

Problemi di decisione

Si delinei un algoritmo che decide se due linguaggi regolari su Σ hanno una stringa in comune.

Si delinei un algoritmo che decide se un DFA accetta un linguaggio infinito

Si delinei un algoritmo che, presi in input due DFA A e B decide se $L(A)$ è contenuto, anche non propriamente, in $L(B)$.

Si delinei un algoritmo che decide se due DFA sono equivalenti.

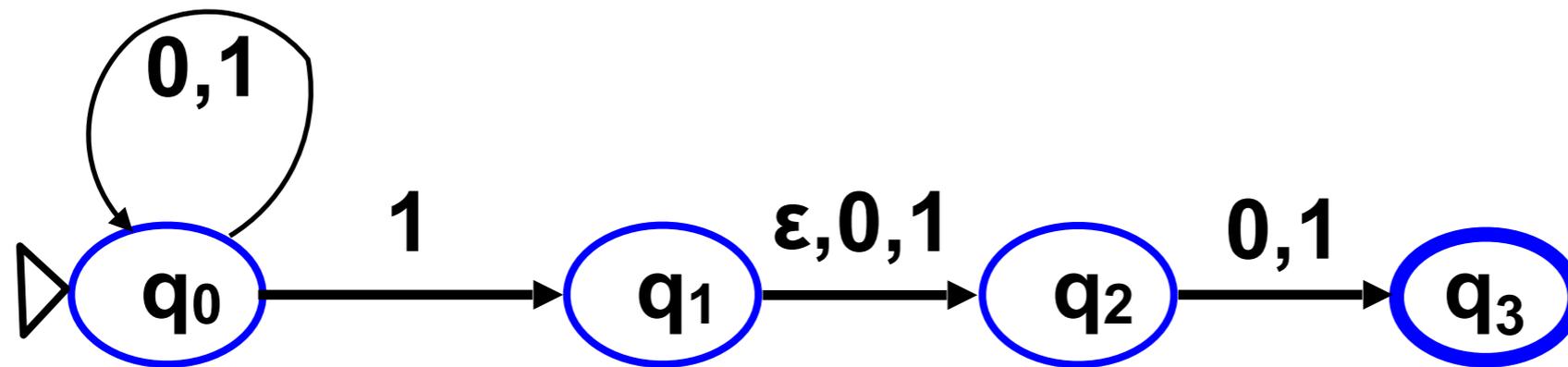
Si delinei un algoritmo che decide se $L = \Sigma^*$, per un linguaggio regolare L.

Esercizio Pumping lemma

Si spieghi dove fallisce la prova di non regolarità, basata sul pumping lemma, per il linguaggio $L = \{00, 11\}^*$

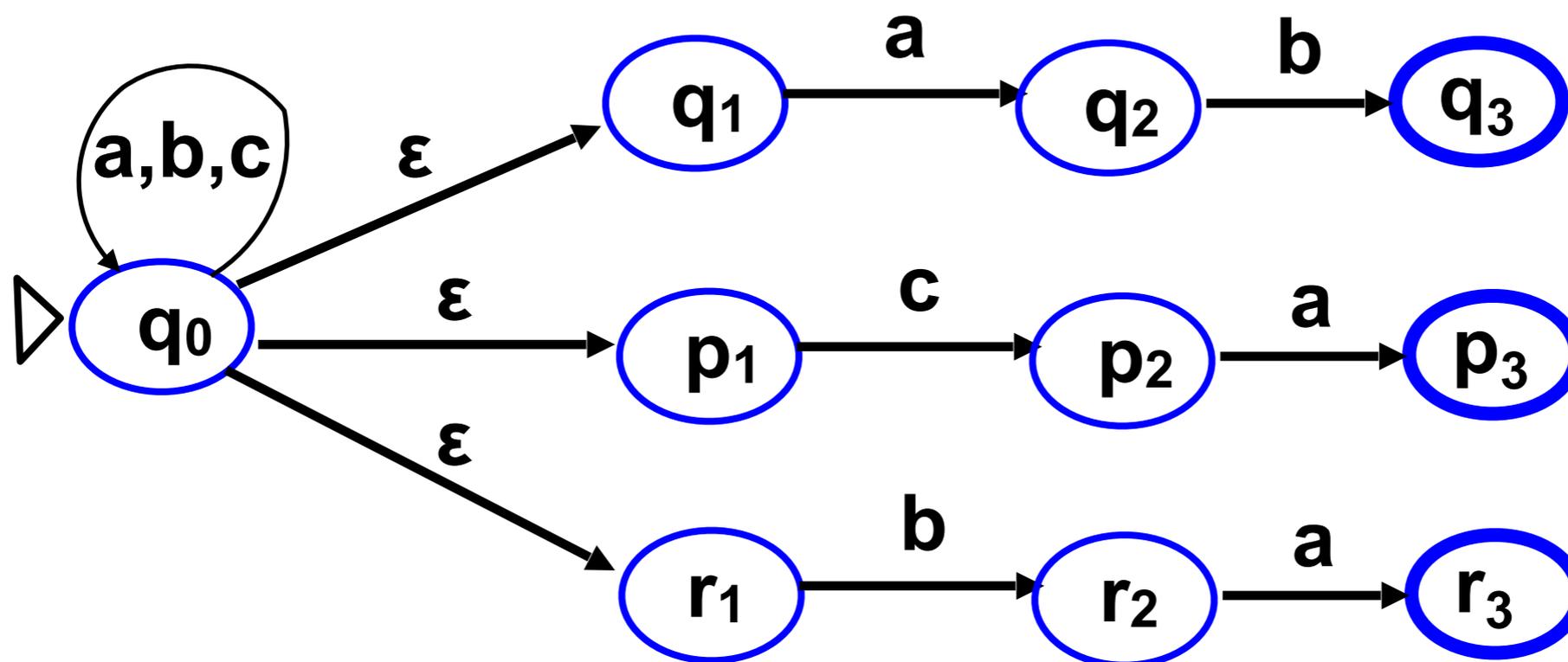
Esercizio NFA

Si costruisca un NFA che accetta tutte le stringhe in $\{0,1\}^*$ che contengono un 1 o nella penultima o nella terz'ultima posizione.

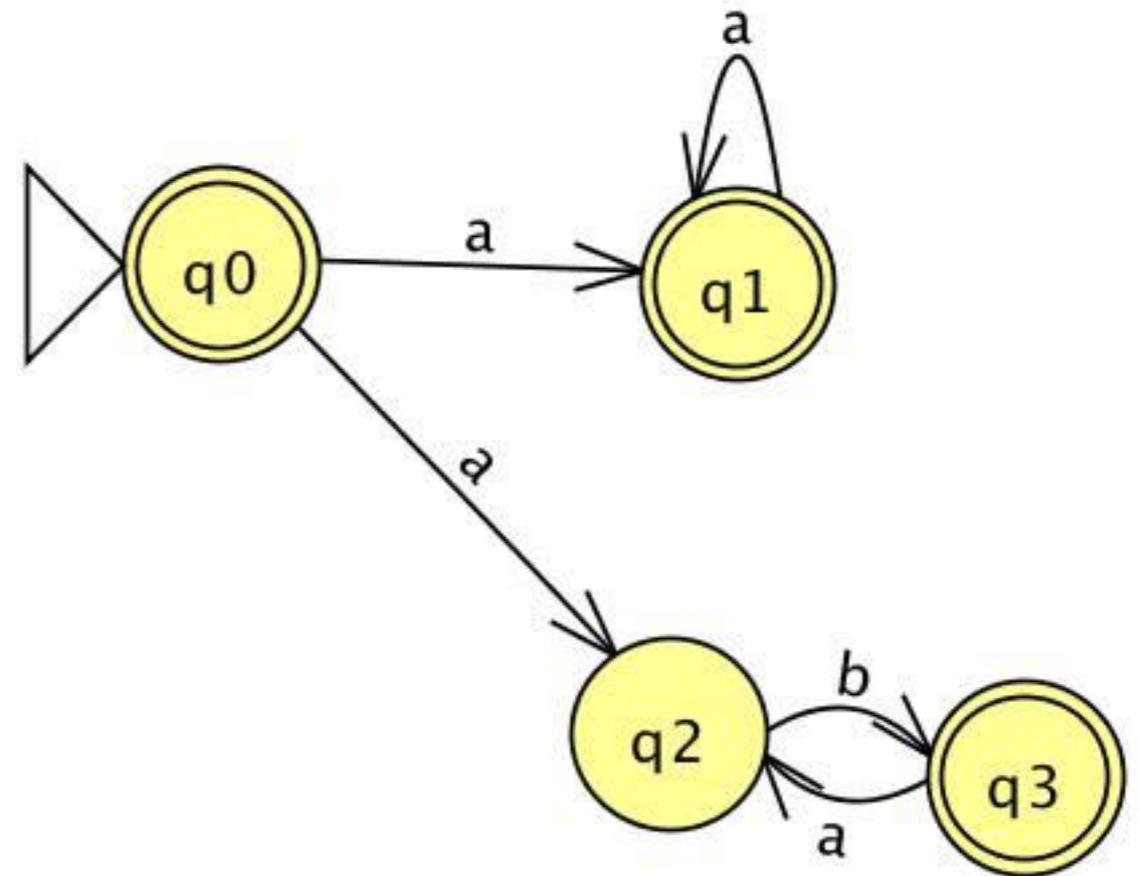
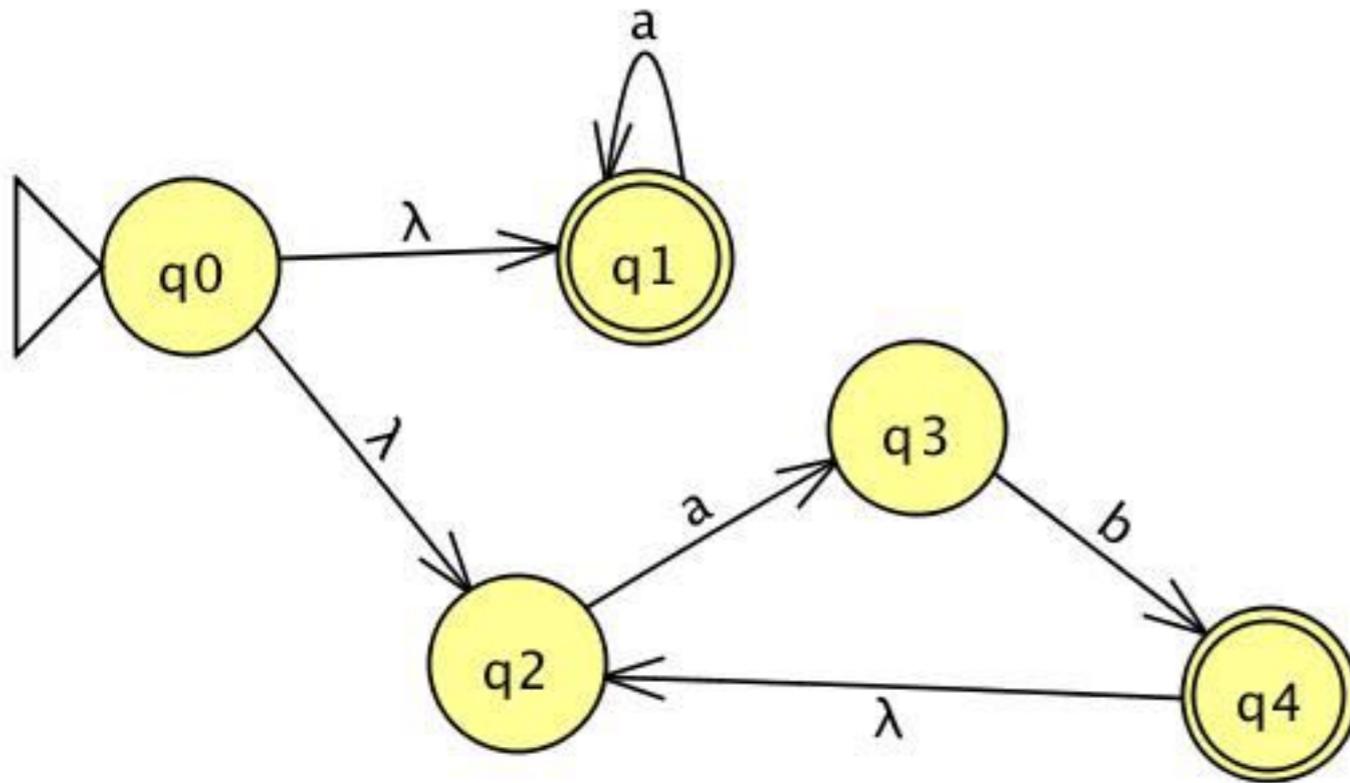


Esercizio NFA

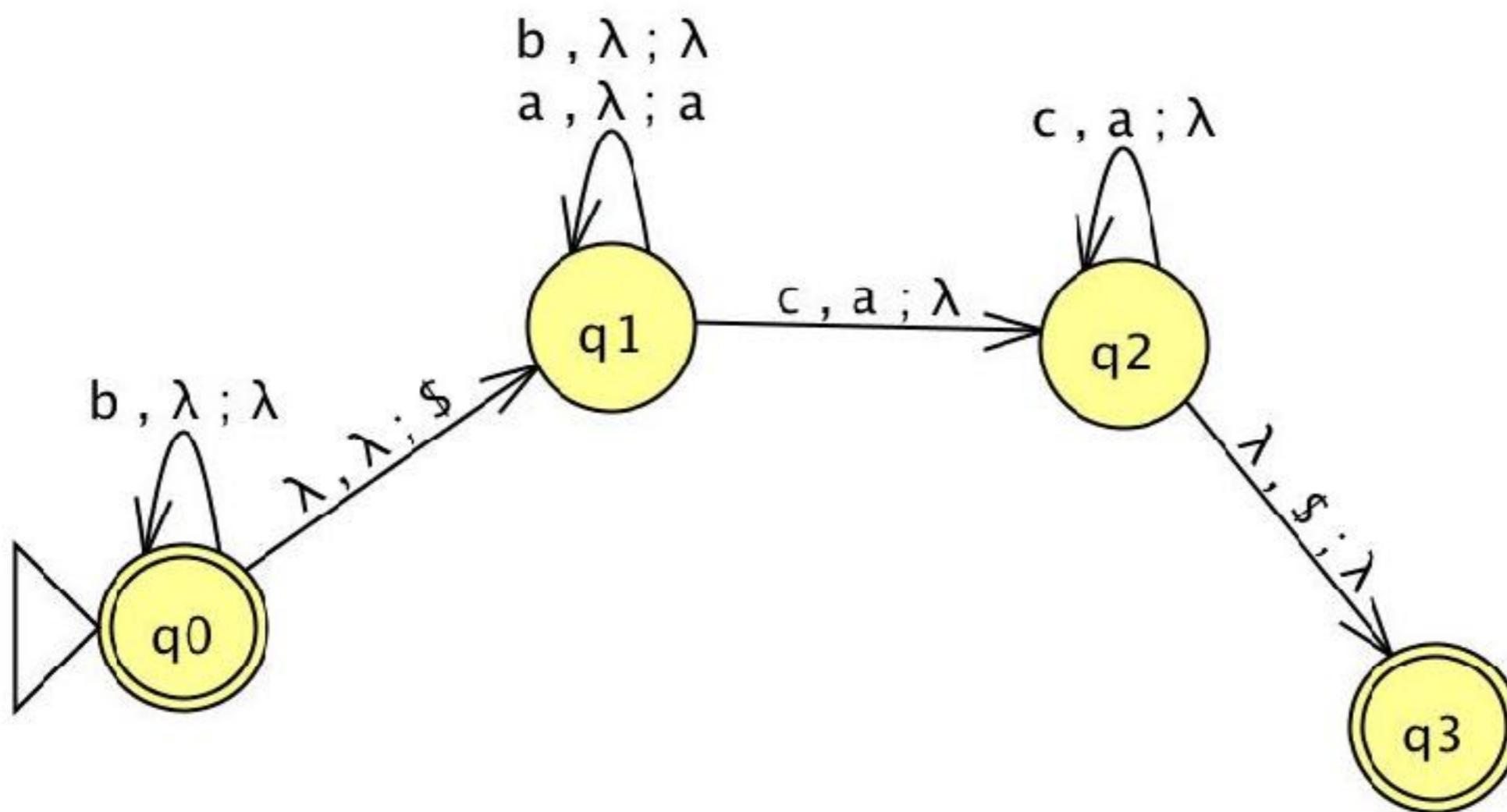
Si costruisca un NFA che accetta tutte le stringhe in $\{a,b,c\}^*$ che terminano con ab,ca oppure ba .



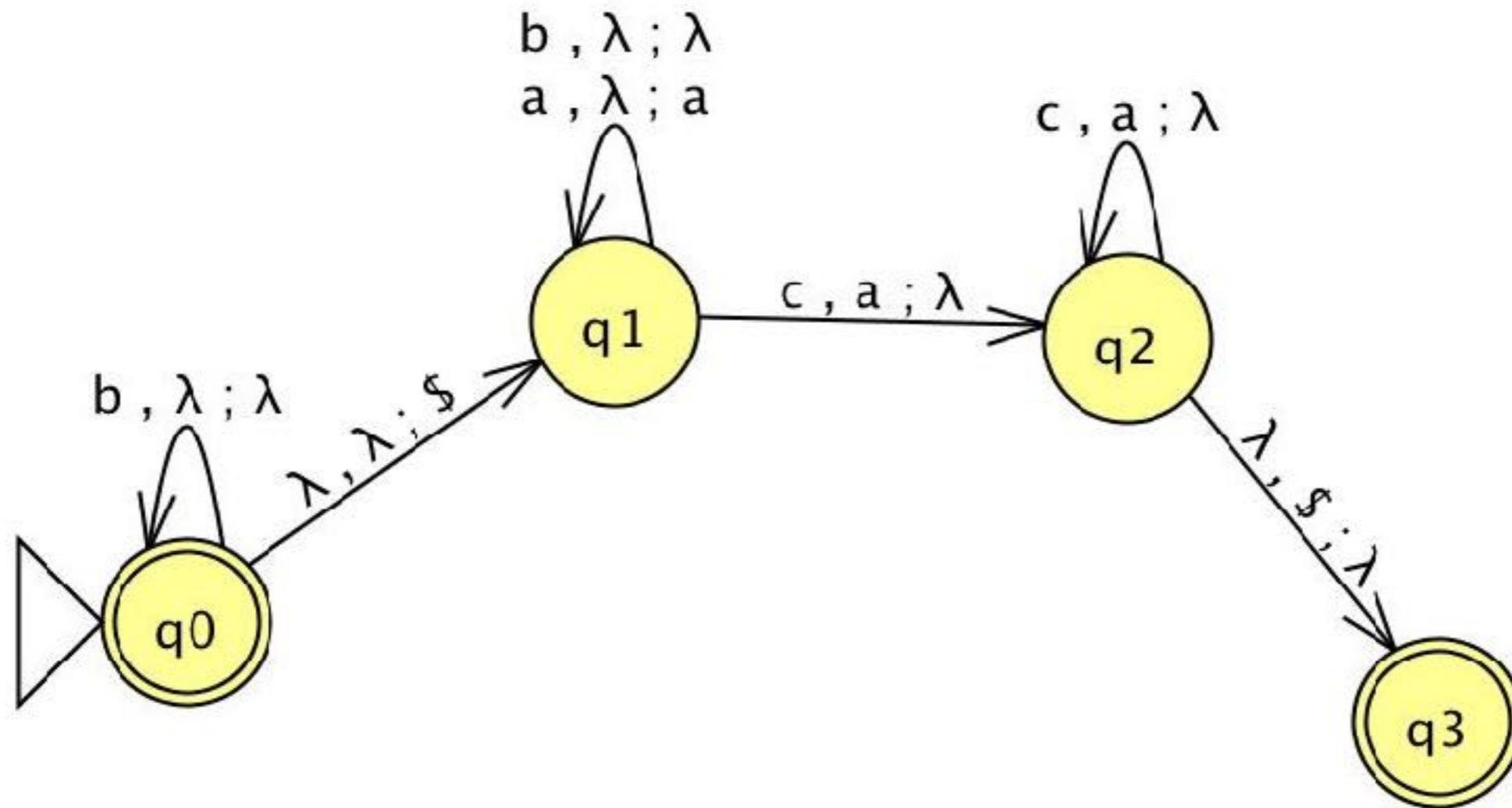
Si costruisca un NFA che accetta il linguaggio delle parole sull'alfabeto $\{a,b\}$ del tipo a^n con $n \geq 0$ o del tipo $(ab)^n$, per $n \geq 1$.
Ne diamo due versioni:



Si costruisca un PDA che riconosce il linguaggio $L = \{wc^k \mid w \in \{a,b\}^* \text{ e } k = n_a(w)\}$ cioè il linguaggio in cui il numero delle c è pari al numero delle occorrenze di a in w. Si spieghi il significato di ogni stato e la ragione delle transizioni.



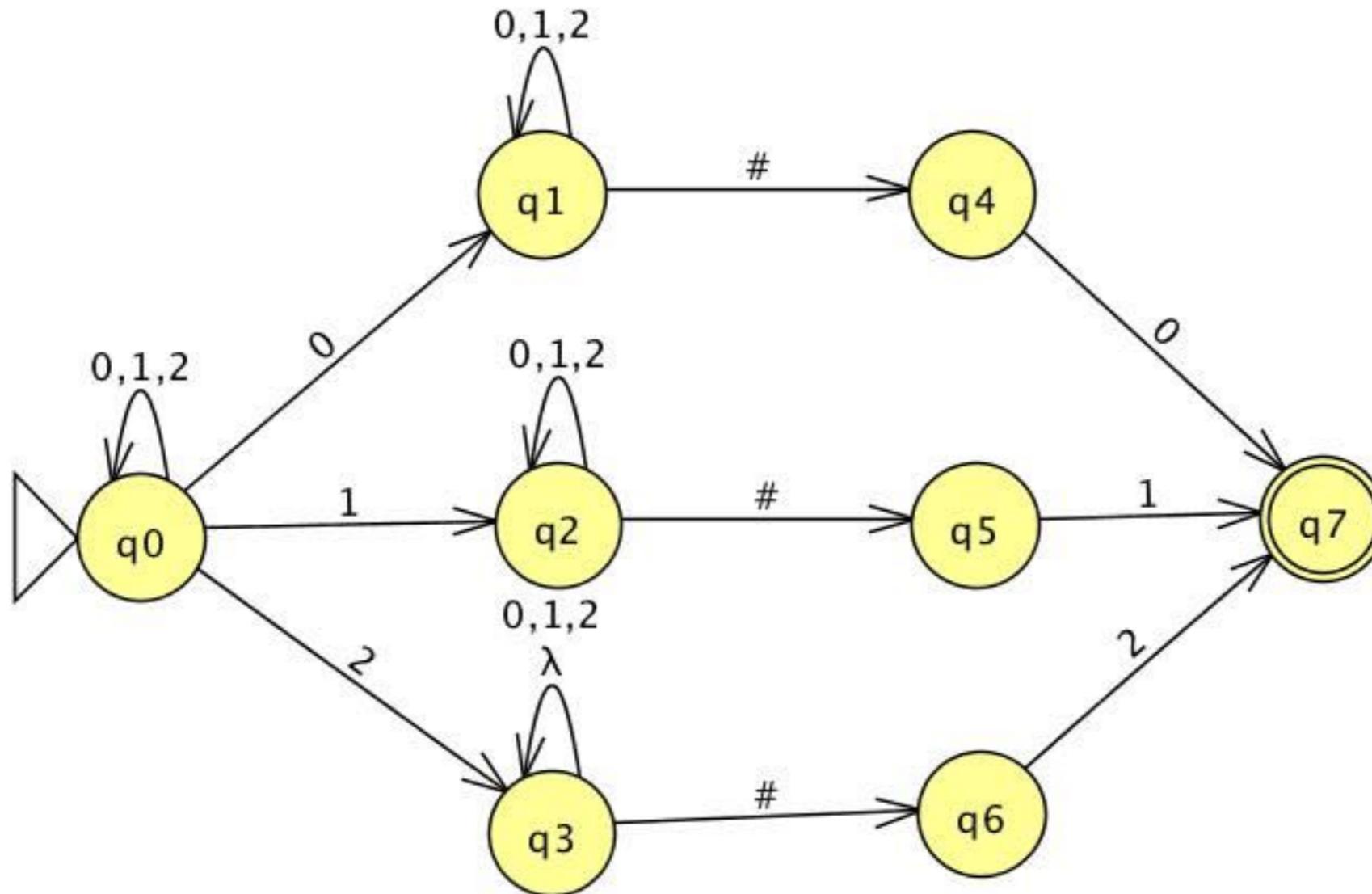
$$L = \{wc^k \mid w \in \{a,b\}^* \text{ e } k = n_a(w)\}$$



Nello stato q_0 si riconosce b^* .

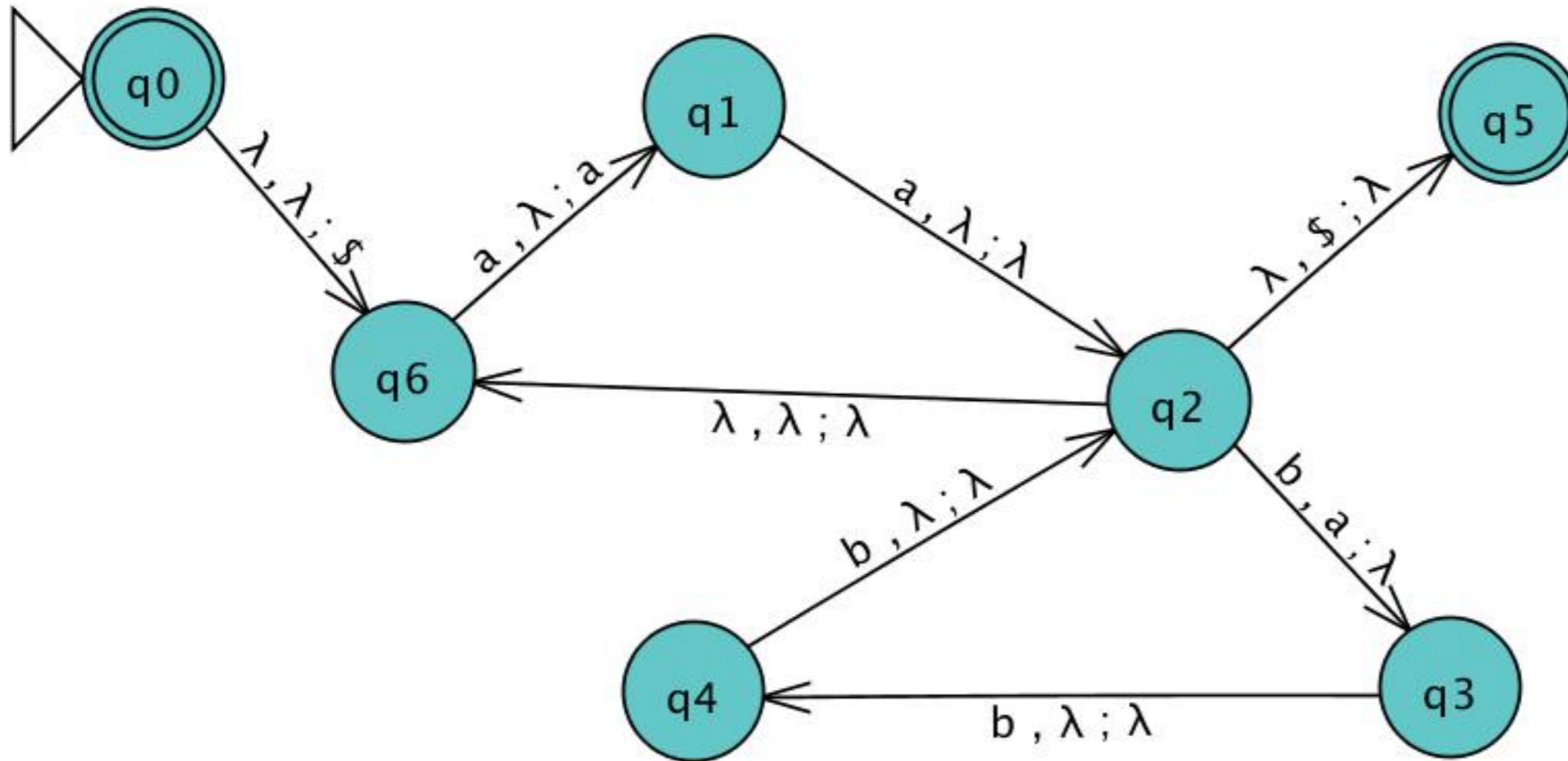
Poi, una volta inizializzata la pila, nello stato q_1 si impilano le a che vengono lette mentre eventuali b presenti nella parola sono lette senza operare sulla pila. Lo stato q_2 è quello in cui per ogni c letta viene eliminata una a dalla pila.

Si costruisca un NFA che accetta il linguaggio $L = \{w\#c \mid w \text{ è in } \{0,1,2\}^* \text{ e } c \text{ in } \{0,1,2\} \text{ e } c \text{ occorre in } w\}$ quindi per esempio $1221\#0$ non è nel linguaggio mentre $1221\#1$ e $1221\#2$ sì.

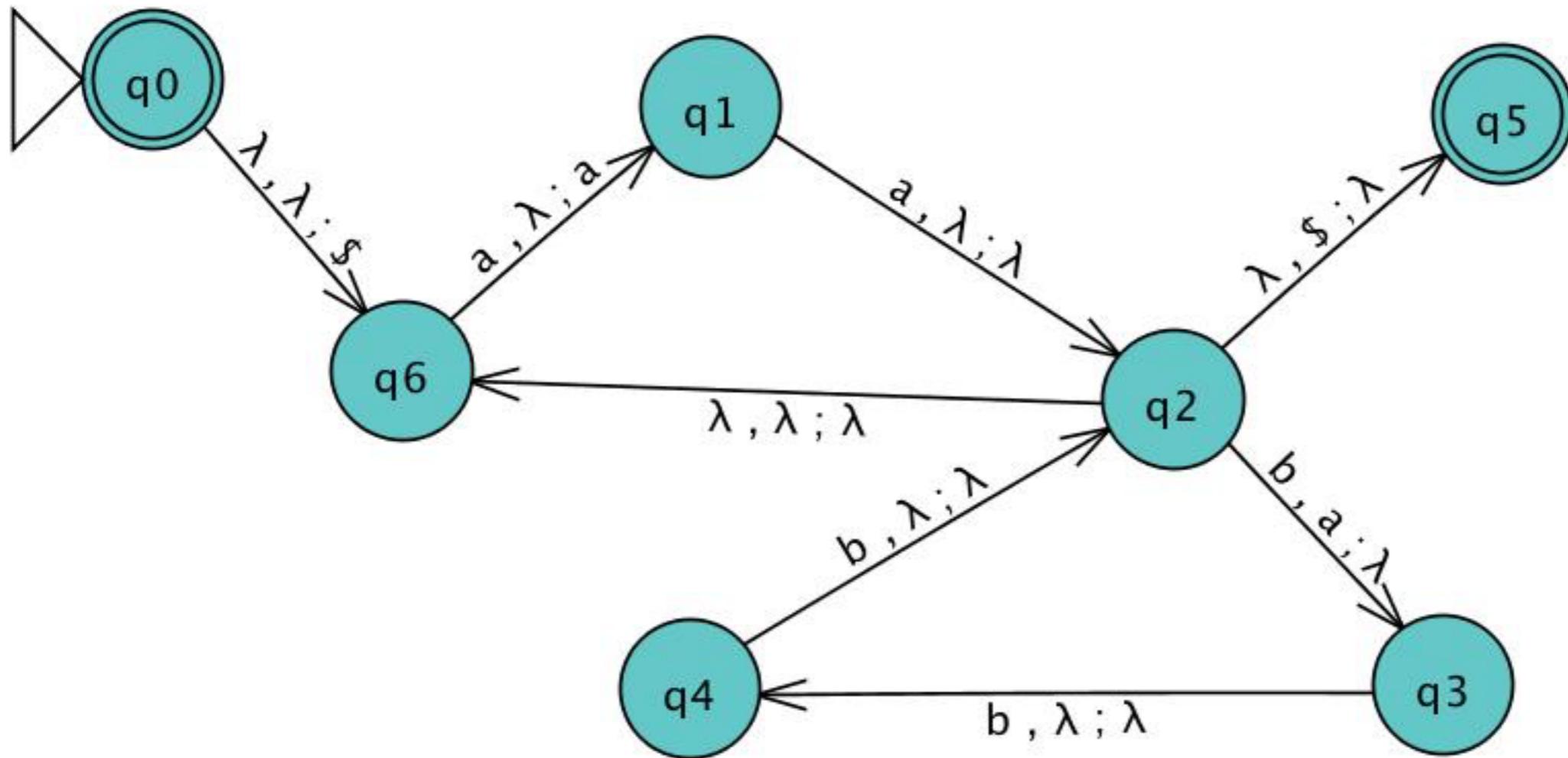


Poiché non possiamo determinare la posizione dell'occorrenza di c in w , usiamo gli stati per ricordare quale occorrenza isoliamo nella parola. In q_1 assumiamo che l'occorrenza è 0 e di conseguenza il cammino porta a uno stato finale solo se, dopo aver letto eventualmente il resto della parola in input, dopo il diesis si legge uno 0 . Gli altri casi sono analoghi

Si costruisca un PDA che riconosce il linguaggio $L = \{a^{2n}b^{3n} \mid n \geq 0\}$
Si spieghi il significato di ogni stato e la ragione delle transizioni.



$$L = \{a^{2n}b^{3n} \mid n \geq 0\}$$

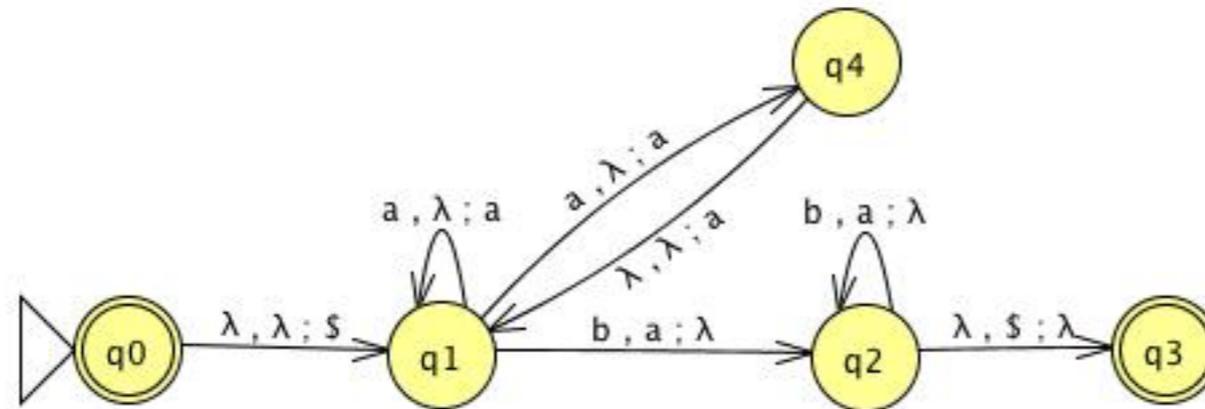


Nello stato q_0 si accetta la parola vuota.

L'esecuzione del ciclo q_6, q_1, q_2 fa sì che venga impilata una a ogni due a lette.

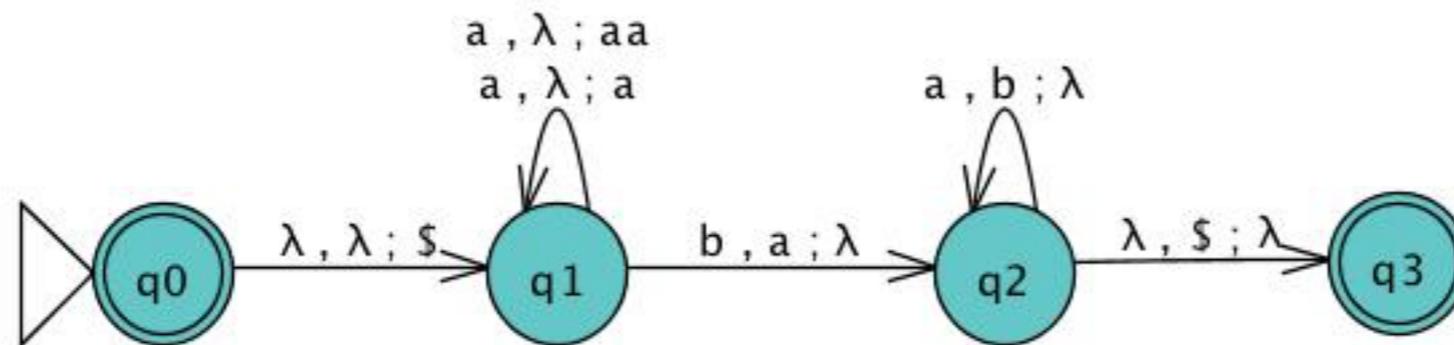
L'esecuzione del ciclo q_2, q_3, q_4 invece fa sì che venga estratta una a dalla pila mentre sono lette tre b .

Si costruisca un PDA che riconosce il linguaggio $L = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$,
 cioè le parole in cui le b seguono le a e le b sono in un numero compreso tra
 quelle delle a e due volte quello delle a.



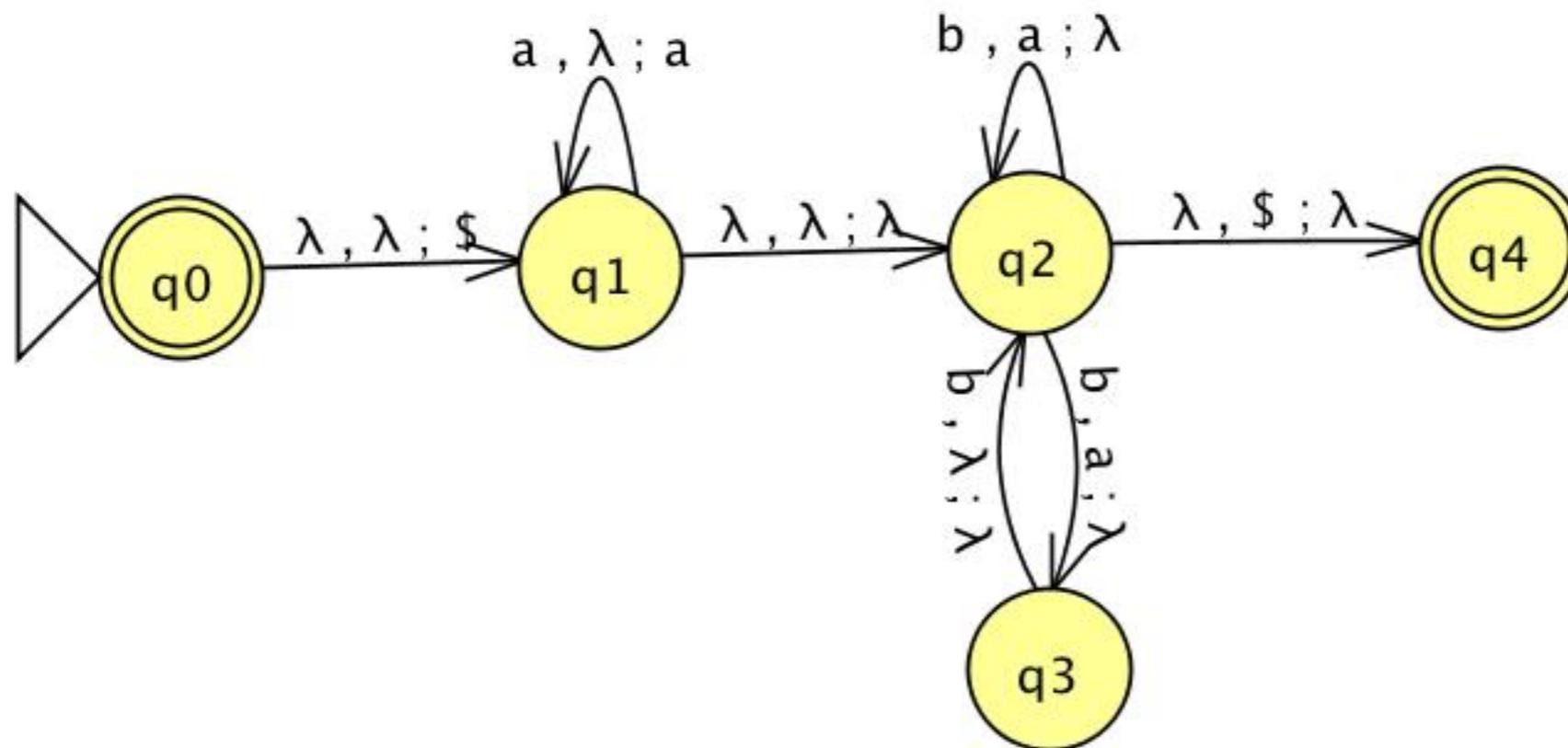
Nello stato q_1 si impilano le a una volta per ogni a letta e non
 deterministicamente, andando nello stato q_4 , si impilano 2 a per ogni a letta.
 Poi basta controllare se ogni b ha una corrispondente a nella pila, come nel
 caso del linguaggio $\{a^n b^n \mid 0 \leq n\}$. Questa operazione viene fatta nello stato q_2 ,
 dove per ogni b letta si estrae una a dalla pila.

Il PDA che riconosce il linguaggio $L = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$, cioè le parole in cui le b seguono le a e le b sono in un numero compreso tra quelle delle a e due volte quello delle a, può anche essere progettato così:



Questa versione usa un'estensione della regola di push nella pila, che ci consente di impilare una parola anziché solo un simbolo dell'alfabeto di input.

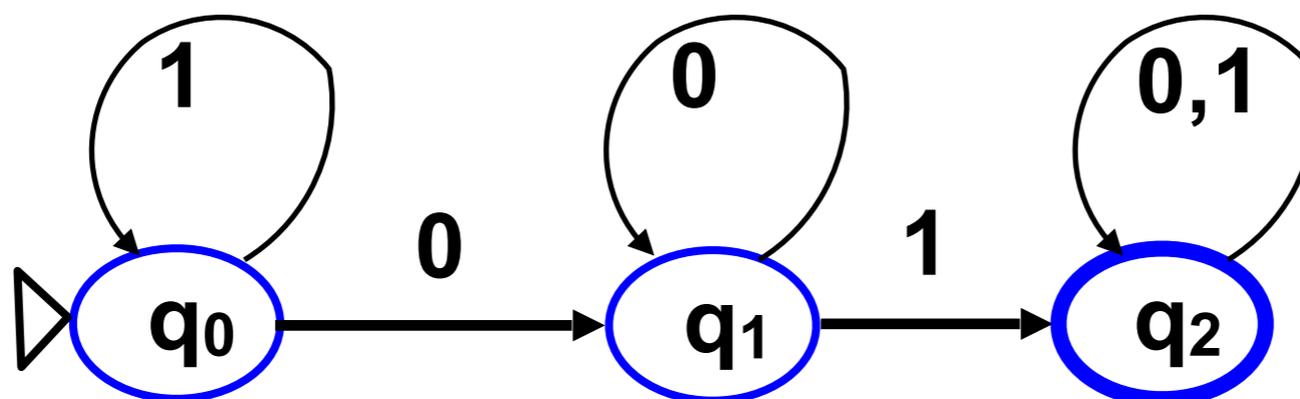
Il PDA che riconosce il linguaggio $L = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$, può anche essere costruito immaginando di impilare tutte le a una volta e poi estraendo una a dalla pila ogni b letta o ogni due b lette, sempre non deterministicamente



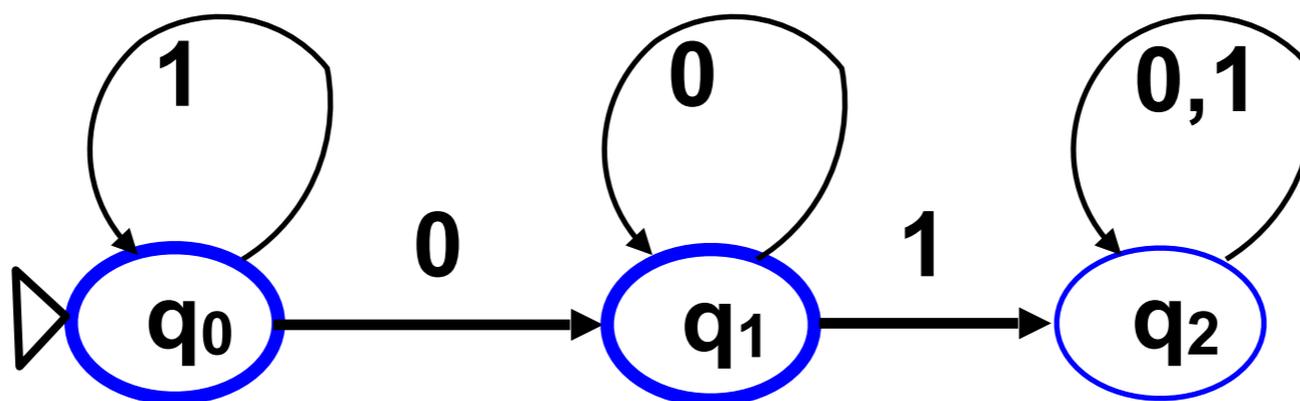
Esercizi

Si costruisca un DFA che accetta tutte le stringhe in $\{0,1\}^*$ che non contengono 01 come sottostringa.

un DFA per il linguaggio delle parole che contengono 01 come sottostringa:



Il DFA per il complemento:



Data un' espressione regolare r che denota il linguaggio L si delinei un algoritmo per produrre un'espressione regolare che denota il complemento di L .

passo 1. si costruisce un DFA A , a partire da r , tale che $L(A) = L$.

passo 2. si considera il DFA A' che riconosce il complemento di L

passo 3. si ricava l'espressione regolare che accetta $L(A')$.