TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INTEGRANTES:

CUANENEMI CUANALO MARIO ALBERTO	181080030
FERMIN CRUZ ERIK	181080007
GUTIERREZ ARELLANO RAFAEL	181080022
PEREZ ARMAS FAUSTO ISAAC	181080037

ISC-6AM

LENGUAJES Y AUTOMATAS I

M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

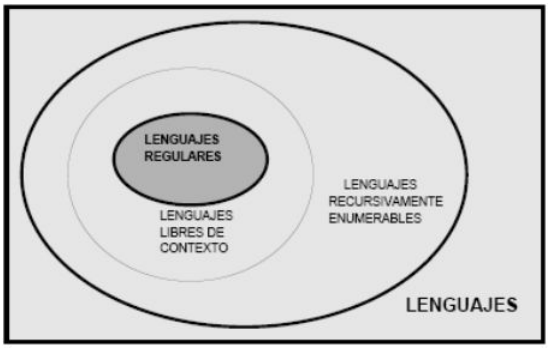
SEP 2020 / FEB 2021

ACTIVIDAD SEMANA 6

Cuananemi Cuanalo Mario Alberto

Gramática Regular

Los LR en la jerarquía de Chomsky La clasificación de lenguajes en clases de lenguajes se debe a N. Chomsky, quien propuso una jerarquía de lenguajes, donde las clases más complejas incluyen a las más simples.



Los “lenguajes regulares” es la clase más pequeña, e incluye a los lenguajes más simples. Por ejemplo, el conjunto de todos los números binarios. Los “lenguajes libres de contexto” incluye a los LR. Por ejemplo, la mayoría de los lenguajes de programación. Los “lenguajes recursivamente enumerables” que incluyen a los dos anteriores.

GRAMATICAS REGULARES - EXPRESIONES REGULARES Gramáticas Las gramáticas formales definen un lenguaje describiendo cómo se pueden generar las cadenas del lenguaje.
Una gramática formal es una cuadrupla G = (N, T, P, S) donde
- N es un conjunto finito de símbolos no terminales
- T es un conjunto finito de símbolos terminales $N \cap T = \varnothing$
- P es un conjunto finito de producciones

Cada producción de P tiene la forma
$\alpha \rightarrow \beta$, $\alpha = \phi A \rho$ y $\beta = \phi \omega \rho$ ϕ, ω, $\rho \in (N \cup T)^*$ y A es S ó $A \in N$
- S es el símbolo distinguido o axioma $S \notin (N \cup T)$

Restringiendo los formatos de producciones permitidas en una gramática, se pueden especificar cuatro tipos de gramáticas (tipo 0, 1, 2 y 3) y sus correspondientes clases de lenguajes.

Gramáticas regulares (Tipo 3)
Generan los lenguajes regulares (aquellos reconocidos por un autómata finito). Son las gramáticas más restrictivas.
El lado derecho de una producción debe contener un símbolo terminal y, como máximo, un símbolo no terminal. Estas gramáticas pueden ser: - Lineales a derecha, si todas las producciones son de la forma $A \in N \cup \{S\}$ $A \rightarrow aB$ ó $A \rightarrow a$ $B \in N$ $a \in T$ (en el lado derecho de las producciones el símbolo no terminal aparece a la derecha del símbolo terminal) - Lineales a izquierda, si todas las producciones son de la forma $A \in N \cup \{S\}$ $A \rightarrow Ba$ ó $A \rightarrow a$ $B \in N$ $a \in T$ (en el lado derecho de las producciones el símbolo no terminal aparece a la izquierda del símbolo terminal) En ambos casos, se puede incluir la producción

S → ε, si el lenguaje que se quiere generar contiene la cadena vacía. Por ejemplo las siguientes gramáticas G1 y G2, son gramáticas regulares lineales a derecha y lineales a izquierda respectivamente, que generan el lenguaje L = {a2n / n ≥ 0} G1 = ({A, B}, {a}, P1, S1) G2 = ({C, D}, {a}, P2, S2) donde P1 es el cjto. donde P2 es el cjto. S1 → ε S2 → ε S1 → aA S2 → Ca A → aB C → Da A → a C → a B → aA D → Ca

Autómatas Finitos

Autómatas Finitos.

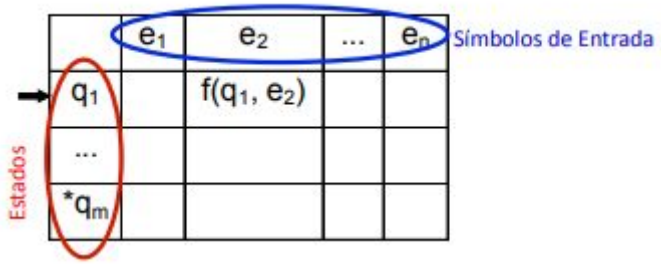
Representación

• Se pueden representar mediante:
1. Diagramas de transición o
2. Tablas de transición

1. Diagramasde transición:
Nodos eDquetados por los estados (qi ∈ Conjunto de estados)

• Arcos entre nodos qi a qj eDquetados con ei(ei es un símbolo de entrada) si existe la transición de qi , a qj con ei
• El estado inicial se señala con →
• El estado final se señala con * o con doble círculo

2. Tablas de transición:
• Filas encabezadas por los estados (qi ∈ Conjunto de estados)
• Columnas encabezadas por los símbolos de entrada (ei ∈ alfabeto de entrada)



Autómatas Finitos Deterministas (AFD)

Este tipo de autómatas admite su definición de dos maneras bien diferentes:: Como autómatas traductores o reconocedores. La definición como autómatas traductores continua a la definición de las máquinas secuenciales, y se los podría definir como una subclase de estas, ya que los autómatas finitos tendrían como limitante no poder iniciar desde cualquier estado como lo hacen en las máquinas secuenciales.

La forma que adoptaremos para la definición de los autómatas finitos deterministas es como autómatas reconocedores, ya que se ajusta con los contenidos de la informática teórica y utilización que se les da dentro del diseño de los analizadores léxicos.

Estos autómatas solo se limitarán a aceptar o no una determinada cadena recibida en la entrada, por lo tanto podemos decir que la salida de los mismos solo tendrá dos valores posibles aceptar o no aceptar a la palabra de entrada.

Al igual que en las máquinas secuenciales, estos autómatas transitarán entre un conjunto finito de estados posibles, a medida que reciban sucesivamente los caracteres de entrada, en un instante determinado de tiempo el autómata solo podrá estar en uno y solo uno de los estados posibles.

Una característica importante de este tipo de autómatas es el determinismo, lo cuál significa que estando en un estado y recibiendo una entrada del exterior el autómata tendrá la posibilidad de transitar a uno y solo un estado del conjunto de estados posibles.

Los autómatas finitos deterministas quedarán formalmente definida mediante una quíntupla como sigue:

$$AFD = (\sum, Q, q_0, F, f)$$

donde:

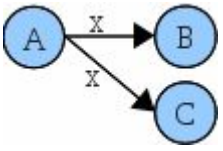
$\sum$	Alfabeto de símbolos de entrada.
Q	Conjunto finito de estados
q_0	$q_0 \varepsilon Q$ – estado inicial previsto
F	$F \subseteq Q$ - es el conjunto de estado finales de aceptación.
f	Función de transición de estados definida como f: Q x $\sum$ ——— Q

Autómatas finitos no deterministas

Autómata finito no determinista. Es el autómata finito que tiene transiciones vacías o que por cada símbolo desde un estado de origen se llega a más de un estado destino.

Los AFND son definiciones no tan deseables dentro de los lenguajes regulares porque dificultan su implementación tanto mecánica como informática; aunque en la mayoría de las transformaciones a lo interno de los LR (expresiones regulares a AF, gramáticas regulares a AF) conducen a AFND. Los AFND, por tanto, son imprescindibles en el análisis lexicográfico y el diseño de los lenguajes de programación.

Sea un autómata finito definido por la 5-tupla *A*=<*Q, T, g, F, q0*>, donde *Q* es el conjunto de estados, *T* el alfabeto de símbolos terminales, la relación de transiciones $g \subseteq Q \times (T \cup \{\xi\}) \times Q$ ó $g = \{(q_i, x, q_j) \mid q_i, q_j \in Q ; x \in T\}$ (léase: del estado *qi* mediante el terminal *x* se va a *qj*), *F* son los estados finales o de llegada dentro de *Q*, *q0* es el estado inicial o de partida; se dice que *A* es un **autómata finito no determinista** (AFND) si y sólo si existen en *g* al menos una de las siguientes transiciones:

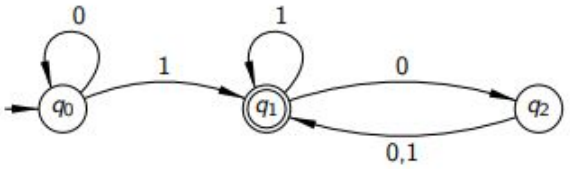
- 

ó <*qi,x,qj*> y <*qi,x,qk*> son transiciones de *g* y $q_j \neq q_k$.
- <*qi,* ξ *,qj*>.

Consecuencias

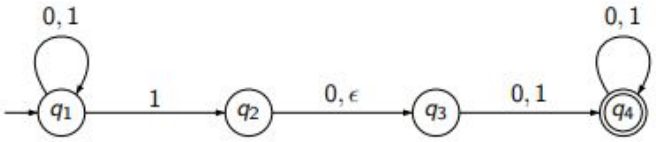
La definición formal de AFND se basa en la consideración de que a menudo según los algoritmos de transformación de expresiones y gramáticas regulares a AF terminan obteniéndose autómatas con transiciones múltiples para un mismo símbolo o transiciones vacías. Independientemente que sean indeseables, sobre todo para la implementación material, fundamentalmente mecánica, de los autómatas finitos, son imprescindibles durante la modelación de analizadores lexicográficos de los elementos gramaticales de los lenguajes de programación, llamados tókenes, como literales numéricos, identificadores, cadenas de texto, operadores, etc.

Automata finito Determinista:



- La computación del autómata con entrada 011 es (q_0, q_0, q_1, q_1) que me dice la secuencia de estados por los que pasa con entrada 011
- Cada entrada me da exactamente una computación. Tengo siempre como mucho una opción desde un estado si leo un símbolo ← Esto se llama determinismo

Automata finito no determinista



- Desde q_1 con el símbolo 1 hay dos opciones posibles
- Desde q_2 hay una posibilidad de moverse sin leer ningún símbolo (la marcada como ϵ)

A continuación se presentan algunos conceptos básicos necesarios para la comprensión de los ejercicios que se presentan en las secciones subsecuentes. Símbolo es un signo que representa algo abstracto. En este material, símbolo se referirá a un caracter alfanumérico. Ejemplos a, b, 1, 0, x, y, z, 9,

Alfabeto es un conjunto de símbolos y normalmente se denota con la letra Σ. Ejemplos Σ = {a,b,c,…z} Σ = {1,2,3,…9} Σ = {0,1} Σ = {a,b} Cadena o palabra es un conjunto de símbolos de algún alfabeto Σ concatenados entre sí, es decir uno enseguida del otro. Ejemplos Para el alfabeto Σ = {a,b,c,…z} algunas cadenas son: ab, z, cc, abc, abab Para el alfabeto Σ = {0,1} algunas cadenas son: 0, 1, 01, 000, 010

Cadena Vacía ε, es la cadena que no contiene ningún símbolo. Lenguaje es un conjunto de cadenas o palabras definido en un alfabeto Σ. Ejemplos Si Σ = {0,1} podríamos definir los lenguajes “conjunto de cadenas en Σ que terminan en 0” algunos de las palabras del lenguajes serían: 0, 10,00,010,100, 110…

Fermin Cruz Erik

GRAMÁTICA REGULAR

La gramática regular en lo autómatas se refiere a una gramática que puede ser clasificada de 2 maneras: de izquierda o de derecha. Estas gramáticas pueden dar paso a los lenguajes que de igual manera llevan su nombre: lenguajes regulares.

se utilizan principalmente para analizar el contenido que tenemos en una cadena de caracteres por medio de patrones, estos patrones son usados para identificar una determinada combinación de estos caracteres dentro de un string o cadena de tipo texto.

En la gramática regular de derecha tenemos que A entonces a. si es un símbolo que no termina en N y termina en Σ.

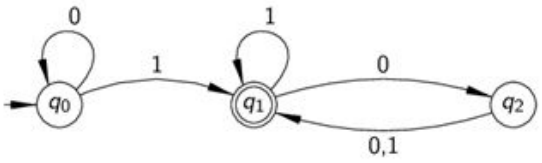
Mientras que en la gramática de izquierda tenemos algunas reglas, que son:

- A $\rightarrow a$, donde A es un símbolo no-terminal en N y a uno terminal en Σ
- $A \rightarrow Ba$, donde A y B pertenecen a N y a pertenece a Σ
- $A \rightarrow \varepsilon$, donde A pertenece a N.

AUTOMATA FINITO

Se trata de un modelo computacional que realiza cómputos o instrucciones de manera automática en una entrada para producir una salida.
Este modelo está conformado por varias partes como por ejemplo un alfabeto, un conjunto de estados finitos, una función de transiciones, un estado inicial y por último, un estado final.

Recordad los autómatas finitos deterministas



- La computación del autómata con entrada 011 es (q_0, q_0, q_1, q_1) que me dice la secuencia de estados por los que pasa con entrada 011
- Cada entrada me da exactamente una computación. Tengo siempre como mucho una opción desde un estado si leo un símbolo ← Esto se llama determinismo

Su funcionamiento se basa valga la redundancia, en una función de transición, que recibe a partir de un estado inicial una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un estado final o de aceptación, que representa la salida.

La finalidad de los autómatas finitos es la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

Estos tienen su origen en las máquinas electromecánicas, en las cuales el matemático ruso Andréi Márkov formalizó un proceso llamado cadena de Markov, donde la ocurrencia de cada evento dependía de una cierta probabilidad del evento anterior. esta particularidad de recordar los estados anteriores, les sirve de ucho a los autómatas para poder tomar el siguiente estado, ya que deben conocer el anterior para saber cuál es el posterior.

AUTÓMATA FINITO NO DETERMINISTA

Son autómatas que, a diferencia de los autómatas finitos deterministas, poseen al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición posible.

Estos pueden darse en cualquiera de estos dos casos:

1.- Que existan transiciones del tipo $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;

2.- Que existan transiciones del tipo $\delta(q, \varepsilon)$, siendo q un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Los autómatas no deterministas pueden pasar por varios estados a la vez, generando una ramificación de las configuraciones existentes en un momento dado. Asimismo, en estos podemos aceptar la existencia de más de un nodo inicial.

Estos autómatas tienen un funcionamiento que es por decirlo así, de la siguiente manera:

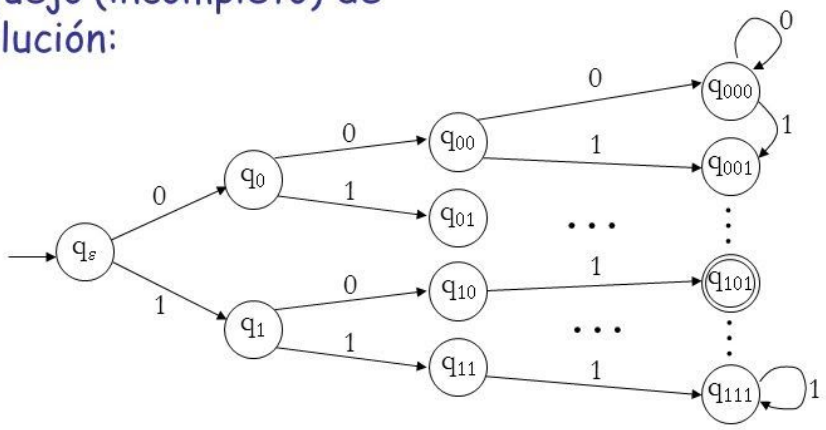
La máquina comienza en el estado inicial que se le ha especificado y lee una cadena de caracteres pertenecientes al alfabeto. El autómata utiliza la función de transición de estados T para determinar el siguiente estado, usando el estado actual y el símbolo que acaba de leer o la cadena vacía. Sin embargo, "el estado siguiente de un Autómata finito no determinado no solo depende del evento de la entrada actual, sino que también depende de un número arbitrario de los eventos de entrada posterior.

Hasta que se producen estos acontecimientos posteriores no es posible determinar en qué estado se encuentra la máquina". Cuando el autómata ha terminado de leer, y pasa a un estado de aceptación, se dice que acepta la cadena, de lo contrario se dice que la cadena de caracteres es rechazada.

AFND: Autómatas finitos *no deterministas*

Veamos un ejemplo más de AFD. Con Σ={0,1}, queremos que acepte el lenguaje de los strings que terminan en 101.

Bosquejo (incompleto) de la solución:



Gutierrez Arellano Rafael

Gramáticas regulares

Las gramáticas formales definen un lenguaje describiendo cómo se pueden generar las cadenas del
lenguaje.
Una gramática formal es una cuadrupla G = (N, T, P, S) donde:

- N es un conjunto finito de símbolos no terminales
- T es un conjunto finito de símbolos terminales $N \cap T = \varnothing$
- P es un conjunto finito de producciones Cada producción de P tiene la forma $\alpha \rightarrow \beta$, $\alpha = \phi A \rho$ y $\beta = \phi \omega \rho$ $\phi, \omega, \rho \in (N \cup T)^*$ y A es S ó $A \in N$
- S es el símbolo distinguido o axioma $S \notin (N \cup T)$

Restringiendo los formatos de producciones permitidas en una gramática, se pueden especificar cuatro tipos de gramáticas (tipo 0, 1, 2 y 3) y sus correspondientes clases de lenguajes.

Gramáticas regulares (Tipo 3)

Generan los lenguajes regulares (aquellos reconocidos por un autómata finito). Son las gramáticas
más restrictivas. El lado derecho de una producción debe contener un símbolo terminal y, como máximo, un símbolo no terminal. Estas gramáticas pueden ser:
- Lineales a derecha, si todas las producciones son de la forma
Existe un algoritmo que permite obtener una gramática regular que genera un lenguaje regular dado a partir del autómata finito que reconoce ese lenguaje. Los pasos a seguir son los siguientes:

1) Asociar al estado inicial el símbolo distinguido S.

2) Asociar a cada estado del autómata (menos el estado inicial) un símbolo no terminal. Si al estado inicial llega algún arco asociar también un símbolo no terminal (además del símbolo distinguido). No asociar símbolo no terminal a aquellos estados finales de los que no salen arcos.

3) Para cada transición definida δ (ei , a) = ej , agregar al conjunto de producciones, la producción A → aB, siendo A y B los símbolos no terminales asociados a ei y ej respectivamente. Si ej es un estado final, agregar también la producción A → a. Si ej es el estado inicial (tiene dos símbolos asociados, el distinguido y uno no terminal), utilizar el símbolo no terminal (de esta manera se evita que el símbolo distinguido aparezca a la derecha de una producción).

4) Si el estado inicial es también final agregar la producción S → ε

Autómata finito determinista.

Un autómata finito determinista es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

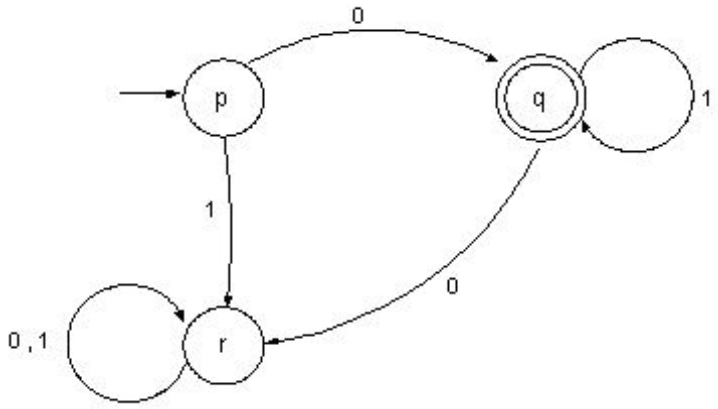
Este tipo de autómatas admite su definición de dos maneras bien diferentes:: Como autómatas traductores o reconocedores. La definición como autómatas traductores continua a la definición de las máquinas secuenciales, y se los podría definir como una subclase de estas, ya que los autómatas finitos tendrían como limitante no poder iniciar desde cualquier estado como lo hacen en las máquinas secuenciales.

La forma que adoptaremos para la definición de los autómatas finitos deterministas es como autómatas reconocedores, ya que se ajusta con los contenidos de la informática teórica y utilización que se les da dentro del diseño de los analizadores léxicos.

Estos autómatas sólo se limitarán a aceptar o no una determinada cadena recibida en la entrada, por lo tanto podemos decir que la salida de los mismos solo tendrá dos valores posibles aceptar o no aceptar a la palabra de entrada.

Una característica importante de este tipo de autómatas es el determinismo, lo cual significa que estando en un estado y recibiendo una entrada del exterior el autómata tendrá la posibilidad de transitar a uno y solo un estado del conjunto de estados posibles.

Con respecto al conjunto de estados ahora se pueden clasificar en tres tipos: Estado Inicial, que es por donde comenzará la ejecución de la máquina; Estados finales o de aceptación que será un subconjunto del conjunto de estados por los que transitó la máquina, y si cuando se hayan terminado de procesar todos los símbolos de entrada y no reste ningún símbolo por leer, la máquina quede posicionada en uno de estos estados de aceptación, se concluirá que la cadena procesada será aceptada por el autómata. y Estados Intermedios, que tienen comportamiento idéntico a los definidos en las máquinas secuenciales.



Autómata finito no determinista.

Un autómata finito no determinista es un autómata finito que, a diferencia de los autómatas finitos deterministas, posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición δ posible.

La definición formal de AFND se basa en la consideración de que a menudo según los algoritmos de transformación de expresiones y gramáticas regulares a AF terminan obteniéndose autómatas con transiciones múltiples para un mismo símbolo o transiciones vacías. Independientemente que sean indeseables, sobre todo para la implementación material, fundamentalmente mecánica, de los autómatas finitos, son imprescindibles durante la modelación de analizadores lexicográficos de los

elementos gramaticales de los lenguajes de programación, llamados tókens, como literales numéricos, identificadores, cadenas de texto, operadores, etc.

Los AFND son definiciones no tan deseables dentro de los lenguajes regulares porque dificultan su implementación tanto mecánica como informática; aunque en la mayoría de las transformaciones a lo interno de los LR (expresiones regulares a AF, gramáticas regulares a AF) conducen a AFND. Los AFND, por tanto, son imprescindibles en el análisis lexicográfico y el diseño de los lenguajes de programación.

Perez Armas Fausto Isaac

GRAMATICAS REGULARES / EXPRESIÓN REGULAR

Una Expresión regular es un generador de lenguajes sobre un alfabeto con ciertas características (restricciones). La aplicación de las expresiones en programación, definen a las ER como: Las expresiones regulares se usan para analizar el contenido de cadenas de caracteres por medio de patrones. Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.

Si ∑ es un alfabeto, ∑* denota el conjunto de todas las cadenas sobre el alfabeto y se conoce como cerradura de ∑ o lenguaje universal sobre ∑

∑* es infinito para cualquier ∑

Ejemplos de expresiones regulares:

1.- Cadenas sobre {a,b} que contengan bb

L={a,b}* {bb}{a,b}*

2.- cadenas que inician con aa o terminan con bb sobre {a,b}

L={aa}{a,b}* u {a,b}*{bb}

3.- Si L1={bb} y L2={e,bb,bbbb} son lenguajes sobre {a,b}

L1* y L2* representan cadenas con numero par de b’s

4.- {aa,ab,ba,bb} son cadenas de longitud par sobre {a,b}

{a,b}{aa,ab,ba,bb} y

{aa,ab,ba,bb}{a,b} son cadenas de longitud non

5.- Cadenas que inician y terminan con a y contienen al menos una b

{a} {a,b}* b {a,b}* {a}

6.- (a u b)* aa (a u b)* cadenas con aa

(a u b)* bb (a u b)* cadenas con bb

(a u b)* aa (a u b)* u (a u b)* bb (a u b)*

cadenas con aa o bb

7.- Contienen exactamente dos b's sobre {a,b}

a*ba*ba*

8.- Cadenas con al menos dos b's

(a u b)* b (a u b)* b (a u b)*

9.- Un numero par de b's

a* (a* b a* b a*) *

10.- Sobre {a,b} que no contengan aa

b* (ab+)* u b* (ab+)* a

(b u ab)* u (b u ab) * a

11.- Cadenas que contienen bc sobre {a,b,c}

(a u b u c) * bc (a u b u c) *

12.- Cadenas sobre {a,b,c} que no contienen bc

c* (b u ac*) *

Principales equivalencias

r u s = s u r

(r u s) u t = r u (s u t)

r u ∅ = ∅ u r = r

r u r = r

rℰ= ℰr = r

r ∅= ∅ r = ∅

(r s) t = r (s t)

r (s u t) = rs u rt

(r u s) t = rt u st

r * = r* r* = (r*) * = (Ɛ u r)*

∅ * = Ɛ* = Ɛ

r * = e u r r*

(r u s)* = (r* u s *) * = (r*s*)* = r*s)*r*= r*(sr*)*

r*r=rr*=r+

r(sr)* = (rs)* r

(r*s)* = Ɛ u (r u s)*s

(rs*)* = Ɛ u r(r u s)

1.- (ba^+) (a*b* u a*) = $(ba)^*$ b a^+ (b* u Ɛ)

=(ba)* b aa* (b* u Ɛ)

=(ba)* ba a* (b* u Ɛ)

=$(ba)^+$ a* (b* u Ɛ)

=$(ba)^+$ (a* b* u a*)

2.- (a u b) * = (b* (a u Ɛ) b*)*

= (b* a u b*) b*)*

= (b*ab* u b*b*) *

= (b*ab* u b*) * (r u Ɛ)* = (r*)* = r*

= b* (ab* u Ɛ) *

= b* ((ab*)*)*

= b* (ab*)* r*(sr*)* = (r u s)*

= (b u a) *

= (a u b) *

Autómata finito determinista

Un **autómata finito determinista** (abreviado **AFD**) es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

Definición formal

Formalmente, se define como una 5-tupla $(Q, \Sigma, q_0, \delta, F)$ donde:[1]

- {\displaystyle Q\,} es un conjunto de estados;
- {\displaystyle \Sigma \,} es un alfabeto;
- {\displaystyle q_{0}\in Q} es el estado inicial;
- {\displaystyle \delta \colon Q\times \Sigma \to Q} es una función de transición;
- {\displaystyle F\subseteq Q} es un conjunto de estados finales o de aceptación.

En un AFD no pueden darse ninguno de estos dos casos:

- Que existan dos transiciones del tipo $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;
- Que existan transiciones del tipo $\delta(q, \varepsilon)$, donde ε es la cadena vacía, salvo que q sea un estado final, sin transiciones hacia otros estados.

Autómata finito no determinista

Un **autómata finito no determinista** (abreviado **AFND**) es un autómata finito que, a diferencia de los autómatas finitos deterministas (AFD), posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición $\delta(q,a)$ posible.

En un AFND puede darse cualquiera de estos dos casos:

- Que existan transiciones del tipo $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;
- Que existan transiciones del tipo $\delta(q, \varepsilon)$, siendo q un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Cuando se cumple el segundo caso, se dice que el autómata es un **autómata finito no determinista con transiciones vacías** o **transiciones ε** (abreviado **AFND-ε**). Estas transiciones permiten al autómata cambiar de estado sin procesar ningún símbolo de entrada. Considérese una modificación al modelo del autómata finito para permitirle ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada.