Sintaxis y Semántica del Lenguaje

Wilo Carpio Cáceres

2013

A la memoria de Roberto, mi amado padre !

En la década del 40, nace el homo digitalis . . .

El exponencial crecimiento de Internet y la descomunal proliferación de las redes sociales que conforman el sistema nervioso de la actual e-cultura, donde multiplican diversos formatos de comunicación, los cuales se remontan a los albores de la cultura humana, cuando en la Grecia del siglo VII AC, nace la primera globalización del lenguaje, producida por la explosión comercial, la aparición de la moneda y la internacionalización de los mercados de aquel entonces.
El contacto con otras culturas y el interés por venderles, provocaba en aquellos comerciantes la necesidad conocer con quién estaban tratando, cuales eran sus valores y convicciones propias, adoptando sus costumbres en el modo de usar sus formas de comunicación. Está antigua globalización generó una suerte de reingeniería del lenguaje, que vigorizó los nacionalismos, revitalizó las identidades de grupos étnicos, los sentimientos religiosos, los diversos fundamentalismos y una renovada presencia del espiritualismo.

El tiempo pasó, ahora es inexorable e indispensable contar con capacidad de acceso al actual formato de la comunicación humana basado en información digitalizada de carácter científico, técnico y humanístico de componentes culturales, donde interactúan como naturales administradores, solo algunas pocas empresas transnacionales y/o naciones privilegiadas, creadoras y dueñas de las tecnologías, desarrolladas sobre sus diversas redes de comunicación, tales como la WEB, o como Internet, para ofrecer, vender, consumir, utilizar información tanto para imponer sus productos comerciales, como para trasmitir sus productos culturales y por ende, para implantar sus proyectos.

Tal estructura digitalizada va transformando en realidad palpable la mutación de aquel lejano homo erectus hacia el nuevo formato humano del homo digitalis, quién como morador artífice de la aldea global, es el único habitante dotado de la potencialidad necesaria para lograr, por un lado la supervivencia menos traumática en la sociedad digitalizada y por otro, para ser el paladín estructurador de esta e-sociedad buscadora de la coexistencia más digna de cada componente de la raza humana.

En el ámbito globalizado resulta cada vez más difícil "aferrarse" a algo razonablemente estable por su validez y permanencia en el tiempo, pues pareciera que estamos migrando a una sociedad nihilista donde existe la "creencia" de que todos los valores no tienen sentido, que nada puede ser totalmente conocido ni comunicado, puesto que el hombre no logrará conocer la verdad cuando se sumerge en la virtualidad de la e-sociedad, donde nada es establecido, todo es cuestionado, todo lo que se consideraba "cierto" o "tradicional" ha sido derribado, invadido por la cultura posmodernista, que dió origen a nuevo ser humano, al hedonista e individualista "hombre postmoderno", en concreto al “homo digitalis”.

En la e-sociedad, los componentes de la e-cultura, no solo carecen de toda regla absoluta en el cual estén de acuerdo sus miembros, sino que cada componente, crece según los intereses de estrategias políticas patrimoniales de los países que administran los e-systems con los que manejan los gobiernos de los demás países consumidores.

Personalmente, anhelo que la proliferación de la información digital como herramientas de apertura de las fronteras del mundo, debería imponer un profundo análisis de los valores sociales y económicos, de acuerdo a la mayor demanda de transparencia e idoneidad, destacando la real importancia del lenguaje como bien, sobre todo lo referido a la calidad de la vida humana y su comportamiento.

$\theta, \rho\ \Phi\ \Sigma \in L_1 \neq w^2 \geq \leq \forall \Leftrightarrow > < \cap \cup \exists\ \subseteq \notin\ \delta$

Teoria de Autómatas

Wilo Carpio Cáceres

2013

TEORIA DE LOS AUTOMATAS

Autómata es una acepción que viene del griego automatos (αὐτόματος) que significa espontáneo o con movimiento propio, puede referirse a:

Los autómatas son una suerte de algoritmos operativos que emulan un elemento motriz de estos mecanismos analíticos, capaces de funcionar como operadores gramaticales.

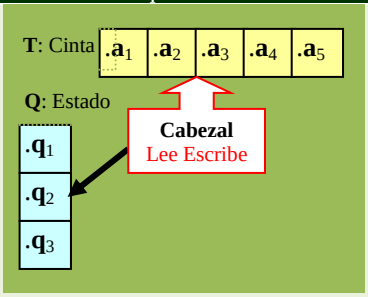
Siguiendo a **Noah Chomsky** podemos clasificarlos en función de su complejidad, determinada por sus capacidades de transición, de su dispositivo de memoria y las capacidades de inspección en su memoria. La jerarquía gramatical implica una jerarquía equivalente en los autómatas.

- **Autómata**: Máquina que imita la figura y los movimientos de un ser animado.
- **Autómata programable**: Equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real procesos secuenciales.
- **Teoría de autómatas**: Estudio matemático de máquinas abstractas que pueden funcionar automáticamente en forma permanente, mediante programas operativos de actividad y objetivo previsto; como en esta teoría que entiende al autómata como mecanismos analíticos de funcionamiento automático para procesar datos en forma de cadenas de un lenguaje

0	Irrestricta	Máquinas de Turing
1	Irrestricta con memoria limitada	Máquinas de Turing con cinta acotada
2	Sensible al contexto con borro	Máquinas de Turing con dominio total
3	Sensible al contexto no reductiva	Máquinas de Turing con “espacio lineal”
4	Libres de contexto	Autómatas de pila no-deterministas
5	Libres de contexto deterministas	Autómatas de pila deterministas
6	Lineales	Autómatas lineales
7	Regulares	Autómatas regulares

MAQUINA DE TURING

Obra del filósofo matemático Alan Mathison Turing (1912 1954), quien plantea la formalización conceptual de algoritmo y computación, base de la **Tesis de Church-Turing**, que postula que cualquier modelo computacional existente tiene las mismas capacidades algorítmicas, o un subconjunto, de las que tiene una MT del grupo de **AUTÓMATAS GENERALES**, como modelo menos restrictivo, capaz de procesar los lenguajes generados por gramáticas:

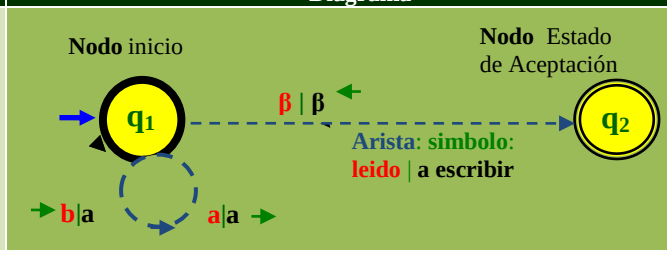
Arquitectura	Caracteristicas
	❑ **Cabezal lectura/escritura**: Movible en ambas direcciones izquierda o derecha y por cada movimiento lee o escribe un símbolo. ❑ **Memoria**: Soporte de los estados de la máqina. ❑ **Cinta**: Soporte en entrada o salida, con una colección infinita en ambas direcciones de celdas, capaces de almacenar cada una, un único símbolo, que puede accederse en cualquier orden. • Ambos lados de la cadena de la cinta posee un símbolo **β** de espacio en blanco. • Según la longitud de la cinta estas máquinas pueden ser: ○ **Autómata Linealmente Acotado**: La cinta está cotada en ambos lados. ○ **Máquinas de Turing**. La cinta es infinita

DEFINICION: La **MAQUINA DE TURING** Es una Tupla: $M= (Q,\Sigma, T, s, \beta, F, \delta)$

Definición	Descripción
Q	Conjunto finito de estados.
Σ	Alfabeto de Entrada
T	Alfabeto de la Cinta
.s ∈ Q	Estado Inicial
β ∈ T	ó **Δ:** Símbolo Blanco (**β ∉ Σ,** ó **Σ ⊆ T – { β }**)
F ⊆ Q	Conjunto de estados finales de aceptación. Incluye **Q**
δ	**Función Parcial o Transición**. Matemáticamente define a la MT como: ❑ **DETERMINISTA MT_D**: Tiene una sola alternativa de transición: **(q, σ) → {(p, t, X)}** donde: Par **(q, σ)**: Parámetros de la posición actual del cabezal ❑ **q ∈ Q: Estado Actual**, que actua como estímulo de la MT ❑ **σ ∈ T: Símbolo de cinta** que señala el cabezal o reacción de la MT Terna **(p, t, X)**: ❑ **p ∈ Q:** Estado siguiente. (Ej: $.q_1$, $.q_2$, $.q_3$,..q_n...) ❑ **t ∈ T**: Símbolo de cinta que reemplaza al que existía en la celda. ❑ **X∈ { I, D, P }:** Movimiento del cabezal (**I**zquierda, **D**erecha, **P**arado) ❑ **NO DETERMINISTA MT_{ND}**: Tiene mas de una alternativa de transición: **(q,σ) →{(p_1,t_1, X_1), (p_2, t_2, X_2), . . (p_n, t_n, X_n) }.** ❑ Naturalmente la **MT** es siempre **MT_D**, así, siempre hay una **MT_D** equivalente a una **MT_{ND}**. ❑ El lenguaje universal **Σ^***, segun Turing, abarca los subconjuntos de palabras, que la **MT**: ○ **Acepta**: Si llega alguno de los estados finales ○ **Rechaza**: Para un símbolo leído, llega a un estado de transición sin definir, que bloquea la MT ○ Que no acepta o rechaza, o sea que no detiene a la MT

CONSTRUCCIÓN DE UNA MAQUINA DE TURING

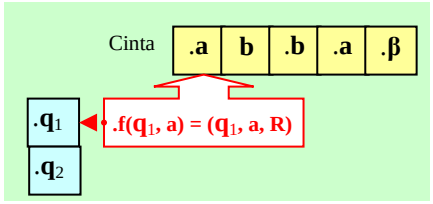
Ejemplo: **MT = (Q, Σ, T, s, β, F, δ) = ({ q_1, q_2 }, { a, b}, { a, b, β}, q_1, β, { q_2 }, δ)**, y funciones de transición

Pares	Diagrama	Matriz
$(q_1,a)\rightarrow(q_1 a\ R)$ $(q_1,b)\rightarrow(q_1 a\ R)$ $(q_1,\beta)\rightarrow$ $\{(q_2 \beta\ L)\}$		

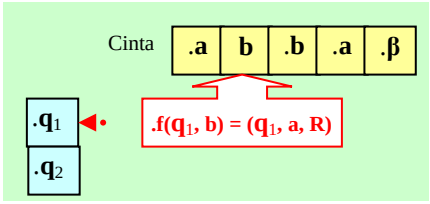
δ	.a	.b	. β
q_1	$.q_1$ a R	$.q_1$ a R	$.q_2$ β L
q_2			

El proceso de la cadena **abba** podemos describirla como:

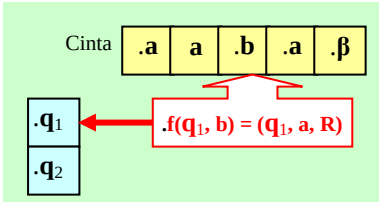
Estado inicial:



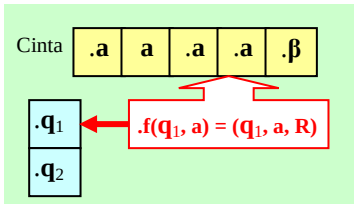
Movimiento 1



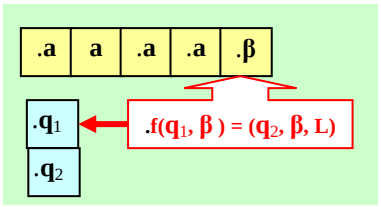
Movimiento 2



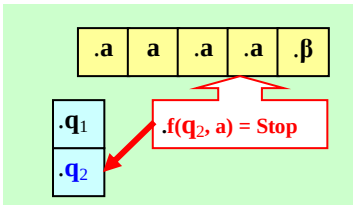
Movimiento 3



Movimiento 4



Estado final:



En síntesis: El proceso de la cadena **abba** es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado final
.a b b a β	**a b b a β**	**a a b a β**	**a a a a β**	**a a a a β**	**a a a a β**
$.f(q_1,a)=(q_1,a,R)$	$f(q_1,b)=(q_1,a,R)$	$f(q_1,b)=(q_1,a,R)$	$f(q_1,a)=(q_1,a,R)$	$f(q_1, \beta)=(q_2, \beta,L)$	$f(q_2,a)$=Stop

Descripcion instantanea:

Representa cada instante de las sucesivas configuraciones por las que pasa la máquina de Turing, bajo el formato de:

- **Par de valores**: (q,w), $q \in Q$, $w \in \Sigma$: q: Estado Actual, w: Entrada No Leída o Restante; ├: Símbolo entre dos descripciones.

  Para el ejemplo anterior la descripción instantánea es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado final
$(q_1,\ a\ bba\beta)$	$\vdash (q_1,\ abba\beta)$	$\vdash (q_1,\ aaba\beta)$	$\vdash (q_1,\ aaaa\beta)$	$\vdash (q_1,\ aaaa\beta)$	$\vdash (q_2,\ aaaa\beta)$

  Todo este proceso se puede representar como: $(q_1,\ a\ bba\beta) \vdash^* (q_2,\ aaaa\beta)$

  Parcialmente este proceso se puede representar como: $(q_1,\ a\ bba\beta) \vdash^+ (q_1,\ aaba\beta)$

- **Cadena Única**: a q b, donde $q \in Q$, $a,b \in \Sigma$: q: Estado Actual, a, b: Cadena Leída; ├: Símbolo entre dos descripciones.

  Para el ejemplo anterior la descripción instantánea es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado final
q_1a bbaβ	$\vdash$ a$q_1$bbaβ	$\vdash$ aa$q_1$baβ	$\vdash$ aaaq_1aβ	$\vdash$ aaaa$q_1\beta$	$\vdash$ aaaq_2aβ

  Todo este proceso se puede representar como: q_1a bba$\beta \vdash^*$ aaaq_2aβ

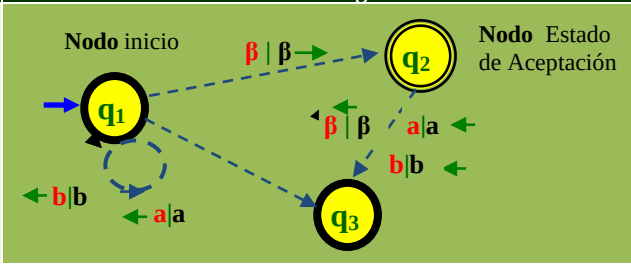
  Parcialmente este proceso se puede representar como: q_1a bba$\beta \vdash^+$ aa$q_1$baβ

CONSTRUCCIÓN DE UNA MAQUINA DE TURING

Ejemplo: **MT = (Q, Σ,T, s, β, F, δ) = ({q_1, q_2, q_3}, {a, b}, {a, b, β}, q_1, β, { q_2 }, δ)**, y funciones de transición:

Pares	Diagrama	Matriz

(q_1,a)→(q_1a I)
(q_1,b)→(q_1b I)
(q_1,β)→(q_2β D)
(q_2,a)→(q_3a I)
(q_2,b)→(q_3b I)
(q_2,β)→(q_3β I)



. δ	.a	.b	. β
q_1	.q_1 a I	.q_1 b I	.q_2 β D
q_2	.q_3 a I	.q_3 b I	.q_3 β I
q_3			

El proceso de la cadena **aababb** es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Movimiento 5	Estado final
.aababbβ	aabbabbβ	aababbβ	aababbβ	βaababb	βaababb	βaababb
.f(q_1,a)=(q_1,a,I)	f(q_1,b)=(q_1,b,I)	f(q_1,a)=(q_1,a,I)	f(q_1,a)=(q_1,a,I)	f(q_1,β)=(q_2,β,D)	f(q_2, a)=(q_3, a,I)	f(q_3, β)=Stop

Descripcion instantanea:
Representa cada instante de las sucesivas configuraciones por las que pasa la máquina de Turing, bajo el formato de:

- **Par de valores**: **(q,w), q∈Q, w∈Σ,** donde: **q**: Estado Actual, **w**: Entrada No Leída o Restante; ├: Símbolo entre descripciones.

Estado inicial Movimiento 1 Movimiento 2 Movimiento 3 Movimiento 4 Movimiento 5 Estado final

(q_1, aab a bb) ├ (q_1, aa b abb) ├ (q_1, a a babb) ├ (q_1, a ababb) ├ (q_1, β aababb) ├ (q_2, a aababb) ├ (q_3, β aababb)

Todo este proceso se puede representar como:
(q_1, aab a bb) ├*(q_3, β aababb)

parcialmente este proceso se puede representar como:
(q_1, aab a bb) ├+ (q_1, a a babb)

- **Cadena Única**: **a q b**, donde **q∈Q, a,b∈Σ,** donde: **q**: Estado Actual, **a, b**: Cadena Leída; ├: Símbolo entre dos descripciones.

Estado inicial Movimiento 1 Movimiento 2 Movimiento 3 Movimiento 4 Movimiento 5 Estado final

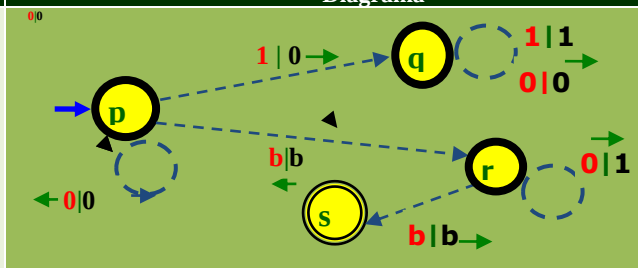
aab q_1 a bb ├ aa q_1 b abb ├ a q_1 a babb ├ q_1 a ababb ├ q_1 β aababb ├ q_2 a aababb ├ q_3 β aababb

Todo este proceso se puede representar como:
aab q_1 a bb ├* q_3 β aababb

Parcialmente este proceso se puede representar como:
aab q_1 a bb ├+ a q_1 a babb

CONSTRUCCIÓN DE UNA MAQUINA DE TURING

Ejemplo: **MT = (Q, Σ,T, s, β, F, δ) = ({ 0, 1, b}, { 1, 0 }, b, { p, q, r, s }, p, f, { s })**, y funciones de transición

Pares	Diagrama
(p,1)→(q,0,D) (p,0)→(p,0,I) (p,b)→(r,b,I) (q,1)→(q1,D) (q,0)→(q,0,D) (r,0)→(r,1,D) (r,b)→(s,b,D)	

Matriz			
.f	1	0	.b
.p	.q 0 D	.p 0 I	.r b D
.q	.q 1 D	.q 0 D	.p 0 I
.r		.r 1 D	.s b P
.s			

El proceso de la cadena **11** es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Movimiento 5	Estado final
.b 1 1 b	b 0 1 b	b 0 1 b	b 0 1 0	b 0 0 0 b	b 0 0 0 b	b 0 0 0 0 b
.f(p,1)=(q,0,D)	f(q,1)=(q,1,D)	f(q,b)=(p,0,I)	f(p,1)=(q,0,D)	f(q,0)=(q,0,D)	f(q,b)=(s,0,I)	f(s,0)=

Descripcion instantanea:
Representa cada instante de las sucesivas configuraciones por las que pasa la máquina de Turing, bajo el formato de:

- **Par de valores**: **(q,w), q∈Q, w∈Σ,** donde: **q**: Estado Actual, **w**: Entrada No Leída o Restante; ├: Símbolo entre descripciones.

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Movimiento 5	Estado final
(p, b 1 1 b)	├ (q, 0 1 b)	├ (q, 0 1 b)	├ (p, 0 1 0)	├ (q, 0 0 0 0)	├ (q, 000 b)	├ (s, 0000)

Todo este proceso se puede representar como:
(p, b 1 1 b) ├*(s, 0000)

Parcialmente este proceso se puede representar como:
(p, b 1 1 b) ├+ (q, 0 1 b)

- **Cadena Única**: **a q b**, donde **q∈Q, a,b∈Σ,** donde: **q**: Estado Actual, **a, b**: Cadena Leída; ├: Símbolo entre dos descripciones.

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Movimiento 5	Estado final
b p 1 b b	├ b 0 q 1 b	├ b 0 1 q b	├ b 0 p 1 0	├ b 0 0 q 0 b	├ b 0 0 0qb	├ b00sb0

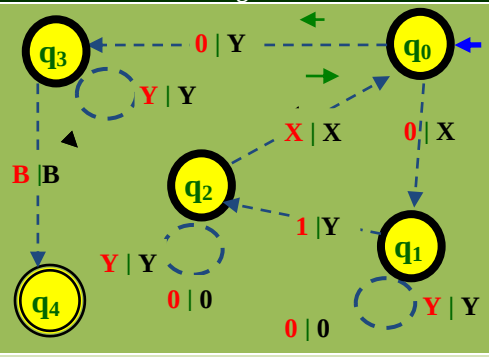
Todo este proceso se puede representar como:
b p 1 b b ├* b00sb0

Parcialmente este proceso se puede representar como:
b p 1 b b ├+ b 0 1 q b

CONSTRUCCIÓN DE UNA MAQUINA DE TURING

Ejemplo: **MT = (Q,Σ,T,s,β,F,δ)=({ 0,1}, {0,1,X,Y,B }, B, { q_o,q_1,q_2,q_3,q_4},q_o,f,{q_4},δ)**,

Para procesar: **$L = (0^n1^n|n\geq1)$** con las funciones de transición:

Pares	Diagrama
(q_o,0)→(q_1XR), (q_o,Y)→(q_3YR) (q_1,0)→($q_1$0R), (q_1,1)→(q_2YL), (q_1,Y)→(q_1YR) (q_2,0)→($q_2$0L), (q_2,X)→(q_oXR), (q_2,Y)→(q_2YL) (q_3,Y)→(q_3YR), (q_3,B)→(q_4BR)	

Matriz					
.f	0	1	X	Y	B
q_o	q_1XR			q_3YR	
q_1	$q_1$0R	q_2YL		q_1YR	
q_2	$q_2$0L		q_oXR	q_2YL	
q_3				q_3YR	q_4BR
q_4					

La secuencia de movimientos para procesar la cadena

- **0011**, es:
  q_o0011 ⊢ X$q_1$011 ⊢ X0$q_1$11 ⊢ X$q_2$0Y1 ⊢ q_2X0Y1 ⊢Xq_o0Y1 ⊢XXq_1Y1 ⊢XXY$q_1$1 ⊢XXq_2YY ⊢
  Xq_2XYY ⊢ XXq_oYY ⊢ XXYq_3Y ⊢ XXYYq_3B ⊢ XXYYBq_4B La cadena **0011**, es aceptada
- **0010**, es:
  q_o0010 ⊢ X$q_1$010 ⊢ X0$q_1$10 ⊢ X$q_2$0Y0 ⊢ q_2X0Y0 ⊢Xq_o0Y0 ⊢XXq_1Y0 ⊢XXY$q_1$0 ⊢XXY0q_1B
  La cadena **0010**, no es aceptada

CONSTRUCCIÓN DE UNA MAQUINA DE TURING

Ejemplo: supongamos dos máquinas MT_1 y MT_2 cuyo funcionamiento individual sería:

MT_1 =({q_1,q_2,q_3,q_4},{a},{a,β},q_1,β,{q_4},δ)		
. δ	.a	. β
. q_1	.q_2 a D	.q_2 β D
. q_2	.q_2 a D	.q_3 β I
. q_3	.q_4 a D	.q_4 β D

Estado inicial
aaaβaaβ
.f(q_1,a)=(q_2,a,D)

Movimiento 1
a**a**aβaaβ
f(q_2,a)=(q_2,a,D)

Movimiento 2.
aa**a**βaaβ
f(q_2,a)=(q_2,a,D)

Movimiento 3
aaa**β**aaβ
f(q_2, β)=(q_3, β,I)

Movimiento 4
aa**a**βaaβ
f(q_3,a)=(q_4 a,D)

Estado final
aaa**β**aaβ
f(q_4, β)=Stop

(q_1, a aa β aa) ⊢ (q_1, a a a β aa) ⊢ (q_1, a a a β aa)

MT_2 =({p_1,p_2},{a},{a,β},p_1,β,{p_2},δ)		
. δ	.a	. β
. p_1	.p_2 a D	.p_2 a D
. p_2		

Estado inicial
. **a**aaβaaβ
.f(p_1,a)=(p_2,a,D)

Estado final
a**a**aβaaβ
f(p_2,a) = Stop

(p_1, a aa β aa) ⊢ (p_2, a a a β aa)

Si estas dos máquinas **MT_1** y **MT_2**, funcionan compartiendo la misma cinta, de modo que cuando **MT_1** termina, empezará **MT_2** operando sobre la cinta con el cabezal ubicado en la posición que dejó **MT_1**

Estado inicial	Movimie 1	Movimie 2	Movimie 3	Movimie 4	Estado final	Estado Inic	Estado final
. **a**aaβaaβ	a**a**aβaaβ	aa**a**βaaβ	aaa**β**aaβ	aa**a**βaaβ	aaa**β**aaβ	aaa**β**aaβ	aaaa**a**aβ
f(q_1,a)=(q_2,a,D)	f(q_2,a)=(q_2,a,D)	f(q_2,a)=(q_2,a,D)	f(q_2, β)=(q_3, β,I)	f(q_3,a)=(q_4 a,D)	f(q_4, β)=Stop	f(p_1, β)= (p_2,a,D)	f(p_2,a)=Stop

MAQUINA UNIVERSAL DE TURING: MUT

Para demostrar que no todo problema bién definido tiene una solución algorítmica, Turing supone que una forma de representación algorítmica de un problema, es su **MT**, la cual codifica y la registra como entrada de otra **MT**, que constituye una **MUT**, modelo, capaz de determinar, si se detendría o no ante cualquier secuencia de entrada en la **MT** codificada.

La **MUT** es un dispositivo análogo a una computadora, para efectuar cualquier cómputo y que puede simular el comportamiento de cualquier otra **MT**, ajustando la arquitectura de su cinta para registrar:

- La descripción de otra **MT**
- El contenido de la cinta de tal **MT**

En este ámbito, Turing plantea el "**Problema de parada de la MT**":

- Suponiendo que una MT_1 recibe como entrada la codificación de otra máquina MT_0 y observa si este se va a detener para cualquier entrada válida de la misma y que luego será capaz de hacerlo para la codificación de si misma.
  Así poder ver si MT_1 se parará para cualquier secuencia de entrada.

- Luego supone una MT_2 que se para si MT_0 no se para y viceversa; para lo cual debía MT_2 entrar en un bucle infinito, cuando se detenga MT_0.
  De este modo, se produce el absurdo de que, cuado MT_2 opere sobre su propia codificación, MT_2 se detendrá si MT_2 no se detiene y no se detendrá si MT_2 se detiene.

Por tanto:

- Tal maquina no puede existir.
- El problema de parada de la MT, no puede ser resuelto algorítmicamente.
- Partiendo del problema de parada de la MT, tampoco son resueltos los problemas de:
  - **Equivalencia** de las gramáticas G_{LDC} libres del contexto.
  - **Ambigüedad** de las G_{LDC}.
  - **Comprobacion de pertenencia:** Verifica si la cadena $w \in L$ que genera la gramatica **G**:
    - Para G_{LDR} Gramaticas Irrestrictas: En los lenguajes irrestrictos L_0, generados por las gramáticas irrestrictas G_0, o G_{LDR} debido a su permisividad en reglas compresoras; cuando se quiere generar cadenas de longitud $\leq n$, las secuencias intermedias del proceso de derivación, pueden disminuir de longitud, luego, al entrar en un bucle infinito, será dudoso el final del proceso. Esto es un problema de parada de la MT.

    - Para el caso de los lenguajes L_{DC}, L_{IC} y L_{RE}, tal Comprobacion de Pertenencia, siempre tiene solución.

LENGUAJE RECONOCIDO POR LA MAQUINA DE TURING

Un LENGUAJE RECURSIVAMENTE ENUMERABLE LRE, es un lenguaje formal **tipo-0** en la Jerarquía de Chomsky también llamado **parcialmente decidible** o **Turing-computable** porque es un lenguaje para el cual existe **una máquina de Turing que acepta y se detiene con cualquier cadena del lenguaje** y puede parar y rechazar, o bien iterar indefinidamente, cadenas que no pertenece al lenguaje, en contraposición a los lenguajes recursivos en cuyo caso se requiere que la máquina de Turing pare en todos los casos.

Todos los lenguajes regulares, dependientes e independientes de contexto y recursivos son LRE. Estos, son cerrados con las siguientes operaciones.

Si **L** y **P** son dos **LRE**, entonces los siguiente lenguajes son también **LRE**:

- El cierre estrella **L*** de **L**
- La concatenación **L** o **P** de **L** y **P**
- La unión $L \cup P$ de **L** y **P**
- La intersección $L \cap P$ de **L** y **P**
- Los **LRE** no son cerrados con la diferencia ni el complementario.
- **L** / **P** puede no ser recursivamente enumerable
- **L** es recursivamente enumerable si y solo si **L** es también recursivo

Si luego de poner en marcha la máquina, con el cabezal posicionado sobre el primer símbolo de la cadena registrada sobre la cinta, se para despues de recorrer toda la palabra, esta habrá sido aceptada y se cumplirá que el lenguaje aceptado por la máquina de Turing **M = (Q, ∑, T, s = q_1, β, F, δ)**, será:

Recursivo enumerable Nivel pragmático

$$L(M) = \{w \in \Sigma^* \mid q_1\, w \vdash^* w_1\, p\, w_2 \quad \text{para} \quad p \in F \quad y \quad w_i \in T^*\}$$

Ejemplo

MT = ((a, b, c), (**a, b, c**, β), f; β, q_o, {**q_o, q_1, q_2, q_3**}, {**q_3**}), puede procesar el lenguaje de formato:

$$L = (a^n b^n c^n \mid n \geq 1)$$

.f	A	.b	.c
q_o	q_oaD	q_1aD	q_ocI
q_1	q_oaI	q_1bD	q_2cD
q_2			**q_3**cD
q_3			

Los movimientos de la máquina y sus correspondientes descripciones instantaneas serán:

Estado inicial Movimiento 1 Movimiento 2 Movimiento 3 Movimiento 4 Movimiento 5 Movimiento 6

. **a a b b c c | a a b b c c | a a b b c c | a a a b c c | a a a b c c | a a a b c c | a a a b c c β |**

.f(**q_o, a**)=(**q_o,a,D**) | f(**q_o, a**)=(**q_o,a,D**) | f(**q_o, b**)=(**q_1,a,D**) | f(**q_1, b**)=(**q_1,b,D**) | f(**q_1, c**)=(**q_2,c,D**)) | f(**q_2, c**)=(**q_3,c,D**) | f(**q_3, β**)=STOP

(q_o,**a**abbcc) ├ (q_o,a**a**bbcc) ├ (q_o,aa**b**bcc) ├ (q_1,aaa**b**cc) ├ (q_1,aaab**c**c) ├ (q_2,aaabc**c**) ├ (**q_3**,aaabcc**β**) ├ STOP

En síntesis las descripciones instantáneas serían:

(qo,aabbcc) ├(qo,aabbcc) ├(qo,aabbcc) ├(q1,aaabcc) ├(q1,aaabcc) ├(q2,aaabcc) ├(q3,aaabccβ) ├Stop

Ejemplo: La **MT = ({0, 1, b}, { 0, 1 }, b, { p, q }, p, f, { q })**, acepta al lenguaje **L = {0*1}**, cuando

. δ	1	0	B
. p	.q 1 D	.p 0 D	.p b P
. q	.p 1 P	.p 1 P	.p b P

Estado inicial	Movimiento 1	Movimiento 2	Estado Final
.b 0 0 0 1 b	b 0 0 0 1 b	b 0 0 0 1 0	b 0 0 0 b
.f(p,0)=(p,0,D)	f(p,0)=(p,0,D)	f(p,1)=(q,1,D)	f(q,b)=(p,b,P)

Ejemplo: El lenguaje **a*** generado sobre el alfabeto ∑ = **{ a, b }**, será aceptado por la máquina de Turing: **M = ({ q_1, q_2, q_3 }, { a, b }, { a, b, β }, β, { q_3 }, δ)** donde δ es definido por

- **δ(q_1, a) = (q_1, a, D), δ(q_2, a) = (q_2, a, D)**
- **δ(q_1, b) = (q_2, b, D), δ(q_2, b) = (q_2, b, D)**
- **δ(q_1, β) = (q_3, β, D), δ(q_2, β) = (q_3, β, D)**

. δ	.a	.b	B
q_1	q_1, a, D	q_2, b, D	q_3, β, D
q_2	q_2, a, D	q_2, b, D	q_3, β, P
q_3			

Estado inicial	Movimiento 1	Movimiento 2	Estado Final
.b a a b	b a a b	b a a b	b a a b β
.f(q_1, a)=(q_1, a, D)	f(q_1, a)=(q_1, a, D)	f(q_1, b)=(q_2, b, D)	f(q_2, β)=(q_3, β, P)

Ejemplo: Verificar si la máquina de Turing: M= ((a, b), (a, b, β), f; β, q_o, {q_o, q_1, q_2, q_3}, {q_3}), puede procesar el lenguaje de formato: L = ($a^n b^n$ | n ≥ 1) , para las siguientes funciones:

F	A	.b	B
q_o	q_oaI		q_1Bi
q_1	q_1aD	q_1bI	q_1βD
q_2	q_2aD	q_2bD	q_3βD
q_3			

(q_o,aaabbb) ├ (q_o,βaaabbb) ├ (q_1,βaaabbb) ├ (q_1,βaaabbb) ├ (q_1,aaabbb) ├ (q_1,aaabbb) ├
(q_2,aaabbb) ├ (q_2,aaabbb) ├ (q_2,aaabbb) ├ (q_2,aaabbb) ├ (q_2,aaabbbβ) ├ STOP

Ejemplo: Verificar si la máquina de Turing: M= ({a, b}, {a, b, β}, f; β, q_o, {q_o, q_1, q_2 }, {q_2}), es capaz de verificar el lenguaje L = ($a^n b^m$ / n,m ≥ 1), cuando sus transiciones son

F	A	.b	B
q_o	q_oaD	q_1aI	q_1Bd
q_1	q_0aD	q_2aI	q_0Ad
q_2			

(q_o,aabbb) ├ (q_o,aabbb) ├ (q_0,aabbb) ├ (q_1,aaabb) ├ (q_0,aaabb) ├ (q_0,aaabb) ├ (q_1,aaaab) ├
(q_0,aaaab) ├ (q_0,aaaab) ├ (q_1,aaaaa) ├ (q_0,aaaaa) ├ (q_0,aaaaaβ) ├ (q_2,aaaaabβ) ├ STOP

RESULUBILIDAD y COMPLEJIDAD COMPUTACIONAL

TEORÍA DE LA COMPUTABILIDAD Analiza la posibilidad de solucionar problemas mediante algoritmos, por ello es importante descubrir problemas imposibles, de modo de evitar resolverlos algoritmicamente, ahorrando asi, tiempo y esfuerzo. A tal fin, esta teoría clasifica los problemas de acuerdo a su **grado de imposibilidad**, en:

- **COMPUTABLES (decidibles, resolubles o recursivos: L_{RE})** Poseen un algoritmo que siempre los resuelve cuando hay una solución y es capaz de distinguir casos que no la tienen.

  Sus complementos son aceptables por la MT, son **Lenguajes Decibles**" y todas sus cadenas producen la detención de la MT; como el problema de decisión que plantea la verificación de pertenencia al lenguaje siempre tiene una solución algorítmica, es "**Resoluble o Decible**" y se afirma que:

  - El lenguaje dependiente de contexto L_{DC} es subconjunto del lenguaje recursivo: L_{RE}.
    : $L_{DC} \subseteq L_{RE}$,
  - El L_{RE} es cerrado para las operaciones de intersección, unión, concatenación y Estrella de Kleene; pero no para el complemento.

  FUNCION COMPUTABLE:
  La función **f** es "Turing Computable" cuando existe una función $f_{(w)}$ para todo **w** perteneciente al dominio de **f**.

  PROBLEMA DE DECISION:
  Su resultado puede expresarse mediante una función de rango binario de valores 0/1, true/false.

  RESOLUBILIDAD O DECIBILIDAD:
  Si el problema de decisión es "Turing Computable" o simplemente "Computable" se dice que es "Decible" o "Resoluble", caso contrario es "Indecible" o "Irresoluble".

  LENGUAJE RECURSIVAMENTE ENUMERABLE
  La gramática irrestricta G_0 genera un lenguaje recursivamente enumerable L_0 o L_{RE}, cuyas cadenas pueden generarse aplicando reglas gramaticales recursivas sobre la cinta de la MT, que es capaz de reconocer palabras del L_{RE}, por tanto, sus cadenas serán **aceptables** por tal MT.
  Cuando, existe la posibilidad en la acción de MT de aparecer bucles infinitos, al procesar los **complementos de un lenguaje**, implica que no es un L_{RE} y que no son aceptables por la MT.

- **SEMICOMPUTABLES** Poseen un algoritmo capaz encontrar una solución si es que existe, pero carecen de algoritmo que determine cuando la solución no existe (en cuyo caso el algoritmo para encontrar la solución entraría a un bucle infinito).

  Ejemplo: Problema de parada o listable, recursivamente enumerable o reconocible, porque si se listan sus casos posibles del problema, es posible reconocer a aquellos que sí tienen solución.

- **INCOMPUTABLES** Carecen de algoritmo que los resuelva, aunque tengan o no solución.
  Ejemplo: Problema de la implicación lógica, que determina cuándo una proposición lógica es un teorema; para este problema no hay ningún algoritmo que en todos los casos pueda distinguir si una proposición o su negación es un teorema.

Los problemas incomputables subdividen los problemas más difíciles que otros, por ejemplo, si una persona sabe sumar, es mas fácil enseñarle a multiplicar haciendo sumas repetidas, de manera que multiplicar se reduce a sumar. Este es el **CONCEPTO DE REDUCIBILIDAD**, que plantea que un problema **A** se reduce al problema **B** bajo la suposición de que se sabe resolver **B** es posible resolver **A**; esto se denota por $A \leq_t B$, que significa que el problema **A** no es más difícil de resolver que el problema **B**.

TEORÍA DE LA COMPLEJIDAD COMPUTACIONAL

Clasifica los problemas por clases de complejidad, de acuerdo a su dificultad, pues, aun cuando un problema sea computable, puede que no sea posible resolverlo en la práctica si se requiere mucha memoria o tiempo de ejecución.

Esta teoría estudia las necesidades de memoria, tiempo y otros recursos computacionales para resolver problemas; así, es posible explicar por qué unos problemas son más difíciles de resolver que otros. La complejidad computacional abarcan la **aproximación** del:

- **Tiempo**: Al número de **pasos de ejecución** que un algoritmo emplea para resolver un problema, o sea el tiempo que una maquina, como la de Turing, ocupa hasta detenerse.

- **Espacio**: A la cantidad de **memoria usada** para resolver el problema. En el caso de una máquina se refiere a la cantidad de celdas que utiliza.

Así, los algoritmos de tiempo polinómico son cerrados respecto a la composición, luego, cuando se escribe una función con un tiempo polinómico determinado y considerando que las llamadas a esa misma función son constantes y, de tiempo polinómico, entonces el algoritmo completo es de tiempo polinómico.

El análisis computacional, requiere del **modelo de computadora** estudiada en términos de tiempo, expresado como una función polinomial, supuesta como **determinista**, pues, dado el estado actual de la computadora y las variables de entrada, existe una única acción posible que la computadora puede tomar y **secuencial** porque realiza las acciones una después de la otra.

Alguna de las clases de complejidad existentes, abarca la clase:

P (tiempo **P**olinomial) para resolver problemas:

- De decisión resueltos en una máquina determinista secuencial en un período de tiempo polinomial en proporción a los datos de entrada.

  Computacionales que son “eficientemente resolubles” o “tratables”.

  Naturales, incluyen versiones de decisión de programa lineal, cálculo del máximo común divisor, y hallan una correspondencia máxima. Incluyen conectividad (accesibilidad) en grafos no dirigidos

PC (tiempo **P**olinomial Completos) para resolver problemas completos.

NP Para resolver problemas de decisión cuyas soluciones positivas/afirmativas pueden ser verificadas en tiempo polinómico a partir de ser alimentadas con la información apropiada, o en forma equivalente, cuya solución puede ser hallada en tiempo polinómico en una máquina no-determinista..

Ejemplo:

MULTIPLICACIÓN MATRICIAL:

Dadas dos matrices A y B de tamaño n x n, hallar C, el producto de las dos.

Descripcion	Algoritmo
Un primer algoritmo se saca de la definición de producto de matrices: Ci,j es el producto escalar de la i-ésima fila por la j-ésima columna. Su tiempo de ejecución sería **O(n3)**:	**int i,j,k;** **int A[n][n],B[n][n],C[n][n];** **// Dar valores a A y B.** **for(i=0;i<n;i++)** **{** **for(j=0;j<n;j++)** **{** **for(k=0;k<n;k++)** **C[i][j]+=A[i][k]*B[k][j];** **}** **}**
Este tiempo de ejecución también se puede mejorar. Si n es una potencia de 2, se pueden dividir A, B y C en cuatro submatrices cada una, de la forma:	$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$
Para hallar las submatrices de C, utilizar:	**C(1,1) = A(1,1)·B(1,1) + A(1,2)·B(2,1)** **C(1,2) = A(1,1)·B(1,2) + A(1,2)·B(2,2)** **C(2,1) = A(2,1)·B(1,1) + A(2,2)·B(2,1)** **C(2,2) = A(2,1)·B(1,2) + A(2,2)·B(2,2)**
El tiempo de ejecución: T(n) = 8·T(n/2) + O(n2), ó O(n3), así no difiere entre este método y el primero propuesto. Para reducir el número de multiplicaciones por debajo de 8 utilizar el algoritmo de Strassen. Definir las siguientes matrices:	**M1 = (A1,2 - A2,2)·(B2,1 + B2,2)** **M2 = (A1,1 + A2,2)·(B1,1 + B2,2)** **M3 = (A1,1 - A2,1)·(B1,1 + B2,1)** **M4 = (A1,1 + A1,2)·B2,2** **M5 = A1,1·(B1,2 - B2,2)** **M6 = A2,2·(B2,1 - B1,1)** **M7 = (A2,1 + A2,2)·B1,1**
Se necesitan 7 multiplicaciones diferentes, y se calculan las submatrices de C utilizando sólo sumas:	**C1,1 = M1 + M2 - M4 + M6 C1,2 = M4 + M5 C2,1 = M6 + M7 C2,2 = M2 - M3 + M5 - M7**
El nuevo tiempo de ejecución:	**T(n) = 7·T(n/2) + O(n2)** **T(n) = O(nlog27) = O(n2.81).**

PROBLEMA DEL CASTOR AFANOSO

Fuente: campusvirtual.unex.es

La existencia de funciones no computables, implica demostrar que hay funciones no Turing computables: hay más funciones que máquinas de Turing para computarlas.
En 1962, Tibor Rado propuso una función basada en lo que hoy se conoce como "**El Problema del Castor Afanoso**", descrita como:

- Suponer una MT con una cinta de doble recorrido infinito a la izquierda y derecha y Alfabeto de cinta={ blanco,1}.
- ¿**Cuál es el número máximo de 1's que pueden ser escritos por una MT de N estados (donde N no incluye el estado final) que termina en parada, y que comienza con una cinta inicialmente en blanco**?
- Este número, varía en función del número de estados de la MT, se denota ∑(N).
- Una máquina que produce ∑(N) celdas no en blanco se denomina Castor Afanoso.
- El problema al estudiar ∑(N) es que crece más deprisa que cualquier función computable, es decir, ∑(N) es no computable.
- Algunos de los valores de ∑(N) y sus correspondientes máquinas de Turing son conocidos hoy, para valores pequeños de N.

Ejemplo: Se sabe que ∑(1)=1, ∑ (2)=4, ∑ (3)=6,□∑ (4)=13. A medida que el número de estados aumenta, el problema se va volviendo más complicado, y, para N≥5, sólo tenemos un conjunto de candidatos que establecen límites inferiores a los valores de ∑(N). Esto se debe en parte al hecho de que no hay ni una teoría general ni una particular sobre la estructura que debe tener un Castor Afanoso.

Originalmente, el problema se definía para MT en forma de quíntupla, donde, las máquinas, dado un estado actual y un símbolo que está siendo buscado en la cinta, escriben un símbolo sobre él, pasa a un nuevo estado y mueve el cabezal de l/e hacia la izquierda o hacia la derecha. Una de las principales variantes consiste en considerar MT en forma de tetratupla.
La diferencia es que, durante la transición a un nuevo estado, una MT o escribe un nuevo símbolo en la cinta o mueve la cabeza de l/e, pero la ejecución de ambas acciones simultaneamente está prohibida.

Definición Formal del Problema: Un MT determinísta es una
SEXTUPLA: (Q, Π, Γ, δ, s, f), donde: Q es un conjunto finito de estados, Π es un alfabeto de símbolos de entrada, Γ es un alfabeto de símbolos de la cinta, δ es la función de transición, s ∈ Q es el estado inicial y f ∈ Q es el estado final.
Si L/R denotan un movimiento del cabezal del/e hacia la Left/Right, La función de transición puede tener el formato:
QUINTUPLA: δ: Q x Γ □ Q x Γ x { L, R }
Escribe un símbolo en la cinta, moverá el cabezal de l/e hacia la izquierda o hacia la derecha y entrará en un nuevo estado.
TETRATUPLA: δ: Q x Γ □ Q x {Γ U { L, R }}
Escribe un nuevo símbolo en la cinta o mueve su cabezal de l/e, antes de entrar en un nuevo estado.

La definición original del Castor Afanoso considera una MT:

- En formato de quíntupla con N+1 estados (N estados más un estado final de parada).
- El alfabeto de la cinta tiene dos símbolos, Γ ∈ { blanco, 1 },
- El alfabeto de entrada tiene sólo uno, Π = { 1 }.
- La productividad de una MT se define como el número de 1's que presenta el resultado, a partir de una cinta en blanco, cuando la MT se para.

Las máquinas que no se paran tienen una productividad de cero. ∑(N) se define como la máxima productividad que se puede obtener a partir de una MT de N estados. Esta MT se denomina Castor Afanoso.
En la variante de la tetratupla, la productividad se define normalmente como la longitud de la secuencia de 1's producida por una máquina de Turing a partir de una cinta en blanco, que para cuando encuentra el 1 más a la izquierda en la secuencia y el resto de la cinta está en blanco.

Las máquinas que no paran, o que paran en una configuración distinta, es decir, en otro 1 que no sea el que se encuentra más a la izquierda, tienen una productividad de 0. De esta manera, la máquina debe parar al leer un 1, este 1 debe ser el que se encuentre más a la izquierda en una cadena de 1's y, con excepción de esta cadena, el resto de la cinta permanece en blanco.

Ejemplo: Castor afanoso de 1 estado y 2 símbolos. Autor: Tibor Rado. ∑(1) = 1 S(1) = 1

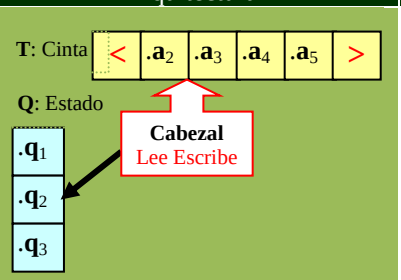
Estado	Entrada 0			Entrada 1		
	Imprime	Movimiento	Nuevo Estado	Imprime	Movimiento	Nuevo Estado
A	1		Parada	1		Parada

AUTOMATA LINEALMENTE ACOTADO: ALA

O Autómata Linealmente Limitado: ALL

Es una versión limitada de la máquina de Turing no determinista, que resuelve el problema de aceptación de lenguajes tipo 1 o L_{DC}. Se caracteriza por su alfabeto de cinta T, provisto de los **símbolos no reemplazables** < y > usados para delimitar los desplazamientos del cabezal, sobre la **cinta de longitud limitada**; así, esta máquina solo se moverá dentro de tal ámbito y puede operar dentro del espacio ocupado por las celdas de la cadena de entrada.

Arquitectura



Caracteristicas

- **Cabezal lectura/escritura**: Movible en ambas direcciones izquierda o derecha y por cada movimiento lee o escribe un símbolo. En el estado inicial se ubica sobre la cota < y luego solo se puede desplazar a la derecha, ya la llegar sobre la otra cota >, solo se puede desplazar a la izquierda.
- **Memoria**: Soporte de los estados de la máqina.
- **Cinta**: Soporte de entrada o salida, con una **colección finita** de celdas, capaces de almacenar cada una, un único símbolo.
  Ambos lados de la cadena registrada en la cinta posee un símbolo < > o cota.

Definición

Es una Tupla:

M= (Q, Σ, T, s, β, F, δ)

Componente

Q Conjunto finito de estados.
Σ Alfabeto de Entrada
T Alfabeto de la Cinta e incluye las cotas < >
.s ∈ Q Estado Inicial
β ∈ T ó **Δ:** Símbolo Blanco (**β** ∉ Σ, ó Σ ⊆ T – { **β** })
F ⊆ Q Conjunto de estados finales de aceptación. Incluye **Q**
δ Función Parcial o Transición. Matemáticamente define el ALA como:

- **DETERMINISTA ALA_D**: Tiene una sola alternativa de transición:
  (q, σ) → {(p, t, X)} donde:
  Par **(q, σ)**: Parámetros de la posición actual del cabezal
  - **q ∈ Q: Estado Actual**, que actua como estímulo del ALA
  - **σ ∈ T: Símbolo de cinta** que señala el cabezal o reacción del ALA

  Terna **(p, t, X)**:
  - **p ∈ Q:** Estado siguiente. (Ej: $.q_1$, $.q_2$, $.q_3$,..q_n...)
  - **t ∈ T:** Símbolo de cinta que reemplaza al que existía en la celda.
  - **X∈ { I, D, P }:** Movimiento del cabezal (**I**zquierda, **D**erecha, **P**arado)
- **NO DETERMINISTA ALA_{ND}**: Tiene mas de una alternativa de transición:
  (q,σ) →{(p_1,t_1, X_1), (p_2, t_2, X_2), . . (p_n, t_n, X_n) }.

Acción:

(q, σ) → {(p, t, X)}

- **Acepta:** cuando llega alguno de los estados finales.
- **Rechaza:** cuando, para un símbolo leído, llega a un estado de transición sin definir, que bloquea el ALA

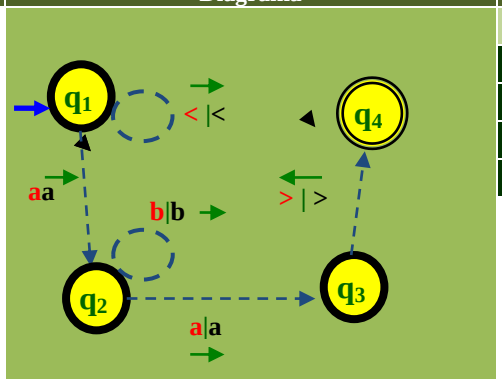
Ejemplo: Diseñar un ALA, que acepte la cadena **aba**:

$M_{ALA} = (Q, \Sigma, T, s, \beta, F, \delta) = (\{q_1,q_2,q_3,q_4\}, \{a, b\}, \{<, a, b, >\}, q_1, \{q_4\}, \beta, \delta)$,

δ: funciones de transición:

Pares	Diagrama	Matriz

$(q_1, <) \rightarrow (q_1 < D)$
$(q_1, a) \rightarrow (q_2\ a\ D)$
$(q_2, b) \rightarrow (q_2\ b\ D)$
$(q_2, a) \rightarrow (q_3\ a\ D)$
$(q_3, >) \rightarrow (q_4 > I)$



δ	<	a	b	>
q_1	$(q_1 < D)$	$(q_2 a D)$		
q_2		$(q_3 a D)$	$(q_2 b D)$	
q_3				$(q_4 > I)$
q_4				

El proceso de la cadena **ab** es:

Estado inicial — Movimiento 1 — Movimiento 2 — Movimiento 3 — Movimiento 3 — Estado Final

$.\underline{<}\ a\ b\ a\ >\quad <\ \underline{a}\ b\ a\ >\quad <\ a\ \underline{b}\ a\ >\quad <\ a\ b\ \underline{a}\ >\quad <\ a\ b\ a\ \underline{>}\quad <\ a\ b\ \underline{a}\ >$

$\Delta \qquad \Delta \qquad \Delta \qquad \Delta \qquad \Delta \qquad \Delta$

$.f(q_1,<)=(q_1,<,D)\quad f(q_1,a)=(q_2,a,D)\quad f(q_2,b)=(q_2,b,D)\quad f(q_2,a)=(q_3,a,D)\quad f(q_3,>)=(q_4,>,I)\quad f(q_4,a)=\text{Stop}$

Descripcion instantanea:
Representa cada instante de las sucesivas configuraciones por las que pasa el autómata acotado, bajo el formato de:

- **Par de valores**:
  $(q,w),\ q \in Q,\ w \in \Sigma^*$, donde: **q**: Estado Actual, **w**: Entrada No Leída o Restante; ├: Símbolo entre descripciones.

  Estado inicial — Movimiento 1 — Movimiento 2 — Movimiento 3 — Movimiento 4 — Estado final

  $(q_1, \underline{<}\ aba>) \vdash (q_1,\ <\underline{a}ba>) \vdash (q_2,\ <a\underline{b}a>) \vdash (q_2,\ <ab\underline{a}>) \vdash (q_3,\ <aba\underline{>}) \vdash (q_4,\ <ab\underline{a}>)$

  Todo este proceso se puede representar como:
  $(q_1, \underline{<}\ aba>) \vdash^* (q_4,\ <ab\underline{a}>)$

  Parcialmente este proceso se puede representar como:
  $(q_1, \underline{<}\ aba>) \vdash^+ (q_2,\ <ab\underline{a}>)$

- **Cadena Única**: **a q b**, donde $q \in Q,\ a,b \in \Sigma^*$, donde: **q**: Estado Actual, **a, b, a**: Cadena Leída; ├: Símbolo entre dos descripciones.

  Estado inicial — Movimiento 1 — Movimiento 2 — Movimiento 3 — Movimiento 4 — Estado final

  $q_1 < aba> \vdash < q_1\ aba> \vdash < a\ q_2 ba> \vdash < ab\ q_2\ a> \vdash < aba\ q_3 > \vdash < aba\ q_4 >$

  Todo este proceso se puede representar como:
  $q_1 < aba> \vdash^* < aba\ q_4 >$

  Parcialmente este proceso se puede representar como:
  $q_1 < aba> \vdash^+ < ab\ q_2\ a>$

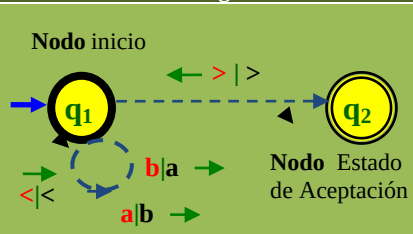
Ejemplo: Transformar cadenas de **a**es en **b**es:

$$M_{ALA} = (Q,\ \Sigma, T, s, \beta, F, \delta) = (\{q_1, q_2\}, \{ a, b\}, \{ <, a, b, >\},\ q_1, \{ q_2 \}, \delta)$$

δ: funciones de transición:

Pares	Diagrama	Matriz

$(q_1,<)\rightarrow(q_1< D)$
$(q_1,a)\rightarrow(q_1 b\ D)$
$(q_1,b)\rightarrow(q_1 a\ D)$
$(q_1,>)\rightarrow(q_2> I)$



δ	<	a	b	>
q_1	$(q_1< D)$	$(q_1 b\ D)$	$(q_1 aD)$	$(q_2>I)$
q_2				

El proceso de la cadena **ab** es:

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Estado Final
.< a b >	< a b >	< b b >	< b a >	< b a >
Δ	Δ	Δ	Δ	Δ
.$f(q_1,<)=(q_1,<,D)$	$f(q_1,a)=(q_1,b,D)$	$f(q_1,b)=(q_1,a,D)$	$f(q_1,>)=(q_2,>,I)$	$f(q_2,a)=(\dots)$

Descripcion instantanea:
Representa cada instante de las sucesivas configuraciones por las que pasa el autómata acotado, bajo el formato de:

- **Par de valores**:
  (q,w), q∈Q, w∈Σ, donde: **q**: Estado Actual, **w**: Entrada No Leída o Restante; ├: Símbolo entre descripciones.

  Estado inicial Movimiento 1 Movimiento 2 Movimiento 3 Estado final
  (q1, < ab>) ├ (q1, < a b>) ├ (q_1,< b b >) ├ (q_1,<ba >) ├ (q_2,< b a >)

  Todo este proceso se puede representar como: **(q1, < ab>) ├* (q_2,< b a >)**

  Parcialmente, este proceso se puede representar como: **(q1, < ab>) ├ + (q_1,< b b >)**

- **Cadena Única**:
  a q b, donde **q∈Q, a,b∈Σ,** donde: **q**: Estado Actual, **a, b**: Cadena Leída; ├: Símbolo entre dos descripciones.

  Estado inicial Movimiento 1 Movimiento 2 Movimiento 3 Estado final
  q_1 < ab> ├ < q_1 a b > ├ <b q_1 b > ├ < ba q_1 > ├ < b q_2 a >

  Todo este proceso se puede representar como: **q_1 < ab> ├* < b q_2 a >**

  Parcialmente, este proceso se puede representar como: **q_1 < ab> ├ + <b q_1 b >**

AUTOMATA de/con PILA AP

El **AP**, por naturaleza es no determinista AP_{NDE}, es un modelo matemático capaz de reconocer cadenas formadas por $a_i \in \Sigma$ símbolos de alfabetos del lenguaje L_{IC} generada por una G_{IC}:

GRAMATICA INDEPENDIENTE o libre de CONTEXTO G_{IC}

Definición:

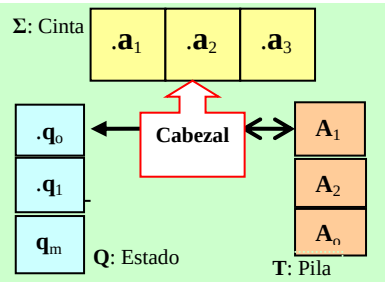
Gramatica estructurada por frases: $G_2 = G_{IC} = (\Sigma_T, \Sigma_N, S, P)$ donde:

- Σ_T: Alfabeto de **símbolos terminales**.
- Σ_N: Alfabeto de **símbolos no terminales**, donde $\Sigma = \Sigma_T \cup \Sigma_N$ y $\Sigma_T \cap \Sigma_N = 0$
- **S**: Símbolo Inicial o **Axioma** de la gramática, es un símbolo no terminal: $S \in \Sigma_N$
- **P**: **Producciones**: $P = \{ (S ::= \lambda) \text{ ó } (A ::= v) \mid A \in \Sigma_N, v \in \Sigma_T + \}$
  - La **longitud de la izquierda debe ser 1 y puede tener un solo símbolo no terminal**.
  - A la derecha admite pares del producto Σ_N x $\Sigma^* (\Sigma_N \cup \varepsilon)$, así puede tener cualquier numero de símbolos terminales o no terminales.
  - La regla de producción es de forma: $A \rightarrow v$, donde: $(A = v \mid A \in \Sigma_N, v \in \Sigma^*)$
    - **A** : Símbolo no terminal
    - **v** : Cadena de terminales y/o no terminales: **A** puede sustituirse por **v** sin considera el contexto

Ejemplo: La G_{IC}: $G_2 = (\Sigma_T, \Sigma_N, S, P) = (\{0,1\}, \{A, B\}, A, P)$ con:

Producciones		Derivaciones		
Pares	Simplificado	A → 11	A →1B1 → 101	A →1B1→ 111
P={(A::=1B1), (A::= 11), (B::= 1), (B ::= 0) }	A ::= 1B1 | 11 B ::= 1 | 0	A / \\ 1 1	A / | \\ 1 B 1 | 0	A / | \\ 1 B 1 | 1

Lenguaje generado: $L_{IC} = \{ w^n \mid w = w^{-1} \}$ = { 11, 101, 111 , . . }

Arquitectura del **AP**	Componentes
	▪ **CINTA SOPORTE:** Registra los símbolos de entrada: $a_i \in \Sigma$ ▪ **CONTROL:** Posee un cabezal cuyo movimiento depende del: ○ **Símbolo de entrada** $(a \in \Sigma)$ que lee, inclusive λ, que permite el $\lambda_{Moviento}$ que para el avance del cabezal ○ **Simbolo de cima de pila** $A_i \in T$, inclusive λ, que permite el borrado del símbolo de la cima. ○ **Conjunto de estados** $\{q \in Q\}$. ▪ **PILA: Memoria auxiliar**: Sus símbolos se insertan o extraen, en modo LIFO: LastInFirstOut

G_{IC} AUTOMATA de PILA DEFINICIÓN: $AP=(\Sigma,T, Q, A_o, q_o, F, \delta)$

Componente
Σ: Alfabeto de símbolos de entrada
T: Alfabeto de símbolos de pila.
Q: Conjunto de estados. (un estado, es estado inicial)
A_o: ∈ T Símbolo inicial de pila
q_o: ∈ Q Estado inicial
F: ⊆ Q Conjunto de estados finales
δ: Función Transición: $\delta(q_i, \sigma, \gamma_i) \rightarrow \{(q_{i+1}, \gamma_{i+1})\}$ δ (q_i Estado actual, σ Símbolo entrada, γ_i Símbolo de Pila) → {(q_{i+1} Estado siguiente, γ_{i+1} Nuevo símbolo cima pila)}

AP determinista: AP_{DE}	AP No determinista: AP_{NDE}
La función de transición **δ** es de forma: $(q_i, \sigma, \gamma_i) \rightarrow \{(q_{i+1}, \gamma_{i+1})\}$ ○ Transición algún símbolo de entrada. ○ Una sola transición para un mismo estímulo de estado **q** y símbolo de pila **Z**, incluyendo las transiciones **λ**. Luego: ▪ $\forall q \in Q,\ Z \in T,$ si $\lvert f(q, \lambda, Z) \rvert > 0$, entonces: $\forall a \in \Sigma,\ f(q, a, Z) = 0$ ▪ $\forall q \in Q,\ Z \in T,\ a \in \Sigma\{\lambda\},\ \lvert f(q, a, Z) \rvert < 2$ $\Delta \subseteq Q \times (\Sigma U\{\lambda\}) \times T \times Q \times T^*$	La función de transición **Δ** es de forma: $(q,a,s) \rightarrow \{(q_1, \gamma_1), (q_2, \gamma_2), \ldots, (q_n, \gamma_n)\}$ Donde: $\gamma i \in \Gamma^*$ $\Delta: Q \times (\Sigma U\{\lambda\}) \times \Gamma \rightarrow Pf(Q \times \Gamma^*)$ **Pf** ($Q \times \Gamma^*$): Conjunto de subconjuntos finitos de $Q \times \Gamma^*$ Para $q \in Q$, $a \in \Sigma U\{\lambda\}$ y $s \in \Gamma$

Funcionamiento:
Si el cabezal lee ***a***, estado ***s***, y cima de pila ***Z***, ocurre: $\delta(s, a, Z)=\{(q_1,\gamma_1), (q_2,\gamma_2),.. (q_n,\gamma_n)\}$
con: $s, q_i \in \Sigma Q,\ a \in (\Sigma U\{\lambda\}),\ \gamma_i \in T$

- Si $a \in \Sigma$, no es palabra vacía, el cabezal avanza una posición y lee el siguiente símbolo.
- Elimina el símbolo **Z** de la pila.
- Selecciona un par (q_i,γ_i) en la definición de $\delta(s, a, Z)$, función de transición.
- Apila la cadena: $\gamma_i = A_1 A_2 \ldots A_k$ en la pila, en cuya cima queda el símbolo A_1.
- Se cambia el control del autómata al estado q_i.

AP_{DE} DESCRIPCIÓN INSTATANEA

Concepto	Contenido
Muestra las sucesivas configuraciones por las que pasa el AP, en cada instante, mediante la terna: (q, w, z), donde: $q \in Q,\ w \in \Sigma^*,\ z \in T^*$	• **q**: Estado Actual, • **w**: Entrada No Leída o Restante • **z**: Contenido de la Pila; • ├:Símbolo entre descripciones

Ejemplo: $(q_0, aabb, Z) \vdash (q_0, abb, AZ) \vdash (q_0, bb, AAZ) \vdash (q_1, b, AZ) \vdash (q_1, \varepsilon, Z) \vdash (q_2, \varepsilon, \varepsilon)$

AP_{DE} DISEÑO:

A partir de una Gramática Independiente de Contexto:

$G_{IC} = (\Sigma_T, \Sigma_N, S, P) = (\{ a, b\}, \{ S_1 \}, S, P) = (\{ a, b \}, \{ S_1 \}, S, P)$, con las siguientes:

Producciones		Derivaciones
Pares	Simplificado	
P={(S_1::=a S_1b), (S_1::= λ) }	S_1 ::= a S_1 b | λ	S_1→ λ S_1→ a S_1b → aλb = $a^1 b^1$ S_1→ a S_1b → aabb = $a^2 b^2$ S_1→ a S_1b → aaS_1bb → aaabb b = $a^3 b^3$ S_1→ a S_1b → aaS_1bb → aaa S_1bbb → aaaabbbb = $a^4 b^4$

Que genera el lenguaje: L_{IC}={ $a^n b^n \mid n \geq 0$ }={ λ, $a^1 b^1$, $a^2 b^2$, $a^3 b^3$, $a^4 b^4$, . }= { λ, ab, aabb, aaabbb, . }

Diseñar un el autómata con pila AP_D, capaz de reconocer este lenguaje: L_{IC}= { $a^n b^n \mid n \geq 0$ } :

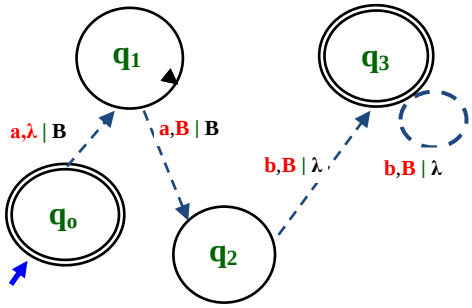
$$AP_D = (\Sigma, T, Q, A_o, q_o, \delta, F) = (\{a, b\}, \{A, B\}, \{q_o, q_1, q_2, q_3\}, \lambda, q_o, \delta, \{ q_0, q_3\})$$

Con las siguientes transiciones, expresados en formato de:

Pares

$\delta(q_o,a,\lambda)=\{(q_1,B)\}$
$\delta(q_1,a,B)=\{(q_2,B)\}$
$\delta(q_2,b,B)=\{(q_3,\lambda)\}$
$\delta(q_3,b,B)=\{(q_3,\lambda)\}$

Diagrama



Matriz

Δ	(a, λ)	(b,B)	(b,B)
→*q_o	{(q_1,B)}		
q_1		{(q_2,B)}	
q_2		{(q_3,λ)}	
*q_3			{(q_3,λ)}

Proceso de la cadena: **a a b b**

Descripción instantanea

Estado inicial — Movimiento 1 — Movimiento 2 — Movimiento 3 — Estado Final

$(q_0, aabb, \lambda) \vdash (q_1, abb, B) \vdash (q_2, bb, BB) \vdash (q_3, b, B) \vdash (q_3, \lambda, \lambda)$

La pila queda vacia y lleva hasta el estado final de aceptación q_3

Descripción Instantánea Total

Estado inicial — Estado Final

$(q_0, aabb, \lambda) \vdash^* (q_3, \lambda, \lambda)$

Descripción Instantánea Parcial

Estado inicial — Movimiento 1 — Movimiento 2

$(q_0, aabb, \lambda) \vdash (q_1, abb, B) \vdash + (q_2, bb, BB)$

AP_{DE} LENGUAJE ACEPTADO

Puede definirse por las siguientes formas equivalentes:

a) **ESTADO FINAL**:

El lenguaje aceptado es el conjunto de entradas que hacen que el **AP** llegue a un estado final

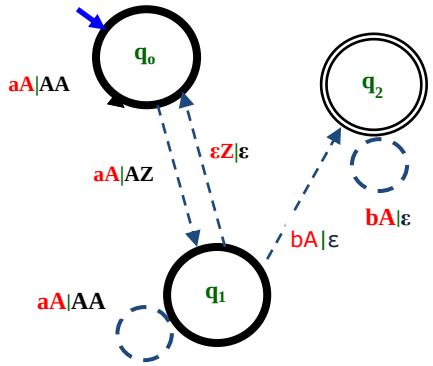
Teorema: El lenguaje **L(M)** aceptado por el **AP = (Σ, T, Q, A_o , q_o , Δ , F)**, se define por el conjunto

$$L(M) = \{ w \in \Sigma^* \mid (s, w, z) \vdash^* (p, \lambda, u) \text{ para } p \in F \text{ y } u \in T^* \}$$

Ejemplo: **AUTOMATA DE PILA CON ESTADOS FINALES**

El $AP_D = (\Sigma, T, Q, A_o, q_o, \delta, F) = (\{a, b\}, \{A, Z\}, \{q_0, q_1, q_2\}, A, q_0, \delta, \{q_2\})$,
procesa: $L_{IC} = \{a^i b^i \mid i \geq 0\}$, para transiciones

$\Delta(q_0,a,Z) = (q_1,AZ)$
$\Delta(q_1,a,A) = (q_1,AA)$
$\Delta(q_1,b,A) = (q_2,\varepsilon)$
$\Delta(q_2,b,A) = (q_2,\varepsilon)$
$\Delta(q_1,\varepsilon,Z) = (q_0,\varepsilon)$



Δ	(a,Z)	(ε,Z)	(a,A)	(b,A)
→q_0	(q_1,AZ)	(q_0,ε)		
q_1			(q_1,AA)	(q_2,ε)
*q_2				(q_2,ε)

Proceso de la cadena: **aabb**

	Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Estado Final
Descripción instantanea	$(q_0, aabb, Z)$	$\vdash (q_1, abb, AZ)$	$\vdash (q_1, bb, AAZ)$	$\vdash (q_2, b, AZ)$	$\vdash (q_2, \varepsilon, Z)$

Lleva hasta el estado final de aceptación q_2

b) **VACIADO DE PILA**:

El lenguaje está formado por el conjunto de entradas que vacian la pila.

El lenguaje será aceptado por el autómata, si este permite procesar a todas sus palabras de manera que al final de cada proceso, la pila siempre quede vacía.

Tienen el formato $L(M) = \{ x \mid (q_O, x, A_O) \vdash^* (q, \lambda, \lambda) \quad q \in Q, \ x \in \Sigma^* \}$

Puede afirmarse que dado un L_{IC} siempre se puede encontrar un **AP** con estados finales y un **AP** con pila vacia que reconozca al L_{IC}.

Ejemplo:

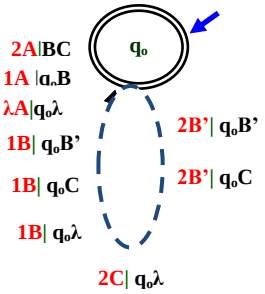
El $AP_{ND} = (\Sigma, T, Q, A_o, q_o, \delta, F) = (\{0,1,2\}, \{A,B,B',C\}, \{q_o\}, A, q_o, \Delta, \emptyset)$, para las transiciones:

Pares	Diagrama	Matriz

$\Delta(q_o,2,A) = \{(q_o,BC)\}$
$\Delta(q_o,1,A) = \{(q_o,B)\}$
$\Delta(q_o,\lambda,A) = \{(q_o,\lambda)\}$
$\Delta(q_o,1,B) = \{(q_o,B')(q_o,C)(q_o,\lambda)\}$

$\Delta(q_o,2,B') = \{(q_o,B')(q_o,C)\}$

$\Delta(q_o,2,C) = \{(q_o,\lambda)\}$



Δ	(2,A)	(1,A)	(λ,A)	(1,B)	(2,B')	(2,C)
→*qo	{(qo,BC)}	{(qo,B)}	{(qo,λ)}	{(qo,B')(qo,C)(qo,λ)}	{(qo,B')(qo,C)}	{(qo,λ)}

Proceso de la cadena **112**

Estado inicial	Movimiento 1	Movimiento 2	Estado final
1 1 2	1 **1** 2	1 1 **2**	1 1 2 **λ**
q_o |(q_o,1,A)={(q_o,B)|**A**	q_o |(q_o,1,B)={(q_o,C)|**B**	q_o |(q_o,2,C)={(q_o, λ) |**C**	q_o |(q_o, λ, λ)={(..)| λ

Descripción instantánea

$$(q_O, 112, A) \vdash (q_O, 12, B) \vdash (q_O, 2, C) \vdash (q_O, \lambda, \lambda)$$

Formato

$$L(M) = \{ (q_O, 112, A) \vdash^* (q_O, \lambda, \lambda) \}$$

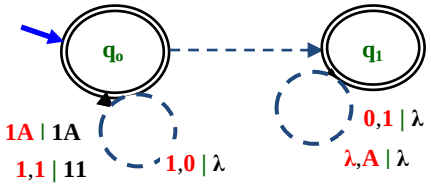
AP_{DE} AUTÓMATA DE PILA DETERMINISTA:

Ejemplo: procesar la cadena: **1 1 0 0**, con

El $AP_{DE} = (\Sigma, T, Q, A_o, q_o, \delta, F) = (\{0,1\}, \{A,1,0\}, \{q_o,q_1\}, A, q_o, \delta, \emptyset)$

Pares	Diagrama	Matriz

$\delta(q_o,1,A) = \{(q_o,1A)\}$
$\delta(q_o,1,1) = \{(q_o,11)\}$
$\delta(q_o,0,1) = \{(q_o,\lambda)\}$
$\delta(q_1,0,1) = \{(q_1,\lambda)\}$
$\delta(q_1,\lambda,A) = \{(q_1,\lambda)\}$



Δ	(1,A)	(1,1)	(0, 1)	(λ, A)
→*q_o	{(q_o,1A)}	{(q_o,11)}	{(q_o,λ)}	
*q_1			{(q_1,λ)}	{(q_1,λ)}

Proceso de cadena **1100**

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Estado final
1 1 0 0	1 **1** 0 0	1 1 **0** 0	1 1 0 **0**	1 1 0 0 **λ**
(q_o,1,A)={(q_o,1A)}	(q_o,1,1)={(q_o,11)}	(q_o,0,1)={(q_o,λ)}	(q_o,0,1)={(q_o,λ)}	(q_o,λ,A)= ?

Descripción instantánea

$$(q_O, 1100, A) \vdash (q_O, 100, 1A) \vdash (q_O, 00, 11A) \vdash (q_O, 0, 1A) \vdash (q_O, \lambda, A)$$

$\vdash^*$ Proceso total: $(q_O, 1100, A) \vdash^* (q_O, \lambda, A)$

$\vdash^+$ Proceso parcial: $(q_O, 1100, A) \vdash^+ (q_O, 00, 11A)$

Como **f(q_O, λ, A):** sin definir, la cima de la pila λ queda vacía, se verifica que 1100 pertenece al lenguaje aceptado por el AP

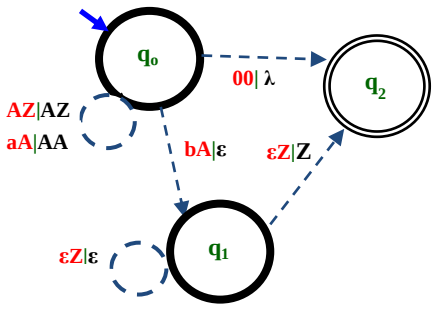
AP_{NDE} AUTÓMATA DE PILA NO DETERMINISTA:

AP_{ND}: Dado un estado **q** y un símbolo de pila **A** tiene alguna transición **λ**, luego puede haber más de una transición para un mismo estado **q** y símbolo de pila **A,** incluyendo transiciones **λ**; asi, para el paso del **AP_{ND}** de un estado a otro y al nuevo símbolo de cima de pila, define la regla de transición:

$$Qx(\Sigma U\{\lambda\})xT \rightarrow P(QxT^*), \text{ donde: } \Delta \subseteq Qx(\Sigma U\{\lambda\})xTxQxT^*$$

Ejemplo:

El $AP_{ND} = (\Sigma,T,Q,A_o,q_o,\delta, F)=(\{a, b\},\{A,Z\},\{q_0,q_1,q_2\},A,q_0,\delta,\{q_2\})$, procesa: $L_{IC} = \{a^i b^i \mid i \geq 0\}$, para transiciones

Pares	Diagrama	Matriz
$\Delta(q_0,a,Z) = (q_0,AZ)$ $\Delta(q_0,\varepsilon,Z) = (q_2,Z)$ (acepta ε) $\Delta(q_0,a,A) = (q_0,AA)$ $\Delta(q_0,b,A) = (q_1,\varepsilon)$ $\Delta(q_1,b,A) = (q_1,\varepsilon)$ $\Delta(q_1,\varepsilon,Z) = (q_2,Z)$		Es AP_{ND}: Por presencia simultánea de: $\Delta(q_0,a,Z)$ y $\Delta(q_0,\varepsilon,Z)$

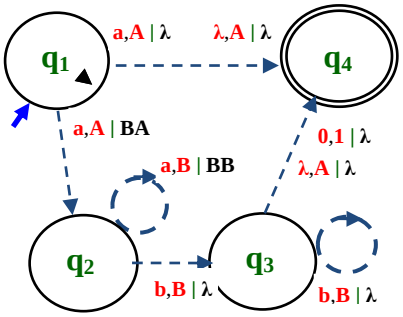
Δ	(a,Z)	(ε,Z)	(a,A)	(b,A)	
→q_0	(q_0,AZ)	(q_2,Z)	(q_0,AA)	(q_1,ε)	
q_1		(q_2,Z)		(q_1,ε)	
*q_2					

Proceso de la cadena: **aabb**

Alternativa	Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado Final
1	$(q_0, aabb, Z)$	$\vdash (q_0, abb, AZ)$	$\vdash (q_0, bb, AAZ)$	$\vdash (q_1, b, AZ)$	$\vdash (q_1, \varepsilon, Z)$	$\vdash (q_2, \varepsilon, \varepsilon)$
					Llevan hasta el estado de aceptación q_2	
2	$(q_0, aabb, A)$	$\vdash (q_2, \varepsilon, \varepsilon)$				

Ejemplo:

El $AP_{NDE} = (\Sigma,T,Q,A_o,q_o,\delta, F)=(\{a, b\},\{A,B\},\{q_1,q_2,q_3,q_4\},A,q_1,\delta,\{q_4\})$, procesa: $L_{IC} = \{a^n b^n \mid n>0\}$, para:

Pares	Diagrama	Matriz
$\delta(q_1,a,A)=\{(q_2,BA),(q_4,\lambda)\}$ $\delta(q_1,\lambda,A)= \{(q_4,\lambda)\}$ $\delta(q_2,a,B) = \{(q_2,BB)\}$ $\delta(q_2,b,B) = \{(q_3,\lambda)\}$ $\delta(q_3,b,B) = \{(q_3,\lambda)\}$ $\delta(q_3,\lambda,A)= \{(q_4,A)\}$		

Δ	(a,A)	(λ, A)	(a, B)	(b, B)
→q_1	$\{(q_2,BA),(q_4,\lambda)\}$	$\{(q_4,\lambda)\}$		
q_2			$\{(q_2,BB)\}$	$\{(q_3,\lambda\}$
q_3		$\{(q_4,A)\}$		$\{(q_3,\lambda\}$
*q_4				

Proceso de la cadena: **aabb**

Alternativa	Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado Final
1	$(q_1, aabb, A)$	$\vdash (q_2, abb, BA)$	$\vdash (q_2, bb, BBA)$	$\vdash (q_3, b, BA)$	$\vdash (q_3, \lambda, A)$	$\vdash (q_4, \lambda, A)$
					Llevan hasta el estado de aceptación q_4	
2	$(q_1, aabb, A)$	$\vdash (q_4, abb, \lambda)$				

AP_{DE} AS: ANALIZADOR SINTÁCTICO (PARSER):

Un compilador es un software traductor un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina puede interpretar, así, se traduce el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior, de manera que se puede diseñar un programa en lenguaje cercano al humano, para luego compilarlo a un programa procesable por la computadora, mediante la siguiente acción:

Lenguaje Fuente → **COMPILADOR** → **Lenguaje Objeto**

La sintaxis enfoca la forma o estructura de los programas en lenguaje fuente, usando una G_{IC} en notación **BNF** o un diagrama sintáctico. A partir de esta, se construye el analizador sintáctico del compilador que:

- Comprueba la corrección de la sintaxis del programa fuente.
- Construye una representación interna del programa o caso contrario envia un mensaje de error.
- Comprueba si los token llegan en el orden definido, para luego generar la salida con formato de árbol sintáctico, según muestra el siguiente esquema:

Así, el analizador sintáctico:

- Acepta solo lo que es válido sintácticamente.
- Explicita el orden jerárquico que tienen los operadores en el lenguaje de que se trate.
- Guia el proceso de traducción dirigida por la sintaxis.

A partir del **árbol sintáctico** que representa la sintaxis de un programa, el AS construye una derivación con recorridos por la izquierda o por la derecha del programa fuente, y partir de ellos construye una representación intermedia de ese programa fuente, que puede ser un **árbol sintáctico abstracto** o bien un programa en un **lenguaje intermedio**. Así para la sentencia T = X – Y + Z

Árbol Sintáctico	Árbol Sintáctico Abstracto	Lenguaje Intermedio
= / \ T + / \ - Z / \ X Y	| ASIGNAR | T | o |	restar X Y t_1
	| SUMAR | o | Z |	sumar t_1 Z t_2
	| RESTAR | X | Y |	asignar t_2 T

TIPOS DE ANÁLISIS SINTÁCTICO:

AP_{DE} RECONOCIMIENTO ASCENDENTE

Desde la cadena de entrada se construye el árbol empezando por las hojas, luego se crean nodos intermedios hasta llegar a la raíz; construyendo así al árbol de abajo hacia arriba. El recorrido se hará desde las hojas a la raíz.

El reconocedor o análisis sintáctico ascendente, construye un árbol sintáctico ascendente que parte de las hojas hasta la raíz. Por ejemplo el reconocedor **LR(1)** que es el más usado para definir los lenguajes de programación, es un algoritmo que recibe como entrada una cadena: $w \in \Sigma_T^+$:

- **LR**: Examina sus símbolos de izquierda a derecha
- **R**: Indica que en cada paso se efectua la derivación por derecha.
- **1**: Indica que solo es necesario un carácter para que el reconocedor decida la acción a efectuar.

Su funcionamiento se registra en una tabla de acciones que contiene:

- **D_i**: Desplazamientos:
  Donde **i** es el estado siguiente del desplazamiento **D**.

- **R_i**: Reducciones:
  Donde **i** identifica la producción utilizada en la reducción, que es, la acción efectuada cuando en el árbol aparece la parte derecha de una producción y se añade la parte izquierda, subiendo un nivel en el árbol.

CONSTRUCCIÓN DE LA TABLA DE ACCIONES

1. **Crear un nucleo del estado q_o**
   $[S::=A,\$] \in q_o$

2. **Cerrar q_o**
   q_o = {[S::=A,$], [A::=(a),$], [A::=a,$]}
   Analizando los 3 RL ítems y los que luego se construyen:
   Acción(q_o,A)= D_1 Acción(q_o,()= D_3 Acción(q_o,a)= D_2

3. **Crear nuevos estados**
   q_1 = {[S::=A,$] } Acción($q_1$,$)= R_1
   q_2 = {[S::=a,$] } Acción($q_2$,$)= R_2
   q_3 = {[S::=(a),$] } Acción($q_3$,$)= D_4
   q_4 = {[S::= (a),$] } Acción($q_4$,$)= D_5
   q_5 = {[S::= (a),$] } Acción($q_5$,$)= R_3

Estos analizadores poseen una pila que almacena los caracteres ya procesados y los estados por los que pasó el reconocedor.

Ejemplo: Para la gramática: **$G = (\Sigma_T, \Sigma_N, S , P) = = (\{a,(,)\}, \{S,A\}, S , \{S::=A, A::=a, A::=a\})$**

Las acciones: define en la tabla en función de:

- Estado del autómata
- Símbolo en proceso de la cadena de entrada.
- La situación no definida es una situación de error.
- $: Marca el final de la cadena de entrada.

q_i	$	a	(	)	A
q_o		D_2	D_3		D_1
q_1	R_1				
q_2	R_2				
q_3		D_4			
q_4				D_5	
q_5	R_3				

Usando esta tabla la cadena (a) se procesa:

Entrada	Pila	Acción
(a)$	q_o	**Desplazar a q_3**
a)$	q_o (q_3	**Desplazar a q_4**
)$	q_o ($q_3 a q_4$	**Desplazar a q_5**
$	q_o ($q_3 a q_4$)q_5	**Reducir** por la 3º producción **A::=(a)**
$	q_oA	**Desplazar a q_1**
$	q_oAq_1	**Reducir por S::=A** (Fin proceso)

AP_{DE} RECONOCIMIENTO DESCENDENTE

Opera desde la raíz de árbol y va aplicando reglas por la izquierda, de forma de obtener una derivación por izquierda de la cadena de entrada. Recorriendo el árbol en profundidad de izquierda a derecha, encontraremos en las hojas los tokens, que nos devuelve el A.L. en ese mismo orden.

El reconocedor es un algoritmo que recibe como entrada una cadena: $w \in \Sigma_T^+$, examina de izquierda a derecha sus símbolos, luego trata de construir su árbol sintáctico, en cuyo proceso se obtine la estructura de la palabra, las producciones gramaticales que han de aplicarse y el órden en que debe utilizarse.

Un reconocedor o analizador sintáctico descendente, es un método de reconocimiento de palabras de un que construye el árbol de derivación de cada palabra, partiendo desde la raíz hasta las hojas.

AP_{DE} GRAMATICA ASOCIADA A UN AUTÓMATA DE PILA:

Desde un AP podemos construir una G_{IC} en la forma normal de Greinbach **$G = (\Sigma_T, \Sigma_N, S , P)$**, para ello, supongamos que a partir del **$AP = (\Sigma, T, Q, A_o , q_o , \Delta , F)$** donde:

- **$\Sigma_N = \{ S \} \cup \{ (p, A, q) \mid p, q \in Q, \ A \in T \}$**
- **P**: Para cada estado **$q \in Q$** se crea una regla **$S ::= (q_o, A_o, q)$**
- Para cada **$a \in \Sigma \cup \{\lambda\}, p, q \in Q, A, B, B_i \in T$** si **$(q, B.B_1.B_2 \ldots .B_N) \in f(p, a, A)$** entonces para todas las combinaciones de estados **$q \in Q$** se crea una regla
  $(p, A, q_N) ::= a. (p, B, q_1). (p_1, B_1, q_2) \ldots (p_{n-1}, B_n, q_n)$
- Para cada **$a \in \Sigma \cup \{ \lambda \}, A \in T$** si **$(q, \lambda) \in f(p, a, A)$** entonces se crea una regla **$(p, A, q) ::= a$**

Luego, si desde la descripción instantánea inicial **$(q_o, x, A_o), x \in \Sigma^*, A \in T, p, q \in Q$** se cumple que se puede transitar en **n** pasos a **$(q, \lambda, \lambda), (q_o, x, A_o) \vdash^* (q, \lambda, \lambda)$**, significa que el AP por vaciado de pila acepta la cadena **x**, así desde el axioma **S** de la gramática, se deriva **x** en **n** pasos **($S \rightarrow^* x$)**.

Ejemplo: Crear el AP que reconoce la gramatica **$G = (\{0, 1\}, \Sigma_N, S , P)$** donde:

- **P**: $S ::= (q_o, A, q_o) \mid (q_o, A, q_1)$
  $(q_o, A, q_o) ::= 1 (q_o, 1, q_o) (q_o, A, q_o)$
  $(q_o, A, q_o) ::= 1 (q_o, 1, q_1) (q_1, A, q_o)$
  $(q_o, A, q_1) ::= 1 (q_o, 1, q_o) (q_o, A, q_1)$
  $(q_o, A, q_1) ::= 1 (q_o, 1, q_1) (q_1, A, q_1)$
  $(q_o, 1, q_o) ::= 1 (q_o, 1, q_o) (q_o, A, q_o)$
  $(q_o, 1, q_o) ::= 1 (q_o, 1, q_1) (q_1, 1, q_o)$
  $(q_o, 1, q_1) ::= 1 (q_o, 1, q_o) (q_o, 1, q_1)$
  $(q_o, 1, q_1) ::= 1 (q_o, 1, q_1) (q_1, 1, q_1)$
  $(q_o, 1, q_1) ::= 0$
  $(q_1, 1, q_1) ::= 0$
  $(q_1, A, q_1) ::= \lambda$

- **$\Sigma_N = \{S, (q_o, A, q_o), (q_o, 1, q_o), (q_o, A, q_1), (q_o, 1, q_1), (q_1, A, q_o), (q_1, 1, q_o), (q_1, A, q_1), (q_1, 1, q_1)\}$**

La descripción instantánea de este proceso es:

$(q_O, 1100, A) \vdash (q_O, 100, 1A) \vdash (q_O, 00, 11A) \vdash (q_1, 0, 1A) \vdash (q_1, \lambda, A) \vdash (q_1, \lambda, \lambda)$

Por otra parte:

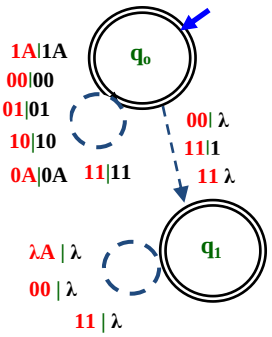
$S \rightarrow (q_o, A, q_1) \rightarrow$ **1** $(q_o, 1, q_1) (q_1, A, q_1) \rightarrow$
$\rightarrow$ **11** $(q_o, 1, q_1) (q_1, 1, q_1) (q_1, A, q_1) \rightarrow$
$\rightarrow$ **110** $(q_1, 1, q_1) (q_1, A, q_1) \rightarrow$
$\rightarrow$ **1100** $(q_1, A, q_1) \rightarrow$ **1100**

EJEMPLOS SUELTOS:

Ejemplo: $AP_{NDE} = (\Sigma,T,Q,A_o,q_o,\delta,F) = (\{0,1\},\{A,1,0\},\{q_o,q_1\},A,q_o,\delta,\emptyset)$

Pares	Diagrama	Matriz

$\Delta(q_o,0,A)=\{ (q_o,0A)\}$
$\Delta(q_o,1,A)=\{(q_o,1A)\}$
$\Delta(q_o,0,0)=\{(q_o,00)(q_1,\lambda)\}$
$\Delta(q_o,0,1)=\{(q_o,01)\}$
$\Delta(q_o,1,0)=\{(q_o,10)\}$
$\Delta(q_o,1,1)=\{(q_o,11)(q_1,1)(q_1,\lambda)\}$
$\Delta(q_1,0,0)=\{(q_1,\lambda)\}$
$\Delta(q_1,1,1)=\{(q_1,\lambda)\}$
$\Delta(q_1,\lambda,A)=\{(q_1,\lambda)\}$



Δ	(0,A)	(1,A)	(0,0)	(0,1)	(1,0)	(1,1)	(λ,A)
→*q_o	(q_o,0A)	(q_o,1A)	(q_o,00) (q_1,λ)	(q_o,01)	(q_o,10)	(q_o,11) (q_1,1) (q_1,λ)	
*q_1			(q_1,λ)			(q_1,λ)	(q_1,λ)

Proceso de la cadena **1100**

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Estado final
1 1 0 0	**1 1 0 0**	**1 1 0 0**	**1 1 0 0**	**1 1 0 0 λ**
(q_o,1,A)=$\{(q_o,1A)\}$	(q_o,1,1)=$\{(q_o,11)\}$	(q_o,0,1)=$\{(q_o,01)\}$	(q_o,0,0)=$\{(q_1,\lambda)$	(q_1,λ,1)= ?

Descripción instantánea → **$(q_O, 1100, A) \vdash (q_O,100, 1A) \vdash (q_O,00,11A) \vdash (q_1,0,011A) \vdash (q_1, \lambda,11A)$**

Ejemplo: Con el **$AP_{DE} = (\Sigma, T, Q, A_o , q_o , f , F) = (\{0, 1\}, \{ A, 1, 0\}, \{ q_o, q_1 \}, A, q_o, f, \emptyset)$**
Procesar la cadena **1 1 0 0**, para la función Transición: $f(q_i, a_i, A_i) \rightarrow (q_{i+1}, A_{i+1})$

$f (q_o, 1, A) = \{ (q_o, 1 A) \}$; $f (q_o, 1, 1) = \{ (q_o, 1 1) \}$; $f (q_o, 0, 1) = \{ (q_o, \lambda) \}$
$f (q_1, 0, 1) = \{ (q_1, \lambda) \}$; $f (q_1, \lambda, A) = \{ (q_1, \lambda) \}$

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Estado Final
1 1 0 0	**1 1 0 0**	**1 1 0 0**	**1 1 0 0**	**1 1 0 0 λ**
q_o |(q_o,1,A)=$\{(q_o,1A)$|**A**	q_o |(q_o,1,1)=$\{(q_o,11)$|**1A**	q_o |(q_o,0,1)=$\{(q_o, \lambda)$ |**11A**	q_o |(q_o,0,1)=$\{(q_o, \lambda)$|**1A**	q_o |(q_o,λ,A)= ? |**A**

Descripción instantánea: **$(q_O, 1100, A) \vdash (q_O, 100, 1A) \vdash (q_O, 00, 11A) \vdash (q_o, 0, 1A) \vdash (q_o, \lambda, A)$**
Como **f(q_o, λ, A)** = …, la cima de pila λ queda vacía, luego 1100 pertenece al lenguaje aceptado por el AP

Ejemplo: Construir un AP: **$M = (\Sigma, T, Q, A_o, q_o, \Delta, F)$** que acepte el lenguaje: **$L=\{w \in \{a, b\}^*\}$**

PARÁMETROS

- Símbolos Entrada. **$\Sigma = \{ a, b \}$**
- Símbolos Pila, **$T = \{ A, B, Z \}$**
- Conjunto Estados. **$Q = \{ q_1 , q_2 \}$**
- Símbolo Inicial Pila **$A_o = Z$**
- Estado Inicial. **$q_o = q_1$**
- Estados Finales **$F = \{ q_2 \}$**
- Reglas de Transición. **Δ:**

REGLAS de TRANSICION

$\Delta(q_1, \lambda, Z) = \{ (q_2, Z) \}$
$\Delta(q_1, a, Z) = \{ (q_1, A Z) \}$
$\Delta(q_1, b, Z) = \{ (q_1, B Z) \}$
$\Delta(q_1, a, A) = \{ (q_1, A A) \}$
$\Delta(q_1, b, A) = \{ (q_1, \lambda) \}$
$\Delta(q_1, a, B) = \{ (q_1, \lambda) \}$
$\Delta(q_1, b, B) = \{ (q_1, B B) \}$

MOVIMIENTOS

$(q_1, abba, Z) \vdash (q_1,bba, AZ)$
$\vdash (q_1, ba, Z)$
$\vdash (q_1, a, BZ)$
$\vdash (q_1, \lambda, Z)$
$\vdash (q_2, \lambda, Z)$

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3
a b b a	**a b b a**	**a b b a**	**a b b a**
q_1 |(q_1,a,Z)=$\{(q_1,AZ)$|**Z**	q_1 |(q_1,b,A)=$\{(q_1, \lambda)$|**AZ**	q_1 |(q_1,b,Z)=$\{(q_1,BZ)$|**Z**	q_1 |(q_1,a,B)=$\{(q_1, \lambda)$|**BZ**

Movimiento 4	Estado Final
a b b a λ	**a b b a λ**
q_1 |(q_1,λ,Z)=$\{(q_2, Z)$|**Z**	q_2 |(q_2,λ,Z)=$\{(q_2, Z)$|**Z**

Descripción instantánea:
$(q_1, abba, Z) \vdash (q_1, bba, AZ) \vdash (q_1, ba, Z) \vdash (q_1, a, BZ) \vdash (q_1, \lambda, Z) \vdash (q_2, \lambda, Z)$
La cadena está en L(M), porque el autómata llega y para en el estado de aceptación **q_2**

Ejemplo: El $AP_{DE} = (\{0, 1\}, \{0, 1, Z_o\}, \delta, q_o, Z_o, \{q_o, q_1, q_2\}, \{q_2\})$, procesa el L_{IC} lenguaje "w-reflejo" o palíndromos de longitud par sobre el alfabeto {0, 1} definido como: $L = \{ ww^R \mid w \in (0+1)^* \}$, cuyas funciones de transición se representan en formatos:

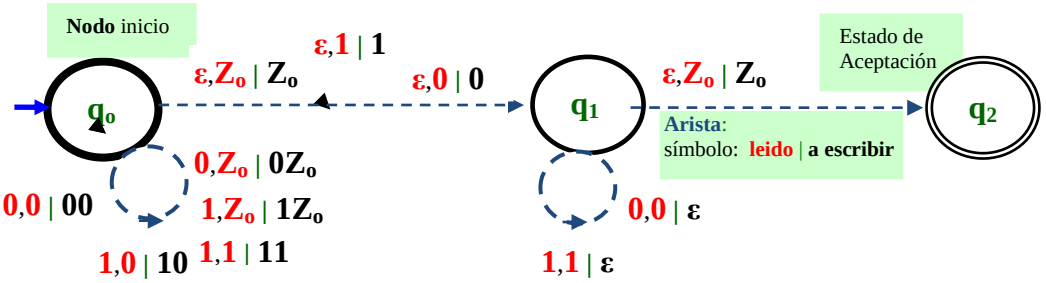
a) Pares:

$\delta(q_o, 0, Z_o) = \{(q_o, 0Z_o)\}$, $\delta(q_o, 1, Z_o) = \{(q_o, 1Z_o)\}$
$\delta(q_o, 0, 0) = \{(q_o, 00)\}$, $\delta(q_o, 0, 1) = \{(q_o, 01)\}$,
$\delta(.q_o, 1, 0) = \{(.q_o, 10)\}$, $\delta(q_o, 1, 1) = \{(q_o, 11)\}$
$\delta(q_o, \varepsilon, Z_o) = \{(q_1, Z_o)\}$, $\delta(q_o, \varepsilon, 0) = \{(q_1, 0)\}$, $\delta(.q_o, \varepsilon, 1) = \{(q_1, 1)\}$
$\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$, $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$, $\delta(q_1, \varepsilon, Z_o) = \{(q_2, Z_o)\}$

b) Matriz:

Δ	$(0,Z_o)$	$(1,Z_o)$	$(0, 0)$	$(0, 1)$	$(1, 0)$	$(1, 1)$	(ε,Z_o)	$(\varepsilon,0)$	$(\varepsilon,1)$
→ q_o	$\{(q_o,0Z_o)\}$	$\{(q_o,1Z_o)\}$	$\{(q_o,00)\}$	$\{(q_o,01)\}$	$\{(q_o,10)\}$	$\{(q_o,11)\}$	$\{(q_1, Z_o)\}$	$\{(q_1,0)\}$	$\{(q_1,1)\}$
. q_1			$\{(q_1, \varepsilon)\}$			$\{(q_1, \varepsilon)\}$	$\{(q_2,Z_o)\}$		
* q_2									

c) Diagrama de transiciones:



Ejemplo: El $AP_{DE} = (\{0,1\}, \{0,1,Z_o\}, \delta, q_o, Z_o, \{q_o, q_1, q_2\}, \{q_2\})$ Procesar la cadena **1111**:

Estado →	Inicial:	Movimiento 1:	Movimiento 2:	Movimiento 3:
Posición del cabezal	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1 ε
Estado	q_o	q_o	q_o	q_o
Cima de pila	Z_o	1 Z_o	1 1 Z_o	1 1 1 Z_o
Transición	$\delta(q_o,1,Z_o) = \{(q_o,1Z_o)\}$	$\delta(q_o,1,1) = \{(q_o,11)\}$	$\delta(q_o,1,1) = \{(q_o,11)\}$	$\delta(q_o,1,1) = \{(q_o,11)\}$
Representación instantanea	$(q_o, 1111, Z_o) \vdash$	$(.q_o, 111, 1Z_o) \vdash$	$(.q_o, 11, 11Z_o) \vdash$	$(.q_o, 1, 111Z_o) \vdash$

Estado →	Movimiento 4:	Movimiento 5:	Final	
Posición del cabezal	1 1 1 1 ε	1 1 1 1 ε	1 1 1 1 ε	
Estado	q_o	q_1	q_1	
Cima de pila	1 1 1 Z_o	1 1 1 1 Z_o	1 1 Z_o	
Transición	$\delta(q_o,\varepsilon,1) = \{(q_1,1)\}$	$\delta(q_1,\varepsilon,1) = ???$	$\delta(q_o,1,1) = \{(q_o,11)\}$	
Representación instantanea	$(.q_1, 1, 1Z_o) \vdash$	$(.q_1, \varepsilon, Z_o) \vdash$	$(.q_2, \varepsilon, Z_o)$	

Estado inicial	Movimiento 1	Movimiento 2	Movimiento 3	Movimiento 4	Estado Final
1 1 1 1	1 1 1	1 1	1	ε	ε
$(q_o,1,Z_o)=\{(q_o,1Z_o)\}$	$(q_o,1,1)=\{(q_o,11)\}$	$(q_o,1,1)=\{(q_o,11)\}$	$(q_o,1,Z_o)=\{(q_o,1Z_o)\}$	$(q_o,\varepsilon,Z_o)=\{(q_1,Z_o)\}$	$(q_1,\varepsilon,Z_o)=\{(q_2,Z_o)\}$

La descripción instantánea para procesar la cadena **1111**:

Estado inicial $(q_o, 1111, Z_o) \vdash$ Movimiento 1 $(q_o, 111, 1Z_o) \vdash$ Movimiento 2 $(q_o, 11, 11Z_o) \vdash$ Movimiento 3 $(q_1, 11, 11Z_o) \vdash$ Movimiento 4 $(q_1, 1, 1Z_o) \vdash$ Movimiento 5 $(q_1, \varepsilon, Z_o) \vdash$ Estado Final (q_2, ε, Z_o)

Se representa, con el símbolo: $\vdash^*$ **El proceso total**: $(q_o, 1111, Z_o) \vdash^* (q_2, \varepsilon, Z_o)$

$\vdash^+$ **Un proceso parcial**: $(q_o, 1111, Z_o) \vdash^+ (q_o, 11, 11Z_o)$

A_F:AUTÓMATA FINITO:

La finalidad del A_F es reconocer palabras del **lenguaje regular** L_{RE} generado por una gramática regular G_{RE}, que se carateriza por:

1. **DEFINICIÓN:** $G_{RE} = G_3 = (\Sigma_T, \Sigma_N, S, P)$

2. **PRODUCCIONES:**
   La **longitud de su parte izquierda debe ser 1, no terminal**, la parte derecha puede ser una secuencia de solo terminales, o de terminales **con un no terminal como sufijo o como prefijo**

   El **FORMATO ESTÁNDAR** es: $N_1 \rightarrow t\,N_2$, $N \rightarrow t$, $S \rightarrow \lambda$, y puede ser:

   - **LINEAL por DERECHA** G_{RED}: $P \subseteq \Sigma_N \times \Sigma_T^* (\Sigma_N \cup \lambda)$
     $P = \{ (S ::= \lambda)$ ó $(A ::= aB)$ ó $(A ::= a) \mid A, B \in \Sigma_N, a \in \Sigma_T + \}$

   - **LINEAL por IZQUIERDA** G_{REI}: $P \subseteq \Sigma_N \times (\Sigma_N \cup \lambda)\,\Sigma_T^*$
     $P = \{ (S ::= \lambda)$ ó $(A ::= Ba)$ ó $(A ::= a) \mid A, B \in \Sigma_N, a \in \Sigma_T + \}$

   EJEMPLO: La G_{RE}: $G_3 = (\Sigma_T, \Sigma_N, S, P) = (\{ 0, 1 \}, \{ A, B \}, S, P)$

Producciones		Derivaciones		
Pares	Simplificado	A→1B→11	A→1B→10C→101	A→1B→11C→111
P = {(A::=1B), (B::=1), (B::= 0C), (B::=1C), (C::= 1)}	P : A ::= 1B B::= 1 |0C| 1C C ::= 1	A / \ 1 B | 1	A / | 1 B / \ 0 C | 1	A / | 1 B / \ 1 C | 1
Genera en **formato estandar** el **lenguaje regular**: $L_{RE} = \{ w^n \mid w = w^{-1} \} = \{11, 101, 111, \ldots\}$				

3. **CONDICIONES:** Un L_{RE} se genera por una G_{RE} si cumple:
   1. Todo lenguaje finito es un L_{RE}
   2. Si L_1 y L_2 son L_{RE}, luego $L_1 \cup L_2$ y $L_1 \cap L_2$ son también L_{RE}
   3. Si L es un L_{RE} luego L^* es también L_{RE}
   4. Todo L_{RE} puede definirse por las condiciones 1, 2 y 3.

Para el alfabeto $\Sigma = \{a, b\}$ Se afirma que:

Condición de **recursividad**	El lenguaje	Es regular porque cumple con la condición:
1. $\emptyset$ Es un L_{RE}.	$\emptyset$ y $\{\varepsilon\}$	1 y 2 de recursividad
2. $\{ \varepsilon \}$ Es un L_{RE}.	{a} y {b}	3 de recursividad
3. Para todo $a \in \Sigma$, {a} Es un L_{RE}	{a, b}	4, unión de los lenguajes {a} y {b}
4. Si A y B son L_{RE}s, entonces $A \cup B$, $A \cap B$ y A^* son L_{RE}s	{a . b}	Las dos razones anteriores
5. Ningún otro lenguaje Σ es regular.	{a, ab, b}	Las razones anteriores
	$\{ a^i \mid i \geq 0 \}$	4, a^i es potencia iésima de {a} que es concatenación sucesiva.
	$\{a^i b^j \mid i \geq 0, j \geq 0 \}$	Las razones anteriores
	$\{ (ab)^i \mid i \geq 0 \}$	Las razones anteriores

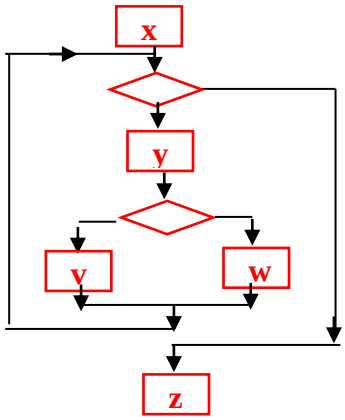
EXPRESION REGULAR: E_{RE}

<table>
<tr><th>E_{RE} es una FORMA</th><th colspan="2">Ejemplo</th></tr>
<tr><td rowspan="7">• SIMPLIFICADA de representación del L_{RE}</td><td>E_{RE}</td><td>L_{RE}</td></tr>
<tr><td>Φ</td><td>{}</td></tr>
<tr><td>. λ</td><td>{ λ }</td></tr>
<tr><td>. x</td><td>{ x }</td></tr>
<tr><td>E_1+E_2</td><td>$L_1 \cup L_2$</td></tr>
<tr><td>$E_1.E_2$</td><td>$L_1 \cap L_2$</td></tr>
<tr><td>E^*</td><td>L^*</td></tr>
<tr><td rowspan="8">• ALGEBRAICA definida sobre el alfabeto Σ y otro especial
Σ'={ +, *, . , ϕ, λ, (,) } si cumple las condiciones:
1. ϕ, λ, x∈Σ son E_{RE}
2. Si E_1 y E_2 son E_{RE}s luego: E_1+E_2 y $E_1.E_2$ son también E_{RE}s
3. Si E es un E_{RE} luego E^* es también E_{RE}
4. Toda E_{RE} se define por las condiciones 1, 2 y 3.</td><td colspan="2">Si r, α, s y t son E_{RE}</td></tr>
<tr><td>Ø.α = α . Ø = Ø</td><td>α * .α* = α*</td></tr>
<tr><td>α . α* = α* . α</td><td>Ø* = ε</td></tr>
<tr><td>α* = λ + α .α*</td><td>r(s r)*= (r s)* r</td></tr>
<tr><td colspan="2">(r* s)* = ε ∪ (r ∪ s)* s</td></tr>
<tr><td colspan="2">(rs*)* = ε ∪r (r ∪ s)* s</td></tr>
<tr><td colspan="2">s(r∪ε)*(r∪ε) ∪s = sr*</td></tr>
<tr><td colspan="2">(a* b)* = {a}∪{b}.{c}*)*</td></tr>
<tr><td colspan="2">(a* b)* = ε∪(a ∪ b)*b</td></tr>
<tr><td rowspan="8">• RECURSIVAMENTE definida sobre el alfabeto Σ:
1. Ø y ε Son E_{RE}s
2. .a Es una E_{RE} para toda: a∈Σ
3. Si r y s son E_{RE}s, luego r+s, r.s y r* serán E_{RE}s
4. Ninguna otra secuencia de caracteres de Σ es E_{RE}.</td><td colspan="2">Si r, s y t son E_{RE}</td></tr>
<tr><td>r . s = s . r</td><td>r . Ø = r = Ø . r</td></tr>
<tr><td>r . r = r</td><td>(r.s) .t = r . (s.t)</td></tr>
<tr><td>r ε = ε r = r</td><td>r Ø = Ø r = r</td></tr>
<tr><td colspan="2">(r s) t = r (s t)</td></tr>
<tr><td colspan="2">r r* = r* r = r.r* = r^+</td></tr>
<tr><td>λ* = λ</td><td>Ø* = λ</td></tr>
</table>

La E_{RE} simplifica la representación y análisis, por ejemplo, de:

- La estructura algorítmica del diagrama: **x.(y.(v+w))*.z**
- Un léxico de un lenguaje de programación: **PalabraReservada=do+while+switch+case+…** .
- Un léxico de un lenguaje numérico: **Digito=0+1+2+…**
- Un lenguaje **01*** formado por palabras iniciadas en **0**, seguido uno, o más o ningún **1**
- Cadenas **(0+1)*** de ninguno, uno o más números **0** ó **1**

x . (y . (v + w))* . z



La E_{RE} posee las siguientes

Caracteristicas	Ejemplo
• **Prioridad** de operadores	**Primero**: * (Potencia); **luego**: . (Producto) y **finalmente**: + (Unión)
• **Abreviatura**:	**{ a } = a**

• **Correspondencia**: A toda expresión regular le corresponde un lenguaje regular y visceversa.

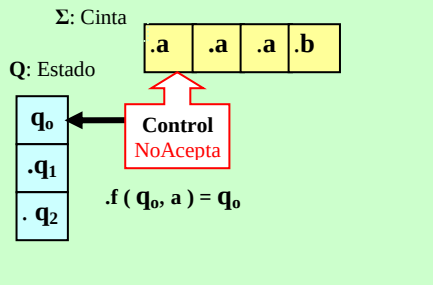
E_{RE}	L_{RE}	E_{RE}	L_{RE}
Φ	{}	a.b	{ a, b }
λ	{ λ }	a+b	{a, b }={a}∪{b}
x	{ x }	a *	{a*}
E_1+E_2	$L_1 \cup L_2$	Ø*	{ε}
E_1.E_2	$L_1 \cap L_2$	a^+	$\{a\}^+$
E*	L*	c*(a+b.c*)*	{c}* ({a}∪{b}∩{c}*)*

	E_{RE}	
• **Propiedades**	Φ*= λ	E*= λ +E.E*
	E.E*=E*.E	$(E_1{}^*+E_2{}^*)^* = (E_1+E_2)^* = (E_1{}^*+E_2{}^*)^*. E_1{}^*$
		$E^* = (\lambda+E^1+E^2+\ldots+E^{n-1}).(E^n)^*$

A_F:AUTÓMATA FINITO: La finalidad del A_F es reconocer palabras del L_{RE}.

Definición: $A_F = (\sum, q_0, Q, \delta, F)$

Arquitectura



- $\sum$: Alfabeto de entrada registrada en la cinta.
- q_0: Estado Inicial: $q_0 \in Q$
- **Q**: Colección de Estados.
- **F** : Colección de Estados Finales $F \in Q$
- **δ**: Función de Transición de Estados:
  $\delta: Q \times \sum \rightarrow Q$: δ (Estado Actual: q_i, Símbolo de Entrada: **a**) → (Nuevo Estado: q_{i+1})

A_F: FUNCIONAMIENTO: El A_F comienza en q_0, procesa cada símbolo de cadena y cambia de estado según **δ**.

- Luego de procesar el último de los símbolos de la cadena de entrada, el A_F se detiene en el estado final del proceso.
- Si el estado final en el que se detuvo es un estado de aceptación, la cadena pertenece al lenguaje reconocido por el A_F; en caso contrario, la cadena no pertenece a tal lenguaje.
- El A_F tiene solo un q_0 y **F** puede contener más de un elemento y un estado final corresponda al mismo q_0.
- El L_{RE} aceptado por un A_F es: $L_{RE} = \{w; \delta^*(q_0, w) \in F \}$

El A_F, es determinista; si, para cada estado $q \in Q$ del autómata, y con cualquier símbolo $a \in \Sigma$ del alfabeto leído, existe siempre una transición posible **δ(q, a)**.

En un A_{FDE} no pueden existir transiciones del tipo:

- $\delta(q, a) = q_1$ y $\delta(q, a) = q_2$, siendo $q_1 \neq q_2$
- $\delta(q, \varepsilon)$, salvo que **q** sea un estado final, sin transiciones hacia otros estados.

A_F: LENGUAJE ACEPTADO por el AF_{NDE}: Es el conjunto de cadenas de símbolos terminales que le permiten llegar a un estado final de aceptación. A tal fin, la:

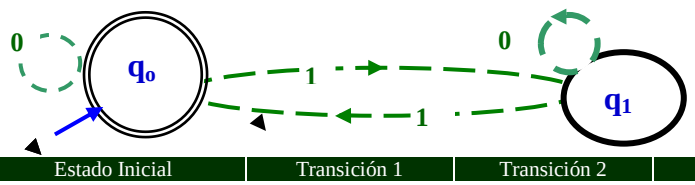
- Función de transición: f. $Q^*\Sigma \rightarrow 2^Q$
- Función de transición ampliada: f. $Q^*\Sigma^* \rightarrow 2^Q$

Donde:

- $f'(q, \lambda) = \lambda$ - clausura(q)
- $f'(q, ax) = \{p \in f'(r, x) \mid r \in f'(q,a)\} = \{p \in Q \mid \exists r \in f'(q,a)$ y $p \in f'(r,x0) \}$ siendo: $a \in \Sigma, r \in \Sigma^*$

El lenguaje aceptado por el AF_{NDE} es: $L_{FNDE} = \{ x \in \Sigma^* \mid f'(q_o, x) \cap F \neq \Phi \}$

Ejemplo: $AF_{DE} = (\Sigma, q_o, Q, f, F) = (\{0, 1\}, q_o, \{ q_o , q_1 \}, f, \{ q_o \})$

Pares	Diagrama	Matriz
.f (q_o, 0) = q_o .f (q_o, 1) = q_1 .f (q_1, 0) = q_1 .f (q_1, 1) = q_o		.f | 0 | 1 →*q_o | .q_o | .q_1 q_1 | .q_1 | .q_o

Proceso de la cadena: 0110	Estado Inicial	Transición 1	Transición 2	Transición 3	Estado final
	0 1 1 0 f(q_o, 0)= q_o	0 **1** 1 0 f(q_o, 1)= q_1	0 1 **1** 0 f(q_1, 1)= q_o	0 1 1 **0** f(q_o, 0)= q_o	0 1 1 0 … f(q_o, ..)= …
Descripción Instantanea Estado actual, luego cadena que falta leer.	q_o 0 1 1 0 ⊢	q_o 1 1 0 ⊢	q_1 1 0 ⊢	q_o 0 ⊢	q_o
	Total: q_o 0110 ⊢* q_o				
	Parcial: q_o0110 ⊢+ q_1 1 0				
El AF_{DE} transita por los estados:	q_o $\xrightarrow{0}$	q_o $\xrightarrow{1}$	q_1 $\xrightarrow{1}$	q_o $\xrightarrow{0}$	q_o

El AF_{DE} puede ser:

- **CONEXO** Sus estados son accesibles desde el estado inicial
- **COMPLETO** Sus estados tienen transiciones con cada simbolos de entrada

Un estado del AF_{DE} puede ser:

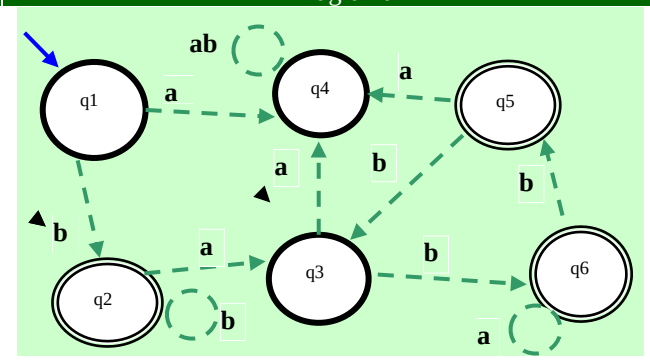
- **SUMIDERO** O de RECHAZO: No es final y transiciona con todos los símbolos de entrada a si mismo
- **GENERADOR** Solo salen transiciones desde el.

El $AF_{DE} = (\{0, 1\}, \{q_o, q_1\}, f, q_o, \{q_o\})$ Es conexo, completo, no tiene estado sumidero ni estado generador

Ejemplo:

$$AF_{DE} = (\Sigma, q_o, Q, f, F) = (\{a,b\}, \{ q_1, q_2, q_3, q_4, q_5, q_6, q_1 \}, f, q_1, \{ q_2, q_5, q_6\})$$

Formato de:

Pares	Diagrama	Matriz
.f (q_1,a)=q_4 .f (q_1,b) = q_2 .f (q_2,a)=q_3 .f(q_2,b) = q_2. .f (q_3,a)=q_4 .f (q_3,b) = q_6 .f (q_4,a)=q_4 .f (q_4,b) = q_4 .f (q_5,a)=q_4 .f (q_5,b) = q_3 .f (q_6,a)=q_6 .f (q_6,b) = q_5		.f | A | b →q_1 | .q_4 | .q_2 *q_2 | .q_3 | .q_2 q_3 | .q_4 | .q_6 q_4 | .q_4 | .q_4 *q_5 | .q_4 | .q_3 *q_6 | .q_6 | .q_5

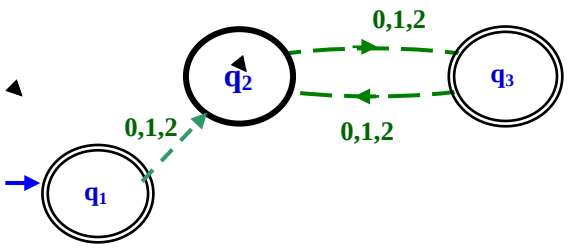
A_F: Diseño de un AF_{DE} para una E_{RE}:

Ejemplo: La E_{RE} = ((0+1+2).(0+1+2))* puede ser procesada por el:

$AF_{DE} = (\Sigma, q_o, Q, f, F) = (\{0, 1, 2\}, \{q_1, q_2, q_3\}, f, q_1, \{q_1, q_3\})$,

Pares	Diagrama	Matriz

.f (q_1, 0) = q_2
.f (q_1, 1) = q_2
.f (q_1, 2) = q_2
.f (q_2, 0) = q_3
.f (q_2, 1) = q_3
.f (q_2, 2) = q_3
.f (q_3, 0) = q_2
.f (q_3, 1) = q_2
.f (q_3, 2) = q_2



.f	0	1	2
→* q_1	.q_2	.q_2	.q_2
q_2	.q_3	.q_3	.q_3
*q_3	.q_2	.q_2	.q_2

AF_{DE}: FUNCIÓN DE TRANSICIÓN EXTENDIDA: Una función de transición **δ** se generaliza a una función **δ***, que opera sobre estados y secuencias de símbolos, en lugar de símbolos individuales del alfabeto .

Esta nueva función de transición se define: **$\delta^*: Q\text{x}\Sigma \rightarrow Q$**, así caracteriza los autómatas de manera más simple.

La función **δ*** se expresa recursivamente, definiendo para toda cadena **x∈ Σ***, todo símbolo **a∈ Σ**, y un estado **q∈ Q**:

- **$\delta^*(q, \varepsilon)=q$**: Base inductiva, siendo **ε** la cadena vacía, y
- **$\delta^*(q, xa)= \delta(\delta^*(q, x), a)$**: La propia inducción.

Ejemplo :

$AF_{DE} = (\Sigma, q_o, Q, f, F) = (\{ a,b\},\{q_1,q_2,q_3,q_4,q_5,q_6\},f,q_1, \{ q_2,q_5,q_6\})$,

Pares de parámetros				Matriz de transición		
				.f	a	b
.f (q_1, a)	= q_4	.f (q_2, a)	= q_3	→ q_1	.q_4	.q_2
.f (q_1, b)	= q_2	.f (q_2, b)	= q_2	*q_2	.q_3	.q_2
.f (q_3, a)	= q_4	.f (q_4, a)	= q_4	.q_3	.q_4	.q_6
.f (q_3, b)	= q_6	.f (q_4, b)	= q_4	.q_4	.q_4	.q_4
.f (q_5, a)	= q_4	.f (q_6, a)	= q_6	*q_5	.q_4	.q_3
.f (q_5, b)	= q_3	.f (q_6, b)	= q_5	*q_6	.q_6	.q_5

Las transiciones extendidas:

Proceso de la cadena: **ab**

$$.f'(q_2, ab) = f'(f(q_2,a), b) = f'(q_3,b) = f'(f(q_3,b), \lambda) = f'(q_6, \lambda) = q_6$$

q_3 q_6

Proceso de la cadena: **bba**

$$f'(q_1, bba)= f'(f(q_1,b), ba) = f'(q_2,ba) = f'(f(q_2,b), a) = f'(q_2,a) = f'(f(q_2,a), \lambda)= f'(q_3, \lambda)= q_3$$

q_2 q_2 q_3

AF_{NDE}: AUTÓMATA FINITO NO DETERMINISTA:

En un AF_{NDE}

- Existe al menos un estado **q∈Q**, tal que para un símbolo **a∈Σ** del alfabeto, existe más de una transición **δ(q,a)**.

- Puede existir transiciones del tipo:
  - **$\delta(q,a)=q_1$** y **$\delta(q,a)=q_2$**, siendo **$q_1 \neq q_2$**;
  - **δ(q,ε)**, siendo **q** un estado no-final, o bien un estado final pero con transiciones hacia otros estados.
    En este caso, el autómata es un **AF_{NDE}** con **transiciones vacías o transiciones ε**: **AF_{NDE}-ε**, capaz de cambiar de estado sin procesar ningún símbolo de entrada.

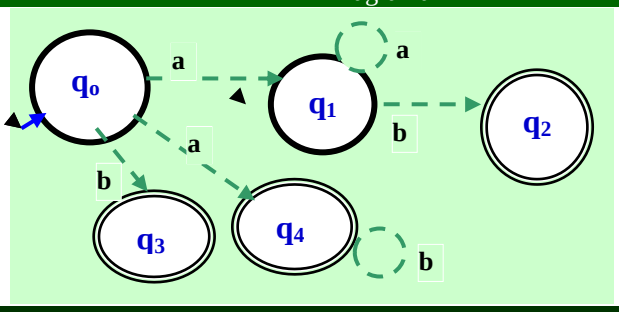
La **FUNCIÓN DE TRANSICIÓN** del autómata:

- **AF_{DE}** se define: **$\delta: Q \times \Sigma \rightarrow Q$**
- **AF_{NDE}** se define como: **$\delta: Q \times \Sigma \rightarrow P(Q)$**
- **AF_{NDE}-ε**, se define de la forma: **$\delta: Q \times \{\Sigma \cup \varepsilon\} \rightarrow P(Q)$**, donde P(Q) es el conjunto potencia de Q. Luego el **AF_{DE}** es un caso particular del **AF_{NDE}**, porque $Q \in P(Q)$.

En el cómputo de un **AF_{NDE}**, puede estar en varios estados a la vez, generando ramificación de configuraciones

Ejemplo: **$AF_{NDE} = (\Sigma, q_o, Q, f, F) = (\{a,b\}, \{q_o, q_1, q_2, q_3, q_4\}, \Delta, q_o, \{q_2, q_3, q_4\})$**

Acepta el lenguaje **$L_{RE} = \{ a^* b \cup a b^* \}$** con las **funciónes Δ de transición**, en forma de:

Pares	Diagrama	Matriz
$\Delta(q_o,a)=\{q_1, q_4\}$ $\Delta(q_o,b)=\{q_3\}$ $\Delta(q_1,a)=\{q_1\}$ $\Delta(q_1,b)=\{q_2\}$ $\Delta(q_4,b)=\{q_4\}$		Δ | A | b →q_o | $\{q_1,q_4\}$ | $\{q_3\}$ q_1 | $\{q_1\}$ | $\{q_2\}$ *q_2 | { } | { } .q_4 | { } | $\{q_4\}$

Proceso de la cadena: aab

Alternativa 1

Estado Inicial	Transición 1	Transición 2	Estado final
a a b	**a a b**	**a a b**	**a a b ..**
$f(q_o, a)= q_1$	$f(q_1, a)= q_1$	$f(q_1, b)= q_2$	$f(q_2, ..)= ..$

- Descripción instantánea: $q_o\ a\ a\ b \vdash_a q_1\ a\ b \vdash_a q_1\ b \vdash_b q_2$
- El autómata transita los estados: $q_o \rightarrow q_1 \rightarrow q_1 \rightarrow q_2$

Alternativa 2

Estado Inicial	Estado final
a a b	**a a b**
$f(q_o, a)= q_4$	$f(q_4, a)= ..$

- Descripción instantánea: $q_o\ a\ a\ b \vdash_a q_4\ a\ b \vdash q_2$
- El autómata transita los estados: $q_o \rightarrow q_4$

AF_{NDE}: CLASES de AUTOMATAS FINITO NO DETERMINISTA:

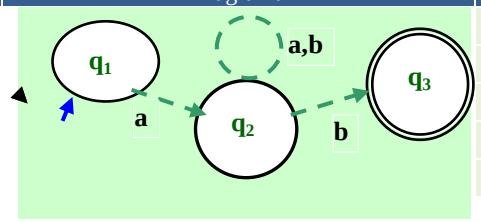
I.- $AF_{NDE} = (\Sigma, Q_o, Q, f, F)$, donde: $f: Q \times \Sigma \rightarrow 2^Q$, ($Q_o$ es subcojunto de Q)

Procesar E_{RE} = a.(a+b)*.b

Pares

$\Delta(q_1, a)=\{q_2\}$ $\Delta(q_1,b)= \emptyset$
$\Delta(q_2, a)=\{q_2\}$ $\Delta(q_2,b) =\{q_2\ q_3\}$
$\Delta(q_3, a)= \emptyset$ $\Delta(q_3, b) = \emptyset$

Diagrama



Matriz

	a	B
→*q_1	{q_2}	Ø
q_2	{q_2}	{q_2 q_3}
*q_3	Ø	Ø

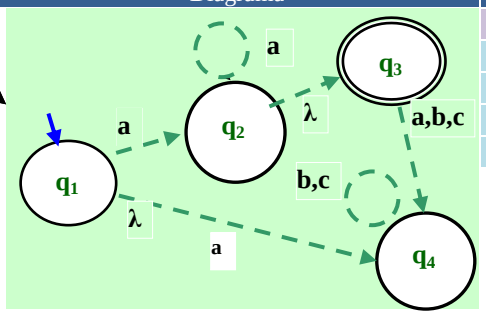
II.- $AF_{Lazy} = (\Sigma, q_o, Q, f, F)$, donde: $f: Q \times \Sigma^* \rightarrow 2^Q$, (La entrada puede ser **cualquier cadena sobre Σ**)

Procesar E_{RE} = c.a*.b*+(c.b)*.c.a.c.b*

Pares

$\Delta(q_1,a)= \emptyset$ $\Delta(q_1,b)= \emptyset$
$\Delta(q_1,c)= \{q_2\}$ $\Delta(q_1,cd)= \emptyset$
$\Delta(q_1,cac)=\{q_4\}$ $\Delta(q_1,\lambda)= \emptyset$
$\Delta(q_2,a)=\{q_2\}$ $\Delta(q_2,b)= \emptyset$
$\Delta(q_2,c)= \emptyset$ $\Delta(q_2,cd)= \emptyset$
$\Delta(q_2,cac)= \emptyset$ $\Delta(q_2,\lambda)= \{q_3\}$
$\Delta(q_3,a)= \emptyset$ $\Delta(q_3,b)= \{q_3\}$
$\Delta(q_3,c)= \emptyset$ $\Delta(q_3,cd)= \emptyset$
$\Delta(q_3,cac)= \emptyset$ $\Delta(q_3,\lambda)= \emptyset$
$\Delta(q_4,a)= \emptyset$ $\Delta(q_4,b)= \emptyset$
$\Delta(q_4,c)= \emptyset$ $\Delta(q_4,cd)= \{q_4\}$
$\Delta(q_4,cac)=\{q_3\}$ $\Delta(q_4,\lambda)= \emptyset$

Diagrama



Matriz

AF_{Lazy}	a	b	c	cd	cac	λ
→q_1	Ø	Ø	{q_2}	Ø	{q_4}	Ø
q_2	{q_2}	Ø	Ø	Ø	Ø	{q_3}
*q_3	Ø	{q_3}	Ø	Ø	Ø	Ø
q_4	Ø	Ø	Ø	{q_4}	{q_3}	Ø

III.- $AF_{ND\lambda} = (\Sigma, q_o, Q, f, F)$, donde: $f: Q \times \Sigma \cup \{\lambda\} \rightarrow 2^Q$ La entrada puede ser vacia, transiciona sin leer

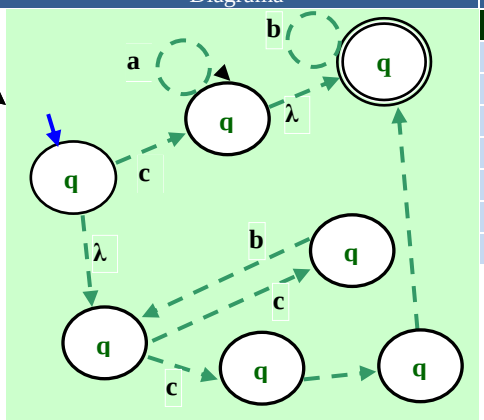
Tienen transiciones de un estado a otro sin depender de una entrada, así no consumen ningún símbolo de entrada.
Para elegir la transición λ se usa igual criterio de las demás transiciones múltiples del AF_{NDE}

Ejemplo: Para procesar :
E_{RE} = c.a*.b*+(c.b)*.c.a.c.b*

Pares

$\Delta(q_1,a)= \emptyset$ $\Delta(q_1,b) = \emptyset$
$\Delta(q_1,c)=\{q_2\}$ $\Delta(q_1, \lambda)= \{q_5\}$
$\Delta(q_2,a)=\{q_2\}$ $\Delta(q_2,b) = \emptyset$
$\Delta(q_2,c)= \emptyset$ $\Delta(q_2, \lambda)= \{q_3\}$
$\Delta(q_3,a)= \emptyset$ $\Delta(q_3,b)= \{q_3\}$
$\Delta(q_3,c)= \emptyset$ $\Delta(q_3, \lambda)= \emptyset$
$\Delta(q_4,a)= \emptyset$ $\Delta(q_4,b)= \{q_4\}$
$\Delta(q_4,c)= \emptyset$ $\Delta(q_4, \lambda)= \{q_4\}$
$\Delta(q_5,a)= \emptyset$ $\Delta(q_5,b)= \emptyset$
$\Delta(q_5,c)=\{q_4q_6\}$ $\Delta(q_5, \lambda)= \emptyset$
$\Delta(q_6,a)= \{q_7\}$ $\Delta(q_6,b)= \emptyset$
$\Delta(q_6,c)= \emptyset$ $\Delta(q_6, \lambda)= \emptyset$
$\Delta(q_7,a)= \emptyset$ $\Delta(q_7,b)= \emptyset$
$\Delta(q_7,c)= \{q_3\}$ $\Delta(q_7, \lambda)= \emptyset$

Diagrama



Matriz

$AF_{ND\lambda}$	a	b	c	λ
→q_1	Ø	Ø	{q_2}	{q_5}
q_2	{q_2}	Ø	Ø	{q_3}
*q_3	Ø	{q_3}	Ø	Ø
q_4	Ø	{q_4}	Ø	Ø
q_5	Ø	Ø	{q_4 q_6}	Ø
q_6	{q_7}	Ø	Ø	Ø
q_7	Ø	Ø	{q_3}	Ø

AF_{DE}: EQUIVALENCIA Y MINIMIZACIÓN:

- Los automatas AF_{DE1} y AF_{DE2} son equivalentes si aceptan los mismos lenguajes.

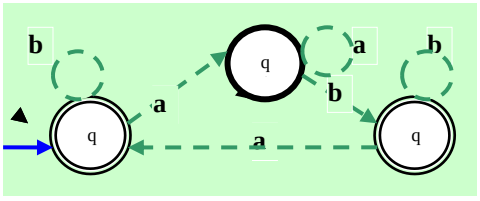
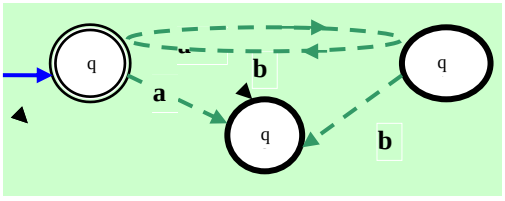
- De los autómata equivalentes, el AF_{DE} de menor cantidad de estados es el AF_{DE} **mínimo**

AF_{DE}: EQUIVALENCIA con el AF_{NDE}:

Como el **AF** es determinista por naturaleza, siempre un AF_{NDE} tiene un equivalente AF_{DE}.

Teorema: Sea M = **$(q_o, \Sigma, Q, \Delta, F)$** un AF_{DE}, luego existe un AF_{NDE} M'= **$(q'_o, \Sigma', Q', f, F')$** que es equivalente a M.

Ejemplo: Para el lenguaje **$A = (a\,b \cup a\,b\,a)^*$**

DEFINICIÓN	DETERMINISTA	NO DETERMINISTA
	$AFD = (q'_o, \Sigma', Q', f, F')$ **$A_1 = (q_o, \{a,b\}, \{q_o, q_1, q_2\}, f, \{q_o, q_2\})$**	**$AFD = (q_o, \Sigma, Q, \Delta, F)$** **$A_1 = (q_o, \{a,b\}, \{q_o, q_1, q_2\}, \Delta, \{q_o\})$**
Estados de TRANSICIÓN: **Pares de valores**	• $.f(q_o, a) = q_1$ • $.f(q_o, b) = q_o$ • $.f(q_1, a) = q_1$ • $.f(q_1, b) = q_2$ • $.f(q_2, a) = q_o$ • $.f(q_2, b) = q_2$	• $\Delta(q_o, a) = \{q_1\}$ • $\Delta(q_o, b) = \emptyset$ • $\Delta(q_1, a) = \emptyset$ • $\Delta(q_1, b) = \{q_o, q_2\}$ • $\Delta(q_2, a) = \{q_o\}$ • $\Delta(q_2, b) = \emptyset$
Tabla de Transición	.f | .a | .b $\rightarrow q_o$ | q_1 | q_o $*q_1$ | q_1 | q_2 $.q_2$ | q_o | q_2	Δ | .a | .b $\rightarrow q_o$ | $\{q_1,q_2\}$ | 0 $*q_1$ | 0 | $\{q_o,q_2\}$ $.q_2$ | 0 | 0
Diagrama Transición		

CONCLUSION: Todo lenguaje aceptado por un AF_{NDE} es también aceptado por un AF_{DE}

Dos **AF** son equivalentes, si ambos reconocen el mismo L_{RE}.y como toda E_{RE} puede ser expresada como un AF_{DE}, y viceversa.

- Dada una, E_{RE} es posible construir un $AF_{NDE\text{-}\varepsilon}$ que reconozca dicho lenguaje, por ejemplo mediante el algoritmo de Thompson.
- Todo $AF_{NDE\text{-}\varepsilon}$ puede transformarse en un AF_{NDE} equivalente, así como todo AF_{NDE} puede transformarse en un AF_{DE} equivalente, mediante el método llamado construcción de conjunto potencia.
- Por transitividad, para cualquier AF_{NDE} siempre existe un AF_{DE} equivalente, y viceversa.

Para diseñar un **AF**, conviene construir previo un $AF_{NDE\text{-}\varepsilon}$, más simple de construir, con menos restricciones en sus transiciones. Luego tal autómata se reduce a un AF_{NDE}, y finalmente a un AF_{DE}, el cual por sus características deterministas ya puede ser implementado sin problemas utilizando un lenguaje de programación.

A_F: MINIMIZACIÓN de un AF_{DE}

Si un AF_{DE1} tiene estados equivalentes, se puede lograr otro equivalente AF_{DE2} **minimizado**, **agrupando los estados equivalentes** de AF_{DE1} y usando menos estados. Ahora si solo se eliminan los estados inaccesibles se habrá logrado un AF_{DE2} **simplificado**.

AF_{DE} EJEMPLO 1 DE MINIMIZACIÓN

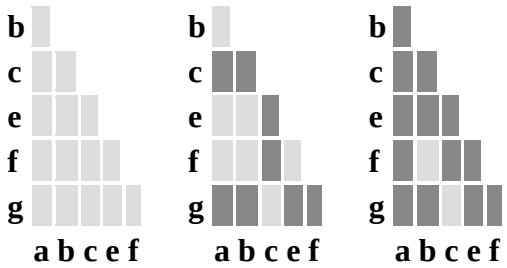
- Dos estados del AF_{DE} son estados equivalentes si al unirse en un sólo estado, pueden reconocer el mismo lenguaje regular que si estuviesen separados.
- Esta unión de estados implica la unión tanto de sus transiciones de entrada como de salida.
- Si dos estados no son equivalentes, se dice que son estados **distinguibles**.
- Un estado final con un estado no-final nunca serán equivalentes.
- Un AF_{DE} está minimizado, si todos sus estados son distinguibles y alcanzables.

ALGORITMO DE MINIMIZACIÓN	Ejemplo

1) Eliminar los estados inaccesibles del autómata.
   Este 1er diagrama, es un autómata con el estado inaccesible **d**, que puede eliminarse inmediatamente.

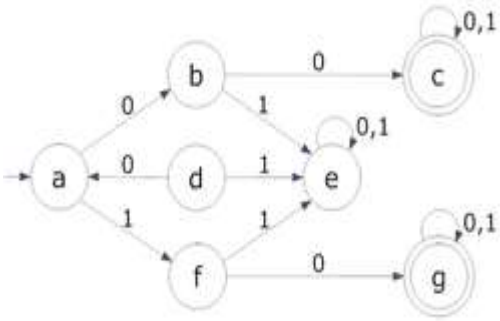
2) Tabular los pares **(p, q)** de estados restantes.
3) Marcar en la tabla las entradas donde un estado es final y el otro es no-final (pares de estados **distinguibles**)
   Construir la tabla de pares de estados, marcando filas y columnas de los estados finales **c** y **g**, salvo la celda que representa el par **(c,g)**, que al ser ambos estados finales, pueden ser estados equivalentes.

Tablas para búsqueda de estados equivalentes

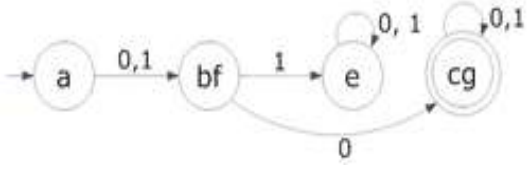


4) Para cada par **(p, q)** y cada símbolo a del alfabeto, tal que
   r = δ(p,a) y **s = δ(q,a)**
   Si **(r, s)** fué marcado, entonces **p** y **q** también son distinguibles, luego tanto marcar la entrada **(p, q)**.
   Caso contrario, colocar **(p, q)** en lista asociada a la entrada **(r, s)**.
   Marcar celdas restantes, ver que el par **(b, f)** queda asociado con el par **(c, g)**, y así se obtiene el autómata final, agrupando los estados **b** y **f**, así como **c** y **g**, tal, muestra en el segundo diagrama.

5) Agrupar los pares de estados no marcados.
6) Luego del 3er paso, si la tabla creada queda marcada completamente, entonces el AF_{DE} inicial ya era mínimo.



AFD con estados redundantes.



AFD minimizado.

La complejidad computacional del problema de minimizar un AF_{DE} es polinomial

Para agrupar estados equivalentes:

- Se arma una partición de **Q** formada por 2 elementos: Los estados de aceptación y no de aceptación.
- Esta partición se va ajustando, separando en distitos elementos a los estados que no son equivalentes, considerando que una partición de **Q** consiste en dividir **Q** en varios subconjuntos $\{ G_i\}_{1<i<n}$ de modo que: $G_i \cap G_j = \Phi \ \forall \ i \neq j$ y $U_{1<i<n} = Q$

AF_{DE} EJEMPLO 2 DE MINIMIZACIÓN

Con el siguiente algoritmo, minimicemos el resultado del paso de AF_{NDE} a AF_{DE} ,
Analizar con **a** y **b** los estados para: Partición = $\{G_1, G_2\}$ $G_1 = \{q_2, q_3, q_4, q_5, q_6\}$ $G_2 = \{ q_0, q_1 \}$

AF_{DE} ALGORITMO DE MINIMIZACIÓN

Input: $AF_{DE1} = (\Sigma, Q, f, q_o, F)$
Output: $AF_{DE2} = (\Sigma, Q', f', q_o', F')$
Begin
Partición = $\{ G_1, G_2\}$ donde $G_1 = F$ y $G_2 = Q / F$
Auxiliar = θ
While Auxiliar ≠ Partición **do**
Auxiliar = Partición
$\forall \ G_1 \in$ Partición y $\forall \ G_1 \in \Sigma$
Separar en distintos grupos a los estados **s** y $t \in G_i$, siempre que $f(s, a) \in G_j$, $f(t, a) \in G_k$ siendo $j \neq k$
end while

G_1	Símbolo **a**
$q_2 \rightarrow$	$q_4 \in G_1$
$q_3 \rightarrow$	$q_3 \in G_1$
$q_4 \rightarrow$	$q_5 \in G_1$
$q_5 \rightarrow$	$q_3 \in G_1$
$q_6 \rightarrow$	$q_6 \in G_1$

G_1	Símbolo **b**
$q_2 \rightarrow$	$q_2 \in G_1$
$q_3 \rightarrow$	$q_2 \in G_1$
$q_4 \rightarrow$	$q_6 \in G_1$
$q_5 \rightarrow$	$q_2 \in G_1$
$q_6 \rightarrow$	$q_6 \in G_1$

G_2	Símbolo **a**
$q_0 \rightarrow$	$q_1 \in G_2$
$q_1 \rightarrow$	$q_3 \in G_1$

G_2	Símbolo **b**
$q_0 \rightarrow$	$q_2 \in G_1$
$q_1 \rightarrow$	$q_2 \in G_1$

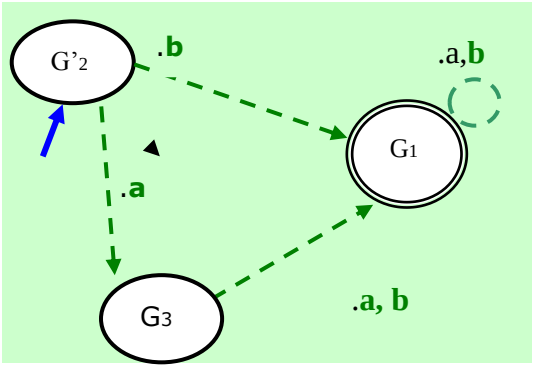
Vemos que todos los estados de:

- G_1 Son equivalentes
- G_2 No son equivalentes, luego, se deben separalos en dos elementos diferentes.

Finalmente para **a** y **b** la nueva partición: $\{G_1, G'_2, G_3\}$

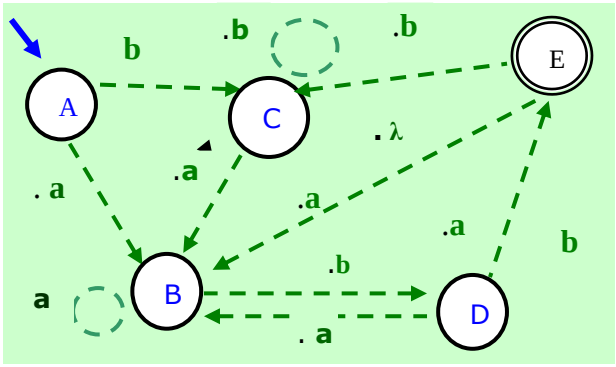
- $G_1 = \{ q_2, q_3, q_4, q_5, q_6 \}$
- $G'_2 = \{ q_0 \}$
- $G_3 = \{ q_1 \}$

Nuevo autómata mínimo:



AF_{DE} EJEMPLO 3 DE MINIMIZACIÓN

Inicio: Partición = { G_1, G_2}
G_1 = { E }
G_2 = {A,B,C,D}



- No se puede refinar el grupo G_1
- Analizar estados de G_2 con símbolos **a** y **b**

G_2	Símbolo a
A→	B ∈ G_2
B→	B ∈ G_2
C→	B ∈ G_2
D→	B ∈ G_2

G_2	Símbolo b
A→	C ∈ G_2
B→	D ∈ G_2
C→	C ∈ G_2
D→	E ∈ G_1

Notamos que **D** no es equvalente a los tres estados, analizando con el símbolo **b**, con los cuatro estados, luego se debe dividir en dos nuevos elementos:

Partición = { G_1, G_2', G_3} → G_1 = { E } → G_2' = {A,B,C} → G_3 = { D }

Se analizan los estados de G_2' con los símbolos **a** y **b**

G_2'	Símbolo a
A→	B ∈ G_2'
B→	B ∈ G_2'
C→	B ∈ G_2'

G_2'	Símbolo b
A→	C ∈ G_2'
B→	D ∈ G_3
C→	C ∈ G_2'

Notamos que **B** no es equvalente a los dos estados, luego se debe dividir G_2' en nuevos elementos:

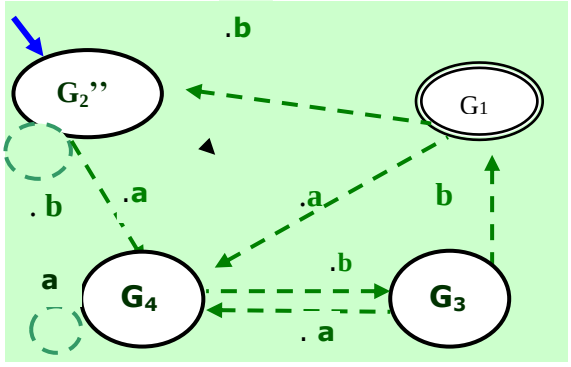
Partición = { G_1, G_2'', G_3, G_4} G_1 = { E } G_2' = {A,C}
G_3 = { D } G_4 = { B }

Se analizan los estados de G_2'' con los símbolos **a** y **b** G_2''

G_2''	Símbolo a
A→	B ∈ G_4
C→	B ∈ G_4

G_2''	Símbolo b
A→	C ∈ G_2''
C→	C ∈ G_2''

AF_{DE} minimizado:
Como ya no es posible seguir con la refinación, notamos que **B** y **C** son equivalentes, luego deben formar un mismo grupo.



A_F: Paso de un AF_{NDE} a un AF_{DE}

Ambos son equivalentes y poseen el mismo nivel computacional, por ende pueden resolver problemas iguales. El proceso genérico del paso de AF_{NDE} a AF_{DE} , (que no necesariamente es el mínimo, pudiendo tener hasta 2^Q estados) abarca:

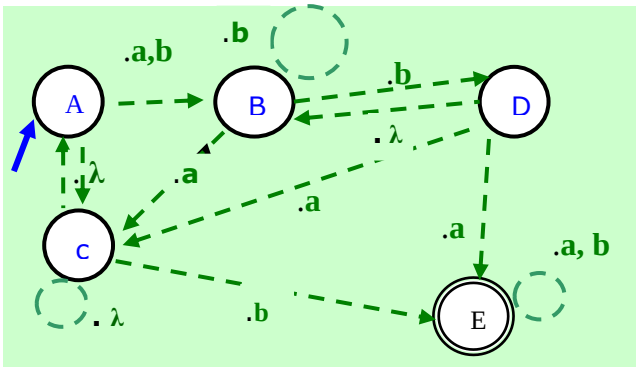
Partir de $AF_{NDE} = (\Sigma, Q, f, q_o F, T)$ cuyo equivalente sea: $AF_{DE} = (\Sigma, Q', f', q_o' F', T)$ donde:

1. $Q' = 2^Q$
2. $q_o' = \lambda\text{-clausura}(q_o)$
3. $F' = \{C \subseteq Q \mid C \cap F \neq \Phi\}$
4. $f'(C,a) = \{C' \subseteq Q \mid C' \cup_{q \in C} \lambda\text{-clausura}(f(q\text{-}a))\}$ donde: $C \subseteq Q$

Ejemplo: Calcular

- Estado inicial del AF_{NDE}

  $q_o = \lambda\text{-clausura}(A) = \{A, C\}$



- Función de transición del estado q_o

  $f(q_o,a) = \lambda\text{-clausura}(f(A,a) \cup f(C,a)) =$

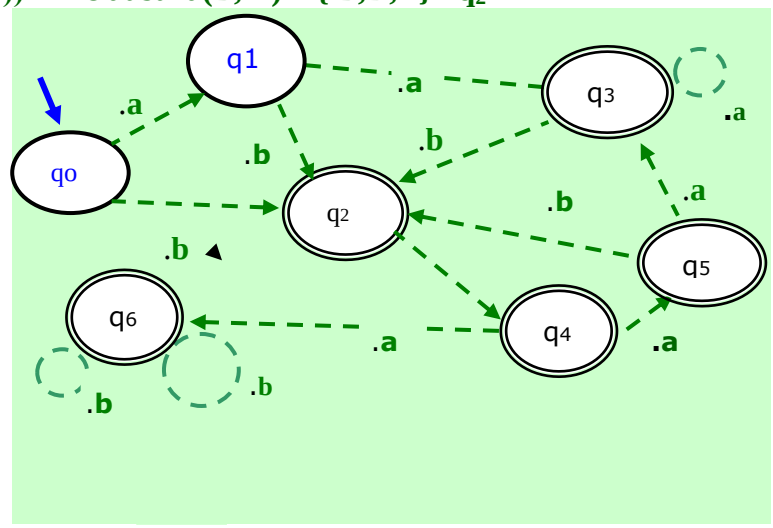
  $= \lambda\text{-clausura}(B,A) = \{A,B,C,D\} = q_1$

  $f(q_o,b) = \lambda\text{-clausura}(f(A,b) \cup f(C,b)) =$

  $= \lambda\text{-clausura}(B,E) = \{B,D,E\} = q_2$

- Luego calcular la función de transición para los nuevos estado que van apareciendo:

$f(q_1,a) = \lambda\text{-clausura}(f(A,a) \cup f(B,a) \cup f(C,a) \cup f(D,a)) = \lambda\text{-clausura}(B,C,A,E) = \{A,B,C,D,E\} = q_3$
$f(q_1,b) = \lambda\text{-clausura}(f(A,b) \cup f(B,b) \cup f(C,b) \cup f(D,b)) = \lambda\text{-clausura}(B, E) = \{B, D,E\} = q_2$
$f(q_2,a) = \lambda\text{-clausura}(f(B,a) \cup f(B,a) \cup f(D,a) \cup f(E,a)) = \lambda\text{-clausura}(C,E) = \{C, E\} = q_4$
$f(q_2,b) = \lambda\text{-clausura}(f(B,b) \cup f(D,b) \cup f(E,b)) = \lambda\text{-clausura}(B,E) = \{B,D,E\} = q_2$
$f(q_3,a) = \lambda\text{-clausura}(f(A,a) \cup f(B,a) \cup f(C,a) \cup f(D,a)) = \lambda\text{-clausura}(B,C,A,E) = \{A,B,C,D,E\} = q_3$
$f(q_3,b) = \lambda\text{-clausura}(f(A,b) \cup f(B,b) \cup f(C,b) \cup f(D,b)) = \lambda\text{-clausura}(B, E) = \{B, D,E\} = q_2$
$f(q_4,a) = \lambda\text{-clausura}(f(C,a) \cup f(E,a)) = \lambda\text{-clausura}(A,E) = \{A, C, E\} = q_5$
$f(q_4,b) = \lambda\text{-clausura}(f(C,b) \cup f(E,b)) = \lambda\text{-clausura}(E) = \{E\} = q_6$
$f(q_5,a) = \lambda\text{-clausura}(f(A,a) \cup f(C,a) \cup f(E,a)) = \lambda\text{-clausura}(B,A,E) = \{A,B,C,D,E\} = q_3$
$f(q_5,b) = \lambda\text{-clausura}(f(A,b) \cup f(C,b) \cup f(E,b)) = \lambda\text{-clausura}(B, E) = \{B,D,E\} = q_2$
$f(q_6,a) = \lambda\text{-clausura}(f(E,a)) =$
$\lambda\text{-clausura}(E) = q_6$
$f(q_6,b) = \lambda\text{-clausura}(f(E,b)) =$
$\lambda\text{-clausura}(E) = q_6$



El resultante es el AF_{DE} →

AF_{DE} CONVERSIÓN del AF_{NDE}-ε a un AF_{NDE}

Se basa en el concepto de **clausura-ε**, que es una clausura transitiva donde, un estado **q**, se llama **clausura-ε(q)** al conjunto de todos los estados a los que se puede acceder a partir de **q**, procesándose a lo más un único símbolo de la entrada. Definida recursivamente como:

- **Base inductiva**: Para todo estado **q, q ∈ clausura-ε(q)**.
- **Inducción**: Dados 2 estados **p** y **r**, si **p ∈ clausura-ε(q)** y r **∈ δ(p,ε)**, entonces **r ∈ clausura-ε(q)**.

ALGORITMO PARA ELIMINAR LAS TRANSICIONES VACÍAS

Descripción	Ejemplo

Calcular la **clausura-ε** del estado inicial, formándo un conjunto **A** que corresponderá al estado inicial del nuevo autómata.

- Para cada símbolo del alfabeto, verifico los estados alcanzables a partir de algún estado contenido en **A**, y calculo la **clausura-ε** de tales estados alcanzables.
- Si esas clausuras producen nuevos conjuntos distintos de **A**, estos serán nuevos estados a los que se accederá desde **A** y del símbolo correspondiente.
- Itero lo anterior para cada nuevo conjunto, hasta que no hayan transiciones posibles para ningún símbolo del alfabeto

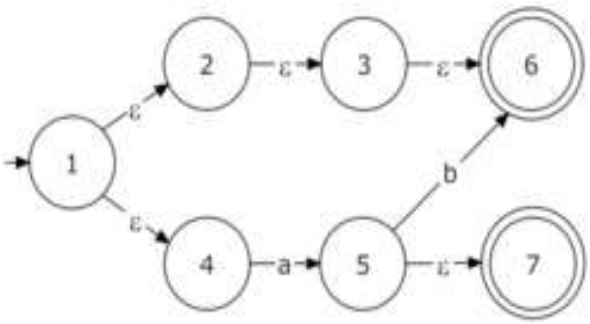
Se obtiene un AF_{DE}, como caso particular de AF_{NDE}.
En el ejemplo, se tendrá inicialmente:
clausura-ε(1) = {1,2,3,4,6} = A
Para A:
- Para el símbolo **a**: 4 va a 5, y **clausura-ε(5)** = {5,7} = B.
- Para el símbolo **b**: no existen transiciones posibles.

Para B:
- Para el símbolo **a**: no existen transiciones posibles.
- Para el símbolo **b**: 5 va a 6, y clausura-ε(6) = {6} = C.

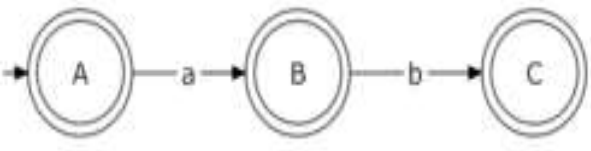
Para C:
- Para el símbolo **a**: no existen transiciones posibles.
- Para el símbolo **b**: no existen transiciones posibles.

Concluido el algoritmo, se obtiene el autómata:

Eliminación de las transiciones vacías de un AF_{NDE}-ε



AF_{NDE}-ε inicial.



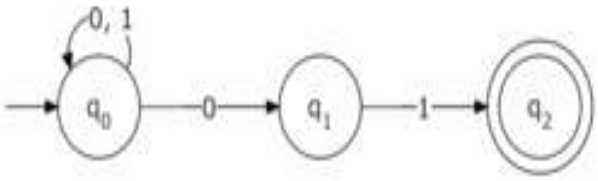
Puede ocurrir que al quitar las **transiciones ε** se obtenga directamente un AF_{DE}, pues la única razón de no-determinismo era la presencia de tales transiciones.

A_F: CONVERSIÓN de un AF_{NDE} a un AF_{DE}

El AF_{NDE} $(Q_N, \Sigma, q_0, \delta_N, F_N)$ puede convertirse en $AF_{DE}(Q_D, \Sigma, q_0, \delta_D, F_D)$ equivalente, que mantiene el alfabeto Σ y el estado inicial q_0 originales.
La conversión requiere pasar por un AF_{DE} intermedio con estados y transiciones redundantes, que al no ser accesibles a partir del estado inicial, son eliminados para obtener el AF_{DE} definitivo.

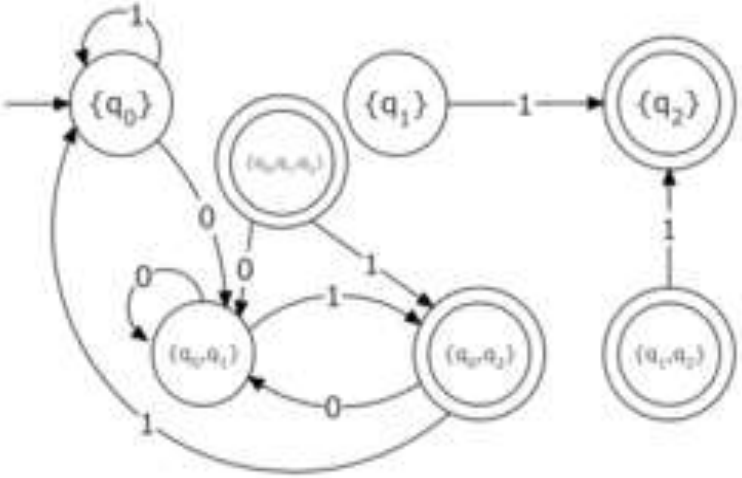
Pasos para definir el AF_{DE}:	Ejemplo

- Redefinir el conjunto de estados $Q_N=\{q_0,q_1, .., q_m\}$ original, como uno conformado por los subconjuntos de Q_N.
  Los nuevos estados finales serán los que contengan a alguno de los estados finales originales.



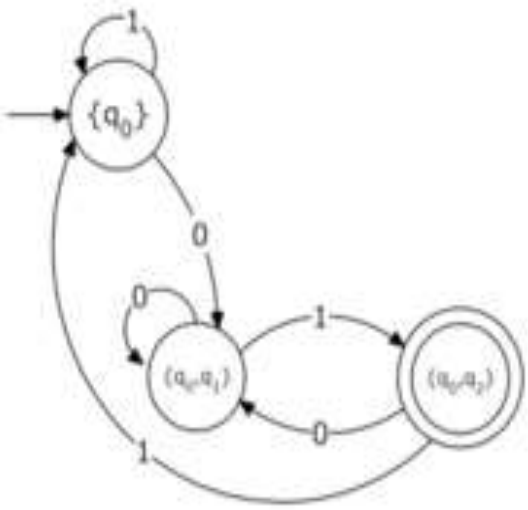
- Redefinir transiciones originales, por otras del tipo $\delta_D(S, a)$

  Donde:
  $a \in \Sigma$, y **S** es ∪ de estados **q** de Q_N donde existía transición $\delta_N(q,a)$.



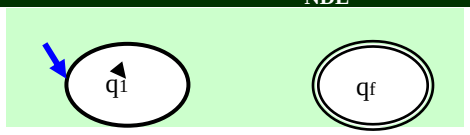
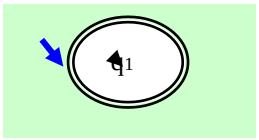
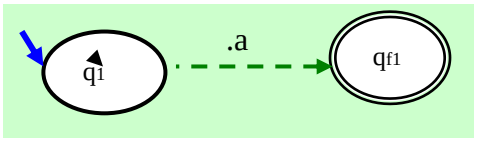
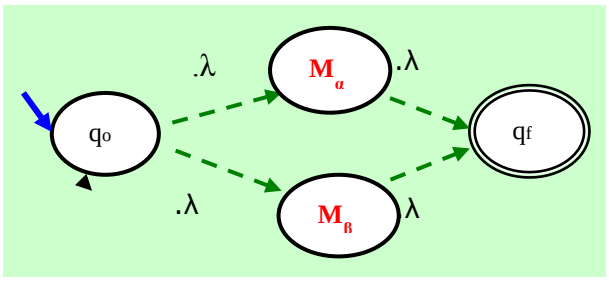
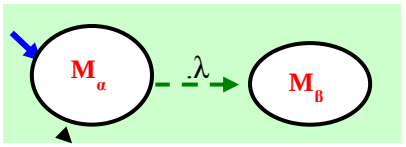
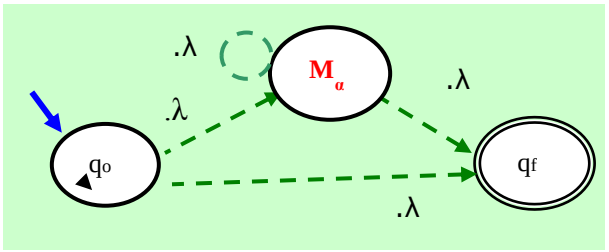
- Eliminar estados inaccesibles (junto con sus transiciones de salida), aquellos que no se puede acceder a partir del estado inicial. Luego de esta depuración, se obtiene el final.

Ejemplo:
Si el AF_{DE} inicial posee 3 estados (q_0, q_1, q_2), el AF_{DE} intermedio poseerá 7:
$(\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{q_0,q_2\},\{q_1,q_2\},\{q_0,q_1,q_2\})$.
Como el estado final original era q_2, los estados finales del AF_{DE} intermedio son:
$\{q_2\}$, $\{q_0, q_2\}$, $\{q_1, q_2\}$ y $\{q_0, q_1, q_2\}$.
En las nuevas transiciones, se mantuvo
$\delta_N(q_0,1)= q_0$,
Ahora llamada: $\delta_D(\{q_0\},1)=\{q_0\}$
Las originales: $\delta_N(q_0,0)=q_0$ y $\delta_N(q_0,0)=q_1$,
se reemplazo por: $\delta_D(\{q_0\},0)=\{ q_0, q_1\}$.



Se eliminan $\{q_1\}$, $\{q_2\}$ y $\{q_1,q_2\}$ no conectados al resto del autómata que posee el estado inicial.
Se elimina $\{q_0, q_1, q_2\}$, a pesar de estar conectado al resto del autómata, no es accesible a partir de $\{q_0\}$.
Finalmente, eliminando estos cuatro estados y sus respectivas transiciones, se obtiene el AF_{DE} buscado

A_F: CONSTRUCCIÓN de un AF_{NDE} que reconoce una E_{RE}

E_{RE} paso a un AF_{NDE}	E_{RE}	AF_{NDE}
Aplicando el concepto recursivo de la E_{RE}: Cada tipo de E_{RE} tiene un AF_{NDE} Luego podemos ensamblar cada uno de estos AF_{NDE} para lograr otro más complejo: Esta es una tabla M_α y M_β autómatas que reconocen las E_{RE} de α y β	$E_{RE} = \varphi$	
	$E_{RE} = \lambda$	
	$E_{RE}=\alpha\in\Sigma$	 ← M_α
	$E_{RE} =\alpha+\beta$	
	$E_{RE} =\alpha.\beta$	
	$E_{RE} =\alpha^n$	

AF_{DE} PASO de una E_{RE} a un AF_{DE}: Sea la E_{RE}= **(a+b) *abb**

Etiquemos (posiciones) con un número cada símbolo de la E_{RE} y usamos # a la derecha de la E_{RE} para indicar el final de las palabras del lenguaje representado

(	a	+	b	)	*	a	b	b	#
1		2				3	4	5	6

Para cada posición definimos su conjunto siguiente que estará formado por las posiciones que a una en cualquier palabra que pertenezca al lenguaje representado por la E_{RE}.

Para calcular estos conjuntos analizamos cada operación que interviene en la E_{RE} y vemos como afecta a las distintas posiciones. Los conjuntos en nuestro caso son:

Sig(1)={1,2,3} **Sig(3)={4}** **Sig(5)={6}**
Sig(2)={1,2,3} **Sig(4)={5}** **Sig(6)=φ**

Cada estado del autómata es un conjunto de posiciones. Los estados finales de aceptación contienen a la posición asociada # , para nosotros la posición 6

Calculamos simultáneamente los estados y las transiciones mediante el siguiente algoritmo, donde se tiene en cuenta que **simb(i)** indica el símbolo del alfabeto asociado a la posición i.

Sig(1)={a} **Sig(3)={a}** **Sig(5)={b}**
Sig(2)={b} **Sig(4)={b}** **Sig(6)=φ**

Si **pp** son las primeras posiciones de la E_{RE}, para nuestro ejemplo seria: **pp={1,2,3}**

Las siglas:

EM (Estados Marcados: Con transiciones que parten de dicho estado)

ENM (Estados No Marcados) son conjuntos del autómata que se construye, cuyo proceso termina cuando no queda ningún estado por marcar.

Aplicando este algoritmo, comenzamos por el estado inicial q_o= **pp ={1,2,3}**

ALGORITMO DE CONSTRUCCIÓN del AF_{DE} a partir de una E_{RE}

```
Input: x∈Σ*
Begin
        ENM = pp
        EM = θ
While ENM = θ do
        Pasar T desde ENM hasta EM
        for all a∈Σ do
        R = U∀i∈T Sig(i) tal que simb(i) = a
                if R ≠ θ and R no∈(EM U ENM) then
                Añadir R a ENM
                f(t,a) = R
                endif
        endfor
end while
```

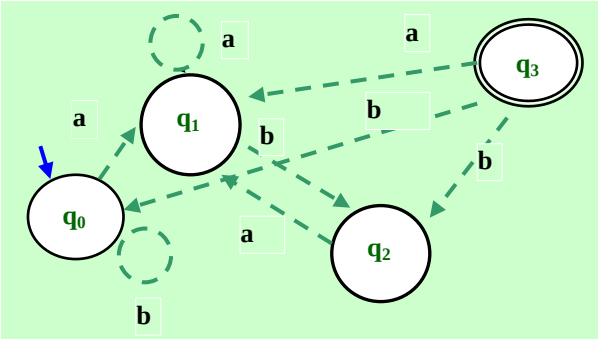
Inicialmente EM = θ y ENM = { q_o } luego para este estado, la función de transición es:

- **$f(q_o,a)$** = **sig(1)** U **sig(3)** ={1,2,3} =q_1
- **$f(q_o,b)$** = **sig(2)** ={1,2,3} = q_0 Luego **EM = { q_o }, EMN = { q_1 }**

Seguimos calculando la función de transición de los nuevos estados que van apareciendo

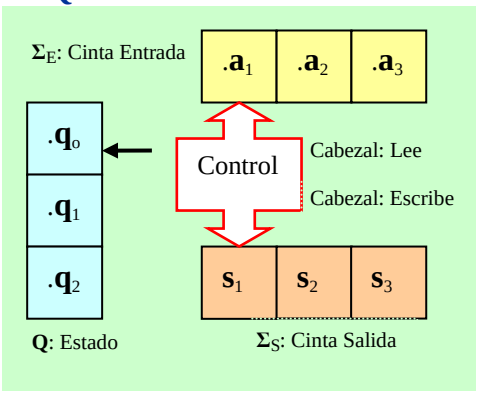
- **$f(q_1,a)$** = **sig(1)** U **sig(3)** ={1,2,3,4} =q_1
- **$f(q_1,b)$** = **sig(2)** U **sig(4)** ={1,2,3,5} = q_2
  Luego **EM = { q_o, q_1,}, EMN = { q_2 }**
- **$f(q_2,a)$** = **sig(1)** U **sig(3)** ={1,2,3,4} =q_1
- **$f(q_2,b)$** = **sig(2)** U **sig(5)** ={1,2,3,6} = q_3
  Luego **EM = { q_o, q_1, q_2,}, EMN = { q_3 }**
- **$f(q_3,a)$** = **sig(1)** U **sig(3)** ={1,2,3,4} =q_1
- **$f(q_3,b)$** = **sig(2)** ={1,2,3} = q_0
  Luego **EM = { q_o, q_1, q_2, q_3}, EMN = θ**

El AF_{DE} es:



MS: MAQUINAS SECUENCIALES:

ARQUITECTURA



Generan a partir de una palabra de entrada registrada en una cinta de entrada, otra palabra de salida escrita sobre otra cinta de salida; usando para ello un conjunto de estados que memorizan en cada momento la parte leída de la palabra de entrada y generan simultáneamente una salida.

Los tipos de máquinas secuenciales pueden ser:

- MAQUINA FINITA DE **MEALY**
- MAQUINA FINITA DE **MOORE**

DEFINICION: MS = (Σ_E, Σ_S, Q, f, g)

Componente	Descripción
Σ_E Alfabeto Símbolos de Entrada	Registrado sobre una cinta de entrada
Σ_S Alfabeto Símbolos de Salida	Escrita sobre una cinta de salida
Q Estados	Conjunto finito no vacío de estados de la máquina
f Función Transición de Estado $f: Q \times \Sigma_E \rightarrow Q$ f (EstadoActual,SímboloLeído) = Nuevo Estado	Define el paso del estado actual al siguiente estado. Ejemplo: $.f(q_o, a_j) = q_o$ $.f(q_o, a_j) = q_1$ $.f(q_1, a_j) = q_1$ $.f(q_1, a_j) = q_o$
g Función Salida	Determina Σ_S : Símbolo a escribir sobre la cinta de salida

MEALY: M_{ME}	MOORE: M_{MO}
g: $Q \times \Sigma_E \rightarrow \Sigma_S$ g(EstadoActual, SímboloLeído) = SímboloSalida	g: $Q \rightarrow \Sigma_S$ g(EstadoActual) = Símbolo deSalida
Ejemplo: $.g(q_o, a_j) = s_1$ $g(q_o, a_j) = s_1$ $.g(q_1, a_j) = s_2$ $g(q_1, a_j) = s_2$	Ejemplo: $.g(q_1) = s_2$ $g(q_o) = s_1$

Sea la máquina: $M_{ME} = (\Sigma_E, \Sigma_S, Q, f, g) = (\{ 0,1 \}, \{ p, i \}, \{ q_o, q_1 \}, f, g)$; Donde:

Las formas para representar las funciones | Q | x | Σ_E | de **f** y **g** son:

1. **Pares:**

Transición:
Pares:
.f(q_o,0) = q_o
.f(q_o,1) = q_1
.f(q_1,0) = q_1
.f(q_1,1) = q_o

Salida:
Pares:
.g(q_o,0) = p
.g(q_o,1) = i
.g(q_1,0) = i
.g(q_1,1) = λ

Transición y Salida
Pares:
.f(q_o,0) = q_o| p
.f(q_o,1) = q_1| i
.f(q_1,0) = q_1| i
.f(q_1,1) = q_o | λ

2. **Matriz:**

Matriz:

f/ Σ_E	0	1
.q_o	.q_o	.q_1
.q_1	.q_1	.q_o

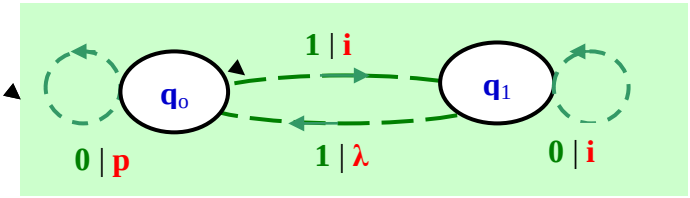
Matriz:

g/ Σ_E	0	1
.q_o	.p	.i
.q_1	.i	.λ

Matriz:

f|g / Σ_E	0	1
.q_o	.q_o | p	.q_1 | i
.q_1	.q_1 | i	.q_o |.λ

3. **Diagrama de transición:**



Elementos del Grafo Dirigido:

- **Nodos** simbolizan sus estados: $q_i \in Q$
- **Arcos** simbolizan sus transiciones:

- Para f (q_o, 0) = q_o y g (q_o, 0) = p se crea un arco de q_o a q_o etiquetado con 0 | p
- Para f (q_o, 1) = q_1 y g (q_o, 1) = i se crea un arco de q_o a q_1 etiquetado con 1 | i
- Para f (q_1, 0) = q_1 y g (q_1, 0) = i se crea un arco de q_1 a q_1 etiquetado con 0 | i
- Para f (q_1, 1) = q_o y g (q_1, 1) = λ se crea un arco de q_1 a q_o etiquetado con 1 | λ

Desde el estado q_o el proceso de la palabra **01001**, generará las siguientes transiciones y salidas:

Estado Inicial	Transición 1	Transición 2	Transición 3	Estado Final
q_o | 0 1 0 0 1 | p	q_o | 0 1 0 0 1 | pi	q_1 | 0 1 0 0 1 | pii	q_1 | 0 1 0 0 1 | piii	q_1 | 0 1 0 0 1 | piii λ
Próxima:				
Transición: f (q_o, 0) = q_o	f (q_o, 1) = q_1	f (q_1, 0) = q_1	f (q_1, 0) = q_1	f (q_1, 1) = q_o
Salida: g (q_o, 0) = p	g (q_o, 1) = i	g (q_1, 0) = i	g (q_1, 0) = i	g (q_1, 1) = λ

EXTENSION DE MEALY: Genera de una sola vez todos los caracteres de la cinta de salida, correspondientes a todos los símbolos que fueron leídos en la cinta de entrada.
Para esto, se replantea una extensión de la anterior definición: **ME = (Σ_E, Σ_S, Q, f, g)**, cuya:

- **Función de transición**: **f: Q x Σ_E*→Q** será:
  - **.f (q, a x) = f (f (q, a), x)** para cada **q ∈ Q, a ∈ Σ_E, x ∈ Σ_E***
  - **.f (q, λ) = q** para cada **q∈Q**

- **Función de salida**: **g: Q x Σ_E*→ Σ_S*** Las anteriores salidas se suman las siguientes salidas de las palabras de longitud cero o mayor que uno.
  - **.g (q, a x) = g (q, a) . g (f (q, a), x)** para cada **q∈Q, a∈Σ_E, x∈Σ_E***
  - **.g (q, λ) = λ** para cada **q∈Q**

  La salida generada corresponde a concatenar la salida producida con cada uno de los símbolos de entrada y sus transiciones.

La máquina del ejemplo anterior, empieza en el estado q_o para procesar la palabra de entrada **01001**, la **máquina de Mealy**, generó las siguientes transiciones y salidas:

Estado Inicial	Transición 1	Transición 2	Transición 3	Estado Final
q_o | 0 1 0 0 1 | p	q_o | 0 1 0 0 1 | pi	q_1 | 0 1 0 0 1 | pii	q_1 | 0 1 0 0 1 | piii	q_1 | 0 1 0 0 1 | piii λ
Próxima:				
Transición: f (q_o, 0) = q_o	f (q_o, 1) = q_1	f (q_1, 0) = q_1	f (q_1, 0) = q_1	f (q_1, 1) = q_o
Salida: g (q_o, 0) = p	g (q_o, 1) = i	g (q_1, 0) = i	g (q_1, 0) = i	g (q_1, 1) = λ

Aplicando la extensión de Mealy, se generarían las siguientes transiciones y salidas:

Estado	Inicial	Transición 1	Transición 2	Transición 3	Estado final
Cabezal en cinta entrada	01001	1001	001	01	1
FunciónEntrada	f(q_o,0) = q_o	f(q_o,1) = q_1	f(q_1,0) = q_1	f(q_1,0) = q_1	f(q_1,1) = q_0
FunciónSalida	g(q_o,0) = p	g(q_o,1) = i	g(q_1,0) = i	g(q_1,0) = i	g(q_1,0) = i
Transición	f(q_o,01001)	f(q_o,1001)	f(q_1,001)	f(q_1,01)	f(q_1,1)
Salida	g(q_o,0)	.p g(q_o,1)	.p i g(q_1,0)	.p i i g(q_1,01)	.p i i i g(q_1,1)
Cinta Salida	p	p i	p i i	p i i i	p i i i λ

En detalle la extensión de Mealy, se generarían las siguientes transiciones y salidas:

Estado	Inicial	Transición 1			Transición 2
Transición	.f (q_o, 0100)	f(f (q_o, 0),100)	f (q_o, 100)	.f(f (q_o, 1),00)	.f (q_1, 00)
Salida	.g (q_o, 0100)	g(q_o,0) g(f(q_o,0),100)	p g(q_o,100)	pg(q_o,1) g(f (q_o,1),00)	.pig (q_1, 00)

	Transición 2		Transición 3		Final
Transición	.f(f (q_1, 0),0)	f (q_1, 0)	.f(f (q_1, 0),λ)	f (q_1, λ)	. q_1
Salida	pi g(q_1, 0)g(f(q_1,0),0)	pii g(q_1, 0)	pii g(q_1,0)g(f(q_1, 0), λ)	piii g(q_1, λ)	.piii

O de manera más simple:

Estado	Inicial	Transición 1	Transición 2	Transición 3	Estado final
Transición	f(q_o,01001)	f(q_o,1001)	f(q_1,001)	f(q_1,01)	f(q_1,1)
Salida	g(q_o,0)	.p g(q_o,1)	.p i g(q_1,0)	.p i i g(q_1,01)	.p i i i g(q_1,1)

Sea la máquina: $M_{MO} = (\Sigma_E, \Sigma_S, Q, f, g) = (\{ 0,1 \}, \{ p, i \}, \{ q_o, q_1 \}, f, g)$; Donde:

Las formas para representar las funciones | Q | x | Σ_E | de **f** y **g** son:

Transición:
Pares:
.f(q_o,0) = q_o
.f(q_o,1) = q_1
.f(q_1,0) = q_1
.f(q_1,1) = …

Matriz:

f/ Σ_E	0	1
.q_o	.q_o	.q_1
.q_1	.q_1	

Salida:
Pares:
.g(q_o) = p
.g(q_1) = i

Matriz:

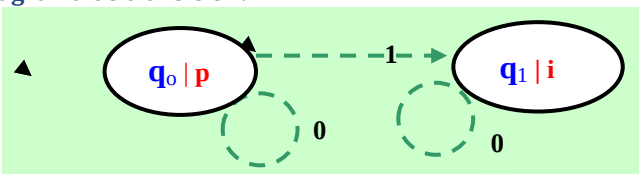
g/ Σ_E		
.q_o	.p	
.q_1	.i	

Transición y Salida
Pares:
.f(q_o,0) = q_o| p
.f(q_o,1) = q_1| i
.f(q_1,0) = q_1| i
.f(q_1,1) = q_o| λ

Matriz:

f|g / Σ_E	0	1
. q_o | p	.q_o	.q_1
. q_1 | i	.q_1	

Diagrama de transición:



Elementos del Grafo Dirigido:

Nodos sus estados: $q_i \in Q$:
Para **g (q_i) = s_j**, se crea un nodo etiquetado con **q_i | s_j**

- Para **g (q_o)** = p, se crea un nodo etiquetado con q_o | p
- Para **g (q_1)** = i, se crea un nodo etiquetado con q_1 | i

Arcos sus transiciones: Para f (q_i, a_j) = q_k, se crea un arco de q_i a q_k etiquetado con a_j

- Para f (q_o, 0) = q_o se crea un arco de q_o a q_o etiquetado con 0
- Para f (q_o, 1) = q_1 se crea un arco de q_o a q_1 etiquetado con 1
- Para f (q_1, 1) = q_1 se crea un arco de q_1 a q_1 etiquetado con 0

Empezando en q_o el proceso de la palabra **01001**, generará las siguientes transiciones y salidas:

Estado Inicial	Transición 1	Transición 2	Transición 3	Estado Final
q_o | 0 1 0 0 1| p	q_o | 0 1 0 0 1| pi	q_1 | 0 1 0 0 1| pii	q_1 | 0 1 0 0 1| piii	q_1 | 0 1 0 0 1 | piiiλ
Próxima:				
Transición: f (q_o, 0) = q_o	f (q_o, 1) = q_1	f (q_1, 0) = q_1	f (q_1, 0) = q_1	f (q_1, 1) = …
Salida: g (q_o) = p	g (q_1) = i	g (q_1) = i	g (q_1) = i	g (..) = λ

EXTENSION DE MOORE: Genera de una sola vez todos los caracteres de la cinta de salida, correspondientes a todos los símbolos que fueron leídos en la cinta de entrada.
Para este objetivo se replantea una extensión de la anterior definición: **MO = (Σ_E, Σ_S, Q, f, g)**

- **Función de Transición** f (Estado Actual, Símbolo Leído)= Nuevo Estado: **f: Q x Σ_E *→Q** será:
  - **.f (q, a x) = f (f (q, a), x)** para cada **q ∈ Q, a ∈ Σ_E, x ∈Σ_E***
  - **.f (q, λ) = q** para cada **q∈ Q**
  - Para una palabra formada por varios símbolos, el estado al que transita la máquina corresponde a transitar con cada uno de los símbolos.

- **Función de Salida** g (Estado Actual, Símbolo Leído)= Símbolo a Escribir, será: **g: Q x Σ_E*→ Σ_S*** y a las anteriores salidas se suman las siguientes salidas de las palabras de longitud cero o mayor que uno. A esta función se agrega una nueva función de salida: **g': Q x Σ_E*→ Σ_S*** Y se cumple que:
  - **.g' (q, a x) = g (q) . g' (f (q, a), x)** para cada **q ∈ Q, a ∈ Σ_E, x ∈ Σ_E***
  - **.g' (q, λ) = λ** para cada **q ∈ Q**

La primera salida es del estado siguiente al estado inicial y el primer símbolo de entrada. Luego en cada salida generada, se concatena la salida producida con cada símbolo de entrada y sus transiciones.

En el ejemplo anterior, la máquina de Moore empieza en el estado q_o para procesar la palabra de entrada **01001**, se generaban las siguientes transiciones y salidas:

	Estado Inicial	Transición 1	Transición 2	Transición 3	Estado Final
	q_o | **0** 1 0 0 1| **p**	q_o | 0 **1** 0 0 1| **pi**	q_1 | 0 1 **0** 0 1| **pii**	q_1 | 0 1 0 **0** 1| **piii**	q_1 | 0 1 0 0 **1** | **piiiλ**
Próxima:					
Transición:	f (q_o, **0**) = q_o	f (q_o, **1**) = q_1	f (q_1, **0**) = q_1	f (q_1, **0**) = q_1	f (q_1, **1**) = …
Salida:	g (**q_o**) = **p**	g (**q_1**) = **i**	g (**q_1**) = **i**	g (**q_1**) = **i**	g (..) = **λ**

Aplicando la extención de Moore, se generan las siguientes transiciones y salidas:

	Estado Inicial	Paso 1	Paso 2	Paso 3	Paso 4
Transición	.f (**q_o**, **0100**)	f(**f (q_o, 0)**,**100**)	f (**q_o**, **100**)	.f(**f (q_o, 1)**,**00**)	f (**q_1**, **00**)
Salida	.g' (**q_o**)	g(**q_o**) g'(**f(q_o, 0)**,**100**)	**p** g'(**q_o** ,**100**)	**p**g(q_1)g'(**f(q_o, 1)**,**00**)	**pi**g' (**q_1**, **00**)

	Paso 5	Paso 6	Paso 7	Paso 8	Estado Final
Transición	.f(**f (q_1, 0)**,**0**)	F (**q_1**, **0**)	.f(f **(q_1, 0)**,**λ**)	f (**q_1**, **λ**)	. **q_1**
Salida	**pi**g (**q_1**)g'(**f(q_1, 0)**,**0**)	**pii**g' (**q_1**, **0**)	**pii**g (**q_1**)g'(**f(q_1, 0)**, **λ**)	**piii** g'(**q_1**, **λ**)	**.piii**

O de manera más simple:

Estado	Inicial	Transición 1	Transición 2	Transición 3	Estado final
Transición	f(**q_o**,01001)	f(**q_o**,1001)	f(**q_1**,001)	f(**q_1**,01)	f(**q_1**,**1**)
Salida	**p** g(**q_o**)	**.pi** g(**q_1**)	**.pii** g(**q_1**)	**.piii** g(**q_1**)	**.p i i i λ** g(**…**)

EQUIVALENCIA las MAQUINAS MEALY y MOORE

Una máquina de Mealy puede transformarse en otra equivalente de Moore y viceversa.

TRANSFORMACION de MEALY a MOORE:

La máquina Mealy: **ME = (Σ_E, Σ_S, Q, f, g)**
Se transforma a otra equivalente de Moore: **MO = (Σ_E, Σ_S, Q', f', g')**
Cuando por cada transición y salida: **f(q,a) = p, g(q,a) = b, (q, p∈Q, a∈Σ_E, b∈Σ_S)**
Se crea:

- Un estado $p^b \in Q'$ cuya función de salida será $g'(p^b) = b$
- Una transición $f'(q^s, a) = p^b$ para cualquier estado $q^S \in \Sigma_S$

Si a un determinado estado $p \in Q$ no llega ninguna transición, se crea un nodo etiquetado con q^λ

Para la máquina **Mealy**: **ME' = ({ 0,1 }, { a, b, c }, { e_o, e_1, e_2 }, f', g')**, definimos la máquina equivalente de **Moore**: **MO = ({ 0,1 }, { a, b, c }, Q', f', g'),**
Donde:

.$f(c_o, 0)= c_o$, $g(c_o, 0)= a$, luego: $c_o^a \in Q'$, $g'(c_o^a)=a$, $f'(c_o^a, 0) = c_o^a$ y $f'(c_o^b, 0) = c_o^a$

.$f(c_o, 1)= c_2$, $g(c_o, 1)= a$, luego: $c_2^b \in Q'$, $g'(c_2^b)=a$, $f'(c_o^a, 1) = c_2^b$ y $f'(c_o^b, 1) = c_2^b$

.$f(c_1, 0)= c_o$, $g(c_1, 0)= a$, luego: $c_o^a \in Q'$, $g'(c_o^a)=a$, $f'(c_1^a, 0) = c_o^a$ y $f'(c_1^b, 0) = c_o^a$

.$f(c_1, 1)= c_1$, $g(c_1, 1)= a$, luego: $c_1^b \in Q'$, $g'(c_1^b)=a$, $f'(c_1^a, 1) = c_1^b$ y $f'(c_1^b, 1) = c_1^b$

.$f(c_2, 0)= c_o$, $g(c_2, 0)= a$, luego: $c_o^b \in Q'$, $g'(c_o^b)=a$, $f'(c_2^a, 0) = c_o^b$ y $f'(c_2^b, 0) = c_o^b$

.$f(c_2, 1)= c_1$, $g(c_2, 1)= a$, luego: $c_1^a \in Q'$, $g'(c_1^a)=a$, $f'(c_2^a, 1) = c_1^a$ y $f'(c_2^b, 1) = c_1^a$

Luego **Q' = {c_o^a, c_o^b, c_1^a, c_1^b, c_2^b}** como el estado c_2^a no se creó, se eliminan sus transiciones.

TRANSFORMACION de MEALY a MOORE.

Ejemplo: En las siguientes tablas de transición y salida, correspondientes a maquinas de Mealy con alfabeto de: Entrada Σ_E = { a, b } y Salida Σ_S ={ 0,1}

.f	.a	.b
q_o	q_o	q_1
q_1	q_o	q_1

g	a	b
q_o	0	1
q_1	1	0

Se define la maquina de Moore equivalente como: **MO = ({ a,b },{ 0,1},Q' , f' , g')**

- $f_{ME}(q_o,a) = q_o$, $g_{ME}(q_o, a) = 0$, luego q_o^0 Q'
  $g'_{MO}(q_o^0) = 0$, $f'_{MO}= (q_o^0,a)=q_o^0$, $f'_{MO}= (q_o^1,a) = q_o^0$
- $f_{ME}(q_o,b) = q_1$, $g_{ME}(q_o, b) = 1$, luego q_1^1 Q' ,
  $g'_{MO}(q_1^1) = 1$, $f'_{MO}=(q_o^0,b) = q_1^1$, $f'_{MO}=(q_o^1,b) = q_1^1$
- $f_{ME}(q_1, a) = q0$, $g_{ME}(q_1, a) = 1$, luego q_o^1 Q' ,
  $g'_{MO}(q_o^1) = 1$, $f'_{MO}=(q_1^0,a) = q_o^1$, $f'_{MO}= (q_1^1,a) = q_o^1$
- $f_{ME}(q_1, b) = q_1$, $g_{ME}(q_1, b) = 0$, luego q_1^0 Q' ,
  $g'_{MO}(q_1^0) = 0$, $f'_{MO}= (q_1^0,b)= q_1^0$, $f'_{MO}=(q_1^1, b) = q_1^0$

Q / ΣS	A	B
q_o^0	$q_o^0/0$	$q_1^1/1$
q_o^1	$q_o^0/0$	$q_1^1/1$
q_1^0	$q_o^1/1$	$q_1^0/0$

Por tanto $\mathbf{Q' = \{ q_o^0 , q_o^1 , q_1^0 , q_1^1 \}}$

Ejemplo: Se define la maquina de Moore equivalente como: **MO = ({ a,b },{ 0,1},Q' , f' , g')**

.f	.a	.b
$\mathbf{q_o}$	$\mathbf{q_1}$	$\mathbf{q_1}$
$\mathbf{q_1}$	$\mathbf{q_1}$	$\mathbf{q_1}$

g	a	b
$\mathbf{q_o}$	0	0
$\mathbf{q_1}$	0	1

- $f_{ME}(\mathbf{q_o}, a) = \mathbf{q_1}$, $g_{ME}(\mathbf{q_o}, a) = 0$, luego $\mathbf{q_1^0}$ Q' ,
  $g'_{MO}(\mathbf{q_1^0}) = 0$, $f'_{MO} = (\mathbf{q_o^0}, a) = \mathbf{q_1^0}$, $f'_{MO} = (\mathbf{q_o^1}, a) = \mathbf{q_1^0}$
- $f_{ME}(\mathbf{q_o}, b) = \mathbf{q_1}$, $g_{ME}(\mathbf{q_o}, b) = 0$, luego $\mathbf{q_1^0}$ Q' ,
  $g'_{MO}(\mathbf{q_1^0}) = 0$, $f'_{MO} = (\mathbf{q_o^0}, b) = \mathbf{q_1^0}$, $f'_{MO} = (\mathbf{q_o^1}, b) = \mathbf{q_1^0}$
- $f_{ME}(\mathbf{q_1}, a) = \mathbf{q_1}$, $g_{ME}(\mathbf{q_1}, a) = 0$, luego $\mathbf{q_1^0}$ Q' ,
  $g'_{MO}(q1^0) = 0$, $f'_{MO} = (\mathbf{q_1^0}, a) = \mathbf{q_1^0}$, $f'_{MO} = (\mathbf{q_1^1}, a) = \mathbf{q_1^0}$
- $f_{ME}(\mathbf{q_1}, b) = \mathbf{q_1}$, $g_{ME}(\mathbf{q_1}, b) = 1$, luego $\mathbf{q_1^1}$ Q' ,
  $g'_{MO}(\mathbf{q_1^1}) = 1$, $f'_{MO} = (\mathbf{q_1^0}, b) = \mathbf{q_1^1}$, $f'_{MO} = (\mathbf{q_1^1}, b) = \mathbf{q_1^1}$

Por lo tanto Q' = { $\mathbf{q_1^0}$, $\mathbf{q_1^1}$ }

Q / ΣS	a	b
$\mathbf{q_o^0}/0$	$\mathbf{q_1^0}$	$\mathbf{q_1^0}$
$\mathbf{q_o^1}/1$	$\mathbf{q_1^0}$	$\mathbf{q_1^0}$
$\mathbf{q_1^0}/0$	$\mathbf{q_1^0}$	$\mathbf{q_1^1}$
$\mathbf{q_1^1}/1$	$\mathbf{q_1^0}$	$\mathbf{q_1^1}$

Luego de la eliminación de [$\mathbf{q_o^1}$, $\mathbf{q_o^0}$] porque no llega ninguna transición , será :

Q / ΣS	a	b
Q_0^{λ}	$\mathbf{q_1^0}$	$\mathbf{q_1^0}$
$\mathbf{Q_1^0}/0$	$\mathbf{q_1^0}$	$\mathbf{q_1^1}$
$\mathbf{Q_1^1}/1$	$\mathbf{q_1^0}$	$\mathbf{q_1^1}$

TRANSFORMACION de MOORE a MEALY:

Una máquina Moore **MO = (Σ_E, Σ_S,Q, f, g)** se transformará a otra máquina equivalente de Mealy
ME = (Σ_E, Σ_S, Q, f, g')
Haciendo cumplir por cada transición y salida

.f(q, a) = p, g (q) = b, (q, p∈Q, a∈ Σ_E, b∈ Σ_S) se define la función de salida **g'(q, a) = b**

Ejemplo: TRANSFORMACION de MOORE a MEALY
Para la máquina Moore **MO = (Σ_E, Σ_S, Q, f, g)**, definida como:
MO = ({ 0,1 }, { p, i }, { q_o, q_1 }, f, g),
La máquina equivalente de Mealy se la define como: **ME = ({ 0,1 }, { p, i }, { q_o, q_1 }, f, g')**
Donde:

- **.f(q_o, 0) = q_o, g(q_o) = p,** luego **g'(q_o, 0) = p**
- **.f(q_o, 1) = q_1, g(q_1) = i,** luego **g'(q_o, 1) = i**
- **.f(q_1, 0) = q_o, g(q_1) = i,** luego **g'(q_1, 0) = i**
- **.f(q_1, 1) = q_o, g(q_o) = p,** luego **g'(q_1, 1) = p**

Ejemplo: TRANSFORMACION de MOORE a MEALY
Dadas las siguientes tablas de transición y salida , correspondientes a maquinas de Moore con alfabeto de entrada Σ_E = { a,b } y de salida ΣS ={ 0,1} .

	A	b	Salida
Q_0	**q_1**	**q_2**	1
Q1	**q_1**	**q_1**	0
Q_2	**q_1**	**q_o**	1

Se define la maquina de Mealy equivalente como : MO = ({ a,b },{ 0,1},{ q_0, q1, q_2} , f , g')

- f (**q_o** , a) = q1, g (**q_o**, a) = 0, Entonces g' (**q_o**, a) = 0
- f (**q_o** , b) = q_2, g (**q_o**, b) = 1, Entonces g' (**q_o**, b) = 1
- f (q1 , a) = q1, g (**q_1**, a) = 0, Entonces g' (**q_1**, a) = 0
- f (q1 , b) = q1, g (**q_1**, b) = 0, Entonces g' (**q_1**, b) = 0
- f (q_2 , a) = q1, g (q_2 , a) = 0, Entonces g' (q_2, a) = 0
- f (q_2 , b) = q0, g (q_2, b) = 1, Entonces g' (q_2, b) = 1

F	A	B
q_o	**q_1**	q_2
q_1	**q_1**	**q_1**
q_2	**q_1**	q_0

G	A	b
q_o	0	1
q_1	0	0
q_2	0	1

Ejemplo: **TRANSFORMACION de MOORE a MEALY:**

	A	B	Salida
Q_0	q_0	q_2	0
Q1	q1	q_0	0
Q_2	q_2	q1	1

Se define la maquina de Mealy equivalente como : MO = ({ a,b },{ 0,1},{ q_0, q1, q_2} , f , g')

- f ($\mathbf{q}_o$, a) = $\mathbf{q}_o$, g ($\mathbf{q}_o$, a) = 0, Entonces g' ($\mathbf{q}_o$, a) = 0
- f ($\mathbf{q}_o$, b) = q_2, g ($\mathbf{q}_o$, b) = 1, Entonces g' ($\mathbf{q}_o$, b) = 1
- f ($\mathbf{q}_1$, a) = $\mathbf{q}_1$, g ($\mathbf{q}_1$, a) = 1, Entonces g' ($\mathbf{q}_1$, a) = 1
- f ($\mathbf{q}_1$, b) = q0, g ($\mathbf{q}_1$, b) = 0, Entonces g' ($\mathbf{q}_1$, b) = 0
- f (q_2 , a) = q_2, g (q_2 , a) = 1, Entonces g' (q_2 , a) = 1
- f (q_2 , b) = $\mathbf{q}_1$, g (q_2, b) = 1, Entonces g' (q_2 , b) = 1

F	A	B
q_0	q_0	q_2
$\mathbf{q}_1$	$\mathbf{q}_1$	q_0
q_2	q_2	$\mathbf{q}_1$

G	A	b
q_0	0	1
$\mathbf{q}_1$	1	0
q_2	1	1

EQUIVALENCIA DE MAQUINAS SECUENCIALES

Dos máquinas secuénciales son equivalentes cuando al ser alimentadas por las mismas palabras de entrada, generan las mismas salidas y entonces habrán ocurrido las equivalencias de estado.

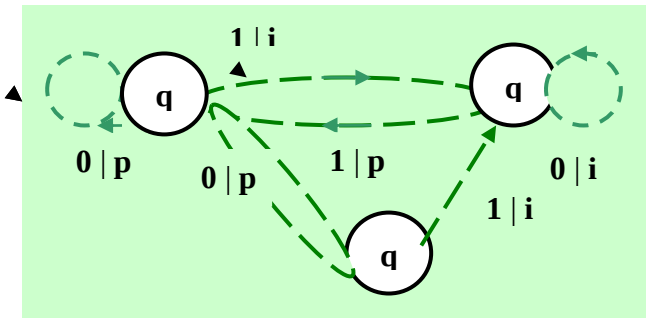
Las máquinas secuénciales $\mathbf{M_1}$ y $\mathbf{M_2}$ son equivalentes $\mathbf{M_1} \in \mathbf{M_2}$, $\Leftrightarrow$ para cada estado $\mathbf{p} \in \mathbf{Q_1}$, existe un estado $\mathbf{q} \in \mathbf{Q_2}$, tal que $\mathbf{p}E\mathbf{q}$ y, además para cada estado $\mathbf{q} \in \mathbf{Q_2}$, existe otro estado $\mathbf{p} \in \mathbf{Q_1}$, tal que $\mathbf{p}E\mathbf{q}$.

A: MAQUINAS SECUENCIALES: EQUIVALENCIAS DE ESTADO:

Dos estados $\mathbf{q_1}$ y $\mathbf{q_2} \in \mathbf{Q}$ son equivalentes $\mathbf{q_1}E\mathbf{q_2}$, si para cada palabra $\mathbf{x} \in \mathbf{\Sigma_E}^*$ la salida generada leyendo $\mathbf{x}$ comenzando con $\mathbf{q_1}$, genera igual salida que leyendo la misma $\mathbf{x}$ comenzando con $\mathbf{q_2}$.Así: $\mathbf{h(q_1,x) = h(q_2,x)}$

Ejemplo: Para funciones de transición y de salida de $\mathbf{MO_1 = (\{ 0,1 \}, \{ p, i \}, \{ q_o, q_1, q_2 \}, f, g)}$:

Diagrama



Transiciones

$.f(\mathbf{q_o}, 0) = \mathbf{q_2}$
$.f(\mathbf{q_o}, 1) = \mathbf{q_1}$
$.f(\mathbf{q_1}, 0) = \mathbf{q_1}$
$.f(\mathbf{q_1}, 1) = \mathbf{q_o}$
$.f(\mathbf{q_2}, 0) = \mathbf{q_o}$
$.f(\mathbf{q_2}, 1) = \mathbf{q_1}$

Salidas

$.g(\mathbf{q_o}, 0) = \mathbf{p}$
$.g(\mathbf{q_o}, 1) = \mathbf{i}$
$.g(\mathbf{q_1}, 0) = \mathbf{i}$
$.g(\mathbf{q_1}, 1) = \mathbf{p}$
$.g(\mathbf{q_2}, 0) = \mathbf{p}$
$.g(\mathbf{q_2}, 1) = \mathbf{i}$

Los estados $\mathbf{q_o}$ y $\mathbf{q_2} \in \mathbf{Q}$ son equivalentes $\mathbf{q_o}E\mathbf{q_2}$ porque producen la misma salida, empezando por cualquiera de los dos estados:

Inicio	Paso 1	Paso 2	Paso 3	Paso 4	Fin
➢ $.h(\mathbf{q_o}, \mathbf{011})$	$h(\mathbf{q_o},0)\, h(f(\mathbf{q_o},0), \mathbf{11})$	$\mathbf{p}h(\mathbf{q_2},\mathbf{11})$	$\mathbf{p}h(\mathbf{q_2},1)\, h(f(\mathbf{q_2},1),\mathbf{1})$	$\mathbf{pi}h(\mathbf{q_1},\mathbf{1})$	**pip**
➢ $.h(\mathbf{q_2}, \mathbf{011})$	$h(\mathbf{q_2},0)\, h(f(\mathbf{q_2},0), \mathbf{11})$	$\mathbf{p}h(\mathbf{q_o},\mathbf{11})$	$\mathbf{p}h(\mathbf{q_o},1)\, h(f(\mathbf{q_o},1),\mathbf{1})$	$\mathbf{pi}h(\mathbf{q_1},\mathbf{1})$	**pip**

B. MAQUINAS SECUENCIALES: CONJUNTO COCIENTE Q|E

La relación de equivalencia entre estados, particiona el conjunto de estados en subconjuntos cuyos estados pertenecientes a una misma clase serán equivalentes entre si. Tal partición se denomina conjunto cociente $\mathbf{Q|E}$ generado por la relación de equivalencia E. Calcular de $\mathbf{Q|E}$ requiere una **función conjunto cociente** $\mathbf{Q_E}$, la cual en cada iteración determina particiones de equivalencia con las relaciones sucesivas de longitud **i**.

- La función empieza calculando la relación de equivalencia de longitud $\mathbf{1}\ \mathbf{Q|E_1}$;
- Seguir hasta que dos $\mathbf{Q|E}$ seguidos sean iguales $\mathbf{Q|E_i = Q|E_{i+1}}$; así se determina el $\mathbf{Q|E}$ de la relación de equivalencia.

ALGORITMO de la FUNCION CONJUNTO COCIENTE (M) $\mathbf{Q_E}$

- **M**: Entrada a la función, Máquina Secuencial:
- $\mathbf{Q_E}$: Salida de la función, Conjunto Cociente de la Máquina Secuencial.

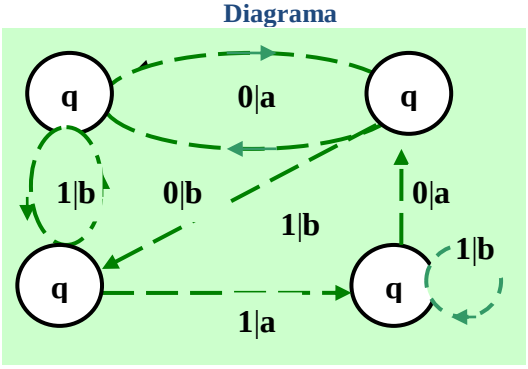
Para todos los estados $\mathbf{p, q} \in \mathbf{Q}$
Si $\mathbf{a} \in \mathbf{\Sigma_E}$ se cumple $\mathbf{h(p, a) = h(q, a)}$
Entonces p y q estarán en la misma clase $\mathbf{c_i} \in \mathbf{Q|E_1}$;
.i = 1;
Repetir
Para todos los estados $\mathbf{p, q} \in \mathbf{Q}$
Si $\mathbf{p, q} \in \mathbf{c_j} \in \mathbf{Q|E_i}$; $\mathbf{a} \in \mathbf{\Sigma_E}$; $\mathbf{f(p, a)} \in \mathbf{c_m}$; $\mathbf{f(q, a)} \in \mathbf{c_m}$; $\mathbf{c_m} \in \mathbf{Q|E_1}$
Entonces p y q seguirán estando en la misma clase $\mathbf{c_j} \in \mathbf{Q|E_{i+1}}$

.i = i + 1
Hasta que $\mathbf{Q|E_{i-1}} = \mathbf{Q|E_i}$
Devolver $\mathbf{Q|E} = \mathbf{Q|E_i}$

FUNCION CONJUNTO COCIENTE (M) Q_E

Ejemplo: Dadas las funciones de transición y de salida máquina $\mathbf{M_1 = (\{0,1\},\{a,b,c\},\{q_o,q_1,q_2,q_3\},f,g)}$:

Diagrama



Transiciones	Salidas
$.f(q_o, 0) = q_3$	$.g(q_o, 0) = a$
$.f(q_o, 1) = q_2$	$.g(q_o, 1) = b$
$.f(q_1, 0) = q_3$	$.g(q_1, 0) = a$
$.f(q_1, 1) = q_1$	$.g(q_1, 1) = b$
$.f(q_2, 0) = q_o$	$.g(q_2, 0) = b$
$.f(q_2, 1) = q_1$	$.g(q_2, 1) = a$
$.f(q_3, 0) = q_o$	$.g(q_3, 0) = a$
$.f(q_3, 1) = q_1$	$.g(q_3, 1) = b$

- En el conjunto cociente inicial $\mathbf{Q|E_1}$, están en igual clase los estados cuya salida es la misma con cada símbolo de entrada, o sea que son equivalentes en longitud 1($p\mathbf{E_1}q$) Así, $\mathbf{q_o}$ y $\mathbf{q_3}$ están en la misma clase, porque se cumplen la igualdades:
  $h(q_o, 0) = a$ $h(q_3, 0) = a$
  $h(q_o, 1) = b$ $h(q_3, 1) = b$
  En cambio, como $h(q_o, 0) = a$ y $h(q_2, 0) = b$; indica que $\mathbf{q_o}$ y $\mathbf{q_2}$ no están en la misma clase, así el conjunto cociente inicial quedará como: $\mathbf{Q|E_1 = [\{q_o, q_1, q_3\}, \{q_2\}]}$

- Luego en el siguiente conjunto cociente $\mathbf{Q|E_2}$, dos estados pertenecerán a la misma clase si ya estaban en la misma clase en $\mathbf{Q|E_1}$ y para cada símbolo de entrada, sus transiciones desde cada estado, llevan a los estados que estuvieron en igual clase del conjunto anterior. Así, $\mathbf{q_o}$ y $\mathbf{q_3}$ seguirán en igual clase de equivalencia, pues pertenecían a la misma en $\mathbf{Q|E_1}$ y, además:
  $f(q_o, 0) = q_3$ $f(q_3, 0) = q_o$
  $f(q_o, 1) = q_2$ $f(q_3, 1) = q_2$
  Como $\mathbf{q_o}$ y $\mathbf{q_3}$, al igual que $\mathbf{q_2}$ y $\mathbf{q_2}$ estaban en igual clase de equivalencia en $\mathbf{Q|E_1}$, el siguiente conjunto cociente, $\mathbf{Q|E_2}$ será: $\mathbf{Q|E_2 = [\{q_o, q_3\}, \{q_1\}, \{q_2\}]}$

- Ahora, como $\mathbf{Q|E_2} \neq \mathbf{Q|E_1}$ se debe continuar aplicando el algoritmo para determinar $\mathbf{Q|E_3}$, que resultará ser: $\mathbf{Q|E_3 = Q|E_2 = [\{q_o, q_3\}, \{q_1\}, \{q_2\}]}$

- El algoritmo termina y el conjunto cociente es: $\mathbf{Q|E = Q|E_3 = Q|E_2 = [\{q_o, q_3\}, \{q_1\}, \{q_2\}]}$

E. MAQUINAS SECUENCIALES: EQUIVALENCIA DE LONGITUD N

Dos estados $\mathbf{q_1}$ y $\mathbf{q_2} \in \mathbf{Q}$ son equivalentes de longitud **n**, $\mathbf{q_1 E q_2}$, cuando para cada palabra de entrada $\mathbf{x \in \Sigma_E^*}$ de longitud n(|x|=n), se genera la misma salida . Así se cumple que: $\mathbf{h(q_1,x)=h(q_2,x)}$

Es una relación de equivalencia $\Rightarrow$ se define el conjunto sobre **Q**, conjunto cociente $\mathbf{Q/E_n}$

LEMAS:

1. $\mathbf{pE_nq \Rightarrow pE_kq \ \forall\ k < n}$
2. $\mathbf{pE_{n+1}q \Leftrightarrow pE_nq \ \&\ f(p,a)E_n\ f(q,a)\ \forall\ a \in \Sigma_E}$ **Corolario:** $\mathbf{|Q/E_n| \leq |Q/E_{n+1}|}$
3. $\mathbf{Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E_{n+i}\ \forall\ i = 0,1,2,.....}$
4. $\mathbf{Q/E_n = Q/E_{n+1} \Leftrightarrow Q/E_n = Q/E}$
5. $\mathbf{|Q/E_1| = 1 \Rightarrow Q/E_1 = Q/E_2}$

6. $n = |Q| > 1 \Rightarrow Q/E_{n-1} = Q/E_n$

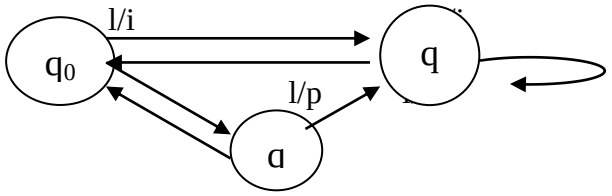
TEOREMA: p E q $\Leftrightarrow$ $pE_{n-1}q$, n = |**Q**|, en general n-1 es el valor más pequeño que cumple el teorema.

ALGORITMO PARA CONSTRUIR: Q/E ∀ MS | |Q| > 1

1. **Q/E_1** se construye aplicando la regla:
   p y q están en la misma clase si y solo si: $h(p, a) = h(q, a)$ ∀ $a \in \Sigma_E$
2. Sea: **Q/E_i** = $\{c_1, c_2, ..., c_j\}$.
   Q/E_{i+1} se construye: p y q están en la misma clase si y solo si
   p, q$\in c_K$ y ∀ $a \in \Sigma_E$ se verifica que $f(p, a)$ y $f(q, a)$ están en la misma clase cm de **Q/E_i**
3. Si **Q/E_i**=**Q/E_{i+1}** Entonces **Q/E_i**=**Q/E,** caso contrario aplicar el paso 2 partiendo de **Q/E_{i+1}**

ALGORITMO PARA CONSTRUIR: Q/E ∀ MS | |Q| > 1

Ejemplo: Sea f y g de la maquina MS=(Σ_E,Σ_S,Q,f,g), definida como: M1=({0.1},{p,i},{q_0,q_1,q_2}, f, g)

Diagrama de trasición	Transición y Salida
	➢ $f(q_0,0)=q_2$ $g(q_0,0)=p$ ➢ $f(q_0,1)=q_1$ $g(q_0,1)=i$ ➢ $f(q_1,0)=q_1$ $g(q_1,0)=i$ ➢ $f(q_1,1)=q_0$ $g(q_1,1)=p$ ➢ $f(q_2,0)=q_0$ $g(q_2,0)=p$ ➢ $f(q_2,1)=q_1$ $g(q_2,1)=i$

Se da la equivalencia q_1 E_3 q_2, porque, para toda cadena x de longitud 1, se cumple: $h(q_1,x) = h(q_2,x)$.
Así, si: **x = 000**:

- $h(q_1,000) = h(q_1,0)h(f(q_1,0),00) = ph(q_2,0) = ph(q_2,0)h(f(q_2,0),0) = pph(q_0,0) = ppp$
- $h(q_2,000) = h(q_2,0)h(f(q_2,0),0) = ph(q_0,0) = ph(q_0,0)h(f(q_0,0),0) = pph(q_2,0) = ppp$

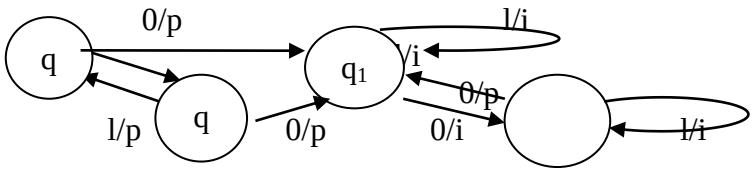
Se cumple que $h(q_1, 000) = h(q_2, 000)$.
Así, para x = 1:

- $h(q_1, 1) = p$
- $h(q_2, 1) = i$

No se cumple que $h(q_1,1) = h(q_2,1)$, porque depende de los estados con los que los compare.

ALGORITMO PARA CONSTRUIR: Q/E ∀ MS | |Q| > 1

Ejemplo: Sea f y g de la maquina MS=(Σ_E,Σ_S,Q,f,g), definida como: M_2=({0.1},{p,i},{q_0,q_1,q_2,q_3}, f, g)

Diagrama de transición	Transición y Salida
	**$f(q_0,0)=q_1$ $g(q_0,0)=i$** **$f(q_0,1)=q_2$ $g(q_0,1)=p$** **$f(q_1,0)=q_3$ $g(q_1,0)=p$** **$f(q_1,1)=q_1$ $g(q_1,1)=i$** **$f(q_2,0)=q_3$ $g(q_2,0)=p$** **$f(q_2,1)=q_0$ $g(q_2,1)=i$** **$f(q_2,0)=q_1$ $g(q_3,0)=p$** **$f(q_3,1)=q_3$ $g(q_3,1)=i$**

La equivalencia $q_1E_1q_2$, se da para cualquier palabra **x** de longitud **1**, se cumple: $h(q_1,x)=h(q_2,x)$. Para x = 1

- $h(q_1,1) = i$
- $h(q_2,1) = i$

Por lo tanto se que $h(q_1,1) = h(q_2,1)$
En cambio si .x = 00 la equivalencia $q_0E_2q_2$, no se cumple $h(q_0, 00) = h(q_2, 00)$

- $h(q_0,00)=h(q_0,0)\ h(f(q_0,0),0)=ih(q_1,0)=ip$

- $h(q_2,00)=h(q_2,0)h(f(q_2,0),0)=ph(q_3,0)=pp$

CONCLUSIÓN: Dos estados son equivalentes si tienen la misma longitud y ambos comparados responden de igual manera (salidas y longitud tienen que ser iguales) siempre que las entradas y la longitud sean las mismas para los dos estados.

MAQUINAS SECUENCIALES: MINIMIZACION

Dado una máquina secuencial, la minimización consiste en calcular otra máquina equivalente secuencial mínima, capaz de comportarse del mismo modo, pero **con el menor número de estados**. Para esto:

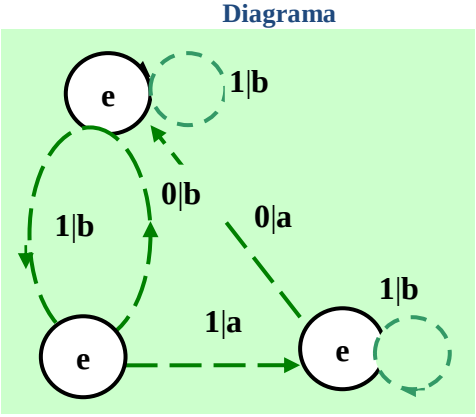
Definicion	Elementos
Si: **$M = (\Sigma_E, \Sigma_S, Q, f, g)$** se define la máquina secuencial mínima como **$M' = (\Sigma_E, \Sigma_S, Q', f', g')$**	○ **Q' = Q|E**: Estados de la máquina mínima son clases de equivalencia del **conjunto cociente**. ○ **.f'(c, a) = c'** si (**q∈c, f(q,a)∈c'**) donde: **c,c'∈Q|E** ○ **.h'(c, a) = h(q, a), (q∈ c)** donde: **c∈Q|E** ○ **.g'**: Función de Salida.

MINIMIZACION DE MAQUINAS SECUENCIALES:

Ejemplo Dada la máquina secuencial **$M_1 = (\{0,1\}, \{a, b, c\}, \{q_o, q_1, q_2, q_3\}, f, g)$** su máquina secuencial mínima equivalente será: **$M'_1 = (\{0,1\}, \{a, b, c\}, \{e_o, e_1, e_2\}, f', g')$**

Cuyos estados equivalentes serán **$e_o = \{q_o, q_3\}$**, **$e_1 = \{q_1\}$**, **$e_2 = \{q_2\}$** y sus funciones de transición y de salida serán:

Diagrama



Transiciones

.f '(e_o, 0) = e_o
.f '(e_o, 1) = e_2
.f '(e_1, 0) = e_o
.f '(e_1, 1) = e_1
.f '(e_2, 0) = e_o
.f '(e_2, 1) = e_1

Salidas

.g' (e_o, 0) = **a**
.g' (e_o, 1) = **b**
.g' (e_1, 0) = **a**
.g' (e_1, 1) = **b**
.g' (e_2, 0) = **b**
.g' (e_2, 1) = **a**

MINIMIZACION DE MAQUINAS SECUENCIALES:

Ejemplo

Q/Σ_E	a	b
Q_0	Q5/0	Q4/1
Q1	Q4/0	Q3/0
Q2	Q2/0	Q0/1
Q3	Q0/0	Q1/0
Q4	Q1/0	Q2/1
Q5	Q3/3	Q5/1

$\Sigma_E = \{ a, b \}$ $\Sigma_S = \{ 0, 1 \}$

proceso de particularizacion

$P_1 = \{[q_o, q_2, q_4, q_5], [q_1, q_3]\}$

. c_1 c1 c2 c2 c1 c1

. c_1 c1 c1 c1 c2 c2

$P_2 = \{[q_o, q_2], [q_4, q_5], [q_1, q_3]\}$

.c2 c1 c3 c3 c2 c1

$Q/\Sigma e$	A	b
C1	C4/0	C3/1
C2	C2/0	C1/1
C3	C5/0	C2/1
C4	C6/0	C4/1

C5	C1/1	C6/0
C6	C1/1	C5/0

.c2 c1 c1 c2 c3 c3

.c1 c2 c3 c4 c5 c6

$P_3 = \{[\mathbf{q}_o][q2][q4][q5][q1][q3]\} \therefore P_4 = P_3$

$\therefore$ Esta Mealy ya es mínima, no se puede simplificar más

$\mathbf{Q}_{me}\{\mathbf{q}_o, q1, q2, q3, \mathbf{q}_4, \mathbf{q}_5\} = \mathbf{Q}_{min}\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$

MINIMIZACION DE MAQUINAS SECUENCIALES:

Ejemplo $\sum e\{a, b\}$ $\sum s\{0,1,2\}$

Q/∑e	a	b
Q0	Q5/1	Q4/0
Q1	Q4/2	Q3/1
Q2	Q2/0	Q0/2
Q3	Q6/2	Q3/1
Q4	Q1/1	Q2/0
Q5	Q3/0	Q5/2
Q6	Q6/2	Q3/1

c1 c2 c3

P1={[$\mathbf{q}_o$, q4][q1, q3, q6,][q2, q5]}

C3 c2 c1 c2 c2 c3 c2

C1 c3 c2 c2 c2 c1 c3

P2={[$\mathbf{q}_o$][q4][q1][q3, q5][q2][q5]} $\therefore$ P3=P2

C4 c4

C4 c4

$\mathbf{Q}_{min}\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$

Q/∑e	A	b
C1	C1/1	C2/0
C2	C3/1	C5/0
C3	C2/2	C4/1
c4	C4/0	C4/1
C5	C5/0	C1/2
C6	C4/0	C6/2

MINIMIZACION DE MAQUINAS SECUENCIALES:

Ejemplo

$\sum e=\{a,b\}$ $\sum s=\{0,1\}$

Q/∑s	A	b
Q0/0	Q1	Q2
Q1/0	Q2	Q3
Q2/1	Q3	Q4
Q3/0	Q4	Q6
Q4/0	Q0	Q5
Q5/0	Q5	Q3
Q6/1	Q3	Q4
Q7/0	Q0	Q6
Q8/0	Q8	7Q

c1 c2 c3

P1={[q0, q3, q7], [q1], [q2, q4, q5, q6, q8]}

C2 c3 c1 c1 c1 c3 c1 c2

C3 c3 c3 c3 c3 c1 c3 c1

C1 c2 c3 c4 c5 c6

P2={[q0], [q3], [q7], [q1], [q2, q4, q6], [q5, q8]}

C2 c1 c2 c6 c6

C5 c6 c5 c2 c3

Q/∑s	A	b
C1/0	C4	C5
C2/0	C6	C5

C3/0	C4	C5
C4/0	C5	C2
C5/1	C2	C6
C6/0	C1	C7
C7/0	C7	C2
C8/0	C8	C3

C1 c2 c3 c4 c5 c6 c7 c8

P3={[q0], [q3], [q7], [q1], [q2, q6], [q4], [q5], [q8]}

P4=P5 Qmin{c1, c2, c3, c4, c5, c6, c7, c8}

∴Qmin={c1, c2, c3, c4, c5, c6, c7}

q8Λc8 son estos inaccesibles los elimino

TEOREMA DE KLEENE

Demuestran la equivalencia entre las Expresiones Regulares y los Autómatas Finitos, los cuales representan a su modo a los Lenguajes Regulares. Para esto el teorema expresa: "**Un lenguaje L es aceptado por un AFD, si y solo si, L es un conjunto regular**".

Ejemplo: Sea L el conjunto aceptado por el AFD M = ({q1,...,qn}, £, S, q1, F)
Donde R^K_{ij} el conjunto de todas las cadenas x tales que S (q_i, x) = q_j, y si S (q_i, y) = q_e, para cualquier y que es un prefijo de x, y ≠ E, entonces e ≤ k.

Es decir, R^k_{ij} es el conjunto de cadenas tales que pasan al autómata finito del estado q_i al estado qj sin pasar por ningún estado enumerado mayor a k.

Dado que no existe ningún estado enumerado mayor que n, R^n_{ij} denota todas las cadenas van de q_i a q_j.

Es posible definir R^k_{ij} de una manera recursiva si procedemos de la siguiente manera:

$$R^k_{ij} = R^{k-1}_{ik} (R^{k-1}_{kk})^* R^{k-1}_{kj} \cup R^{k-1}_{ij}$$

$$R^0_{ij} = \begin{cases} \{a \mid S(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid S(q_i, a) = q_j\} \cup \{E\} & \text{if } i = j \end{cases}$$

Debemos demostrar, que para cada i,j y k , existe una expresión regular r^k_{ij} que denota el lenguaje R^k_{ij} .
Procedamos por inducción sobre k:

Base:

$$r^o_{ij} = \begin{cases} a_1 + a_2 + \ldots + a_p & \text{si } i \neq j \\ a_1 + a_2 + \ldots + a_p + E & \text{si } i = j \end{cases}$$

Donde $\{a_1, a_2, \ldots, a_p\}$ son todos los símbolos de £ tales que S (q_i, a) = q_j

Si este es un conjunto vacío, entonces:

$$r^{o}_{ij} = \begin{cases} 0 & \text{si } i \neq j \\ E & \text{si } i = j \end{cases}$$

Inducción: La formula recursiva para R^{k}_{ij} tan solo incluye las generaciones: unión, concatenación y cerradura. Por la hipótesis de inducción, para cada l y m existe una expresión regular r_{lm}^{k-1} tal que: $L(r_{lm}^{k-1}) = R_{lm}^{k-1}$ Entonces para r_{ij}^{K} podemos elegir la expresión regular:

$(r_{ik}^{k-1})(r_{kk}^{k-1})^{*}(r_{kj}^{k-1}) + r_{ij}^{k-1}$ lo cual completa la inducción.

Para terminar la demostración, solo debemos de observar que: $L(M) = U\, R^{n}_{ij}$

TEOREMAS DE ANÁLISIS Y DE SÍNTESIS DE KLEENE

1) TEOREMA DE ANÁLISIS:

Todo lenguaje aceptado por un AF es un lenguaje regular.
Solución al problema de análisis: encontrar el lenguaje asociado a un determinado AF: "Dado un AF, A, encontrar la E. R. que describe L(A)".

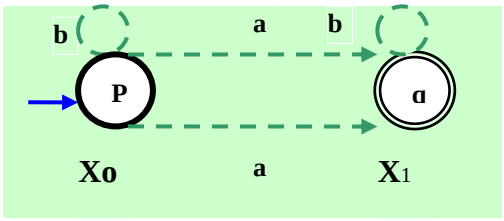
2) TEOREMA DE SÍNTESIS:

Todo lenguaje regular es el lenguaje aceptado por un AF.
Solución al problema de síntesis: encontrar un reconocedor para un lenguaje regular dado: "Dada una E. R. que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular".

ECUACIONES CARACTERÍSTICAS: Describen todas las cadenas que se pueden reconocer desde un estado dado.

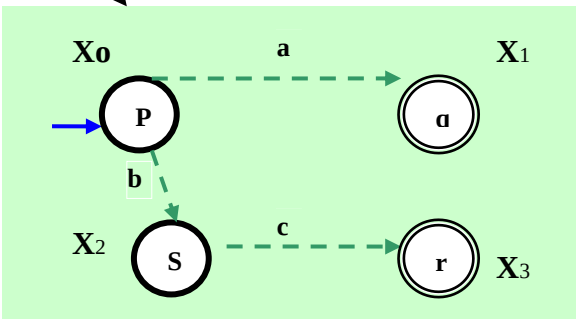
- Se escribe la ecuación x_i por estado q_i
- Primer miembro x_i ; el segundo miembro tiene un termino por cada rama que salga de q_i.
- Las ramas tienen la forma $a_{ij} \bullet x_j$ donde a_{ij} es la etiqueta de la rama que une q_i con q_j, x_j es la variable correspondiente a q_j.
- Se añade un termino a_{ij} por cada rama que une q_i con un estado final.
- Si de un estado no sale ninguna rama, el segundo miembro será:
  si es final: $x_i = \lambda$ si no es final: $x_i = \Phi$

SOLUCIÓN AL PROBLEMA DE ANÁLISIS



$X_0 = bX_0 + a\,X_1 + a$
$X_1 = b\,X_1 + a\,X_0 + b$

EJEMPLO



$X_0 = a\,X_1 + b\,X_2 + a$
$X_1 = \lambda$
$X_2 = c\,X_3 + c$
$X_3 = \lambda$

AF: AUTOMATAS ESPECIALES

AFP: AUTOMATAS PROBABILISTICOS: En estos autómatas a partir de sus símbolos de entrada, las correspondientes producciones de transiciones entre estados son acontecimientos probabilísticos, con valores de 1 ó 0, según la certeza que estos ocurran o no.
Por tal razón siempre existirá para un instante t, un comportamiento probabilístico de transición, de que se encuentre en algún estado de la máquina; tal como ocurriría con los movimientos de un robot.

Definición: **AFP = (Σ, Q, M, P(o), F)** Donde

- Σ Alfabeto de símbolos de entrada
- Q Conjunto de estados
- P(o) Vector de estado inicial: En un instante t el vector de estados P(t) por cada estado del autómata, tiene una componente y el contenido de cada posición i del vector, Pi(t), se corresponde con la probabilidad de que tal instante el autómata se encuentre en el estado i. Por ello se define:

  $P_{(t)} = (P_{1(t)}, P_{2(t)}, P_{2(t)}, \ldots P_{n(t)})$
- M Conjunto de matrices de probabilidad de transición entre los estados $M = \{ M(a) | a \in \Sigma \}$
  Existe una matriz probabilística de transición M(a) para cada símbolo a del alfabeto de entrada Σ, que define la probabilidad de que transite a cada uno de los estados, cuando el autómata se encuentre en un determinado estado y reciba el símbolo de entrada a. Se la representa así:

$$M(a) = \begin{vmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \ldots\ldots & p_{1,,n} \\ p_{2,1} & p_{2,2} & p_{2,3} & \ldots\ldots & p_{2,,n} \\ p_{3,1} & p_{3,2} & p_{3,3} & \ldots\ldots & p_{3,,n} \end{vmatrix}$$

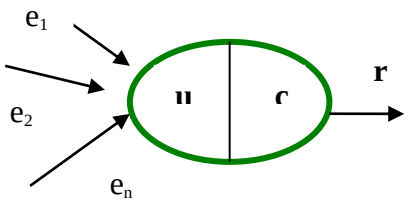
$$\begin{vmatrix} \ldots & \ldots\ldots\ldots\ldots\ldots\ldots \\ p_{n,1} \quad p_{n,2} \quad p_{n,3} & \ldots\ldots\ldots \; p_{n,,n} \end{vmatrix}$$

Donde:

- . n = | Q | Número de estados
- . $p_{i,,j}$ Probabilidad de transitar al estado **j**, al recibir **a** como entrada, estando en estado **i**
- Por cada $p_{i,,j}$ se cumple $0 \leq p_{i,,j} \leq 1$
- Para cada estado i se cumple $\sum_{j=1}^{n} p_{i,,j} = 1$

▪ F $\subseteq$ Q Conjunto de estados finales, definen la probabilidad que el autómata se encuentre en cada estado

ACMP: AUTOMATA DE CELULAS de McCULLOCH-PITS



Dan basamento al estudio de las denominadas "Redes de Neuronas Artificiales" las cuales son analogías que tratan de imitar el funcionamiento de las neuronas humanas.

Para ello están constituidas por un conjunto de células o neuronas interconectadas entre si, y poseen:

- **.e_1 , e_2 , . . .e_n** : Vectores de entrada
- **.u**: Umbral
- **.c**: Nombre de la célula
- **.r**: Salida

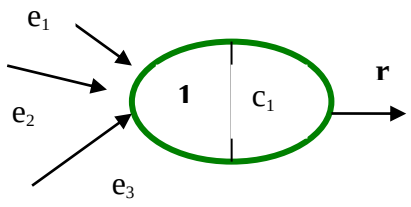
Estas células de comportamiento similar a la Máquina Secuencial de Moore, poseen las siguientes propiedades:

- Pueden estar en estado 1: Activo y 0: Inactivo.
- A partir de otras células o como entrada del autómata, pueden recibir dos tipos de señales: 1: Positivas y 0: Negativas
- Pueden tener una o varias entradas $\mathbf{e_i}$ de dos tipos: (→): Excitadoras o (¬): Inhibidoras.
- En el instante t, si su estado q(t) es activo, tendrá una salida r(t)=1 y r(t)=0 si tal estado es inactivo
- La salida que puede ir a una o a varias células inclusive a ella misma, en un instante t, depende solo del estado en ese instante, por lo que r(t) = q(t)

- Su estado **q(t+1)** en el instante t+1, no depende del estado en el instante t, si no solo de la entrada $\mathbf{e_i(t)}$ en tal momento **t**, de modo que su valor será calculado por la función de transición **g** a partir de las entradas a partir del instante t: **q(t+1) = g(e(t))**
- Poseen una función de transición **g** que les permiten transitar entre estados a partir de las entradas a una célula
- Poseen un umbral u ∈ Z que les permiten definir las funciones de transición.
- La salida en el instante t+1 depende del estado al que transite la célula a partir de las entradas en el instante t, de modo que: $r(t+1) \overset{1}{=} q(t+1) \overset{2}{=} g(e(t))$
- Para **n** entradas podemos obtener distintos tipos de células, con solo variar la función de transición **g**, así para $\sum_{i=0}^{n} = 0.e_i(t) \geq u$ será g(t) = 1, ó g(t) = 0 en otro caso. Además la célula se activará si la suma de la señal que recibe en su entrada es mayor o igual que el umbral, cuando:
  - $.e_i(t)$ = 1 si la entrada es excitadora y la señal es 1
  - $.e_i(t)$ = -1 si la entrada es inhibidora y la señal es 1
  - $.e_i(t)$ = 0 en caso contrario

AUTOMATA DE CELULAS de McCULLOCH-PITS: Ejemplo:

Para la función de transición g(t) = 1 si $\sum_{i=0}^{n} = 0.e_i(t) \geq u$, ó g(t) = 0 en otro caso



Transiciones			
$.e_i(t)$			$.q(t+1) = r(t+1)$
$.e_1(t)$.	$e_2(t)$.	$e_3(t)$	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Bibliográfica

- **Teoría de autómatas y lenguajes formales**.
  Autómatas y complejidad. Kelly Dean Editorial Prentice Hall

- **Introducción a la teoría de autómatas, lenguajes y computación**
  John E. Hopcroft; Jeffrey D. Ullman Editorial Cecsa

- **Teoría de la computación**
  J. Gleuu Brokshear Editorial Addison Wesley Iberoamericana

CONTENIDOS. 2013

- **Unidad 1)** Lingüística Matemática: Alfabetos, palabras y lenguajes. Operaciones con cadenas y con lenguajes. Niveles de un lenguaje. Gramáticas para estructuras de frases. Diagramas de Sintaxis y Formato BNF. Formalismos para el análisis semántico: Sistemas Canónicos de Donovan y Esquemas de Traducción. Proceso de Compilación.
- **Unidad 2)** Gramáticas y Modelos: Jerarquía de Chomsky y formatos estándares para tipos 0 y 1. Los aceptores de lenguajes formales: Máquina de Turing (MT) y Autómata Linealmente Limitado (ALL). Lenguajes Recursivos y recursívamente enumerables. Resolubilidad y complejidad computacional.
- **Unidad 3)** Lenguajes Regulares (LR) y Autómatas Finitos (AF): Gramáticas Regulares (GR), Expresiones Regulares (ER). Máquinas Secuenciales: Modelos de Mealy y de Moore. Autómata Finito Determinista (AFD). Minimización. Propiedades de los LR. Autómatas Finitos No Deterministas (AFND). Operaciones con AF. Conversión AFND/AFD. Obtención del AF a partir de una ER. Analizador Lexico (Scanner).
- **Unidad 4)** Lenguajes Independientes del Contexto (LIC) y Autómatas de Pila: Gramáticas Independientes del Contexto (GIC). Árboles de derivación. Ambigüedad. Árbol Total del Lenguaje. Simplificaciones de una GIC. Formas Normales. LIC y Autómatas de Pila (AP). Criterios de aceptación por estado final y por pila vacía. Diseño de un AP a partir de una GIC. Obtención de la GIC a partir de un AP. Propiedades de los LIC. Analizador Sintáctico (Parser) Descendente (LL) y Ascendente (LR).