define del siguiente modo:

$M \equiv (\Sigma, Q, \delta, q_0, F, \Gamma, Z)$

Γ es el denominado Alfabeto de Pila (es decir, los símbolos que se pueden usar en la pila).

Z es el símbolo inicial de la pila. Cuando empecemos a hacer operaciones con la pila será el único símbolo que se encontrara en ella. $Z \in \Gamma$.
Los demás elementos los conocemos (Σ, Q, F, q_0, δ).

Esta vez la función δ viene definida del siguiente modo:

$$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \longrightarrow \wp(Q \times \Gamma^*)$$
$$(p, a, b) \alpha \quad \{(q, w)\}$$

$p, q \in Q, \quad a \in (\Sigma \cup \{\varepsilon\}), \quad b \in \Gamma, w \in \Gamma^*$

Puede ocurrir:
- $\delta(p, a, b) = \{(q, \varepsilon)\}$ (el autómata quita el símbolo b y no inserta nada También se llama desapilar)
- $\delta(q, \varepsilon, a) = \{((p|q), aa)\}$ (el autómata apila el símbolo a)
- $\delta(q, \varepsilon, a) = \{(q, a)\}$ (el autómata no modifica la pila, se queda igual)
- $\delta(q, a, b) = \{\varnothing\}$ (el autómata no puede transitar, se detiene la ejecución)

Ejemplo
Sea M un autómata definido así:
$Q = \{q_0, q_1, q_2, q_3\}$ $F = \{q_3\}$ $\Sigma = \{a, b\}$ $\Gamma = \{Z, A\}$ $q_0 = q_0$

Q / δ	(a, Z)	(b, Z)	(ε, Z)	(a, A)	(b, A)	(ε, A)
q_0	$\{q_1, AZ)(q_3, \varepsilon)\}$		$\{(q_3, \varepsilon)\}$			
q_1				$\{(q_1, AA)\}$	$\{(q_2, \varepsilon)\}$	
q_2			$\{(q_3, \varepsilon)\}$		$\{(q_2, \varepsilon)\}$	
q_3						

$L = \{a^i b^i \mid i \geq 0\} \cup \{a\}$

Descripción instantánea de un autómata a pila.

Se define una descripción instantánea del siguiente modo:

D.I.= $(p,\ u,\ w) \in Q \times \Sigma^* \times \Gamma^*$

p es el estado actual
u es la cadena que queda por leer
w es el contenido de la pila (se lee en este sentido: ←)

$(p,\ ax,\ b\alpha) \vdash (q,\ x,\ \beta\alpha) \Leftrightarrow (q,\ \beta) \in \delta\ (p,\ a,\ b)$

Símbolo para representar una transición

$p,\ q \in Q$
$a \in \Sigma$
$x \in \Sigma^*$
$b \in \Gamma$
$\alpha, \beta \in \Gamma^*$

Nota: $\vdash^*$ significa que en cierto número de pasos llega a otra D.I.

Sea $M \equiv (\Sigma, Q, \delta, q_0, F, \Gamma, Z)$

· $L(M)=\{w \in \Sigma^* \mid (q_0,\ w,\ z) \vdash^* (p,\ \varepsilon, \alpha)$ para algún $p \in F,\ \alpha \in \Gamma^*\}$

Se define otro lenguaje,

· $N(M)=\{w \in \Sigma^* \mid (q_0,\ w,\ z) \vdash^* (r,\ \varepsilon, \varepsilon)$ para algún $r \in Q\}$

Éste último se denomina Lenguaje reconocido por pila vacía.

Teor: $L(M) \neq N(M)$ en general, aunque ambos criterios de aceptación son equivalentes.

Teor: Un autómata a pila es determinista si en una D. I. El autómata sólo tiene una forma de evolucionar. Si no cumple esta regla, es no determinista.
P.ej:

$$(q,\ au,\ x\alpha) \vdash^* \begin{cases} (p, u, w\alpha) \\ (r, au, \gamma\alpha) \end{cases}$$

Notación: Los APND también se denotan por PDA

Teor: La condición para ser determinista es:
a) $\text{card}(\delta\ (q,\ a,\ x)) \leq 1\ \wedge\ \text{card}(\delta\ (q,\ \varepsilon,\ x))=0$
ó b) $\text{card}(\delta\ (q,\ a,\ x))=0,\ \forall a \in A\ \wedge\ \text{card}(\delta\ (q,\ \varepsilon,\ x)) \leq 1$

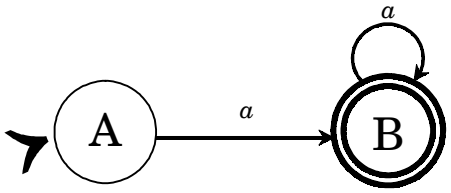
Teor: Si un DFA/NFA reconoce un lenguaje, un PDA también puede.

P.ej.:

$NFA \equiv (\sum, Q, F, q_0, \delta)$

$PDA \equiv (\sum, Q, \delta, q_0, F, \Gamma, Z)$.

En este caso no se usarán Γ ni Z, para que el comportamiento del PDA sea igual al del NFA



Q={A, B}
$\sum$={*a*}
q_0=A
F={B}

Q/δ	**(*a*, Z)**
A	{(B, Z)}
B	{(B, Z)}

Teor: El conjunto de los lenguajes que reconocen los autómatas a pila es al menos el de los lenguajes regulares

Sea

$G \equiv (V, \sum, S, P)$, y $M \equiv (\sum', Q, \delta, q_0, F, \Gamma, Z)$ tal que L(M) = L(G)

Q={A, B, C}
F={C}
q_0=A
$\sum = \sum'$
$\Gamma = \{Z\} \cup V \cup \sum$
Z=Z

δ (A, ε, Z) = {(B, SZ)}
δ (A, ε, R) = {(B, *w*) | (R → *w*) ∈ P}, ∀ R∈ V
δ (A, *a*, *a*) = {(B, ε, ε}}, ∀ *a*∈ $\sum$
δ (B, ε, Z) = {(C, ε)}

L(M) = L(G)

Ejemplo:
Buscar un autómata tal que:
L={w∈ {*a*, *b*}* | $n_a(w)$=$n_b(w)$} (el número de "*a*" sea igual al de "*b*")

Q={q_0, q_1}, F={q_1}
Γ={Z, A, B}

Q/δ	(ε,Z)	(*a*,Z)	(*b*,Z)	(*a*,A)	(*b*,A)	(ε,A)	(*a*,B)	(*b*,B)	(ε,B)
q_0	{(q_1, ε)}	{(q_0,AZ)}	{(q_0,BZ)}	{(q_0,AA)}	{(q_0, ε)}	{∅}	{(q_0, ε)}	{(q_0,BB)}	{∅}
q_1									

Ejemplo:

$S \rightarrow aSa \mid bSb \mid \varepsilon$

$L(G)=\{w \cdot w^i \mid w \in \{a, b\}^*\}$

Si quisiéramos hacer el PDA, la función δ se define de este modo:

$\delta(q_0, \varepsilon)=\{(q_1, SZ)\}$
$\delta(q_1, a, a)=\delta(q_1, b, b)=\{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, S)=\{(q_1, aSa), (q_1, bSb), (q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, Z)=\{(q_2, \varepsilon)\}$

Sea por ejemplo $w = abba$
$(q_0, abba, z) \vdash (q_1, abba, SZ) \vdash (q_1, abba, aSaZ) \vdash (q_1, bba, SaZ) \vdash$
$(q_1, bba, bSbaZ) \vdash (q_1, ba, SbaZ) \vdash (q_1, ba, baZ) \vdash^* (q_1, \varepsilon, Z) \vdash (q_2, \varepsilon, \varepsilon)$

Teor: Dado un PDA $M \equiv (\sum, Q, \delta, q_0, F, \Gamma, Z)$, $\exists M' \equiv (\sum', Q', \delta' q_0', F', \Gamma', Z')$
tal que N(M')=L(M)
$Q' = Q \cup \{q_0', q_5\}$
$q_0' = q_0$
$\Gamma' = \Gamma \cup \{Z'\}$
$\delta' = \delta$
$Z' = Z$
$F' = \{\varnothing\}$

a) $\delta'(q_0', \varepsilon, Z') = \{(q_0, ZZ')\}$
b) $\delta'(q, \varepsilon, A) = \{(q_5, A)\}, \forall q \in F, \forall A \in \Gamma \cup \{Z'\}$
c) $\delta'(q_5, \varepsilon, A) = \{(q_5, \varepsilon)\}, \forall A \in \Gamma \cup \{Z'\}$

Nota: M nunca quitaría Z' de la pila y por lo tanto no podría aceptar por pila vacía.

Teor: La pila de un PDA no es finita.

Teor: Dado un PDA $M \equiv (\sum, Q, \delta, q_0, F, \Gamma, Z)$, $\exists M' \equiv (\sum', Q', \delta' q_0', F', \Gamma', Z')$ |
L(M') = N(M)

$Q' = Q \cup \{q_0', q_f\}$ $\Gamma' = \Gamma \cup \{Z'\}$
$q_0' = q_0$ $Z' = Z$
$\delta' = \delta$ $F' = \{q_f\}$

a) $\delta'(q_0', \varepsilon, Z') = \{(q_0, ZZ')\}$

b) $\delta'(q_i', \varepsilon, Z') = \{(q_f, \varepsilon)\}, \forall q_i \in Q$

Teor: Dado un PDA $M \equiv (\sum, Q, \delta, q_0, F, \Gamma, Z)$, $\exists G \equiv (V, \sum, S, P) \mid L(G)=N(M)$

V: $<p\ A\ q>$ donde p, q$\in$ Q y A$\in \Gamma$.

El objetivo del autómata es ir de p a q eliminando A de la pila.

Tipos de producciones:

a) $(S \rightarrow <q_0\ Z\ p>) \in P,\ \forall p \in Q$

b) Si $(p, \varepsilon) \in \delta(q, a, A) \Rightarrow (<q\ A\ p> \rightarrow a) \in P$

c) Si $(p_0, \alpha) \in \delta(q, a, A),\ \alpha = B_1 B_2 ... B_m$,

$<p_0\ \alpha\ p_m> = <p_0\ B_1\ p_1><p_1\ B_2\ p_2> = <p_{m-1}\ B_m\ p_m>,\ \forall p_i \in Q$

Las producciones son:

$(<q\ A\ p_m> \rightarrow a<p_0\ B_1\ p_1><p_1\ B_2\ p_2> ... <p_{m-1}\ B_m\ p_m>) \in P,\ \forall p_i \in Q$

Ejemplo:

Sea el PDA, $Q=\{q_1, q_2, q_3\}$, $F=\{q_3\}$, $q_0=q_1$, $\Gamma=\{A, Z\}$, $\sum=\{a, b\}$

$\delta(q_1, a, Z)=\{(q_1, AZ)\}$

$\delta(q_1, a, A)=\{(q_1, AA)\}$

$\delta(q_1, b, A)=\{(q_2, \varepsilon)\}$

$\delta(q_2, b, A)=\{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, A)=\{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, Z)=\{(q_3, \varepsilon)\}$

Aplicando a) $\longrightarrow$ $S \rightarrow <q_1\ Z\ q_1> \mid <q_1\ Z\ q_2> \mid <q_1\ Z\ q_3>$

Aplicando b) $\longrightarrow$ 3) $<q_1\ A\ q_2> \rightarrow b$

4) $<q_2\ A\ q_2> \rightarrow b$

5) $<q_2\ A\ q_2> \rightarrow \varepsilon$

6) $<q_2\ Z\ q_3> \rightarrow \varepsilon$

Aplicando c) $\longrightarrow$ 1) $<q_1\ Z\ q_1> \rightarrow a<q_1\ A\ q_1><q_1\ Z\ q_1> \mid a<q_1\ A\ q_2><q_2\ Z\ q_1>$
$\mid a<q_1\ A\ q_3><q_3\ Z\ q_1>$

$<q_1\ Z\ q_2> \rightarrow a<q_1\ A\ q_1><q_1\ Z\ q_2> \mid a<q_1\ A\ q_2><q_2\ Z\ q_2>$
$\mid a<q_1\ A\ q_3><q_3\ Z\ q_2>$

$<q_1\ Z\ q_3> \rightarrow a<q_1\ A\ q_1><q_1\ Z\ q_3> \mid a<q_1\ A\ q_2><q_2\ Z\ q_3>$
$\mid a<q_1\ A\ q_3><q_3\ Z\ q_3>$

$$2)\ <q_1\ A\ q_1> \rightarrow a<q_1\ A\ q_1><q_1\ A\ q_1> \mid a<q_1\ A\ q_2><q_2\ A\ q_1> \mid a<q_1\ A\ q_3><q_3\ A\ q_1>$$

$$<q_1\ A\ q_2> \rightarrow a<q_1\ A\ q_1><q_1\ A\ q_2> \mid a<q_1\ A\ q_2><q_2 A\ q_2> \mid <q_1\ A\ q_3><q_3\ A\ q_2>$$

$$<q_1\ A\ q_3> \rightarrow a<q_1\ A\ q_1><q_1\ A\ q_3> \mid a<q_1\ A\ q_2><q_2\ A\ q_3> \mid a<q_1\ A\ q_3><q_3\ A\ q_3>$$

Otro ejercicio

Sea el PDA: $Q=\{q_1, q_2\}$, $\Gamma=\{A, Z\}$, $\sum=\{a, b\}$, $F=\{q_2\}$

Hallar la gramática.

1) $\delta(q_1, a, Z) = \{(q_1, AZ)\}$

2) $\delta(q_1, b, A) = \{(q_1, AA)\}$

3) $\delta(q_1, a, A) = \{(q_2, \varepsilon)\}$

0) $S \rightarrow <q_1\ Z\ q_1> \mid <q_1\ Z\ q_2>$

3) $<q_1\ A\ q_2> \rightarrow a$

1) $<q_1\ Z\ q_1> \rightarrow a<q_1\ A\ q_1><q_1\ Z\ q_1> \mid a<q_1\ A\ q_2><q_2 Z\ q_1>$

$<q_1\ Z\ q_2> \rightarrow a<q_1\ A\ q_1><q_1\ Z\ q_2> \mid a<q_1\ A\ q_2><q_2 Z\ q_2>$

2) $<q_1\ A\ q_1> \rightarrow b<q_1\ A\ q_1><q_1\ A\ q_1> \mid b<q_1\ A\ q_2><q_2 A\ q_1>$

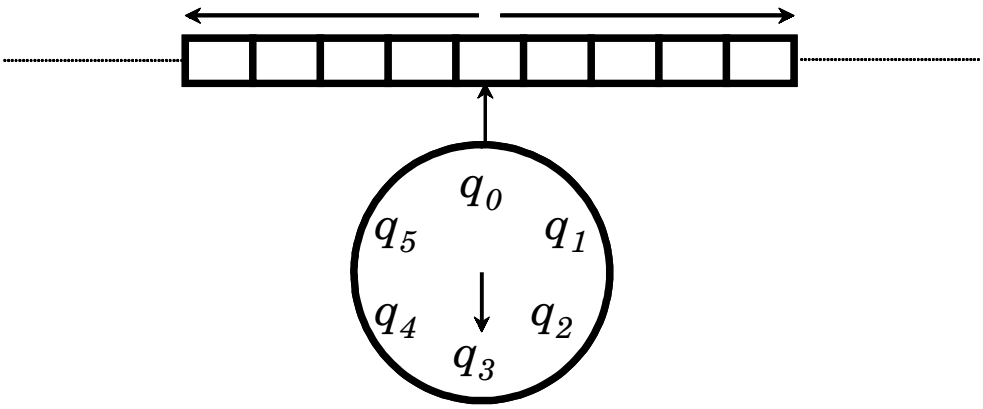
$<q_1\ A\ q_2> \rightarrow b<q_1\ A\ q_1><q_1\ A\ q_2> \mid b<q_1\ A\ q_2><q_2 A\ q_2>$

Teor: $L \subseteq \sum^* \Leftrightarrow \exists \text{PDA } M \equiv (\sum, Q, \delta, q_0, F, \Gamma, Z) \mid L=L(M)$

Tema 5
Máquinas de Turing.

Una máquina de Turing se caracteriza porque tiene memoria infinita. La máquina puede escribir en la cinta, y luego moverse.

Dibujo ilustrativo



Una Máquina de Turing se define del siguiente modo:

$TM \equiv (Q, \Sigma, \delta, q_0, F, \Gamma, \not\emptyset)$

Γ será en este caso el alfabeto de cinta, y $\not\emptyset$ es un símbolo nuevo llamado "blanco".

$\not\emptyset \in \Gamma, \not\emptyset \notin \Sigma$.

Σ es la entrada, y como ya dijimos, Γ será la cinta.

La función δ viene definida así:

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

$(q, a) \quad \alpha \quad (p, b, x)$

$p, q \in Q$

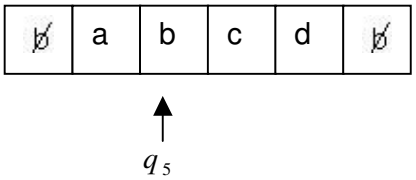
$a, b \in \Gamma$

$x \in \{L, R\}$

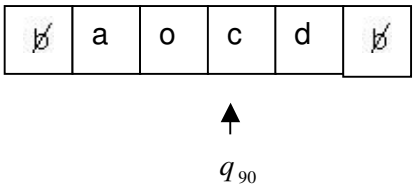
{L, R} es el conjunto que define el movimiento del autómata por la cinta. L indica que después de realizar la operación, dicha máquina se mueve hacia la izquierda (Left), y R, que se mueve a la derecha (Right). Hay también otra opción que es S, y significa quedarse parado (Stop), que equivale a moverse primero a la izquierda y luego a la derecha, o primero a la derecha y luego a la izquierda.

$L = \{a^n b^n c^n \mid n \geq 0\}$ es Turing-Computable.

Ejemplo



$\delta(q_5, b) = (q_{90}, o, R)$



Ejemplo

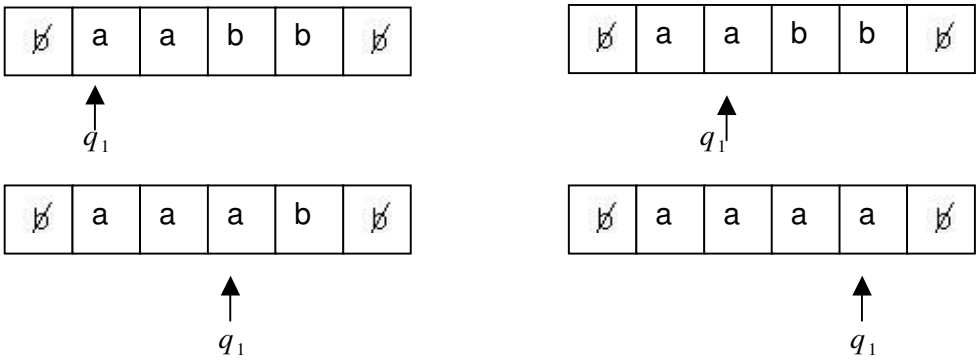
$Q=\{q_1, q_2\}$, $F=\{q_2\}$, $\Sigma=\{a, b\}$, $\Gamma=\{a, b, ø\}$, $q_0=q_1$.

$\delta(q_1, a)=(q_1, a, R)$

$\delta(q_1, b)=(q_1, a, R)$

$\delta(q_1, ø)=(q_2, ø, L)$

Sea por ejemplo la cadena $w = aabb$



Teor: Si $\delta(q, a)$ no está definida, la máquina se dice que está parada.

Descripción instantánea de una Máquina de Turing.

Para hacer las descripciones instantáneas en una TM, hay dos formas:

a) $(q_i, w_1 a w_2)$, $q_i \in Q$, $w_1, w_2 \in \Gamma^*$, $a \in \Gamma$

b) $a_1 a_2 ... a_k q_i a_{k+1} a_{k+2} a_n \equiv (q_i, a_1 a_2 ... a_k a_{k+1} a_{k+2} a_n)$

En la TM de antes hagamos la descripción instantánea:

$(q_1, \underline{a}abb) \vdash (q_1, a\underline{a}bb) \vdash (q_1, aa\underline{b}b) \vdash (q_1, aaa\underline{b}) \vdash (q_1, aaaa\underline{\not\emptyset}) \vdash (q_2, aaa\underline{a})$

Usando la forma b)

$q_1\ aabb \vdash aq_1abb \vdash aaq_1bb \vdash aaaq_1b \vdash aaaaq_1\not\emptyset \vdash aaaq_2a$

Ejemplo

$Q=\{q_1,q_2,q_3\}$, $F=\{q_3\}$, $\Sigma=\{a, b\}$, $\Gamma=\{a, b, \not\emptyset\}$, $q_0 = q_1$

$\delta(q_1, a) = (q_1, a, L)$

$\delta(q_1, b) = (q_1, b, L)$

$\delta(q_1, \not\emptyset) = (q_2, \not\emptyset, R)$

$\delta(q_2, a) = (q_3, a, L)$

$\delta(q_2, b) = (q_3, b, L)$

$\delta(q_2, \not\emptyset) = (q_3, \not\emptyset, L)$

$(q_1, aab\underline{a}bb) \vdash (q_1, aa\underline{b}abb) \vdash (q_1, a\underline{a}babb) \vdash (q_1, \underline{a}ababb) \vdash (q_1, \underline{\not\emptyset}aababb)$
$\vdash (q_2, \underline{a}ababb) \vdash (q_3, \underline{\not\emptyset}aababb)$

A partir de ahora, los $q \in F$ no tendrán transiciones (por convenio).

Def: Se denomina computación de una TM a la secuencia de movimientos que conduce a la máquina a una parada.

Def: Sea $TM \equiv (Q, \Sigma, \delta, q_0, F, \Gamma, \not\emptyset)$, se define el lenguaje aceptado por una TM:

$L(M)=\{w \in \Sigma^* \mid (q_0, w) \vdash^* (p, w_1aw_2)$, para algún $p \in F\}$

La máquina tiene que estar parada, pero se asume que no hay transiciones para p.

Ejemplo

$L(M)=L(a^*)$

$Q = \{q_0, q_1\}$, $\Gamma=\{a, b, \not\emptyset\}$, $q_0=q_0$, $F = \{q_1\}$, $\Sigma=\{a\}$

$\delta(q_0, a)=(q_0, a, R)$

$\delta(q_0, \not\emptyset)=(q_1, \not\emptyset, L)$

Veamos ahora otro ejemplo para el mismo lenguaje pero con una TM distinta.

$Q = \{q_1, q_2, q_3\}$, $\Gamma = \{a, b, \not\emptyset\}$, $q_0=q_1$, $F = \{q_3\}$, $\Sigma=\{a, b\}$

$\delta(q_1, a)=(q_1, a, R)$
$\delta(q_1, b)=(q_2, b, R)$
$\delta(q_1, \not\emptyset)=(q_3, \not\emptyset, S)$

$\delta(q_2, a)=(q_2, a, R)$
$\delta(q_2, b)=(q_2, b, R)$
$\delta(q_2, \not\emptyset)=(q_2, \not\emptyset, R)$

Este es un bloque de transiciones definido de tal manera que si entra en el estado q_2, la máquina nunca se parará. Es otro modo de decir que la máquina rechaza la cadena. Como vemos esto ocurre por ejemplo cuando llega una "*b*". Entonces hemos visto dos formas de que la máquina rechace las $w \notin$ L(M), en estos dos últimos ejemplos.

<u>Ejercicio</u>
Hacer una TM que reconozca el siguiente lenguaje.

$L=\{a^n b^n \mid n \geq 1\}$

$Q=\{q_0, q_1, q_2, q_3, q_4\}$, $F=\{q_4\}$, $\Gamma=\{a, b, \not\emptyset, c, d\}$, $q_0=q_0$

$\delta(q_0, a)=(q_1, c, R)$
$\delta(q_1, a)=(q_1, a, R)$
$\delta(q_1, b)=(q_1, d, L)$
$\delta(q_1, d)=(q_2, d, L)$

Cambia "*a*" por "*c*" y busca a la derecha la primera "*d*"

$\delta(q_2, d)=(q_2, d, L)$
$\delta(q_2, a)=(q_2, a, L)$
$\delta(q_2, c)=(q_0, c, R)$

Se mueve a la izquierda buscando la primera "*c*"

$\delta(q_0, d)=(q_3, d, R)$ Si estamos aquí, es porque se marcó la última "*a*"

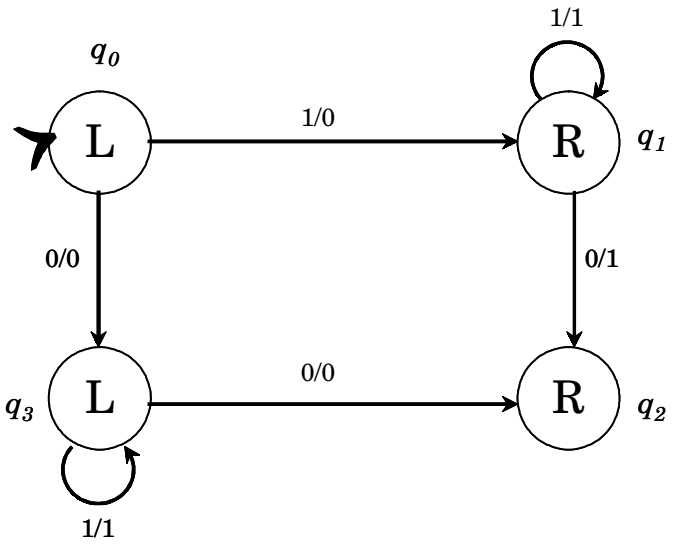
$\delta(q_3, d)=(q_3, d, R)$
$\delta(q_3, \not\emptyset)=(q_4, \not\emptyset, S\}$ Aceptará si estando en q_3 lee un blanco en la cinta, porque habrá igual número de "*c*" que de "*d*"

Diagramas de transición para una Máquina de Turing.

También existen los diagramas de transición para TM. Supongamos que tenemos como ejemplo la siguiente función δ :

δ	**0**	**1**
q_0	----------	$(q_1, 0, R)$
q_1	$(q_2, 1, R)$	$(q_1, 1, R)$
q_2	$(q_3, 0, L)$	----------
q_3	$(q_0, 0, R)$	$(q_3, 1, L)$

Si quisiéramos dibujar el diagrama de transiciones, sería del siguiente modo:

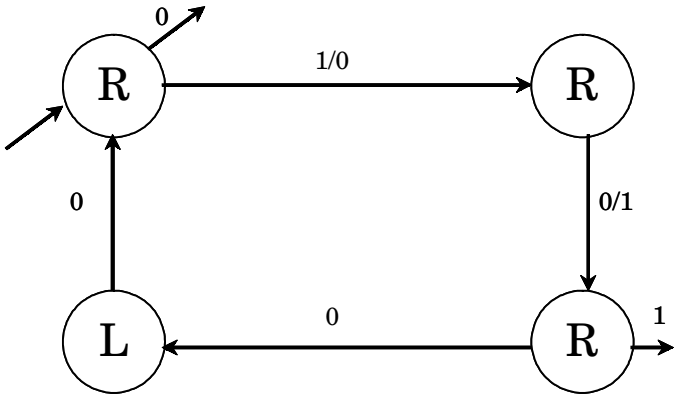


Como vemos, cada transición tiene asociada un i/j. La i corresponde al símbolo que lee en la cinta actualmente, y la j al símbolo con el que reemplazará antes de moverse dicho símbolo i. Por ejemplo, 0/1 implica que leyendo un 0 en la cinta, lo sustituirá por un 1.

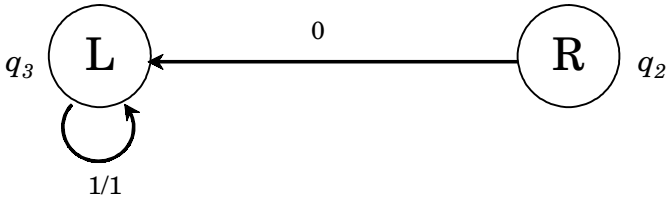
Simplificaremos aún más el diagrama anterior, cuando tengamos transiciones del tipo 0/0, escribiremos simplemente un 0, e igualmente con 1/1, escribiremos un 1. Es decir: $a/a \Rightarrow a, \forall a \in \Gamma$

Por otro lado, si tenemos una transición *i/j*, donde $i = j$, hacia un mismo estado, esa transición se puede obviar y no dibujarla, ya que en realidad nos estamos quedando en el mismo estado.

Por último resaltar que aquellas transiciones que no estén definidas, como por ejemplo en este caso, $\delta(q_0, 0)$, se debe dibujar una transición saliente del estado, pero que no incide en ningún otro estado, sino se queda "en el aire". Entonces, el anterior diagrama nos quedaría del siguiente modo

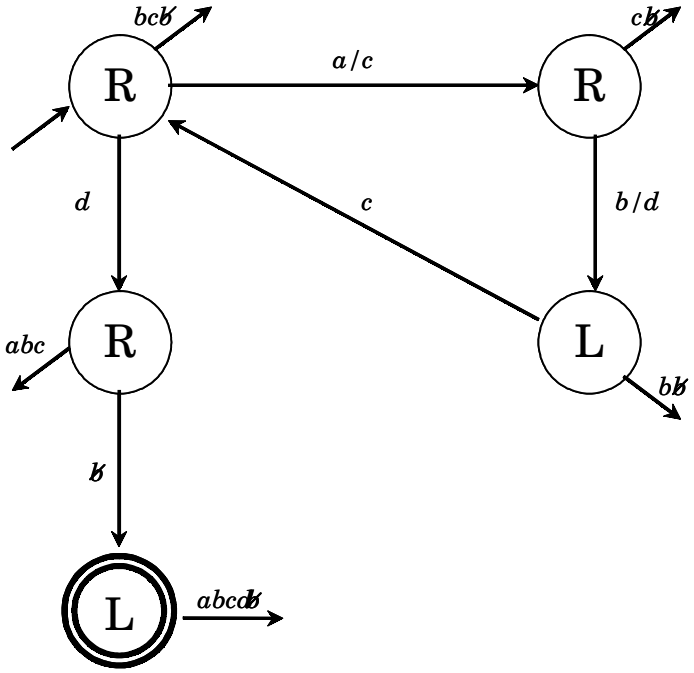


Nota: Si hubiera algún estado de aceptación se representaría igual que en los demás autómatas.



Ejercicio

Hacer el diagrama de transiciones para el autómata L=$\{a^n b^n \mid n \geq 1\}$



Teor: Los lenguajes reconocidos por una TM son los **lenguajes recursivamente enumerables.** La TM procesa, para y acepta. La máquina también puede no parar. Evidentemente, si no para, la cadena ante la que no paró no pertenece al lenguaje.

Def: Se llaman **lenguajes recursivos** a aquellos donde existe al menos una TM que se pare ante cualquier cadena, es decir, no puede quedarse "colgada" (no parar).

Teor: Lenguajes Recursivos $\subset$ Lenguajes Recursivamente Enumerables.

Def: Definición de Función Turing-Computable

f: $\Sigma^* \longrightarrow \Sigma^*$ es Turing-Computable sii $\exists$ TM

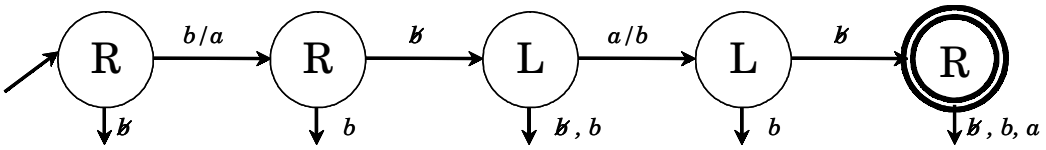
$M \equiv (Q, \Sigma, \delta, q_0, F, \Gamma, \not\emptyset) \mid q_0 w \longrightarrow pu$, para algún $p \in F$, cuando $f(w)=u$

Teor: Las TM pueden hacer cualquier operación (sumar, restar...)

Ejemplo de suma

$F(m, n)=n + m$

q / a	a	$\not\emptyset$
q_1	(q_1, a, R)	----------
q_2	(q_2, a, R)	$(q_3, \not\emptyset, L)$
q_3	(q_4, b, L)	----------
q_4	(q_4, a, L)	$(q_5, \not\emptyset, R)$



Combinación de Máquinas de Turing.

Es posible combinar varias máquinas de Turing y así su funcionamiento.

$M_1 \equiv (\Sigma, Q_1, q_{01}, \delta_1, \Gamma, F_1, \not\emptyset)$

$M_2 \equiv (\Sigma, Q_2, q_{02}, \delta_2, \Gamma, F_2, \not\emptyset)$

Se supone que $Q_1 \cap Q_2 = \varnothing$.

La máquina M_1 combinada con M_2 sería:

$M_1 \cdot M_2 \equiv (Q, \Sigma, \delta, q_0, F, \Gamma, \emptyset)$

$Q = Q_1 \cup Q_2$

$q_0 = q_{01}$

$F = F_2$

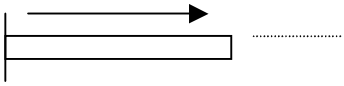
$\underline{\delta(q,a)}$

- $\delta_1(q, a)$, si $q \in Q_1$ y $\delta_1(q, a) \neq (p, b, x)$, $\forall p \in F_1$
- (q_{02}, b, x) si $q \in Q_1$ y $\delta_1(q, a) = (p, b, x)$, $\forall p \in F$
- $\delta_2(q, a)$ si $q \in Q_2$

Otros tipos de Máquinas de Turing.

M. T. con múltiples pistas $\rightarrow \delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}$

M. T. con cinta infinita en un único sentido $\rightarrow$



M. T. Multicinta $\rightarrow \delta : Q \times \Gamma^n \longrightarrow Q \times \Gamma^n \times \{L, R, S\}^n$

Tiene n cintas con las que se puede operar simultáneamente.

M. T. con cinta multidimensional $\rightarrow$ Permiten movimientos en diagonal.

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, U, D, S\}$$

M. T. no determinista $\rightarrow \delta : Q \times \Gamma \longrightarrow \wp(Q \times \Gamma \times \{L, R\})$

M. T. universal (Mu)

Toma como entrada una cadena y una codificación de una T. M. y simula el comportamiento de la T. M. ante la entrada. Mu(M, *w*)

Mecanismo para codificar la Máquina de Turing Universal.

$Q = \{q_1, \ldots, q_n\}$, $q_0 = q_1$, $F = \{q_n\}$, $\Gamma = \{a_1, \ldots, a_n\}$, $\Sigma = \{0, 1\}$

Viendo el alfabeto, sabemos que vamos a tener que codificar en unario.

Por ejemplo, $q_3 = 111$

$q_5 = 11111$

$q_1 = 1$

Del conjunto {L, R}, codificaremos L = 1 y R = 11

Entonces, si tenemos por ejemplo:
$\delta(q_3, a_2) = (q_5, a_1, \mathrm{R})$
La codificación sería: 0<u>111</u>0<u>11</u>0<u>1111</u>0<u>1</u>0<u>11</u>
$q_3 \quad a_2 \quad q_5 \quad a_1 \quad \mathrm{R}$

$\delta(q_1, a_1) = (q_1, a_1, \mathrm{R}) \rightarrow 010101010110$
$\delta(q_1, a_2) = (q_3, a_1, \mathrm{L}) \rightarrow 01011011101010$

<u>Teor:</u> Sea el DFA $M \equiv (\Sigma, Q, F, q_0, \delta)$, $\exists$ TM $M' \equiv (Q', \Sigma', \delta', q_0', F', \Gamma, \not\emptyset)$ | $L(M)=L(M')$

$Q' = Q \cup \{q_f\}$, $F' = \{q_f\}$, $\Sigma' = \Sigma$, $q_0' = q_0$

$\delta'(q, a) = (\delta(q, a), a, \mathrm{R})$, $\forall q \in Q$, $\forall a \in \Sigma$
$\delta'(q, \not\emptyset) = (q_f, \not\emptyset, \mathrm{S})$, $\forall q \in F$

<u>Teor:</u> Si L es regular, entonces L es recursivamente enumerable.

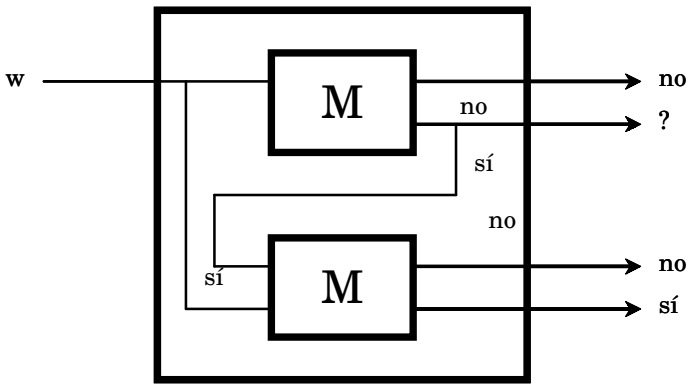
<u>Teor:</u> Si L es regular, entonces L es recursivo.

<u>Teor:</u> Una TM se comporta un PDA simulando la pila con otra cinta.

<u>Teor:</u> Si L es CFL, entonces L es recursivo

<u>Teor:</u> Si L_1 y L_2 son recursivos, $L_1 \cap L_2$ es recursivo

Sean M_1 y M_2 las TM que reconocen L_1 y L_2 respectivamente.
Si $w \in L_i \Rightarrow M_i$ para y acepta w
Si $w \notin L_i \Rightarrow M_i$ para y rechaza w



$w \in L(M) \Leftrightarrow w \in L(M_1) \wedge w \in L(M_2) \Leftrightarrow w \in L(M_1) \cap L(M_2) = L_1 \cap L_2$

Teor: Si L es recursivo sobre $\sum$, $\exists$ TM M | L=L(M). M para ante cualquier cadena. En este caso, $\overline{L}$ también para siempre.

Teor: Si L es recursivo, entonces $\overline{L}$ es recursivo, es decir, para siempre. Esto no ocurre en los recursivamente enumerables.

Teor: $\exists$ L. R. E. | $\overline{L}$ no es L. R. E.

Teor: Si dos lenguajes son recursivos, la unión también lo es.

Teor: Si dos lenguajes son recursivamente enumerables, la unión también lo es.

Teor: Si un lenguaje L y su complementario $\overline{L}$ son recursivamente enumerables, entonces L y $\overline{L}$ son recursivos.

Dados L, $\overline{L} \subseteq \sum^*$

a) L y $\overline{L}$ son recursivos

b) Ni L ni $\overline{L}$ son recursivamente enumerables.

c) Uno de los dos es recursivamente enumerable pero no recursivo y el otro no es recursivamente enumerable.

Procedimiento efectivo (o algoritmo) M.

1) Está especificado por un número finito de instrucciones (cada instrucción especificada con un número finito de símbolos)

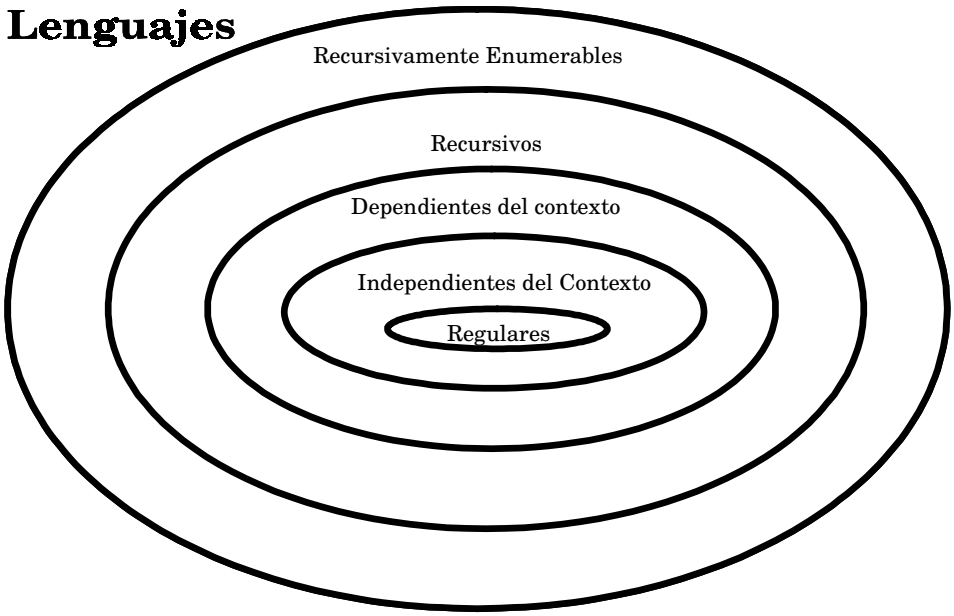
2) M produce un resultado deseado en un número finito de pasos si se ejecuta sin error

3) M puede ser realizada por un humano usando lápiz y papel sin ayuda externa.

4) No se requiere del humano, ni trucos, ni ingenuidad. A las funciones que ejecutan estos procedimientos se les llama efectivas.

Tesis de Church-Turing: La clase de las funciones efectivamente computables son las funciones Turing-computables.

Teor: Un L es recursivamente enumerable sii L = L(G), para alguna gramática G no restringida (de tipo 0)


Lenguajes
Recursivamente Enumerables
Recursivos
Dependientes del contexto
Independientes del Contexto
Regulares

Tema 6
Resolubilidad.

Dado Σ, sea $L \subseteq \Sigma^*$

Def: Se denomina X_L a la función característica del lenguaje. Se define:

$X_L : \Sigma^* \longrightarrow \{0, 1\}$

$X_L(w)=0$ si $w \notin L$, 1 si $w \in L$

Teor: X_L es computable (Turing-computable) si L es recursivo.

Teor: X_L no es computable si L es recursivamente enumerable.

Problemas

π_{AMB} Dada una CFG G, ¿es G ambigua? (sin solución)

π_{DFA} Dado un DFA M y $w \in \Sigma^*$, ¿M acepta w?

π_{CFG} Dada una CFG G y $w \in \Sigma^*$, $\text{¿}S \overset{*}{\Rightarrow} w\text{?}$

$\pi_{CFG-VAC}$ Dada una CFG G, ¿es $L(G)=\varnothing$?

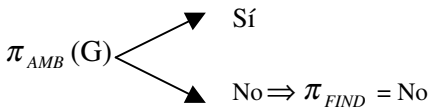
π_{HP} Dada una TM M, $w \in \Sigma^*$, ¿M para ante w?

Problemas de decisión (sin solución).

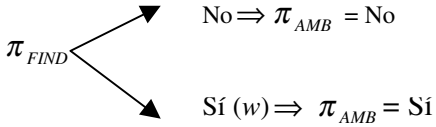
π_{FIND} Dada una CFG G, hallar $w \in \Sigma^*$ | w tiene 2 o más árboles de derivación.

Este último problema sin embargo no es de decisión.

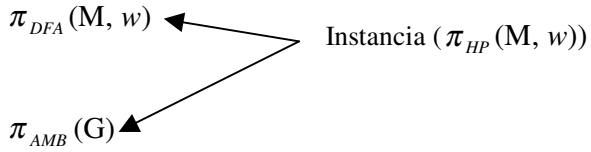
Supongamos que π_{AMB} tiene solución.



Supongamos que pudiéramos contestar a π_{FIND}.



Teor: A partir de un problema de decisión se pueden resolver problemas que no son de decisión.



Dado π, D_π={ instancias de π }

$D_\pi = S_\pi \cup N_\pi$

Donde S_π={ instancias de π con respuesta afirmativa }

N_π={ instancias de π con respuesta negativa }

$S_\pi \cap N_\pi = \varnothing$

π es resoluble sii $\exists$ TM $M \equiv (Q, \Sigma, \delta, q_0, F, \Gamma, \not{b})$, que para ante cualquier entrada $w \in \Sigma^*$, y que ante C(I) (siendo C(I) una codificación de Σ en $I \in D_\pi$ determina si la instancia pertenece o no a S_π, $I \in S_\pi$

π es resoluble sii existe un algoritmo que es capaz de responder sí o no a cada una de las $I \in D_\pi$

π es resoluble sii el lenguaje que codifica S_π es recursivo

π es irresoluble si se niega alguna de las anteriores.

Def: Dado π y π', se dice que π se reduce a π' si un algoritmo que resuelva π' pueda usarse para resolver π.

Teor: Si π se reduce a π', y π' es resoluble, entonces π es resoluble

Teor: Si π se reduce a π', y π' es irresoluble, entonces π es irresoluble

Problema de la parada

π_{HP}

Dada TM M arbitraria con alfabeto Σ y $w \in \Sigma^*$, ¿Para M ante w?

Dem: Demostración de que π_{HP} es irresoluble.

$\Sigma^* = \{w_1, w_2, w_3 .....\}$

$TM(\Sigma) = \{M_1, M_2, M_3 .....\}$

$L=\{w_i \mid w_i$ no es aceptada por $M_i\}$

Lema: L no es recursivamente enumerable.

Supongamos que L es recursivamente enumerable $\Rightarrow L=L(M_k)$

Consideremos w_k, si $w_k \in L(M_k) \Rightarrow$ w no es aceptada por $M_k \Rightarrow w \notin L(M_k)$
Lo que es absurdo.

Conclusión: L no es recursivamente enumerable.

Supongamos $M_{HP}(M, w) \longrightarrow$ sí o no.

Construyamos $M_L \mid L=M_L$ (si se consigue construir, es que algo va mal...)

Sea $w \in \Sigma^*$,

1) M_L enumera Σ^*: $w_1, w_2, \ldots$ Hasta hallar $w_k = w$
2) M_L genera M_k
3) M_L pasa (M_k, w_k) a la M_{HP}

$M_{HP}(M_k, w_k)$ → Para $\Rightarrow$ Paso 4)
→ No para $\Rightarrow M_L$ para y dice que acepta $w = w_k$

4) M_L pasa (M_k, w_k) a Mu (M.T. Universal).

$Mu(M_k, w_k)$ → Acepta $\Rightarrow M_L$ para y rechaza w_k
→ Rechaza $\Rightarrow M_L$ para y acepta w_k

$w \in L(M_L) \Leftrightarrow w \in L \Rightarrow L = L(M_L) \Rightarrow$ L es recursivo (absurdo)

Conclusión: $\nexists M_{HP}$