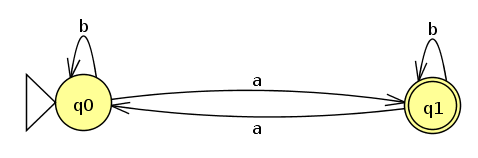
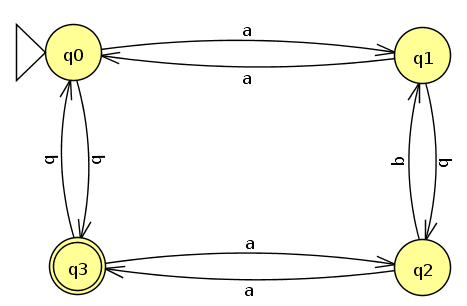
Alfabeto {a,b}

1 - stringhe con un numero dispari di a

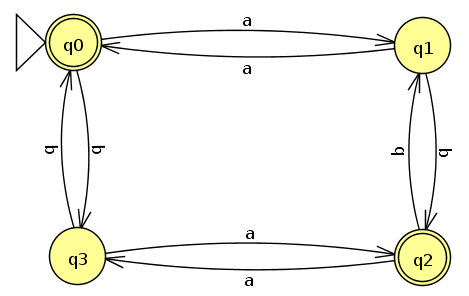


2 - stringhe con un numero dispari di b e pari di a

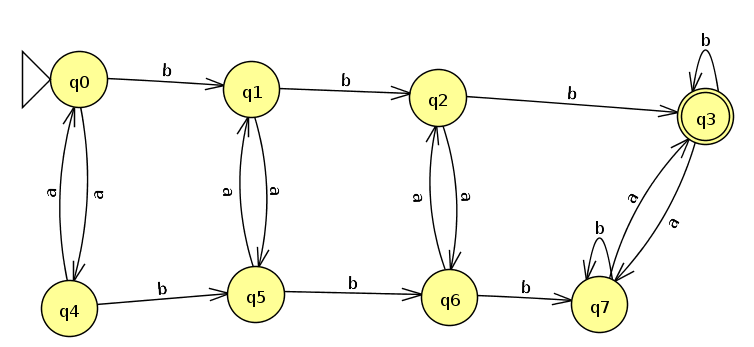


3 - stringhe con un numero in cui le a e le b risultano essere entrambe

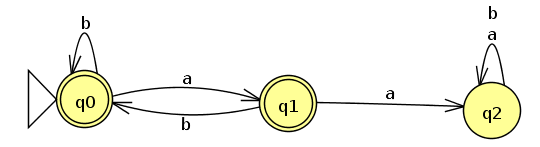
in numero pari, oppure entrambe in numero dispari



4 - stringhe con un numero pari di a ed almeno 3 b

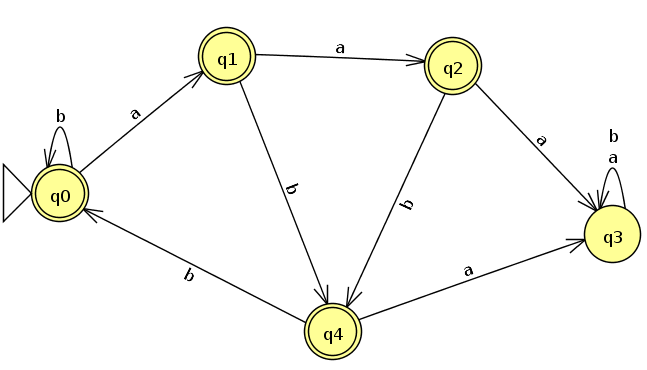


5 - stringhe che non contengono la sottostringa aa



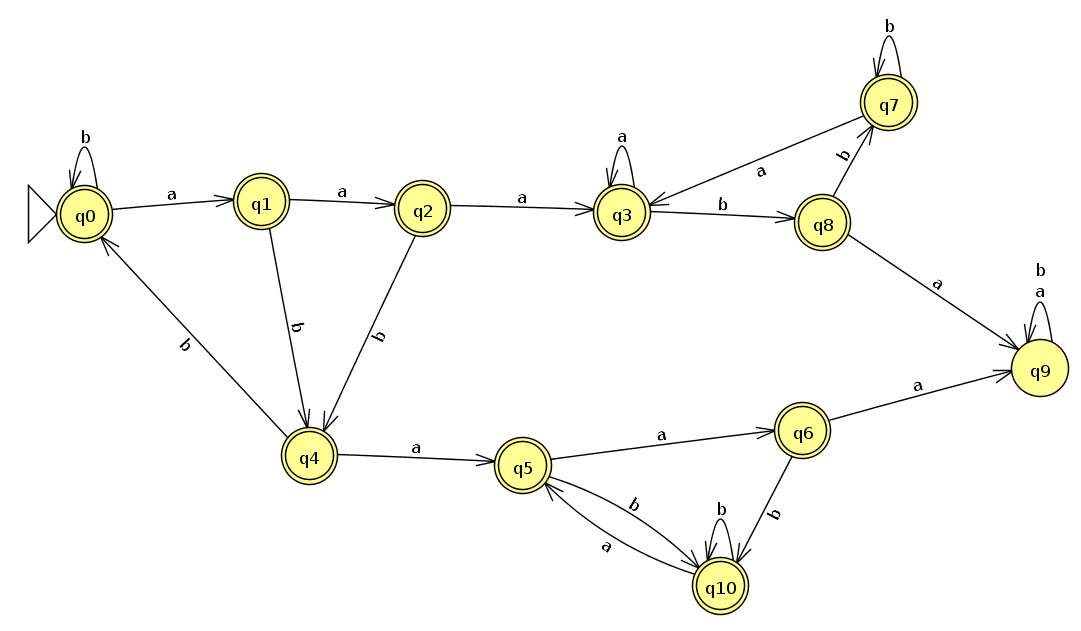
6 - stringhe che non contengono la sottostringa aaa e non

contengono la sottostringa aba



8- stringhe che non contengono sia la sottostringa aaa

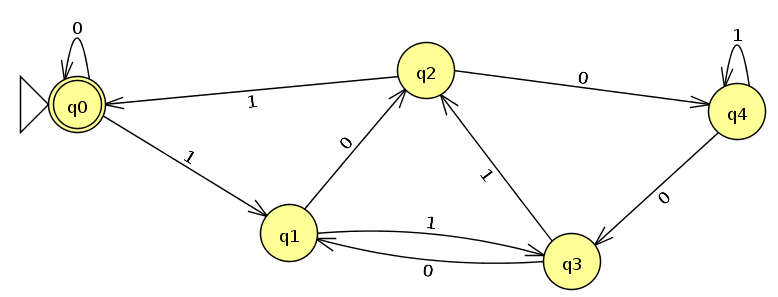
che la sottostringa aba (ma possono contenerne una delle due)



\*\* Consideriamo l'alfabeto {0,1}

8 - stringhe sull'alfabeto {0,1} che interpretate come numero binario

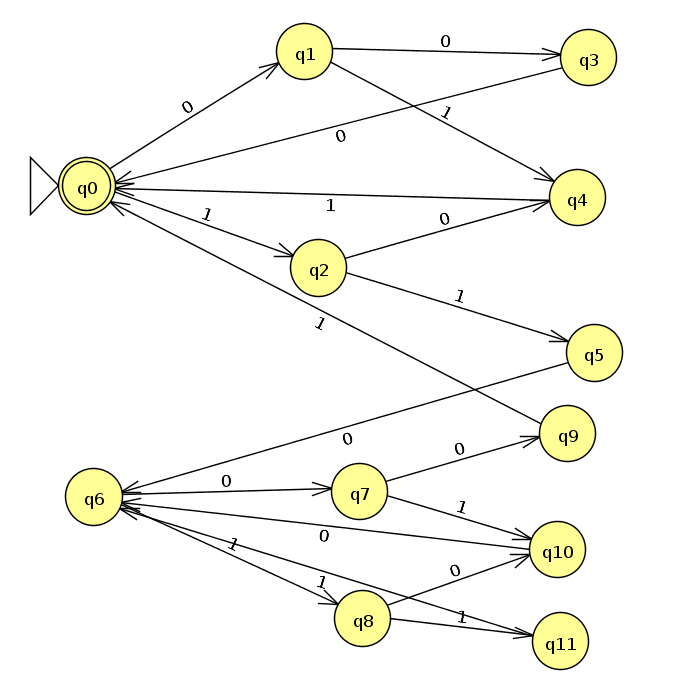
risultano un multiplo di 5



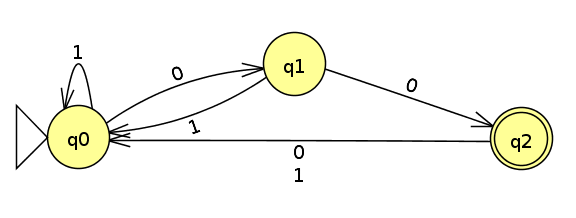
9 - Realizzare un DFA che controlla la correttezza delle somme binarie:

data la stringa: a\_0 b\_0 c\_0 a\_1 b\_1 c\_1 ... a\_n b\_n c\_n controlla

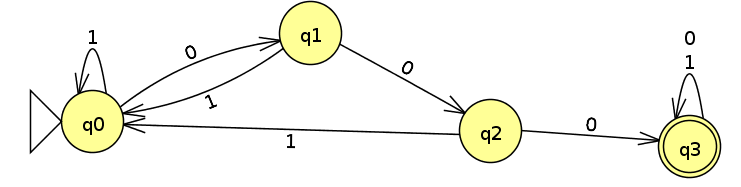
se a\_n...a\_1 a\_0 + b\_n...b\_1 b\_0 = c\_n...c\_1 c\_0



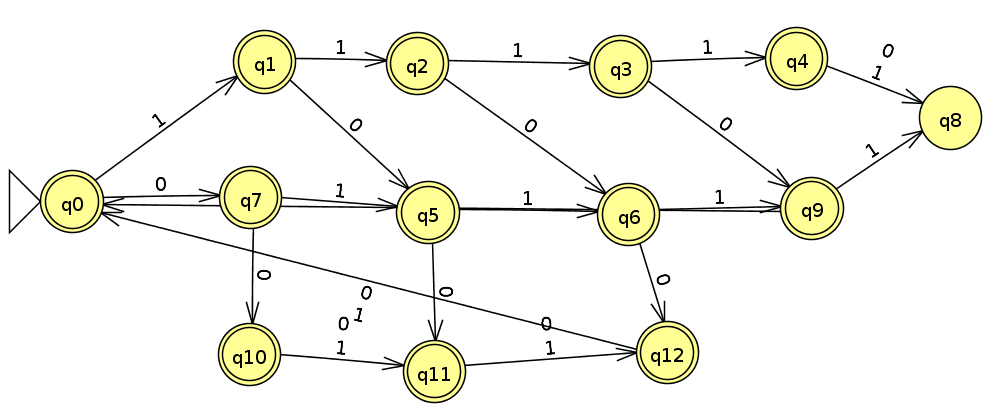
Insieme delle stringhe che finiscono con 00



Insieme delle stringhe che ha 3 zeri consecutivi, non necessariamente alla fine.

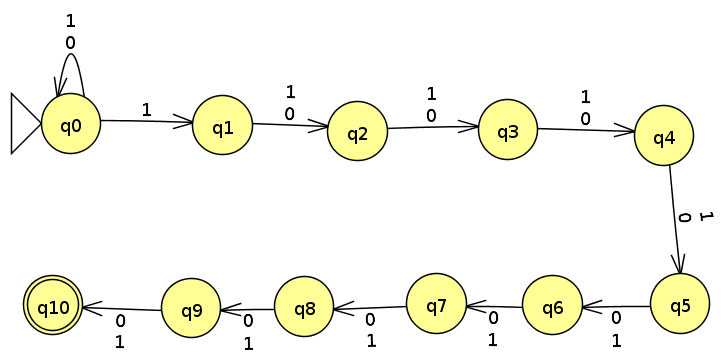


L'insieme delle stringhe che, ogni 5 caratteri, hanno almeno 2 zeri:



il decimo simbolo 10th a partire dalla fine destra è 1:

(prima faccio l'NFA):



poi lo converto in DFA:

NB

Se si applica la conversione da NFA a DFA si trovano tantissimi sottoinsiemi raggiungibili, è quindi impraticabile fare la tabella.

L'unica cosa che si può fare è definire il DFA con i 5 parametri:

A = (Q, alfabeto, funzione di transizione, stato iniziale, stati finali)

L'insieme degli stati è formato dagli stati che si ricordano gli ultimi 10 bit letti, quindi sono 2^10 = 1024.

L'insieme degli stati finali è formato dagli stati che hanno come primo bit (il decimo prima della fine) = 1.

La funzione di transizione shifta i bit a sinistra:

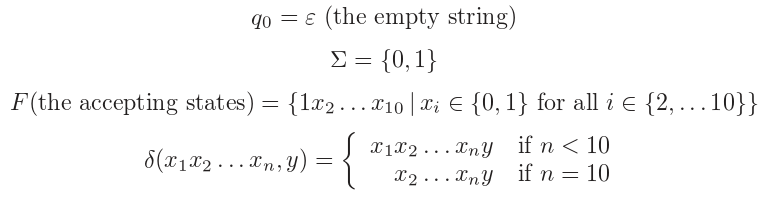
se n < 10, => aggiunge l'input a nel bit a destra

se n = 10 => aggiunge l'input a nel bit a destra e cancella il bit più a sinistra

(n indica l'ultimo stato della sequenza di bit x1 x2 … xn, 0 <= n <= 10)

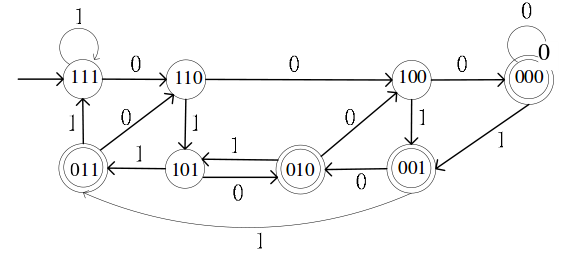
Definizione:





Un esempio è riportato sotto, che considera solo il terzultimo bit = 1:

L'insieme delle stringhe tali che il 3 bit letto da destra è 0:



ogni stato indica gli ultimi 3 bit letti, si parte da 111, così lo 0 arriva in terzultima posizione dopo che sono stati letti almeno 3 caratteri.

Gli stati finali sono tutti quelli che hanno lo 0 in terzultima posizione.

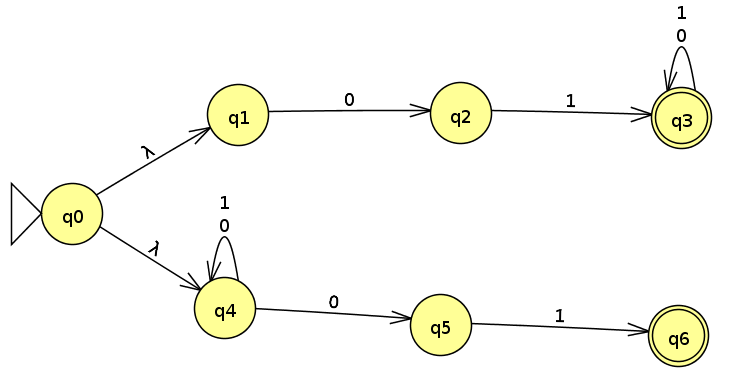
NB:

gli stati in tutto sono 2^3 = 8.

Se era la decima posizione da considerare, gli stati erano 2^10 = 1024, quindi si fa l'NFA e poi lo si converte in DFA.

L'insieme delle stringhe che iniziano per 01, finiscono per 01, o entrambe.

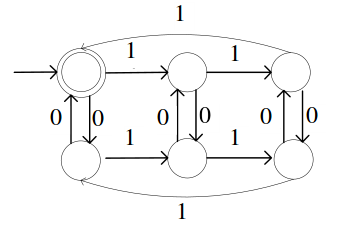
Faccio prima l'epsilon\_NFA (E\_NFA), che rappresenta l'unione dei 2 linguaggi:



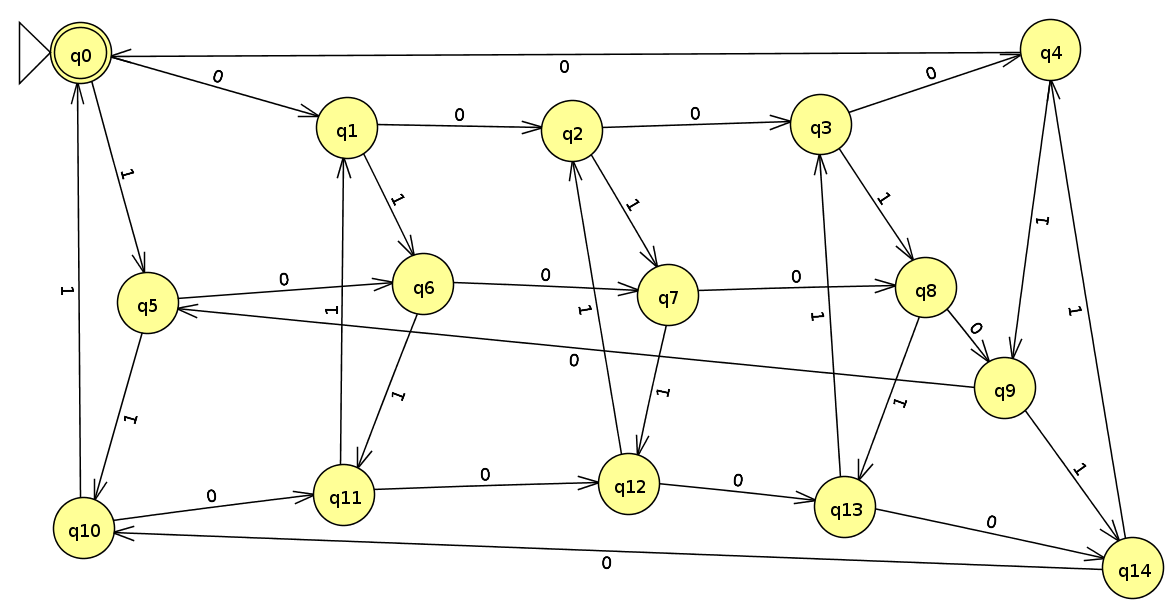
poi lo converto in DFA:

TODO

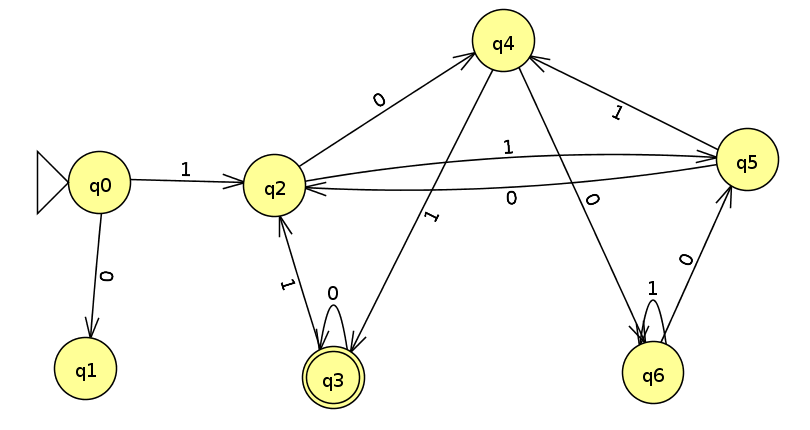
L'insieme delle stringhe che hanno numero di zeri divisibile per 2 e il numero di 1 divisibile per 3:



L'insieme delle stringhe che hanno un numero di zeri divisibile per 5 e un numero di 1 divisibile per 3:



L'insieme delle stringhe che iniziano per 1 e che interpretate come numero binario sono un multiplo di 5:



Mi servono 2 stati in più oltre ai 5 che ricordano il resto:

lo stato iniziale che fa partire il DFA vero e proprio già con resto 1, se il primo bit è 1, o non accetta mai se il primo bit è 0.