Propiedades de las Expresiones Regulares

Fabio Martínez Carrillo

Autómatas y Lenguajes Formales
Escuela de Ingeniería de Sistemas e Informatica
Universidad Industrial de Santander - UIS

9 de octubre de 2017

Agenda

Álgebra de las expresiones regulares

Dos expresiones con variables son **equivalentes** si al sustituir las variables por cualquier lenguaje, el resultado de las dos expresiones es el mismo lenguaje. Ej:

- $1 + 2 = 2 + 1$
- $x + y = y + x$

Asociatividad y conmutatividad

- *conmutatividad de la unión:* $L + M = M + L$
- *Asociatividad de la unión:* $(L + M) + N = L + (M + N)$
- *Asociatividad de la concatenacón:* $(LM)N = L(MN)$
- **La concatenación no es conmutativa**

Elemento identidad y elemento nulo

- *Identidad para la unión:* $\emptyset + L = L + \emptyset = L$
- *Identidad para la concatenación:* $\varepsilon L = L\varepsilon = L$
- *Nulo para la concatenación:* $\emptyset L = L\emptyset = \emptyset$

Ley Distributiva

- *Ley distributiva por la izquierda:* $L(M + N) = LM + LN$
- *Ley distributiva por la derecha:* $(M + N)L = ML + NL$

Ley de idempotencia

Si el resultado de aplicarlo a dos valores iguales es dicho valor

- *Ley de idempotencia para la unión* $L + L = L$

Leyes relativas a las clausuras

- $(L^*)^*$
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $L^+ = LL^* = L^*L$ siendo $L = L + LL + LLL + \ldots$
- $L^* = L^+ + \varepsilon$
- $L? = \varepsilon + L$
- $(L + M)^* = (L^*M^*)^*$

Agenda

Si ciertos lenguajes son regulares y se forma un lenguaje L a partir de ellos mediante determinadas operaciones, entonces L también es regular

Propiedades de Clausura: a partir de lenguajes regulares el resultado es regular

1. La unión
2. La intersección
3. El complementario
4. La diferencia
5. La reflexión
6. La clausura (operador *)
7. La concatenación
8. Un homomorfismo (sustitución de símbolos por cadenas)
9. El homomorfismo inverso.

Clausura para operaciones booleanas

Unión, intersección y complemento: L y M son lenguajes con alfabeto Σ

1. $L \cup M$ contiene las cadenas de L y M. Si $M = L(S)$ y $L = L(R)$ entonces $L \cup M = L(R + S)$
2. $L \cap M$ contiene las cadenas que pertencen tanto a L como a M
3. $\bar{L}$ cadenas que pertenecen a Σ^* y no pertencen a L

Clausura para la complementación

A partir de un AFD, se hace otro automata que acepte el lenguaje complementario

- Convertir la expresión regular en un AFN-ε

Clausura para la complementación

A partir de un AFD, se hace otro automata que acepte el lenguaje complementario

- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)

Clausura para la complementación

A partir de un AFD, se hace otro automata que acepte el lenguaje complementario

- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD

Clausura para la complementación

A partir de un AFD, se hace otro automata que acepte el lenguaje complementario

- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

- Convertir la expresión regular en un AFN-$_{\varepsilon}$
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

Contruya una ER que acepte cadenas formadas por **0's** ó **1's** que terminen en 01 La expresión es: $(0 + 1)^{*}01$

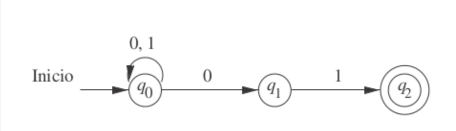
- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

La expresión es: $(0+1)^*01$

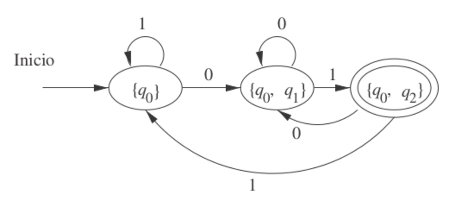
- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

Convierta la ER a un AFN-ε y luego a un AFD

- Convertir la expresión regular en un AFN-$_\varepsilon$
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

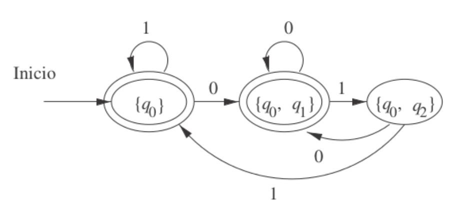


- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

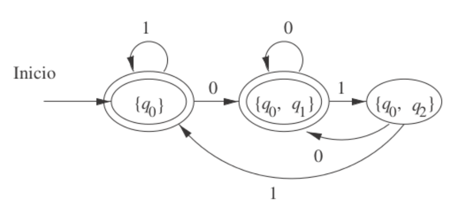


- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

Complemente los estados de aceptación



- Convertir la expresión regular en un AFN-ε
- Convertir el AFN-ε en un AFD (Construcción de subconjuntos)
- Complementar los estados de aceptación del AFD
- Convertir el AFD complementado en una expresión regular

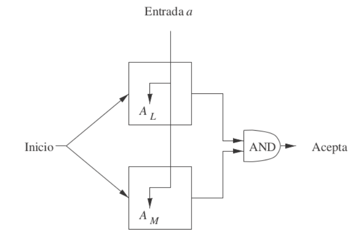


Clausura para la intersección

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

conjunto de elementos que no pertenecen al complementario de ninguno de los dos conjuntos.

Contrucción de dos Automatas en paralelo



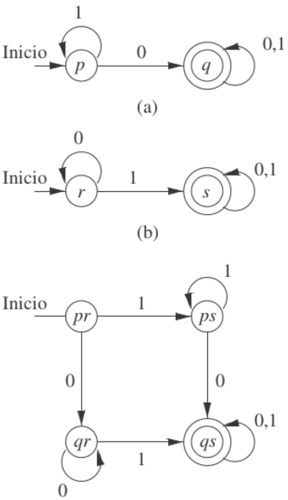
- Dado $A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$ y $A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$
- A simula el cabio de (p, q) a (s, t) ante un estimulo a

Clausura para la intersección

$$A = (Q_L \times Q_M, \Sigma, \delta_M, (q_L, q_M), F_L \times F_M)$$

- $\delta((p,q),a) = (\delta(p,a), \delta(q,a))$
- A acepta a w si y solo si $\hat{\delta}((q_L, q_M), w)$ es una pareja formada por estados de aceptación
  - $\hat{\delta}(q_L, w) \in F_L$
  - $\hat{\delta}(q_M, w) \in F_M$

Ejemplo



Producto de los dos automatas que aceptan tanto $1's$ como $0's$

Clausura para la diferencia

$L - M$

El conjunto de cadenas que pertenecen al lenguaje L pero no al lenguaje M

Demostración

$L - M = L \cap \bar{M}$

- $\bar{M}$ es regular
- $L \cap \bar{M}$ es regular

Reflexión

La *reflexión* de una cadena $a_1 a_2 \dots a_n$ es la cadena escrita en orden inverso, es decir,$a_n a_{n-1} \dots a_1$.

- w^R reflejo de w
- 0010^R es 0100
- Reflexión de un lenguaje es L^R. $L = \{001, 10, 111\}$ es $L^R = \{100, 01, 111\}$

Dado un lenguaje $L(A)$ para un autómata finito, contruimos L^R

1. Reflejamos todos los arcos del diagrama de transiciones de A.
2. El estado inicial de A es el único estado de aceptación
3. Creamos un nuevo estado inicial p_0 con transiciones ε sobre los estados de aceptación

Demostración para la reflexión

Caso Base

- $\{\varepsilon\}^R = \{\varepsilon\}$
- $\{\emptyset\}^R = \{\emptyset\}$
- $\{a\}^R = \{a\}$

Paso Inductivo

- $E = E_1 + E_2$ es $E^R = E_1^R + E_2^R$
- $E = E_1 E_2$ es $E^R = E_2^R E_1^R$
- $E = E_1^*$ es $E^R = (E_1^R)^*$, entonces:

$$w^R = w_n^R w_{n-1}^R \dots w_1^R$$

$(0 + 1)0^*$

Homorfismo

Un *homomorfismo* de cadenas es una función sobre cadenas que sustituye cada símbolo por una cadena determinada.

Ejemplo

La función h definida por $h(0) = ab$ y $h(1) = \varepsilon$ es un homomorfismo. Por ejemplo para:

- 0011 es *abab*

Podemos aplicar un homomorfismo a un lenguaje. Para 10^*1 es:

Homorfismo

Un *homomorfismo* de cadenas es una función sobre cadenas que sustituye cada símbolo por una cadena determinada.

Ejemplo

La función h definida por $h(0) = ab$ y $h(1) = \varepsilon$ es un homomorfismo. Por ejemplo para:

- 0011 es *abab*

Podemos aplicar un homomorfismo a un lenguaje. Para 10^*1 es: $(ab)^*$

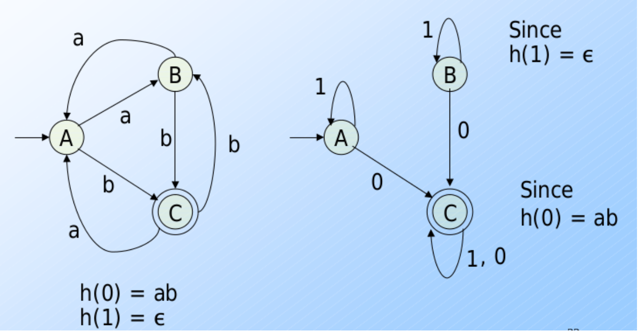
Homorfismo inverso

- Dado que h es el homorfismo y L el lenguaje de salida de h
- $h^{-1}(L) = \{w \mid h(w) \text{ están en } L\}$

Demostración

- Contruir un AFD A para L
- construir un AFD B para $h^{-1}(L)$
  - Igual número de estados
  - El mismo estado inical
  - El mismo estado final
  - El alfabeto de entrada son los simbolos para el homorfismo.
- Formalmente: $\delta_B(q, a) = \delta_A(q, h(a))$

Ejemplo



Agenda

Conversión entre representaciones

Conversión de un AFN a un AFD

La conversión puede ser **exponencial.**

- Si es AFN-ε, el cálculo de la clausura es $O(n^3)$
- La construcción de subconjuntos es 2^n
- El tiempo total es $O(n^3 2^n)$
- Si el número de estados creados es mucho menor que 2^n, entonces $O(n^3 s)$

Conversión de un AFD a un AFN

Es de complejidad $O(n)$

Conversión entre representaciones

Conversión de un Automata en una expresión regular

- Según el número de iteracciones puede tener una complejidad de $O(n^3 4^n)$
- Si convertimos un AFN a un AFD y luego a una ER $O(8^n 4^{2^n})$

Conversión de una expresión regular a un Autómata

- Requiere un tiempo lineal $O(n)$ para una expresión de longitud n

Comprobar la pertenencia de un lenguaje

Dada una cadena w y un lenguaje L

- Si L esta representado en AFD, entonces $\mid w \mid = n$. $O(n)$
- Si L esta representado en AFN con s estados, y $\mid w \mid = n$ entonces $O(ns^2)$
- Si es AFN-ε es $O(2ns^2)$ y por lo tanto $O(ns^2)$

Como comprobar si un Lenguaje es $\emptyset$?

- Proporcionar alguna representación y establecer si es vacia.
- Si existe un camino que vaya desde un estado inicial hasta un estado final.

Agenda

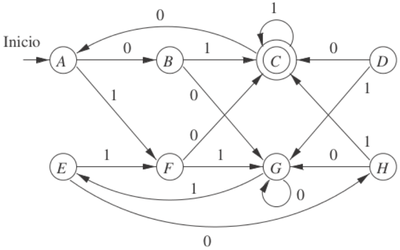
Equivalencia de estados (AFD)

Dos descripciones de dos lenguajes regulares definen el mismo lenguaje

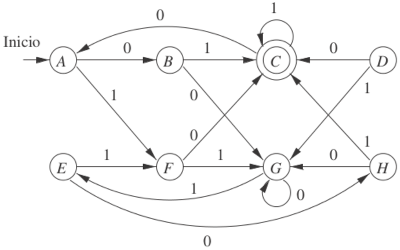
cuándo dos estados distintos p y q pueden reemplazarse por un único estado de igual comportamiento. Entonces son equivalentes.

- Para w, $\hat{\delta}(p, w)$ es un estado de aceptación si y solo si $\hat{\delta}(q, w)$ es un estado de aceptación
- Si los dos estados no son equivalentes, entonces decimos que son **distinguibles**

Cuales estados son distinguibles?

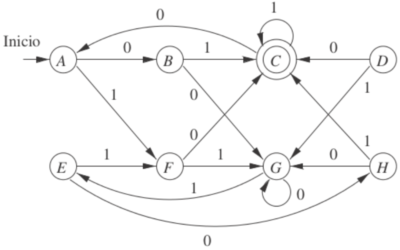


Cuales estados son distinguibles?



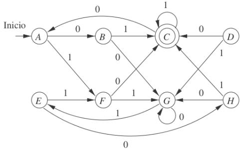
- C y G no son equivalentes, porque $\hat{\delta}(C, \varepsilon) \in$ EA, pero $\hat{\delta}(C, \varepsilon) \notin$ EA
- B y G no los distingue el 0, porque no son estados de aceptación
- A y G no lo distingue el 1 porque conducen a (F, E) que no son estados de aceptación

Cuales estados son distinguibles?



- $\hat{\delta}(A, 01) = C$ y $\hat{\delta}(G, 01) = E$. 01 demuestra que (A, G) son distiguibles o **NO** equivalentes
- $\hat{\delta}(A, 01) = C$ y $\hat{\delta}(E, 01) = C$. 01 demuestra que (A, G) son equivalentes
- $\hat{\delta}(A, 1x) = \hat{\delta}(E, x1)$.

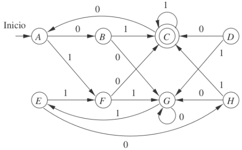
Algoritmo de llenado de tabla



	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

La *x* son pares distinguibles

Algoritmo de llenado de tabla



	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

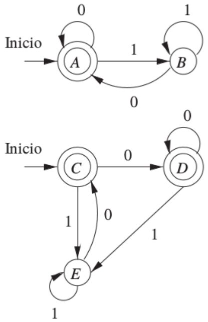
Si dos estados no son distiguibles entonces son equivalentes

Comprobar la equivalencia de Lenguajes regulares

Comprobar si dos lenguajes regulares L y M son iguales. Estos lenguajes pueden estar representados de diferentes formas

- Convertir las representaciones a **AFD**
- Hacer AFD como la unión de los $AFD(L)$ y $AFD(N)$
- Se considera solo un estado inicial.
- Con el llenado de la tabla se comprueba si los estados iniciales son equivalentes.

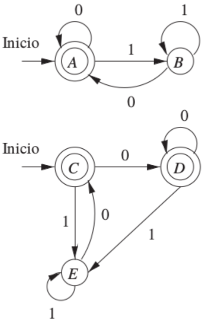
Comprobar la equivalencia de Lenguajes regulares



B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

- Cada AFD representa la cadena regular $\varepsilon + (0+1)^{*}0$

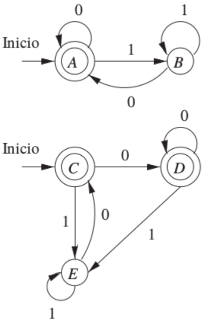
Comprobar la equivalencia de Lenguajes regulares



B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

- Cada AFD representa la cadena regular $\varepsilon + (0 + 1)^{*}0$
- Si los dos representan un solo automata, entonces se llena la tabla

Comprobar la equivalencia de Lenguajes regulares



B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

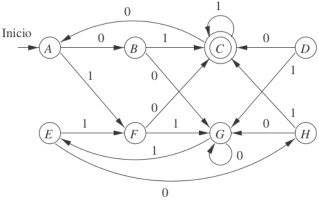
- Cada AFD representa la cadena regular $\varepsilon + (0 + 1)^*0$
- Si los dos representan un solo automata, entonces se llena la tabla
- Como A y C son equivalentes. Entonces dichos AFD aceptan el mismo lenguaje.

Minimización de un AFD

"Mminimizar" un AFD significa que podemos encontrar otro AFD equivalente que tenga menos estados

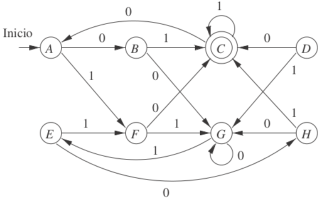
1. Eliminamos cualquier estado al que no se pueda llegar desde el estado inicial.
2. se dividen los restantes estados en bloques,
    - los estados de un mismo bloque son equivalentes
    - No hay ningún par de estados de bloques diferentes que sean equivalentes

Minimización de un AFD



	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

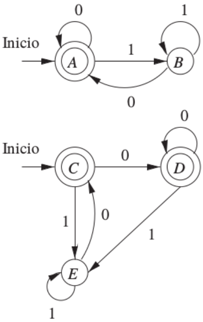
Minimización de un AFD



	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

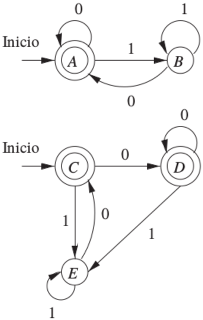
La partición de los estados en bloques equivalentes es
$(\{A, E\}, \{B, H\} \{C\}, \{D, F\}, \{G\})$

Minimización de un AFD



	A	B	C	D
B	x			
C		x		
D		x		
E	x		x	x

Minimización de un AFD



	A	B	C	D
B	x			
C		x		
D		x		
E	x		x	x

La partición de los estados en bloques equivalentes es $(\{A, C, D\}, \{B, E\})$

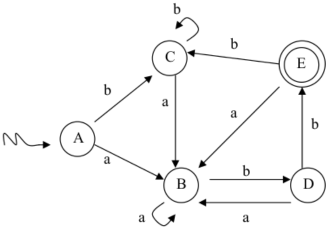
Teorema

La equivalencia de estados es transitiva. Es decir, si en un AFD $A = (Q, \Sigma, \delta, q_0, F)$ los estados p y q son equivalentes, y q y r son equivalentes, entonces p y r son equivalentes.

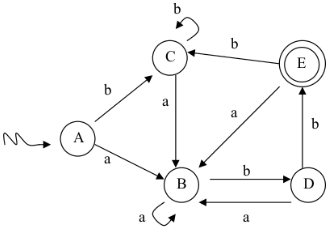
	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$*D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

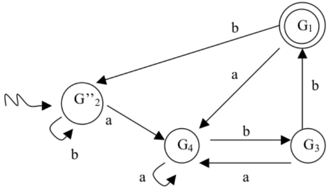
- Dibuje la tabla de estados distinguibles
- Construya el AFD equivalente con el número mínimo de estados.

Minimizar el siguiente Automata



Minimizar el siguiente Automata





Muchas gracias por su atención

Universidad Industrial de Santander