LENGUAJES FORMALES AUTOMATAS Y COMPUTABILIDAD

LENGUAJES, GRAMÁTICAS FORMALES, AUTÓMATAS Y EXPRESIONES REGULARES

(material de apoyo de los temas 1, 2, 3 y 4)

INDICE:

Este documento es un material de apoyo para los temas siguientes:

TEMA 1. LENGUAJES FORMALES

Definiciones:

Alfabeto (∑): Conjunto finito de símbolos gráficos. $\sum_1$ = { 0, 1 } , $\sum_2 = \{ a, b, c, d, \ldots, z \}$

Subalfabeto (∑): Subconjunto de un alfabeto. $\sum_4 = \{c, d\}$ es un subalfabeto de $\sum_2$

Palabra sobre un alfabeto (x): Conjunto ordenado, finito de símbolos del alfabeto con o sin repetición.

$x, y \rightarrow palabras \,/\, a, b \rightarrow$ símbolos $\quad x = ababa.$

Palabra vacía (λ): La palabra que no contiene símbolos. λ no ϵ ∑

Longitud de una palabra (| x |): es igual al nº de símbolos de una palabra.

$x = aabb. \rightarrow$ | x | = 4 , z = λ $\rightarrow$ | z | = 0

Palabra inversa de una dada (x^{-1}): es la palabra con todos sus símbolos a la inversa.

$x^{-1} = bbaa\,, \quad x^{-1} = aabb$

Definición Recursiva: Si | x | = 0 $\rightarrow$ x = λ , x^{-1} = λ // | x | > 0 $\rightarrow$ ∃ w palabra y ∃ a ϵ ∑ tal que $x = wa\,,\ x^{-1} = aw^{-1}$

Palabra simétrica: Una palabra es simétrica si x = x^{-1}

Ejemplo: aba es simétrica

Lenguajes universal (∑*): El conjunto de todas las palabras que se pueden formar sobre ∑ , incluyendo λ.

Concatenación de palabras (·): Es una operación entre palabras.

$x = ab\,, y = cd;\ x \cdot y = abcd$

Propiedades:

- Operación bien definida: $x, y \in \sum * \rightarrow \ x \cdot y \in \sum *$

- Asociativa: $x\ ,\ y\ ,\ z \in \sum *;\ x \cdot (y \cdot (z)) = ((x \cdot y) \cdot z)$

- Elemento neutro: $\forall \mathrm{x} \in \sum * \ \exists\ \lambda \text{ tal que } \mathrm{x} \cdot \lambda = \lambda \cdot \mathrm{x} = \mathrm{x}$

- No es conmutativa

- (∑*, ·) Tiene estructura de semigrupo.

- $|x \cdot y| = |x| + |y|$

- $\forall x\ ,\ y \in \sum^*\ (x \cdot y)^{-1} = y^{-1} \cdot x^{-1}$

- $x^i = x \cdot x \cdot x\ (i\ veces)\ \ x^0 = \lambda$

- $|x^i| = i \cdot |x|\ //\ \ x^{i+j} = xi \cdot xj$

Lenguaje Formal: Dado un alfabeto ∑ , L es un lenguaje formal sobre ∑ cuando sus palabras cumplen una propiedad. L = { x ϵ ∑* | x cumple P }

$\mathfrak{T}$ = Todos los lenguajes sobre ∑* (todos los lenguajes posibles sobre el lenguaje universal)

Operaciones con lenguajes:

Unión de lenguajes: Dado un alfabeto ∑ y L_1 y L_2 lenguajes sobre ∑, la unión de dichos lenguajes:

$$L_1 \cup L_2 = \{ x \in \sum^* \mid x \in L_1 \cup L_2 \}$$

Propiedades:

- Operación bien definida.

- Asociativa: $\forall \mathrm{L_1}, \mathrm{L_2}, \mathrm{L_3} :\ (\mathrm{L_1} \cup \mathrm{L_2}) \cup \mathrm{L_3} = \mathrm{L_1} \cup (\mathrm{L_2} \cup \mathrm{L_3})$

- Elemento neutro: El vacío $L \cup \Phi = \Phi \cup L = L$

- Es conmutativa: $L_1 \cup L_2 = L_2 \cup L_1$

- (∑*, ·) Tiene estructura de semigrupo con elemento neutro y conmutativa.

Concatenación de lenguajes: Dado un alfabeto $\sum$ y L_1 y L_2 lenguajes sobre $\sum$, la concatenación de dichos lenguajes:

$$L_1 \cdot L_2 = \{ x \in \sum^* \mid \exists\ u \in L_1, v \in L_2,\ x = uv \}$$

Propiedades:

- Operación bien definida: $x, y \in \sum^* \rightarrow x \cdot y \in \sum^*$
- Asociativa: $\forall L_1, L_2, L_3 :\ (L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$
- Elemento neutro: $\forall L \in \mathfrak{L}\ \exists \{\lambda\},\ L \cdot \{\lambda\} = \{\lambda\} \cdot L = L$
- No es conmutativa
- Tiene estructura de semigrupo con elemento neutro.

Intersección de lenguajes:

Dado un alfabeto $\sum$ y L_1 y L_2 lenguajes sobre $\sum$, la intersección de dichos lenguajes:

$$L_1 \cap L_2 = \{ x \in \sum^* \mid x \in L_1 \wedge x \in L_2 \}$$

Propiedades:

- Operación bien definida:
- Asociativa: $\forall L_1, L_2, L_3 :\ (L_1 \cap L_2) \cap L_3 = L_1 \cap (L_2 \cap L_3)$
- Elemento neutro: $\forall L \in \sum^* \rightarrow L \cap \sum^* = \sum^* \cap L = L$
- Es conmutativa: $\forall L_1, L_2, L_3 :\ (L_1 \cap L_2) \cap L_3$
- Tiene estructura de semigrupo con elemento neutro y conmutativa.

Potencia de un lenguaje:

Dado un alfabeto $\sum$ y L lenguaje sobre $\sum$

$L^i = L \cdot L \cdot L \ldots$ (i veces).

Cierre de un lenguaje (Estrella de Kleene):

$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots;$

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

Complementario de un lenguaje:

Dado $\sum$, $\bar{L}$ complementario de L

$\bar{L} = \{ x \in \sum^* \mid x \notin L \}$

Leyes de Morgan:

$$\forall\ L_1, L_2 \in \mathfrak{T}\ ;\ \overline{L_1} \cap \overline{L_2} = \overline{L_1 \cup L_2}$$

$$\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$$

TEMA 2. GRAMATICAS FORMALES:

Gramática Formal

Llamaremos Gramática Formal a la cuádrupla $\boldsymbol{G} = \{ \sum_{\boldsymbol{T}} , \sum_{\boldsymbol{N}} , S , P \}$

$\sum_T$ = Alfabeto de Símbolos Terminales: Símbolos que forman parte de las palabras del lenguaje generado por la gramática.

$\sum_N$ = Alfabeto de Símbolos No Terminales: Símbolos que participan en la elaboración de las palabras.

S = Axioma.

P = Conjunto de las reglas de producción: De la forma

$$u::= v \; ; \; u \in \sum^{+}, v \in \sum^{*}, u = xAy, \;\; x , y \in \sum^{*} , A \in \sum N$$

Ejemplo de una gramática definida por:

$\mathrm{G} = \{ \sum_{\mathrm{T}} , \sum_{\mathrm{N}} , \mathrm{S} , \mathrm{P} \}$

$\sum_T$ = { 0 , 1, 2 , 3..., 9 }

$\sum_N$ = { S , A }

S = Axioma

$$P = \left\{ \begin{array}{c} S ::= SA \mid A \\ A ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{array} \right\}$$

Lenguaje asociado a una gramática:

Dada una gramática: $\mathrm{G} = \{ \sum_{\mathrm{T}} , \sum_{\mathrm{N}} , \mathrm{S} , \mathrm{P} \}$

Se llama *" Lenguaje asociado a G"* o *" Lenguaje generado por G"* al conjunto

$$L(G) = \{ \; x \mid S \Rightarrow xy \wedge \; x \in \sum_{\mathrm{T}}{}^{*} \}$$

Es decir, el conjunto de todas las sentencias o palabras de G.

- $L(G) = \{ x \in \sum_{\mathrm{T}}{}^{*} \mid S \Rightarrow_{*} x \; , \; siendo \; \Rightarrow_{*} derivación \}$

Formas Sentenciales y sentencias: $\Rightarrow_{*}$

Dada una gramática $\mathrm{G} = \{ \sum_{\mathrm{T}} , \sum_{\mathrm{N}} , \mathrm{S} , \mathrm{P} \}$

Se denomina *forma sentencial* de G si se verifica $S \Rightarrow_{*} x \;\; , \;\; x \in \sum^{*}$

Se denomina *palabra* x de G si se verifica $S \Rightarrow_* x \wedge x \in \sum_T{}^*$

Tipos de Gramáticas:

Chomsky clasificó las gramáticas en cuatro grandes grupos (G_0, G_1, G_2, G_3), cada uno de los cuales incluye a los siguientes:

$$(G_3 \subseteq G_2 \subseteq G_1 \subseteq G_0)$$

Gramáticas Tipo 0:

Son gramáticas que generan los "lenguajes sin restricciones", sus producciones son de la forma:

$$u ::= v \; ; \; u \in \sum^+, \qquad v \in \sum^*, \quad u = xAy, \qquad x, y \in \sum^* \; , \qquad A \in \sum_N$$

Puede demostrarse que todo lenguaje representado por una gramática de tipo 0 puede ser descrito también por una gramática de estructura de frases, cuyas producciones tienen la forma $xAy::=xvy$ donde $x, y, v \in \sum^* \; y \; A \in \sum_N$.

Ejemplo 1: Dada la gramática $G = \{ \{a, b, c\}, \{A, B, C\}, A, P \}$

$$P \left| \begin{array}{l} A ::= aABC \mid abC \\ CB ::= BC \\ bB ::= bb \\ bC ::= b \end{array} \right.$$

La producción $CB ::= BC$, no está en forma de estructura de frases. Las demás producciones si están en forma de estructura de frases. Se cambian por:

$$\begin{array}{l} CB ::= XB \\ XB ::= XY \\ XY ::= BY \\ BY ::= BC \end{array}$$

La gramática queda entonces:

$$P \left| \begin{array}{l} A ::= aABC \mid abC \\ CB ::= XB \\ XB ::= XY \\ XY ::= BY \\ BY ::= BC \\ bB ::= bb \\ bC ::= b \end{array} \right.$$

Es una gramática de Tipo 0, está en forma de estructura de frases.

Gramáticas Tipo 1:

Sus producciones son de la forma: $xAy::=xvy$;

donde $x, y \epsilon \sum^*, v \epsilon \sum^+, A \epsilon \sum_N$; ($v$ no puede ser igual a λ).

Por tanto, estas gramáticas no pueden contener reglas compresoras. Se admite una excepción en el hecho de que la regla S::= λ puede pertenecer al conjunto de producciones de una gramática de Tipo 1.

Gramáticas Tipo 2:

Sus producciones son de la forma: A :: = v; donde $v \epsilon \sum^+, A \epsilon \sum_N$;

(v no puede ser igual a λ) y se admite S::= λ.

Los lenguajes representados por las gramáticas de Tipo 2 se llaman "lenguajes independientes del contexto".

Gramáticas Tipo 3:

Estas gramáticas se clasifican en dos grupos:

a) Gramáticas lineales por la izquierda:

En éstas las reglas de producción son de la forma:

$A ::= a$
$A ::= Va$
$S ::= \lambda$

Donde: $a \in \sum_T, A, V \in \sum_N$

S = axioma.

b) Gramáticas lineales por la derecha:

Cuyas reglas de producción son de la forma:

$A ::= a$
$A ::= aV$
$S ::= \lambda$

Donde: $a \in \sum_T, A, V \in \sum_N$

S = axioma.

Los lenguajes generados por una gramática de Tipo 3 se denominan "Lenguajes Regulares".

Para toda gramática lineal derecha existe una gramática lineal izquierda equivalente y para toda gramática lineal izquierda existe una gramática lineal derecha equivalente.

Arboles de derivación:

En general a toda derivación de una gramática de Tipo 2 o 3 le corresponde un árbol.

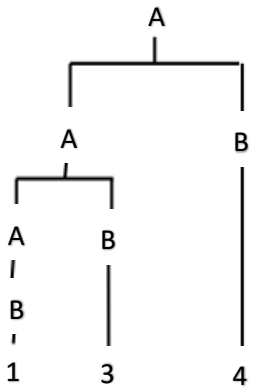
Ejemplo:

Sea la gramática G= { { 0, 1 , 2 , 3...,9 } , { A , B } , A , P }

$$P = \begin{cases} A ::= AB \mid B \\ B ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{cases}$$

Y sea la derivación: $A \rightarrow AB \rightarrow ABB \rightarrow BBB \rightarrow 1BB \rightarrow 13B \rightarrow 134$

El Árbol generado corresponde a esta derivación:



Si una gramática es de Tipo 3, el árbol correspondiente a cualquier derivación será un árbol binario.

Ambigüedad:

1. Palabra ambigua: una palabra o sentencia es ambigua si tiene, al menos, dos árboles de derivación diferentes.
2. Una gramática es ambigua si contiene al menos una sentencia o palabra ambigua.

Gramáticas Bien formadas

Una gramática bien formada es una gramática que no tiene que tener: reglas innecesarias, ni símbolos inaccesibles, ni reglas no generativas, ni reglas de redenominación, ni reglas reductoras y, si se tiene $S ::= \lambda$, hay que eliminar el axioma inducido.

1. Reglas innecesarias

Son de la forma $A ::= A \ , \ A \in N$

2. Símbolos inaccesibles:

Un símbolo no-terminal inaccesible desde del axioma se elimina.

Se quiere una gramática conexa, es decir, todos los símbolos no-terminales tienen que ser accesibles desde el axioma.

3. Reglas superfluas o no generativas:

Reglas superfluas son producciones que tienen en su cadena símbolos no-terminales que no participan en la elaboración de palabras.

4. Reglas de redenominación:

Son de la forma $A ::= B \quad | \quad A , B \in \Sigma_N$

5. Reglas reductoras:

Un producción de la forma $A ::= \lambda \mid \mathrm{A} \neq \mathrm{S}$, es una producción o regla reductora.

6. Axioma inducido:

Si se tiene en la gramática la producción $S ::= \lambda$ y el axioma S aparece en la derecha de una producción se dice que el axioma esta inducido.

TEMA 3. AUTÓMATAS FINITOS

Un autómata finito es un modelo computacional que realiza cómputos de forma automática sobre una entrada para producir una salida. Se representan mediante la quíntupla: $A = (\Sigma, Q, f, q_{0,}, F)$ donde:

$\boldsymbol{\Sigma}$: es un alfabeto, llamado "**alfabeto de entrada**"

$\boldsymbol{Q}$: es un conjunto finito no vacio llamado **conjunto de estados.**

$\boldsymbol{f}$: es el conjunto de **funciones de transición** $(f: Q\ x\ \Sigma\ \rightarrow \mathrm{Q})$

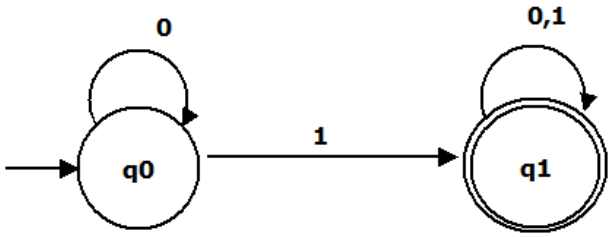
$\boldsymbol{q_0}$: es el **estado inicial** del autómata$(q_0 \in Q)$

$\boldsymbol{F}$: es el conjunto no vacío de "**estados finales**" $(F \subset Q)$

Autómata Finito Determinista (AFD)

En un autómata finito determinista, cada estado $q \in Q$ en que se encuentre el autómata, y con cualquier símbolo $a \in \Sigma$ del alfabeto de entrada, existe siempre una transición posible $f(q, a)$.

Para representar los AFD se utilizan dos métodos:



$$A = (\{0,1\}, \{q_0, q_1\}, f, q_{0,}, F)$$

La Tabla de Transición

Es una tabla que representa un AFD cuyas filas están encabezadas por los estados y los símbolos del Σ son el encabezamiento de las columnas. Tanto el estado inicial como los estados iníciales vienen marcados debidamente.

	0	1
$\rightarrow \boldsymbol{q_0}$	$\boldsymbol{q_0}$	$\boldsymbol{q_1}$
$\boldsymbol{(q_1)}$	$\boldsymbol{q_1}$	$\boldsymbol{q_1}$

El Diagrama de Transición

El segundo método de representación de los AFD es un grafo dirigido que se forma de la siguiente manera:

1. El grafo tendrá tantos nodos como $|Q|$, cada nodo será etiquetado con un elemento de Q.
2. Si $f(q_i, e_j) = q_k$, dibujaremos una rama dirigida desde el nodo $q_i$hasta el nodo q_k y se etiqueta dicha rama con e_j.
3. El estado inicial se representara con una flecha " →" apuntando al estado.
4. Los estados finales se señalarán con un doble circulo.

Definiciones adicionales:

$$f(q\ , \lambda) = q\ \forall\ q \in Q$$

$$f(q\ , ax) = f(f(q\ , a), x)\ \forall\ a \in \Sigma\ , a \in \Sigma^*, q \in Q$$

Reconocimiento de un lenguaje por parte de un AFD

Una palabra es reconocida o aceptada por un autómata si partiendo del estado inicial se llega al estado final y la concatenación de las etiquetas de las transiciones realizadas conforman la palabra.

Se llama lenguaje asociado o reconocido por un autómata al conjunto de todas las palabras aceptadas por este.

$$L = \{x \mid x\ \in \Sigma^*\ \ \&\ f(q_0, x) = \in F$$

Estados Accesibles:

Un estado es accesible si existen transiciones mediante las cuales se llegue a ese estado.

Todo estado es accesible a si mismo dado que $f(q\ ,\lambda) = q\ \forall\ q \in Q$

Autómatas Conexos

Un autómata es conexo si todos los estados de dicho autómata son accesibles desde el estado inicial tras un nº finito de transiciones.

Dado un autómata no conexo, se puede obtener un autómata que si es conexo y que reconozca el mismo lenguaje al eliminar todos los estados que no sean accesibles desde el estado inicial.

Estados Equivalentes:

Sea el autómata finito determinista $A = \{\ \Sigma\ , Q\ , f\ , q_0\ , F\ \}$
Dos estados $p\ , q\ \in Q$ son equivalentes si para toda palabra $x\ \in \Sigma^*$, se verifica que: $f(p\ , x)\ \epsilon\ F \Leftrightarrow f(q\ , x)\ \epsilon\ F$
Se representa pEq.
También se puede decir que son equivalentes en función de su longitud "n" si para toda palabra $x\ \in \Sigma^*, |\mathrm{x}| \leq \mathrm{n}$, se verifica que:

$$f(p\ , x)\ \epsilon\ F \Leftrightarrow f(q\ , x)\ \epsilon\ F$$

$$pEq \Rightarrow pE_n q\ \forall n$$

Autómatas Equivalentes:

Dados dos autómatas finitos deterministas:

$\mathrm{A} = \{\ \Sigma\ , Q\ , f\ , q_0\ , F\ \}$ y $\mathrm{B} = \{\ \Sigma', \mathrm{Q}', \mathrm{f}', \mathrm{p_0}\ , \mathrm{F}'\}$

se dice que son equivalentes entre sí, si ambos reconocen el mismo lenguaje.

Minimización de Autómatas Finitos Deterministas/ Hallar Autómata Finito Mínimo equivalente:

Dado un autómata se puede obtener un autómata equivalente con el mínimo nº de estados si hallamos los conjuntos de estados equivalentes entre si dentro del mismo autómata.

Suma Directa de Autómatas:

Dados dos autómatas finitos deterministas $\mathrm{A} = \{ \Sigma , Q , f , q_0 , F \}$ y $\mathrm{B} = \{ \Sigma', \mathrm{Q}', \mathrm{f}', \mathrm{p}_0 , \mathrm{F}' \}$ y no tienen elementos comunes (estados con el mismo nombre).

Se llama Suma Directa de A y B al autómata C = A $\oplus$ B;

$$C = A \oplus B = \{ \Sigma \cup \Sigma' , Q \cup Q', f , q_0 , F \cup F' \}$$

donde q_0 , es uno cualquiera de los estados iniciales, y f se define como:

$$\begin{matrix} f(q , a) = f_1(q , a) \; si \; q \in Q \\ f(q , a) = f_2(q , a) \; si \; q \in Q' \end{matrix} \quad \forall \; a \in \Sigma$$

El procedimiento para calcular la suma directa es:

1. Verificar que no hay estados con el mismo nombre, en caso de haberlos, renombrarlos.
2. Calcular $Q \mid E$ (Conjuntos de estados equivalentes) de ambos autómatas $(Q_1 \cup Q_2 \mid E)$
3. Construir el nuevo autómata cuyo estado inicial será uno de los dos estados iníciales anteriores.
4. Los estados finales serán aquellos que contengan estados finales.

Isomorfismo:

Dados dos autómatas finitos deterministas

$A = \{ \Sigma , Q , f , q_0 , F \}$ y $B = \{ \Sigma', Q', f', p_0 , F' \}$

se dice que son isomorfos si:

$| Q | = | Q' |$ $(tienen \; el \; mismo \; nº \; de \; estados)$

y existe una aplicación biyectiva $i: Q \rightarrow Q'$ que cumple:

- $i(q_0) = i(p_0)$. Es decir los estados iniciales son correspondientes
- $q \in F = i(q) \in F'$. Es decir los estados finales se corresponden uno a uno
- $i\big(f_{1,}(q , a)\big) = f_2(i(q) , a) \;\; \forall \, a \in \Sigma \;\; y \;\; \forall \, q \in \mathrm{Q}$

Si dos autómatas son isomorfos, serán también equivalentes.

$\boldsymbol{A. Isom. B. \rightarrow A. Eq \; B.}$

(que sean equivalentes no quiere decir que sean isomorfos)

$$\boldsymbol{A.Eq.B.} \nrightarrow \boldsymbol{A.Isom.B.}$$

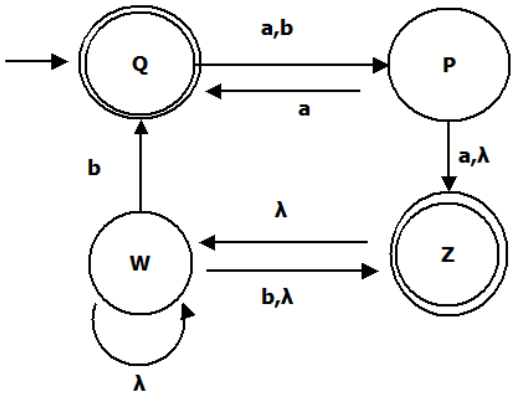
(pueden serlo, pero hay que comprobarlo previamente)

Autómata Finito No Determinista (AFND)

Es un autómata que a diferencia de los AFD tiene más de una o no tiene ninguna transición posible para un mismo símbolo. Se puede dar que:

$$f(q\,,a) = q_1 \,\&\, f(q\,,a) = q_2 \; siendo \; q_1 \neq q_2 \;\;,\;\; f(q\,,b) = \lambda$$

Un AFND es una séxtupla del tipo $A = \{\,\Sigma\,, Q\,, f\,, q_0\,, F\,, T\,\}$ donde T es el conjunto de transiciones entre estados mediante el símbolo λ.



f	a	b
Q	{ P }	{ P }
P	{ Q , Z }	∅
W	∅	{ Q , Z }
Z	∅	∅

$T = ((W, W), (W, Z), (Z, W), (P, Z))$

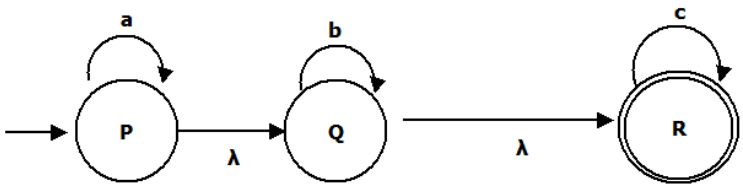
Cierre de Transiciones:

$E(q) =$ cierre de λ Transiciones desde

$q =$
$\{\, conjunto\ de\ \ estados\ a\ los\ que\ se\ llega\ por\ medio\ de\ \lambda\ transiciones\ sucesivas\ desde\ q\}$

$$\boldsymbol{f(q\,,\lambda) = q \;\forall\; q \in Q}$$

Ejemplo:



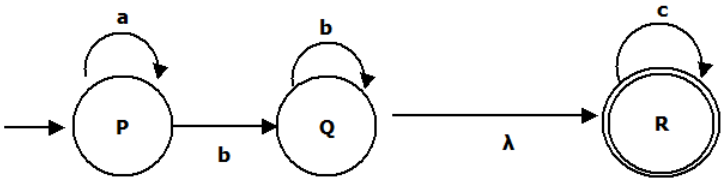
Calcular

$$E(P) = \{P, Q, R\}$$
$$E(Q) = \{Q, R\}$$
$$E(R) = \{R\}$$

Extensión de $\boldsymbol{f}$

$f'(q, x)$
$= \{estados\ a\ los\ que\ se\ llega\ por\ medio\ de\ la\ palabra\ x\ desde\ q\ incluyendo\ \lambda\ transiciones\}$

Ejemplo:



Calcular

$$f'(P, a) = \{P\}$$
$$f'(P, b) = \{P, Q, R\}$$

Dado un autómata finito no determinista $A = (\Sigma, Q, f, q_{0,}, F, T)$

x es una palabra reconocida o aceptada por AFND si:

$$f'(q_{0,}, x) = \{contiene\ algun\ estado\ final\}$$

Lenguaje reconocido por un AFND llamado *A:*

$$A\ \ si\ \ = \{x\ \in\ \Sigma^*\ |x\ es\ aceptado\ por\ A\}$$

Para que $\lambda\ \in L\ (AFND)$ $q_{0,}$ debe ser estado inicial y final.

Dado un AFND se puede obtener un AFD equivalente.

TEMA 4. LENGUAJES REGULARES:

Definición:

Una expresión regular (ER) expresa el conjunto de palabras aceptadas por autómatas finitos.

Dado el alfabeto Σ y los símbolos ϕ, λ,+, ·,*, se cumple que:

1. El símbolo ϕ es una expresión regular
2. El símbolo λ es una expresión regular
3. Cualquier símbolo a ϵ Σ , es una expresión regular
4. si α y β son ER's → α + β y α · β son ER's
5. si α ϵ ER → α* es una expresión regular
6. Solo son ER's las que se pueden obtener las que se pueden
7. obtener aplicando las reglas anteriores un nº finito de veces
8. sobre los símbolos deΣ , la palabra vacía λ y el conjunto vacio ϕ.

Propiedades de las ER

1. $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
2. $\alpha + \gamma = \gamma + \alpha$
3. $\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$
4. $\alpha \cdot (\beta + \gamma) = \alpha\beta + \alpha\gamma$
5. $\lambda^* = \lambda$
6. $\alpha \cdot \lambda = \lambda \cdot \alpha = \alpha$
7. $\alpha + \phi = \phi + \alpha = \alpha$
8. $\phi\alpha = \alpha\phi = \phi$
9. $\phi^* = \lambda$
10. $\alpha^*\alpha^* = \alpha^*$
11. $\alpha^*\alpha = \alpha\alpha^*$
12. $(\alpha^*)^* = \alpha^*$
13. $\alpha^* = \lambda + \alpha^1 + \alpha^2 + \cdots + \alpha^n + \alpha^{n+1}$ ….
14. $\alpha^* = \lambda + \alpha \cdot \alpha^*$
15. $\alpha^* = (\lambda + \alpha)^{n-1} + \alpha^n \cdot \alpha^*$
16. $(\alpha \cdot \beta)^* \cdot \alpha = \alpha \cdot (\beta \cdot \alpha)^*$
17. $(\alpha^* \cdot \beta^*)^* \cdot \alpha^* = (\alpha + \beta)^*$

Prioridad:

El orden de prioridad de las operaciones cuando aparecen varias simultáneamente es una expresión regular será el siguiente:

1) * (cierre)
2) · (concatenación)
3) + (unión)

El orden de prioridad puede modificarse mediante paréntesis.

Cada Expresión Regular define un lenguaje regular:

1. $\alpha = \phi \rightarrow L(\alpha) = \{\phi\}$
2. $\alpha = \lambda \rightarrow L(\alpha) = \{\lambda\}$
3. $\alpha = a,\ a \in \sum \rightarrow L(\alpha) = \{a\}$
4. $\alpha\ y\ \beta\ son\ ER's \rightarrow L(\alpha + \beta) = L(\alpha)\ U\ L(\beta)$
5. $\alpha\ y\ \beta\ son\ ER's \rightarrow L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
6. $\alpha^*\ es\ ER \rightarrow L(\alpha^*) = L(\alpha)^*$

Teorema de Análisis y Síntesis de Kleene.

Teorema de análisis: Todo lenguaje aceptado por un autómata finito es un lenguaje regular.

Esto está relacionado con el "problema de análisis": "Dado un autómata A encontrar la expresión regular que representa L{ A }".

Teorema de Síntesis: Todo lenguaje regular es el lenguaje aceptado por un autómata finito.

Dado un lenguaje L representado por una expresión regular, construir a partir de él un autómata A tal que L(A) = L .

Ecuaciones características.

Sea un autómata finito $A = \{\Sigma, Q, f, q_0, F\}$. Sea x_i el conjunto de las cadenas capaces de conducir al autómata desde el estado q_i hasta un estado final.

(si desde q_i no se puede acceder a un estado final, entonces $x_i = \phi$).

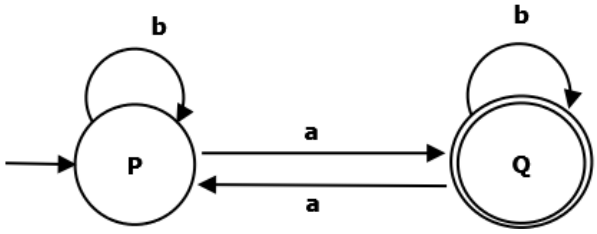
Por tanto, cada estado tendrá su ecuación característica de la forma

$$X = A \cdot X + B$$

Ejemplos:

Dados los autómatas (A, B, C) con sus diagramas de transición:

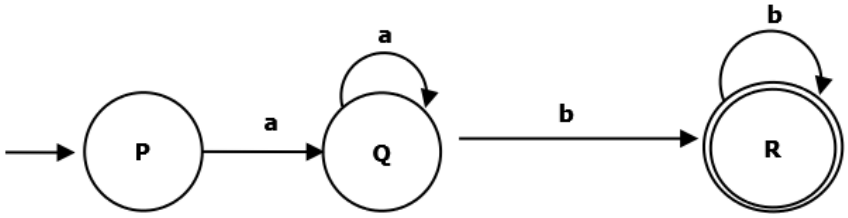
A



1) Sus ecuaciones características serán:

$$x_0 = b \cdot x_0 + a \cdot x_1 + a$$
$$x_1 = a \cdot x_0 + b \cdot x_1 + b$$

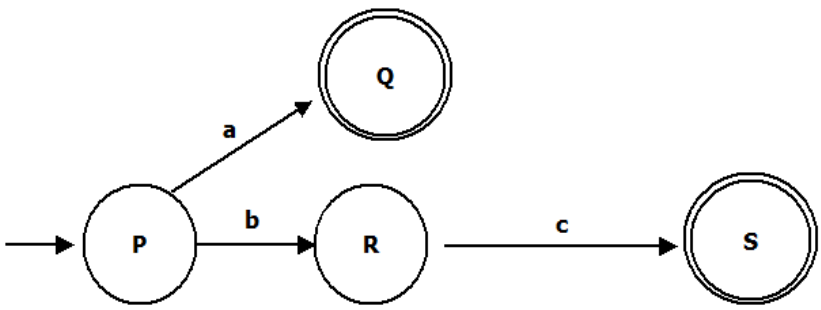
B



2) Sus ecuaciones características serán:

$$x_0 = a \cdot x_1$$
$$x_1 = a \cdot x_1 + b \cdot x_2 + b$$
$$x_2 = b \cdot x_2 + b$$

C



3) Sus ecuaciones características serán:

$$x_0 = a \cdot x_1 + b \cdot x_2 + a$$
$$x_1 = \lambda$$
$$x_2 = c \cdot x_3 + c$$
$$x_3 = \lambda$$

Las ecuaciones características no pueden resolverse despejando las variables como si se tratara de ecuaciones algebraicas.

Puesto que el signo " + " no representa la suma, sino la unión de conjuntos, el signo " · " no representa la multiplicación sino la concatenación de cadenas.

Resolución de la ecuación fundamental

Para resolver la ecuación fundamental que ya hemos visto que tiene la forma $X = A \cdot X + B$ donde X, A y B representan conjuntos de cadenas, la solución a dicha ecuación tiene la forma:

$$X = A^* \cdot B$$

La solución del problema de análisis queda simplificada al siguiente algoritmo:

1. Escribir las ecuaciones características.
2. Resolverlas utilizando la regla:

   $X = A \cdot X + B \Rightarrow X = A^* \cdot B \ (si\ \lambda\ no\ pertenece\ a\ A)$
3. Si el estado inicial es q_0, x_0 nos da las cadenas que conducen desde q_0 hasta un estado final y, por tanto, el lenguaje aceptado por el autómata.

Derivada de una expresión regular:

Para dar solución al problema de Síntesis se introduce el concepto de derivada de una expresión regular.

Derivada de una ER respecto del símbolo $a \in \sum$, es el conjunto de las colas de todas las palabras representadas por R cuya cabeza es a.

Formalmente: $D_a(R) = \{ x \mid a \cdot x \in R \}$

Dicho concepto permite obtener una gramática de Tipo 3 que describa el lenguaje L(R).

A partir de las derivadas se puede construir el autómata finito que reconozca dicho lenguaje.

Calculo de Derivadas:

Dado el alfabeto Σ y los símbolos: ϕ , λ , $+$, $\cdot$, $*$, ...

1. $D_a(\phi) = \phi$
2. $D_a(b) = \phi$
3. $D_a(a) = \lambda$
4. $D_a(R + S) = D_a(R) + D_a(S)$
5. $D_a(R \cdot S) = D_a(R) \cdot S + \beta(R) \cdot D_a(S)$

   $\beta(R) = \lambda$ si $\lambda \in R$
   $\beta(R) = \phi$ si $\lambda \notin R$
6. $D_a(R^*) = D_a(R) \cdot R^*$
7. Derivadas Sucesivas $D_{a,b}(R) = D_b(D_a(R))$
8. Dada una Expresión Regular, R cualquiera, el nº de derivadas sucesivas de R es finito.

Ejemplo:

Sea $R = abb^*$ Calcular todas las derivadas distintas de R.

$R = R_0 = abb^*$

$D_a(R_0) = D_a(abb^*) = bb^* = R_1$

$D_b(R_0) = D_b(abb^*) = \phi$

$D_a(R_1) = D_a(bb^*) = \phi$

$D_b(R_1) = D_b(bb^*) = b^* = R_2$

$D_a(R_2) = D_a(b^*) = \phi$

$D_b(R_2) = D_b(b^*) = D_b(b) \cdot b^* = \lambda \cdot b^* = b^* = R_2$

No hay más Derivadas. $D_\lambda(R_0) = R_0$

$GLD = \{ \{ a, b \}, \{ R_0 , R_1 , R_2 \}, R_0 , P \}$

$$P = \left| \begin{array}{l} R_0 ::= aR_1 \\ R_1 ::= bR_2 \mid b \\ R_2 ::= bR_2 \mid b \end{array} \right.$$

Una vez obtenidas las derivadas, construir el autómata tal que L(A) = R

$$A = \{ \Sigma , Q , f , q_0 , F \}$$

Σ = Es el alfabeto de R $\{ a , b \}$

Q= {Son las Distintas Derivadas $\{ R_0 , R_1 , R_2 \}$

f = está definida $f(R , a) = S$, si $D_a(R) = S$

q_0 = $D_\lambda(R) = R$

F = $D_x(R)$ tal que $\lambda \in D_x(R)$