CAPÍTULO 8: AUTÓMATAS

- La palabra *autómata* evoca algo que pretende imitar las funciones propias de los seres vivos, especialmente relacionadas con el movimiento, por ejemplo el típico robot antropomorfo.
- En el campo de los Traductores, Procesadores, Compiladores e Intérpretes, lo fundamental no es la simulación del movimiento, sino **la simulación de procesos para tratar información**.
- La información se codifica en cadenas de símbolos, y **un autómata es un dispositivo que manipula cadenas de símbolos que se le presentan a su entrada, produciendo otras tiras o cadenas de símbolos a su salida**.
  - El autómata recibe los símbolos de entrada, uno detrás de otro, es decir secuencialmente.
  - El símbolo de salida que en un instante determinado produce un autómata, no sólo depende del último símbolo recibido a la entrada, sino de toda la secuencia o cadena, que ha recibido hasta ese instante.
- Todo lo anterior conduce a definir un concepto fundamental : **estado de un autómata**.

El estado de un autómata es toda la información necesaria en un momento dado, para poder deducir, dado un símbolo de entrada en ese momento, cual será el símbolo de salida.

  - Es decir, conocer el estado de un autómata, es lo mismo que conocer la historia de los símbolos de entrada, así como el **estado inicial**, estado en que se encontraba el autómata al recibir el primero de los símbolos de entrada.

AUTÓMATAS

- El autómata tendrá un determinado *número de estados* (pudiendo ser infinitos), y se encontrará en uno u otro según sea la historia de símbolos que le han llegado.
- Se define ***configuración de un autómata*** **a su situación en un instante**.
- Se define ***movimiento de un autómata*** **como el transito entre dos configuraciones**.
- Si un autómata se encuentra en un estado determinado, recibe un símbolo también determinado, producirá un símbolo de salida y efectuará un cambio o *transición* a otro estado (también puede quedarse en el mismo estado).
  - El campo de estudio de los Traductores, Procesadores e Intérpretes son los lenguajes y las gramáticas que los generan.
  - Los elementos del lenguaje son sentencias, palabras, etc... formadas a partir de un alfabeto o vocabulario, que no es otra cosa que un conjunto finito de símbolos.
- Establecidas las reglas gramaticales, una cadena de símbolos pertenecerá al correspondiente lenguaje si tal cadena se ha formado obedeciendo esas reglas.
  - Entonces un **autómata reconocedor de ese lenguaje**, funciona de tal forma que cuando reciba a su entrada una determinada cadena de símbolos indica si dicha cadena pertenece o no al lenguaje.
- También se mostrará como existe un tipo de autómata para reconocer cada uno de los tipos de lenguajes generados por las correspondientes gramáticas.

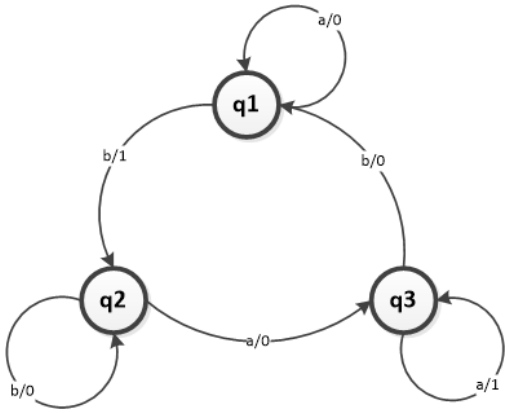
Definición formal de autómata

- Un autómata es una quíntupla A = (E, S, Q, f, g) donde :
  - E = {conjunto de entradas o vocabulário de entrada}
    - E es un conjunto finito, y sus elementos se llaman entradas o símbolos de entrada.
  - S = {conjunto de salidas o vocabulario de salida}
    - S es un conjunto finito, y sus elementos se llaman salidas o símbolos de salida.
  - Q = {conjunto de estados}
    - Q es el conjunto de estados posibles, puede ser finito o infinito.
  - f :$ExQ \rightarrow Q$
    - es la *función de transición* o función del estado siguiente, y para un par del conjunto
    - ExQ devuelve un estado perteneciente al conjunto Q.

      ExQ es el conjunto producto cartesiano de E por Q.
  - g : $ExQ \rightarrow S$
    - es la *función de salida*, y para un par del conjunto E Q, devuelve un símbolo de salida del conjunto S.
- **Representación de autómatas**
  - Los autómatas se pueden representar mediante :
    - Tabla de transiciones
    - Diagramas

Tabla de transiciones

- Las funciones f y g pueden representarse mediante una tabla, con tantas filas como estados y tantas columnas como entradas.
- Así por ejemplo se puede representar el autómata
- A = (E, S, Q, f, g) donde
  - E = {*a,b*},
  - S = {*0,1*},
  - Q = {q1, q2, q3}
  - f y g se pueden representar por las siguientes relaciones:



f	*a*	*b*
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_1

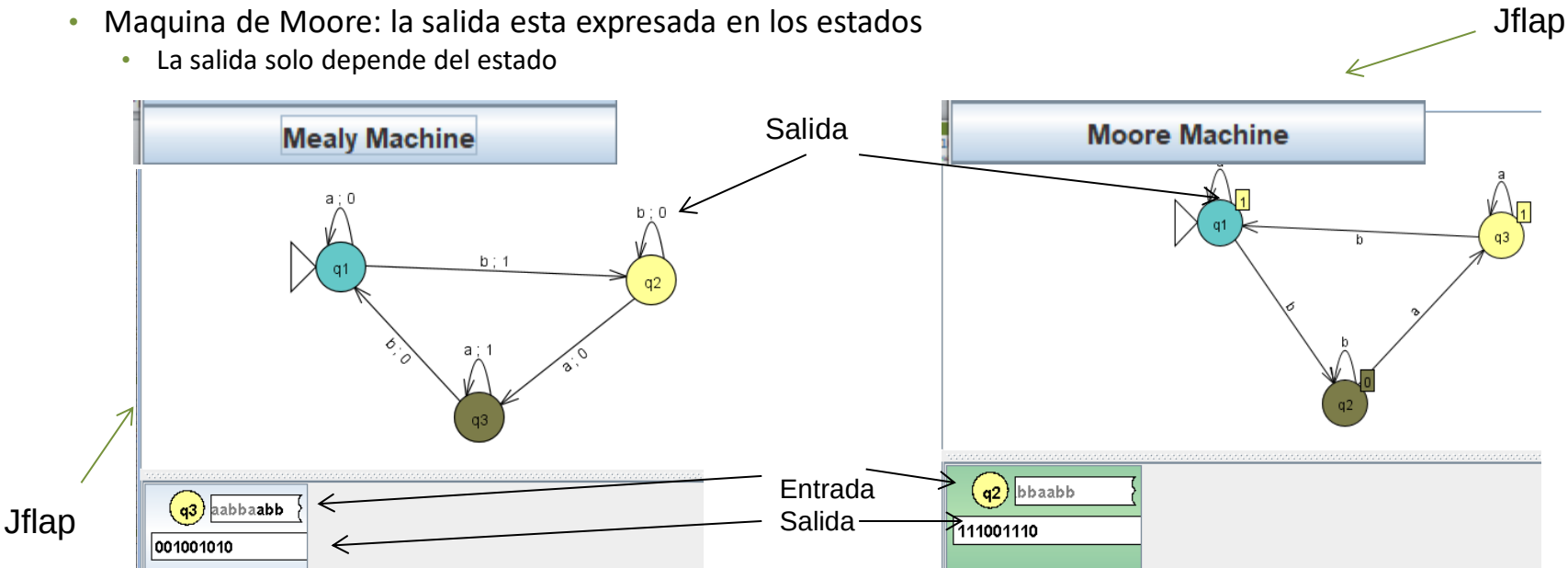
g	*a*	*b*
q_1	*0*	*1*
q_2	*0*	*0*
q_3	*1*	*0*

f / g	*a*	*b*
q_1	$q_1/0$	$q_2/1$
q_2	$q_3/0$	$q_2/0$
q_3	$q_3/1$	$q_1/0$

Así se tiene que $f(a, q_1) = q_1$; $g(a, q_1) = 0$; o también $f(a, q_2) = q_3$; y $g(a, q_3) = 1$

Diagramas de Mealy / Moore

- Los diagramas de Moore/Mearley son otra forma de representar las funciones de transición y salida de un autómata.
- Son grafos dirigidos en el que:
  - Cada nodo corresponde a un estado
  - Cada transición se corresponde con un evento
- Por lo cual, una relacione donde para cada par se obtiene el siguiente estado
  - $(evento, estadoactual) \rightarrow estadosi_{guiente}$
- Mientras que para la salida:
  - Maquina de Meale: la salida esta expresada en los arco junto al evento
    - La salida depende del estado y de donde se viene
  - Maquina de Moore: la salida esta expresada en los estados
    - La salida solo depende del estado



Máquinas de Moore y Mealy

- El modelo general de autómata que se ha definido se llama *máquina de Mealy*.
  - Se puede plantear el siguiente problema : si se considera que además de los elementos del vocabulario de entrada E, *un elemento vacío* λ , que físicamente indica que no hay entrada, se han de ampliar los dominios de las definiciones de la siguiente forma :

$$f : \{E \cup \{\lambda\}\} \times Q \to Q$$

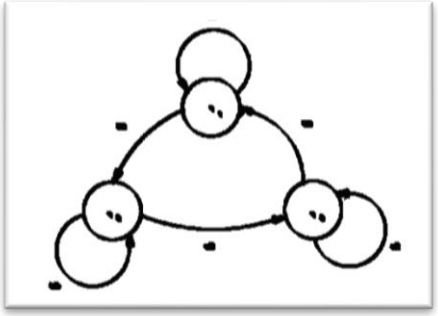
$$g : \{E \cup \{\lambda\}\} \times Q \to S$$

  - La ampliación del dominio f no plantea ningún problema, pues se puede convenir que $f(\lambda, q)=q$, es decir *si no hay entrada, no se cambia de estado*.
  - No ocurre lo mismo con la ampliación del dominio de g, ya que $g(\lambda, q)$, *produce una salida indeterminada*, pues depende del estado anterior al q.
    - Así por ejemplo en el autómata de la figura, se observa que :
    - ya que si se llega a q2 desde q1 la salida es 1,
      mientras que si se alcanza desde el propio q2 la salida es 0.

    Entonces sólo se puede ampliar el dominio de g, si a cada estado q se le puede asociar una salida y sólo una.

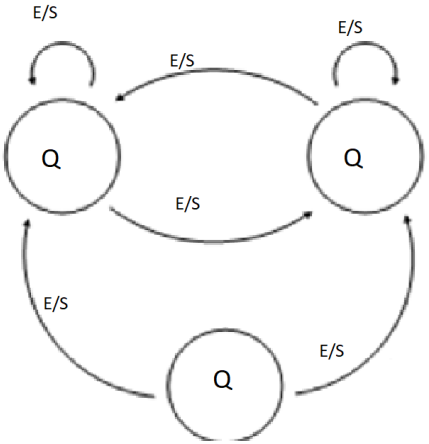
$g(\lambda, q_2)=1$, si el estado anterior es q_1.

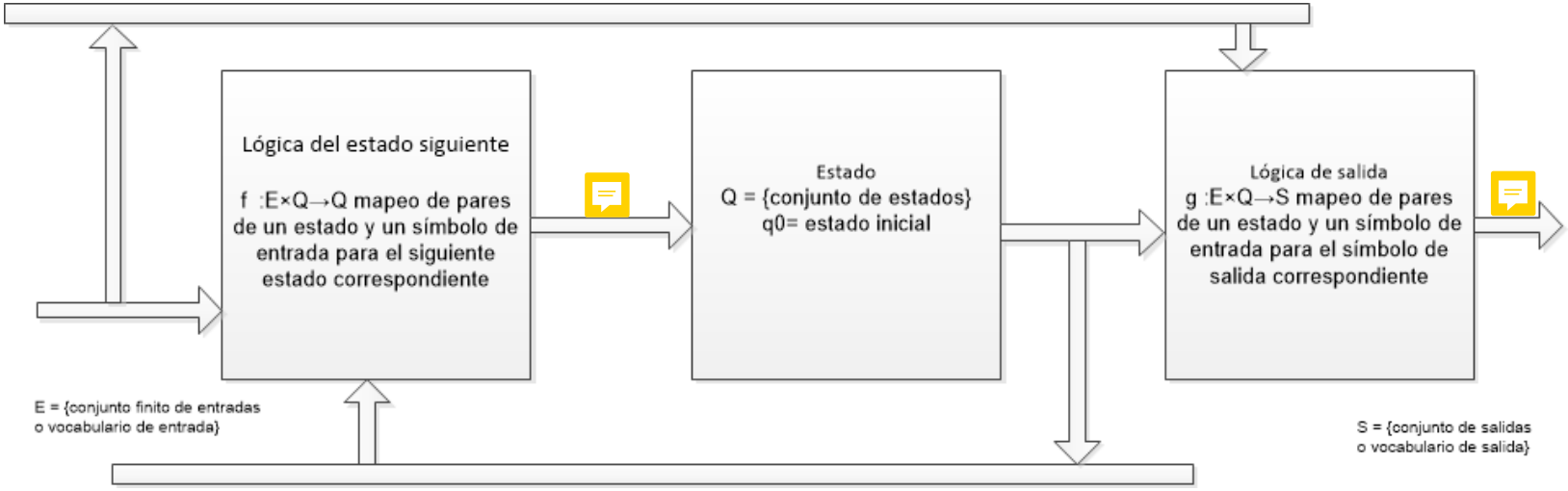
$g(\lambda, q_2)=0$, si el estado anterior es q_2.



Máquinas Mealy -Definición formal

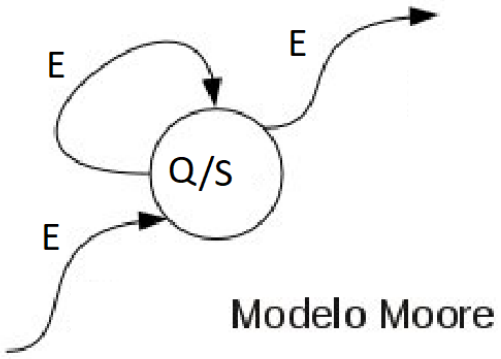
- **Autómata de Mealy , una 6-tupla =(E, S, Q, q0, f, g)**
  - E = {conjunto finito de entradas o vocabulário de entrada}
  - S = {conjunto de salidas o vocabulário de salida}
  - Q = {conjunto de estados}
  - q0= estado inicial
  - f :E×Q→Q
    - mapeo de pares de un estado y un símbolo de entrada para el siguiente estado correspondiente
  - g :E×Q→S
    - mapeo de pares de un estado y un símbolo de entrada para el símbolo de salida correspondiente



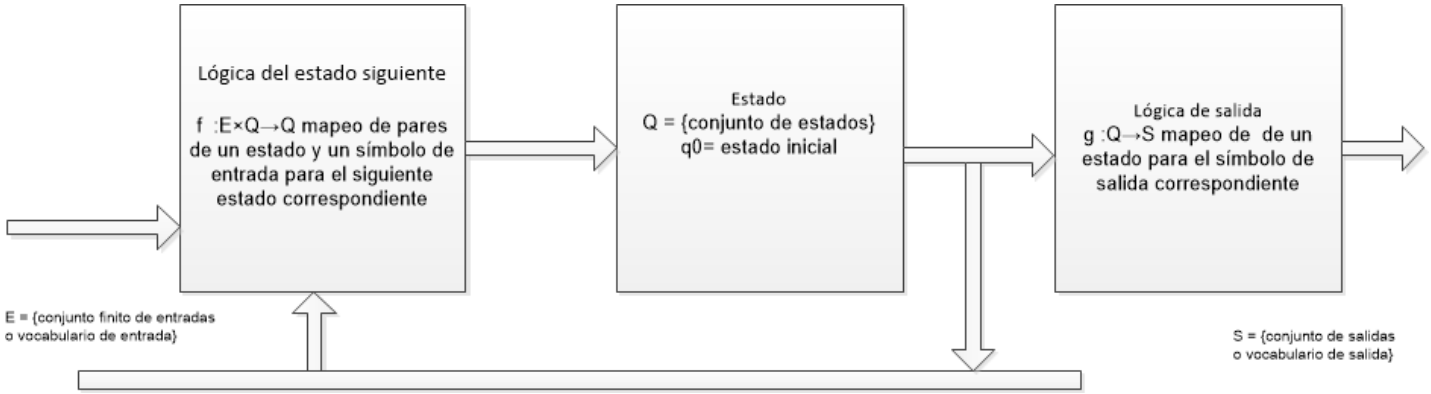


Máquinas de Moore -Definición formal

- **Autómata de Moore, una 6-tupla =(E, S, Q, q0, f, g)**
  - E = {conjunto finito de entradas o vocabulário de entrada}
  - S = {conjunto de salidas o vocabulário de salida}
  - Q = {conjunto de estados}
  - q0= estado inicial
  - f :E×Q→Q
    - mapeo de un estado y el alfabeto de entrada al siguiente estado
  - g :Q→S
    - mapeo de cada estado al alfabeto de salida



Q: Estado
E: Entrada
S: Salida



Comparación de máquinas Mealy y máquinas Moore

- Las máquinas de Mealy tienden a tener menos estados:
  - Diferentes salidas en arcos (n^2) en lugar de estados (n).
- Las máquinas Moore son más seguras de usar:
  - Las salidas cambian en el borde del reloj (siempre un ciclo más tarde).
  - En las máquinas Mealy, el cambio de entrada puede provocar un cambio en la salida tan pronto como se realiza la lógica, un gran problema cuando dos máquinas están interconectadas, la retroalimentación asíncrona puede ocurrir si no se tiene cuidado.
  - para una máquina Moore, cada nodo (estado) está etiquetado con un valor de salida;
  - para una máquina Mealy, cada arco (transición) está etiquetado con un valor de salida.
- Las máquinas Mealy reaccionan más rápido a las entradas:
  - Reaccionar en el mismo ciclo; no es necesario esperar el reloj.
  - En las máquinas Moore, es posible que se necesite más lógica para descodificar el estado en salidas: más retardos de puerta después del flanco del reloj.
- Como las máquinas Moore y Mealy son ambos tipos de máquinas de estado finito, son igualmente expresivas: cualquier tipo se puede usar para analizar un lenguaje regular
- No todos los circuitos secuenciales pueden implementarse utilizando el modelo Mealy. Algunos circuitos secuenciales solo pueden implementarse como máquinas Moore

- https://www.researchgate.net/publication/305268049_A_Note_on_Moore_Model_for_Sequential_Circuits

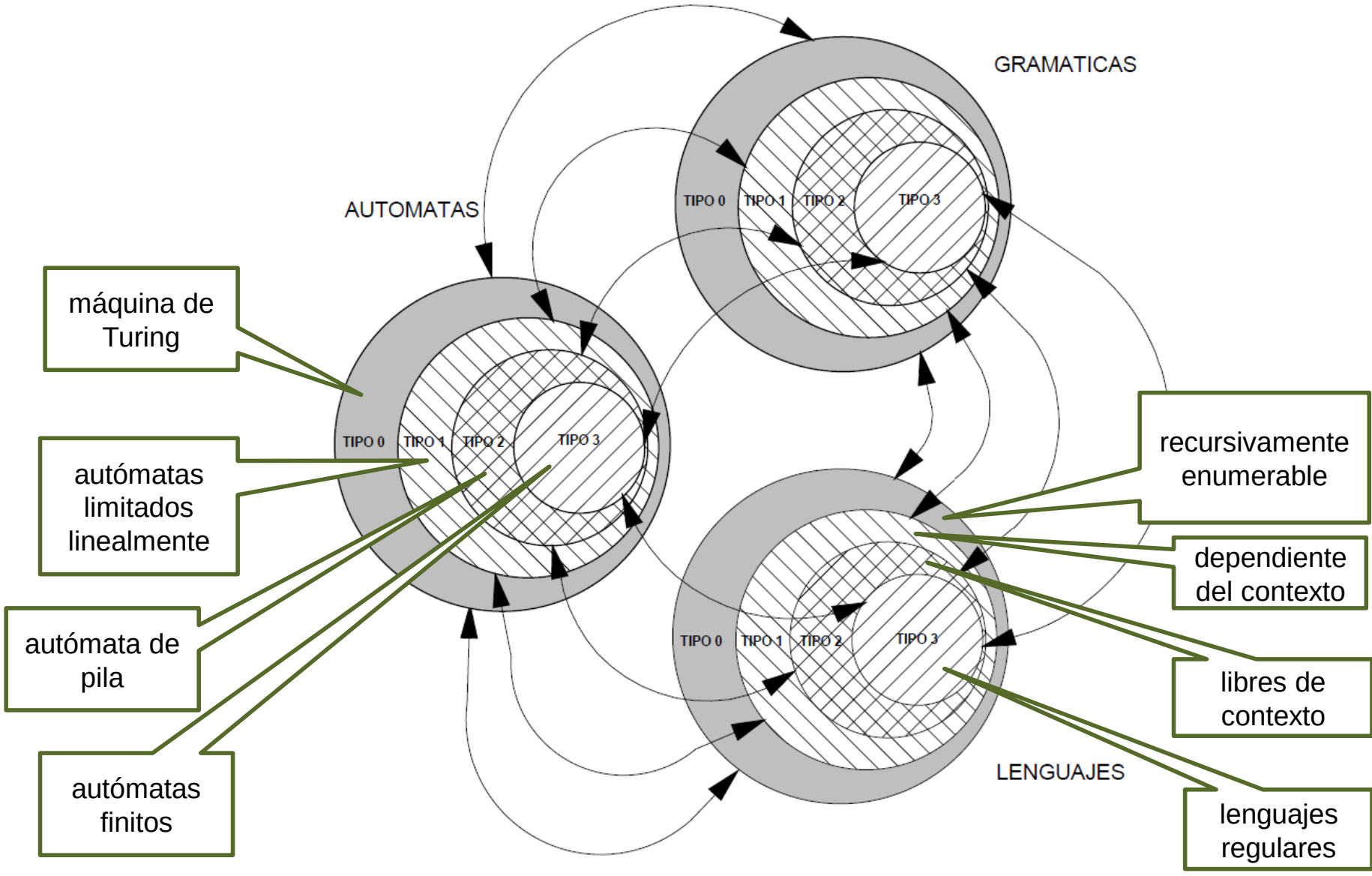
Autómatas

- **Estados accesibles de un autómata**
  - Sea un autómata A=(E,S,Q,f,g), se dice que un estado qj es accesible desde otro estado qi, si existe una entrada e E* tal que f(qi,e)=qj.
  - Evidentemente todo estado es accesible desde sí mismo, puesto que: $f(q_i,\lambda)=q_i$
- **Autómatas conexos**
  - Sea un autómata A=(E,S,Q,f,g), se dice que es *conexo* si todos los estados de Q son accesibles desde el estado inicial q0.
  - Dado un autómata no conexo se puede encontrar otro equivalente y conexo eliminando los estados inaccesibles.
  - Es evidente que los dos autómatas aceptarán el mismo lenguaje.
- **Autómatas deterministas y no deterministas**
  - Se denomina autómata determinista cuando la función de cambio de estado f es determinista. En caso contrario se dice no determinista

CAPÍTULO 9: JERARQUÍA DE LOS AUTÓMATAS

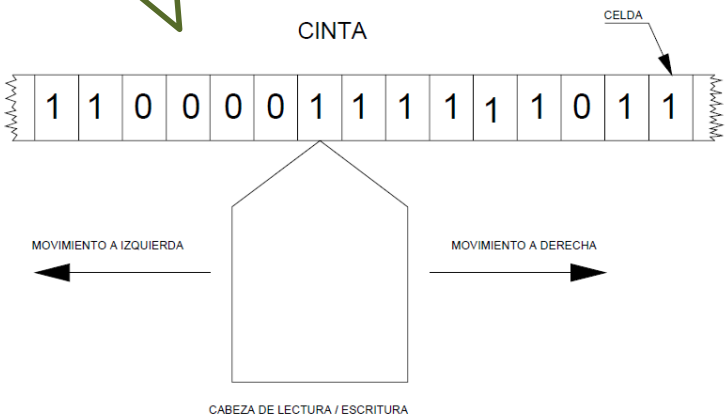
- Relación entre el tipo de lenguaje y la estructura de la máquina capaz de reconocerlo, pueden ser en dos sentidos :
  - a) Dada una gramática G ¿Qué estructura deberá tener una máquina, M, tal que el lenguaje reconocido por la máquina M, es igual al lenguaje generado por G?
  - b) Dada una máquina M ¿Cuál será la gramática G, tal que el lenguaje generado por G, es igual al lenguaje reconocido por M?

CORRESPONDENCIA ENTRE LOS LENGUAJES LAS GRAMATICAS Y LOS AUTOMATAS



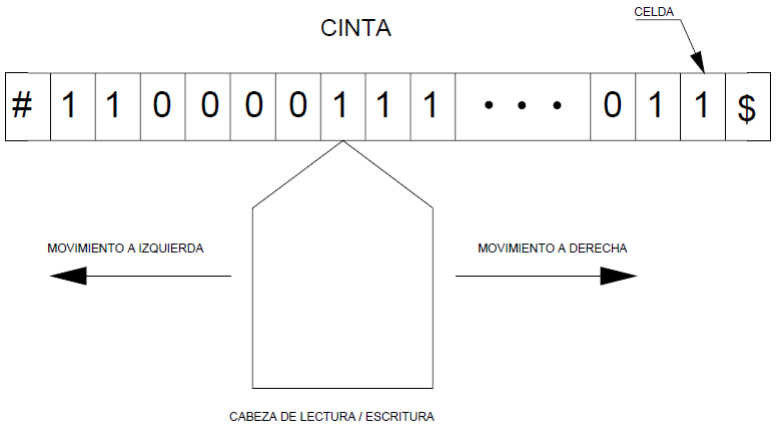
Distintos autómatas

reconocen los lenguajes recursivamente enumerable



máquina de Turing.

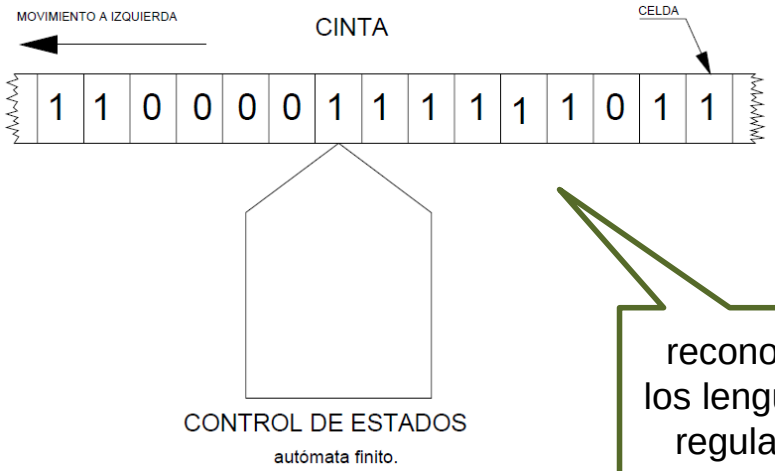
reconocen los lenguajes dependiente del contexto



autómata lineal acotado.

reconocer los lenguajes libres de contexto

autómata de pila.

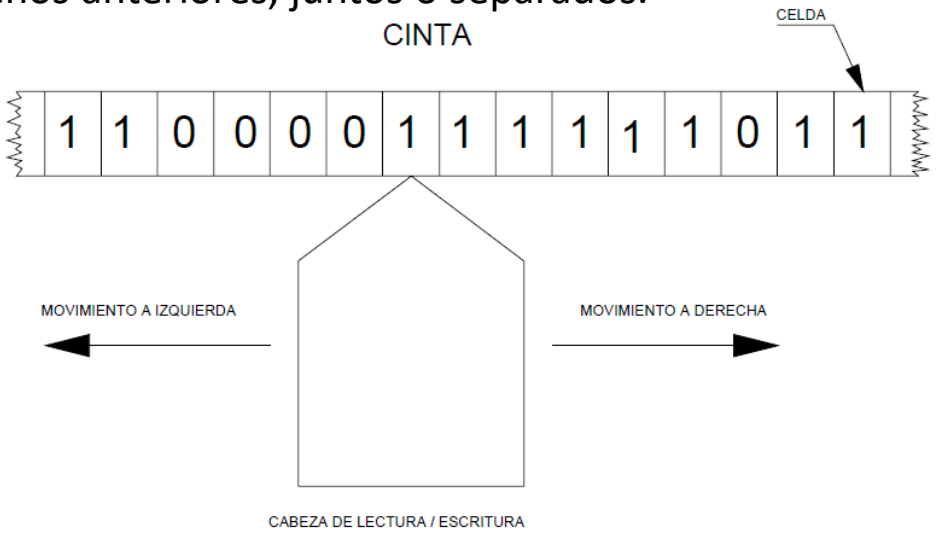


autómata finito.

reconocen los lenguajes regulares

CAPÍTULO 10: MÁQUINAS DE TURING

- Una máquina de Turing o autómata de tipo 0 es una construcción lógica, que se puede **representar intuitivamente** como un dispositivo mecánico, formado por:
  - una **cinta infinita**, dividida en celdas,
  - un cabezal de l**ectura/escritura** que se mueve sobre dicha cinta, avanzando una celda de cada vez.
    - En la figura se representa el caso particular de un conjunto de símbolos en la cinta formados por 0 y 1.
  - Un movimiento de la máquina de Turing, depende del símbolo explorado con la cabeza, y del estado actual en el que se encuentra la máquina, el resultado puede ser :
    - a) Cambio de estado
    - b) Imprime un símbolo en la cinta reemplazando el símbolo leído.
    - c) Se mueve la cabeza de la cinta a la izquierda, a la derecha o se para.
  - Pueden darse los tres fenómenos anteriores, juntos o separados.



máquina de Turing.

MÁQUINAS DE TURING

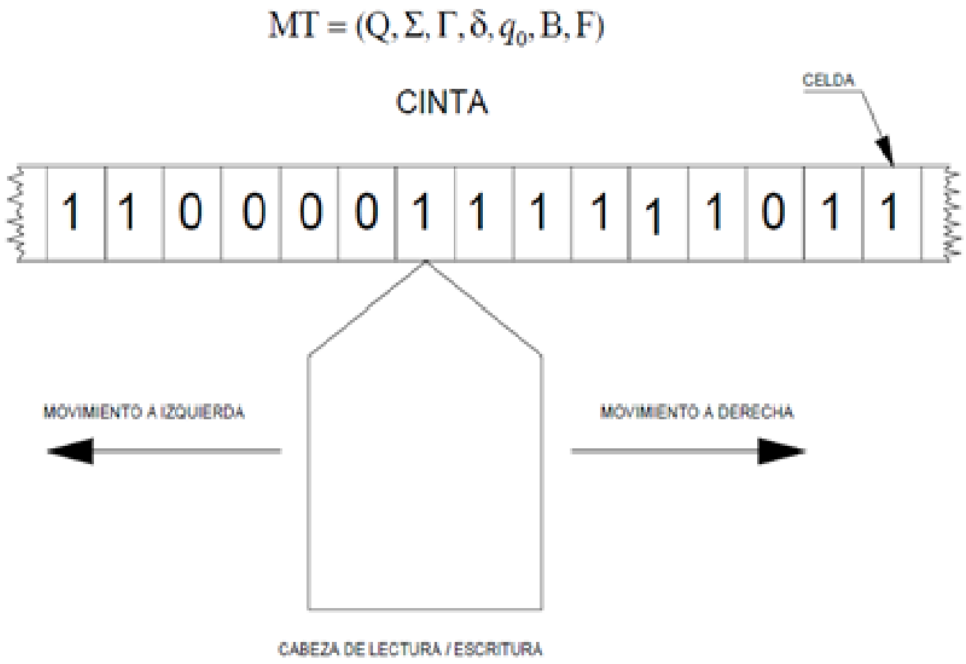
- Formalmente una máquina de Turing es un autómata, y como todo autómata está formado por una quíntupla

$$MT = (E, S, Q, f, g)$$

- sin embargo suele usarse la notación equivalente :

$$MT = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- Q = {conjunto de estados}
- Γ = {conjunto de símbolos permitidos en la cinta}
- B ∈ Γ es el símbolo blanco.
- Σ ∈ Γ es el subconjunto de símbolos de entrada no incluyendo el blanco.
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{I, D, S\}$ donde δ es la función del siguiente movimiento, I significa movimiento a izquierda, D significa movimiento a derecha, y S parada (stop).
- $q_0 \in Q$ es el estado inicial.
- $F \subset Q$ es el subconjunto de estados finales



Lenguaje aceptado

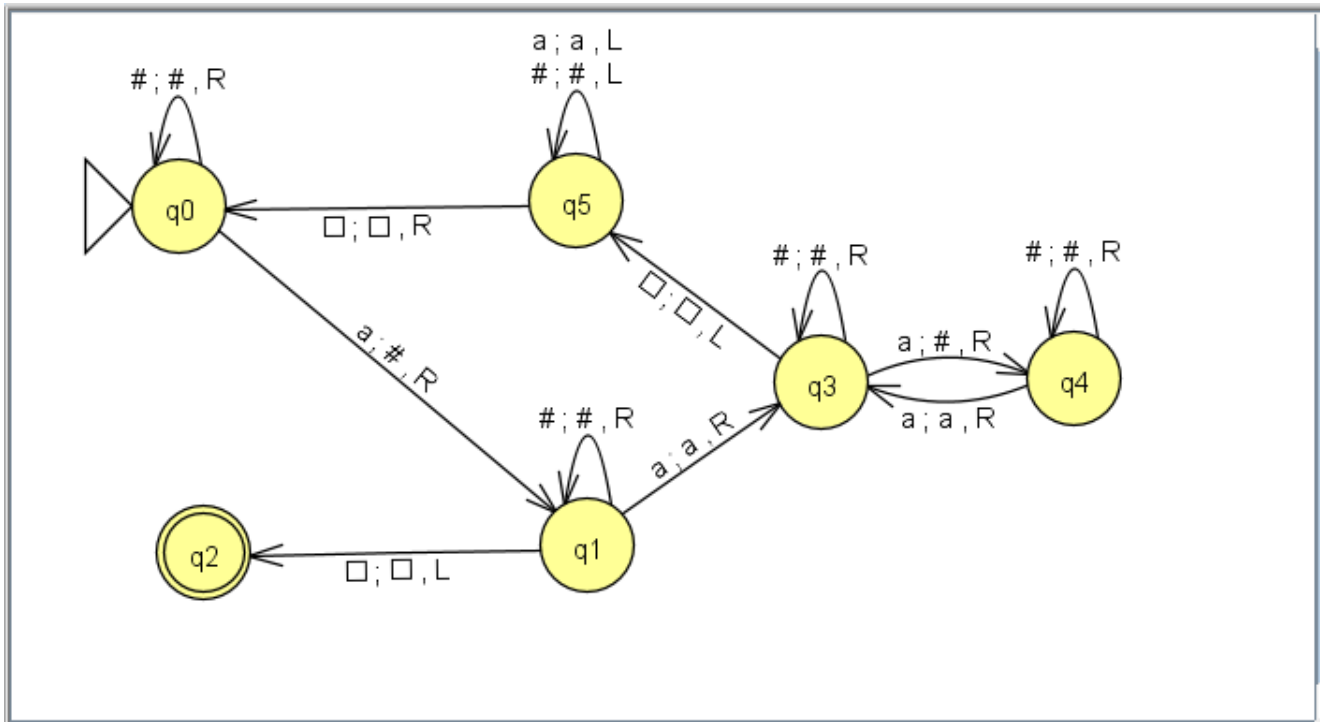
- El lenguaje aceptado o reconocido por una máquina de Turing, que denotaremos por L(MT), es el conjunto de palabras formadas con el alfabeto Σ^* , que hace que la máquina de Turing se pare al alcanzar un estado final
- **Teorema**
  - Para toda gramática de tipo 0, existe una máquina de Turing que reconoce el lenguaje generado por dicha gramática.
- **Teorema**
  - Para toda máquina de Turing, existe una gramática de tipo 0 que genera un lenguaje igual al reconocido por la máquina de Turing.
- **Corolario**
  - Existe una correspondencia entre gramáticas, lenguajes y autómatas de tipo 0, tal y como se mostró en el diagrama anterior.

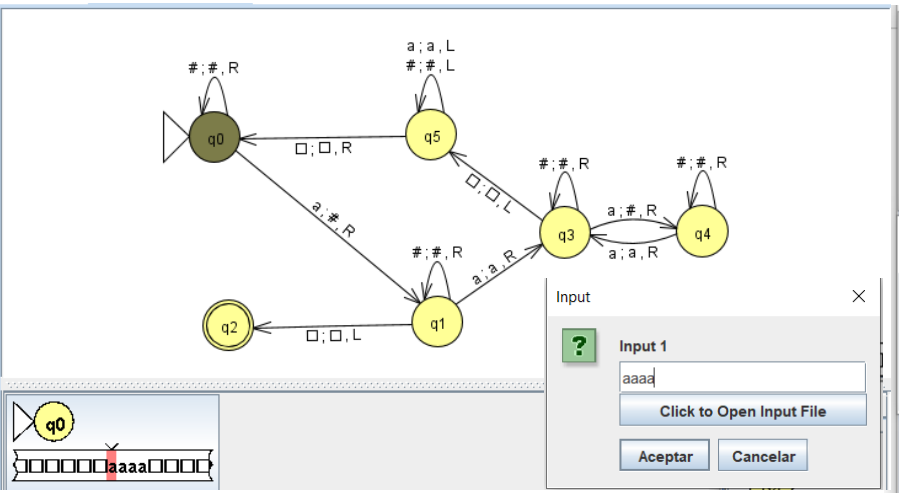
Ejemplo:
Turing Machines (JFLAP)

- Máquinas de cinta única
  - cada transición implica la lectura de un solo símbolo de una cinta
  - la sobreescritura de ese símbolo con un nuevo símbolo
  - el movimiento de la cabeza de la cinta ya sea a la izquierda, a la derecha o no.
  - Una transición se define como:
    - $r; w, d$. Donde
      r y w $\in$
      $\Gamma\ (alfabeto), d\ indica\ si\ el\ cabezal\ se\ mueve\ izquierda, derecha\ o\ se\ para$
      - r símbolo que se leer de la cinta
      - w símbolo para escribir en la cinta sobre la r
      - d es la dirección para mover la cabeza de la cinta después de escribir la w

Building a Turing Machine

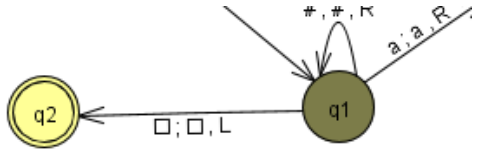
- Esta TM acepta el lenguaje definido por el conjunto $\{a^{2n}\}$
- El alfabeto es $\{a\}$
  - La estrategia es realizar una serie de pasos sobre la cinta de izquierda a derecha; en cada pase la mitad de las “a” se sobre-escriben con #.
  - Si el número de “a” es una potencia de dos, en una pasada hay exactamente una izquierda, en cuyo momento la TM acepta.

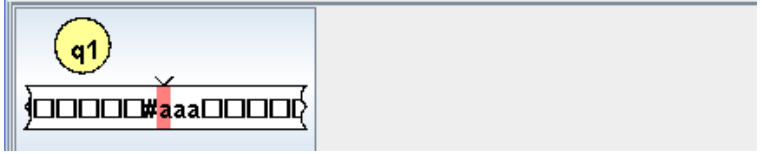


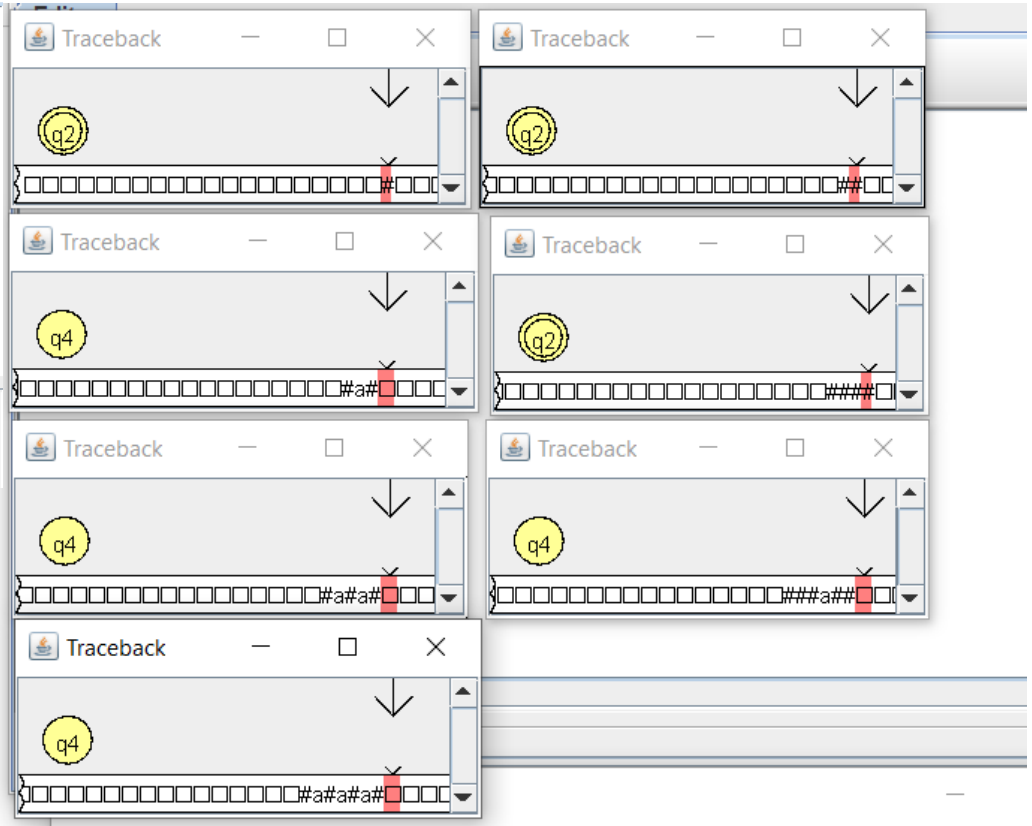

#;#,R
a;a,L
#;#,L
q0
q5
□;□,R
□;□,L
#;#,R
a;#,R
q3
a;#,R
q4
a;a,R
#;#,R
a;a,R
q2
q1
□;□,L

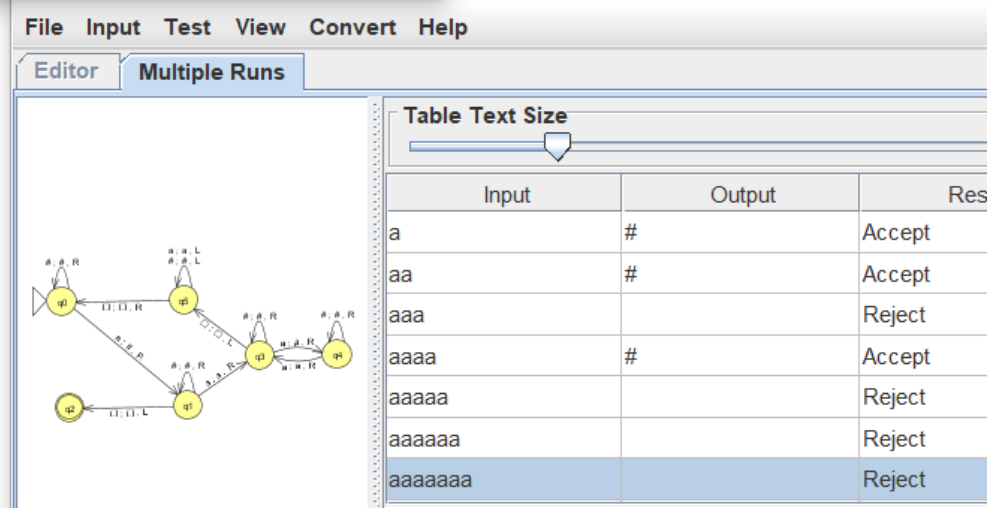
q0
aaaa

Input
Input 1
aaaa
Click to Open Input File
Aceptar
Cancelar


#;#,R
a;a,R
q2
q1
□;□,L


q1
#aaa


Traceback
q2
#
Traceback
q2
#
Traceback
q4
#a#
Traceback
q2
###
Traceback
q4
#a#a#
Traceback
q4
###a##
Traceback
q4
#a#a#a#


File
Input
Test
View
Convert
Help
Editor
Multiple Runs
Table Text Size
Input
Output
Res
a
#
Accept
aa
#
Accept
aaa
Reject
aaaa
#
Accept
aaaaa
Reject
aaaaaa
Reject
aaaaaaa
Reject

máquinas de Turing

- ¿Por que creemos que las máquinas de Turing son una buena formalización del concepto de algoritmo?
  - Porque cada programa de una máquina de Turing puede ser implementado
  - Porque todos los algoritmos conocidos han podido ser implementados en máquinas de Turing
  - Porque todos los otros intentos por formalizar este concepto fueron reducidos a las maquinas de Turing
  - Los mejores intentos resultaron ser equivalentes a las maquinas de Turing
  - Todos los intentos “razonables” fueron reducidos eficientemente

Tesis de Church: Algoritmo = máquina de Turing

"todo algoritmo es equivalente a una máquina de Turing“

No es un teorema matemático, es una afirmación formalmente indemostrable que, no obstante, tiene una aceptación prácticamente universal.

CAPÍTULO 11: AUTÓMATAS LINEALES ACOTADOS

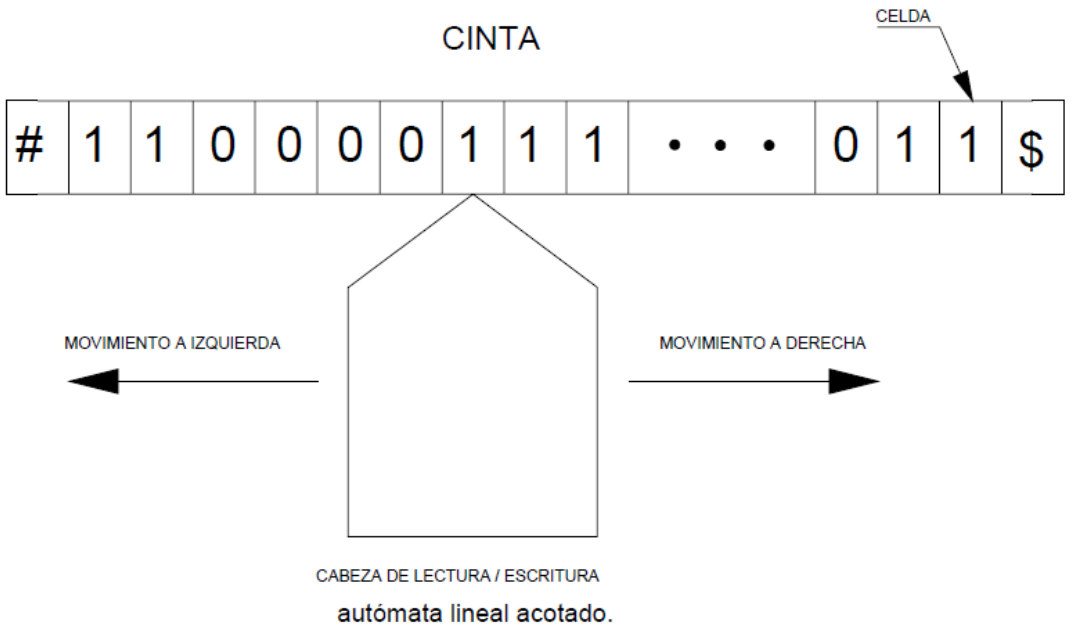
- Los autómatas de tipo 1 son los autómatas limitados linealmente o autómatas lineales acotados, en inglés *linear bounded automaton*.
- Un autómata lineal acotado es una máquina de Turing que satisface las siguientes condiciones :
  - a) El alfabeto de entrada incluye dos símbolos especiales # y $, que son las marcas fin de cinta, izquierda y derecha respectivamente.
  - b) La cabeza del autómata no puede desplazarse fuera de los limites izquierdo y derecho de la cinta, y no puede imprimir otro símbolo sobre # y $.
    - las marcas fin de cinta y comienzo de cinta no son consideradas como parte de la sentencia a reconocer.
  - c) Un autómata lineal acotado no puede moverse fuera de la cadena de entrada

- Se puede **definir formalmente** como una máquina de Turing con dos símbolos límite de la cinta.
- $ALA = \{Q, \Sigma, \Gamma, \delta, q0, \#, \$, F\}$

- Q = {conjunto de estados}
- Γ = {conjunto de símbolos permitidos en la cinta}
- $\Sigma \in \Gamma$ es el subconjunto de símbolos de entrada no incluyendo el blanco.
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{I, D, S\}$ donde δ es la función del siguiente movimiento, I significa movimiento a izquierda, D significa movimiento a derecha, y S parada (stop).
- $q_0 \in Q$ es el estado inicial.
- $F \subset Q$ es el subconjunto de estados finales
- # y $ $\in \Sigma$, correspondientes a la marca izquierda y derecha de la cinta

El lenguaje aceptado por un autómata lineal acotado,

$L(\Lambda L \Lambda) = \{ W / W \in \{ \Sigma - \{\#,\$\}\}^* \text{ y } q_0 \#W\$ \rightarrow \alpha q \beta \text{ para algún } q \in F\}$



AUTÓMATAS LINEALES ACOTADOS

- **Teorema**
  - Para toda gramática sensible al contexto G1 existe un autómata reconocedor lineal acotado R_{ALA}, tal que el lenguaje generado por la gramática L(G1), es reconocido por el autómata R_{ALA}.

$$L(G1) = L(R_{ALA})$$

- **Teorema**
  - Si $L(R_{ALA})$ es el lenguaje reconocido por un autómata lineal acotado, existe una gramática sensible al contexto G1, tal que el lenguaje reconocido por el autómata es igual al generado por la gramática.

$$L(R_{ALA}) = L(G1)$$

- **Corolario**
  - De los dos teoremas anteriores se deduce que existe una correspondencia entre las gramáticas de tipo 1, los lenguajes de tipo 1, y los autómatas lineales acotados.

CAPÍTULO 12: AUTÓMATAS DE PILA

- Un autómata de pila AP, en inglés *pushdown automata*, es un autómata capaz de reconocer los lenguajes libres de contexto, o de tipo 2.
- Los autómatas de pila se pueden representar como una máquina de Turing, que sólo puede leer de una cinta, y que puede guardar resultados intermedios en una pila.
- De hecho, su capacidad de procesamiento es inferior a los ALA, debido a las siguientes restricciones sobre las posibles operaciones con la cinta y la pila :
  - - La cinta se desplaza en un sólo sentido, y su cabeza sólo puede leer.
  - - La pila, está limitada en un extremo por definición, cuando se lee un elemento de la pila, este desaparece o se saca, y cuando se escribe en la pila, se introduce un elemento.

AUTÓMATAS DE PILA

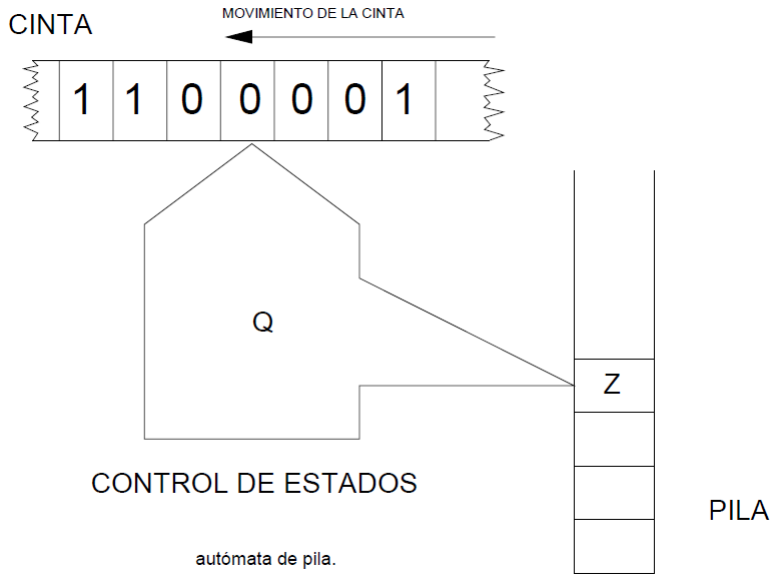
- Las operaciones elementales, que se pueden realizar con un AP, son de dos tipos :
  - ***Dependientes de la entrada*** : se lee $ei \in \Sigma$, y se desplaza la cinta, y en función de ei, qj (el estado en que se encuentra la cinta), y Z (el valor de la pila), el control pasa a otro estado ql, y en la pila se introduce Z', o se extrae Z, o no se hace nada.
  - ***Independientes de la entrada*** : puede ocurrir lo mismo que en el caso anterior, que ei no interviene, la cinta no se mueve, lo que permite manejar la pila sin las informaciones de entrada.
- En cualquier caso, si se vacía la pila (es decir se extrae todas las Z) el autómata se para.
- Un autómata de pila se puede representar intuitivamente según el esquema de la figura
- Un autómata de pila se puede definir formalmente como una séptupla :

$$AP = \{Q, \Sigma, \Gamma, \delta, q0, Z0, F\}$$

- Q es el *conjunto finito de estados*.
- Σ es el *alfabeto de entrada*, es finito.
- Γ es el *alfabeto de pila*.
- δ es la *función de transición*, y es una aplicación de la forma :

$$\delta : Q \times \{\Sigma \cup \{\lambda\}\} \times \Gamma \rightarrow Q \times \Gamma^*$$

- q_0 es el *estado inicial*, y cumple $q_0 \in Q$.
- Z_0 es el *símbolo inicial que contine la pila* antes de comenzar, evidentemente $Z_0 \in \Gamma$.
- F es el *conjunto de estados finales*, evidentemente $F \subset Q$.



autómata de pila.

AUTÓMATAS DE PILA

- Tal y como ha sido definida la función de transición , el autómata es en general *no determinista*.

$$\delta : Q \times \{\Sigma \cup \{\lambda\}\} \times \Gamma \rightarrow Q \times \Gamma^*$$

  - Se entiende por no determinista, cuando el resultado de la función no está determinado, es decir pueden resultar dos o más valores, sin que la función precise cual va a tomar de ellos.

- Se define **configuración** de un autómata de pila a su situación en un instante, que se puede expresar formalmente mediante el terceto :

$$(q, W, \alpha)$$

  - q representa el *estado actual del autómata*, y evidentemente $q \in Q$.
  - W es la *cadena de entrada que resta por analizar*, siendo $W \in \Sigma^*$; si $W = \lambda$, se asume que toda la cadena de entrada ya ha sido leída.
  - α es el *contenido de la pila*, en el instante considerado, y indica que $\alpha = \lambda$ la pila está vacia. Por supuesto $\alpha \in \Gamma^*$.

AUTÓMATAS DE PILA

- Se entiende por **movimiento** de un autómata a una transición entre configuraciones, y se representa por el operador binario $\rightarrow$.
- Así por ejemplo sea el siguiente movimiento :

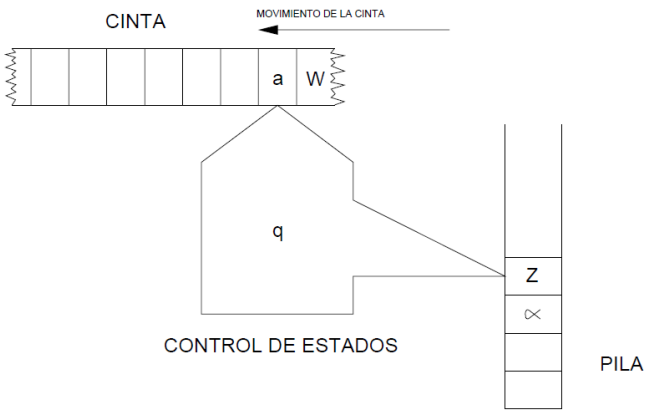
$$(q, aW, z\alpha) \rightarrow (q', W, \Upsilon\alpha)$$

$$q \in Q, a \in \Sigma \cup \{\lambda\}, W \in \Sigma^*, Z \in \Gamma \; y \; \alpha \in \Gamma^*$$

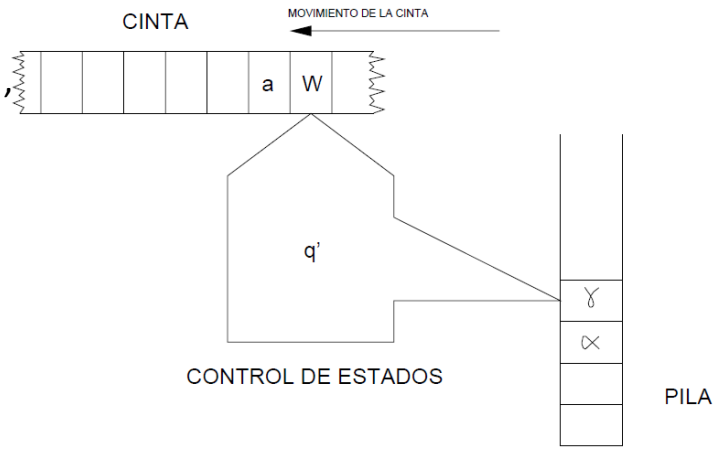
- donde la función de transición para esta entrada toma el valor:

$$\delta(q, aW, Z) \rightarrow (q', \Upsilon)$$

- Es decir que el autómata se encuentra en el estado q, que la cabeza de lectura de la cinta se encuentra sobre el símbolo a, y que la pila contiene determinados símbolos representados por la concatenación de Z y α , siendo el situado más a la izquierda Z el que se encuentra en cabeza de la pila



El autómata pasa de la configuración a la configuración , es decir pasa a un estado q', avanzando la cabeza de lectura al siguiente símbolo y procediendo a realizar determinadas sustituciones en la cabeza de la pila



Lenguaje reconocido por un autómata de pila

- Se puede definir de dos formas :
- a) El autómata reconoce una cadena de entrada, cuando alcanza el estado final, es decir el lenguaje reconocido por un autómata de pila se puede expresar de la siguiente forma :

$$L(AP) = \{W/W \in \Sigma^* \quad y \quad (q_0, W, Z_0) \rightarrow (q_f, \lambda, \alpha)\}$$

- b) El autómata reconoce la cadena cuando la pila queda vacía, independientemente del estado al que se llegue, entonces el lenguaje reconocido por el autómata es :

$$L(AP) = \{W/W \in \Sigma^* \quad y \quad (q_0, W, Z_0) \rightarrow (q', \lambda, \lambda)\}$$

- Se puede demostrar que ambas definiciones de L(AP), son equivalentes, en el sentido de que la clase de lenguajes aceptados por los autómatas de pila es la misma en ambos casos.

Lenguaje reconocido por un autómata de pila

- **Teorema**
  - Para toda gramática libre de contexto G2, existe un reconocedor constituido por un autómata de pila RAP, tal que el lenguaje generado por la gramática L(G2) es reconocido por el autómata RAP.

$$L(G_2)=L(R_{AP})$$

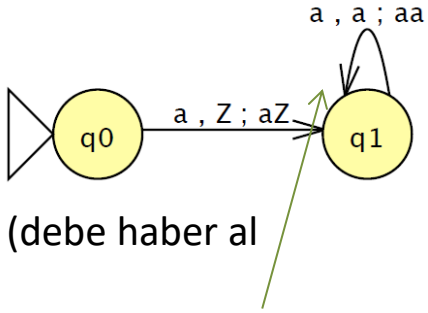
- **Teorema**
  - Para todo reconocedor constituido por un autómata de pila RAP, existe una gramática libre de contexto G2, tal que el lenguaje reconocido por el autómata es igual al generado por la gramática.

$$L(R_{AP})=L(G_2)$$

- **Corolario**
  - De los dos teoremas anteriores se deduce que el conjunto de lenguajes reconocidos por los autómatas de pila, son los lenguajes de tipo 2 y que todo lenguaje de tipo 2 se puede reconocer por un autómata de pila. También se puede deducir que existe una correspondencia entre las gramáticas, los lenguajes y los autómatas de tipo 2.

Como crear un Pushdown Automata

- **Nondeterministic NPDAs**
  - Los NPDA no deterministas funcionan de manera similar.
  - Son no determinísticos si:
    - un estado donde $con\ una\ \lambda - transition\ (\lambda, \lambda; \lambda\ )$
    - Dos estados de salida con el mismo símbolo para la entrada y el pop
      - Esto es determinista a, **b**; λ and a, **a**; λ
      - Esto no es determinista **a, ab**; λ and **a, ab**; λ
  - Queremos construir un automata no determinístico NPDA que reconozca la formula $\{a^n b^m \,|n\ >\ 0, n \leq\ m\ \leq 3n\}$
    - Por cada "a" procesada, debes saber cuántas "b" (1, 2 o 3) serán procesadas.
    - Pero para sabes el número de “b” por “a” hay que examinar toda la cadena.
    - Por lo tanto, un PDA no puede definir este lenguaje es necesario un NPDA.
  - Por ejemplo, estas cadenas deben ser aceptadas
    - aabb, aabbb, aabbbb, aabbbbbbb, y aabbbbbb
    - para cada "a" debe haber una a tres “b”
  - Estas cadenas deben ser rechazadas
    - bba (a no es primero), bab (a no es primero), abbbb (demasiadas b), aa (debe haber al menos una b), y bb (debe haber al menos un a)

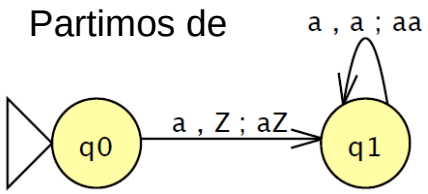


Como las "a" vienen primero, crear un estados que empujen todas las "a" a la pila

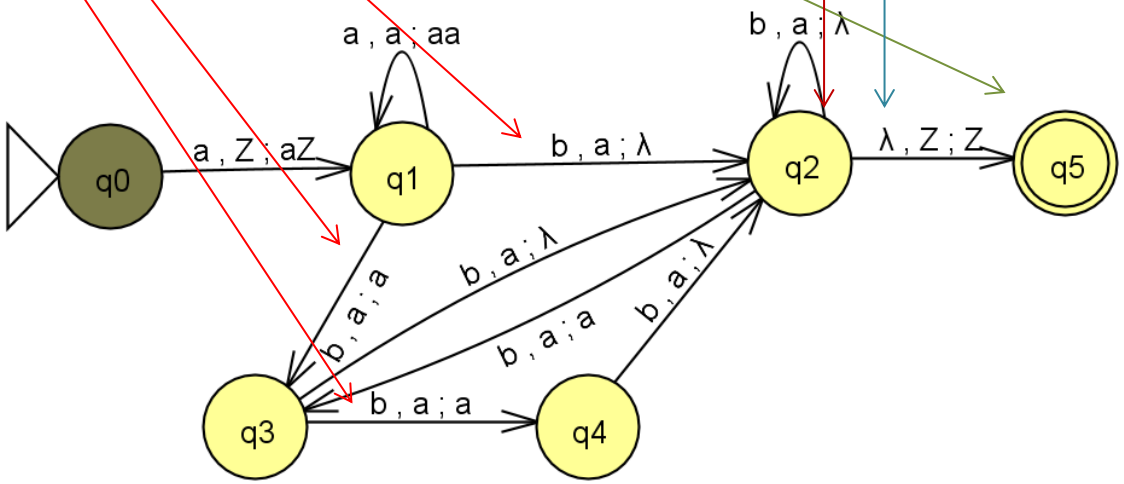
Como crear un Pushdown Automata

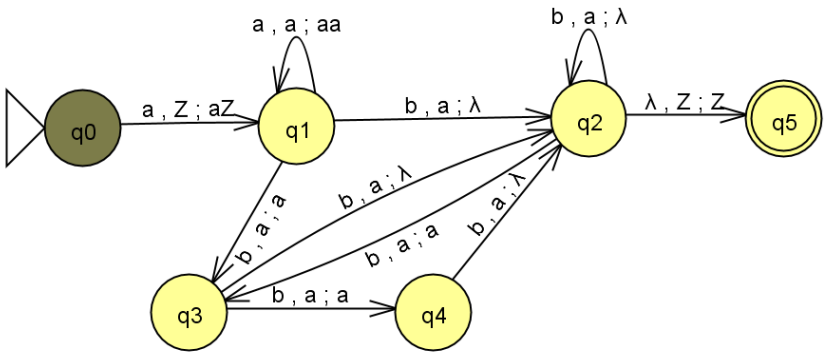
- **Nondeterministic NPDAs**
  - Queremos detectar la siguiente formula

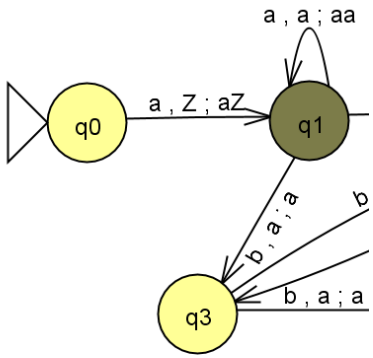
$$\{a^n b^m \mid n > 0, n \leq m \leq 3n\}$$

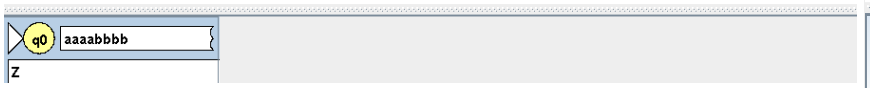


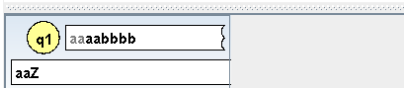
    - para cada “a” debemos tener de una a tres “b”
    - Después de que todas las "a" están en el stack
    - Debemos tener tres caminos que salgan uno a uno y terminen en el mismo punto; un camino para la lectura una “b”, un camino para leer dos “b”, y un camino para leer tres “b”, por cada “a”.
    - En el punto donde los tres caminos terminan, necesitamos tener tres bucles: uno para emparejar una “a” con una “b”, otro para cada “a” con dos “b”, y una con una “a” con tres “b”.
    - Cuando la pila está vacía, la cadena es aceptada si toda la entrada ha sido procesada.
    - Añadimos uno estado y arco final como se muestra en la figura
  - ¿Qué estados son no deterministas?

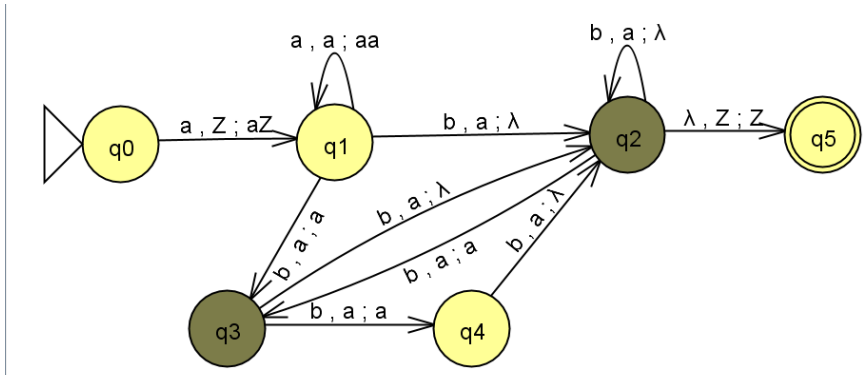


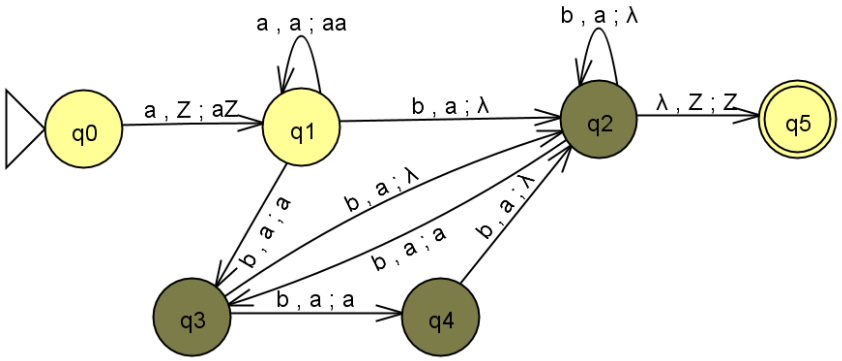

a , a ; aa
b , a ; λ
a , Z ; aZ
b , a ; λ
λ , Z ; Z
q0
q1
q2
q5
b , a ; a
b , a ; λ
b , a ; a
b , a ; λ
q3
b , a ; a
q4

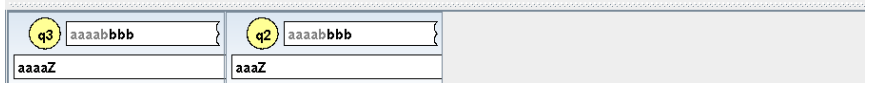

a , a ; aa
a , Z ; aZ
q0
q1
b , a ; a
b
q3
b , a ; a


q0
aaaabbbb
Z

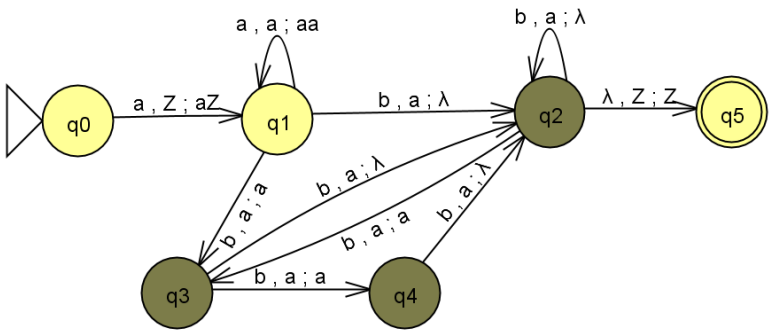

q1
aaaabbbb
aaZ

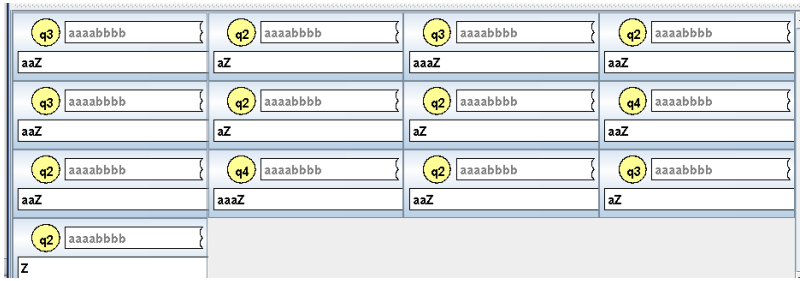

a , a ; aa
b , a ; λ
a , Z ; aZ
b , a ; λ
λ , Z ; Z
q0
q1
q2
q5
b , a ; a
b , a ; λ
b , a ; a
b , a ; λ
q3
b , a ; a
q4

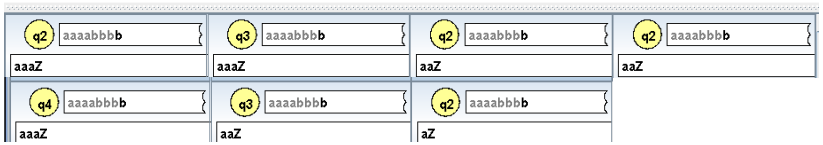

a , a ; aa
b , a ; λ
a , Z ; aZ
b , a ; λ
λ , Z ; Z
q0
q1
q2
q5
b , a ; a
b , a ; λ
b , a ; a
b , a ; λ
q3
b , a ; a
q4

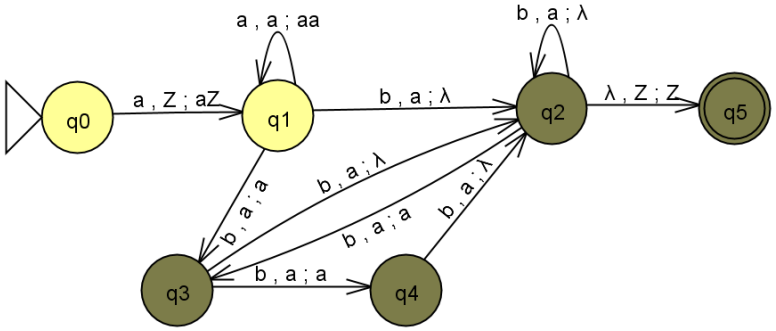

q3
aaaabbbb
aaaaZ
q2
aaaabbbb
aaaZ

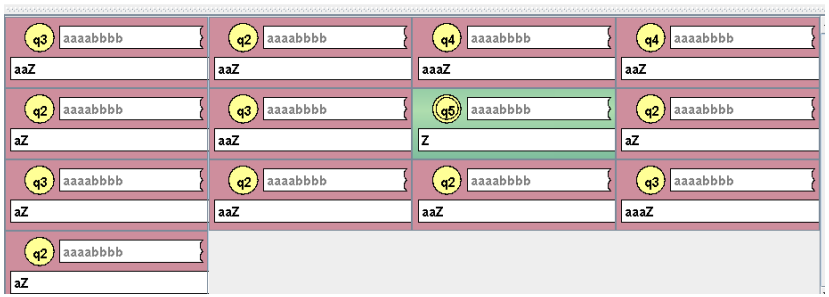

q2
aaaabbbb
aaaZ
q4
aaaabbbb
aaaaZ
q3
aaaabbbb
aaaZ
q2
aaaabbbb
aaZ


a , a ; aa
b , a ; λ
a , Z ; aZ
b , a ; λ
λ , Z ; Z
q0
q1
q2
q5
b , a ; a
b , a ; λ
b , a ; a
b , a ; λ
b , a ; a
q3
q4

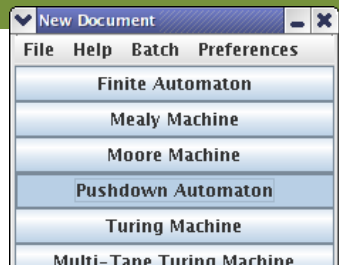





a , a ; aa
b , a ; λ
a , Z ; aZ
b , a ; λ
λ , Z ; Z
q0
q1
q2
q5
b , a ; a
b , a ; λ
b , a ; a
b , a ; λ
b , a ; a
q3
q4



Ejercicio: crear un Pushdown Automata



- Pushdown Automata deterministic PDA for the language

$$L = \{a^n b^n : n > 0\}$$

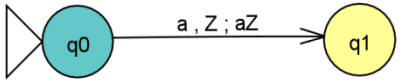
  - Para cada "a" de entrada enpujamos una "a" en la pila, y luego sacamos una "a" de la pila para cada "b" en la entrada.

1. recuadro entrada a procesar
2. recuadro el valor actual en la parte superior de la pila,
3. representa el nuevo valor a empujar en la parte superior de la pila después de sacar el valor en la parte superior de la pila.

No hay límite para el tamaño de los valores en ninguna de estas cajas.

"a" primer cuadro
"Z" segundo cuadro (el carácter por defecto para el elemento inferior de la pila), "aZ" en el tercer cuadro.

Esta transición significa: Si el primer símbolo de la entrada es "a", el primer símbolo de la pila es "Z", y la máquina está en el estado "q0", entonces saque "Z" y presione "aZ" en la pila. Por lo que se añade una "a" a la pila.



El autómata $M = (Q, \Sigma, \Gamma, \delta, q_s, Z, F)$

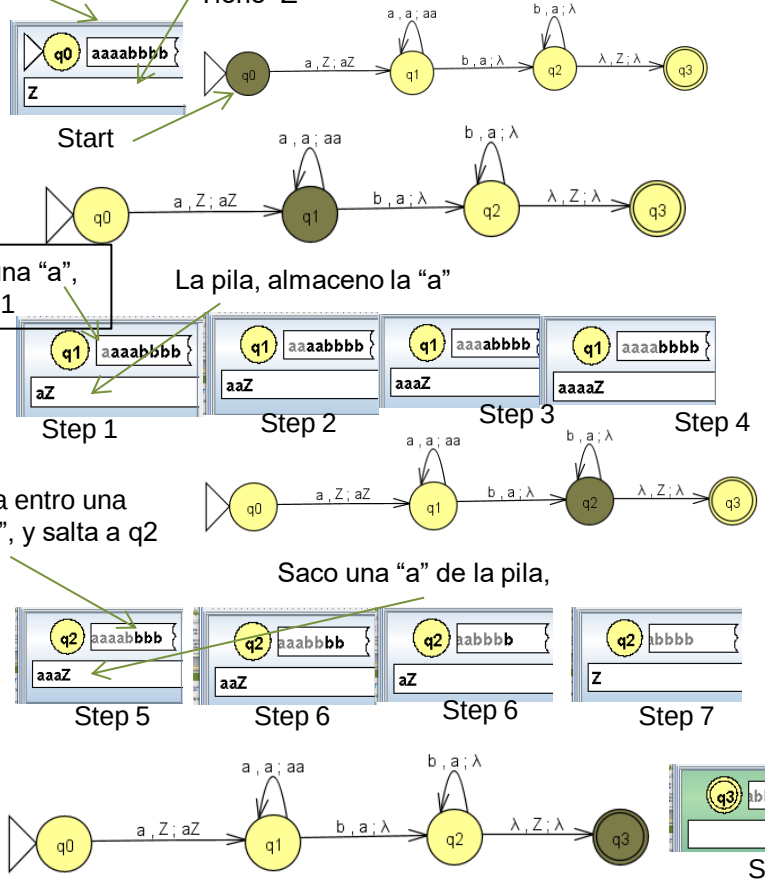
- $Q = \{q0, q1, q2, q3\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{a, Z\}$
- $\delta = \{(q0, a, Z, q1, aZ), (q1, a, a, q1, aa), (q1, b, a, q2, \lambda),$
- $(q2, b, a, q2, \lambda), (q2, b, Z, q3, Z), (q3, b, Z, q3, Z)\}$
- $qs = q0$
- $F = \{q3\}$

cinco-tuplas

{q actual, $\Sigma_{entrada}$, $\Gamma_{saco\ pila}$, q próximo , $\Gamma_{pongo\ pila}$ }

String de entrada

La pila,
en el estado inicial
Tiene "Z"

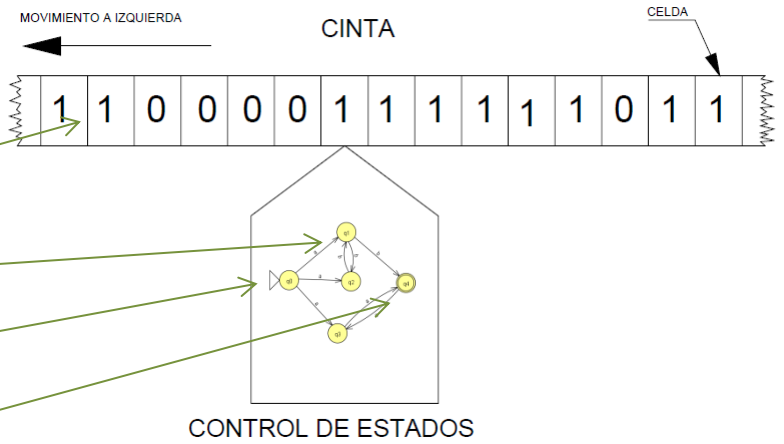


CAPÍTULO 13: AUTÓMATAS FINITOS

- Los autómatas finitos reconocen los lenguajes regulares, o de tipo 3 y se pueden representar intuitivamente por una cinta y una cabeza de lectura
  - La ***cinta de entrada***, sólo contiene **símbolos de un determinado alfabeto**, y **se mueve en una sola dirección**.
  - El ***control de estados***, determina el funcionamiento del autómata.
  - **Una sentencia** de un lenguaje determinado, colocada en la cinta, y leída por el autómata finito, es reconocida por éste, si el control de estados llega a un estado final

- Un autómata finito se puede definir formalmente como una séptupla :

$$AF = \{E, Q, f, q1, F\}$$



E = {conjunto finito de símbolos de entrada, que constituye el vocabulario}

Q = {conjunto finito de estados}

$f: E^* \times Q \rightarrow Q$ es la *función de transición*

$q_1 \in Q$, es el *estado inicial*

$F \subset Q$ es el *conjunto de estados finales*

AUTÓMATAS FINITOS

- **Teorema**
  - Para toda gramática regular, G_3, existe un autómata finito, AF, tal que el lenguaje reconocido por el autómata finito es igual al lenguaje generado por la gramática.

$$L(AF) = L(G_3)$$

- **Teorema**
  - Para todo autómata finito, AF, existe una gramática regular, G3, tal que el lenguaje generado por la gramática es igual al lenguaje reconocido por el autómata finito

$$L(G_3) = L(AF)$$

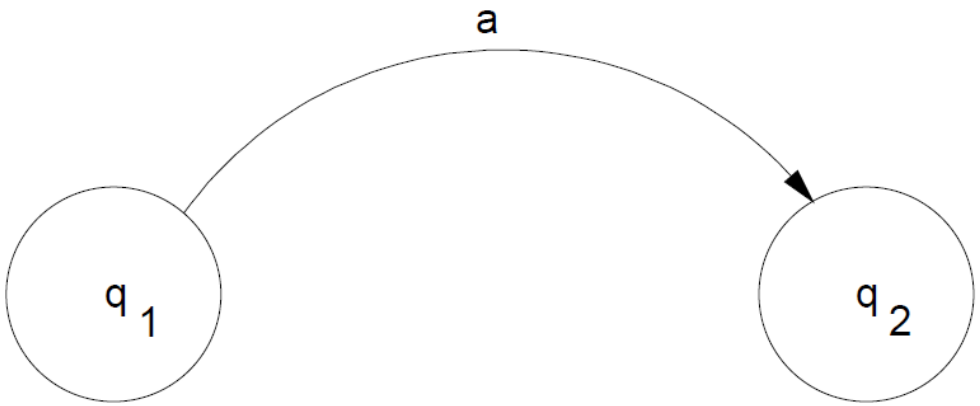
- **Corolario**
  - Según el primer teorema, se tiene que $\{L(G3)\} \subset \{L(AF)\}$ y por el segundo teorema $\{L(AF) \subset \{L(G3)\}$, luego

$$\{L_{regulares}\} = \{L(AF)\} = \{L(G_3)\}$$

AUTÓMATAS FINITOS

- La forma habitual de representar los autómatas finitos es mediante un grafo o diagrama de estados, donde los nodos son los estados y las ramas están marcadas con los símbolos del alfabeto de entrada.
- Las ramas se construyen según la función de transición, así debe de cumplir $f(q_1, a) \rightarrow q_2$

  - Los nodos que representan los *estados finales*, suelen marcarse con un doble círculo, y
  - el *estado inicial* también se marca con una flecha, encima de la cual se coloca la palabra INICIO.



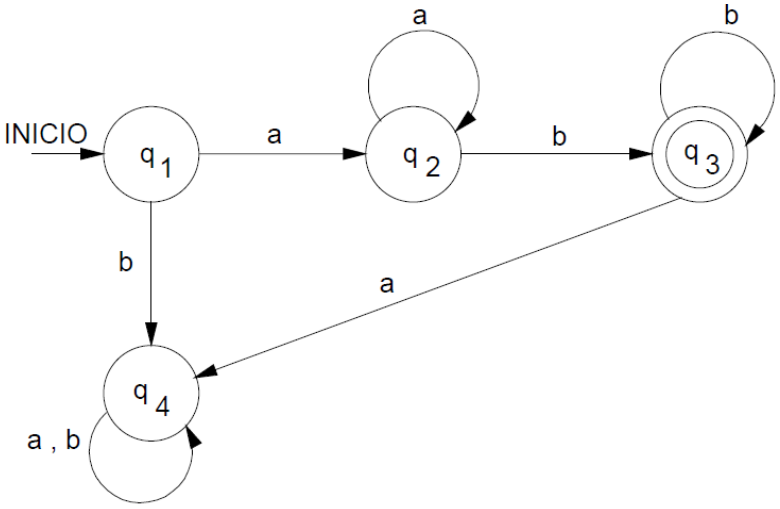
Transición entre dos estados.

Ejemplo

- Sea el autómata finito $AF = (E, Q, f, q1, F)$ donde $E = \{a, b\} \cup \{\lambda\}$: $Q = \{q1, q2, q3, q4\}$
- y la función f viene dada por la tabla siguiente y el conjunto de estados finales es $F = \{q3\}$

- Determinar el lenguaje que reconoce, representar el diagrama de Moore, e indicar la expresión regular que representa al lenguaje.

f	a	b
q_1	q_2	q_4
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_4	q_4



El lenguaje generado se obtiene partiendo del estado inicial y recorriendo todos los caminos posibles para alcanzar el estado final. Así se obtiene que este autómata reconoce el lenguaje :

$$L(A1) = \{ab, aab, \dots, abbb, \dots, aabb, \dots\}$$

$$L(A1) = \{a^n b^m / n \geq 1, y\ m \geq 1\}$$

Clasificación de los autómatas finitos

- Cuando se definió autómata finito, la función , es en general no determinista.
- Así en función de $f: E^* xQ \rightarrow Q$, se hablará de *autómatas finitos deterministas AFD* y *autómatas finitos no deterministas AFND*.
- Un autómata finito no determinista AFND se caracteriza por la posibilidad de que dada una entrada *e* en un estado q*i*, se pueda pasar a un estado qj, qk,...,qn sin saber a ciencia cierta, a cual de esos estados pasará.
- Existiendo la misma probabilidad de que pase a cualquiera de dichos estados.

Autómatas finitos no deterministas

- La definición de autómata finito no determinista AFND coincide con la de autómata finito (pero)#
- $AFND = (E, Q, f, q_i, F)$
  - E = {conjunto de entradas o vocabulário de entrada}
  - Q = {conjunto de estados} es el conjunto de estados posibles, puede ser finito o infinito.
  - f :$E^*xQ \rightarrow$P(Q), esta función es no determinista o función no determinista del estado siguiente#

    donde **P(Q)** es el **conjunto potencia de Q**

    (por ejemplo se A={1,2,3} en conjunto de estados,
    sus conjuntos potencial serán P(A)={vacío, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}. {1,2,3}}

    Para cada pareja estado-símbolo puede no definirse un único estado, sino un conjunto de ellos.

    El autómata finito no determinístico tiene la capacidad de estar **en varios estados a la vez**

    E^*xQ devuelve un estado perteneciente al conjunto Q.
  - q_i Es el estado inicia; El estado inicial puede no ser único #.
    - los AFND a menudo son más compactos y fáciles de diseñar que los AFD, además siempre es posible convertir un AFND en un AFD
    - La máquina comienza en el estado inicial especificado y lee una **cadena de caracteres** pertenecientes al alfabeto.
    - El autómata utiliza la función de transición F de estados para determinar el siguiente estado, usando el estado actual y **el símbolo que acaba de leer o la cadena vacía**.
    - El estado inicial puede no ser único y que para cada pareja estado-símbolo puede no definirse un único estado, sino un conjunto de ellos..
    - Hasta que se producen estos acontecimientos posteriores no es posible determinar en qué estado se encuentra la máquina" . Cuando el autómata ha terminado de leer, y se encuentra en un estado de aceptación, se dice que el AFND acepta la cadena, de lo contrario se dice que la cadena de caracteres es rechazada.

Como crear un autómata finito no determinístico

Jflap

- Crear un AFND es parecido a crear un AFD
  - El autómata del ejemplo acepta el lenguaje que conforma el conjunto

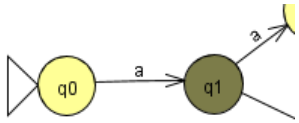
$$\{a^n b^m | \, n > 0 \wedge m \geq 0, \text{a y b son divisibles por 2 o 3}\}$$

    - El primer paso es crear los trece estados del autómata, y hacer q0 el estado inicial y hacer q6 y q11 los estados finales.

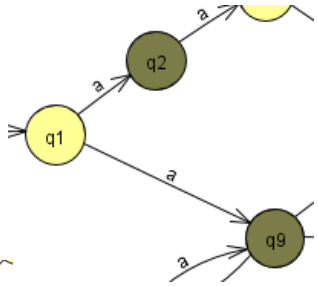
Simulación, la entrada en una máquina no determinista puede producir múltiples vías de configuraciones.

las cuatro transiciones de la Figura con un λ. Estas transiciones λ son transiciones en la cadena vacía.

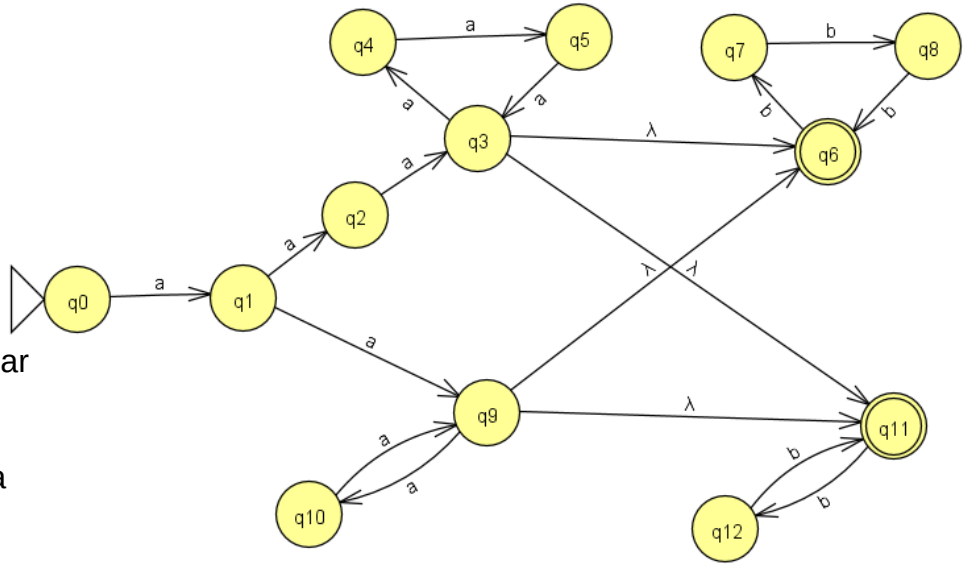
1-Sea la siguiente entrada “aaabbb”, la primera ‘a’ nos lleva a q1

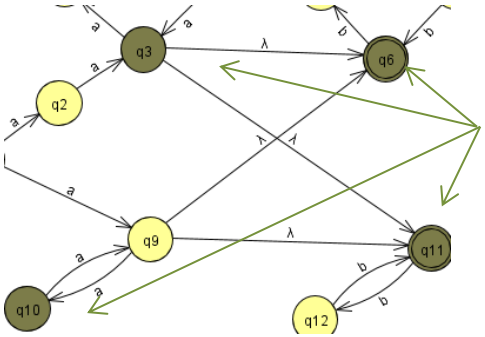


2-Pero la segunda ‘a’, nos lleva a q2



3-Mientras que la tercer ‘a’ nos lleva a q3, q6, q10 y q11
Tanto q6 y q11 son debido a λ



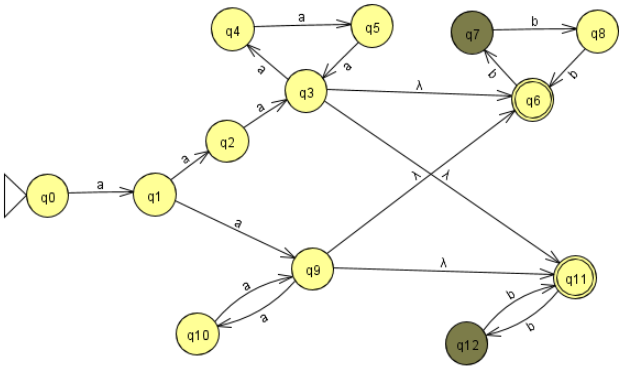


Como podemos apreciar existen estados simultáneos, par (3) el autómata se encuentra en q11, q6, q10, q3

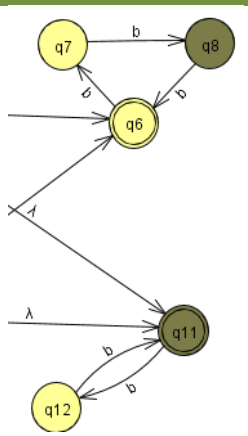
para poner una λ, en jplap, cree una transición y deje el campo vacío

Como crear un autómata finito no determinístico

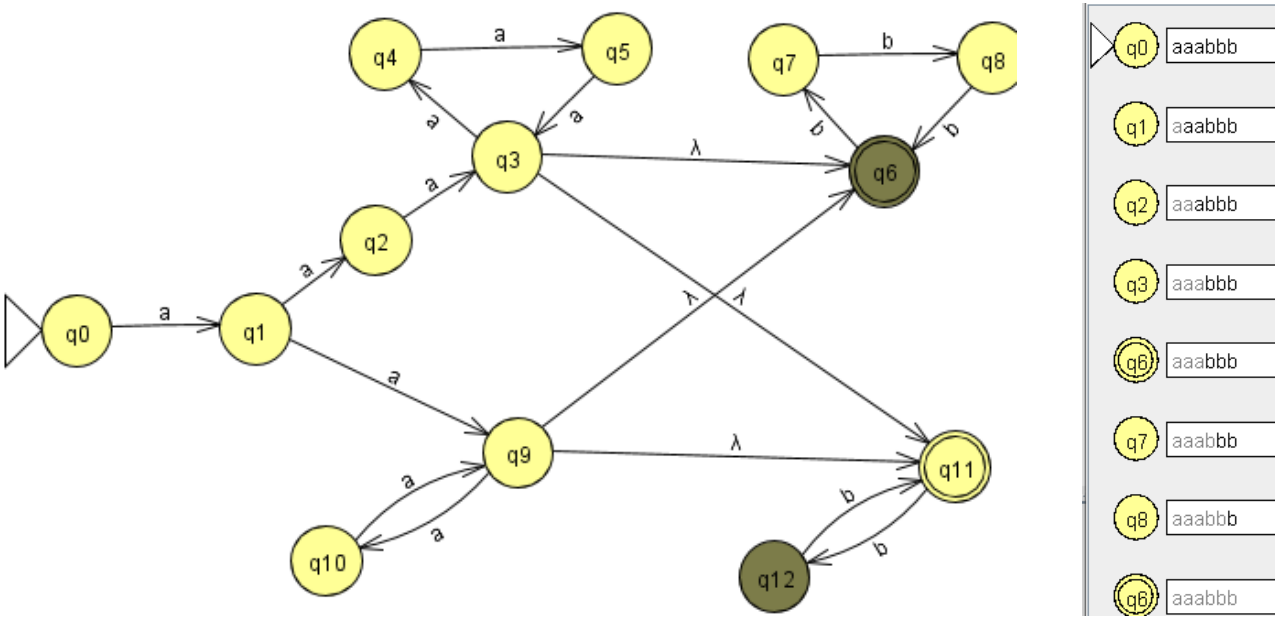
4-Después de las “aaa” y la primera b



5-Después de las “aaa” y la segunda b

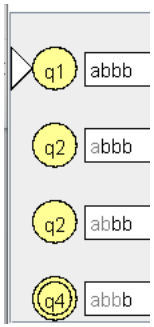


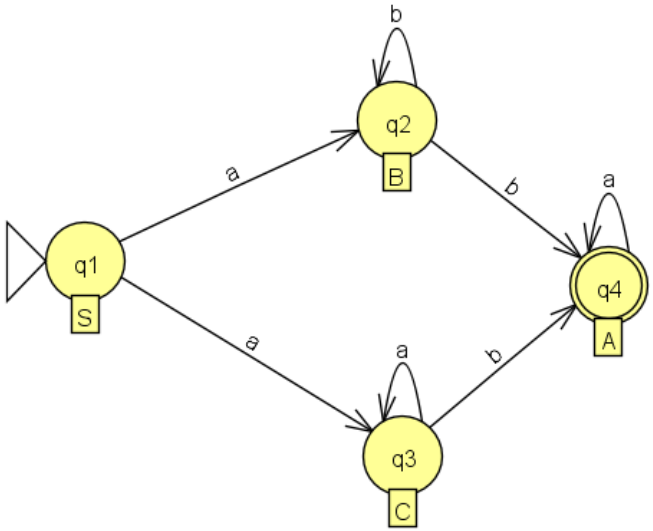
6- Después de las “aaa” y la tercera ‘b’, el autómata se detiene y tiene uno de los estados (q6) como aceptación
Dado que el autómata es no-determinista tiene múltiples estados

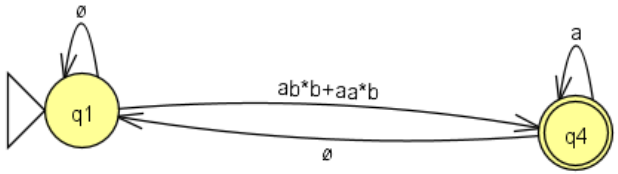


Autómatas finitos no deterministas

- La definición de autómata finito no determinista AFND coincide con la de autómata finito : $AFND = (E, Q, f, q1, F)$
- Donde $f: E^* xQ \rightarrow Q$ es No determinista
- Ejemplo
  - Sea el autómata finito no determinista $AFND = (E, Q, f, q1, F)$ donde: E={a, b}, Q={q1,q2,q3, q4}, F={q4} y la función F esta dada por la tabla:







A	→	aA
C	→	aC
B	→	bB
S	→	aB
A	→	λ
B	→	bA
C	→	bA
S	→	aC

f	*a*	*b*
q_1	$\{q_2, q_3\}$	λ
q_2	λ	$\{q_2, q_4\}$
q_3	q_3	q_4
q_4	q_4	λ

El lenguaje:

- $a(b^*b \mid a^*b)a^*$

o también

- $a(b^* \mid a^*)ba^*$

Autómatas finitos deterministas

- Un autómata finito determinista AFD es un caso particular de los autómatas finitos, en el que la función de transición no presenta ningún ambigüedad en las transiciones de estados para una entrada dada.
- Un autómata finito determinista es una
  - quíntupla $AFD = (E, Q, f, q_1, F)$
  - E = {conjunto de entradas o vocabulario de entrada}
  - Q = {conjunto de estados} es el conjunto de estados posibles, puede ser finito o infinito.
  - función $f: E^* xQ \rightarrow Q$ es determinista.

    $E^* xQ$ devuelve un estado perteneciente al conjunto Q.
  - q_i Es el estado inicia; El estado inicial es único .
  - F conjunto de estados finales, $F \subset Q$

Teorema sobre la transformación de AFND en AFD

- Para todo autómata finito no determinista

$$AFND = (E, Q, f, q1, F)$$

- se puede construir un autómata finito determinista

$$AFD = (E, Q', f', q'1, F')$$

  - tal que el lenguaje reconocido por el autómata finito determinista AFD coincida con el lenguaje reconocido por el autómata finito no determinista AFND, es decir

$$L(AFD) = L(AFND).$$

Algoritmo de transformación de una gramática de tipo 3 en un autómata finito

- Sea una gramática de tipo 3 o regular G=(VT,VN,S,P) y se desea obtener un autómata finito AF=(E,Q,f,q1,F) que reconozca el lenguaje generado por la gramática. El autómata obtenido, en general será no determinista.
- **Solución:** Se determinan los distintos elementos del autómata de la siguiente forma:

E=VT

$Q = VN \cup \{qf\}$

A cada símbolo no terminal de la gramática se le asocia un estado del autómata. Además se introduce un nuevo estado, denominado qf, que será el único estado final del autómata

q1=S

F={qf}

La función de transición f se determina a partir de la forma de las reglas de producción P, de la manera siguiente:

a) Para reglas de la forma A a B se obtiene f(A,a)=B siendo A y B los estados correspondientes a los no terminales A y B.

b) Para reglas de la forma A a se obtiene f(A,a)=qf.

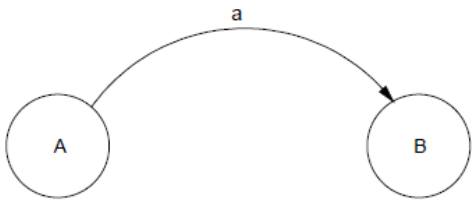


Diagrama de Moore de f(A,a)=B

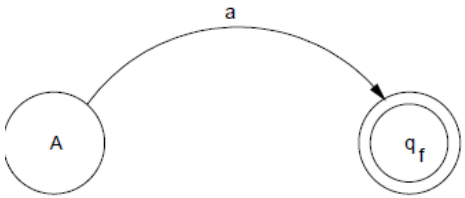


Diagrama de Moore de $f(A,a)=q_f$

Gramática G = (VT , VN , S , P)
VT = {conjunto finito de símbolos terminales}
VN = {símbolos no terminales}
S es el *símbolo inicial* y pertenece a VN.
P = {conjunto de producciones o de reglas de derivación}

Automata A = (E, S, Q, f, g) donde :
E = {conjunto de entradas o vocabulario de entrada}
S = {conjunto de salidas o vocabulario de salida}
Q = {conjunto de estados}

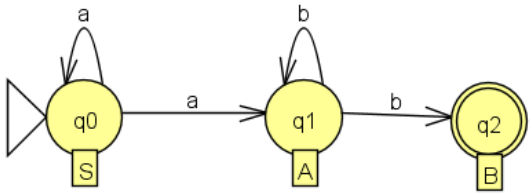
Ejemplo

- Sea la gramática de tipo 3 siguiente :

$$G = (VN = \{A, S, B\}, VT = \{a, b\}, S, P)$$

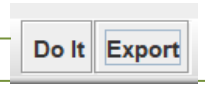
- donde P son las reglas :

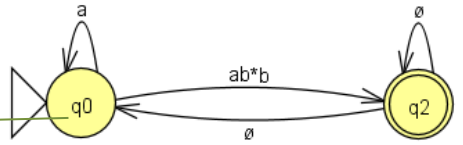
S	$\longrightarrow$	aS
A	$\longrightarrow$	bA
B	$\longrightarrow$	λ
S	$\longrightarrow$	aA
A	$\longrightarrow$	bB



f	*a*	*b*
S	{S,A}	-
A	-	{A,B}
B	-	-

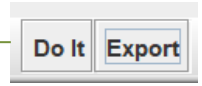
- Obtener un AFND y otro AFD que reconozcan el mismo lenguaje que esta gramática genera.
- Solución : Se define el AFND a partir de la gramática como donde
- E={a,b}, Q={A,S,B}, q1=S, F={B}, y f viene dada por la tabla siguiente :
- Reconoce el lenguaje que forma el conjunto: {a*ab*b)

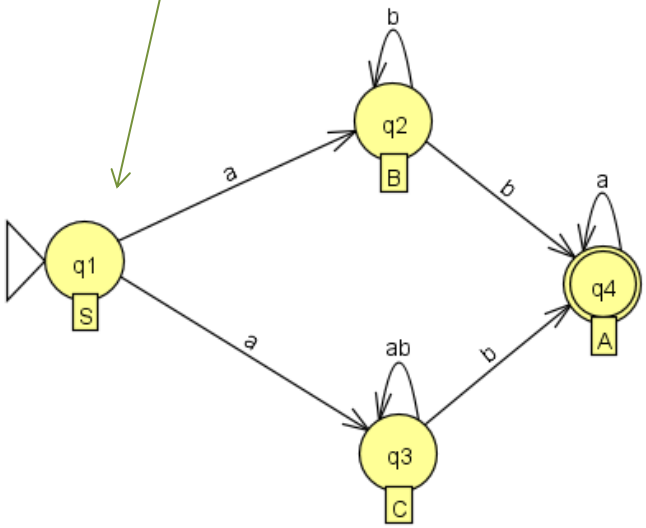




En autómata de la figura

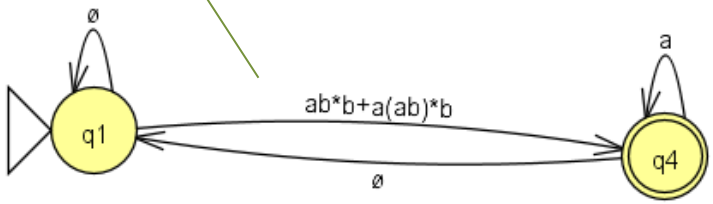
- Reconoce el lenguaje que forma el conjunto : {(ab*b+a(ab)*b)a*}





Gramática

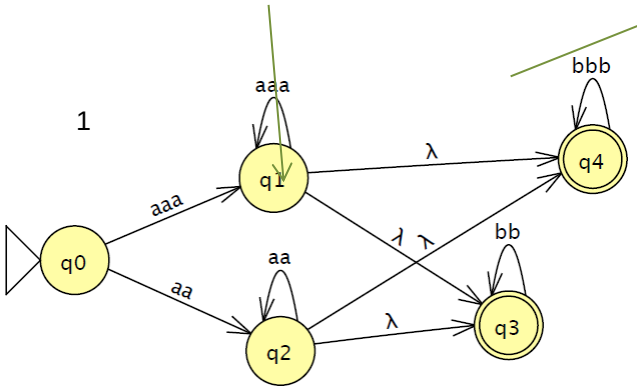
C	$\longrightarrow$	abC
A	$\longrightarrow$	aA
A	$\longrightarrow$	λ
B	$\longrightarrow$	bA
B	$\longrightarrow$	bB
S	$\longrightarrow$	aC
C	$\longrightarrow$	bA
S	$\longrightarrow$	aB



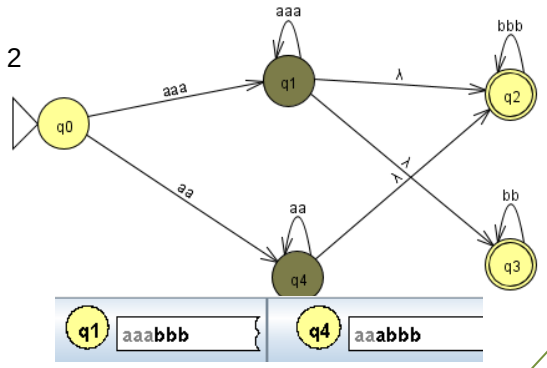
Alternative Multiple Character Transitions

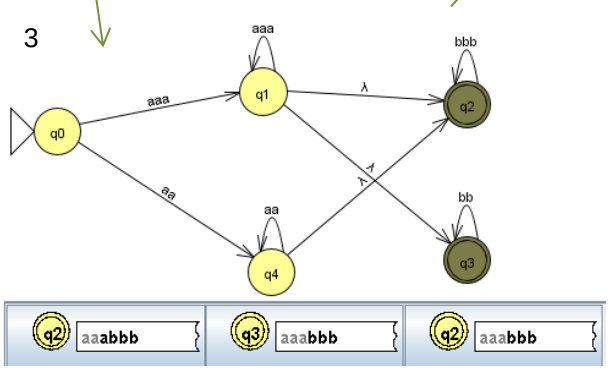
- JFLAP permite múltiples caracteres en una transición
  - n characteres de la trancicion s1s2 . . . sn

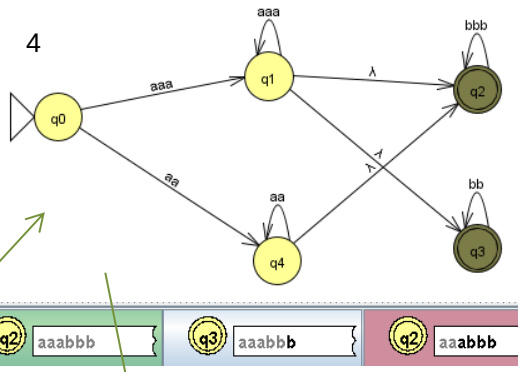
aaa from q0 to q1

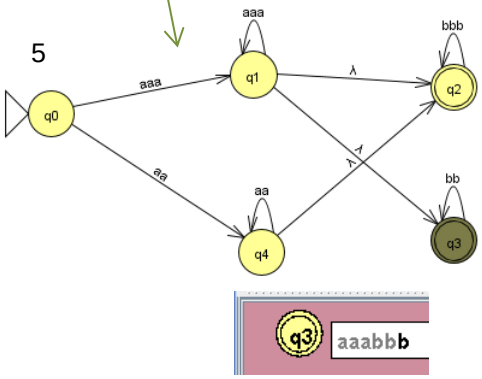


q0 es un estado no determinístico: tiene dos transiciones, una de aaa y otra de aa









Transformación de una expresión regular en un autómata finito

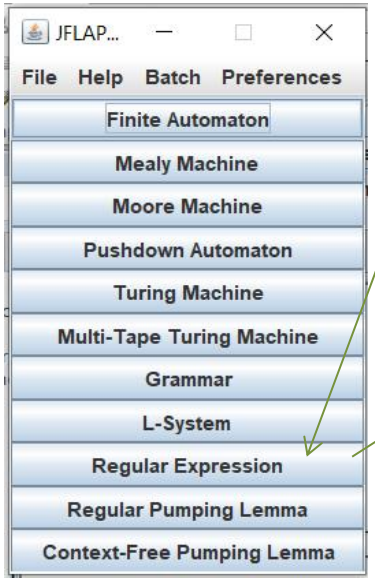
- Dada una expresión regular existe un autómata finito capaz de reconocer el lenguaje que ésta define.
- Recíprocamente, dado un autómata finito, se puede expresar mediante una expresión regular el lenguaje que reconoce.
- Para la transformación de una expresión regular en un autómata finito, se definirán en un principio las equivalencias entre las expresiones regulares básicas y sus autómatas finitos.

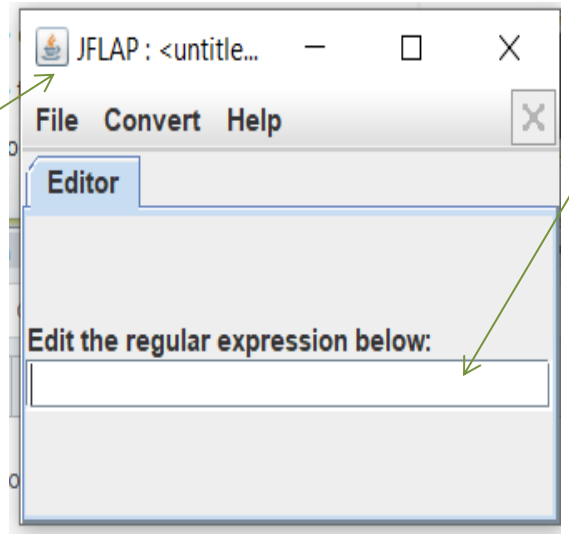
Expresiones regulares

- En este apartado introducimos las RE en JFLAP
  - Expresiones regulares
  - En este capítulo introducimos
  - como representar lenguajes regulares
  - expresiones regulares (REs).
  - cómo editar las REs
  - convertir una RE en una AFND equivalente
  - convertir una AF en una RE equivalente
  - definición formal de una RE de JFLAP.

Edición de expresiones regulares en JFLAP

- Edición de expresiones regulares
  - En esta sección aprendemos a editar las REs. Inicie JFLAP,
  - elija crear una nueva estructura a través de la opción de menú Archivo Nuevo.
  - Seleccione Expresión Regular.
  - Aparecerá una ventana como la de figura.
  - Dado que las RE son cadenas, el editor consiste en un pequeño campo de texto





Edición de expresiones regulares en JFLAP

- Los RE utilizan tres operadores básicos (son operadores en el sentido matemático) estos son:
  - la estrella Kleene (representada por el carácter de asterisco *)
  - el operador de concatenación (implícito al hacer dos expresiones adyacentes)
  - el operador de unión (operador "o", representado por el signo más +).
  - Se pueden utilizar paréntesis para especificar el orden de las operaciones.
  - El signo de exclamación (!) designa la cadena vacía, y es una forma fácil de introducir λ.

Equivalencia entre expresiones regulares básicas y autómatas finitos

- **Expresión regular λ**

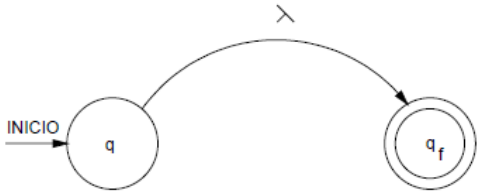


Diagrama de Moore de λ

- **Expresión regular *a***

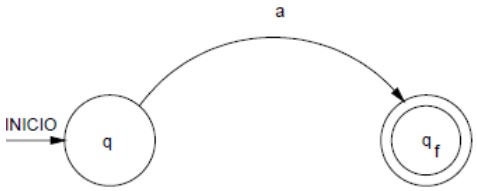


Diagrama de Moore de *a*

- **Expresión regular *a**

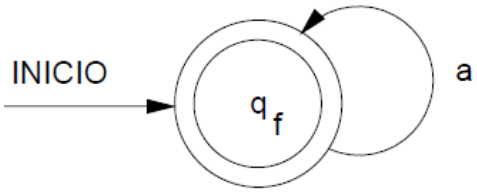


Diagrama de Moore de a^*

- **Expresión regular *a+***

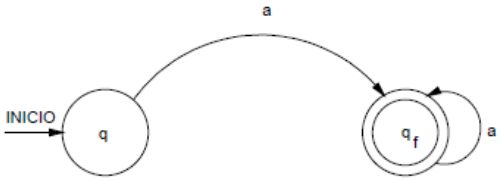
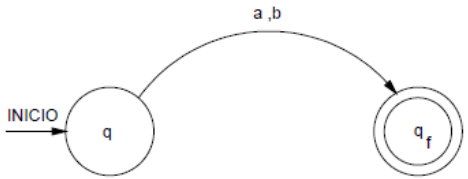


Diagrama de Moore de a^+

Equivalencia entre expresiones regulares básicas y autómatas finitos

- **Expresión regular *a+b***



Otro diagrama de Moore de *a|b*

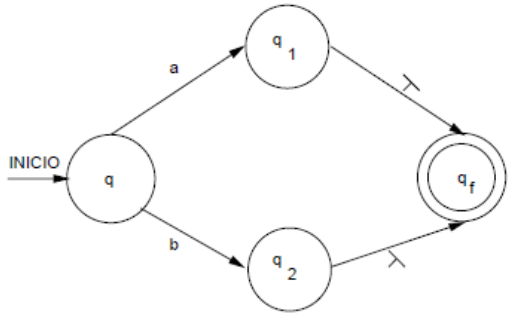


Diagrama de Moore de *a|b*

- **Expresión regular *(a+b)****

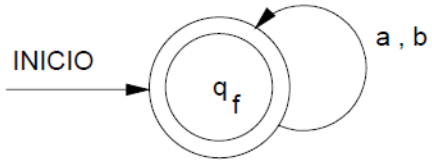


Diagrama de Moore de $(a|b)^*$

- **Expresión regular *(ac+b)****

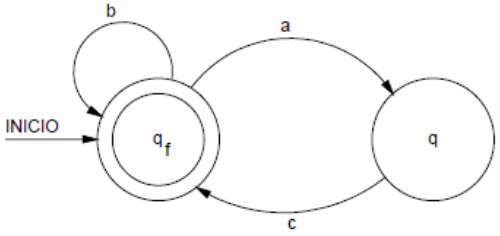


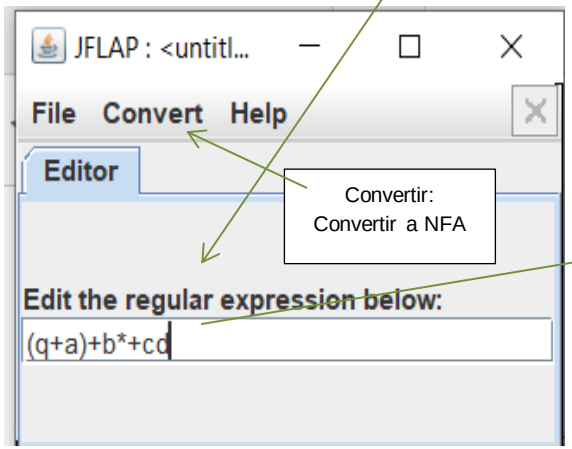
Diagrama de Moore de $(ac|b)^*$

Edición de expresiones regulares en JFLAP

- Ejemplos de REs
  - La expresión $a + b + cd$ describe el lenguaje $\{a, b, cd\}$
  - mientras que $abcd$ describe el lenguaje de una palabra (singleton) $\{abcd\}$.
  - La expresión $a(b + c)d$ describe el lenguaje $\{abd, acd\}$
  - mientras que $ab + cd$ describe el lenguaje $\{ab, cd\}$.
  - La expresión abc^* describe el lenguaje $\{ab, abc, abcc, abccc, \dots\}$
  - mientras que $(abc)^*$ describe el lenguaje $\{\lambda, abc, abcabc, abcabc, \dots\}$
  - La expresión $a + b^*$ describe el lenguaje $\{a, \lambda\ , b, bb, bbb, \dots\}$
  - mientras que $(\mathrm{a} + \mathrm{b})^*$ describe el lenguaje $\{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$.
  - La expresión $(! + a)bc$ describe el lenguaje $\{bc, abc\}$; recuerde que ! es la forma en que el usuario introduce λ.
- Aquí solo usaremos minúsculas, pero JFLAP permite cualquier carácter excepto $*, +, (,),$ ó ! como parte del lenguaje de un RE.
- Tarea
  - Convertir todas las expresiones reculares de esta filmina en autómatas

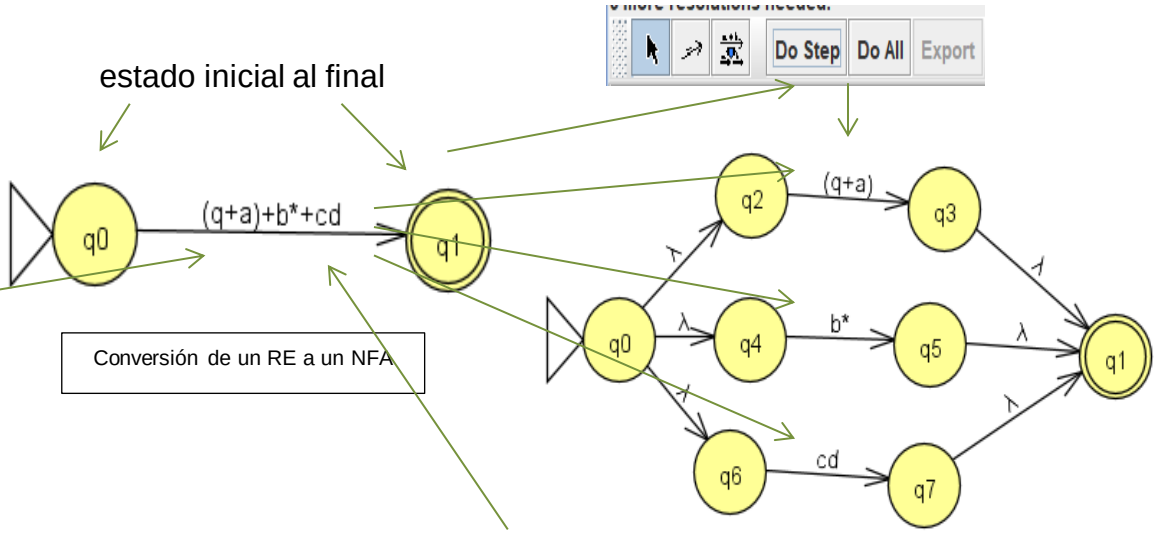
Edición de expresiones regulares en JFLAP

- Vamos a introducir el RE $(q + a) + b^* + cd$, un RE muy simple que indica que queremos una cadena que consista en q o a, o en cualquier número de b, o la cadena cd.
  - Específicamente, la tecla de espacio es un carácter legal para un lenguaje. Por ejemplo, $a\ ^*$ en el que un espacio sigue a la "a" (por lo que a va seguido de cualquier número de espacios) es distinto de a (cualquier número de a).
    - Obsérvese que ninguna de las expresiones regulares presentadas tiene espacios en ellas, por lo que no las escribas.



Convertir:
Convertir a NFA

estado inicial al final



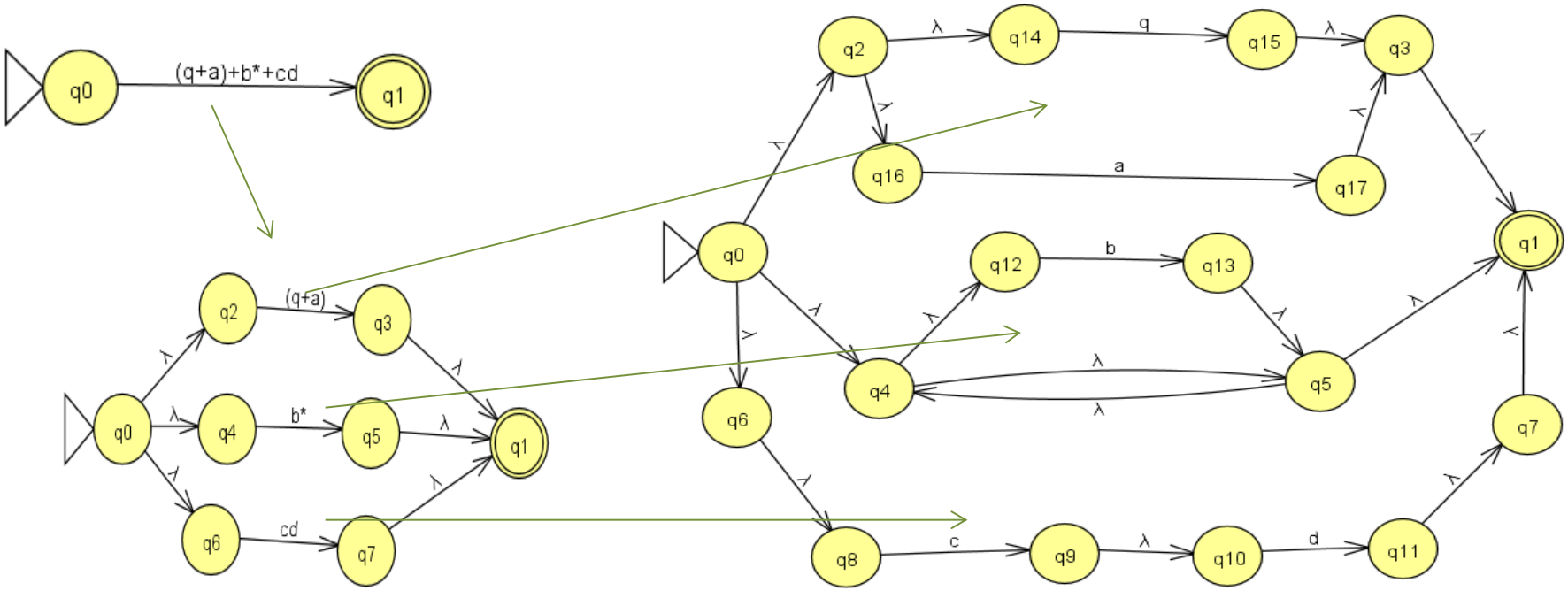
Conversión de un RE a un NFA

En la RE lo operadores se evalúan según el orden de presidencia de los operadores

Edición de expresiones regulares en JFLAP

- Comenzamos con un GTG de dos estados, y una transición de una sola expresión con nuestra expresión regular del estado inicial al final.
- La idea del convertidor es que sustituimos cada transición por nuevos estados conectados por transiciones en los operandos del operador de nivel superior de esa expresión.
  - Intuitivamente, el operador de nivel superior es el operador de una expresión que debe ser evaluado en último lugar. Por ejemplo, en ab+c, el operador de nivel superior es +, ya que el operador de concatenación tiene mayor prioridad y será evaluado antes que el +.
- A continuación, conectamos estos operandos con transiciones-λ para duplicar la funcionalidad del operador perdido.
- De esta manera, en cada paso mantenemos un GTG equivalente al RE original.

Edición de expresiones regulares en JFLAP

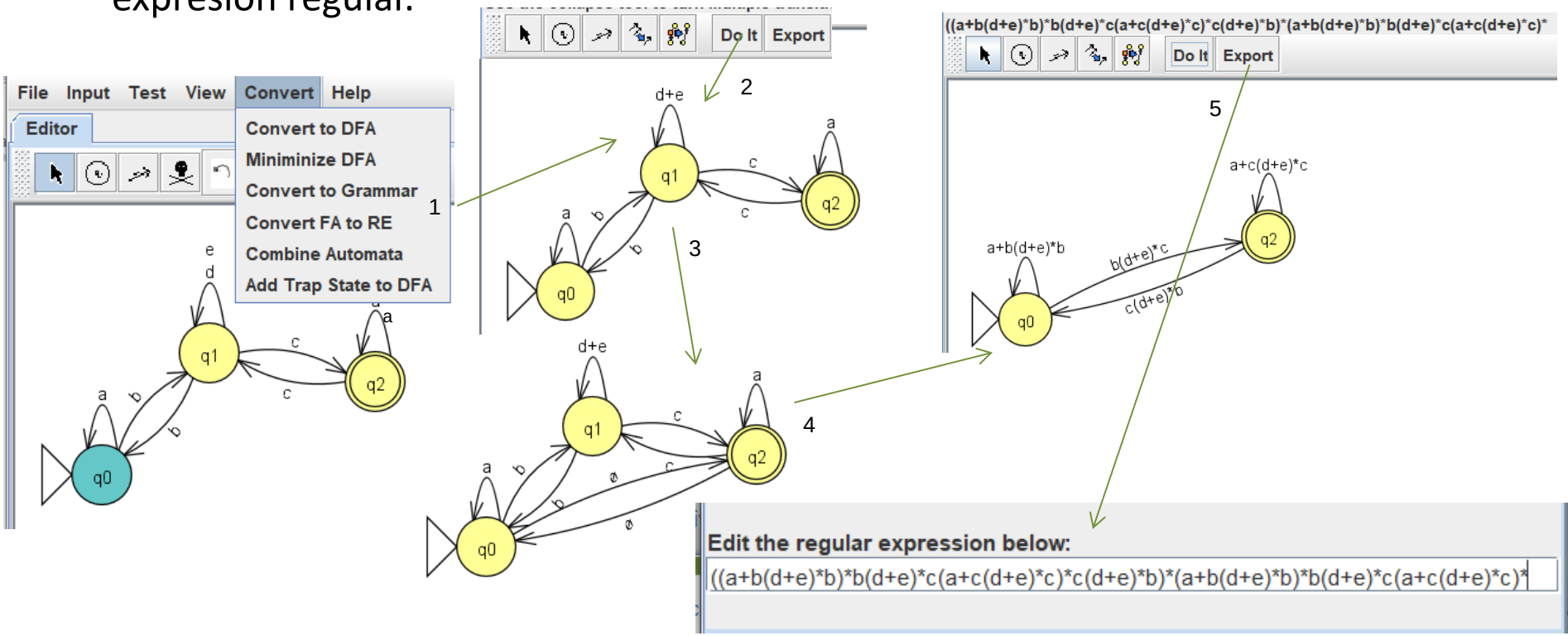


El AFND que reconoce el lenguaje $(q + a) + b^* + cd$.

- Eventualmente todos los operadores son eliminados y nos quedamos con un solo carácter y λ por trancicion, en cuyo momento el GTG puede ser considerado como una NFA.
- El botón "Do it" realiza de a un paso de la conversión transformado la transición de una sola expresión.
- En nuestro caso, comenzamos con la transición (q+a)+b+cd, de modo que se reduce y se añaden las transiciones y estados necesarias sin la intervención del usuario.

Convertir un FA en una expresión regular

- Convertir un NDAF en un RE tiene una lógica similar a la conversión de RE a NDFA
  - Comenzamos con un NDFA que consideramos un GTG a efectos de la conversión.
  - Luego eliminamos los estados sucesivamente, generando GTG equivalentes hasta que sólo quede un único estado inicial y un único estado final.
  - El JFLAP utiliza entonces una fórmula para expresar el GTG simplificado como una expresión regular.



Definition of an RE in JFLAP

- Que sea un alfabeto Σ. El conjunto de todas las expresiones regulares posibles está dado por R.

$$R = \{\phi, \lambda\} \cup \Sigma \cup \{ab | a, b \in R\} \cup \{a + b | a, b \in R\} \cup \{a^* | a \in R\} \cup \{(a) | a \in R\}$$