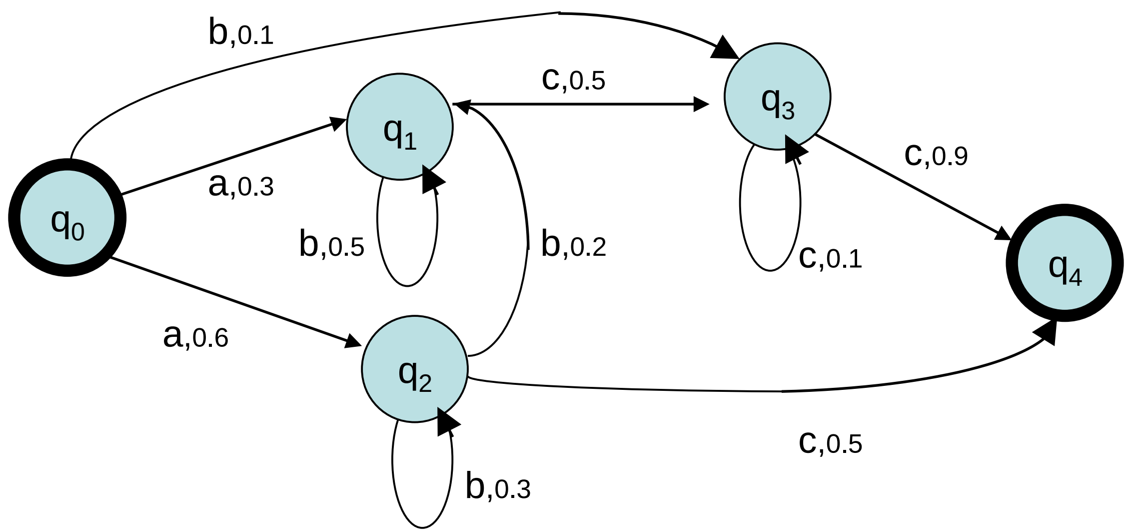
Algoritmo de Viterbi



$$P(q_0 q_1 q_2 .. q_n \mid \alpha_1 \alpha_2 .. \alpha_n) = p(q_0).p(q_1, \alpha_1, q_2).p(q_2, \alpha_2, q_3)....p(q_{n-1}, \alpha_n, q_n)$$

Ejemplo: α=abbcc

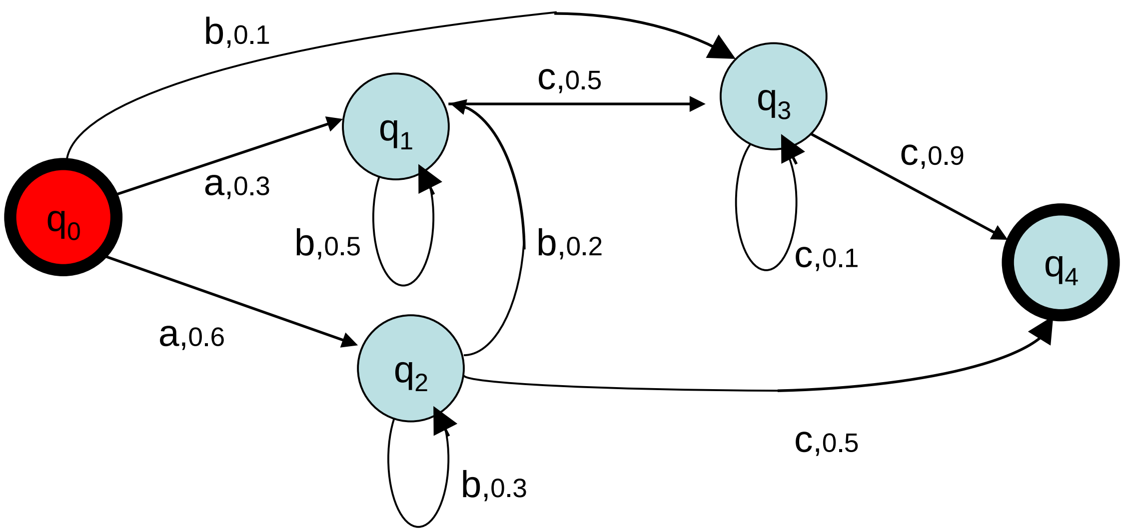
$P(q_0,q_1,q_1,q_1,q_3,q_4|abbcc)=1*0,3*0,5*0,5*0,5*0,9$

$P(q_0,q_2,q_1,q_1,q_3,q_4|abbcc)=1*0,6*0,2*0,5*0,5*0,9$

Algoritmo recursivo

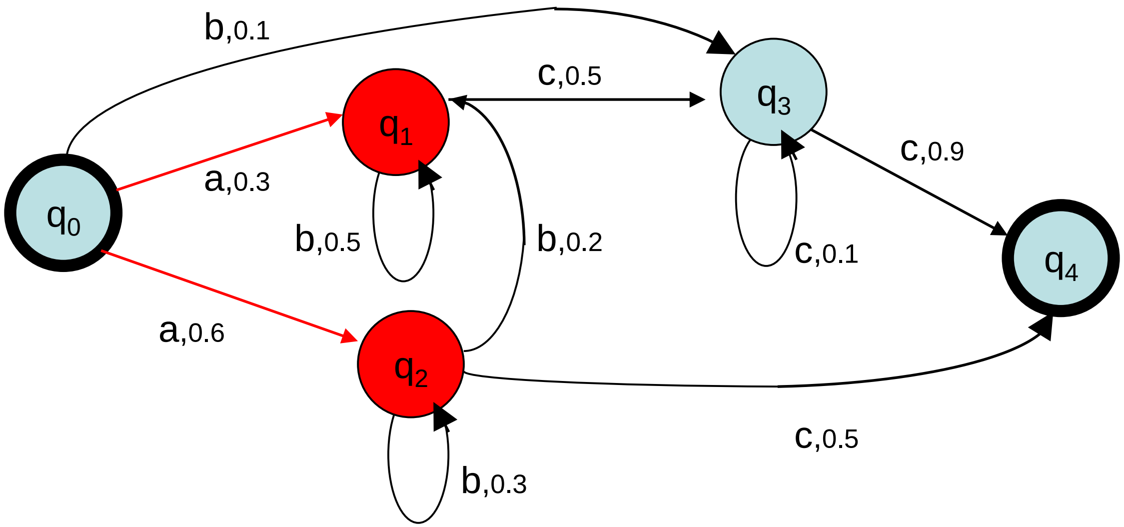
$$\mathrm{Prob}(\mathrm{q_i}, t) = \begin{cases} 1 & t = 0 \wedge q_i = q_0 \\ 0 & t = 0 \wedge q_i \neq q_0 \\ \max\limits_{q_j \in pred(q_i)} \left\{ \mathrm{Prob}(\mathrm{q_j}, t-1) * P(q_j, \alpha_t, q_i) \right\} & t > 0 \end{cases}$$

Algoritmo de Viterbi



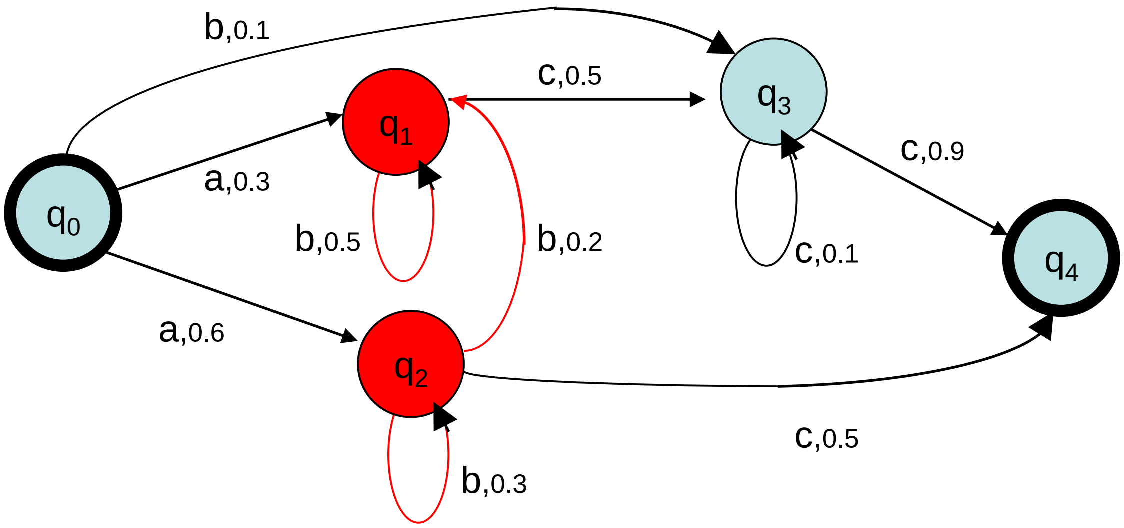
Ejemplo: α=

Algoritmo de Viterbi



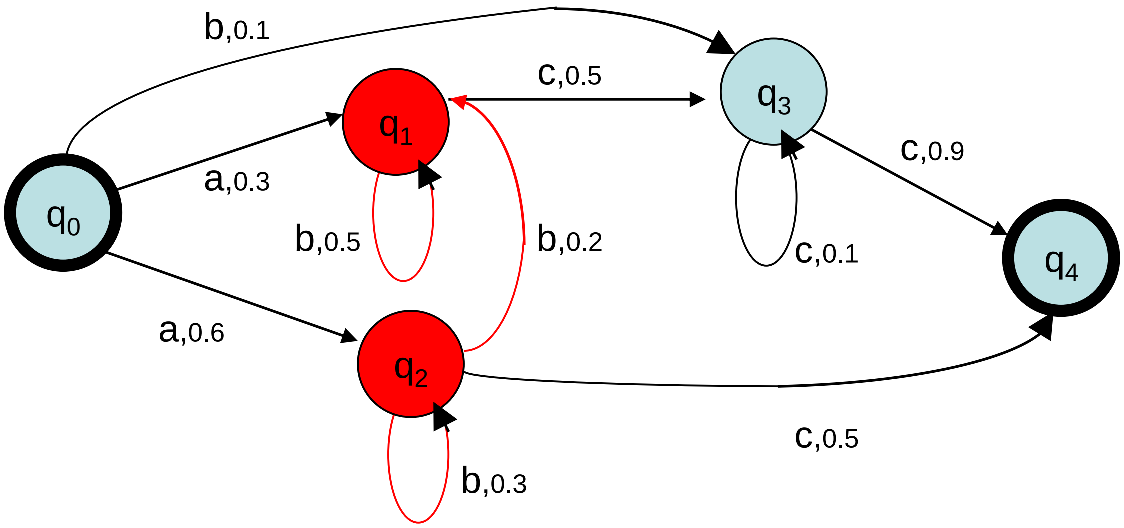
Ejemplo: α=a

Algoritmo de Viterbi



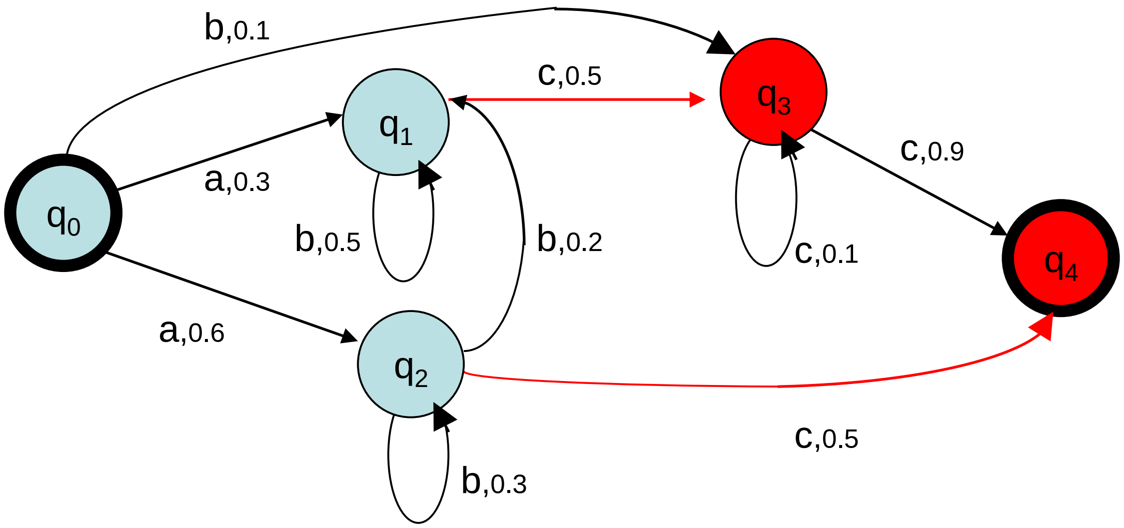
Ejemplo: α=ab

Algoritmo de Viterbi



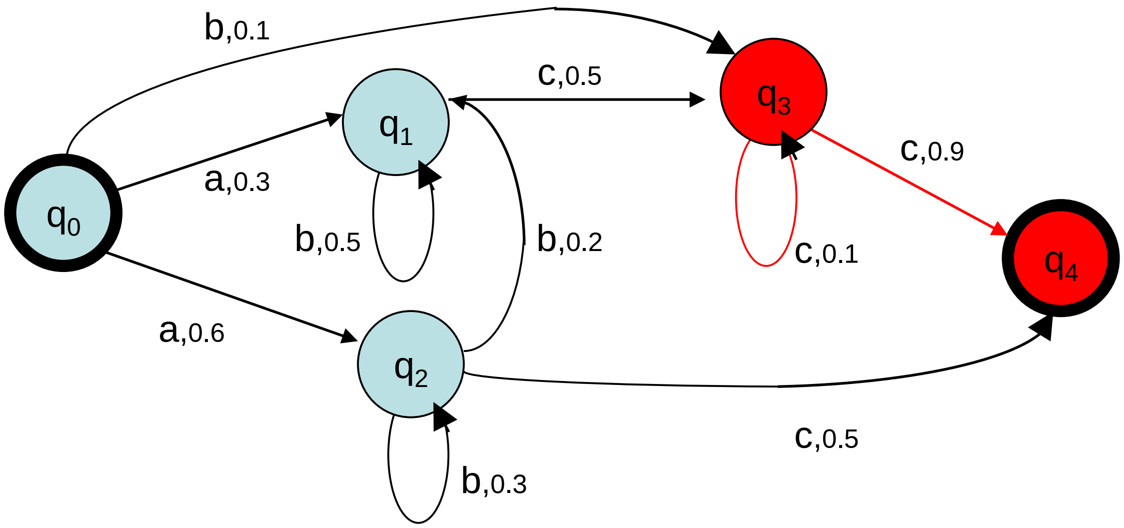
Ejemplo: α=abb

Algoritmo de Viterbi

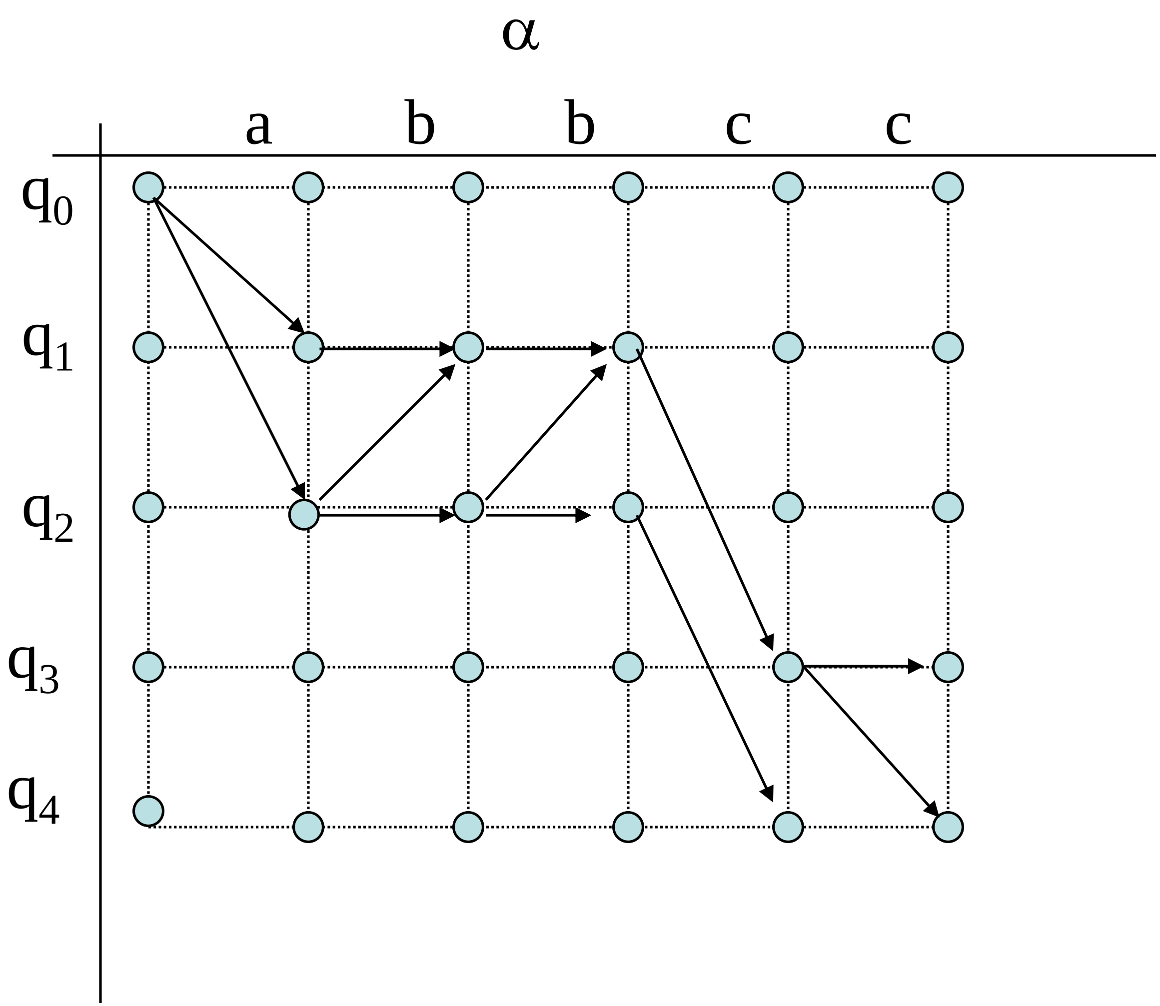


Ejemplo: α=abbc

Algoritmo de Viterbi



Ejemplo: α=abbcc

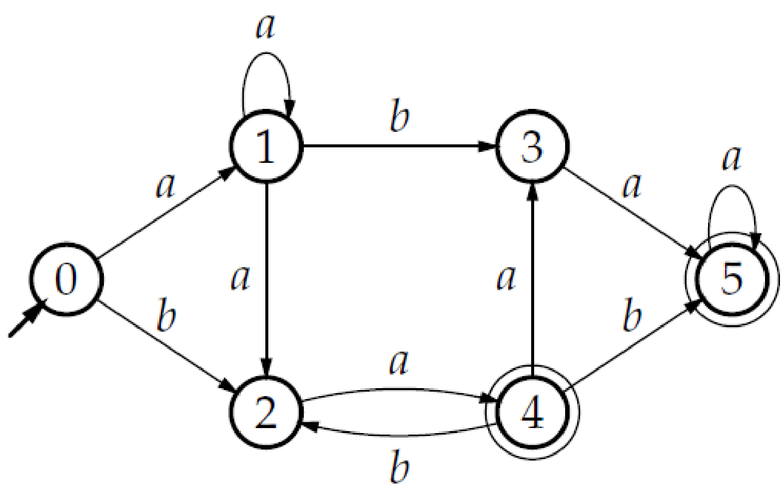

α
a
b
b
c
c
q0
q1
q2
q3
q4

Análisis de una cadena con un autómata finito no determinista **(NO es de optimización)**

Un **autómata finito no determinista** (AFND) es una quíntupla
A = (Σ,Q, q_0, E, F) donde

- Σ es un alfabeto de símbolo **terminales**;
- Q es un **conjunto** finito **de estados**;
- q_0 es un elemento de Q y se denomina **estado inicial**;
- $E : Q \times \Sigma \times Q$ es un **conjunto de transiciones entre estados**;
- y F es un subconjunto de F que recibe el nombre de **conjunto de estados finales**.

Ejemplo:



Dada una cadena x y un autómata A, deseamos saber si x pertenece a $L(A)$

Ecuación recursiva

Denotemos con $L(q)$, para todo q de Q, al lenguaje $L((\Sigma, Q, q_0, E, \{q\}))$. El conjunto $L(q)$ puede definirse recursivamente:

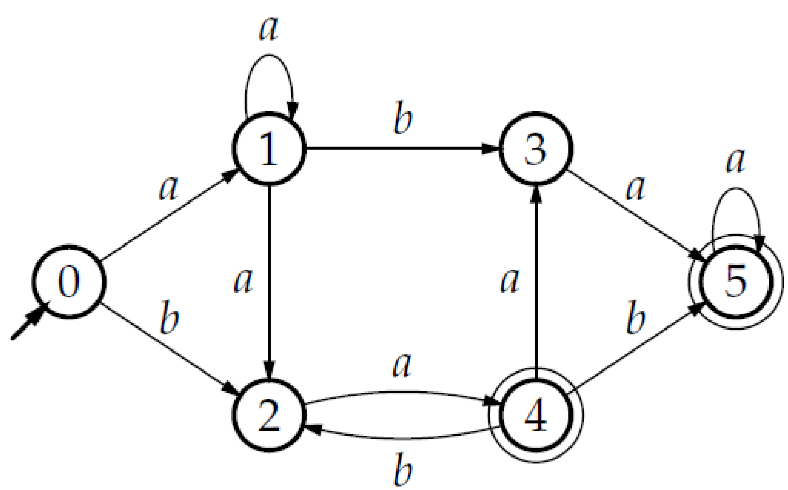
$$L(q) = \bigcup_{(q',a,q) \in E} \{xa \mid x \in L(q')\}.$$

$$P(i,q) = \begin{cases} q = q_0, & \text{si } i = 0; \\ \bigvee_{(q',x_i,q) \in E} P(i-1,q'), & \text{si } i > 0; \end{cases}$$

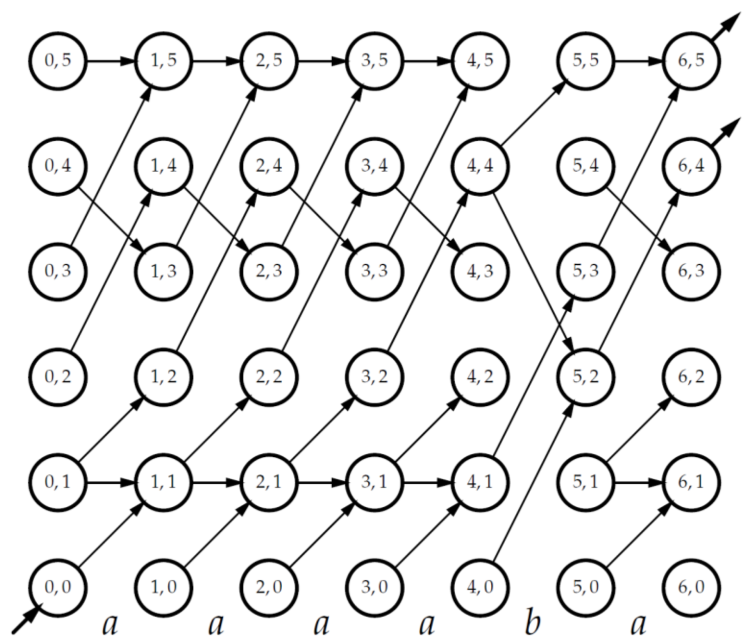
donde $\bigvee$ calcula la o-lógica de varios valores.

La pertenencia de x a $L(A)$ se determina calculando

$$\bigvee_{q \in F} P(|x|, q).$$



Ejemplo: Grafo de dependencias para la cadena aaaaba



```
def accepts(x, Q, q0, preds, F):
    P = {}
    for q in Q:
        P[q,0] = False
    P[q0,0] = True
    for i in xrange(1, len(x)+1):
        for q in Q:
            P[q,i] = False
            for (q1, c) in preds[q]:
                if c == x[i-1] and P[q1,i-1]:
                    P[q,i] = True
                    break
    accepted = False
    for q in F:
        if P[q,len(x)]:
            accepted = True
            break
    return accepted
```

Coste temporal $O(|x||Q|^2)$

Coste espacial $O(|x||Q|)$