Lenguajes regulares

Curso 2018-2019

Senén Barro Ameneiro, CiTIUS

@SenenBarro

Material elaborado fundamentalmente por
el profesor Manuel Mucientes Molina

Bibliografía

- J.E. Hopcroft, R. Motwani y J.D. Ullma, "Teoría de Autómatas, Lenguajes y Computación", Addison Wesley, 2008.
  - Capítulos 3 y 4

- P. Linz, ''An Introduction to Formal Languages and Automata", Jones and Bartlett Publishers, Inc., 2001.
  - Capítulo 3 y 4

Expresiones regulares (ER)

- Descripción algebraica de los lenguajes regulares
- Forma declarativa de expresar las cadenas que queremos aceptar
- Se usan como lenguaje de entrada en muchos sistemas de proceso de cadenas
  - Especificación de caracteres en el comando *grep* de UNIX
  - Diseño de analizadores léxicos mediante *Lex* o *Flex*
    - Acepta expresiones regulares (formas de las unidades sintácticas) y produce un AFD que reconoce la unidad sintáctica que aparece a continuación en la entrada
  - Diseño de verificadores del formato del texto en formularios web (JavaScript, Perl, etc.)

Operadores de las ER

- Las expresiones regulares denotan lenguajes: $01^* + 10^*$
- Unión de dos lenguajes *L* y *M*, $L \cup M$: conjunto de cadenas que pertenecen a *L*, a *M* o a ambos
- Concatenación de dos lenguajes *L* y *M*, *L*.*M* (o *LM*): conjunto de cadenas formadas por la concatenación de una cadena de *L* y otra de *M*
- Clausura, estrella o clausura de Kleene de un lenguaje *L*, L^*: conjunto de cadenas formado por la concatenación de cualquier número de cadenas de *L* (se admiten repeticiones). Formalmente:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

- Sólo existen dos lenguajes con clausura no infinita:
  - L = {∅}. L^* = {**ε**}, ya que $\varnothing^0$ = {**ε**} y $\varnothing^i$ = ∅ para i > 0
  - L = {**ε**}. L^* = {**ε**}

Precedencia de los operadores

- El operador * tiene la mayor precedencia
- El operador de concatenación (.) es el segundo en orden de precedencia
- Finalmente se aplican los operadores de unión con sus operandos
- Utilizando los paréntesis se modifican las reglas de precedencia

Álgebra de ER

- Propiedad conmutativa de la unión: $L + M = M + L$
- Propiedad asociativa de la unión: $(L + M) + N = L + (M + N)$
- Propiedad asociativa de la concatenación: $(LM)N = L(MN)$
  - la concatenación no es conmutativa: $LM \neq ML$
- $\emptyset$ es el elemento identidad de la unión: $\emptyset + L = L + \emptyset = L$
- ε es el elemento identidad de la concatenación: $\varepsilon L = L\varepsilon = L$
- $\emptyset$ es el elemento nulo de la concatenación: $\emptyset L = L\emptyset = \emptyset$
- Propiedad distributiva por la izquierda de la concatenación respecto de la unión: $L(M + N) = LM + LN$
- Propiedad distributiva por la derecha de la concatenación respecto de la unión: $(M + N)L = ML + NL$

Álgebra de ER

- Operador idempotente: el resultado de aplicarlo a dos valores iguales es el mismo valor
  - Propiedad de idempotencia de la unión: $L + L = L$
- Propiedades relativas a las clausuras
  - $(L^*)^* = L^*$
  - $\emptyset^* = \boldsymbol{\varepsilon}$
  - $\boldsymbol{\varepsilon}^* = \boldsymbol{\varepsilon}$
  - $L^+ = LL^* = L^*L$
  - $L^* = L^+ + \boldsymbol{\varepsilon}$
  - $L? = L + \boldsymbol{\varepsilon}$

Construcción de ER

- Base:
  - Las constantes ε y $\varnothing$ son ER. L(ε) = {ε} y L($\varnothing$) = {$\varnothing$}
  - Si *a* es un símbolo, **a** es la ER del lenguaje L(**a**) = {*a*}
- Paso inductivo:
  - si *E* y *F* son ER, *E* + *F* es una ER, y $L(E+F) = L(E) \cup L(F)$
  - si *E* y *F* son ER, *EF* es una ER, y *L*(*EF*) = *L*(*E*)*L*(*F*)
  - si *E* es ER, E^* es una ER, y $L(E^*) = (L(E))^*$
  - si *E* es ER, (*E*) es una ER, y *L*((*E*)) = *L*(*E*)
- Ejemplo: escribir la ER para el conjunto de cadenas que constan de ceros y unos alternos

Problemas

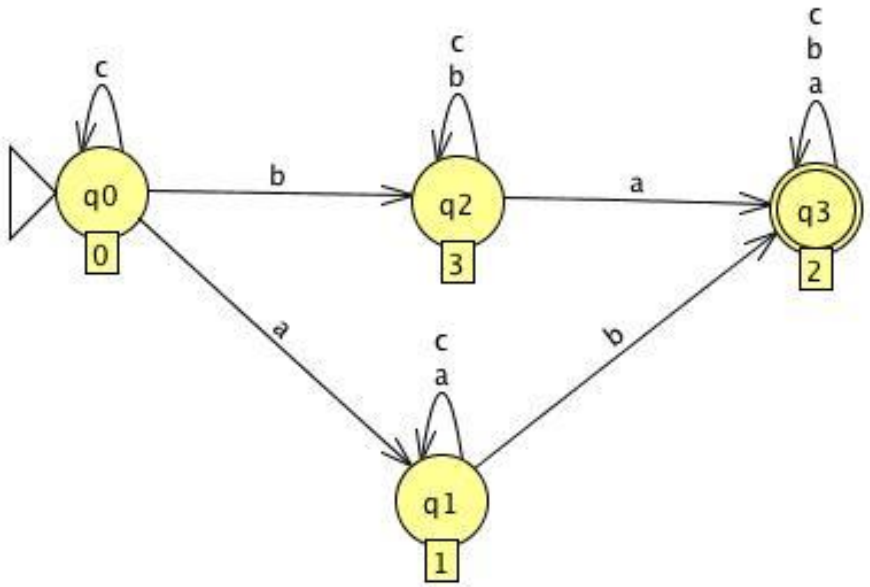
Escribir las ER para los siguientes lenguajes:

1. El conjunto de cadenas del alfabeto {a, b, c} que contengan al menos una a y una b.
2. El conjunto de cadenas del alfabeto {0, 1} cuyo cuarto símbolo empezando por la izquierda sea un 1.
3. El conjunto de cadenas del alfabeto {0, 1}, tales que todos los pares de ceros adyacentes aparecen antes que cualquier par de unos adyacentes.

Problemas

El conjunto de cadenas del alfabeto {a, b, c} que contengan al menos una a y una b.

Solución:



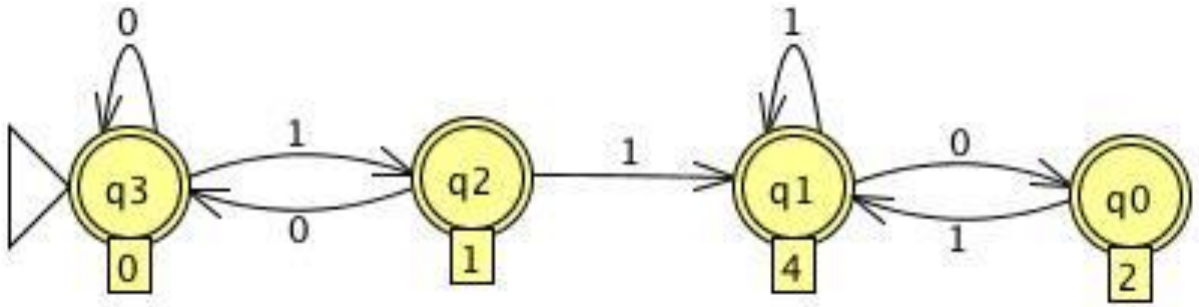
c*a(c+a)*b(a+b+c)* + c*b(c+b)*a(a+b+c)* o, mejor:

c*(b(c+b)*a + a(c+a)*b)(a+b+c)*

Problemas

Escribir las ER para el conjunto de cadenas del alfabeto {0, 1}, tales que todos los pares de ceros adyacentes aparecen antes que cualquier par de unos adyacentes.

Solución, diseñando primero el AFD:



(0+10)*+(0+10)*1+(0+I0)*II(I+01)*+(0+I0)*II(I+0I)*0

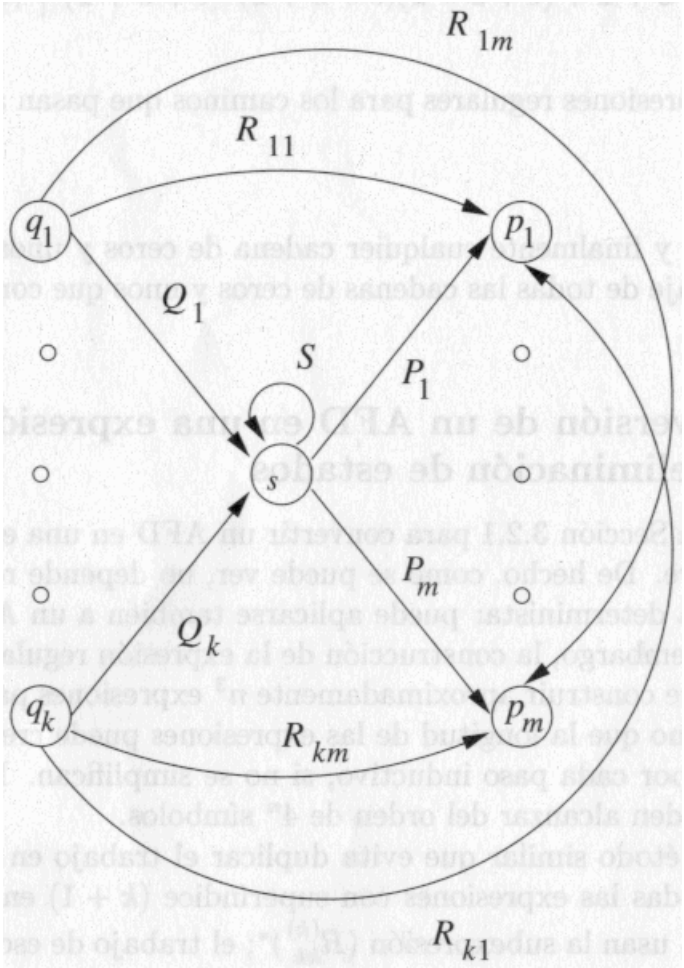
Autómatas finitos y ER

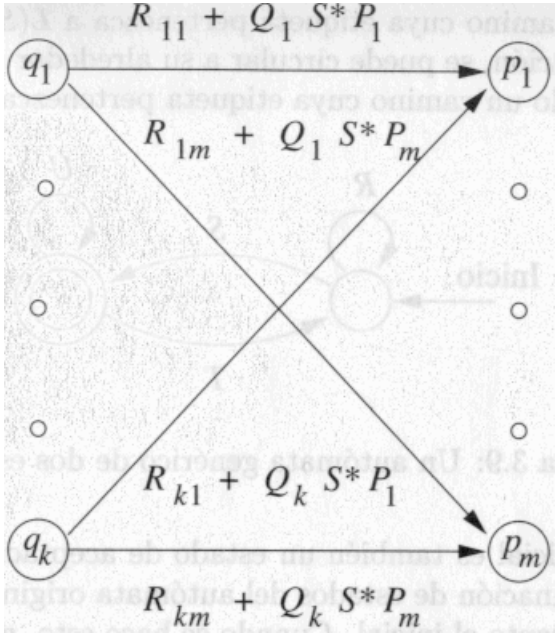
Las ER definen los lenguajes regulares exactamente igual que los autómatas finitos:

- Todo lenguaje definido por un AF (AFD, AFN, AFN- ε) también puede definirse mediante una ER
- Todo lenguaje definido por una ER puede definirse mediante un AF

Autómatas finitos a ER

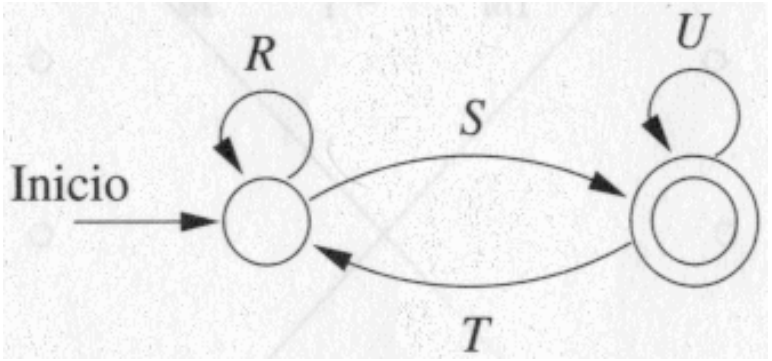
Eliminación de estados: en los arcos aparecen ER



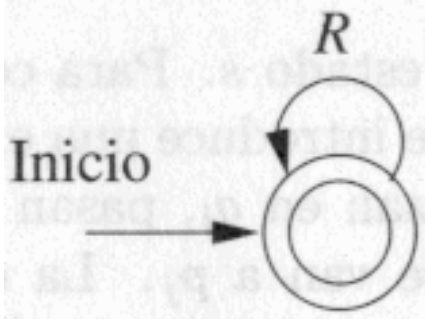


Autómatas finitos a ER

- Para cada estado de aceptación *q*, se aplica el proceso de reducción. Se eliminan todos los estados excepto *q* y el estado inicial q_0
- Si $q \neq q_0$, $L = (R^* + SU^*T)^*SU^*$

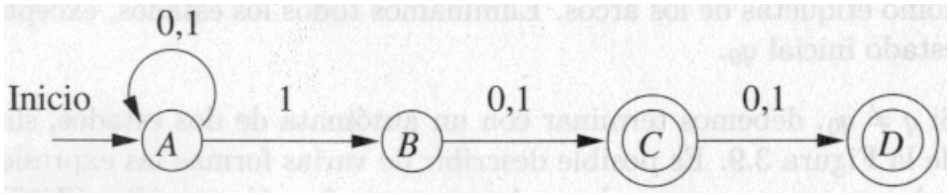


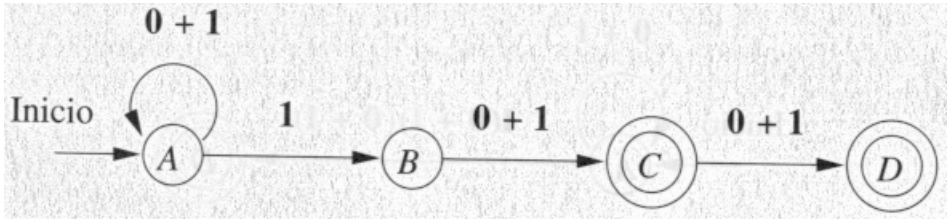
- Si el estado inicial es estado de aceptación, habrá que eliminar todos los estados excepto el inicial: $L = R^*$
- La ER deseada es la unión de las cadenas obtenidas del autómata para cada estado de aceptación

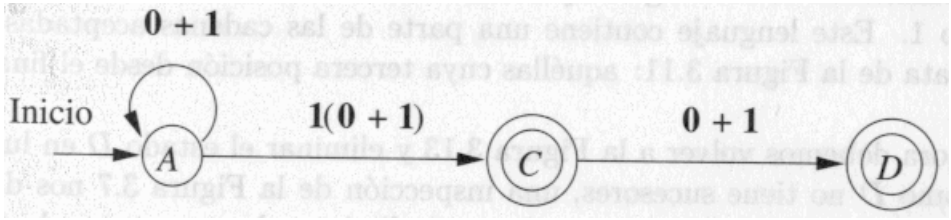


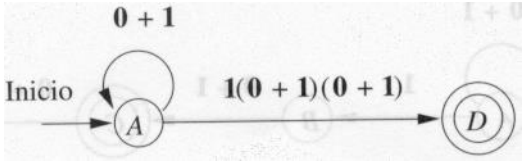
Autómatas finitos a ER

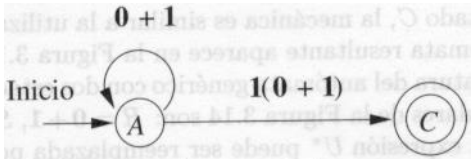
Ejemplo:











Problemas

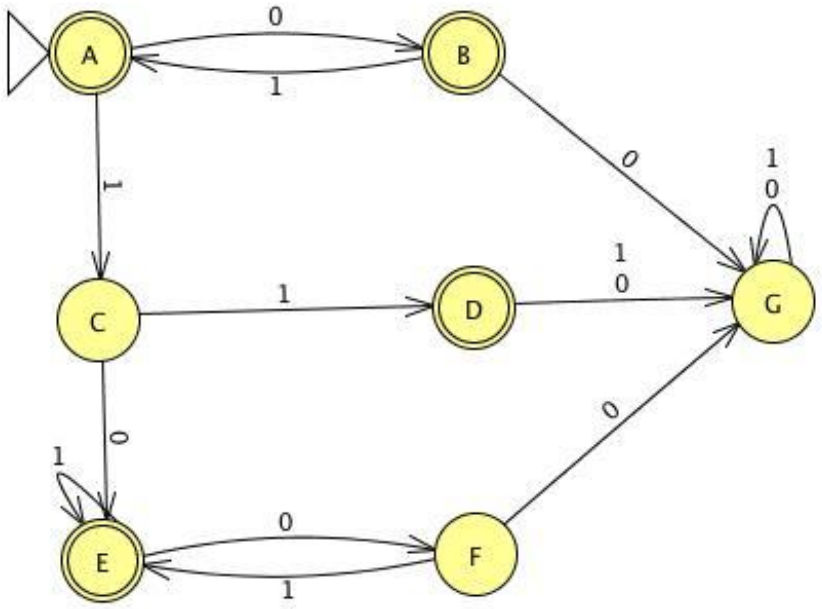
Obtener la expresión regular del AFD definido en la siguiente tabla de transiciones:

	0	1
->*A	B	C
*B	G	A
C	E	D
*D	G	G
*E	F	E
F	G	E
G	G	G

Problemas

Primero obtendremos el AFD asociado y de ahí es fácil obtener la ER

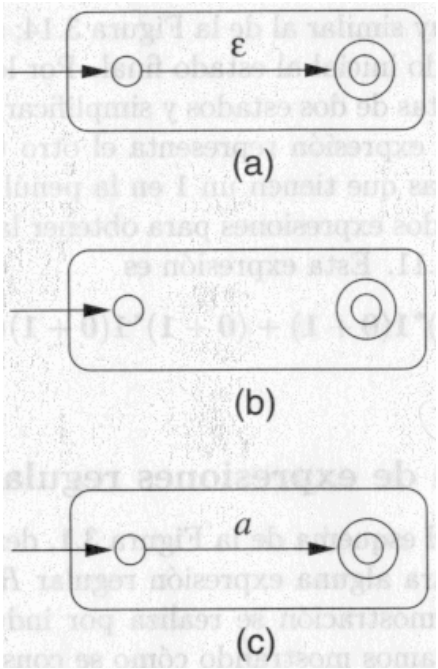
Solución:



$$(01)^* + (01)^*0 + (01)^*11 + (01)^*10(1+01)^*$$

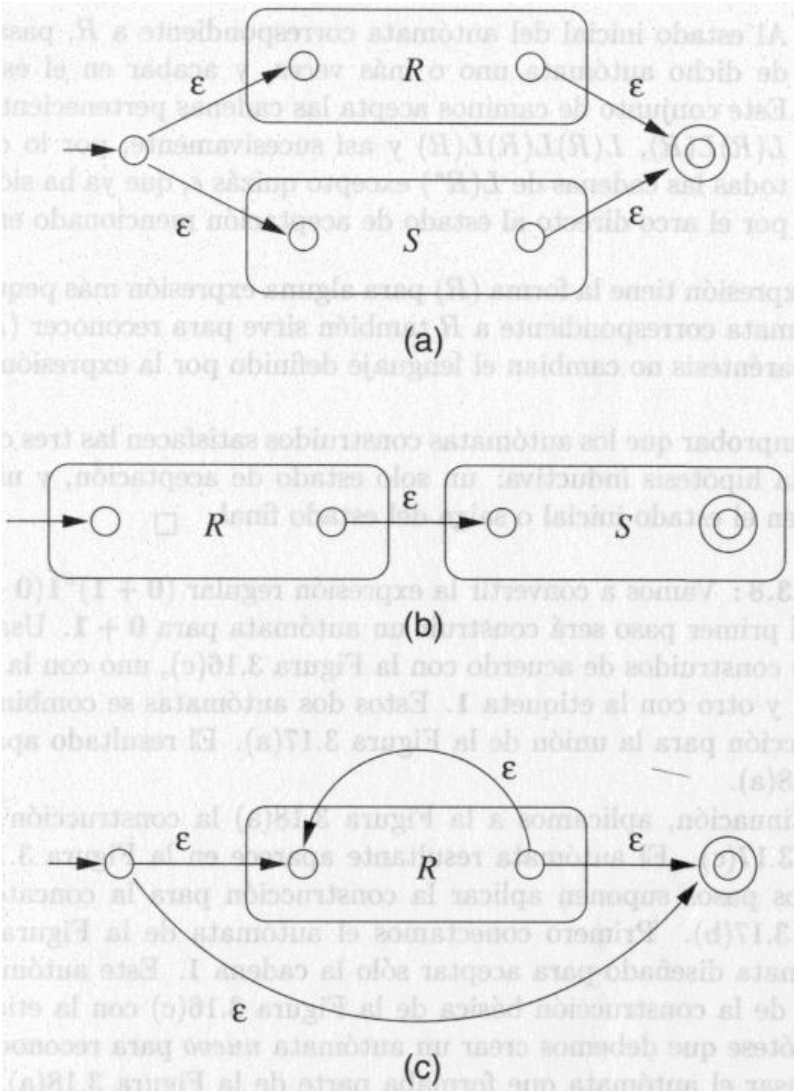
Conversión de ER en autómatas

- Teorema: todo lenguaje definido por una ER puede definirse también mediante un AF
- Prueba: sea $L = L(R)$ para la ER R. Se demostrará que $L = L(E)$ para algún AFN- ε E
- Base:



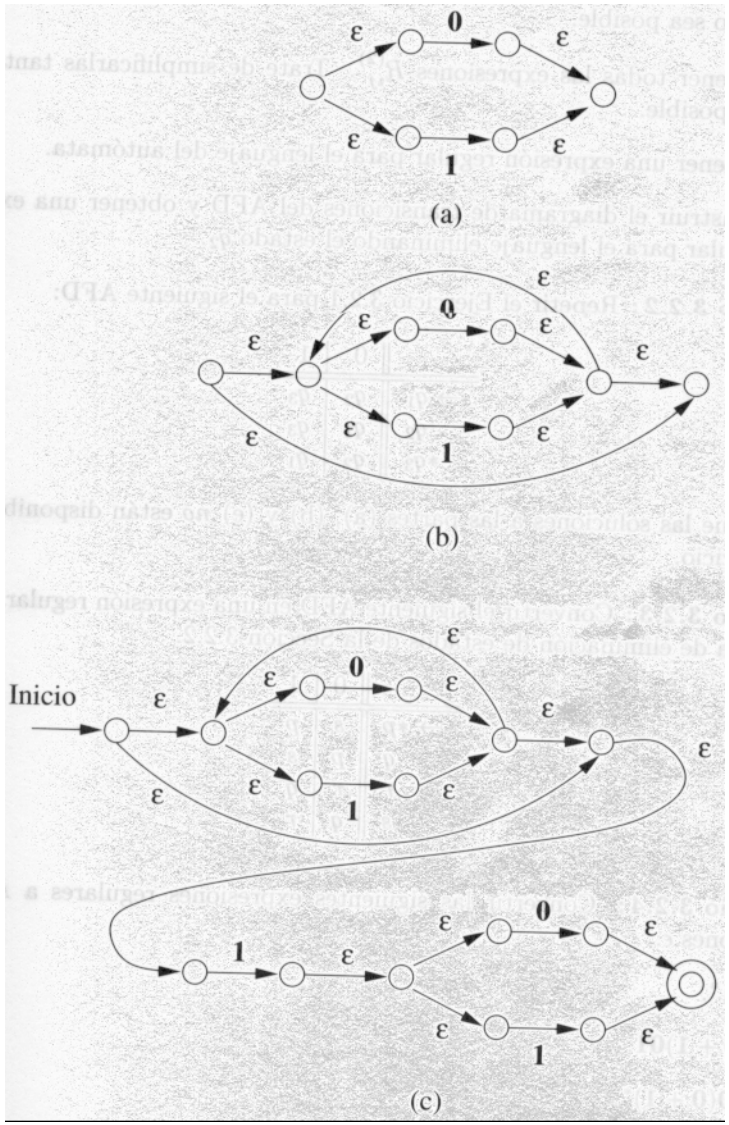
Conversión de ER en autómatas

- Paso inductivo:
  - $R + S$: $L(R) \cup L(S)$
  - RS: $L(R)L(S)$
  - R^*: $L(R^*)$



Conversión de ER en autómatas

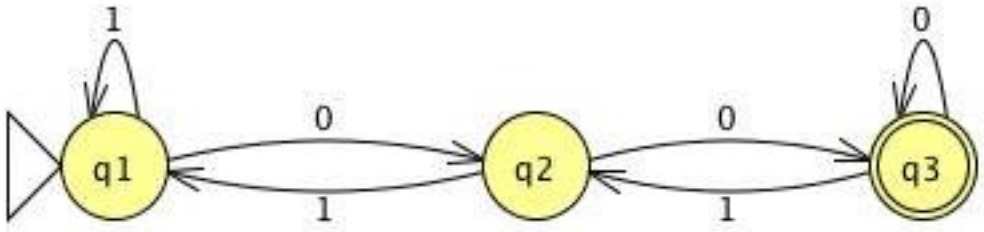
Ejemplo: $(0 + 1)^*1(0 + 1)$



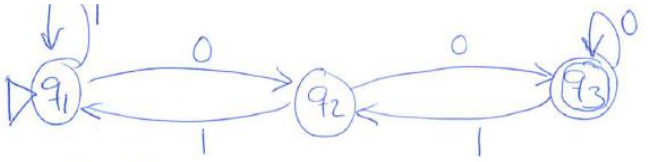
Problemas

1. Dado el siguiente AFD, obtener la ER para el lenguaje del autómata

	0	1
->q_1	q_2	q_1
q_2	q_3	q_1
*q_3	q_3	q_2



1. Convertir las siguientes ER en AFN
   - 01^*
   - 0^*1^*



Eliminamos q_2

Hacia q_2	Desde q_2
q_1	q_1
q_3	q_3

Escribimos las transiciones → $\overline{q_1q_1}$, $\overline{q_1q_3}$, $\overline{q_3q_1}$, $\overline{q_3q_3}$

$\overline{q_1q_1} = 1 + 0\phi^*1 = 1+01$

ϕ indica que en q_2 no hay realimentación a sí misma

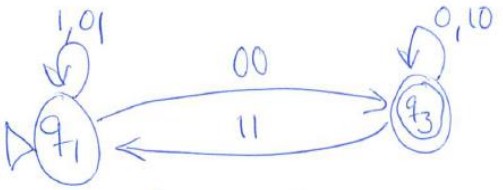
$\overline{q_1q_3} = \phi + 0\phi^*0 = 00$

ϕ no hay transición directa entre q_1 y q_3

$\overline{q_3q_1} = \phi + 1\phi^*1 = 11$

$\overline{q_3q_3} = 0 + 1\phi^*0 = 0+10$

Luego:



La expresión regular será:

$$((1+01)+00(0+10)^*11)^*00(0+10)^*$$

Recordemos:

: $(R+SU^*T)SU^*$

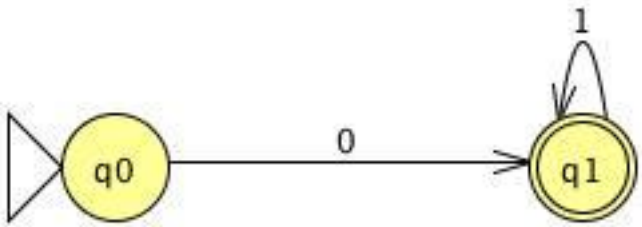
Problemas

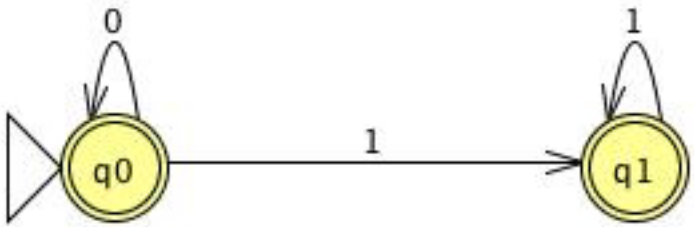
Solución al primer ejercicio:

$$((1+01)+00(0+10)^*11)^*00(0+10)^*$$

Solución a la conversión de las siguientes ER en AFN

- 01^*
- 0^*1^*





Aplicaciones de las ER

- Búsqueda de patrones de texto mediante ER que dan una “imagen” del patrón que se quiere reconocer
- Aplicaciones:
  - analizadores léxicos
  - búsqueda de textos
  - Software de verificación del formato del texto en formularios web (asignatura de Desarrollo de Aplicaciones Web, 3er curso, 1er cuatrimestre)

Búsqueda de patrones en textos

- Búsqueda eficiente de palabras en un gran repositorio de texto, como la Web
- La notación de las ER es valiosa para describir patrones de búsqueda interesantes
- Posibilidad de pasar de ER a una implementación eficiente (autómatas)
- Descripción de patrones de texto vagamente definidos: resulta útil emplear ER, ya que es posible modificarlas con poco esfuerzo
- Ejemplo: direcciones de calles en páginas web
  - (Calle|c/|Avenida|Avda\.|Plaza|Pza\.) [A-Z][a-z]*([A-Z][a-z]*)* [0-9]$^{+}$[A-Z]?

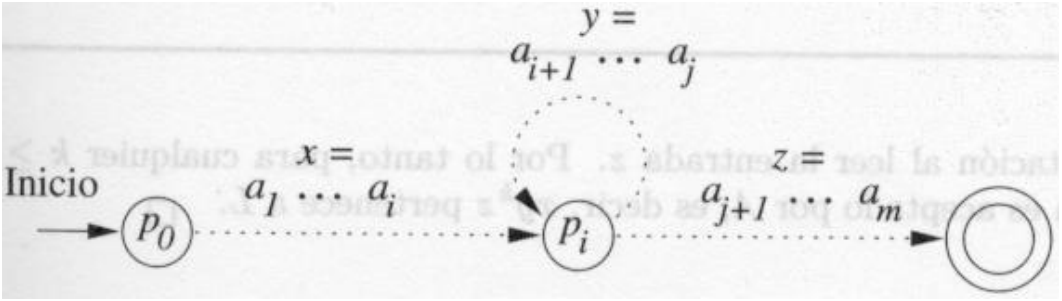
Propiedades de los lenguajes regulares

- Descripción de los lenguajes regulares
  - AFD
  - AFN
  - AFN- ε
  - Expresiones regulares

- No todos los lenguajes son regulares
  - $L_{01} = \{0^n1^n \mid n \geq 1\}$
    - si fuese regular existiría un AF con *k* estados que lo reconocería

Lema del bombeo para lenguajes regulares

- Para un lenguaje regular [infinito], el cumplimiento del lema de bombeo (LB) es una condición **necesaria**, pero **no suficiente**
- Teorema: sea *L* un lenguaje regular. Entonces existe una constante *n* (que depende de *L*), tal que, para toda cadena *w* perteneciente a *L*, con |*w*| ≥ *n*, podemos dividir *w* en tres cadenas, *w* = *xyz*, de modo que:
  - $y \neq \varepsilon$
  - $|xy| \leq n$
  - para todo k≥0, la cadena $xy^k z$ también pertenece a *L*
- Siempre es posible encontrar una cadena no vacía *y*, no demasiado lejos del comienzo de *w*, que se puede "bombear"
- Si se repite *y* cualquier número de veces o se borra (k = 0), la cadena resultante también pertenece al lenguaje *L*



Aplicación del lema del bombeo

1. Elegimos un lenguaje *L* para el que tratamos de demostrar que no es regular
2. El valor de *n* es desconocido, por lo que debemos considerar cualquier posible valor
3. Elegimos *w* (podemos usar *n* como parámetro)
   - Si para un *n* suficientemente grande no podemos escoger w, *L* será regular
4. Repetir para todas las descomposiciones
   1. Escoger una descomposición de *w* en *xyz*, sujeta a las restricciones:
      1. $y \neq \varepsilon$
      2. $|xy| \leq n$
   2. Si xy^kz pertenece a *L* para todo valor de *k*
      1. Se verifica el LB
      2. No se puede afirmar que el lenguaje sea regular
      3. No es necesario probar con otras descomposiciones (finaliza el algoritmo)
5. Si 4.2 no se ha cumplido para ninguna descomposición, no se verifica el LB y, por tanto, el lenguaje no es regular

Problemas

Demostrar si se verifica el lema de bombeo para el lenguaje descrito por la siguiente ER: 0^m1^m

Demostración:

Recordemos que debe existir una constante *n* tal que, para toda cadena *w* perteneciente al lenguaje, con |*w*| ≥ *n*, podamos dividir *w* en tres cadenas, *w* = *xyz*, de modo que:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. para todo k≥0, la cadena xy^kz también pertenece a al lenguaje

Si tomamos n=5 y w=0000011111, podremos considerar: x=0000; y=0; y z=11111, verificándose los puntos 1 y 2 previos, pero no el punto 3, ya que si k=0, por ejemplo, resultará xy^0z=000011111, que no pertenece al lenguaje.

Problemas

Dados los siguientes lenguajes:

1. Razonar si son regulares, es decir, si es posible reconocerlos con un autómata finito.

Lenguajes:

a. El que consta de todas las cadenas con el mismo número de 0 y 1 (sin ningún orden particular)

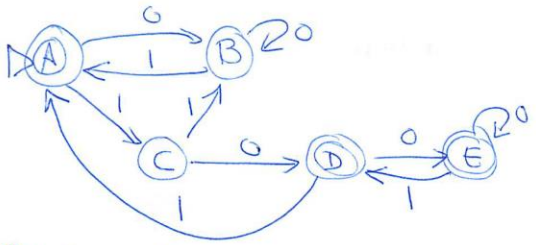
b. El conjunto vacío

c. {00, 11}

d. $(00 + 11)^*$

e. $L = \{h \in \{a, b\}^* : N_a(h) < N_b(h)\}$

f. $L = \{a^i b^j c^k / k=i-j; i, j, k >= 0\}$



Eliminamos B

HACIA B | B→
A | A
C

$\overline{AA} = 00^*1$

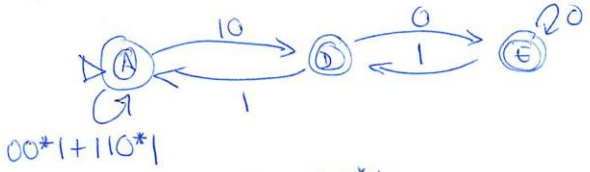
$\overline{CA} = 10^*1$

Eliminamos C

→C | C→
A | A
A | D

$\overline{AA} = 00^*1 + 110^*1$

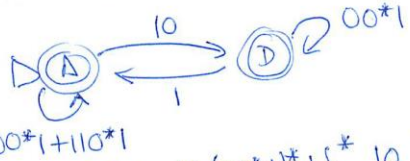
$\overline{AD} = 10$



Eliminamos E

→E | E→
D | D

$\overline{DD} = 00^*1$



$L_D = \{(00^*1+110^*1) + 10(00^*1)^*1\}^* 10(00^*1)^*$

Eliminamos D

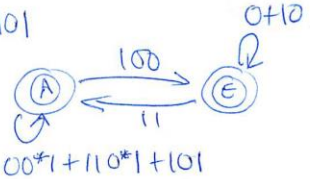
→D | D→
A | A
E | E

$\overline{AA} = 00^*1 + 110^*1 + 101$

$\overline{AE} = 100$

$\overline{EA} = 11$

$\overline{EE} = 0+10$



$L_E = \{(00^*1+110^*1+101) + 100(0+10)^*11\}^* \cdot 100(0+10)^*$