
Tema 1: Autómatas finitos y lenguajes regulares

Autómatas y Matemáticas Discretas

Escuela de Ingeniería Informática

Universidad de Oviedo

CONTENIDO

- 1.1 Alfabetos, palabras y lenguajes
- 1.2 Expresiones regulares
- 1.3 Autómatas finitos deterministas
- 1.4 Minimización de autómatas finitos deterministas
- 1.5 Autómatas finitos no deterministas
- 1.6 Autómatas finitos no deterministas con λ -movimientos
- 1.7 Equivalencia entre expresiones regulares y autómatas finitos
- 1.8 Propiedades de los lenguajes regulares

1. ALFABETOS, PALABRAS Y LENGUAJES

- Los Lenguajes son fundamentales en computación. Sobre ellos se basa la comunicación ya sea hombre-máquina (lenguajes de programación, lenguajes de consultas, etc.) o máquina-máquina (protocolos de comunicación, etc.).
- Gracias a ellos también se definen gran cantidad de formatos de ficheros (postscript, ficheros de configuración, etc.)

-
- Se trabaja con Lenguajes Formales que poseen reglas sintácticas y semánticas rígidas, concretas y bien definidas que indican cómo construir programas u órdenes o consultas válidas.
 - El procesamiento de los lenguajes formales es importante en la informática ya que se necesita traducir los programas escritos en lenguajes de alto nivel a programas en lenguaje máquina. Este proceso de traducción lo realizan los compiladores e intérpretes.

Definición de los Lenguajes

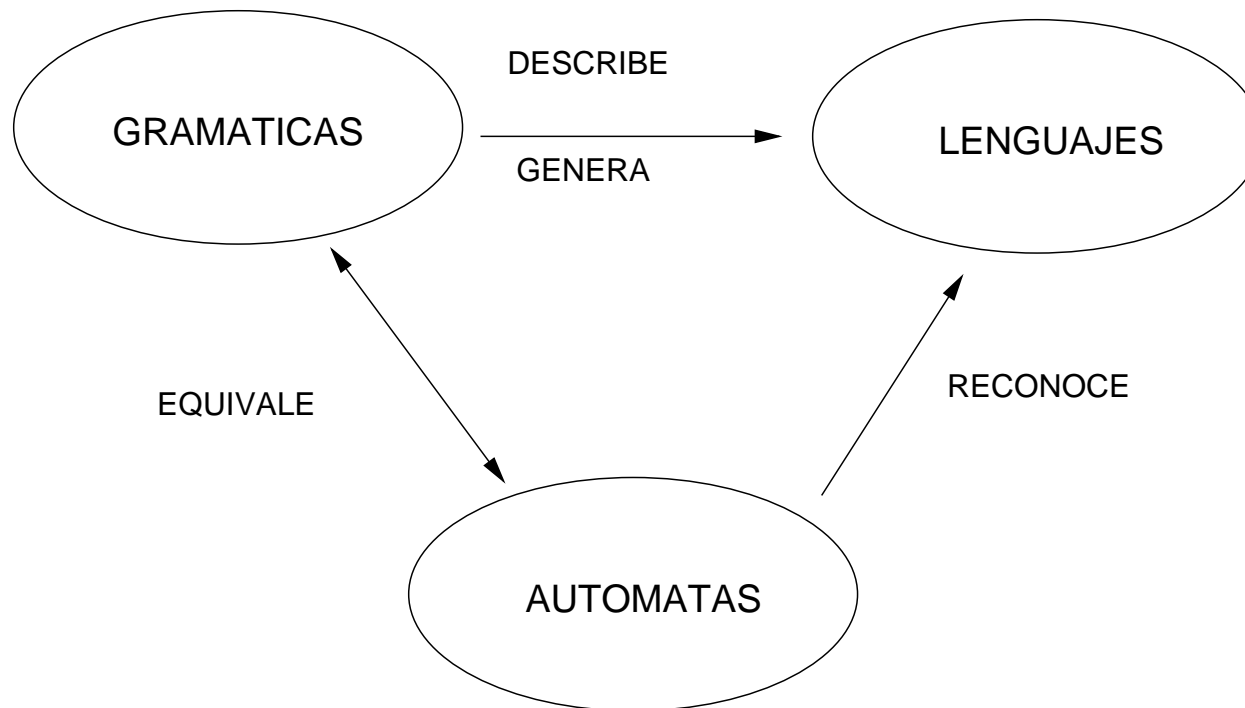
- En general el conjunto de las cadenas válidas de un determinado lenguaje es infinito (el conjunto de programas correctos escritos en C++ es infinito).
- Es necesario poder describir de forma sintética los lenguajes. Especificando mediante reglas, cómo se escriben programas sintácticamente correctos en un determinado lenguaje.

-
- Las Gramáticas: Describen la estructura de un lenguaje, proporcionando las reglas que nos permiten construir secuencias válidas dentro de un lenguaje. (Cómo se escribe un programa correcto en Java)
 - Las Expresiones Regulares: Describen de manera declarativa cómo son las cadenas que pertenecen a un cierto tipo de lenguajes llamados Lenguajes Regulares (LR).
 - Los Lenguajes Regulares (LR) son unos lenguajes más sencillos que los habituales lenguajes de programación, pero que son muy utilizados en informática y serán objeto de estudio en el presente tema.

Reconocimiento de lenguajes

- Los programas fuente (un programa en C++) que son generados por un emisor (el programador), deben ser reconocidos por un receptor (un compilador).
- Formalmente definimos un Autómata como una máquina abstracta capaz de recibir, procesar y transmitir información.
- En la práctica el proceso suele consistir en comprobar si una secuencia de órdenes es correcta y, si lo es, ejecutar dichas órdenes o traducirlas a otro lenguaje más sencillo. Esto es, más o menos, lo que hacen los compiladores y los intérpretes.

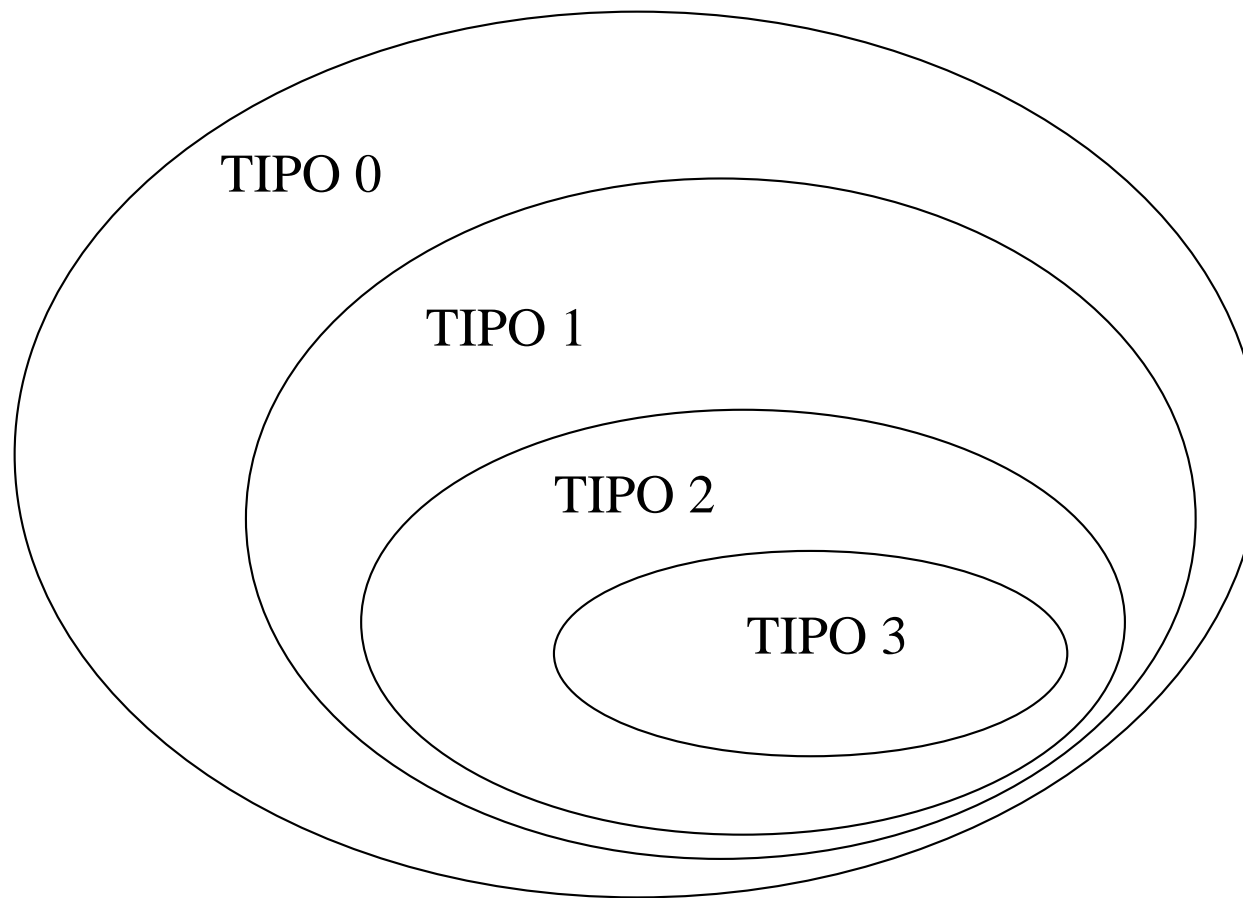
Existe una correspondencia entre Gramáticas, Lenguajes y Autómatas.



Clasificación de Chomsky de las gramáticas.

	GRAMÁTICA	LENGUAJE	AUTÓMATA
Tipo 0	Sin restricciones	Recursivamente Enumerable	Máquina de Turing
Tipo 1	Sensible al contexto	Dependientes del contexto	A. linealmente acotados
Tipo 2	Libre de contexto	Libre de contexto	A. de Pila
Tipo 3	Regular	Regular	A. Finitos

Existe una relación de inclusión entre los diferentes tipos de gramáticas, lenguajes y autómatas.



-
- **Alfabeto:** Conjunto finito y no vacío de símbolos.
 - Ejemplos: $X_1 = \{0, 1\}$, $X_2 = \{a, b, c\}$, $X_3 = \{00, 01, 10, 11\}$
 - Será válido si no se genera ambigüedad en la formación de las palabras, por ejemplo:
 $X = \{00, 11, 100, 111\}$ y la palabra 11100
 - **Palabra o cadena:** Secuencia finita de símbolos de un alfabeto. Ejemplo. Sea $X = \{a, o, l, h\}$, son palabras $p_1 = hola, p_2 = ola$
 - **Longitud de una palabra:** número de símbolos que la forman. Se denota mediante $| \cdot |$. Ejemplo: Si $X = \{0, 1, 2, \dots, 9\}$ entonces $|41| = 2$ y $|12356| = 5$.

-
- **Palabra vacía** (λ): Dado un alfabeto X , se define λ como la única palabra construida con 0 símbolos del alfabeto. λ es una palabra no un símbolo del alfabeto ($\lambda \notin X$).
 - **Universo del discurso**, X^* . Se compone de todas las palabras que se pueden formar con símbolos del alfabeto X . Contiene un número infinito de palabras. La palabra vacía pertenece a todos los universos. Ejemplo: Sea $X = \{a, b\}$, $X^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

-
- **Lenguaje:** Dado un alfabeto X , un lenguaje sobre X será un subconjunto de X^* .
 - Conjunto de palabras, también llamadas sentencias o cadenas, formadas por símbolos de un alfabeto.
 - Ejemplo: dos posibles lenguajes sobre $X = \{a, b\}$ serían:
 $L_1 = \{aaa, aba, bbb, bab\}$ y $L_2 = \{\lambda, a, aa, aaa, aaaa, \dots\}$
 - Ejemplo. Sea el alfabeto de los símbolos ASCII. Un lenguaje sobre este alfabeto serían todas aquellas cadenas que representen identificadores válidos en Java. Empiezan con letra, \$, _ y sigue con cero o más letras, dígitos, \$ ó _ , $L_1 = \{a, aux, cont, i1, _sum, \dots\}$

Operaciones con cadenas

- **Concatenación de cadenas.** Dado un alfabeto X , sean $a = a_1 a_2 \dots a_n$ y $b = b_1 b_2 \dots b_m$ cadenas donde $\forall i a_i \in X$ y $\forall j b_j \in X$. La concatenación de las cadenas a y b , $a \cdot b$, es otra cadena formada por los símbolos de a seguidos de los símbolos de b , es decir, $a \cdot b = a_1 a_2 \dots a_n b_1 b_2 \dots b_m$.
 - Además se cumple que $|a \cdot b| = |a| + |b|$
 - λ es el elemento neutro para la concatenación. Para cualquier cadena x , $x \cdot \lambda = \lambda \cdot x = x$

-
- **Potencia de una cadena:** la potencia i -ésima de una cadena x , denotada por x^i , se forma por la concatenación i veces de x . Por definición, para toda cadena x , se cumple que

$$x^0 = \lambda$$

$$\text{Ejemplos: } a^3 = aaa \quad (ac)^2 = acac$$

- Con la operación de concatenación el conjunto de cadenas X^* forma un **monoide** (semigrupo con identidad).

Operaciones con alfabetos

- **Potencia k de un alfabeto:** Dado un alfabeto X y un número no negativo $k \in \mathbb{N}$, definimos $X^k = \{x \mid x \text{ es una palabra sobre } X \text{ y } |x| = k\}$

Ejemplo:

$$X = \{0, 1\} \quad X^0 = \{\lambda\}$$

$$X = X^1 = \{0, 1\}$$

$$X^2 = \{00, 01, 10, 11\}$$

-
- **Cierre Positivo de un alfabeto:** Dado un alfabeto X , definimos su cierre positivo como:

$$X^+ = X^1 \cup X^2 \cup X^3 \cup \dots$$

- **Cierre estrella de un alfabeto:** Dado un alfabeto X , definimos su cierre estrella o cierre de Kleene como:

$$X^* = X^0 \cup X^1 \cup X^2 \cup \dots$$




- **Nota:** Dado un alfabeto X , entonces $\lambda \notin X^+$ y $\lambda \in X^*$

Operaciones con lenguajes

Ejemplo: Consideremos $L_1 = \{a, b, c\}$, $L_2 = \{c, d\}$ y $L_3 = \phi$

- **Unión o alternativa:**

$$L_1 \cup L_2 = \{w | w \in L_1 \vee w \in L_2\}$$

Ejemplo: $L_1 \cup L_2 = \{a, b, c, d\}$ 

- **Concatenación:**

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 | w_1 \in L_1 \wedge w_2 \in L_2\}$$

Ejemplos:

$$L_1 \cdot L_2 = \{ac, ad, bc, bd, cc, cd\}$$

$$L_1 \cdot L_3 = \phi$$



- **Intersección:**

$$L_1 \cap L_2 = \{w | w \in L_1 \wedge w \in L_2\}$$

Ejemplo: $L_1 \cap L_2 = \{c\}$

- **Potencia k de un lenguaje:** Dado un lenguaje L y un número no negativo $k \in N$, definimos

$$L^k = \{w_1 w_2 \cdots w_k | w_i \in L, i = 1, 2, \cdots k\}.$$



Ejemplo:

$$L = \{\lambda, a, b, ab\}$$

$$L^2 = \{\lambda, a, b, aa, ab, aab, ba, bb, bab, aba, abb, abab\}$$

-
- **Cierre Positivo de un lenguaje:** Dado un Lenguaje L , definimos su cierre positivo como:

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \dots$$



- **Cierre estrella de un lenguaje:** Dado un lenguaje L , definimos su cierre estrella o cierre de Kleene como:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Nota: Dado un lenguaje L , entonces si $\lambda \in L$ entonces $\lambda \in L^+$

2. EXPRESIONES REGULARES

Definiciones Básicas

- Las **expresiones regulares (er)** definen las cadenas válidas de un lenguaje mediante una descripción algebraica (fórmula).
- Los lenguajes que pueden describirse mediante expresiones regulares se denominan **lenguajes regulares**.
- Las er se utilizan como lenguaje de entrada en muchos sistemas de proceso de cadenas como el comando *grep* de UNIX u otros similares en navegadores, procesadores, etc.



Sea X un alfabeto finito. Las expresiones regulares sobre X y los lenguajes que denotan se definen recursivamente como:

- \emptyset es una er y denota al conjunto vacío.
- λ es una er y denota al lenguaje $\{\lambda\} \subset X^*$
- Si $x \in X$ entonces x es una er y denota al lenguaje $\{x\} \subset X^*$

Ejemplo: $X = \{a, b, c, d\}$ la er a representa al lenguaje $L = \{a\}$



- Si r y s son er que denotan a los lenguajes R y $S \subseteq X^*$
 - $r + s$ o $r|s$ denota al lenguaje $R \cup S$
Si $r = a$, $s = b$, entonces $r + s = a + b$ y $L(r + s) = \{a, b\}$
 - $r \cdot s$ denota al lenguaje $R \cdot S$ (el \cdot suele omitirse)
Si $r = a$, $s = b$, entonces $r \cdot s = a \cdot b$ y $L(r \cdot s) = \{ab\}$
 - r^* denota al lenguaje R^*
Si $r = a$, entonces $r^* = a^*$ y $L(a^*) = \{\lambda, a, aa, aaa, \dots\}$
 - r^+ denota al lenguaje R^+
Si $r = a$, entonces $r^+ = a^*$ y $L(a^+) = \{a, aa, aaa, \dots\}$

Precedencia de los operadores.

- 1^o Operación cierre y cierre positivo. $(*, +)$ + prioridad
- 2^o Operación concatenación (\cdot)
- 3^o Unión o alternativa $(+, |)$ - prioridad
- Se permite el uso de paréntesis para indicar la precedencia

NOTA: si asociamos los cierres con potencias, la concatenación con la multiplicación y la unión con la suma, la precedencia es análoga a la de expresiones aritméticas. (OJO, que la concatenación no es conmutativa).

Ejemplos sobre el alfabeto $X = \{a, b\}$

Expresión regular	Lenguaje
$a b a \cdot b$	$\{a, b, ab\}$ Ya no ponemos el \cdot
$a(a b)$	$\{aa, ab\}$
$(a b)^*$	$\{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$
$(bb)^*$	$\{\lambda, bb, bbbb, bbbbbb, \dots\}$
$(aa b)(aa \lambda)$	$\{aaaa, aa, baa, b\}$
ab^*a	$\{aa, aba, abba, \dots\}$

Ejemplos sobre el alfabeto $X = \{a, b\}$



- Palabras que empiezan por a
- Palabras que terminan en a
- Palabras que empiezan y terminan por la misma letra
- Palabras que empiezan y terminan por distinta letra
- Palabras que contienen la subcadena aa
- Palabras que no contienen la subcadena aa
- Palabras con un número par de a 's

Ejemplos sobre el alfabeto $X = \{0, 1, \dots, 9, +, -, e, E, .\}$

- Dígitos
- Números naturales (123)
- Números enteros (-134, +7, 0105)
- Números decimales (la parte entera puede omitirse (.007), y la decimal también (69.) pero no a la vez
- Números decimales con exponente (31.4e-1)
- Números decimales con o sin exponente

Equivalencia de expresiones regulares

- Dos er son equivalentes si denotan el mismo lenguaje.
- Sean x, y y z expresiones regulares
 - $+$ es asociativa: $x + (y + z) = (x + y) + z$
 - $+$ es conmutativa: $x + y = y + x$
 - \cdot es asociativa: $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
 - \cdot es distributiva respecto a $+$
 $x \cdot (y + z) = x \cdot y + x \cdot z$
 $(x + y) \cdot z = x \cdot z + y \cdot z$
 - \cdot tiene como elemento neutro a λ :
 $x \cdot \lambda = \lambda \cdot x = x$
 - $+$ tiene como elemento neutro a \emptyset :
 $x + \emptyset = \emptyset + x = x$

-
- $\lambda^* = \lambda$
 - $\emptyset \cdot x = x \cdot \emptyset = \emptyset$
 - $\emptyset^* = \lambda$
 - $x + x = x$
 - $x^* x^* = x^*$
 - $xx^* = x^*x = x^+$
 - $(x^*)^* = x^*$
 - $(xy)^*x = x(yx)^*$
 - $x^* = \lambda + x^+ = (\lambda + x)^*$
 - $(x + y)^* = (x^* + y^*)^* = (x^*y^*)^*$
 - $(x + y)^* = (x^*y)^*x^* = x^*(yx^*)^*$
 - $(x + y)^* \neq x^* + y^*$

La regla de Arden

- Si r, s y t son expresiones regulares y $\lambda \notin s$ se cumple que

$$r = sr + t$$

si y solo si

$$r = s^*t$$

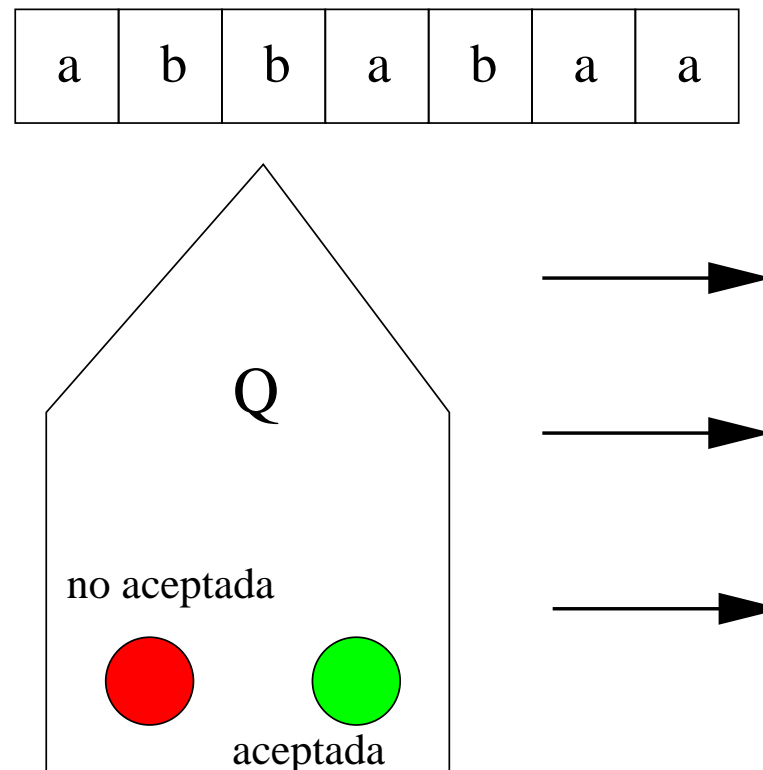
- Se utiliza para resolver sistemas de ecuaciones cuyas variables son expresiones regulares
- Recuerda a la suma de progresiones geométricas de razón $|s| < 1$

$$\frac{1}{1-s} = 1 + s + s^2 + s^3 + \dots$$

3. AUTÓMATAS FINITOS DETERMINISTAS

- Informalmente podemos ver un AF como una máquina que recibe a su entrada una cadena de símbolos y nos da como salida un *SI* o un *NO*, dependiendo de si dicha cadena pertenece o no a un determinado lenguaje.
- Según van *llegando* los símbolos el AF va cambiando de estado. Si al finalizar la cadena el AF se encuentra en uno de los estados de aceptación de palabra, entonces se dice que dicha cadena pertenece al lenguaje que reconoce dicho AF.

- Cada AF A reconoce un determinado lenguaje regular $L(A)$. En consecuencia, cada palabra de $L(A)$ debe hacer transitar al AF desde el estado inicial a un estado final; y cada palabra que no pertenece a $L(A)$ debe hacerle transitar desde el estado inicial a un estado no final.



-
- Podemos ver los estados un un AF como un mecanismo de memoria: cada estado nos dice qué tipo de cadena ha llegado hasta ese momento.
 - El conjunto de estados es finito. Según se van leyendo los símbolos de entrada el Autómata irá cambiando de estado. Existe un estado diferenciado que es el inicial y es en el que se encuentra el Autómata cuando empieza a funcionar.
 - Los estados se clasifican generalmente como finales o no finales. Aunque puede resultar más claro denominarlos **estados de aceptación** o **estados de no aceptación**. De hecho un AF puede encontrarse varias veces en un estado de aceptación antes de finalizar la lectura de la cadena.

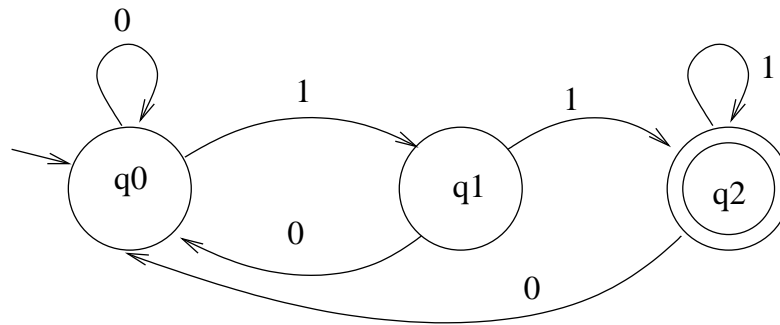
Autómatas Finitos Deterministas

- Formalmente definimos un Autómata Finito Determinista (**AFD**) como una quintupla de la forma $AFD = (X, Q, \delta, q_0, F)$, donde:
 - X es un conjunto finito denominado alfabeto de entrada;
 - Q es un conjunto finito llamado conjunto de estados;
 - $\delta : Q \times X \rightarrow Q$ es la función de transición de estados;
 - q_0 es el llamado estado inicial;
 - $F \subseteq Q$ recibe el nombre de conjunto de estados de aceptación o finales.

-
- Determinista hace referencia al hecho de que δ debe estar definida para cada estado y símbolo del alfabeto y además para cada estado y para cada posible símbolo de entrada, existe un único estado al que el AFD puede llegar.

Representación de AFDs

- **Diagrama de estados:** Se trata de un grafo dirigido donde los nodos representan los estados y los arcos las transiciones entre ellos. El estado inicial se identifica con una flecha y los estados de aceptación con un doble círculo.



Ejemplo 1

Representación de AFDs

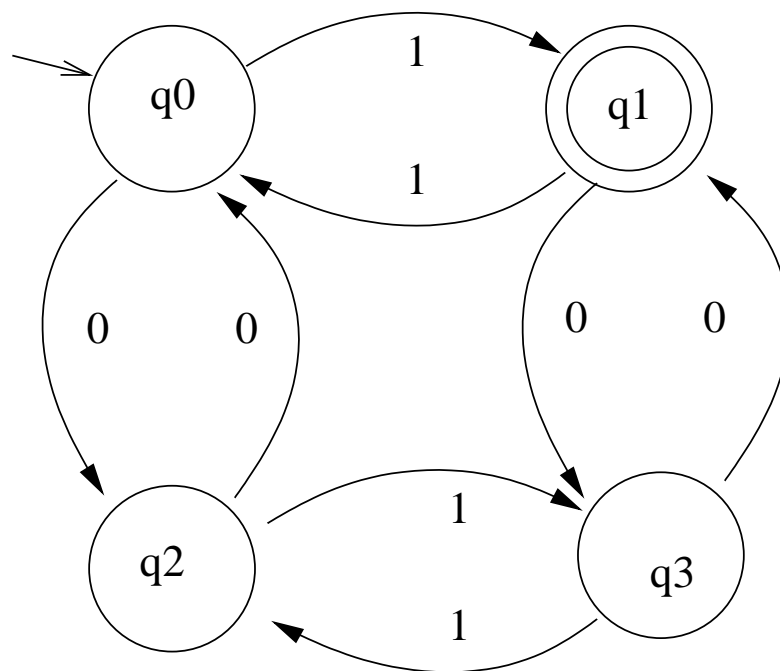
- **Tabla de transiciones:** Representaremos la función δ de forma tabular y marcamos el estado inicial, que suele ser el primero, con una flecha y los estados de aceptación con un asterisco (*).

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_0	q_2
q_2^*	q_0	q_2

Ejemplo: Diseñar un AFD que reconozca el lenguaje formado por aquellas cadenas de ceros y unos que tienen un número par de ceros y número impar de unos.

- $X = \{0, 1\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
 - q_0 : ha llegado un nº **par** de ceros y un nº **par** de unos
 - q_1 : ha llegado un nº **par** de ceros y un nº **impar** de unos
 - q_2 : ha llegado un nº **impar** de ceros y un nº **par** de unos
 - q_3 : ha llegado un nº **impar** de ceros y un nº **impar** de unos
- q_0 es el estado inicial.
- $F = \{q_1\}$

- La función de transición δ queda definida en el diagrama.



Ejemplo 3

Ejemplos: Dado el alfabeto $X = \{0, 1\}$, construir un AFD

- AFD que reconozca las palabras en el que, el símbolo 1 aparece un n^o par de veces.
- AFD que reconozca las palabras en el que, el símbolo 1 aparece un n^o impar de veces.
- AFD que reconozca las palabras del alfabeto que empiecen por dos ceros.
- AFD que reconozca las palabras que empiecen y terminen con el mismo símbolo. Ojo con casos como '0' que empiezan y terminan con el mismo símbolo.
- AFD que reconozca las palabras del alfabeto que terminen con dos ceros.
- AFD que reconozca las palabras del alfabeto que no terminen con dos ceros.

Descripción Instantánea de un AFD (configuración):

Es una tupla de la forma (q, w) , donde $q \in Q$ y $w \in X^*$. Siendo q el estado en el que se encuentra el AFD y w la cadena que falta por procesar.

Configuración inicial: (q_0, w)

Configuración de aceptación: (q, λ) y $q \in F$

Movimiento:

Es el paso de una configuración a otra. Se representa por el símbolo \vdash .

$(q, aw) \vdash (q', w)$ entonces $\delta(q, a) = q'$

$(q, w) \vdash^* (q', w')$, cuando se producen cero o más movimientos.

Ver la secuencia de movimientos que provoca la cadena 0010111 en el AFD del Ejemplo 1.

Extensión de la función δ a palabras: δ^*

- δ^* : dado un estado y una cadena, nos devuelve el estado al que llega el AFD.
- Se define como $\delta^*: Q \times X^* \rightarrow Q$
- Se calcula apartir de la función δ :
$$\delta^*(q, \lambda) = q$$
$$\delta^*(q, wa) = \delta(\delta^*(q, w), a), \text{ donde } a \in X, w \in X^*, q \in Q$$

Lenguaje Aceptado por un AFD:

- Una cadena w es aceptada por una AFD sii $\delta^*(q_0, w) \in F$
- Dado un AFD $A = (X, Q, \delta, q_0, F)$, el **lenguaje aceptado** por A es $L(A) = \{w \in X^* \mid \delta^*(q_0, w) \in F\}$.
- El conjunto de los lenguajes aceptados por los AFD coincide con los **lenguajes regulares** (lenguajes denotados por expresiones regulares)
 - Para cada lenguaje denotado por una expresión regular, existe un AFD que reconoce todas cadenas de ese lenguaje y ninguna que no pertenezca a él.
 - Para cada AFD existe una expresión regular que denota exactamente al lenguaje reconocido por el AFD.

-
- Describir mediante lenguaje natural y mediante una e.r. el lenguaje reconocido por los siguientes AFDs.

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1^*	q_1	q_0

δ	0	1
$\rightarrow q_0^*$	q_1	q_0
q_1^*	q_2	q_0
q_2	q_2	q_2

4. MINIMIZACIÓN DE AFDs

Construir un AFD mínimo

- **Ejercicio:** Construir un AFD que reconozca el lenguaje de las palabras sobre $\{0, 1\}$ cuyo penúltimo símbolo es un 0 y que tenga el menor número posible de estados.
- ¿Cómo podemos saber si es el AFD más pequeño posible?
- Dado un AFD, ¿podemos hallar el AFD más pequeño que reconoce el mismo lenguaje que el de partida?

Minimización de AFD

- **Autómatas equivalentes:** son aquellos que reconocen el mismo lenguaje.
- **Estados equivalentes:** Dado un AFD, dos estados p y q son equivalentes ($p \equiv q$) si para toda cadena de entrada w , $\delta^*(p, w)$, es un estado final si y sólo si $\delta^*(q, w)$, también lo es. Es decir, serán equivalentes si tienen el mismo comportamiento ante toda palabra.
- **Estados distinguibles:** p se distingue de q si $\exists w \in X^* | \delta^*(p, w) \in F$ y $\delta^*(q, w) \notin F$ o viceversa.

-
- Si dos estados son equivalentes, pueden ser sustituidos por uno sólo con el mismo comportamiento que los originales.
 - Existen algoritmos para minimizar un AFD, es decir, hallar otro AFD equivalente con el número mínimo de estados.

Algoritmo de Minimización de AFD

- Partimos de un AFD $A = (X, Q, \delta, q_0, F)$, y hallaremos un AFD $A' = (X, Q', \delta', q'_0, F')$ que acepta $L(A)$ y que tiene el menor número de estados.
- El algoritmo consiste en determinar las clases de equivalencia en el conjunto de estados y sustituir cada conjunto de estados de cada clase de equivalencia por un único estado.

PASOS

- **Paso 1:** Se elimina cualquier estado que no sea accesible desde el inicial.
- **Paso 2:** Se dividen los estados restantes en particiones de forma que todos los estados de una partición sean equivalentes y no haya pares de estados equivalentes en particiones distintas.
- **Paso 2.1:** Construir una partición inicial Π dividiendo el conjunto Q en dos grupos: estados finales F , y no finales $Q - F$.

-
- **Paso 2.2:** Determinar una nueva partición Π_{nueva} a partir de Π , aplicando el siguiente procedimiento a cada grupo C_x de Π .
 - Dividir C_x en subgrupos tales que dos estados q_i y q_j de C_x estarán en el mismo subgrupo sii para cada símbolo de entrada e , los estados q_i y q_j tienen transiciones con e hacia estados del mismo grupo de Π .
 - Sustituir C_x por sus grupos en Π_{nueva} .
 - **Paso 2.3:** Si $\Pi_{nueva} \neq \Pi$, (hay nuevos subgrupos)
 - entonces: $\Pi = \Pi_{nueva}$ y volver al Paso 2.2
 - en caso contrario $\Pi_{final} = \Pi$, e ir al Paso 3.

-
- **Paso 3:** Se escoge un estado de cada grupo de la partición Π_{final} como representante para formar el nuevo conjunto Q' .
 - **Paso 4:** Cálculo de δ' en A' . Sea q_i un estado representante, sea un símbolo a tal que $\delta(q_i, a) = q_j$ y sea q_k el representante del grupo de q_j , entonces $\delta'(q_i, a) = q_k$.
 - **Paso 5:** Estado inicial y finales. El estado inicial q'_0 de A' se elige como el representante del grupo que contiene al estado inicial q_0 de A .
El conjunto de estados finales F' estará formado por los representantes de grupos donde haya estados finales.

Ejemplo de minimización de AFD

Minimizar el AFD $A_1 = (X, Q, \delta, q_0, F)$, donde:

$X = \{a, b\}$, $Q = \{1, 2, 3, 4, 5\}$, $q_0 = 1$, $F = \{5\}$

y δ viene dada por la siguiente tabla:

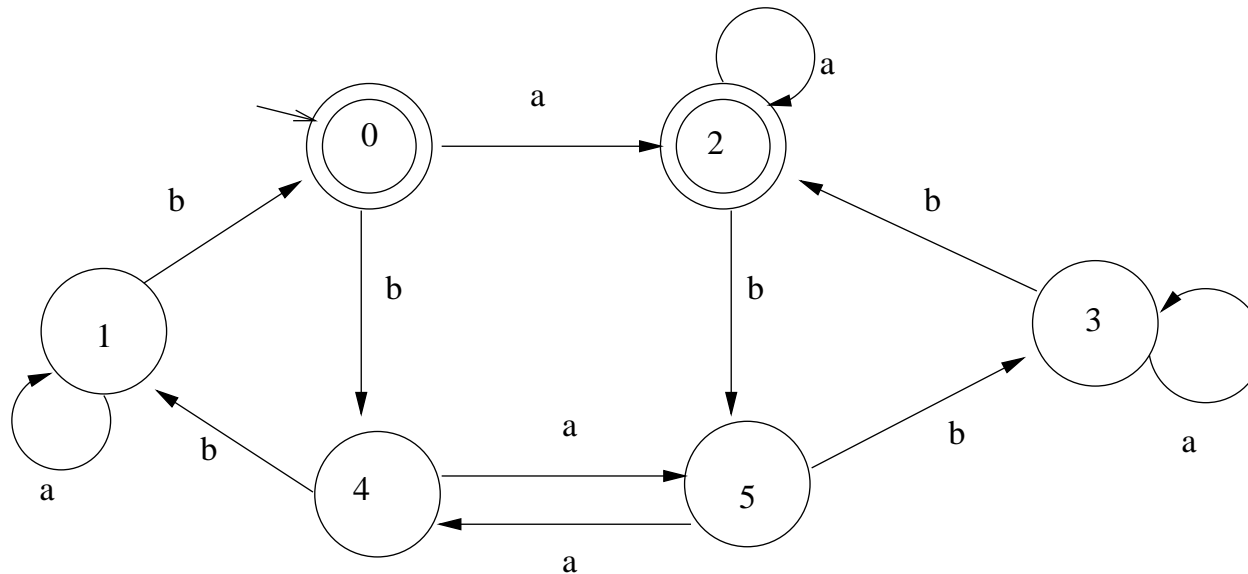
δ	a	b
1	2	3
2	2	4
3	2	3
4	2	5
5	2	3

Ejemplo de minimización de AFD

Minimizar el AFD $A_2 = (X, Q, \delta, q_0, F)$, donde:

$X = \{a, b\}$, $Q = \{0, 1, 2, 3, 4, 5\}$, $q_0 = 0$, $F = \{0, 2\}$

y δ viene dada por el siguiente diagrama:



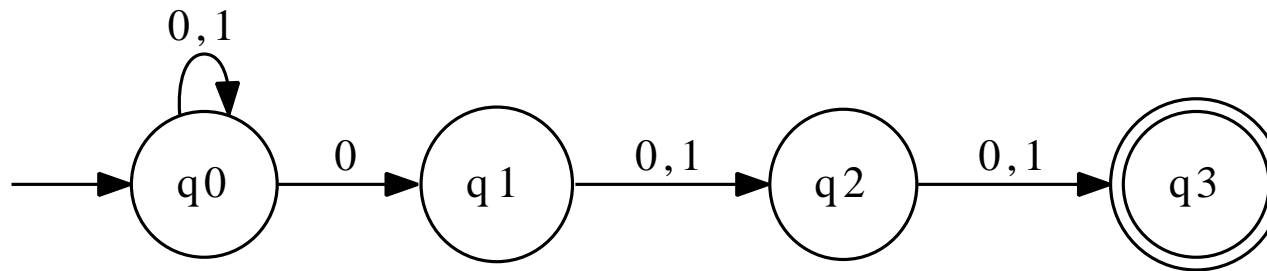
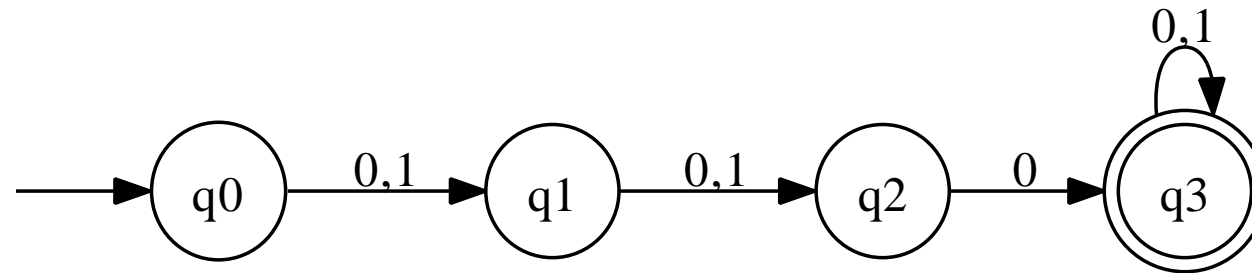
Ejemplo 4

5. AUTÓMATAS FINITOS NO DETERMINISTAS

Construir AFDs para los siguientes lenguajes:

- Palabras sobre $\{0, 1\}$ que terminan en 0
- Palabras sobre $\{0, 1\}$ que empiezan por 0
- Palabras sobre $\{0, 1\}$ cuyo penúltimo símbolo es un 0
- Palabras sobre $\{0, 1\}$ cuyo segundo símbolo es un 0
- Palabras sobre $\{0, 1\}$ cuyo antepenúltimo símbolo es un 0
- Palabras sobre $\{0, 1\}$ cuyo tercer símbolo es un 0
- ...

Soluciones deseadas:

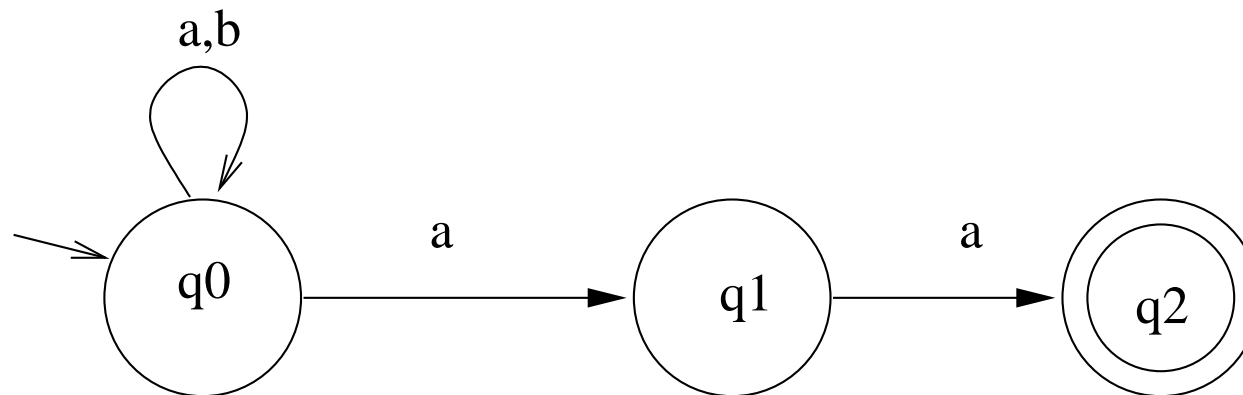


- Nótese la simetría y la similitud con las expresiones regulares $(0 + 1)(0 + 1)0(0 + 1)^*$ y $(0 + 1)^*0(0 + 1)(0 + 1)$

Autómatas Finitos No Deterministas

- Informalmente podemos ver un AFND como un AF en el que, para un determinado estado con un determinado símbolo, podemos hacer 0, 1 o más transiciones.
- Es decir, un AFD sería un caso concreto de AFND en el que se permite una y solo una transición.
- Como máquinas reconocedores de lenguajes son equivalentes a los AFD. Es decir, aunque sean más flexibles, reconocen los mismos lenguajes.

- Ejemplo: AFND que reconoce las cadenas de a's y b's terminan en dos a's.



Ejemplo 5

-
- En la representación tabular de los AFND las casillas contienen conjuntos de estados.

δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	\emptyset
q_2	\emptyset	\emptyset

-
- Formalmente definimos un Autómata Finito No Determinista (**AFND**) como una quintupla

$$(X, Q, \delta, q_0, F)$$

donde, a diferencia de los AFD, la función de transición se define como:

$$\delta : Q \times X \rightarrow P(Q)$$

siendo $P(Q)$ el conjunto de partes de Q

Descripción Instantánea de un AFND (configuración):

Es idéntica a la de los AFDs, es decir (q, w) , donde $q \in Q$ y $w \in X^*$, siendo q el estado en el que se encuentra el AFND y w la cadena que falta por procesar.

Configuración inicial: (q_0, w)

Configuración de aceptación: (q, λ) con $q \in F$

Movimiento:

En AFNDs el movimiento se define como:

$(q, aw) \vdash (q', w)$ para cada $q' \in \delta(q, a)$

- Desde una configuración podríamos movernos a varias.
- Se crea una árbol de movimientos.
- **Ejercicio:** Ver la secuencia de movimientos que provoca la cadena aabaa en el AFND del Ejemplo 5.

Extensión de la función δ a palabras: δ^*

- δ^* : dado un estado y una cadena devuelve el conjunto de estados a los que puede llegar el AFND desde dicho estado con dicha cadena.
- Se define como $\delta^*: Q \times X^* \rightarrow P(Q)$
- Se calcula a partir de la función δ :
$$\delta^*(q, \lambda) = \{q\}$$
$$\delta^*(q, wa) = \cup \{\delta(q', a) : q' \in \delta^*(q, w)\} \text{ con } a \in X, w \in X^*$$

Lenguaje Aceptado por un AFND:

- Una cadena $a_1 a_2 \dots a_n$ es aceptada por un AFND si existe una sucesión de transiciones correspondientes a arcos etiquetados con a_1, a_2, \dots, a_n , que va desde el estado inicial a algún estado de aceptación
- Equivalentemente, una cadena w es aceptada por una AFND si $\delta^*(q_0, w) \cap F \neq \emptyset$
- Dado un AFND $A = (X, Q, \delta, q_0, F)$, el **lenguaje aceptado** por A es
$$L(A) = \{w \in X^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\} \text{ ó }$$
$$L(A) = \{w \in X^* \mid (q_0, w) \vdash^* (q', \lambda), q' \in F\}.$$

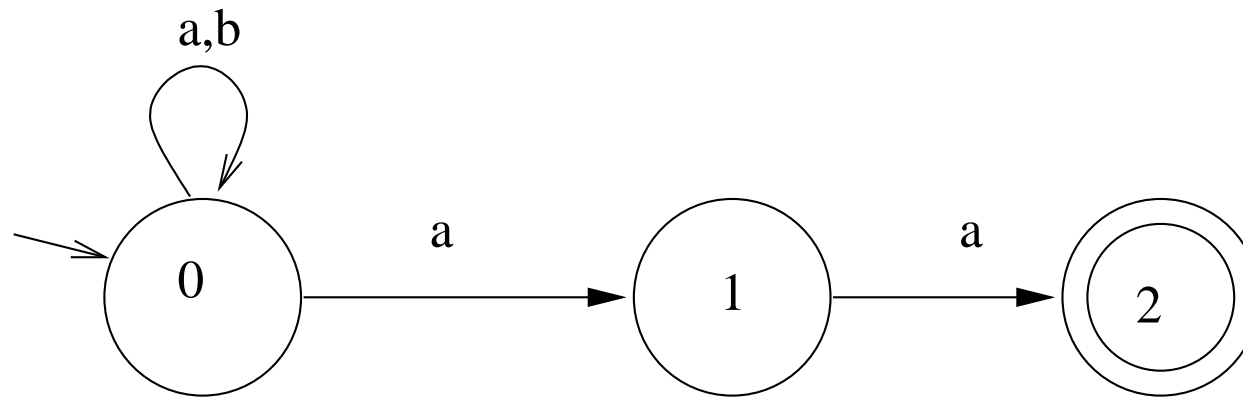
-
- El conjunto de los lenguajes aceptados por los AFND coincide con los **lenguajes regulares** (lenguajes denotados por expresiones regulares)
 - Para cada lenguaje denotado por una expresión regular, existe un AFD que reconoce todas cadenas de ese lenguaje y ninguna que no pertenezca a él.
 - Para cada AFD existe una expresión regular que denota exactamente al lenguaje reconocido por el AFD.

Equivalencia entre AFD y AFND:

- **Teorema:** Si $L \subseteq X^*$ es aceptado por un AFND, entonces existe un AFD que acepta L .
- **Teorema:** Si $L \subseteq X^*$ es aceptado por un AFD, entonces existe un AFND que acepta L .
 - Es trivial, ya que podemos ver un AFD como un caso concreto de AFND.

Teorema: Si $L \subseteq X^*$ es aceptado por un AFND, entonces existe un AFD que acepta L .

Vamos a ilustrar con un ejemplo el algoritmo que nos permite obtener un AFND equivalente a un AFD dado.



Ejemplo 6

Dado un AFND $A = (X, Q, \delta, q_0, F)$, vamos a obtener un AFD equivalente $A' = (X, Q', \delta', q'_0, F')$,

- **Paso 1:** Definición del nuevo conjunto de estados Q' .
 $Q' = P(Q)$ (conjunto de partes de Q).

En nuestro ejemplo

$$Q' = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$$

- **Paso 2:** Definición del nuevo estado inicial: $q'_0 = \{q_0\}$
- En nuestro ejemplo $q'_0 = \{0\}$

-
- **Paso 3:** Definición de la nueva función de transición δ'
Para cada nuevo estado $Q_i \in Q'$ y para cada símbolo $a \in X$

$$\delta'(Q_i, a) = \cup \{ \delta(p, a) : p \in Q_i \}$$

- En nuestro ejemplo:

$$\delta'(\{0\}, a) = \{0, 1\}$$

$$\delta'(\{0, 1\}, a) = \delta(0, a) \cup \delta(1, a) = \{0, 1\} \cup \{2\} = \{0, 1, 2\}$$

$$\delta'(\{0, 1, 2\}, b) = \delta(0, b) \cup \delta(1, b) \cup \delta(2, b) = \{0\} \cup \emptyset \cup \emptyset = \{0\}$$

...

- **Paso 4:** Eliminar de Q' los estados que no son accesibles desde el estado inicial.

En nuestro ejemplo obtendríamos

$$Q' = \{\{0\}, \{0, 1\}, \{0, 1, 2\}\}$$

-
- **Paso 5:** Definición del nuevo conjunto de estados de aceptación

$$F' = \{q \in Q' : q \cap F \neq \emptyset\}$$

- En nuestro ejemplo $F' = \{\{0, 1, 2\}\}$

-
- **NOTA:** Lo más práctico es calcular Q' y δ' de forma simultánea. Definiendo la δ' únicamente para los estados que vamos obteniendo, empezando por el estado inicial.

δ'	a	b
$\rightarrow \{0\}$	$\{0, 1\}$	$\{0\}$
$\{0, 1\}$	$\{0, 1, 2\}$	$\{0\}$
$\{0, 1, 2\}^*$	$\{0, 1, 2\}$	$\{0\}$

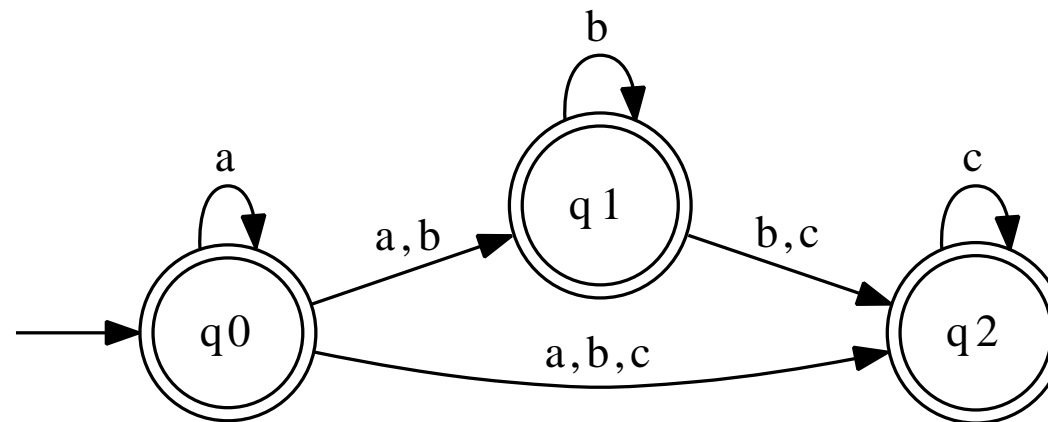
Calcular el AFD equivalente a este AFDN

δ	a	b
$\rightarrow 0$	$\{0, 3\}$	$\{0, 1\}$
1	\emptyset	$\{2\}$
2^*	$\{2\}$	$\{2\}$
3	$\{4\}$	\emptyset
4^*	$\{4\}$	$\{4\}$

6. AUTÓMATAS FINITOS NO DETERMINISTAS CON λ -MOVIMIENTOS

Obtener AFNDs para los siguientes lenguajes

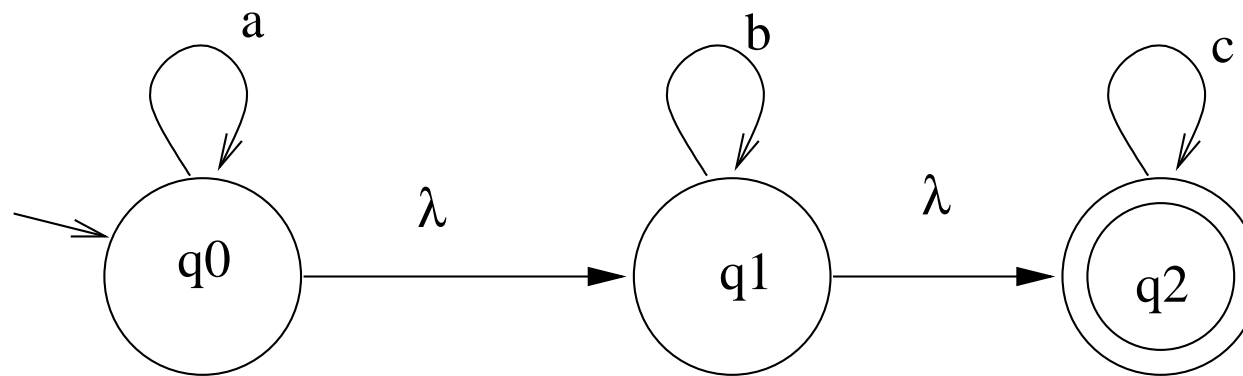
- a^*
- a^*b^*
- $a^*b^*c^*$
- ...



Autómatas Finitos No Deterministas con λ -movimientos

- Los AFND con λ -movimientos son una extensión de los AFND que permite cambios de estado con la entrada vacía (λ), es decir, permite una transición de un estado a otro sin 'leer' ningún símbolo de la cadena de entrada.
- Los autómatas aceptarán las secuencias de etiquetas que pasan por algún camino que lleve desde el estado inicial a algún estado de aceptación. Cada λ que se encuentre en el camino es 'invisible'.
- Los AFDN sin λ -movimientos son un caso particular de los AFND con λ -movimientos.

- Como máquinas reconocedores de lenguajes son equivalentes a los AFD. Es decir, aunque sean más flexibles, reconocen los mismos lenguajes.
- Ejemplo: λ -AFND que reconoce el lenguaje $a^*b^*c^*$.



Ejemplo 7

- Nótese la semejanza con la expresión regular y la existencia de simetría.

-
- En la representación tabular de los AFND con λ -movimientos las casillas contienen conjuntos de estados al igual que los AFND, pero añadimos una columna para representar los λ -movimientos.

δ	a	b	c	λ
0	{0}	\emptyset	\emptyset	{1}
1	\emptyset	{1}	\emptyset	{2}
2	\emptyset	\emptyset	{2}	\emptyset

-
- Formalmente definimos un Autómata Finito No Determinista con λ -movimientos (**λ -AFND**) como una quintupla de la forma (X, Q, δ, q_0, F) , donde ahora la función de transición se define como:

$$\delta : Q \times (X \cup \{\lambda\}) \rightarrow P(Q)$$

siendo $P(Q)$ el conjunto de partes de Q .

-
- Los conceptos de Descripción Instantánea (configuración) y movimientos son iguales que para los AFND, es decir (q, w) , donde $q \in Q$ y $w \in X^*$, siendo q el estado en el que se encuentra el λ -AFND y w la cadena que falta por procesar.

Configuración inicial: (q_0, w)

Configuración de aceptación: (q, λ) con $q \in F$

Movimiento:

En AFNDs el movimiento se define como:

$(q, aw) \vdash (q', w)$ para cada $q' \in \delta(q, a)$ y $a \in X \cup \{\lambda\}$

- Desde una configuración podríamos movernos a varias.
- Se crea una árbol de movimientos.
- **Ejercicio:** Ver la secuencia de movimientos que provoca la cadena aabcc en el AFND del Ejemplo 7.

Definición de la λ -clausura de un estado

- La λ -clausura de un estado q es el conjunto de estados a los que se puede llegar desde q a través de caminos cuyos arcos estén todos etiquetados λ .
- Se define como $\lambda-cl(q): Q \rightarrow P(Q)$
- Se calcula de la forma:
 - $q \in \lambda-cl(q)$
 - Si $p \in \lambda-cl(q) \Rightarrow \delta(p, \lambda) \subseteq \lambda-cl(q)$

Definición de la λ -clausura de un conjunto de estados

- La λ -clausura de un conjunto de estados A es la unión de las λ -clausuras de cada uno de los estados de A .
- Es decir, $\lambda-cl(A) = \bigcup \{\lambda-cl(q) : q \in A\}$
- Se cumple que: $\lambda-cl(A) \subseteq \lambda-cl(B)$, si $A \subseteq B$
- Se cumple que: $\lambda-cl(\lambda-cl(A)) = \lambda-cl(A)$

Extensión de la función δ a palabras: δ^*

- δ^* : dado un estado y una cadena, nos devuelve el conjunto de estados a los que puede llegar el AFND con λ -movimientos desde dicho estado con dicha cadena.
- Se define como $\delta^*: Q \times X^* \rightarrow P(Q)$
- Se calcula mediante
 - $\delta^*(q, \lambda) = \lambda\text{-cl}(q)$
 - $\delta^*(q, a) = \lambda\text{-cl}[\cup\{\delta(q', a) : q' \in \lambda\text{-cl}(q)\}]$ donde $a \in X$ y $q \in Q$
 - $\delta^*(q, wa) = \lambda\text{-cl}[\cup\{\delta(q', a) : q' \in \delta^*(q, w)\}]$ donde $a \in X, w \in X^*$ y $q \in Q$

Equivalencia entre AFND con λ -movimientos y AFND sin λ -movimientos :

- **Teorema:** Si $L \subseteq X^*$ es aceptado por un AFND con λ -movimientos, entonces existe un AFND sin λ -movimientos que acepta L , y, por tanto, L es un lenguaje regular.
 - Es trivial, ya que podemos ver un AFND como un caso concreto de λ -AFND.
 - La columna correspondiente a los λ -movimientos tendría \emptyset en todas las filas.

Algoritmo para calcular un AFND sin λ -movimientos equivalente a otro AFND con λ -movimientos dado.

Dado un λ -AFND $A = (X, Q, \delta, q_0, F)$, calcularemos otro $A' = (X, Q, \delta', q_0, F')$ equivalente.

- **Paso 1:** Definición de la nueva δ' .

$$\delta'(q, a) = \delta^*(q, a), \forall a \in X, \forall q \in Q$$

Para calcular cada $\delta^*(q, a)$ haremos:

1. $Q_1 = \lambda\text{-cl}(q)$
2. $Q_2 = \cup\{\delta(q', a) : q' \in Q_1\}$
3. $Q_3 = \lambda\text{-cl}(Q_2)$

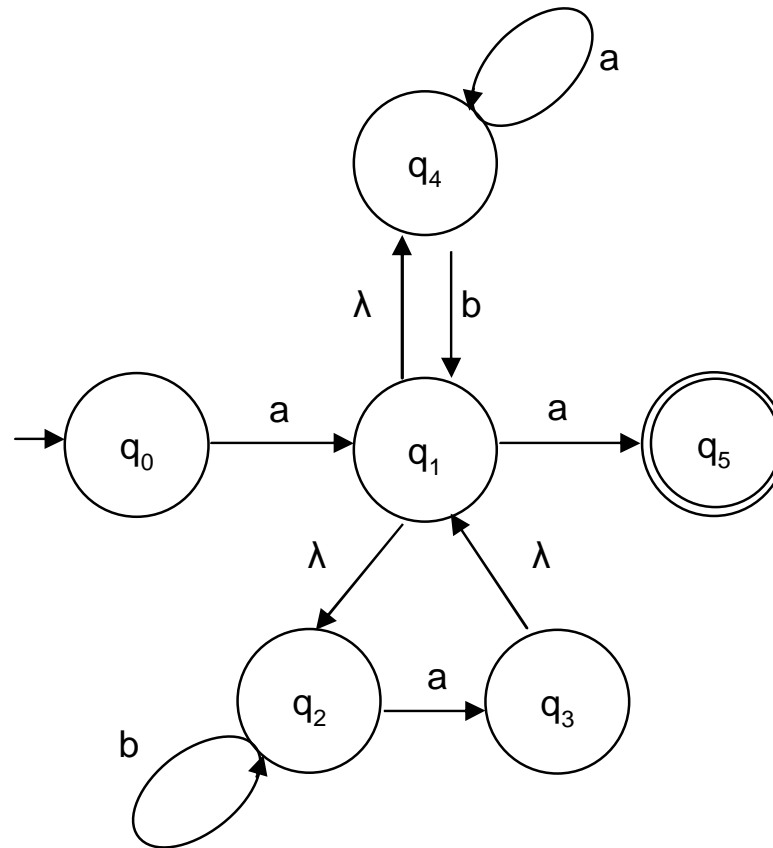
- **Paso 2:** Definición del nuevo conjunto F'

Si $\lambda\text{-cl}(q_0) \cap F \neq \emptyset$, entonces $F' = F \cup \{q_0\}$

Si $\lambda\text{-cl}(q_0) \cap F = \emptyset$, entonces $F' = F$

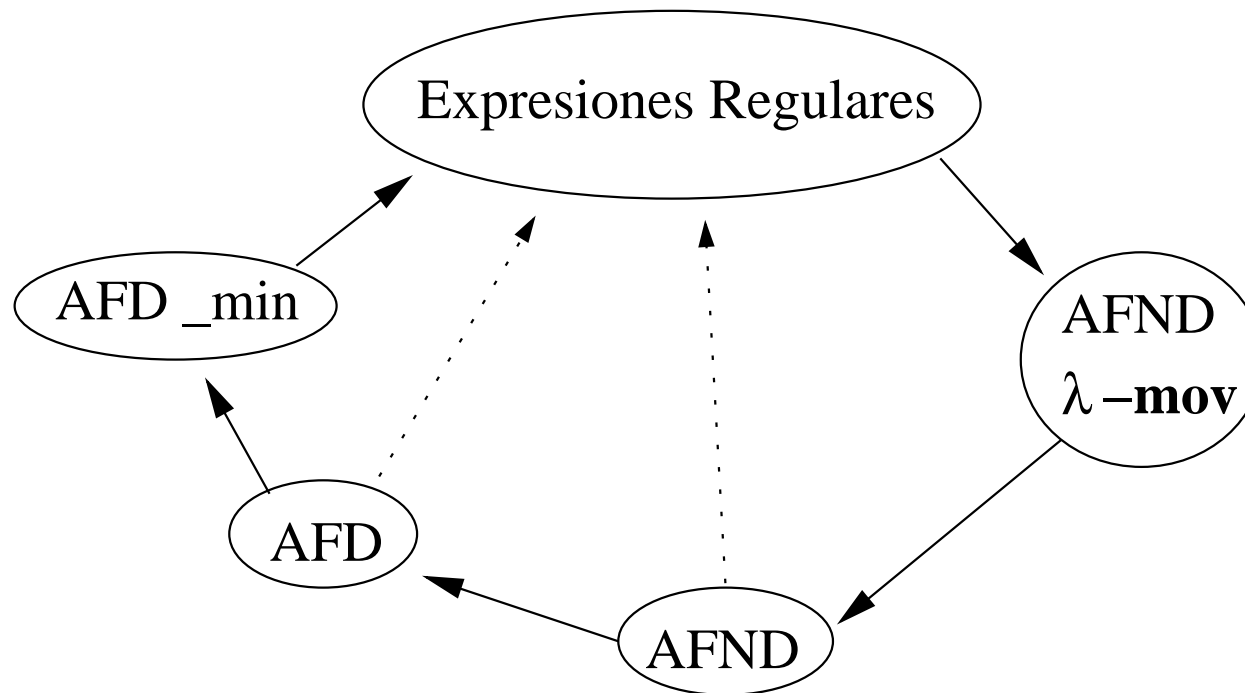
Ejemplo de eliminación de λ -movimientos

- Calcular el AFND equivalente al λ -AFND de la figura



7. EQUIVALENCIA ENTRE AUTÓMATAS FINITOS Y EXPRESIONES REGULARES

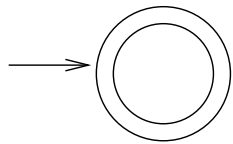
En este apartado vamos a cerrar el ciclo que muestra la siguiente figura



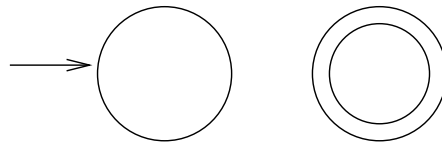
-
- **Teorema.** Si r es una expresión regular, entonces existe un AFND con λ – *movimientos* (y con a lo sumo un estado final, del que no sale ninguna transición) que acepta $L(r)$.
 - Vamos a demostrarlo mediante inducción en el número de operadores de la expresión regular. La demostración consiste en un paso básico y otro inductivo.

Paso Básico: Para una e.r. con cero operadores

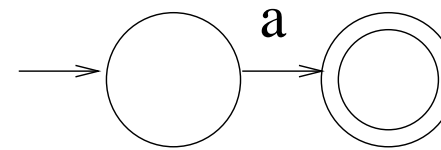
$r = \lambda$



$r = \phi$



$r = a$



Paso Inductivo: Para una e.r. con uno o más operadores

Sean r_1 y r_2 dos e.r., donde

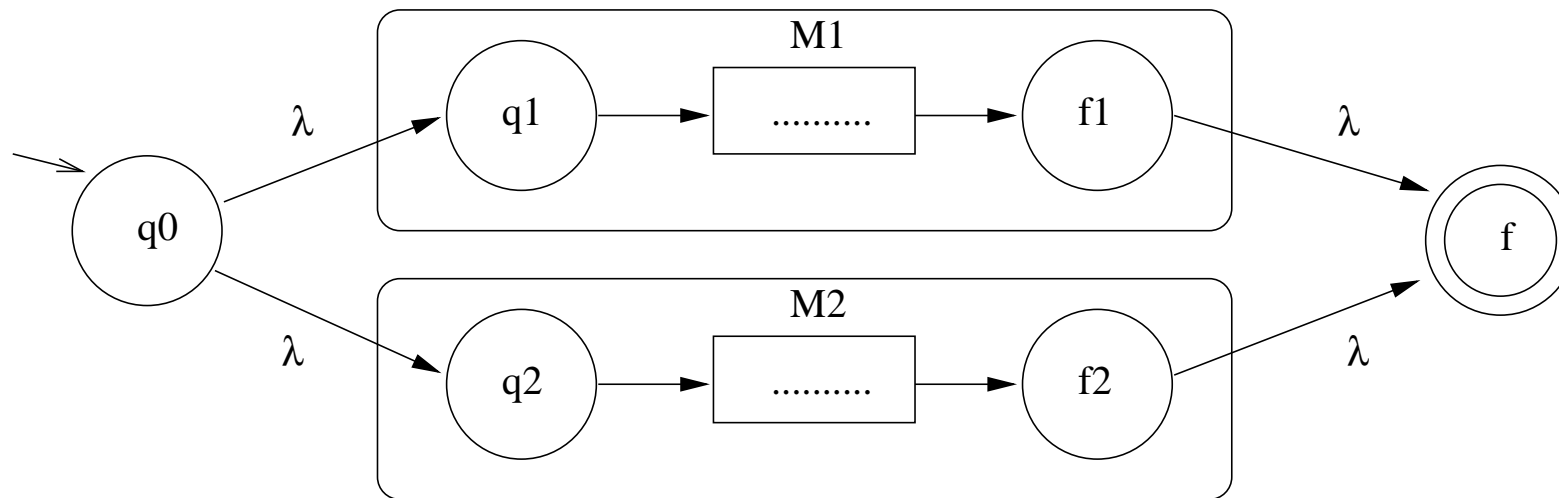
$M_1 = (X_1, Q_1, \delta_1, q_1, F_1)$ tal que $L(M_1) = L(r_1)$ y

$M_2 = (X_2, Q_2, \delta_2, q_2, F_2)$ tal que $L(M_2) = L(r_2)$.

Supongamos $Q_1 \cap Q_2 = \emptyset$ (si fuese necesario renombramos los estados) y construimos M .

- **Caso A:** $r = r_1 + r_2$ (unión)

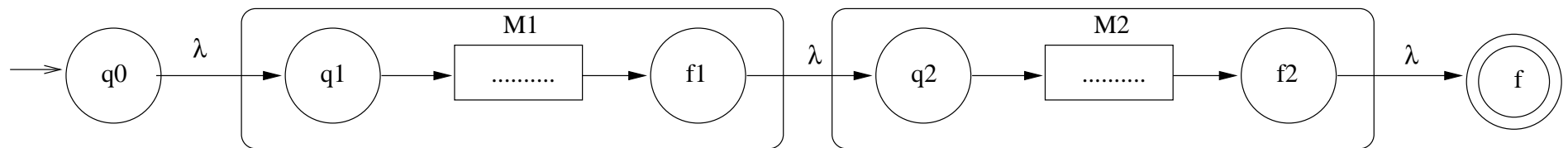
$M = (Q_1 \cup Q_2 \cup \{q_0, f\}, X_1 \cup X_2, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w | w \in L(M_1) \vee w \in L(M_2)\}$$

- **Caso B: $r = r_1 \cdot r_2$ (concatenación)**

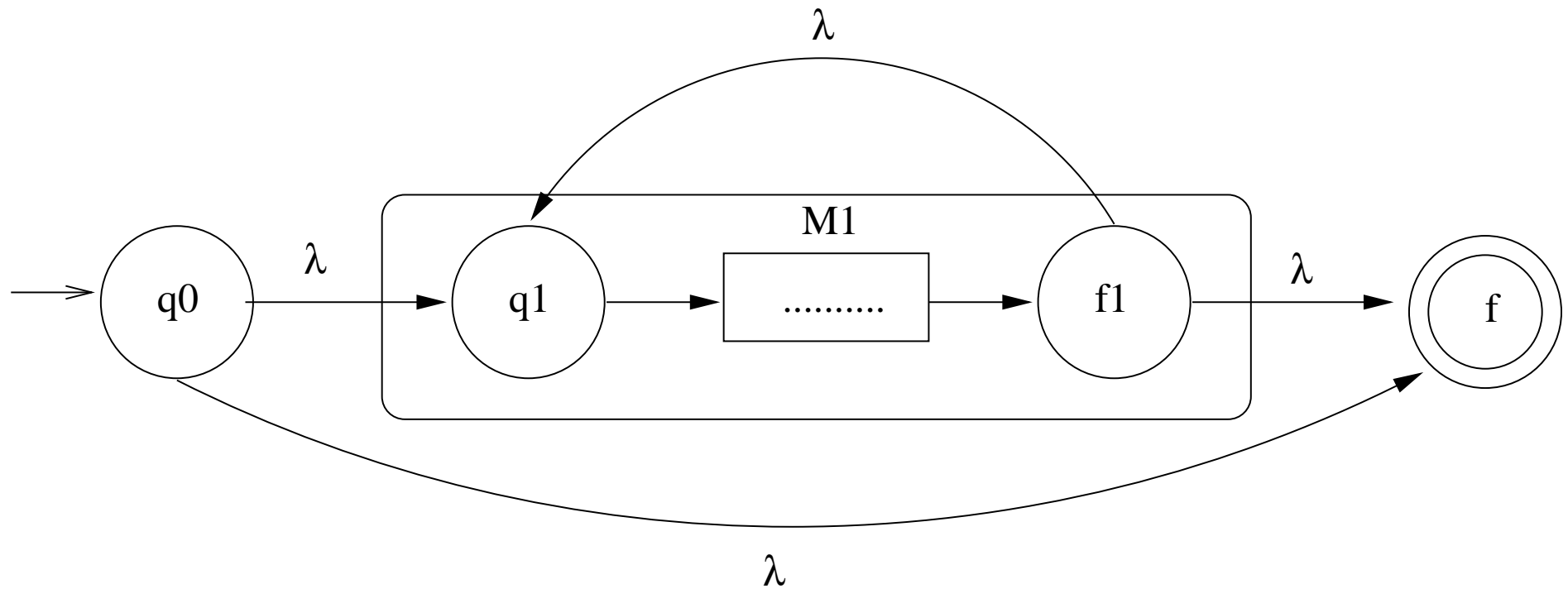
$M = (Q_1 \cup Q_2 \cup \{q_0, f\}, X_1 \cup X_2, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w_1 \cdot w_2 \mid w_1 \in L(M_1) \wedge w_2 \in L(M_2)\}$$

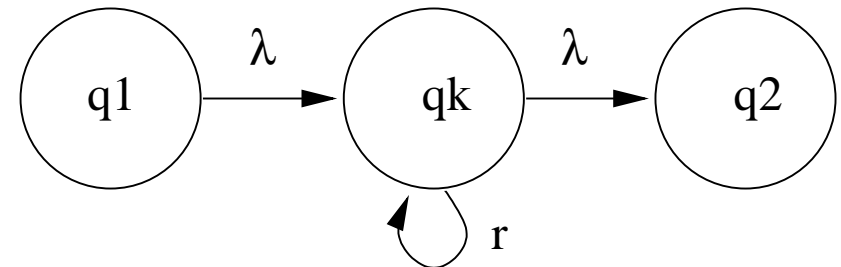
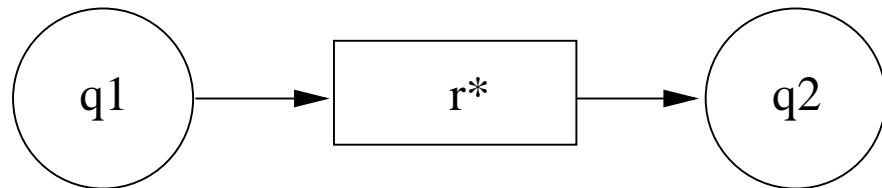
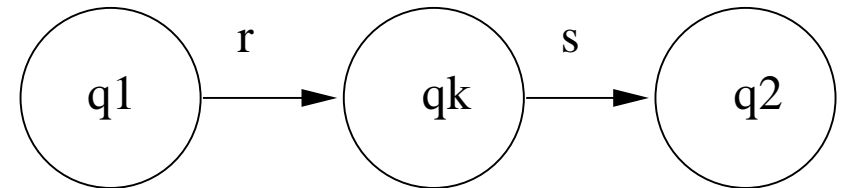
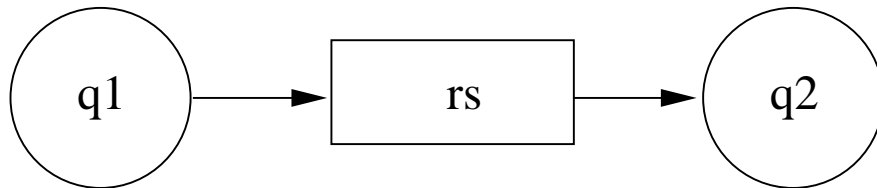
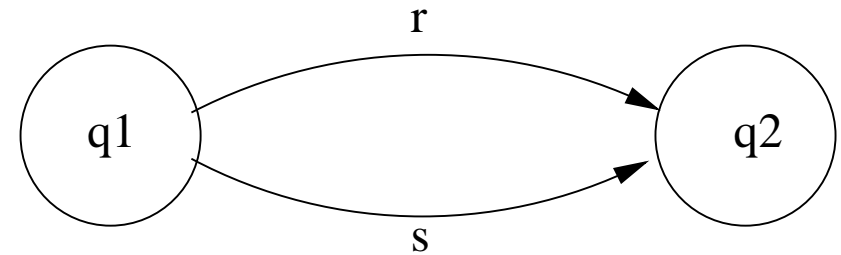
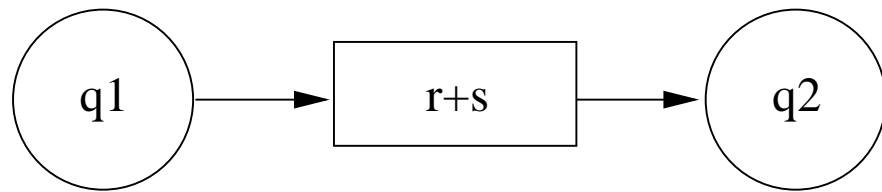
- **Caso C:** $r = r_1^*$ (**cierre**)

$M = (Q_1 \cup \{q_0, f\}, X_1, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w \mid w \in L(M_1)^*\}$$

Normas de desarrollo, para obtener un AF a partir de una e.r.



Ejemplos:

$$r = (0 + 10^*1)1(01)^*$$

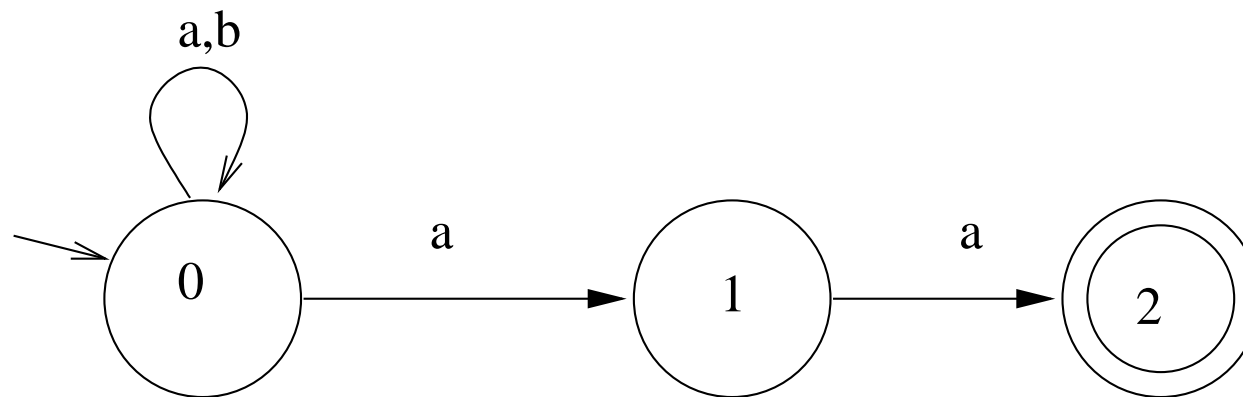
$$r = (a + b)^*(aa + bb)(a + b)^*$$

-
- **Teorema:** Si $L \subseteq X^*$ es un lenguaje aceptado por un AFD, entonces L se puede describir por una expresión regular.
 - **Corolario:** Sea X un conjunto finito, y $L \subseteq X^*$. Son equivalentes las siguientes afirmaciones:
 - L es un lenguaje regular.
 - L es un lenguaje aceptado por algún AFD.
 - L es un lenguaje aceptado por algún AFND sin λ – movimientos.
 - L es un lenguaje aceptado por algún AFND con λ – movimientos.
 - L se puede describir por una expresión regular.

Sistema de ecuaciones asociado a un autómata finito:

- Dado un AF se denota por L_q al lenguaje reconocido por el AF cuando se considera al estado q como estado inicial.
- Se denota por l_q a la e.r. que denota el lenguaje L_q , por tanto, $L_q = L(l_q)$

Ejemplo



Ejemplo 6

-
- Definimos $m(q, q') = \sum_{q' \in \delta(q, x)} x$ (unión de los símbolos que hacen transitar al AF desde el estado q al q').
 - Deber verificarse

$$l_q = \sum_{q' \in Q} m(q, q') \cdot l_{q'} + t_q$$

donde $t_q = \emptyset$ si $q \notin F$ ó $t_q = \lambda$ si $q \in F$.

- En el ejemplo anterior

$$l_0 = (a + b)l_0 + al_1$$

$$l_1 = al_2$$

$$l_2 = \lambda$$

-
- Tendremos un sistema de ecuaciones con n ecuaciones y n incógnitas (donde n es el número de estados del autómata)
 - El sistema codifica toda la información contenida en el autómata (es muy semejante a una gramática regular)
 - Queremos hallar l_{q_0}
 - El sistema tiene solución única

La regla de Arden

- Si r, s y t son expresiones regulares y $\lambda \notin s$ se cumple que

$$r = sr + t$$

si y solo si

$$r = s^*t$$

- Se utiliza para resolver sistemas de ecuaciones cuyas variables son expresiones regulares
- Recuerda a la suma de progresiones geométricas de razón $|s| < 1$ menor que 1

$$\frac{1}{1-s} = 1 + s + s^2 + s^3 + \dots$$

Algoritmo para obtener la e.r. que denota el lenguaje aceptado por un AF

- **Paso 1:** Obtenemos las ecuaciones características del AF calculando l_q para todo $q \in Q$.
- **Paso 2:** Despejar cada l_q aplicando las propiedades de las e.r. (principalmente regla de Arden y la distributiva).
- **Paso 3:** Si q_0 es el estado inicial, l_{q_0} es la e.r. que denota aquellas cadenas que partiendo de q_0 llegan a un estado final y por tanto l_{q_0} es la e.r. que denota el lenguaje reconocido por el AFD.

8. PROPIEDADES DE LOS LENGUAJES REGULARES

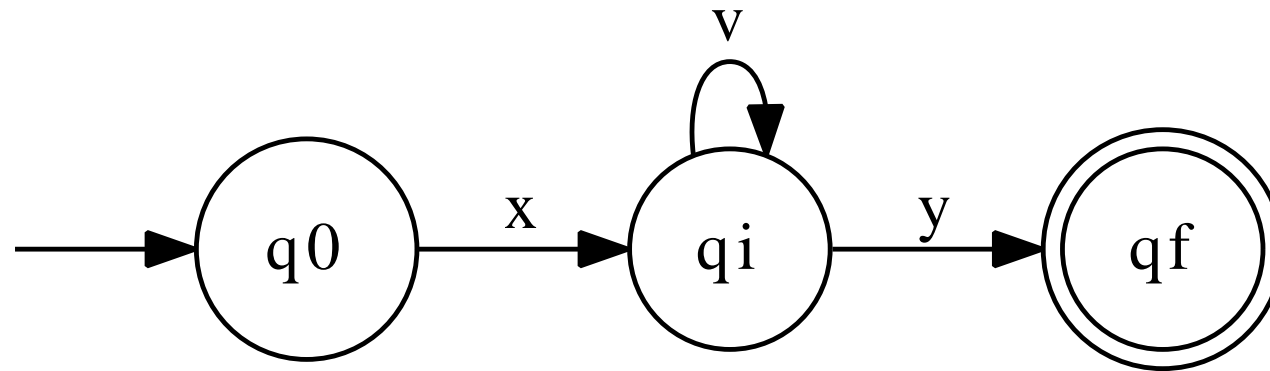
- **Teorema:** Sea X un conjunto finito, y $L \subseteq X^*$. Son equivalentes las siguientes afirmaciones:
 - L es un lenguaje regular.
 - L es un lenguaje aceptado por algún AFD.
 - L es un lenguaje aceptado por algún AFND sin λ – movimientos.
 - L es un lenguaje aceptado por algún AFND con λ – movimientos.
 - L se puede describir por una expresión regular.

LEMA DEL BOMBEO

- Dado un lenguaje ¿podemos saber si es o no regular?
- El **lema del bombeo** se utiliza para demostrar que algunos lenguajes NO son regulares.
- **Lema del bombeo:** Sea L un lenguaje aceptado por un AFD M con n estados. Sea $w \in L$ y $|w| \geq n$. Entonces, es posible descomponer w en la forma $w = xvy$, donde:
 1. $v \neq \lambda$
 2. $|xv| \leq n$
 3. $xv^i y \in L$ para todo $i \geq 0$.

Demostración:

- Si tenemos una palabra w suficientemente grande se repetirá algún estado



- La subcadena x es la que lleva hasta el primer estado en repetirse, v es la que efectúa la repetición e y es la que lleva al estado de aceptación.
- Lo que indica el lema del bombeo es que un lenguaje regular solo puede ser infinito por medio de una clausura de Kleene

-
- Sea $w = x_1x_2 \cdots x_m$ ($|w| = m$). En el proceso de aceptación de w por el autómata M , se recorren una sucesión de estados de M : $s_0, s_1, \cdots s_m$, donde $s_i = \delta^*(q_0, x_1x_2 \cdots x_i)$ será el estado en que nos encontramos después de haber leído los primeros i símbolos de w .
 - M sólo tiene n estados, estamos pasando por $m + 1$ estados ($m + 1 > n$). Por lo que dos de los s_i deben ser el mismo estado:

$$s_i = \delta^*(q_0, x_1x_2 \cdots x_i) = \delta^*(q_0, x_1x_2 \cdots x_i \cdots x_j) = s_j$$

Dicho de otra forma, el trayecto que w nos obliga a hacer a través de M contiene una cadena cerrada:

$$s_i = \delta^*(s_i, x_{i+1}x_{i+2} \cdots x_j)$$

-
- Descomponemos $w = xvy$, y v es no vacía (aunque x e y pudieran serlo). Donde $x = x_1x_2 \cdots x_i$, $v = x_{i+1}x_{i+2} \cdots x_j$ e $y = x_{j+1}x_{j+2} \cdots x_m$
 - Si cogemos s_i como el primer estado que se repite entonces $|xv| \leq n$. Además,
$$\delta^*(q_0, xy) = \delta^*(q_0, xvy) = \delta^*(q_0, xvvv) = \delta^*(q_0, xv^i y).$$
 - Como $\delta^*(q_0, xv^i y) = \delta^*(q_0, w) = s_m$ es final, cualquier $xv^i y$ pertenece a L , que es lo que se quería demostrar.

Ejemplo: $L = 0^n 1^n$ no es regular.

- Si L fuese regular, sería aceptado por un AFD de n estados. Esto, en combinación con la propiedad garantizada por el lema del bombeo, conduce a una contradicción.
- $w = 0^n 1^n \in L$ de longitud $2n$. Aplicamos las hipótesis del lema de Pumping y podemos escribir $0^n 1^n = xvy$ de forma que $xvvy \in L$.
- Como $|xv| \leq n$ se tiene que v está formada sólo por ceros, con lo que $xvvy$ tiene más ceros que unos (ya que v no puede ser vacía) y es imposible que pertenezca a L .
- Tenemos una contradicción, por lo que deducimos que L no es regular

PROPIEDADES DE CLAUSURA DE LOS LENGUAJES REGULARES

- Las propiedades de clausura expresan la idea de que, cuando uno o varios lenguajes son regulares, otros relacionados con ellos también lo son.
- Los LR son cerrados para la **unión**, **concatenación** y la **clausura** (cierres $*$ y $+$). Esto queda demostrado a partir de la propia definición de e.r.

Sean M_1 y M_2 dos λ -AFNDs

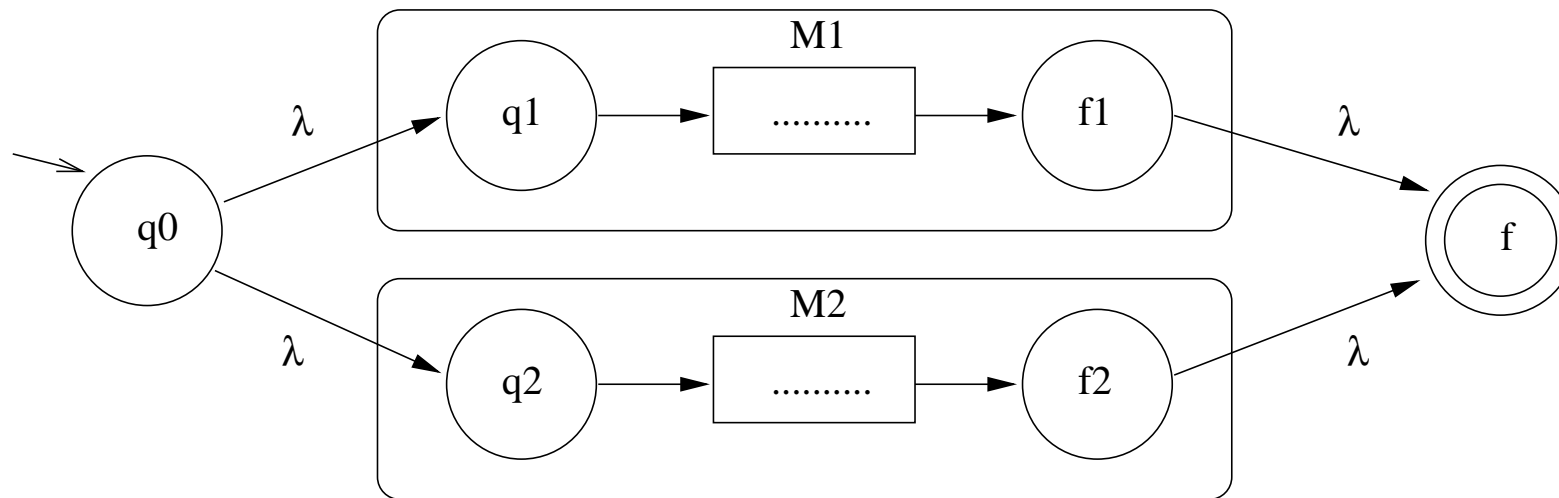
$$M_1 = (X_1, Q_1, \delta_1, q_1, F_1)$$

$$M_2 = (X_2, Q_2, \delta_2, q_2, F_2)$$

Supongamos $Q_1 \cap Q_2 = \emptyset$ (si fuese necesario renombramos los estados) y construimos M .

- **Caso A: unión**

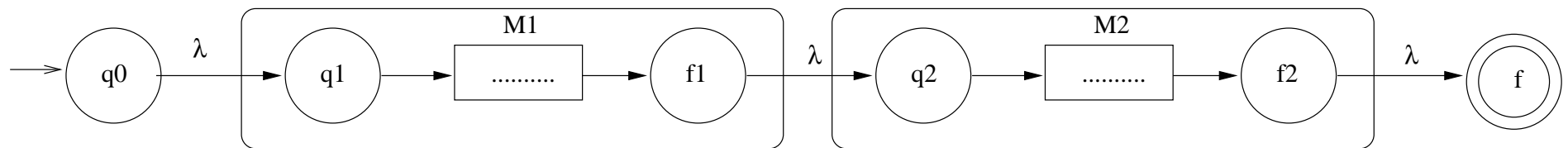
$M = (Q_1 \cup Q_2 \cup \{q_0, f\}, X_1 \cup X_2, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w | w \in L(M_1) \vee w \in L(M_2)\}$$

- **Caso B: concatenación**

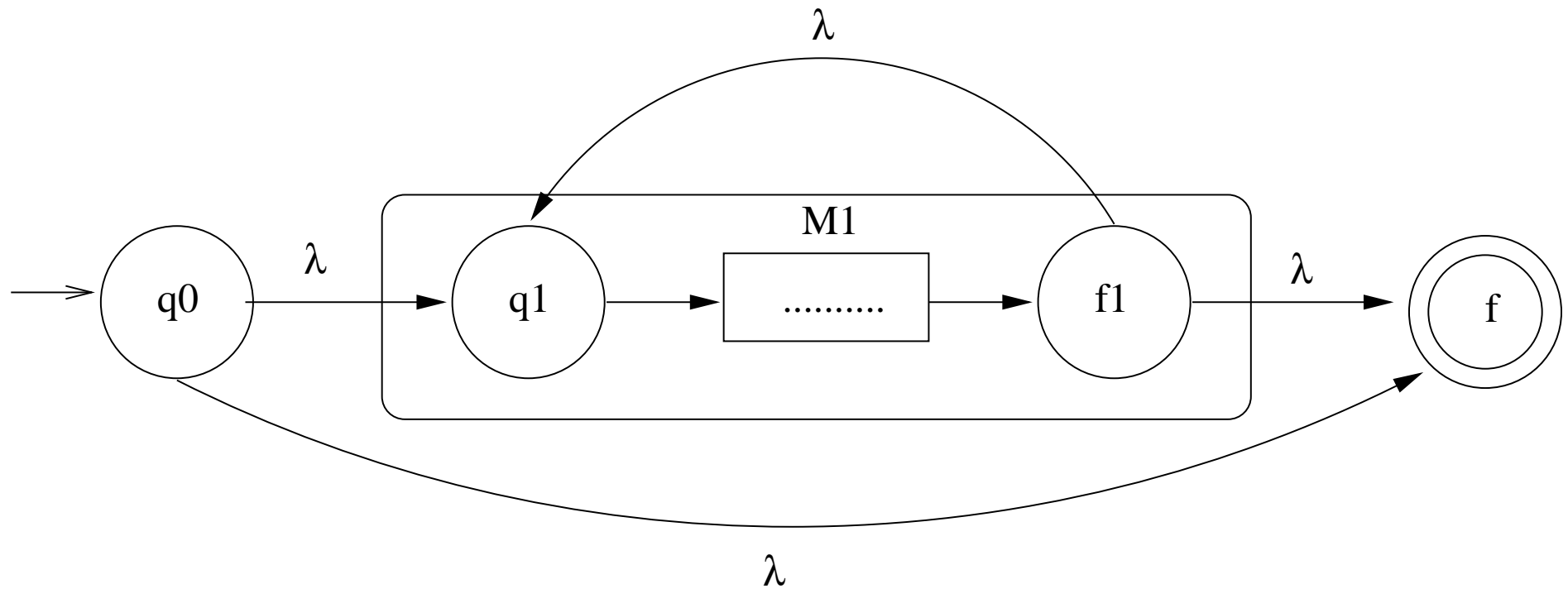
$M = (Q_1 \cup Q_2 \cup \{q_0, f\}, X_1 \cup X_2, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w_1 \cdot w_2 \mid w_1 \in L(M_1) \wedge w_2 \in L(M_2)\}$$

- **Caso C: clausura de Kleene**

$M = (Q_1 \cup \{q_0, f\}, X_1, \delta, q_0, \{f\})$, con δ :



$$L(M) = \{w | w \in L(M_1)^*\}$$

Teorema: Si $L \subseteq X^*$ es un LR, entonces su complementario también lo es.

Demostración

- Sea $L = L(A)$ para un AFD $A = (X, Q, \delta, q_0, F)$
- Se define el AFD $B = (X, Q, \delta, q_0, Q - F)$
- $w \in L(B)$ sii $\delta^*(q_0, w) \in Q - F$, lo que significa que $w \notin L(A)$
- Por tanto $\bar{L} = L(B)$

Teorema: Si L y M son LR, entonces su $L \cap M$ también lo es.

Demostración: $L \cap M = \overline{\overline{L} \cup \overline{M}}$

Construcción del AFD:

- Sea $A_L = (X_L, Q_L, \delta_L, q_L, F_L)$ y $A_M = (X_M, Q_M, \delta_M, q_M, F_M)$, tal que $L = L(A_L)$ y $M = L(A_M)$
- Calculamos el AFD $A = (X, Q_L \times Q_M, \delta, (q_L, q_M), F_L \times F_M)$
- Los estados de A son pares de estados (q, p) , tal que $q \in Q_L$ y $p \in Q_M$.
- Estado inicial: (q_L, q_M)
- Estados finales: $F_L \times F_M$
- $\delta((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$