# Tema 1: Autómatas, lenguajes y gramáticas regulares

1. Introducción
2. Lenguajes Formales
3. Expresiones Regulares
4. Autómatas Finitos
5. Propiedades de los lenguajes regulares
6. Aplicaciones de los Autómatas Finitos



# 1. Introducción

- Lenguajes: Fundamental en la Computación
  - Mediante ellos se expresan los programas, protocolos de comunicación, etc
- Se trabaja con Lenguajes Formales: Poseen reglas sintácticas y semánticas rígidas, concretas y bien definidas
  - Ejemplo: lenguaje de programación
    - Reglas que indican cómo construir cadenas válidas (palabras reservadas, identificadores de variables, etc)
    - Reglas que indican cómo combinar estas cadenas para formar programas válidos

# 1. Introducción

- 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
- También se necesita describir de forma sintética los lenguajes
  - Especificando mediante reglas, cómo se escriben programas sintácticamente correctos en un determinado lenguaje

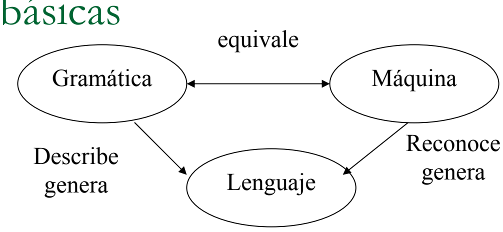


# 1. Introducción

- **Descripción de lenguajes**
  - **Gramáticas**: Describen la estructura de un lenguaje, proporcionando las reglas que determinan las combinaciones válidas de los símbolos del alfabeto.
  - **Expresión regular**: Describen de manera declarativa las cadenas aceptables o pertenecientes a un lenguaje regular.
- **Reconocimiento de lenguajes**
  - **Autómatas**: Mecanismos o máquinas abstractas que son dispositivos teóricos capaces de recibir, procesar y transmitir información (cadenas de un lenguaje).

## 2. Lenguajes Formales
### 2.1. Definiciones básicas

- **Relación Gramática – Autómata – Lenguaje**

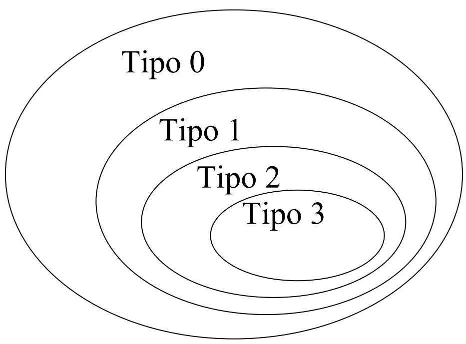



| Gramática | Lenguaje | Autómata |
|---|---|---|
| Tipo 0: Gramática Sin Restricciones | Recursivamente enumerable / Sin Restricciones | Máquina de Turing (MT) |
| Tipo 1: Gramática Sensible al Contexto | Dependiente del contexto | Autómata Linealmente Acotado (ALA) |
| Tipo 2: Gramática Libre de Contexto | Independiente del Contexto | Autómata con Pila (AP) |
| Tipo 3: Gramática Regular | Regular | Autómata Finito (AF) |



## 2. Lenguajes Formales
### 2.1. Definiciones básicas

- **Relación de inclusión: Jerarquía de Chomsky**



## 2. Lenguajes Formales
### 2.1. Definiciones básicas

- **Alfabeto.** Conjunto finito y no vacío de símbolos.
  - **Ejemplo.** $X_1$= {0, 1} $X_2$= {a, b, c} $X_3$= {00, 01, 10, 11}
  - Es válido si **no se genera ambigüedad** en la formación de las palabras
    - **Ejemplo**. X= {00, 11, 100, 111}; palabra: 11100 ¿11·100 o 111·00?
- **Palabras** o **cadenas**. Secuencias finitas de símbolos de un alfabeto.
  - **Ejemplo.** Sea X = {a, o, l, h}, son palabras $p_1$= hola, $p_2$= ola
- **Longitud de una palabra**. Nº de símbolos que la forman
  - **Ejemplo.** Sea X = {0, 1, 2, ..., 9}
    - **41** |41| = 2 **23456** |23456| = 5



## 2. Lenguajes Formales
### 2.1. Definiciones básicas

- Palabra Vacía, $\lambda$. Dado un alfabeto X, $\lambda$ se define como la única palabra construida con 0 símbolos del alfabeto.
  - Aunque se represente por un carácter simple, 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, ... }

## 2. Lenguajes Formales
### 2.1. Definiciones básicas

- Lenguaje sobre un alfabeto X, L(X). Todo 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 }
    - $L_2$={λ, a, aa, aaa, aaaa, ...}
  - Ejemplo. Sea el alfabeto de los símbolos ASCII. Un lenguaje sobre este alfabeto serían todas aquellas cadenas que representen identificadores válidos en Java.
    - Empieza con letra, $, _ y sigue con cero o más letras, dígitos, $ o _
    - L1={a, aux, cont, i1, i_2, ……}
  - Ejemplo. Sea el alfabeto de todos los identificadores, signos de puntuación, palabras reservadas en Java. Un lenguaje sobre este alfabeto sería el formado por todos los programas bien construídos.



## 2. Lenguajes Formales
### 2.2. Operaciones con palabras

- **Concatenación.** Dado un alfabeto X, sean $a = a_1a_2....a_n$ y $b = b_1b_2....b_m$ palabras donde $\forall i\ a_i \in X$ y $\forall j\ b_j \in X$. La **concatenación de las palabras** a y b, **a.b**, es una palabra formada por los símbolos de a seguidos de los símbolos de b, es decir, **$a.b = a_1a_2....a_nb_1b_2....b_m$.**
  - Además se cumple que |a.b| = |a| + |b|
  - λ elemento neutro para la concatenación. Para cualquier palabra x, x.λ = λ.x = x.
  - **Ejemplo.**
    - boca.dillo = bocadillo
    - coca.cola = cocacola

## 2. Lenguajes Formales
### 2.2. Operaciones con palabras

- **Potencia**. La **potencia i-ésima** de una palabra x, $x^i$, se forma por la concatenación i veces de x. Por definición, para toda palabra x, se cumple que $x^0 = \lambda$
  - **Ejemplo.** $a^3$ = aaa   $(ac)^2$ = acac
- **Potencia k.** Dado un alfabeto X y un nº no negativo k $\in \mathbb{N}$, definimos $X^k$ = {x | x es una palabra sobre X y |x| = k}
  - **Ejemplo.** X={0 ,1}
    - $X^0 = \{\lambda\}$
    - $X = X^1 = \{0, 1\}$
    - $X^2 = \{00, 01, 10, 11\}$



## 2. Lenguajes Formales
### 2.2. Operaciones con palabras

- **Cierre positivo y estrella.** Dado un alfabeto X, definimos :
  - $X^* = \bigcup_{k=0}^{\infty} X^k = X^0 \cup X^1 \cup \ldots$
    - $\{0,1\}^*$
  - $X^+ = \bigcup_{k=1}^{\infty} X^k = X^1 \cup X^2 \ldots$
    - $\{0,1\}^+$

## 2. Lenguajes Formales
### 2.3. Operaciones con lenguajes

- **Unión o alternativa**
  - $L_1 \cup L_2 = \{ x / x \in L_1 \vee x \in L_2 \}$
  - **Ejemplo.**
    - $L_1=\{a, b\}$ y $L_2 =\{c, d\}$
    - $L_1 \cup L_2 = \{a, b, c, d\}$
- **Concatenación**
  - $L_1 \cdot L_2 = \{ x_1 x_2 / x_1 \in L_1 \wedge x_2 \in L_2 \}$
  - **Ejemplo.**
    - $L_1=\{a, b\}$ , $L_2 =\{c, d\}$ y $L_3 =\varnothing$
    - $L_1 L_2 = \{ac, ad, bc, bd\}$
    - $L_1 L_3 = \varnothing$
- **Intersección**
  - $L_1 \cap L_2 = \{ x / x \in L_1 \wedge x \in L_2 \}$



## 2. Lenguajes Formales
### 2.3. Operaciones con lenguajes

- **Potencia de un lenguaje**
  - $L^n = \{ x_1 \cdot x_2 \cdot \ldots \cdot x_n \mid x_i \in L \}$
  - $L^0 = \{\lambda\}$
  - **Ejemplo.**
    - $L=\{a, b\}$
    - $L^0 = \{\lambda\}$
    - $L^1=\{a, b\}$
    - $L^2 =\{aa, ab, ba, bb\}$
- **Clausura o cierre positivo**
  - $L^+ = L^1 \cup L^2 \cup L^3 \ldots$
- **Cierre u Operación Estrella** (Clausura de Kleene)
  - $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

# 3. Expresiones Regulares
## 3.1. Definiciones

- Definen las cadenas válidas de un lenguaje mediante una descripción algebraica (fórmula)
  - Los lenguajes que pueden describirse mediante expresiones regulares se denominan **lenguajes regulares**
- Se utilizan como lenguaje de entrada en muchos sistemas de proceso de cadenas
  - grep de UNIX y otros similares en navegadores, procesadores, etc. usan una notación similar a las expresiones regulares para describir los patrones que el usuario quiere encontrar en un archivo
  - Ejemplo. Buscar en word [a-f] busca cualquier carácter de *a* a *f*



# 3. Expresiones Regulares
## 3.1. Definiciones

- Sea X un alfabeto finito, las expresiones regulares sobre X y los lenguajes que denotan se definen recursivamente como
  - $\varnothing$ es una expresión regular y denota al conjunto vacío.
  - $\lambda$ es una expresión regular y denota a $\{\lambda\} \subset X^*$
  - Si $x \in X$ entonces **x** es una e.r. y denota al lenguaje $\{x\} \subset X^*$
    - Ejemplo. X={a, b, c, d}, la e.r. a, representa L={a}
  - Si r y s son e.r. que denotan a los lenguajes R y $S \subset X^*$
    - **r+s** o **r|s** denota al lenguaje RUS
      - r=a, s=b, r+s=a+b L(r+s)={a, b}
    - **r·s** denota al lenguaje R·S (el . suele omitirse)
      - r.s=a.b L(r.s)={a.b}
    - **$r^*$** denota al lenguaje $R^*$
      - $r^*=a^*$ $L(a^*)=\{\lambda, a, a.a, a.a.a, a.a.a.a, \ldots\ldots\}$
    - **$r^+$** denota al lenguaje $R^+$
      - $r^+=a^+$ $L(a^+)=\{a, a.a, a.a.a, a.a.a.a, \ldots\ldots\}$

## 3. Expresiones Regulares
### 3.1. Definiciones

- Precedencia en la utilización de los operadores
  - 1º Operación cierre y cierre positivo. (* $^{+}$) + prioridad
  - 2º Operación concatenación (·)
  - 3º Alternativa. (+, |) - prioridad
  - Se permite el uso de paréntesis para indicar la precedencia



## 3. Expresiones Regulares
### 3.1. Definiciones

- Ejemplos. Sea X={a,b}

| Expresión regular | Lenguaje |
|---|---|
| a+b+(a.b) | {a, b, ab} |
| a(a+b) | {aa, ab} |
| (bb)* | {λ, bb, bbbb, bbbb, bbbbbb, …….} |
| a(a+b)* | |
| | Palabras con un número par de a |
| | Palabras que terminan en a |
| a.b* | |
| | Empiezan y terminan con la = letra |
| | Igual número de a que de b |

## 3. Expresiones Regulares
### 3.1. Definiciones

- Ejemplos

| Expresión regular | Lenguaje |
|---|---|
| 0+1+2+3+4+5+6+7+8+9 | Dígitos |
| | Números naturales |
| | Nº enteros |
| | Nº reales sin exponente en Java |
| | Identificadores en Java |
| | Expresión de adición en Java |



## 3. Expresiones Regulares
### 3.2. Equivalencia de expresiones regulares

- Dos expresiones regulares son equivalentes si designan al mismo lenguaje regular.
- Sean x, y, z expresiones regulares.
  - + es asociativa: $x + ( y + z ) = ( x + y ) + z$
  - + es conmutativa: $x + y = y + x$
  - · es asociativa: $x \cdot ( y \cdot z ) = ( x \cdot y ) \cdot z$
  - · es distributiva respecto a +
    - $x \cdot ( y + z ) = x \cdot y + x \cdot z$
    - $( x + y ) \cdot z = x \cdot z + y \cdot z$
  - · tiene elemento neutro $\lambda$: $x \cdot \lambda = \lambda \cdot x = x$
  - + tiene elemento neutro $\phi$ : $x + \phi = \phi + x = x$
  - $\lambda^* = \lambda$

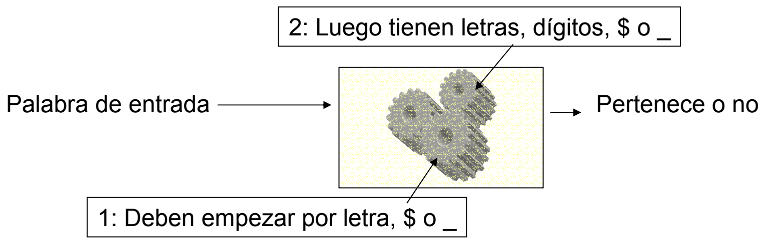
## 3. Expresiones Regulares
### 3.2. Equivalencia de expresiones regulares

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



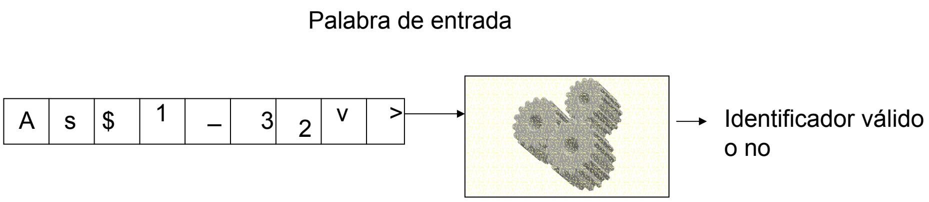
## 4. Autómatas Finitos
### 4.1. Introducción

- Dispositivo para procesar palabras pertenecientes a un lenguaje regular.
- Se construyen siguiendo las reglas de formación de palabras de un lenguaje: recibe una palabra y determina si pertenece o no al lenguaje, es decir, si sigue las reglas



# 4. Autómatas Finitos
## 4.1. Introducción

- Disponen de una cinta de entrada, de donde van leyendo las palabras, procesando sus símbolos secuencialmente, de izquierda a derecha






# 4. Autómatas Finitos
## 4.1. Introducción

- En cada momento, almacenan información acerca de la historia del sistema. Cada almacén de información se denomina estado
- Tienen un número finito de estados
  - Es necesario diseñarlos con cuidado para recordar la información interesante
  - Estado: El primer símbolo fue letra, $ o _
  - No es interesante recordar qué fue exactamente.
- En cada momento, el AF puede estar en alguno de ellos y, según va leyendo símbolos podrá ir cambiando de estado, por lo que el estado nos va a decir qué tipo de cadena ha llegado hasta ese momento.
- Habrá estados que activen la luz verde y otro que activen la roja.
- Ejemplo de Autómata finito: los analizadores léxicos
  - Ejemplo: Analizador léxico que determine si un identificador es válido en Java

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Un **Autómata Finito Determinista**, **AFD**, es una quíntupla **AFD = (X, Q, δ, $q_0$, F)** donde
  - X es un conjunto finito denominado **alfabeto de entrada**;
  - Q es un conjunto finito llamado **conjunto de estados**;
  - δ: X×Q→Q llamada **función de transición de estados**;
  - $q_0 \in Q$ es el llamado **estado inicial**;
  - **F⊂Q** recibe el nombre de **conjunto de estados finales** que corresponde a los estados **q** ∈**Q** en que se acepta la cadena de entrada.
- Determinista hace referencia al hecho de que
  - δ debe estar definida para cada estado y símbolo del alfabeto
  - Para cada símbolo de entrada, existe un único estado al que el AFD puede llegar partiendo del actual



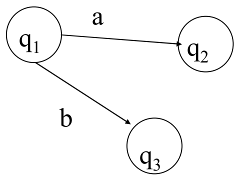
## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Representación de la función de transición de un AFD
  - **Tabla de Transiciones**
    - Se representa δ mediante una matriz de |Q| x |X|
      - Filas: estados q ∈ Q, estado inicial precedido de → y estados finales de *
      - Columnas: símbolos de entrada a∈X y en (a,q) el estado determinado por δ(a,q)
    - **Ejemplo.**

| Q | X | |
|---|---|---|
| | a | b |
| $q_1$ | $q_2$ | $q_3$ |
| $q_2$ | …. | …… |

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Representación de la función de transición de un AFD
  - **Diagrama de Transición de estados**
    - Se crea un grafo dirigido en el que, para cada estado $q_i \in Q$ se crea un nodo y, para cada transición $\delta(x_j, q_i) = q_k$, se crea un arco de $q_i$ a $q_k$ etiquetado con $x_j$.
    - El estado inicial tendrá un arco entrante no etiquetado y los estados finales estarán rodeados de doble círculo.
    - **Ejemplo.**






## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplo. AFD que reconozca las palabras formadas por 'a' y 'b' con un número par de 'a' y un número par de 'b'
  - X={ a, b}
  - Q
    - $q_0$: El número de 'a' es par y el número de 'b' es par
    - $q_1$: El número de 'a' es par y el número de 'b' es impar
    - $q_2$: El número de 'a' es impar y el número de 'b' es par
    - $q_3$: El número de 'a' es impar y el número de 'b' es impar
  - $q_0$
  - $F=\{q_0\}$

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- $\delta$: X × Q→Q

| q origen | Símbolo a la entrada (X) | q destino |
|---|---|---|
| $q_0$ | a | $q_2$ |
| $q_0$ | b | $q_1$ |
| $q_1$ | a | $q_3$ |
| $q_1$ | b | $q_0$ |
| $q_2$ | a | $q_0$ |
| $q_2$ | b | $q_3$ |
| $q_3$ | a | $q_1$ |
| $q_3$ | b | $q_2$ |



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplo. AFD que reconozca los comentarios en un programa Java de la forma
  - /* texto, texto y más texto */
  - X={ Conjunto de caracteres ASCII}
  - Q
    - $q_0$: Todavía no ha llegado el símbolo /
    - $q_1$: Ha llegado una /
    - $q_2$: Inmediatamente después de llegar /, ha llegado un *
    - $q_3$: Después de encontrar la secuencia /*, llega *
    - $q_4$: Inmediatamente después de llegar el * de $q_3$ llega /
  - $q_0$
  - F={$q_4$}

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- δ: X × Q→Q

| q origen | Símbolo a la entrada (X) | q destino |
|---|---|---|
| $q_0$ | Cualquier símbolo excepto / | $q_0$ |
| $q_0$ | / | $q_1$ |
| $q_1$ | Cualquier símbolo excepto * | $q_0$ |
| $q_1$ | * | $q_2$ |
| $q_2$ | Cualquier símbolo excepto * | $q_2$ |
| $q_2$ | * | $q_3$ |
| $q_3$ | Cualquier símbolo excepto / | $q_2$ |
| $q_3$ | / | $q_4$ |
| $q_4$ | Cualquier símbolo excepto / | $q_0$ |
| $q_4$ | / | $q_1$ |



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Representar mediante un grafo el AFD de la página 28-39
- Idem, mediante tabla de transiciones
- Idem AFD páginas 30-31

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplos
  - AFD que reconozca las palabras formadas por símbolos del alfabeto X={0,1} en el que, el nº de veces que aparece el símbolo 1 es un número par
  - AFD que reconozca las palabras formadas por símbolos del alfabeto X={0,1} en el que, el nº de veces que aparece el símbolo 1 es un número impar
  - AFD que reconozca las palabras formadas por símbolos del alfabeto X={0,1} que empiecen y terminen con el mismo símbolo.
    - Ojo con casos como 0 que empiezan y terminan con el mismo símbolo



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplos. AFD que reconozcan
  - Dígitos
  - Números naturales
  - Nº enteros
  - Nº reales sin exponente en Java
  - Identificadores en Java
  - Expresión aritmética de adición en Java
  - Expresión de asignación en Java

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Procesamiento de palabras en un AFD
  - El “lenguaje del AFD” es el conjunto de palabras aceptadas por el mismo
  - Supongamos que $a_1a_2a_3a_4a_5$ es una secuencia de símbolos
  - Comenzamos con el AFD en el estado inicial
  - Se procesa el primer símbolo $a_1$ aplicando $\delta(a_1, q_0)$ para encontrar el estado al que pasa el AFD tras procesar el primer símbolo de la entrada
  - Supongamos que fue $q_1$. Se procesa el segundo símbolo aplicando $\delta(a_2, q_1)$
  - Continuamos así hasta procesar el último de los símbolos, llegando al final a un estado $q_n$.
  - Si $q_n$ pertenece a F, la palabra es aceptada. Si no, es rechazada



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Configuración
  - Descripción instantánea de un AFD
  - $(w, q)$, $q \in Q$ , $w \in X^*$
    - q representa el estado actual y w la cadena que queda por procesar
  - Configuración inicial : $(w, q_0)$
  - Configuración final o de aceptación de palabra: $(\lambda, q_f)$ , $q_f \in F$
- Movimiento
  - $(aw, q) \vdash (w, q')$ , $\delta(a, q) = q'$
  - $(w, q) \vdash^* (w', q')$ si hay cero o más configuraciones intermedias tales que

$$(w,q) \vdash (w_1, q_1) \vdash (w_2, q_2) \vdash \ldots \vdash (w', q')$$

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplo: Sea el AFD de la página 30-31 y la palabra /* no comment */
  - Configuración inicial
  - Procese la palabra representando el procesamiento mediante configuraciones y movimientos
  - ¿Cuál es la configuración final?



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Extensión a palabras de la función de transición
  - Si δ es la función de transición, la función de transición extendida se llamará δ* y se construirá a partir de δ
  - Describe el comportamiento de un AFD partiendo de un estado y procesando una cadena de símbolos
  - Recibe un estado q y una cadena w y devuelve un estado p, estado que alcanza el autómata cuando comienza en el estado q y procesa la secuencia de símbolos w
  - Se define la función de transición extendida a partir de
    - $\delta : \mathbf{X} \times \mathbf{Q} \rightarrow \mathbf{Q}$ a $\delta^* : \mathbf{X}^* \times \mathbf{Q} \rightarrow \mathbf{Q}$ de la forma siguiente:

$$\delta^*(\lambda, q) = q$$

$$\delta^*(wx, q) = \delta(x, \delta^*(w, q))$$

donde $x \in X$, $w \in X^*$ y $q \in Q$

# 4. Autómatas Finitos
## 4.2. Autómatas finitos deterministas (AFD)

- Lenguaje aceptado por un AFD
  - **Aceptación de palabras**
    - Una **palabra w** será **aceptada** por un AFD $\Leftrightarrow$ $\boldsymbol{\delta^*(w, q_0) \in F}$
  - **Lenguaje reconocido por un AFD**
    - Sea el AFD M= (X, Q, $\delta$, $q_0$, F), el **lenguaje aceptado por M** que se representa como L(M) es el conjunto $L(M)=\{w \in X^* \mid \delta^*(w, q_0) \in F\}$.
    - Al conjunto de los lenguajes reconocidos por algún AF se les denomina **lenguajes regulares.**
    - Para cada lenguaje regular, existe un AF que reconoce palabras de ese lenguaje y no reconoce ninguna palabra que no pertenezca al lenguaje.



# 4. Autómatas Finitos
## 4.2. Autómatas finitos deterministas (AFD)

- Ejemplos : Describa informalmente el lenguaje. Busque una e.r
  - Considere el AFD siguiente $AFD_1$=({0,1}, {A, B}, δ, A, B) donde:

| δ | 0 | 1 |
|---|---|---|
| →A | A | B |
| B* | B | A |

  - Considere el AFD siguiente $AFD_2$=({0,1}, {A, B, C}, δ, A, {A,B}) donde:

| δ | 0 | 1 |
|---|---|---|
| →A* | B | A |
| B* | C | A |
| C | C | C |

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Autómatas Equivalentes
  - Dos autómatas son equivalentes si aceptan el mismo lenguaje
- Equivalencia de estados
  - **Estados equivalentes.**
    - Sea el AFD=(X, Q, $q_0$, $\delta$ , F) dos estados p y q son equivalentes ($p \equiv q$) si para toda cadena de entrada w, **$\delta^*$ (w, p) es un estado final si y sólo si $\delta^*$ (w, q) también lo es**
    - Es decir, serán equivalentes si tienen el mismo comportamiento ante toda palabra.
  - **Estados distinguibles.** p se distingue de q sii **$\exists$ w $\in$ X* | $\delta^*$ (w, p) $\in$ F y $\delta^*$ (w, q) $\notin$ F o viceversa**.
  - Si dos estados son equivalentes, pueden ser sustituidos por uno sólo con el mismo comportamiento que los originales
  - Consecuencia: Existen algoritmos para minimizar un AF, es decir, hallar otro AF equivalente con el número mínimo de estados



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Minimización de Autómatas Finitos Deterministas. Algoritmo
  - Partimos de un AFD A=(X,Q,$\delta$, $q_0$,F). Hallaremos un AFD que acepta L(A) y tiene el menor número de estados posibles A´=(X,Q´,$\delta$´, $q_0$´,F´).
  - El algoritmo consiste en determinar las clases de equivalencia en el conjunto de estados.
  - **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.

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- **Paso 2.1**.- Construir una partición inicial $\Pi$ dividiendo el conjunto Q en dos grupos: **estados finales** F, y **no finales** Q-F.
- **Paso 2.2.-** Determinar una nueva partición $\Pi_{nueva}$ a partir de $\Pi$ aplicando el siguiente procedimiento a cada grupo G de $\Pi$
  - Dividir G en subgrupos tales que dos estados $q_i$ y $q_j$ de G 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 $\Pi$.
  - Sustituir G por sus grupos en $\Pi_{nueva}$.
- **Paso 2.3.-** Si $\Pi_{nueva} \neq \Pi$, se han creado nuevos subgrupos
  - $\Pi = \Pi_{nueva}$
  - volver al **paso 2.2**
  - en caso contrario, es decir, si ya no salen más grupos
    - $\Pi_{final} = \Pi$
    - ir al **paso 3**



## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

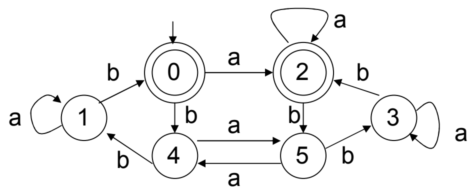
- **Paso 3**.- Se escoge un estado de cada grupo de la partición $\Pi_{final}$ como **representante** para formar el nuevo conjunto Q´.
- **Paso 4.-** Cálculo de δ' en A´. Sea $q_i$ un estado representante, sea un símbolo *a* tal que $\delta(a,q_i)=q_j$ y sea $q_k$ el representante del grupo de $q_j$, entonces $\delta'(a,q_i)=q_k$
- **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.

## 4. Autómatas Finitos
### 4.2. Autómatas finitos deterministas (AFD)

- Ejemplo. Minimizar el AFD $A=(X,Q,\delta, q_0,F)$ donde $X=\{a,b\}$ $Q=\{1,2,3,4,5\}$ $q_0=1$, $F=\{q_5\}$ y $\delta$ viene dada por la siguiente tabla:

| | a | b |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 2 | 4 |
| 3 | 2 | 3 |
| 4 | 2 | 5 |
| 5 | 2 | 3 |

- Ejemplo. Minimizar el AFD $A=(X,Q,\delta, q_0,F)$ donde $X=\{a,b\}$ $Q=\{0,1,2,3,4,5\}$ y el resto de la información viene descrita en el siguiente diagrama de estados



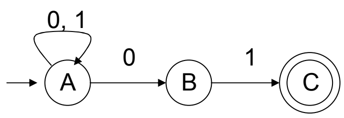



## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Permiten 0, 1 o más transiciones desde un estado con cada símbolo de entrada.
  - Tiene la capacidad de disparar varias transiciones a la vez con el mismo símbolo de entrada y, por tanto, puede estar en varios estados simultáneamente
    - Por ejemplo, si estamos buscando la palabra clave implements en un programa en Java, y la configuración actual es (iResto_Cadena, q) es útil, al llegar el símbolo i, suponer que estamos al inicio de la palabra clave buscada y utilizar una secuencia de estados únicamente para comprobar que efectivamente llega esa palabra.
    - Sólo definimos los estados que necesitamos para aceptar palabras
- Reconocen los mismos lenguajes que los AFD

## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Similares a los AFD con un conjunto finito de estados y símbolos de entrada, un estado inicial, un conjunto de estados finales y una función de transición de estados δ
- Diferencia
  - AFD: δ define para cada posible combinación (x,q) un estado nuevo
  - AFND: δ puede no estar definida para alguna combinación (x,q) y, por el contrario, puede definir para otras combinaciones (x,q) más de un estado
- Ejemplo






## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Un **AFND** es una quíntupla **AFND = (X, Q, $\delta$, $q_0$, F)** con el mismo significado que para un AFD salvo que $\delta$ es en este caso una aplicación no determinista o relación de la forma: $\delta: X \times Q \rightarrow P(Q)$ donde P(Q) es un subconjunto de Q
- Configuración. Descripción instantánea de un AFND
  - (w, q), $q \in Q$ , $w \in X^*$, donde q representa el estado actual y w la cadena que queda por procesar
  - Configuración inicial : (w, $q_0$)
  - Configuración final: ($\lambda$, $q_f$) , $q_f \in F$
- Movimiento Es el tránsito entre dos configuraciones.
  - Se representa: (aw ,q) $\vdash$ (w, q') donde $q' \in \delta(a, q)$
  - Se extiende a $\vdash^*$ como en los AFD

## 4.3. Aut. finitos no deterministas (AFND)

- **Representación de AFND.**
  - **Tabla de Transiciones.** Se diferencian de los AFD en que el contenido de las casillas es un conjunto (incluso el vacío)

| | 0 | 1 |
|---|---|---|
| A | {A,B} | A |
| B | ∅ | C |
| C | ∅ | ∅ |

  - **Diagrama de Transición de estados.**





## 4.3. Aut. finitos no deterministas (AFND)

- **Ejemplos.**
  - Sea el AFND de la página 49. ¿Qué lenguaje representa?
  - Descríbase, mediante configuraciones y movimientos el procesamiento de las palabras 01, 001, 011, 1001

## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Función de transición extendida
  - Se extiende $\delta : X \times Q \rightarrow P(Q)$ a $\delta^* : X^* \times Q \rightarrow P(Q)$ como sigue:

$$\delta^*(\lambda, q) = q$$

$$\delta^*(wx, q) = \bigcup_{q' \in \delta^*(w, q)} \delta(x, q'),\ x \in X \text{ y } w \in X^*$$

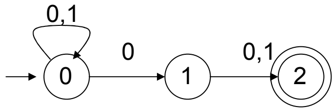
- Palabra aceptada por un AFND
  - Una palabra $a_1 a_2 \ldots a_n$ es aceptada por un AFND si existe una sucesión de transiciones correspondientes a arcos etiquetados $a_1 a_2 \ldots a_n$ que va desde el estado inicial a algún estado final
- Lenguaje aceptado por un AFND
  - Si $M = (X, Q, \delta, q_0, F)$ es un AFND, llamamos **lenguaje aceptado por M**, $L(M)$, al conjunto

$$L(M) = \{ w \in X^* \mid \delta^*(w, q_0) \cap F \neq \emptyset \}$$



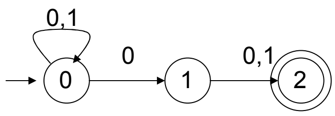
## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Equivalencia entre AFD y AFND
  - Teorema: Si $L \subset X^*$ es aceptado por un AFND, existe un AFD que acepta L.
  - **Ejemplo.** Dado el AFND $M = (X, Q, \delta, q_0, F)$ donde $X = \{0,1\}$, $Q = \{0, 1, 2\}$, $q_0 = 0$, $F = \{2\}$ y $\delta$ dada en el diagrama, trataremos de obtener el AFD $M' = (X, Q', \delta', q_0, F')$ equivalente



## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Algoritmo AFND ⇒ AFD
  - Paso 1. Definición del nuevo conjunto de estados Q’
    - Q' = P(Q) (conjunto de partes de Q)
    - Notación: al conjunto { $q_i$ , $q_j$, ... , $q_n$ } lo denotamos por [ $q_i$ $q_j$ ... $q_n$ ]
    - A menudo, no todos los estados de Q’ serán accesibles desde el estado inicial $q_0$’. En el Paso 2 se determinará cuáles son accesibles
    - Q’={[∅], [0], [1], [2], [0,1], [0,2], [1,2], [0,1,2]}





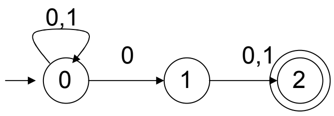

## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Algoritmo AFND ⇒ AFD
  - Paso 2. Definición de la nueva función de transición δ’
    - Para cada conjunto de estados C⊆Q’ y para cada símbolo a∈X

      $$\delta'(a, C) = \bigcup_{p \text{ en } C} \delta(a,p)$$

      - Para cada conjunto de estados que sea accesible desde el $q_0$’, miramos para todos los estados del conjunto a qué estados transita con la entrada a y tomamos la unión de todos ellos
    - **Q’ estará formada por aquellos estados que sean accesibles desde el inicial al definir δ’**
    - Ejemplo: δ'(0, [0]) = [0,1]; δ'(1, [0,1]) = [0,2] .......

## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)




| δ´ | 0 | 1 |
|---|---|---|
| [0] | [0,1] | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |



## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

- Algoritmo AFND ⇒ AFD
  - Paso 3. Definición del nuevo estado inicial $q_0'$
    - $q_0'$ : $[q_0]$
    - Ejemplo
  - Paso 4. Definición del nuevo conjunto de estados finales F´
    - $F' = \{ [q_1 q_2 ... q_n] \in Q' \mid \exists\, i \text{ con } q_i \in F \}$
    - Ejemplo
  - Paso 5. Eliminación de estados inaccesibles
    - Todos aquellos estados de Q’ (ver paso 1) que no hayan hecho falta en el paso 2.

## 4. Autómatas Finitos
### 4.3. Aut. finitos no deterministas (AFND)

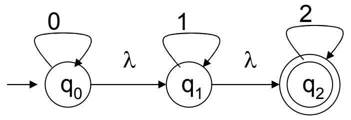
- Ejemplo. Sea el siguiente AFND, calcular el AFD equivalente

| δ | 0 | 1 |
|---|---|---|
| → A | {0, 3} | {0, 1} |
| B | ∅ | 2 |
| C* | 2 | 2 |
| D | 4 | ∅ |
| E* | 4 | 4 |



## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

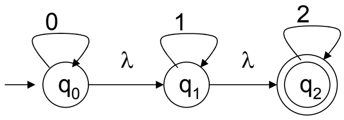
- Extensión de AFND que permite cambios de estado con la entrada vacía (λ), es decir, permite evolucionar de un estado a otro sin consumir ningún símbolo de la cadena de entrada.
- Los autómatas aceptarán las secuencias de etiquetas que pasan por algún camino que lleve desde el estado inicial a algún estado final. Cada λ que se encuentre en el camino es "invisible"
- No expande la clase de lenguajes hasta ahora aceptados por los AF, pero proporciona facilidades para construirlos.



## 4. Autómatas Finitos

### 4.4. AFND con λ-movimientos

- Un AFND con λ-movimientos se define como $M=(X,Q,\delta,q_0,F)$ donde sólo δ difiere de un AFND $\delta:\{X\cup\{\lambda\}\} \times Q \rightarrow P(Q)$
- Ejemplo:




| δ | 0 | 1 | 2 | λ |
|---|---|---|---|---|
| $q_0$ | 0 | ∅ | ∅ | $q_1$ |
| $q_1$ | ∅ | $q_1$ | ∅ | $q_2$ |
| $q_2$ | | | | |

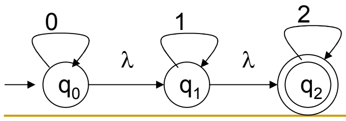
- Configuración de un AFND con λ-movimientos
- Movimiento: Es el tránsito entre dos configuraciones.
  - Se representa: $(aw, q) \vdash (w, q')$ donde $q' \in \delta^*(a, q)$
  - Se extiende a $\vdash^*$ de la forma habitual



## 4. Autómatas Finitos

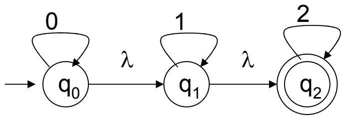
### 4.4. AFND con λ-movimientos

- Ejemplo: Sea el AFND con λ-movimientos siguiente, establezca la configuración inicial y los movimientos que siguen hasta llegar a la configuración final con las palabras
  - 012
  - 02
  - 12
  - 2
  - 00122
  - 00000



## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

- λ-Clausura de un estado, λ-Cl(q): λ-Cl(q): $Q \rightarrow P(Q)$
  - λ -Cl (q) = {q} ∪ {q'∈Q| ∃ camino de q a q' con λ}
  - Es decir:
    1. $q \in \lambda$ -Cl (q)
    2. si $p \in \lambda$ -Cl (q)$\Rightarrow \delta^*(\lambda, p) \in \lambda$ -Cl (q)
- Ejemplo:

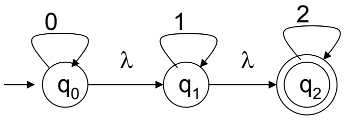



| | λ-Cl |
|---|---|
| $q_0$ | $\{q_0, q_1, q_2\}$ |
| $q_1$ | |
| $q_2$ | |



## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

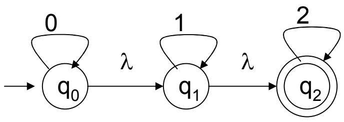
- λ-Clausura de un conjunto de estados, λ-Cl(A):P(Q)→P(Q)
  - λ- Cl(A) = ∪{λ-Cl(q);q∈A}
  - Observaciones.
    - λ-Cl(A) ⊆ λ-Cl(B) si A ⊆ B
    - λ-Cl (λ -Cl (A)) = λ -Cl (A)
- Ejemplo:




| | λ-Cl |
|---|---|
| $\{q_0\}$ | $\{q_0, q_1, q_2\}$ |
| $\{q_0, q_1\}$ | |
| $\{q_0, q_1, q_2\}$ | |

# 4. Autómatas Finitos
## 4.4. AFND con λ-movimientos

- Función de transición extendida
  - Se extiende δ: {X∪{λ}} × Q → P(Q) a $\delta^*: X^* \times Q \rightarrow P(Q)$ donde:
    1. $\delta^*(\lambda, q) = \lambda\text{-Cl}(q)$
    2. $\delta^*(x, q) = \lambda\text{-Cl}\,[\cup \{\delta(x, s)\ ;\ s \in \lambda\text{-Cl}(q)\}]$
    3. $\delta^*(wx, q) = \lambda\text{-Cl}\,[\cup\{\delta(x, r) \mid r \in \delta^*(w, q)\}]$




| | |
|---|---|
| $\delta^*(0, q_0)$ | $\{q_0, q_1, q_2\}$ |
| $\delta^*(02, q_1)$ | |
| $\delta^*(0011, q_0)$ | |



# 4. Autómatas Finitos
## 4.4. AFND con λ-movimientos

- Palabra aceptada por un AFND con λ-movimientos
  - $\delta^*(w, q_0) \cap F \neq \emptyset$
  - Una palabra $a_1\ a_2 \ldots a_n$ es aceptada por un AFND si existe algún camino etiquetado con los símbolos de la palabra que, partiendo del estado inicial lleve a algún estado final. Pueden aparecer en el camino arcos etiquetados con λ
- Lenguaje aceptado por un AFND con λ-movimientos $M=(X, Q, \delta, q_0, F)$
  - $L(M) = \{ w \in X^* \mid (w, q_0) \vdash^* (\lambda, q_f), q_f \in F\}$
  - O también $L(M) = \{ w \in X^* \mid \delta^*(w, q_0) \cap F \neq \emptyset \}$.

## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

- Ejercicios. Definir un AFND con λ-movimientos que reconozca el lenguaje L={palabras formadas por símbolos del alfabeto ASCII que representen un número real, con signo y sin exponente en JAVA}
  - 2, 2.3, -3, -3.4,.........

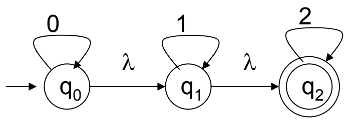


## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

- Equivalencia entre AFND con y sin λ-movimientos
  - Teorema. Si $L \subset X^*$ es aceptado por un AFND con λ-movimientos, entonces L es aceptado por un AFND sin λ-movimientos y, por tanto, **L es un lenguaje regular**.
- Algoritmo AFND con λ-movimientos ⇒ sin λ-movimientos
  - Dado un AFND con λ-movimientos $M=(X, Q, \delta, q_0, F)$, el nuevo autómata lo definimos como: $M'=(X, Q, \delta', q_0, F')$
  - Paso 1. Definición de δ': $\delta': X \times Q \to P(Q)$
    - $\delta'(x, q) = \delta^*(x, q)$
      1. Calculamos $P1 = \lambda\text{-Cl}(q)$
      2. Calculamos $P2 = [\cup\{\delta(x, q_i) \mid q_i \in P1]$
      3. Calculamos $P3 = \lambda\text{-Cl}(P2)$

## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

- Algoritmo AFND con λ-movimientos ⇒ sin λ-movimientos
  - Paso 1. Definición de δ': δ': X × Q → P(Q)




| δ´ | 0 | 1 | 2 |
|---|---|---|---|
| $q_0$ | | | |
| $q_1$ | | | |
| $q_2$ | | | |

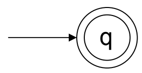
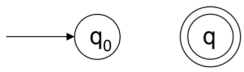
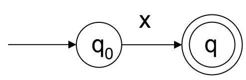


## 4. Autómatas Finitos
### 4.4. AFND con λ-movimientos

- Algoritmo AFND con λ-movimientos ⇒ sin λ-movimientos
  - Paso 2. Definición de F'
    - Si λ -Cl ($q_0$) ∩ F ≠∅ entonces F' = {$q_0$} ∪ F
    - Si λ -Cl ($q_0$) ∩ F = ∅ entonces F' = F
  - Calcular F' y dibujar el diagrama de transición de estados de M'

## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Teorema. Si r es una expresión regular, entonces existe un AFND con $\lambda$-movimientos y con a lo sumo un estado final, del que no sale ninguna transición, que acepta L(r).
- Demostración (por inducción en el nº de operadores de la e.r)
  - **Paso básico**. Con 0 operadores.

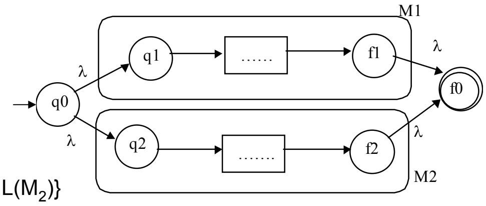
| $r = \lambda$ | $r = \varnothing$ | $r = x$ con $x \in X$ |
|---|---|---|
|  |  |  |





## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Demostración
  - **Paso de inducción**. Con 1 o más operadores
    - Sean $r_1$ y $r_2$ dos e.r., por tanto para $r_1$ existe $M_1 = (Q_1, X_1, \delta_1, q_1, F_1)$ tal que $L(M_1) = L(r_1)$ y para $r_2$ existe $M_2 = (Q_2, X_2, \delta_2, q_2, F_2)$ tal que $L(M_2) = L(r_2)$
    - Supongamos $Q_1 \cap Q_2 = \varnothing$ (renombrar los estados) y construimos M.
    - **Caso A: $r = (r_1 + r_2)$**
      - **$M = (Q_1 \cup Q_2 \cup \{q_0, f_0\}, X_1 \cup X_2, \delta, q_0, \{f_0\})$, con $\delta$:**

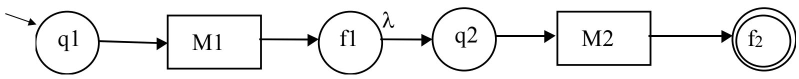




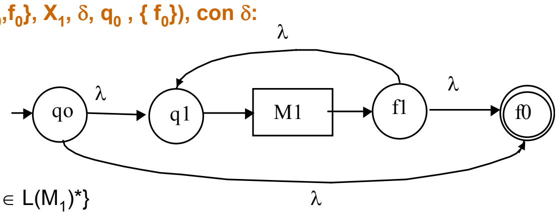
      - $L(M) = \{ w \mid w \in L(M_1)$ o $w \in L(M_2)\}$

## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Caso B: $r = (r_1 . r_2)$
  - **M = ( $Q_1$ U $Q_2$, $X_1$ U $X_2$, δ, $q_1$, { $f_2$}) , con δ:**




  - $L(M) = \{ w_1w_2 / w_1 \in L(M_1) , w_2 \in L(M_2)\}$
- Caso C: $r = (r_1)^*$
  - **M = ( $Q_1$ U {$q_0$,$f_0$}, $X_1$, δ, $q_0$ , { $f_0$}), con δ:**

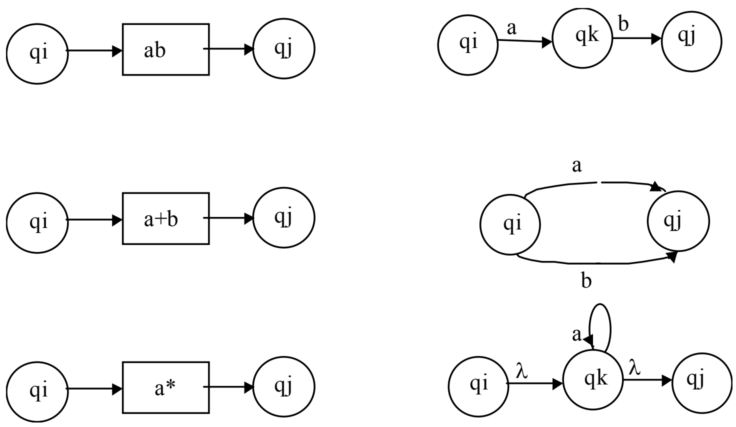



  - $L(M) = \{ w / w \in L(M_1)^*\}$



## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Reglas de desarrollo



## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Ejercicios
  - $r = ( 0 + 1\ 0^{*}\ 1)\ \ 1\ ( 0\ 1 )^{*}$
  - $r = ( a + b )^{*}\ ( aa + bb )\ (a + b )^{*}$

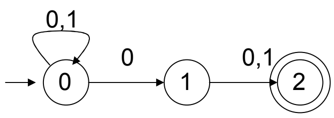


## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- **Teorema**: Si $L \subset X^{*}$ es un lenguaje aceptado por un AFD, entonces L se puede describir por una expresión regular.
- **Corolario**: Sea X un conjunto finito, y $L \subset X^{*}$. Son equivalentes las siguientes afirmaciones:
  1. L es un lenguaje regular
  2. L es un lenguaje aceptado por algún AFD
  3. L es un lenguaje aceptado por algún AFN sin $\lambda$-mov.
  4. L es un lenguaje aceptado por algún AFN con $\lambda$-mov.
  5. L se puede describir por una expresión regular

## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

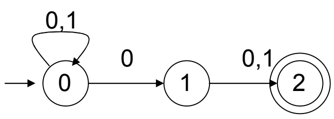
- Se trata de, dado un AFD, encontrar una expresión regular de tal forma que describan el mismo lenguaje.
- Dado un AFD se denota por $L_q$ al lenguaje reconocido por el AFD cuando se considera al estado q como estado inicial.
- Se denota por $l_q$ a la e.r. que denota el lenguaje $L_q$, por tanto, $L_q = L(l_q)$
- Ejemplo
  - $l_q$=
  - $L_q$= {palabras formadas por 0, 1 que …..}






## 4. Autómatas Finitos
### 4.5. Equivalencia entre AF y e.r.

- Ecuación característica
  - $m(q, q') = \sum_{q' \in \delta(x,q)} x$




- $m(q_0, q_0)$= 0+1
- $m(q_0, q_1)$= 0 $\quad$ $m(q_0,q_2)= \varnothing$
- $m(q_1,q_0)$= $\quad$ $m(q_1,q_1)$= $\quad$ $m(q_1,q_2)$=
- $m(q_2,q_0)$= $\quad$ $m(q_2, q_1)$= $\quad$ $m(q_2,q_2)$=

  - $l_q = \sum_{q' \in Q} m(q,q')\cdot l_{q'} + t_q$ donde $t_q = \phi$ si $q \notin F$ ó $t_q = \lambda$ si $q \in F$
    - $lq_0= (0+1).lq_0+ 0.lq_1+ \varnothing.lq_2$
    - $lq_1$=
    - $lq_2$=

# 4. Autómatas Finitos

## 4.5. Equivalencia entre AF y e.r.

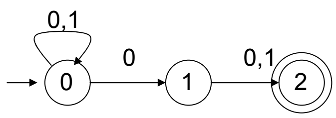
- Se trata de, dado un AFD encontrar una expresión regular de tal forma que describan el mismo lenguaje.
- Regla de Arden
  - Sean R, S, y T tres expresiones regulares tal que $\lambda \notin S$, entonces
    - $R = SR + T \Leftrightarrow R = S^*T$
    - $R = RS + T \Leftrightarrow R = TS^*$



# 4. Autómatas Finitos

## 4.5. Equivalencia entre AF y e.r.

- Algoritmo AFD $\Rightarrow$ e.r.
  - **Paso 1.-** Obtenemos las ecuaciones características del AF calculando $l_q$ para todo $q \in Q$.
    - $lq_0 = (0+1).lq_0 + 0.lq_1 + \varnothing.lq_2$
    - $lq_1 =$
    - $lq_2 =$




  - **Paso 2.-** Despejar $l_q$ aplicando las propiedades de las e.r. (principalmente regla de Arden y distributiva)
    - $lq_0 = (0+1).lq_0 + 0lq_1$ Aplicando la regla de Arden: $lq_0 = (0+1)^*0lq_1$
    - Idem $lq_1$ y $lq_2$
  - **Paso 3.-** Si $q_0$ es el estado inicial $l_{q0}$ es la e.r. que denota aquellas cadenas que partiendo de $q_0$ llegan a un estado final y por tanto la e.r. que denota el lenguaje reconocido por el AFD.

# 5. Propiedades de los lenguajes regulares
## 5.1. Lema de Pumping

- Cuestión pendiente: Dado un lenguaje L, ¿es regular?.
  - El Lema de Pumping se puede utilizar para **demostrar que un lenguaje L no es regular**.
- **Lema de Pumping:** 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 la subcadena $v$ es no vacía, $|xv| \leq n$ y $xv^iy \in L$ , $\forall\, i \geq 0$.
- Demostración:
  - Sea $w = x_1 x_2 \ldots x_m$ ( $|w|= m$ ). En el proceso de aceptación de $w$ por el autómata M, se recorren una sucesión de estados de M: $s_0, s_1, \ldots s_m$, donde $s_i = \delta^*(x_1 x_2 \ldots x_i , q_0 )$ será el estado en que nos encontramos después de haber leído los primeros $i$ símbolos de $w$.



# 5. Propiedades de los lenguajes regulares
## 5.1. Lema de Pumping

- **Lema de Pumping. Demostración**
  - M sólo tiene $n$ estados, estamos pasando por $m+1>n$ estados. Por lo que dos de los $s_i$ deben ser el mismo estado:
    - $s_i = \delta^*(x_1 x_2 \ldots x_i , q_0 ) = \delta^*(x_1 x_2 \ldots x_i \ldots x_j , q_0 ) = s_j$
  - Dicho de otra forma, el trayecto que $w$ nos obliga a hacer a través de M contiene una cadena cerrada: $s_i =\delta^*(x_{i+1} x_{i+2} \ldots x_j , s_i )$
  - Pongamos $x = x_1 x_2 \ldots x_i$ $v = x_{i+1} x_{i+2} \ldots x_j$ $y = x_{j+1} x_{j+2} \ldots x_m$
  - Está claro entonces que $w = xvy$, y $v$ es no vacía (aunque $x$ e $y$ pudieran serlo).
  - Si cogemos $s_i$ como el primer estado que se repite entonces $|xv|\leq n$
  - Además, $\delta^*(xy, q_0 ) = \delta^*(xvy, q_0 )= \delta^*(xvvy, q_0 )= \ldots =\delta^*(xv^iy, q_0 )$
  - Como $\delta^*(xv^iy, q_0) = \delta^*(w, q_0 ) = s_m$ es final, cualquier $xv^iy$ pertenece a $L$, que es lo que se quería demostrar.

## 5. Propiedades de los lenguajes regulares
### 5.1. Lema de Pumping

- **Teorema:** $L = 0^k 1^k$ no es regular.
- Demostración:
  - Si $L$ fuese regular, sería aceptado por un AFD de $n$ estados. Esto, en combinación con la propiedad garantizada por el lema de pumping, conduce a una contradicción.
  - $w = 0^n 1^n \in L$ de longitud $2n$. Le aplico 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



## 5. Propiedades de los lenguajes regulares
### 5.1. Lema de Pumping

- **Lema de Pumping. Ejemplo**
  - Sea el lenguaje L que consta de todas aquellas cadenas de paréntesis balanceados, demostrar que no es regular
  - Demostrar que no son regulares los siguientes lenguajes
    - $\{0^n 1^m 2^n$ | siendo n, m enteros arbitrarios$\}$
    - $\{0^n 1 2^{2n}$ | siendo n, m enteros arbitrarios$\}$
    - {cadenas formadas por 0 y 1 de la forma ww, es decir una subcadena repetida}
    - {cadenas formadas por 0 y 1 de la forma a$\bar{a}$, es decir una subcadena reflejada}

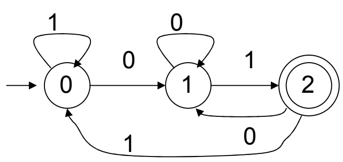
## 5. Propiedades de los lenguajes regulares
### 5.2. Propiedades de clausura de los l.r.

- Las propiedades de clausura expresan la idea de que, cuando uno o varios lenguajes son regulares, otros relacionados con ellos también lo son.
- **Los lenguajes** regulares son cerrados para la unión, concatenación y clausura (operador *).
  - Es inmediato por la definición de e.r.
- Si $L \subset X^*$ es un LR, entonces su complementario $\overline{L} = X^* - L$ también lo es.
- Demostración
  - Sea L=L(A) para un AFD A=(X, Q, $\delta$, $q_0$, F)
  - Se define B como el AFD (X, Q, $\delta$, $q_0$, Q-F)
  - $w \in L(B)$ sii $\delta^*(w, q_0) \in Q-F$, lo que significa que $w \notin L(A)$
  - Por tanto $\overline{L} = L(B)$



## 5. Propiedades de los lenguajes regulares
### 5.2. Propiedades de clausura de los l.r.

- Ejemplo. Sea el siguiente AFD
  - ¿ Quién es L(A)?
  - Buscar una e.r.




  - Encontrar Ā, AFD que reconozca el complementario de L(A)
  - Y una e.r. equivalente a Ā

## 5. Propiedades de los lenguajes regulares
### 5.2. Propiedades de clausura de los l.r.

- Si L y M son lenguajes regulares, también lo es $L \cap M$

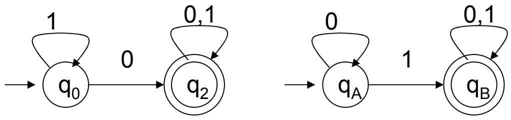
$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$

- Construcción del AFD
  - Sea $A_L = (X_L, Q_L, \delta_L, q_L, F_L)$ y $A_M = (X_M, Q_M, \delta_M, q_M, F_M)$ AFD tal que $L = L(A_L)$ y $M = L(A_M)$
  - AFD A que simule el comportamiento de ambos autómatas
    - $A = (X, Q_L x Q_M, \delta, (q_L, q_M), F_L x F_M)$
    - Los estados de A son pares de estados, el 1º de $A_L$ y el 2º de $A_M$
    - Estado inicial $(q_L, q_M)$
    - Estados finales $F_L x F_M$
    - Transiciones en A: Si A está en el estado (p,q) y a es el símbolo a la entrada, suponiendo que $\delta_L(a,p)=r$ y $\delta_M(a,q)=s$, $\delta(a, (p,q))=(r,s)$
      - $\delta(a, (p,q)) = (\delta_L(a,p), \delta_M(a,q))$



## 5. Propiedades de los lenguajes regulares
### 5.2. Propiedades de clausura de los l.r.

- Sean los siguientes AFD




  - ¿ Qué lenguajes aceptan? Definir e.r. para cada uno de ellos
  - Definir un AFD que acepte la intersección de los lenguajes reconocidos por ambos autómatas.

# 6. Propiedades de los lenguajes regulares
## 6.1. Aplicaciones de los A.F

- La aplicación más inmediata de los AF es la construcción de analizadores léxicos.
- La tarea del analizador léxico es la de leer carácter a carácter el fichero de entrada y reconoce las unidades sintácticas
  - subcadenas de caracteres consecutivos que forman una agrupación lógica con significado léxico para el lenguaje
  - palabra reservada, identificador, número, etc.
  - Ejemplo
    - Subcadena — unidad sintáctica

| Subcadena | unidad sintáctica |
|---|---|
| if | palabra reservada if |
| [A-Za-z][A-Za-z0-9]* | identificador |

- El analizador léxico se construirá como un AF, que habitualmente será un AFN con λ-movimientos de la siguiente forma:



# 7. Aplicaciones de los Autómatas Finitos