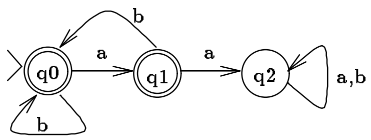
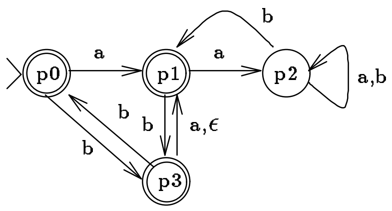
CC30B - Pauta Control 1

Prof. Ricardo Baeza Yates

P1 (1 + 1 ptos)





a) Exprese el lenguaje aceptado por el autómata de la izquierda como una expresión regular.

El primer estado final acepta los strings que tienen 0 o más b's o ab's, es decir $(b + ab)^*$. El segundo acepta lo que acepta el primer estado seguido de una a. Uniendo ambos conjuntos: $(b + ab)^*(\epsilon + a)$.

b) Convierta el autómata de la derecha en un autómata finito determinístico.

El único estado que tiene transiciones ϵ es $p3$ y su clausura es $\{p1, p3\}$. Luego si hay que incluir $p3$ despues de una transición, debemos incluir también $p1$. La tabla queda:

Estado	a	b	¿Final?
$p0$	$p1$	$p1, p3$	Si
$p1$	$p2$	$p1, p3$	Si
$p1, p3$	$p1, p2$	$p0, p1, p3$	Si
$p2$	$p2$	$p1, p2$	No
$p1, p2$	$p2$	$p1, p2, p3$	Si
$p0, p1, p3$	$p1, p2$	$p0, p1, p3$	Si
$p1, p2, p3$	$p1, p2$	$p0, p1, p2, p3$	Si
$p0, p1, p3$	$p1, p2$	$p0, p1, p3$	Si
$p0, p1, p2, p3$	$p1, p2, p3$	$p0, p1, p2, p3$	Si

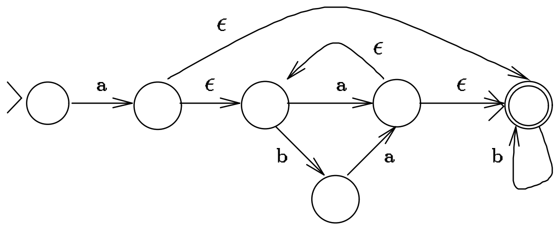
P2 (.5 + 1 ptos)

Dada la expresión regular $a(a + ba)^*b^*$:

a) Indique todas las palabras de largo 4 que pertenecen al lenguaje representado por esta expresión regular.

$aaaa$, $aaba$, $abaa$, $abab$, $aaab$, $aabb$, $abbb$

b) Construya un autómata finito no determinístico que reconozca este lenguaje.



P3 (0.5 + 0.5 ptos)

Cierto o falso? Demuestre o dé contraejemplos

a) Todo subconjunto de un lenguaje regular es regular.

   Falso. Por ejemplo, Σ^* es un lenguaje regular que contiene muchos lenguajes que no son regulares, entre ellos $L = \{a^n b^n, n \geq 0\}$.

b) Si L es regular también lo es $\{xy/x \in L, y \notin L\}$

   Verdadero. El complemento de un lenguaje regular es regular (las palabras que no están en L) y la concatenación de dos lenguajes regulares es un lenguaje regular.

P4 (1.5 ptos)

Escriba en pseudo-lenguaje una función $Acepta(M, w)$ que dado un AFND $M = (K, \Sigma, \Delta, q, F)$ y una palabra w, retorne V o F, si pertenece o no al lenguaje que acepta M. Puede usar la notación $M.K$, $M.q$, etc., para obtener cada elemento del autómata y suponer que todas las operaciones básicas que necesite ya existen (por ejemplo, operaciones de conjuntos).

Solución:

Es posible escribir una función recursiva que sigue todos los posibles caminos y retorna V si acepta w. Sin embargo no es muy eficiente porque puede repetir trabajo. Una versión no iterativa que va manteniendo todos los estados en que podría estar el autómata, incluyendo los que se obtienen al seguir transiciones ϵ, es mejor:

```
Acepta(M, w)
conjunto S, S'
{
    S ← Clausura-ε({M.q})                         Estados activos iniciales
    for( k ← 1; k ≤ |w|; k ← k + 1 ) {
        S' ← φ                                     Nuevos estados activos
        for( s ∈ S ) S' ← S' ∪ Δ(s, w[k])
        S ← Clausura-ε(S')                         Expandir los estados activos
    if( S ∩ M.F ≠ φ ) return( V )
    else return( F )
}

Clausura-ε(Q)      Calcula la clausura de transiciones ε de un conjunto de estados Q
conjunto S, N, P
{
    N ← Q; P ← Q; S ← Q        Estados por incluir, ya incluidos y resultado
    while( N ≠ φ )
        for( n ∈ N ) S ← S ∪ Δ(n, ε)
    N ← S − P
    P ← P ∪ N
}
```