Equivalencia y minimización de AFDs

Semana 5

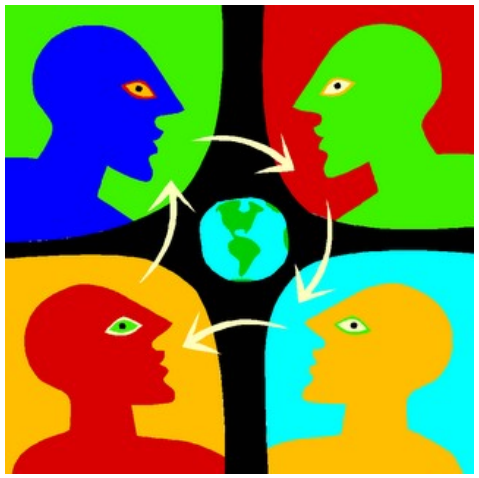
Teoría de la Computación

UNIVERSIDAD PRIVADA DEL NORTE

Laureate International Universities®

Definición

- Decimos que dos autómatas M1 y M2 son equivalentes cuando que aceptan exactamente el mismo lenguaje, es decir L(M1) = L(M2)

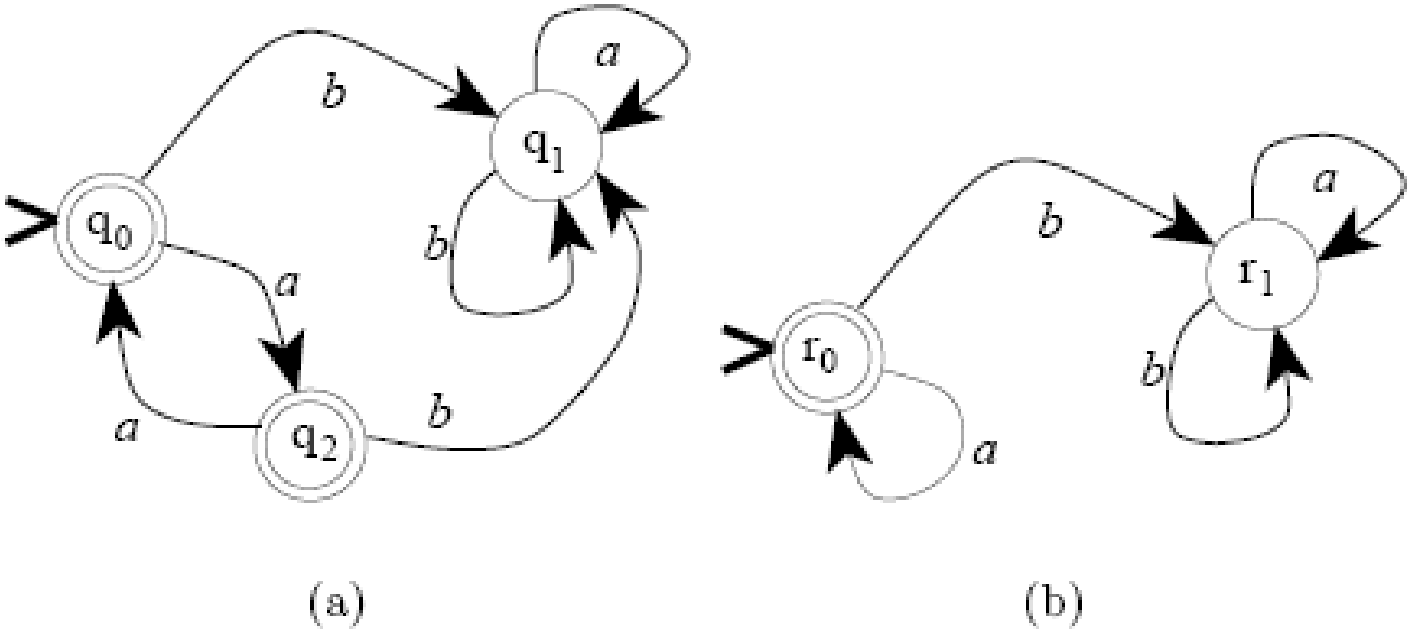


¿Cuándo dos autómatas no son equivalentes?

Cuando uno de los dos autómatas acepta una palabra que no es aceptada por el otro autómata, podemos concluir que los dos autómatas no son equivalentes

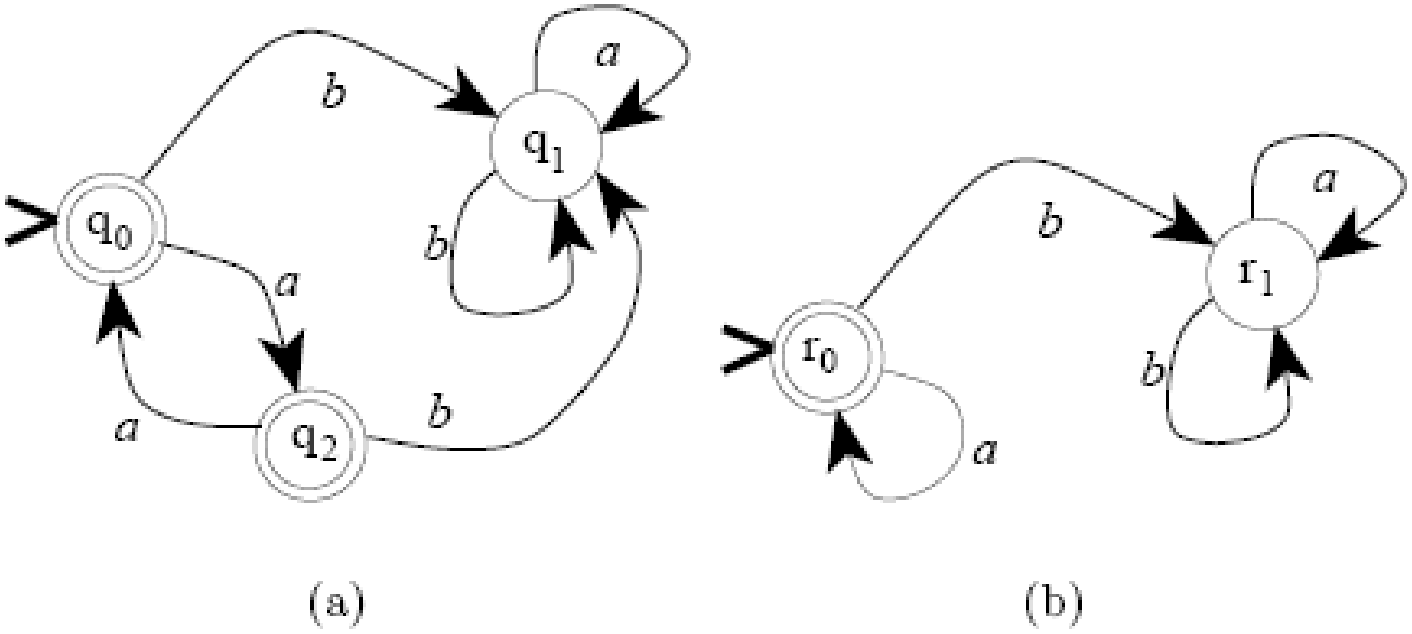
Ejemplo

- ¿Estos AFDs son equivalentes?



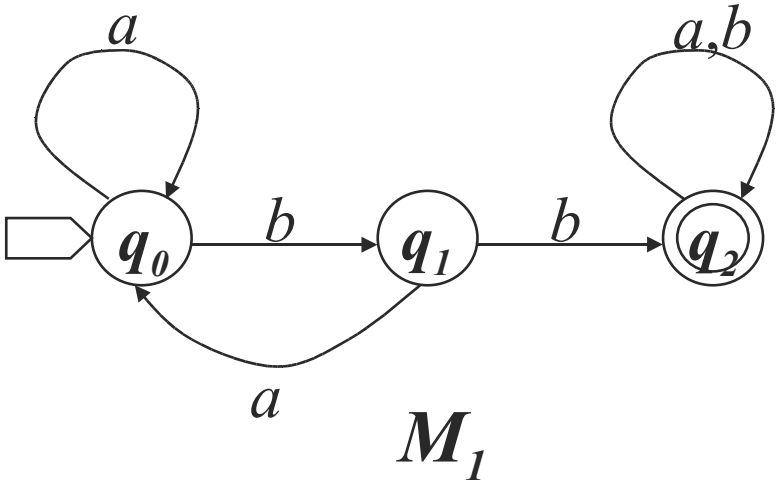
Ejemplo

- ¿Estos AFDs son equivalentes?

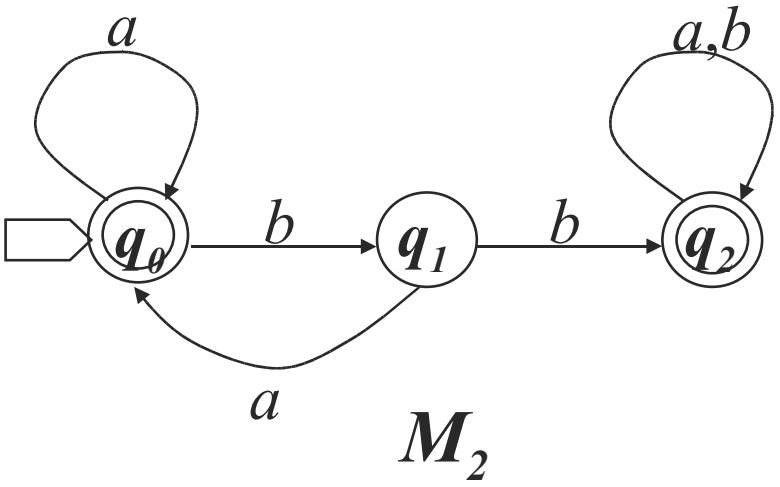


Son equivalentes

Ejemplo



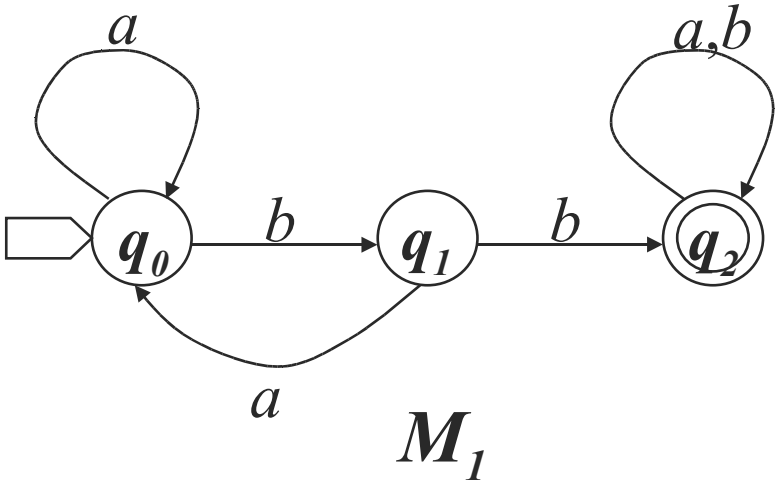
M_1



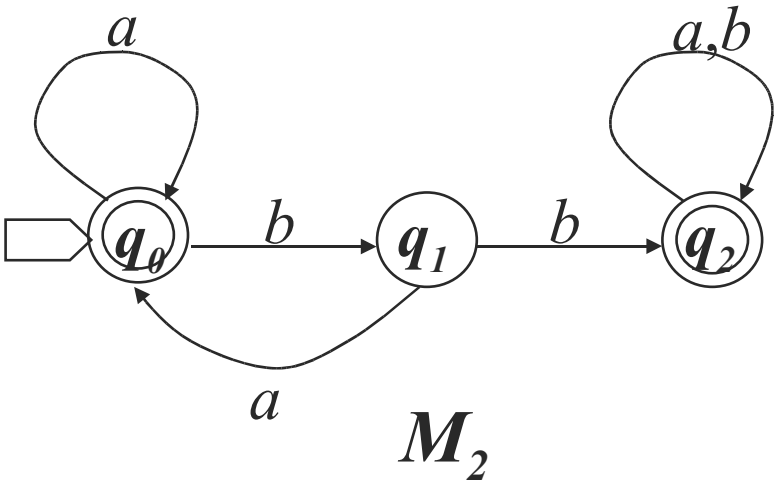
M_2

¿Son M_1 y M_2 equivalentes?

Ejemplo



M_1

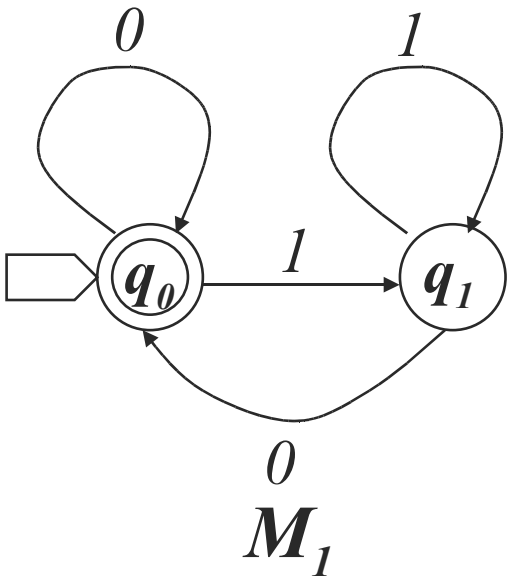


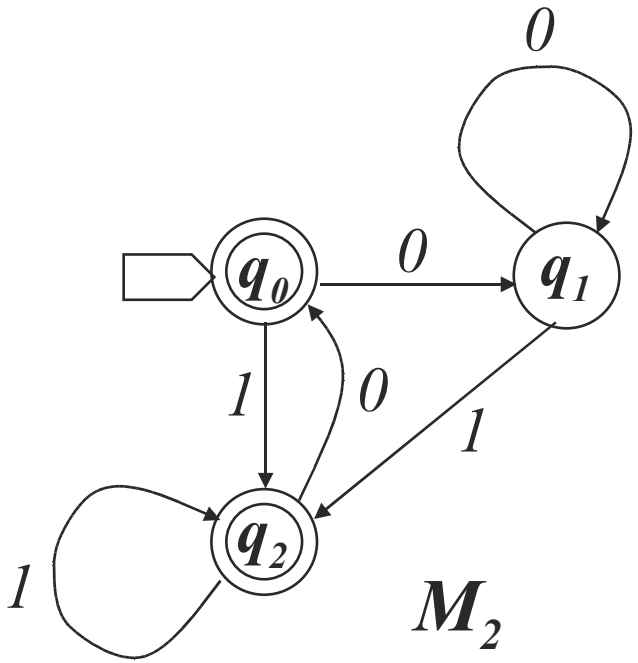
M_2

¿Son M_1 y M_2 equivalentes? ¡No!

¿Por qué? Porque M_1 no acepta λ y M_2 sí.

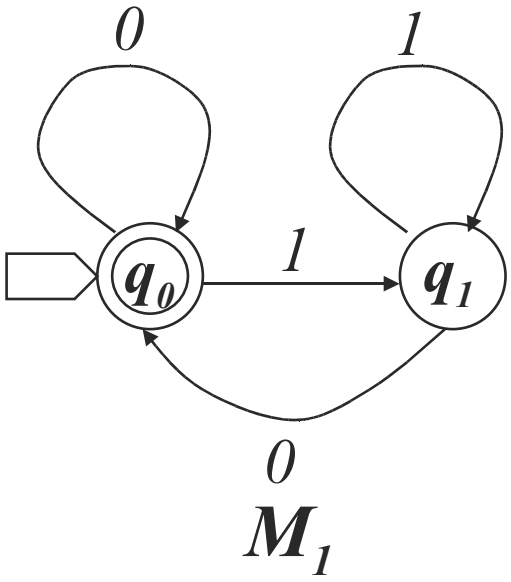
Ejemplo



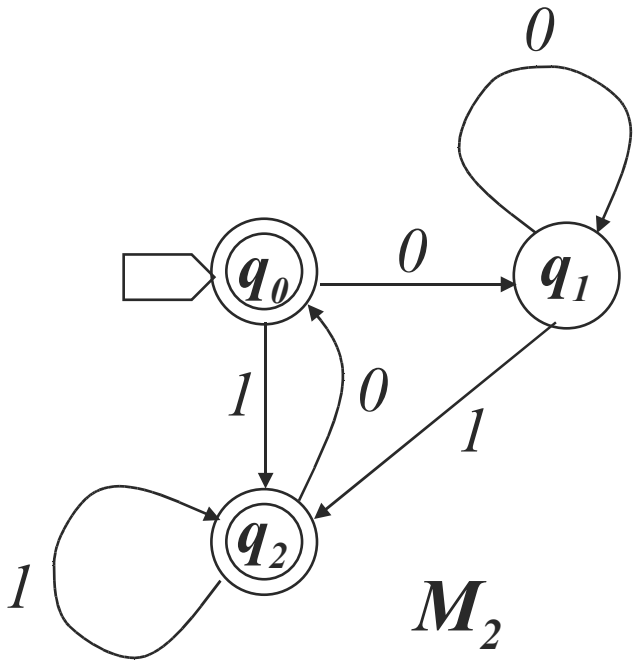


¿Son M_1 y M_2 equivalentes?

Ejemplo



M_1



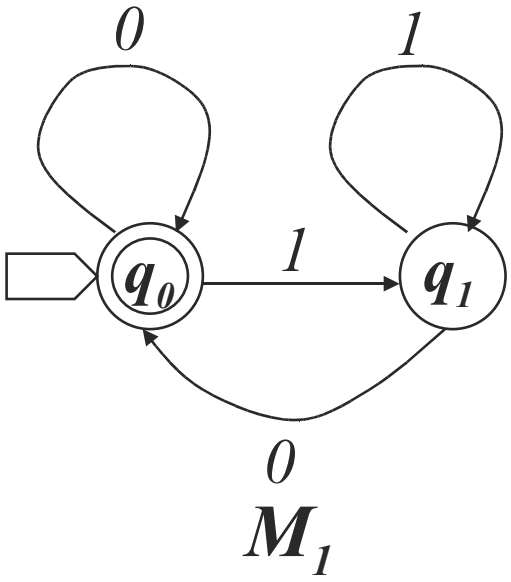
M_2

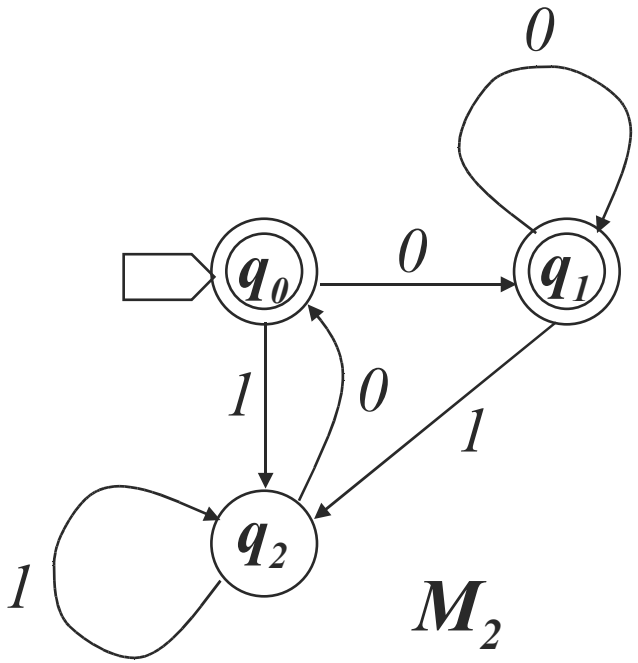
¿Son M_1 y M_2 equivalentes? **¡No!**

¿Por qué? **Porque M_1 acepta *0* y M_2 no.**

Porque M_2 acepta *01* y M_1 no.

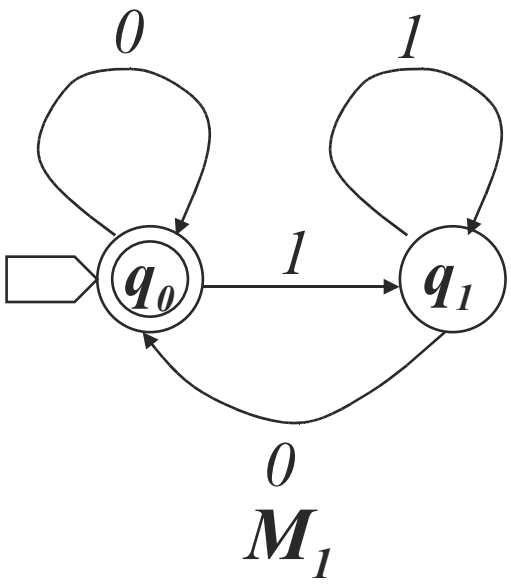
Ejemplo



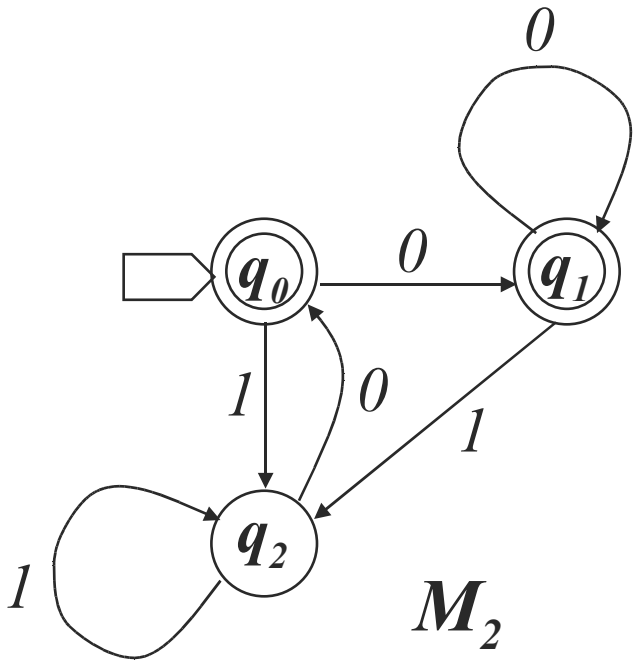


¿Ahora son M_1 y M_2 equivalentes?

Ejemplo



M_1



M_2

¿Ahora son M_1 y M_2 equivalentes? **¡SI!**

¿Por qué? **Porque todas las palabras que son aceptadas por *M1* también lo son por *M2* y viceversa.**

¿TODAS las palabras?

- Para poder decir que dos autómatas son equivalentes, debemos verificar que **TODAS** las palabras aceptas por uno de los autómatas son aceptadas por el otro y viceversa.

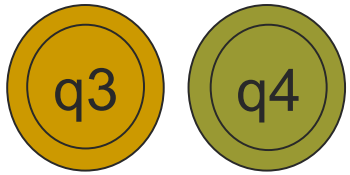
¿Cómo podemos verificar TODAS las palabras? ¿Cómo podemos encontrar una palabra que es aceptada por uno de los autómatas pero no por el otro?

¿Cómo saber si dos autómatas son equivalentes?

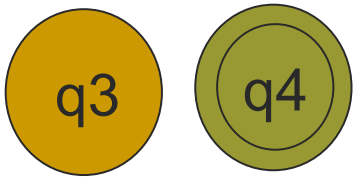
- **Teorema de Moore**. Existe un algoritmo basado en la comparación de estados para saber si dos autómatas son equivalentes.
- **Definición**. Dos estados son *compatibles* si ambos son finales o ninguno de los dos lo es. Si uno es final y el otro no lo es, entonces se dice que son *incompatibles*.

Estados compatibles

Compatibles



Compatibles



Incompatibles

Teorema de moore

- El teorema de moore nos permite crear un algoritmo para definir si dos autómatas son equivalentes o no.
- Consiste en la construcción de un árbol de comparación de autómatas.
- Este árbol permite convertir el problema de la comparación de los lenguajes aceptados en un problema de comparación de estados de los autómatas.

Teorema de Moore

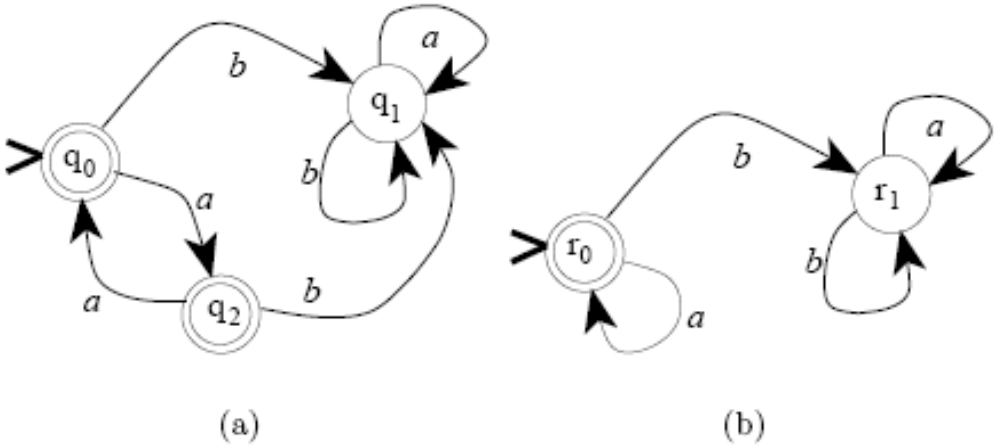
- Definición: Decimos que dos estados q y q' son compatibles si ambos son finales o ninguno de los dos es final. En caso contrario, son estados incompatibles.
- La idea del algoritmo de comparación de AFD1 y AFD2 consiste en averiguar si existe alguna secuencia de caracteres w tal que siguiéndola simultáneamente en AFD1 y AFD2 se llega a estados incompatibles. Si dicha secuencia no existe, entonces los autómatas son equivalentes.
- El único problema con esta idea estriba en que hay que garantizar que sean cubiertas todas las posibles cadenas de caracteres w, las cuales son infinitas en general. Por ello se pensó en explorar todas las posibles combinaciones de estados mediante un árbol.

Construcción del árbol de comparación

- Para dos autómatas
  M = (Q, Σ, δ, q0, F) y M' = (Q',Σ', δ', q0', F):
  - Inicialmente la raíz del árbol es el par ordenado (q_0, q_0') que contiene los estados iniciales de M y M' respectivamente.
  - Para cada n ∈ Σ se calcula $q_n = \delta(q,n)$ y $q'_n = \delta(q',n)$. Se forma el par ordenado (q_n, q'_n) como hijos de (q, q') para cada rama n, si no estuvieran ya.
  - Si aparece en el árbol un par (q, q') de estados incompatibles, se interrumpe la construcción del mismo, concluyendo que los dos autómatas no son equivalentes. En caso contrario se continúa a partir del paso 2 para cada nuevo nodo hijo.
  - Si no aparecen nuevos pares (q, q') que no estén ya en el árbol, se termina el proceso, concluyendo que los dos autómatas son equivalentes.

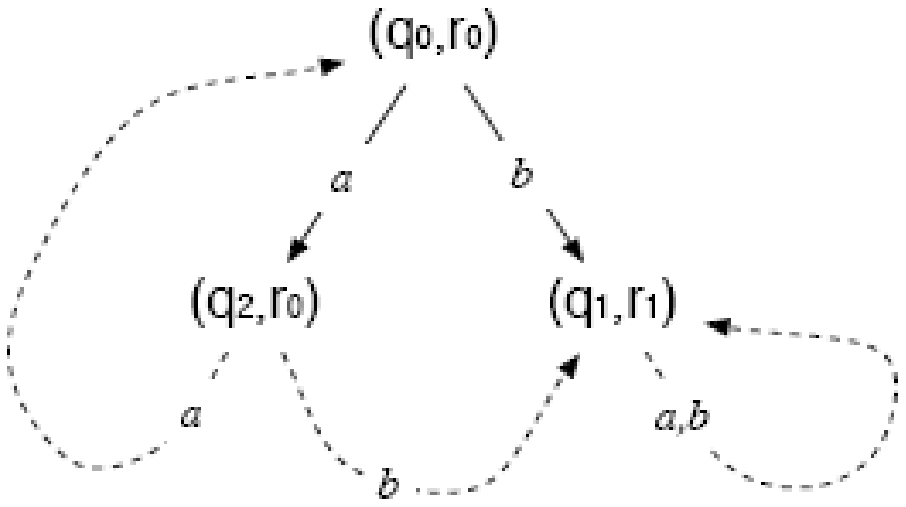
Ejemplo 1

- Árbol de comparación de AFD a y AFD b



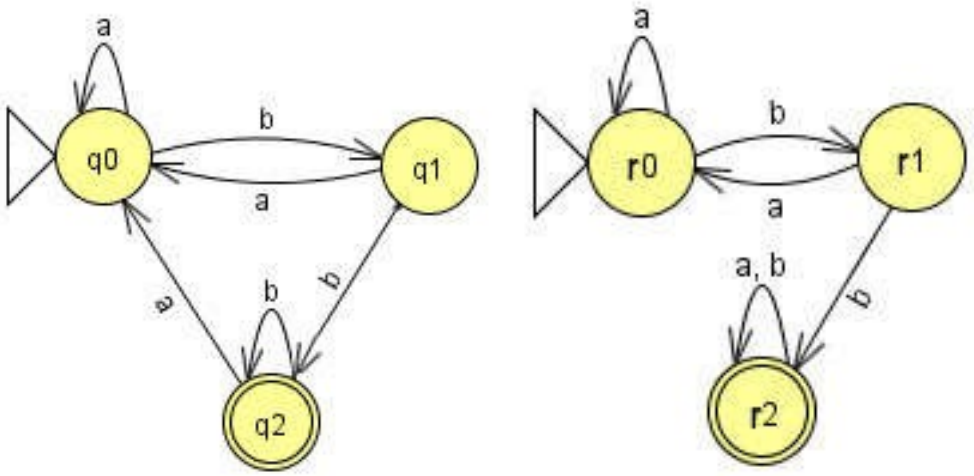
Ejemplo

- Árbol de comparación de AFD a y AFD b

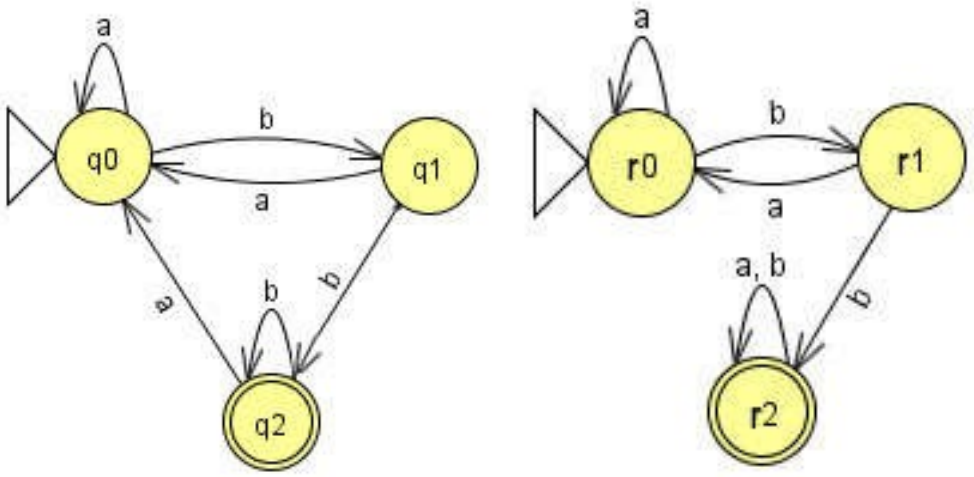


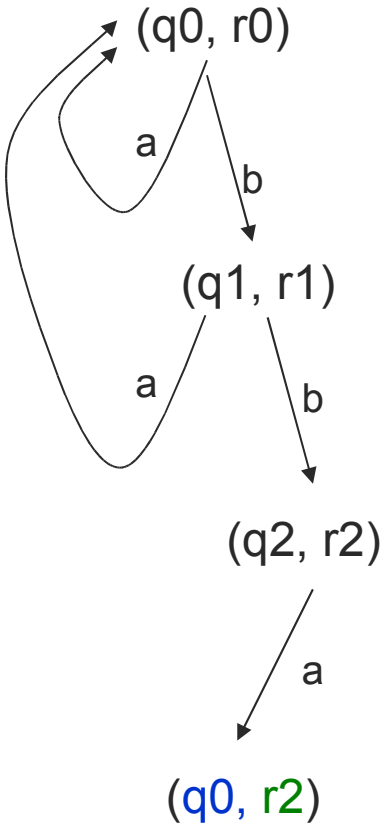
Ejemplo 2

- ¿Son estos AFD equivalentes?



Ejemplo 2

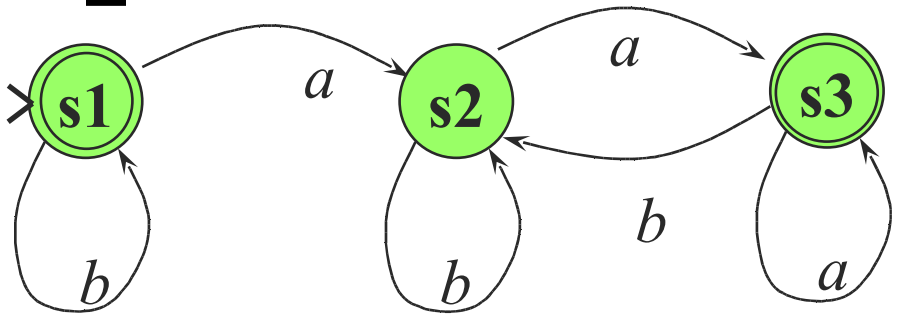


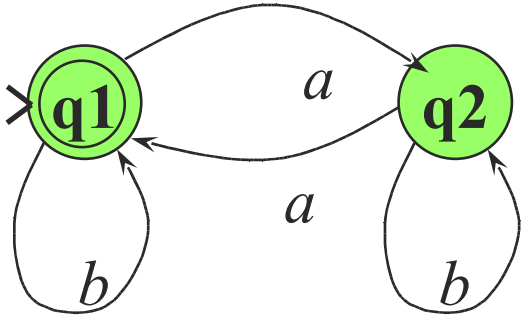


No son equivalentes

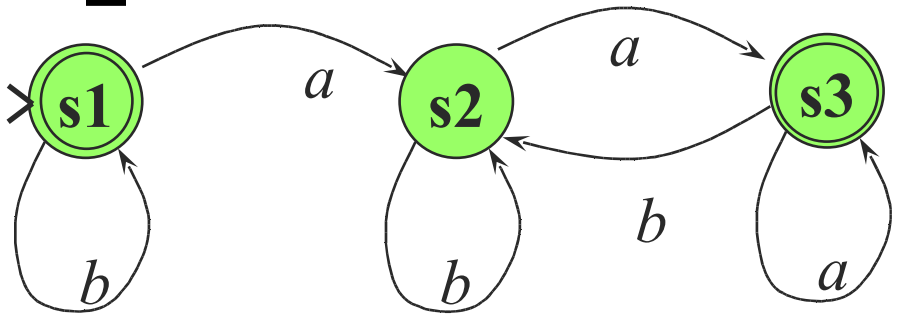
¡incompatibles!

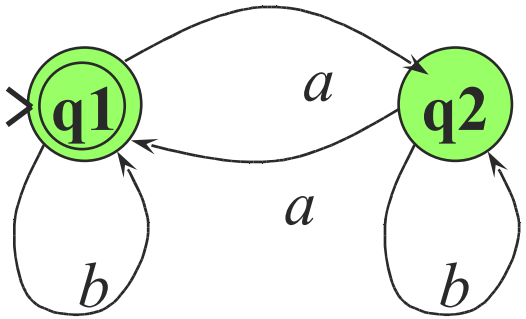
Ejercicio 1

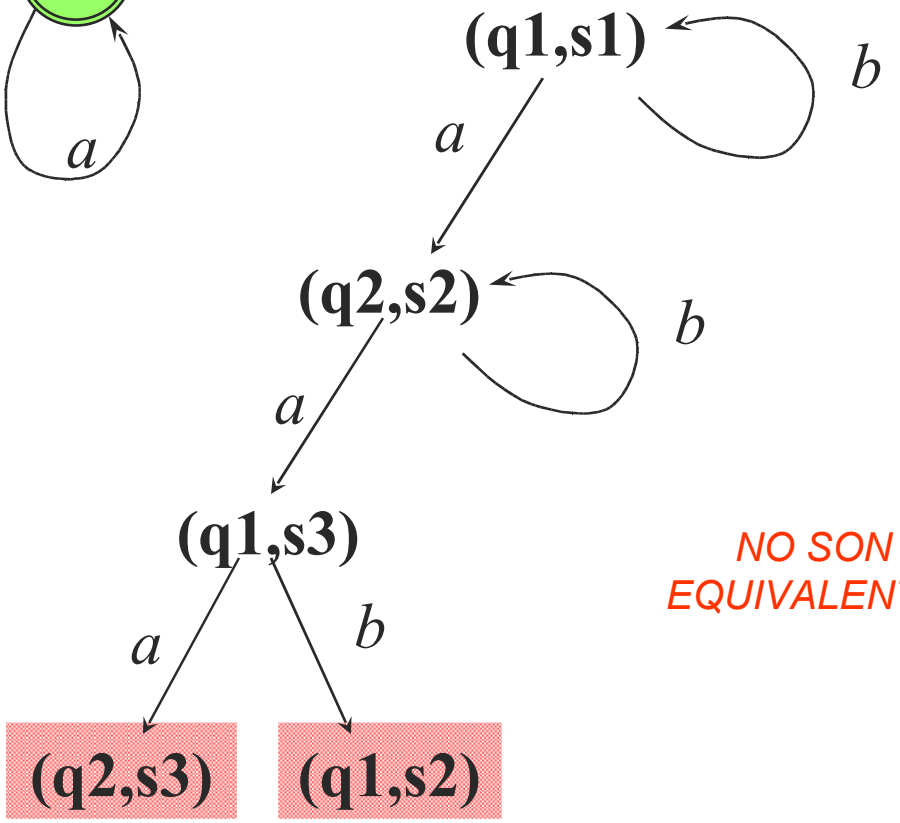




Ejercicio 1







¡incompatibles!

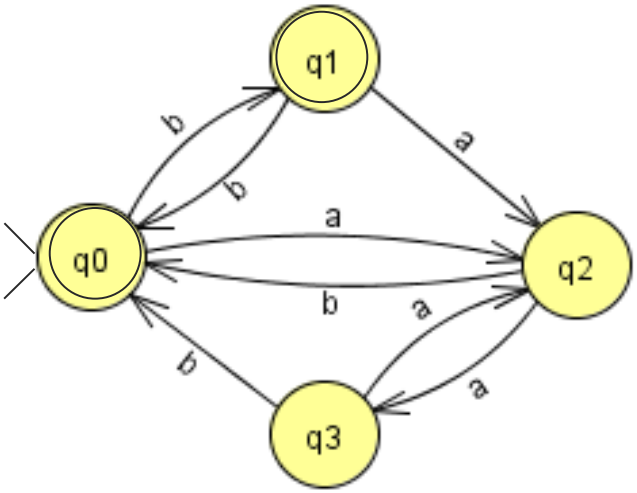
¡incompatibles!

NO SON EQUIVALENTES

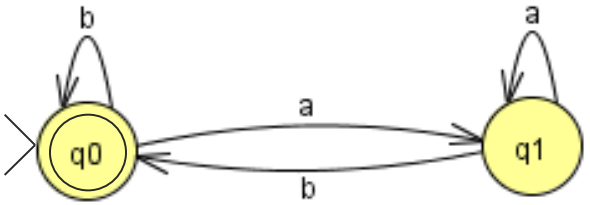
Ejercicio 2

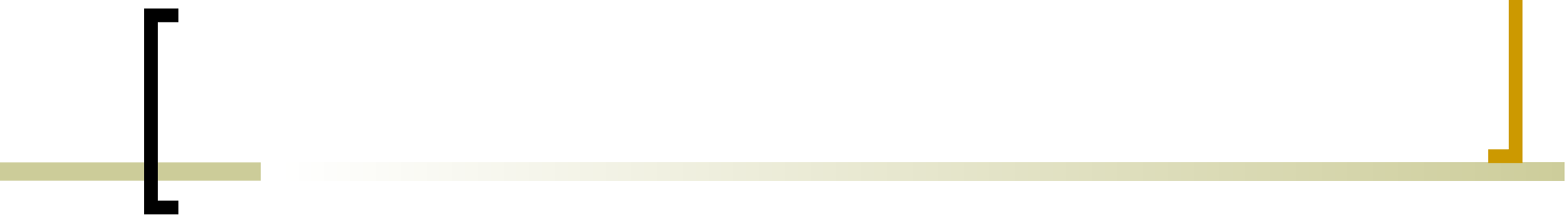
- ¿Son equivalentes?

(a)

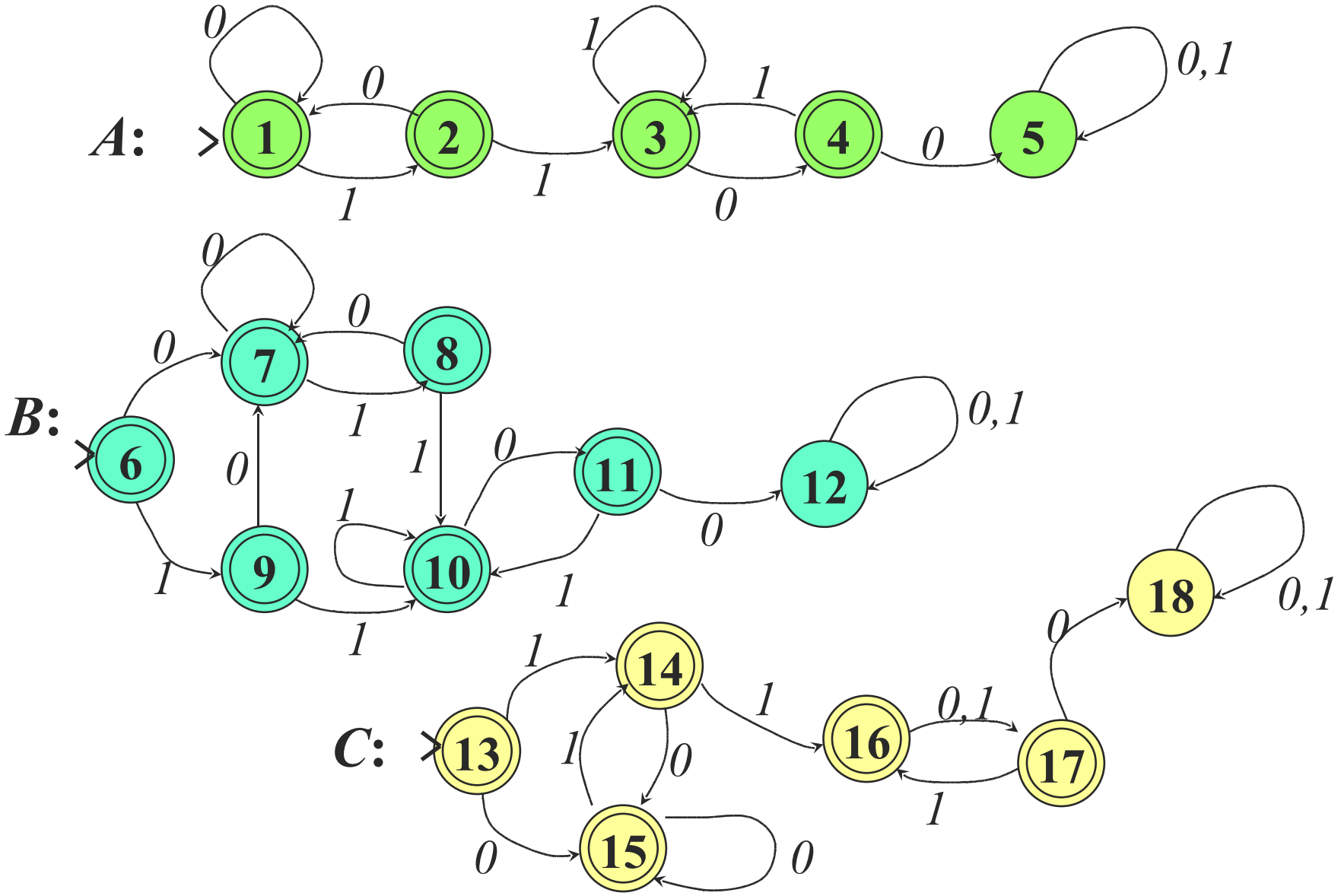


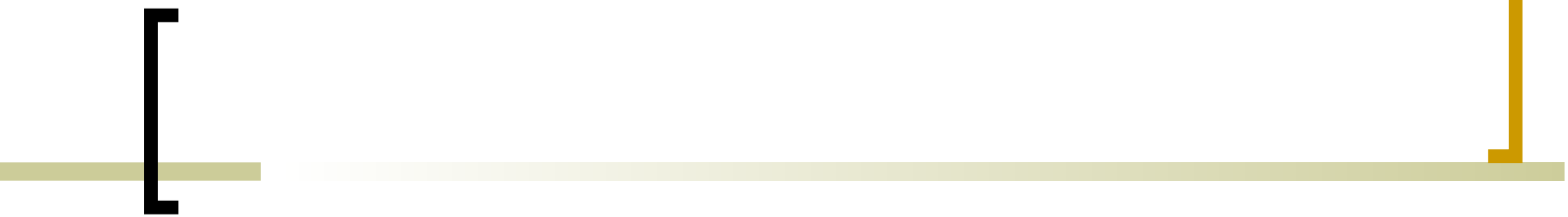
(b)





Determinar si los siguientes AFD's son/no son equivalentes:





Respuesta al ejercicio anterior

- A y B sí son equivalentes.
- B y C no son equivalentes.
- A y C no son equivalentes.

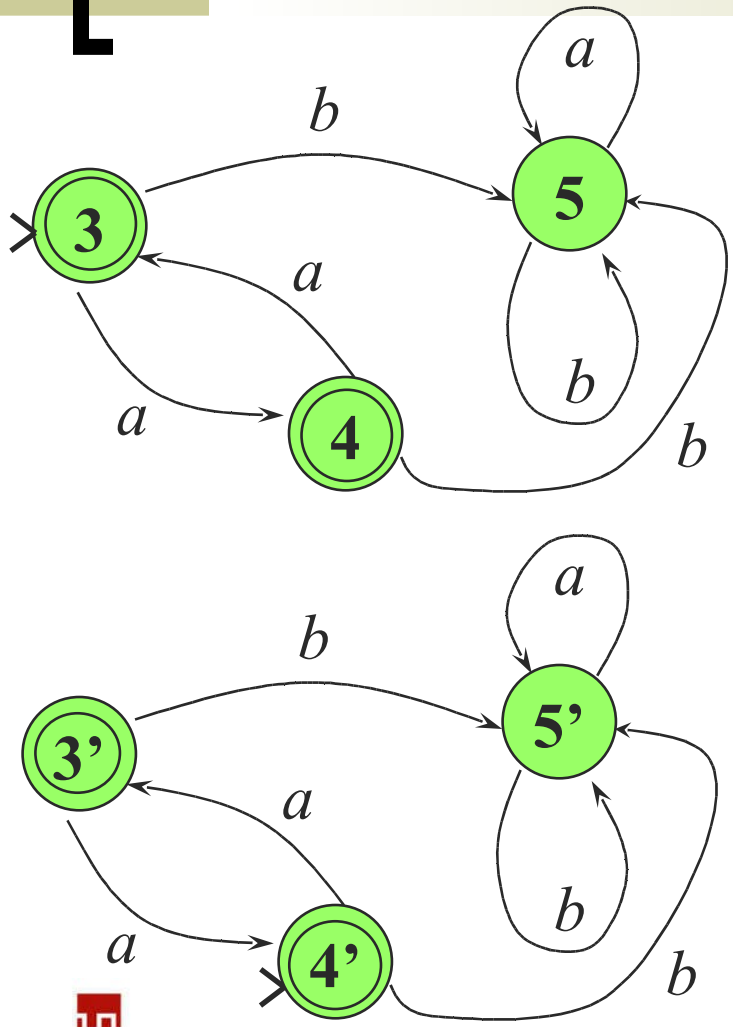
Simplificación de AFD

- Decimos que un autómata es una simplificación de otro si tiene menos estados pero acepta el mismo lenguaje.
- Vamos a entender por simplificación la reducción en el número de estados, pero aceptando el mismo lenguaje que antes de la simplificación. Más aún, llamaremos **minimización** a la obtención de un autómata con el menor número posible de estados.

Método de simplificación por estados equivalentes

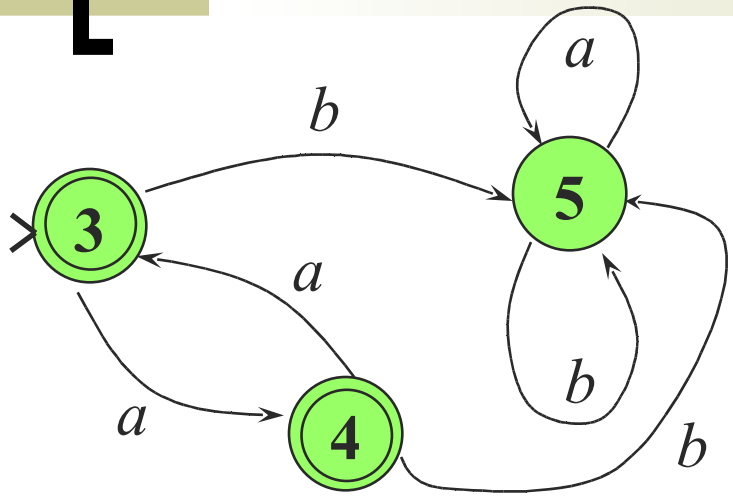
- Decimos que en un AFD dos estados son *equivalentes* si al tomar uno o el otro como estado inicial, los lenguajes aceptados por los AFD's resultantes son iguales. En otras palabras, dado un AFD $M = (K, \Sigma, \delta, s_0, F)$ y dos estados q_0 y $q_1 \in K$, decimos que q_0 y q_1 son *equivalentes* o *redundantes* ($q_0 \approx q_1$) si $(K, \Sigma, \delta, q_0, F) \approx (K, \Sigma, \delta, q_1, F)$.
- Una vez que sabemos que dos estados son equivalentes, entonces podemos eliminar uno de ellos. Pero, ¿y las flechas que entran y salen del estado eliminado?
  - Las flechas que salen del estado eliminado son eliminadas.
  - Las flechas que entraban al estado eliminado se redirigen al estado equivalente.

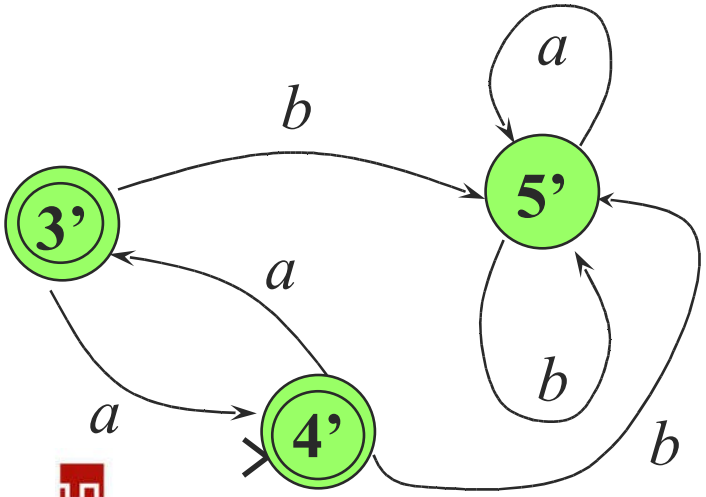
Estados equivalentes



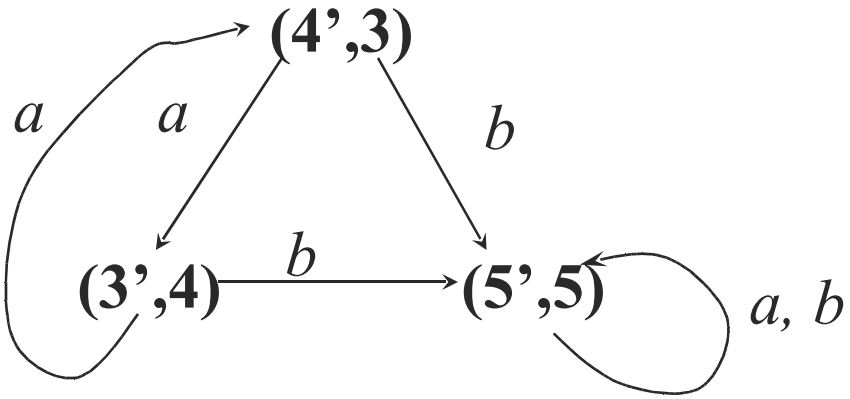
Si el AFD inicia con el estado 3 o con 4, ¿aceptará el mismo lenguaje?

Estados equivalentes





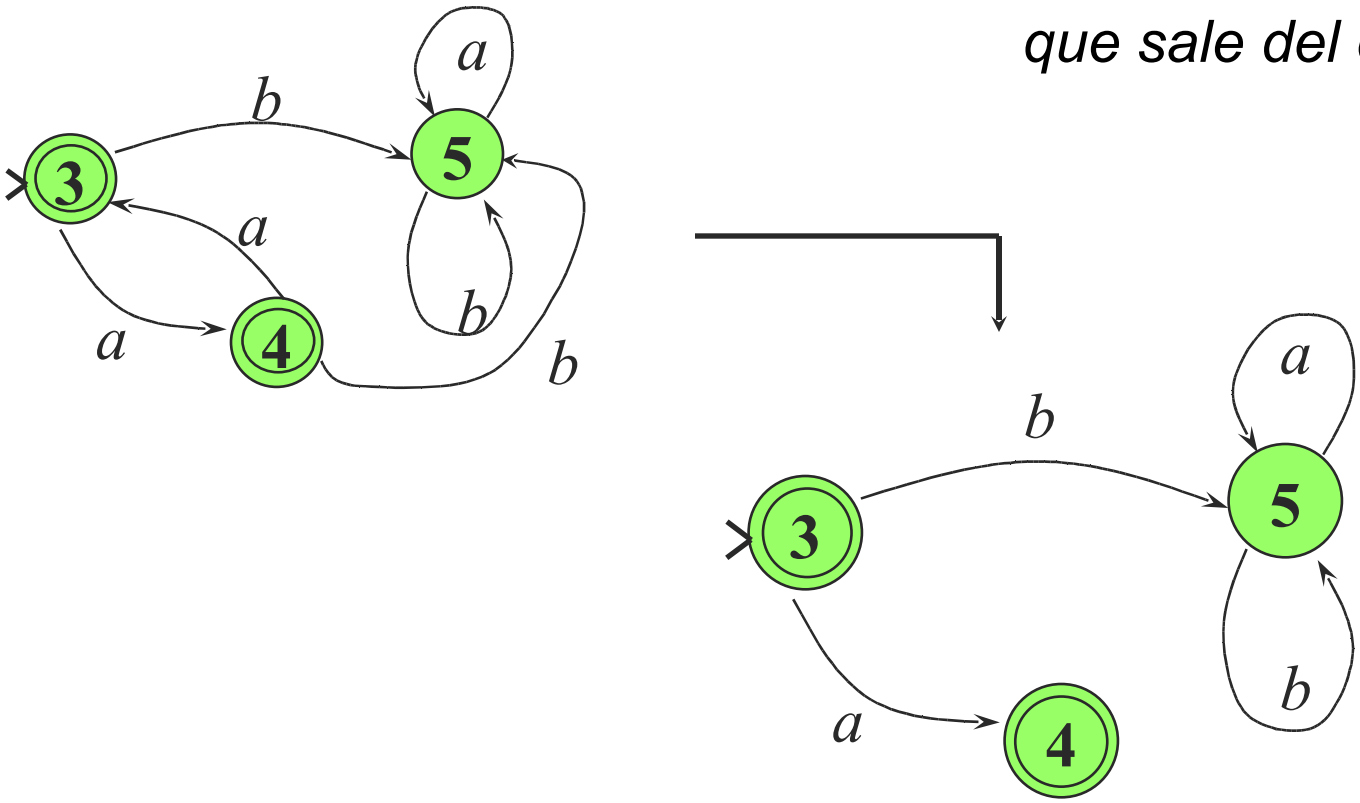
¿Son equivalentes?



Lo son, entonces se puede eliminar uno de ellos. Escogemos eliminar el estado 4'.

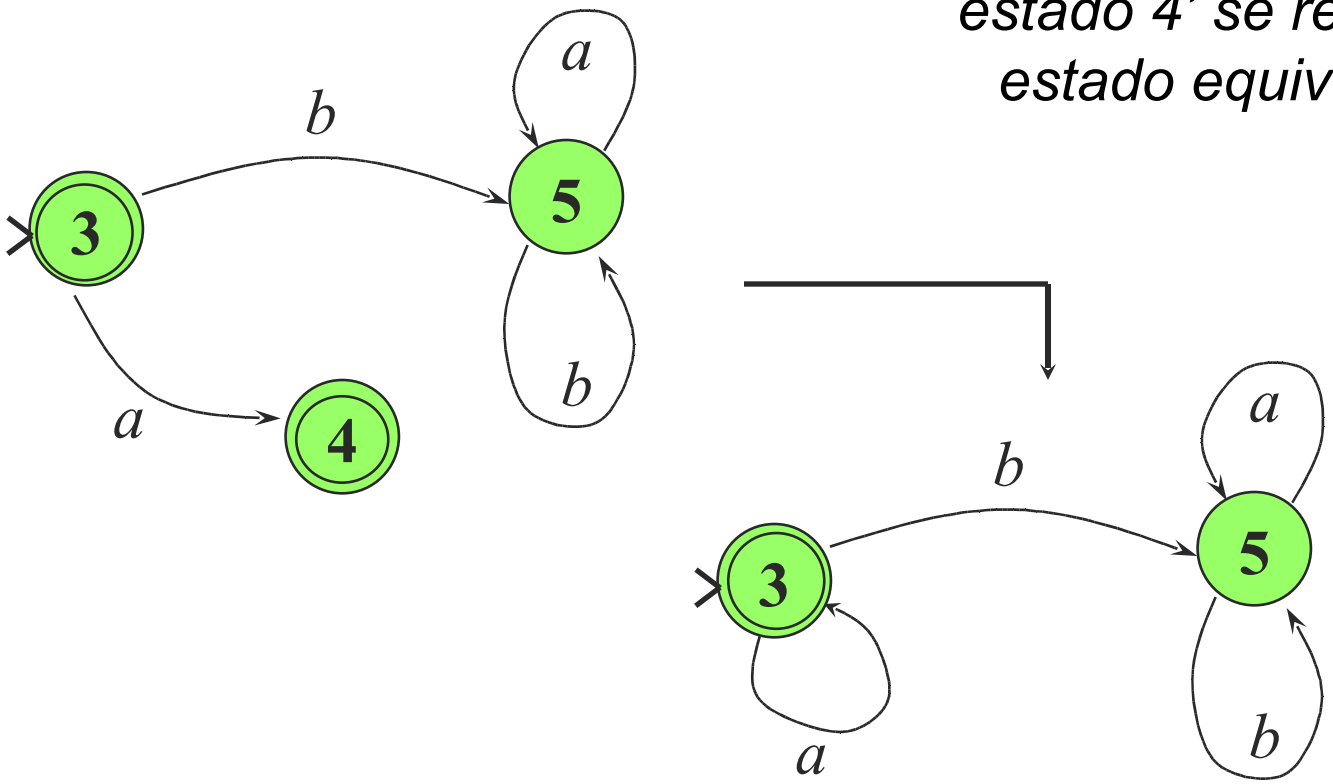
Borrar transiciones

Se borra toda transición que sale del estado 4'.



Redirigir transiciones

Toda transición que llega al estado 4' se redirige a su estado equivalente, 3.



AFD simplificado.

Algoritmo de minimización por estados equivalentes

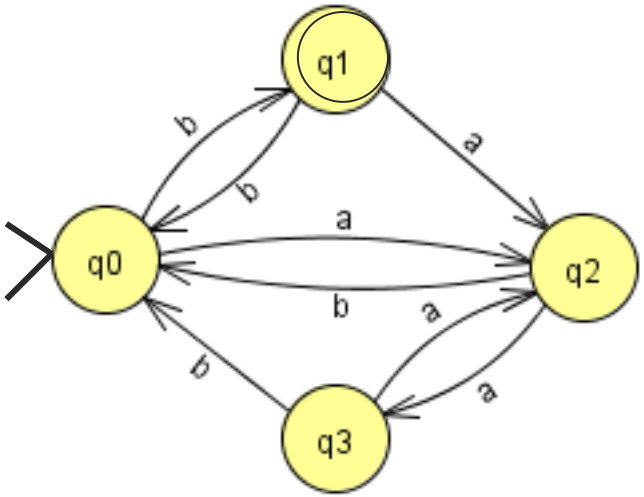
- Teorema. Al eliminar estados redundantes de un AFD se obtiene el único AFD mínimo que acepta el mismo lenguaje que el original.

- Algoritmo:
  Para cada par de estados (p, q) del autómata
  - Ver si son equivalentes.
  - En caso de que sí, entonces eliminar uno de ellos y volver a empezar con otros dos estados.

  Hasta que no haya estados que eliminar.

Ejercicio: Minimizar

- Minimizar el siguiente AFD usando el método de los estados equivalentes.

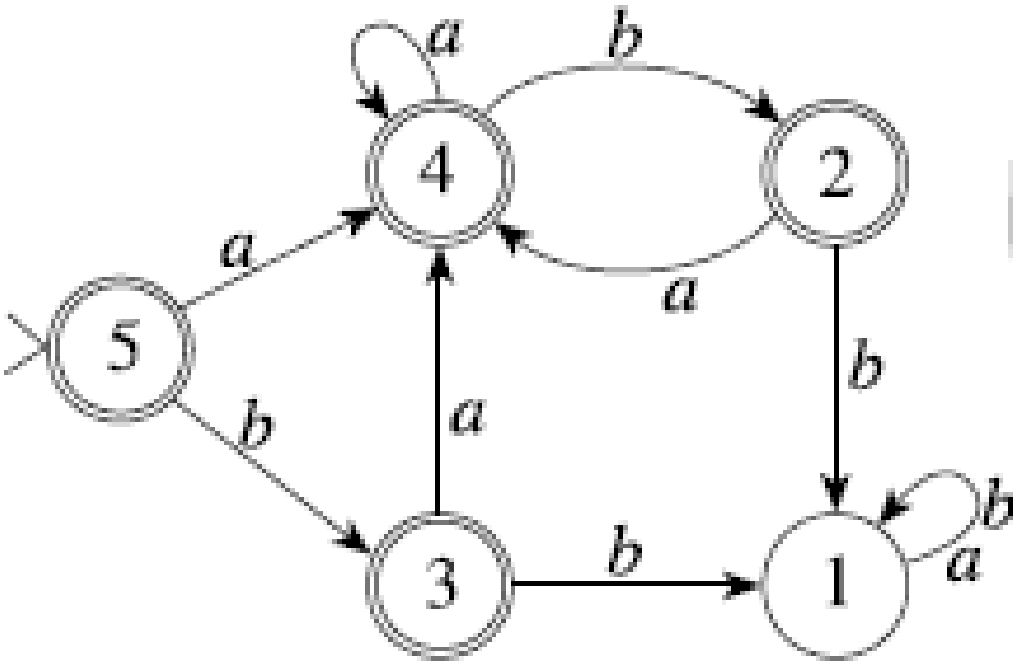


Obtención de AFD mínimo utilizando clases de equivalencia

Dado un AFD $M = (K, \Sigma, \delta, s_0, F)$, el procedimiento para simplificarlo es:

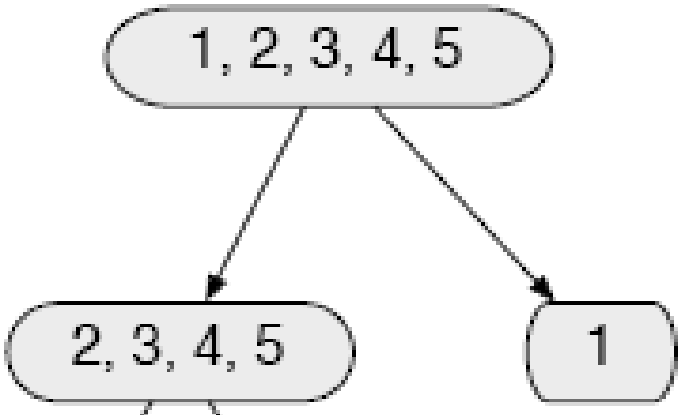
- Definimos dos clases de equivalencia, F y $K - F$.
- Para cada clase
  - Sea q un estado en la clase. Poner en una misma clase a todos los estados q' que tienen transiciones "iguales" a las de q, es decir, q y q' pertenecen a la misma clase si para cada símbolo $\sigma \in \Sigma$, $\delta(q', \sigma)$ "cae" en la misma clase que $\delta(q, \sigma)$. Ponemos en otra clase a los que tienen transiciones "distintas" a las de q.
  - Si todos los estados de la clase tienen transiciones iguales, entonces la clase no se divide y analizamos otra clase.
  - Continuar analizando clases hasta que ninguna se divida.

Ejemplo: Minimizar



Clase de equivalencia organizada en árbol

- Es conveniente organizar la información de las clases de equivalencia en árboles, en donde cada nodo contiene los estados de una clase de equivalencia.
- Inicialmente están todos los estados del AFD en una clase, como en la raíz del árbol. Inmediatamente se dividen en finales y en no finales.

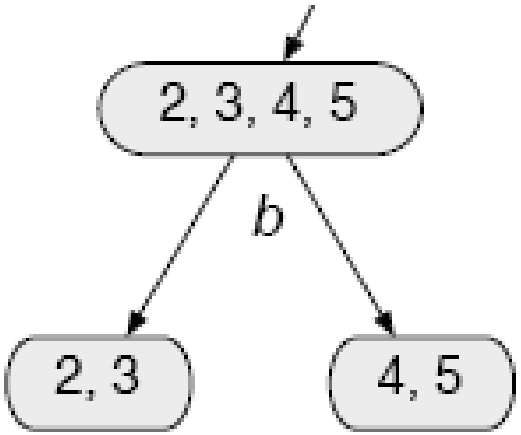


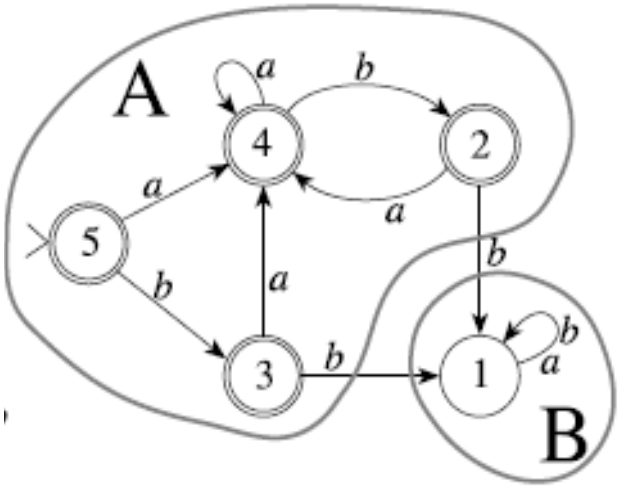
Todos los estados

Finales / No finales

Clase de equivalencia organizada en árbol

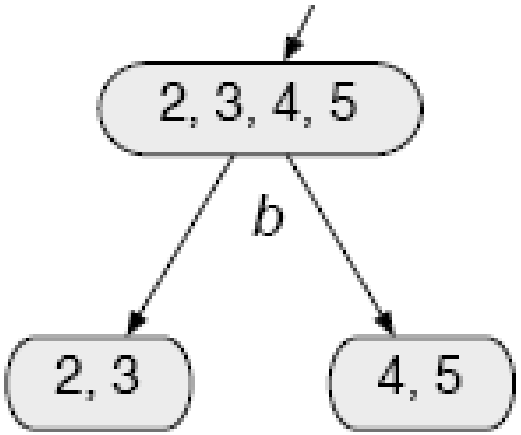
- Luego, para el nodo {2, 3, 4, 5} examinamos si las transiciones con los caracteres de entrada, en este caso *a* y *b*, llevan a las mismas clases y verificamos que en el caso de b los estados 2 y 3 van a un no final, mientras que 4 y 5 van a un final, por lo que ese nodo se divide en dos, como se aprecia en el tercer nivel de la figura.

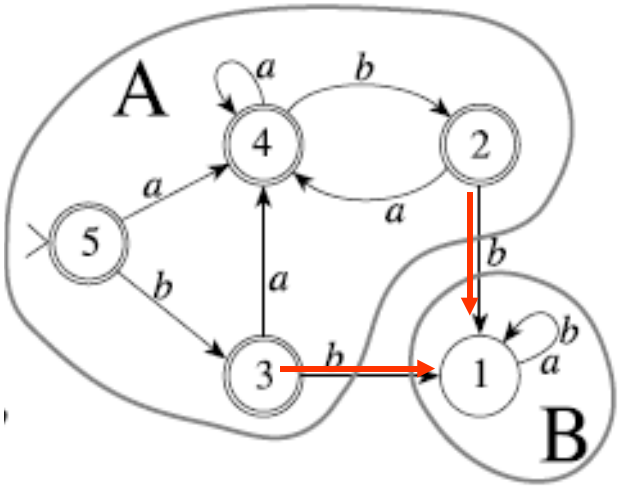




Clase de equivalencia organizada en árbol

- Luego, para el nodo {2, 3, 4, 5} examinamos si las transiciones con los caracteres de entrada, en este caso *a* y *b*, llevan a las mismas clases y verificamos que en el caso de b los estados 2 y 3 van a un no final, mientras que 4 y 5 van a un final, por lo que ese nodo se divide en dos, como se aprecia en el tercer nivel de la figura.

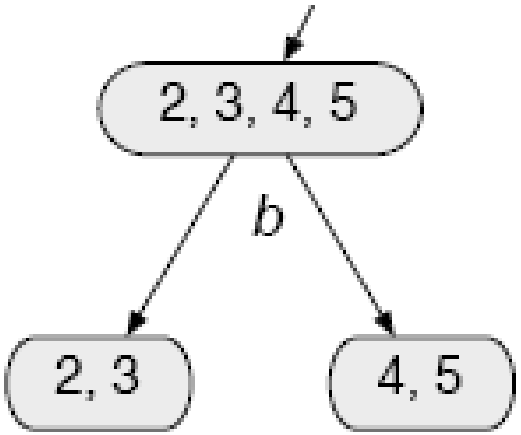


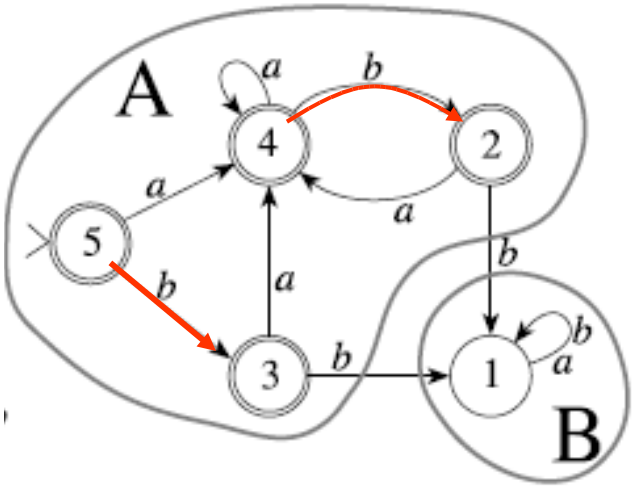


El estado 2 y 3 con la letra *b* caen en la clase B (no final)

Clase de equivalencia organizada en árbol

- Luego, para el nodo {2, 3, 4, 5} examinamos si las transiciones con los caracteres de entrada, en este caso *a* y *b*, llevan a las mismas clases y verificamos que en el caso de b los estados 2 y 3 van a un no final, mientras que 4 y 5 van a un final, por lo que ese nodo se divide en dos, como se aprecia en el tercer nivel de la figura.





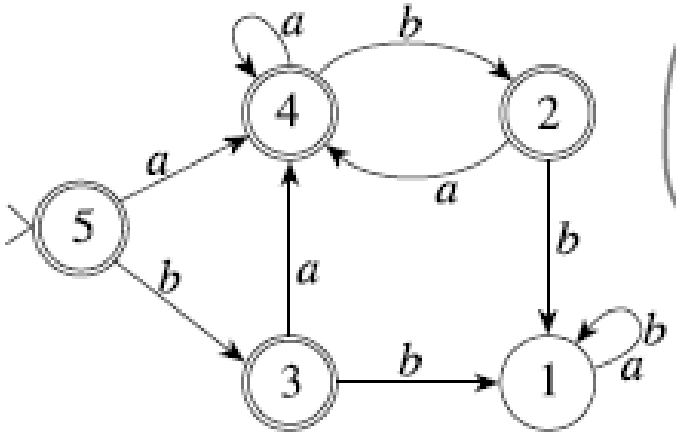
El estado 4 y 5 con la letra *b* caen en A (estados finales)

Clase de equivalencia organizada en árbol

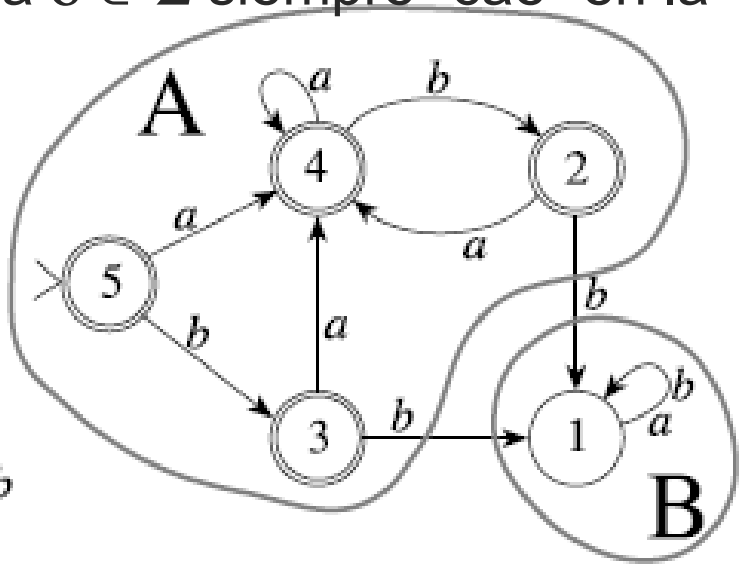
- Ahí también se puede apreciar un símbolo b bajo el nodo {2, 3, 4, 5}, indicando a causa de qué carácter la clase de equivalencia se dividió.
- Examinando las transiciones en las clases de equivalencia que quedan en las hojas del árbol, vemos que ya no hay razón para dividirlas más.
- Finalmente, las clases de equivalencia resultantes son {1}, {2, 3} y {4,5}, que corresponden a los 3 estados que tendrá el AFD minimizado.

Graficamente

- Clase A: Terminales, Clase B: No terminales.
- Note que en B para cada $\sigma \in \Sigma$ siempre “cae” en la misma clase.

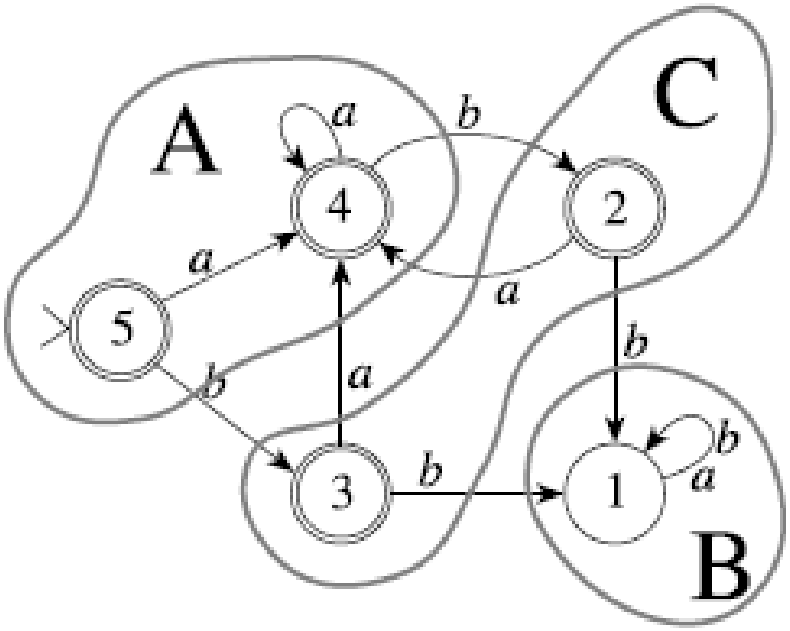


(a) AFD a simplificar

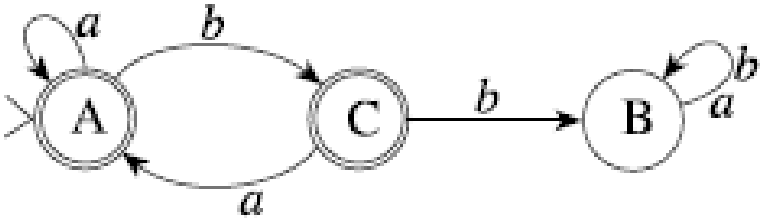


(b) Clases iniciales

- Partición de Clase A en A y C. Nótese que ahora todas las transiciones de A, B y C caen en las mismas clases.



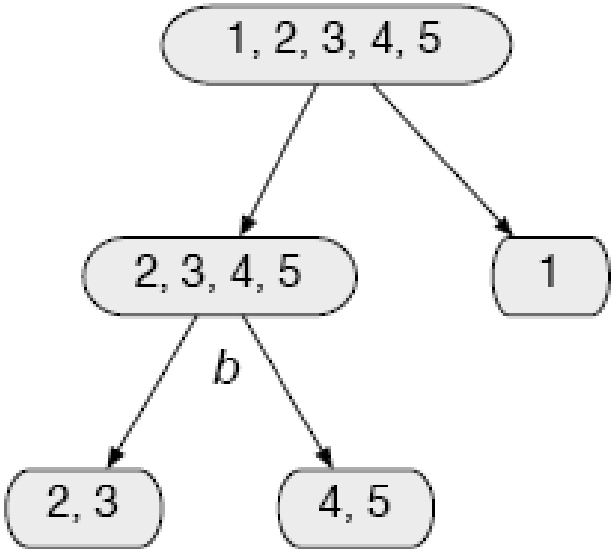
(c) Clases al final



(d) AFD simplificado

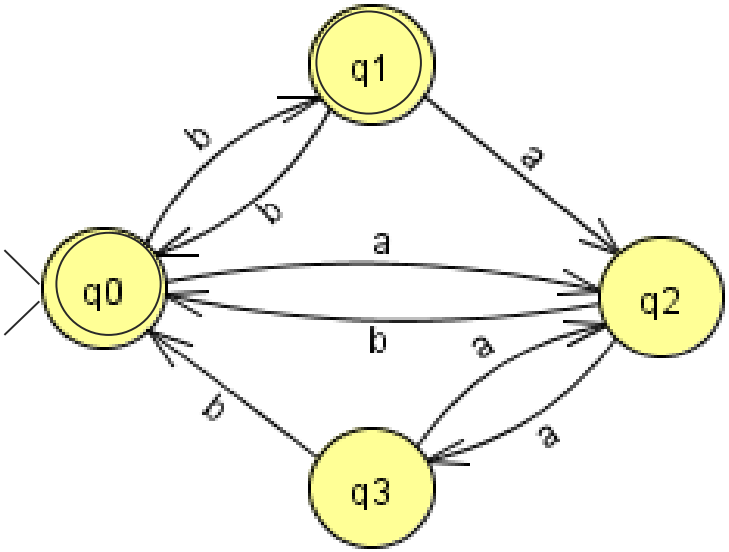
Clase de equivalencia organizada en árbol

- Podemos ver el árbol de partición de clase final:



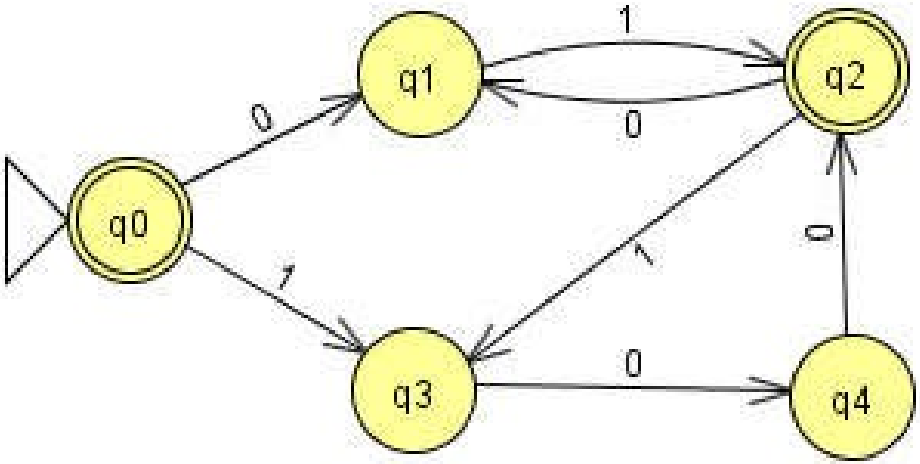
Ejercicio: Minimizar

- Minimizar el siguiente AFD usando el método de las clases de equivalencia.



Solución

Ejercicio: simplificar



Solución