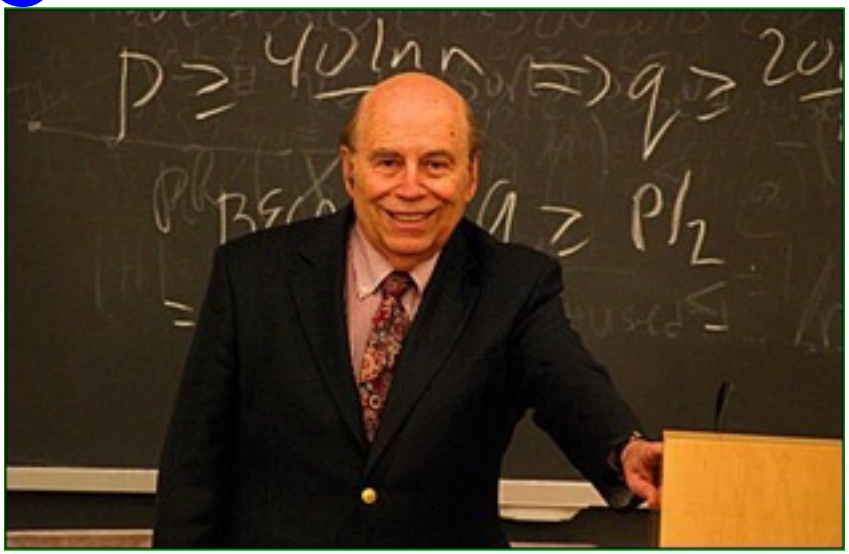
Sommario

1. Introduzione, definizione ed esempi di automai a stati finiti nondeterministici

2. Dimostrazione equivalenza tra automi a stati finiti nondeterministici e deterministici (1959)

3. Dimostrazione chiusura rispetto a prodotto e stella di Kleene



Michael O. Rabin (1931)

Dana Scott (1932)

MICHAEL O. RABIN

Michael O. Rabin è nato nel 1931 a Breslau, Germany (oggi Wrocław, in Polonia). In 1935 è emigrato in Palestina, dove si è laureato. Ha poi perfezionato la sua preparazione con un Ph.D. alla Princeton nel 1956. Poco dopo viene invitato dall'IBM, nel centro di ricerca di New York, dove insieme a Dana Scott ha scritto l'articolo dal titolo "Finite Automata and Their Decision Problems", nel quale sono introdotti gli automi a stati finiti non deterministici e che valse loro, nel 1976, il premio Turing. Tra l'altro, ha anche introdotto gli automi a stati finiti probabilistici, il test di primalità noto come Miller-Rabin test. Questo test è un algoritmo randomizzato molto veloce. Più tardi ha introdotto un criptosistema, noto come Rabin cryptosystem, un sistema asimmetrico la cui sicurezza si basa sull'intrattabilità della fattorizzazione intera.

Ha anche inventato con Karp un ben noto algoritmo per il problema della ricerca di un pattern in un testo.

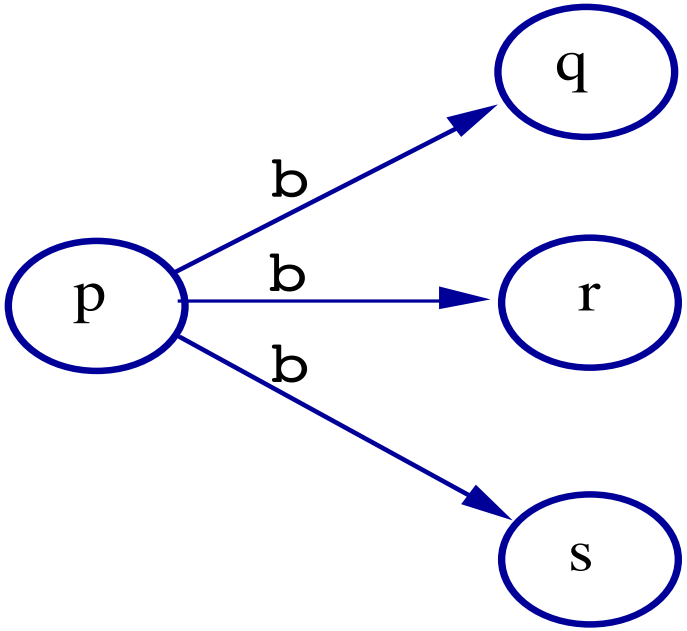
Dana Scott

E’ nato nel 1932 a Berkeley in California, è professore emerito di Computer Science, Philosophy, and Mathematical Logic at Carnegie Mellon University, ed è ora in pensione.

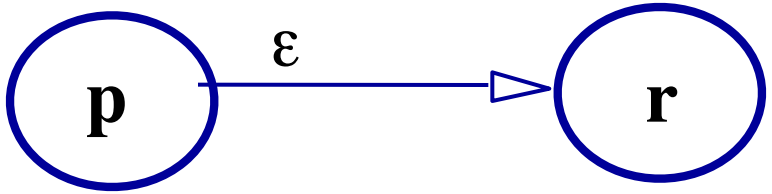
Si è occupato di logica, in particolare di logica modale e di semantica dei linguaggi di programmazione, cercando di dotarla di una solida base matematica. Questo è sfociato nel cosiddetto approccio alla Scott-Strachey per la semantica denotazionale

AUTOMI A STATI FINITI NON DETERMINISTICI

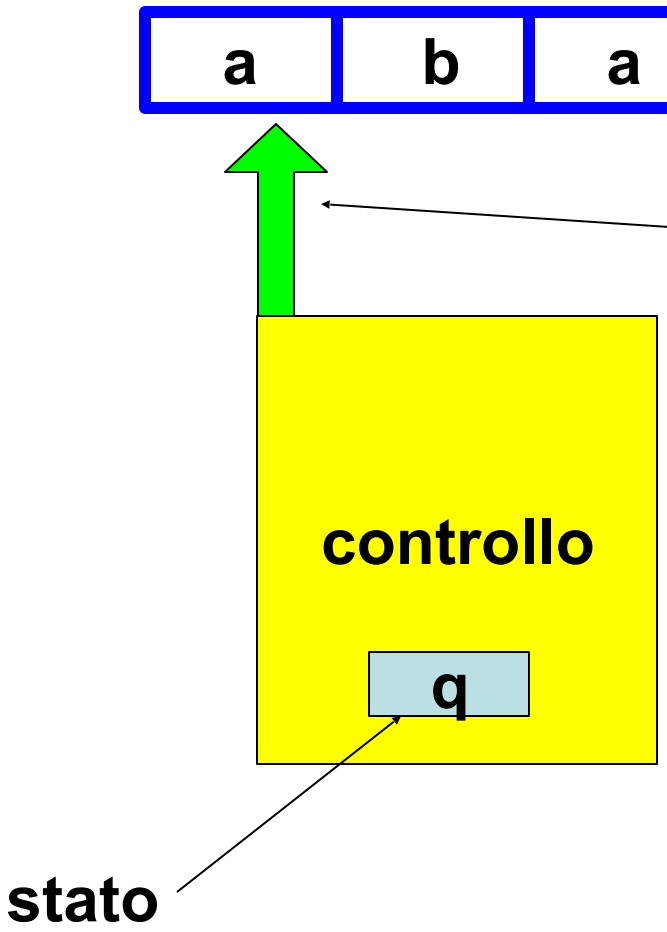
Un automa finito non deterministico può avere a disposizione più stati nei quali passare a seguito della lettura di un simbolo



oppure può cambiare stato senza leggere e consumare input.

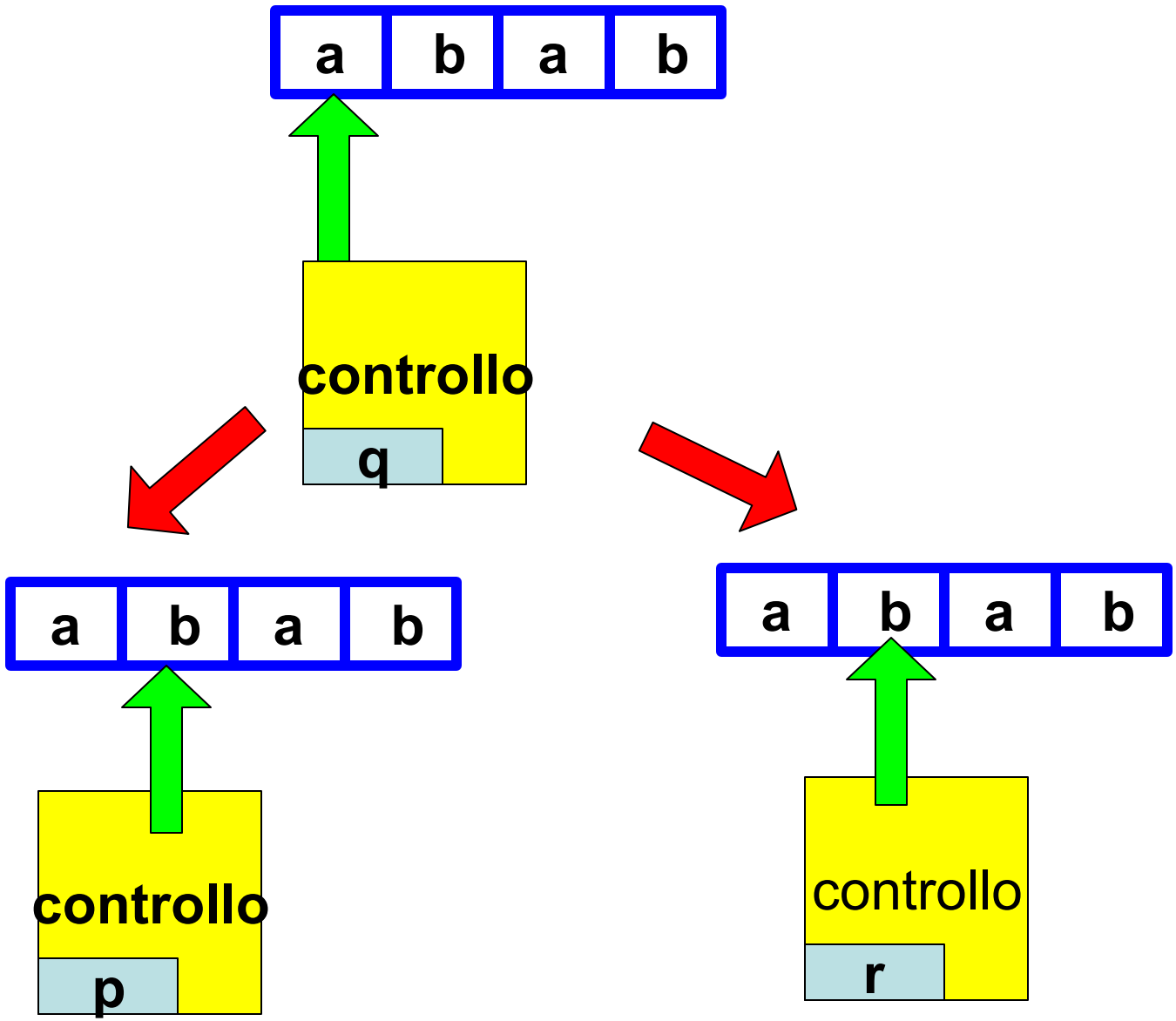


Possibile (non unico) modello mentale



In funzione dello stato in cui si trova e dell'input letto la macchina può moltiplicarsi: le macchine così create continuano il calcolo come DFA in parallelo sullo stesso input.

Un esempio parziale di esecuzione



Modello formale

Un automa a stati finiti non deterministico, in breve NFA (Nondeterministic Finite Automaton), è una quintupla $M = (Q,\Sigma,\delta,q_0,F)$ dove

- **Q è l' insieme finito degli stati;**
- **Σ è l' insieme finito dei simboli di input;**
- **$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ è la funzione (totale) di transizione;**
- **q_0 è lo stato iniziale;**
- **$F \subseteq Q$ è l' insieme degli stati finali o di accettazione**

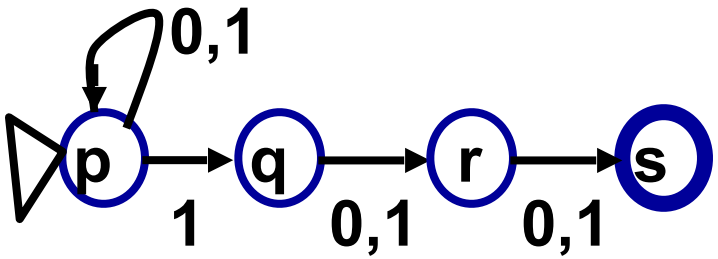
Esempio

Consideriamo il linguaggio L={x | x è in {0,1}* e x ha 1 come terzultima lettera}.

Sia M = (Q,Σ,δ,p,F) il DFA dato da: Q={p,q,r,s},
Σ={0,1}, F={s}

La funzione di transizione δ : Q × (Σ∪{ε}) → P(Q) è descritta nella tabella:

δ	1	0	ε
p	{p,q}	{p}	∅
q	{r}	{r}	∅
r	{s}	{s}	∅
s	∅	∅	∅



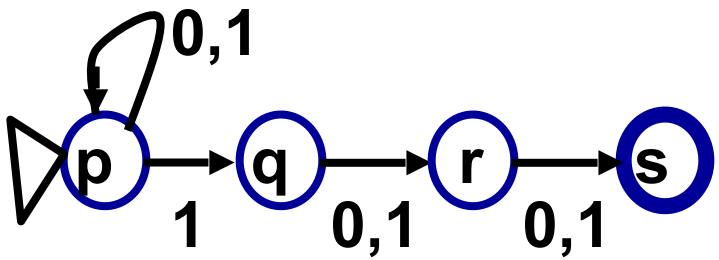
CONFIGURAZIONI E PASSI

Una configurazione di un NFA M è un elemento di $Q \times \Sigma^*$.

Per un input $x \in \Sigma^*$,
la configurazione iniziale o di partenza è (q_0,x)

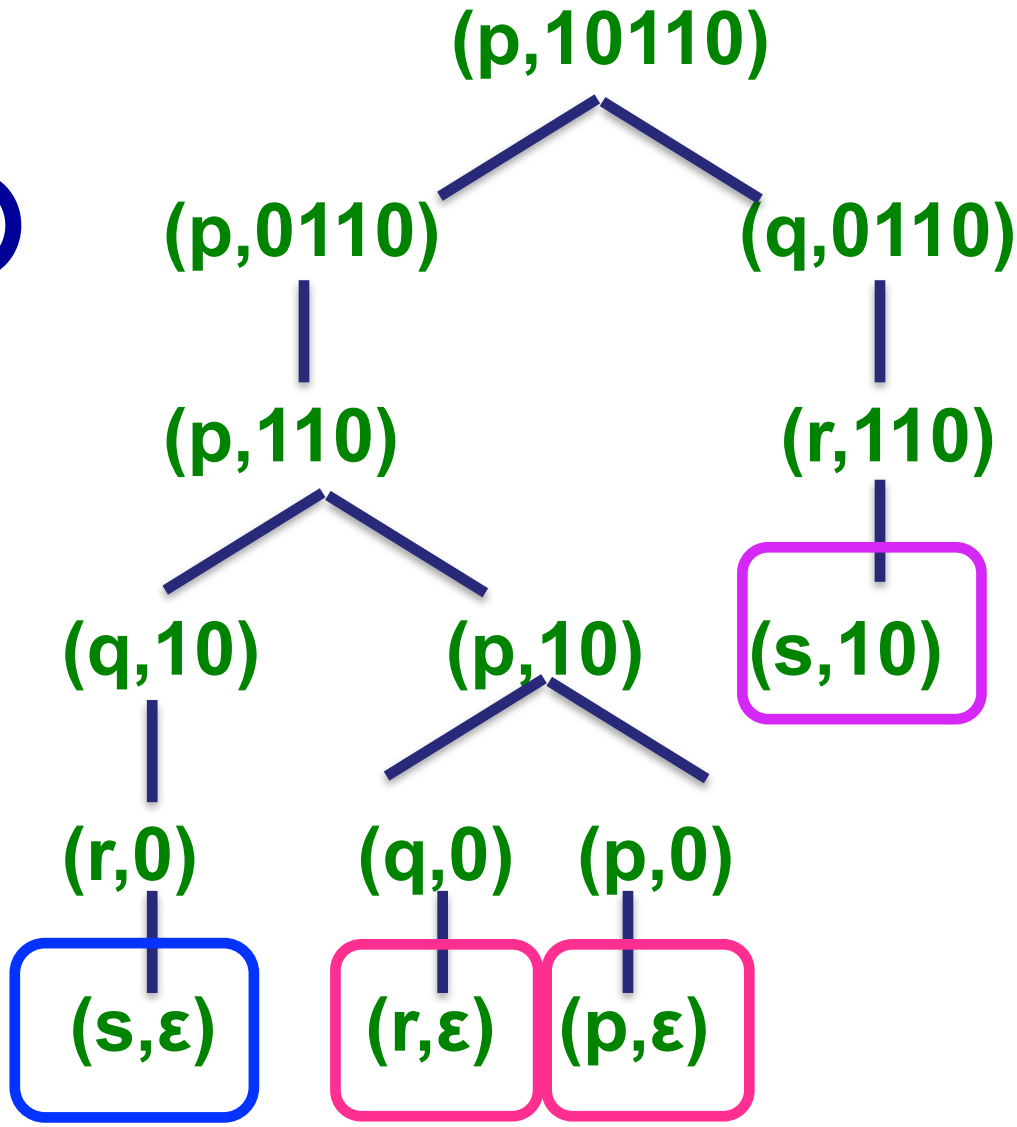
La relazione "porta a" (in un passo) $\Rightarrow_M$ è una relazione binaria su $Q \times \Sigma^*$, così definita:
se $p,q \in Q$, $a \in \Sigma$ e $x \in \Sigma^*$,
allora $(p,ax) \Rightarrow_M (q,x)$ se e solo se $q \in \delta(p,a)$

Esempio di parola accettata

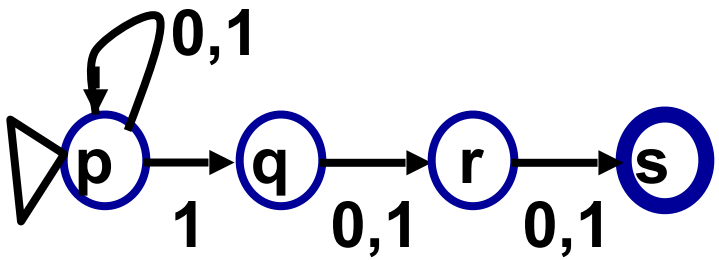


Per questo automa e l'input 10110 la configurazione iniziale è (p, 10110).

Si può usare un albero per rappresentare tutte le configurazioni raggiunte a partire da quella iniziale.

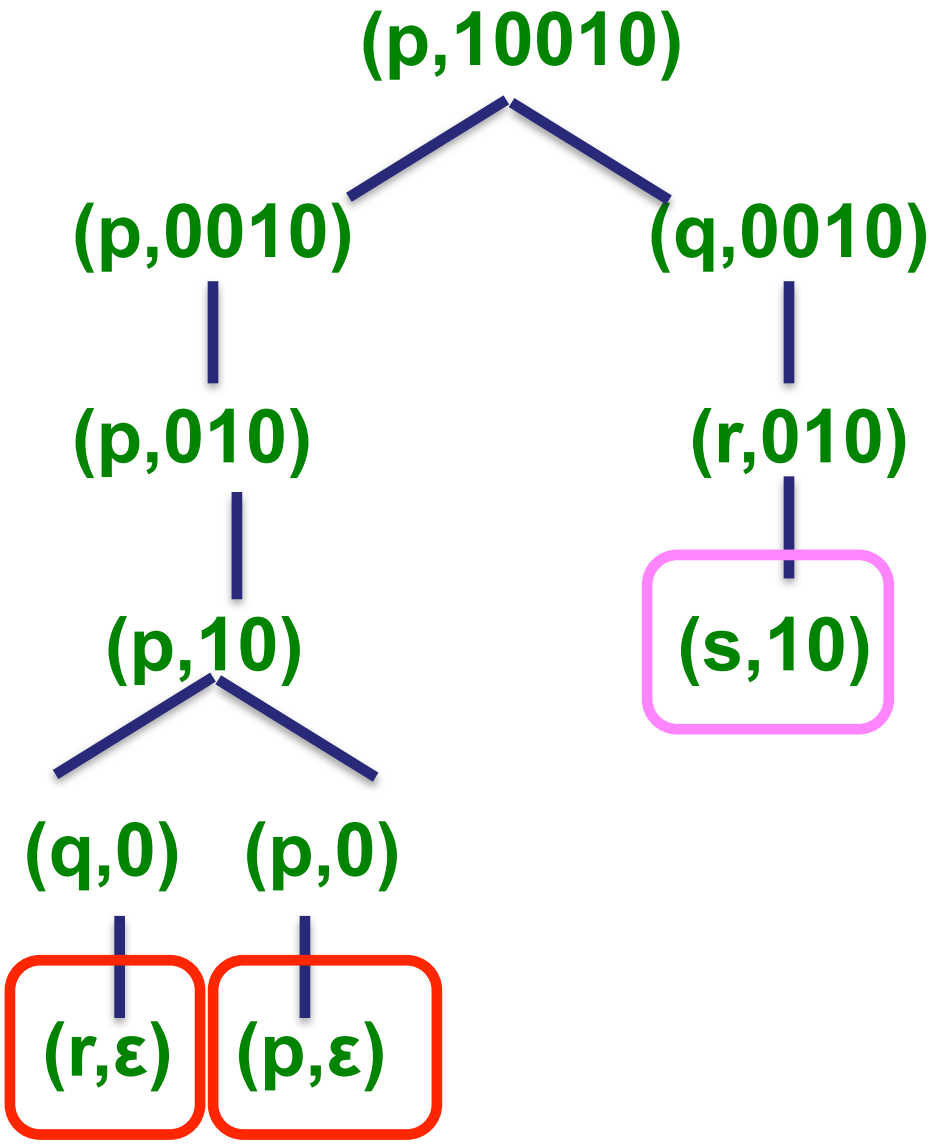


Esempio di parola non accettata



Per questo automa e l'input 101010 la configurazione iniziale è (p, 10010).

Usiamo ancora un albero per rappresentare tutte le configurazioni raggiunte a partire da quella iniziale.



Accettazione

Dato un NFA $M = (Q,\Sigma,\delta,q_0,F)$, una parola $x \in \Sigma^*$ è accettata da M se $\exists\ q \in F$ tale che

$$(q_0,x) \Rightarrow_M^* (q,\varepsilon)$$

dove $\Rightarrow_M^*$ la chiusura riflessiva e transitiva di $\Rightarrow_M$

e il linguagggio accettato da M è

$$L(M) = \{x \mid \in \Sigma^* \text{ ed è accettata da M}\}$$

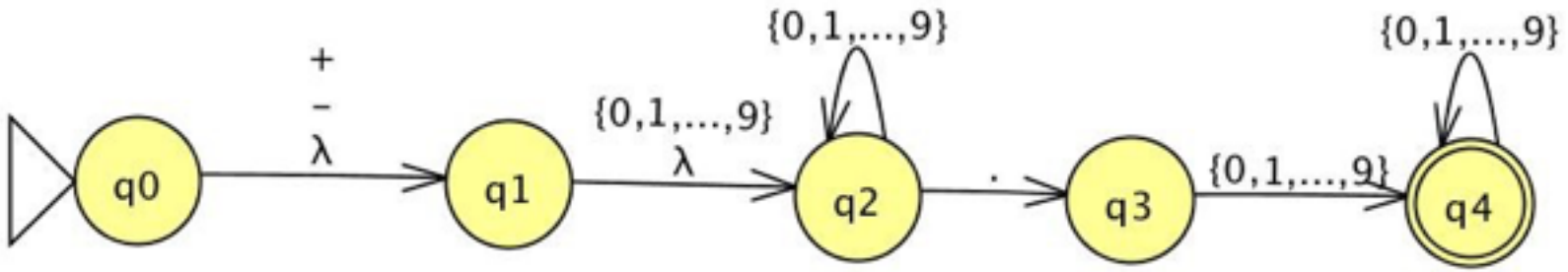
La classe dei linguaggi accettati da un NFA è

$$\mathcal{L}(NFA) = \{L \mid \exists\ M \in NFA \text{ e } L(M) = L\}$$

Dato un NFA $M = (Q,\Sigma,\delta,q_0,F)$, una parola $x \in \Sigma^*$ è rifiutata da M se $\forall$ q tale che $(q_0,x) \Rightarrow_M^* (q,\varepsilon)$ q non è in F.

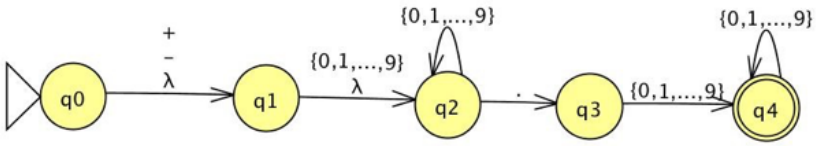
Può capitare che $(q_0,x) \Rightarrow_M^* (q,ay)$, con q in F, $y \in \Sigma^*$ e $\delta(q,a)= \varnothing$, poiché la parola non viene tutta letta questo è un cammino bloccato che non può influire sull'accettazione o meno di x.

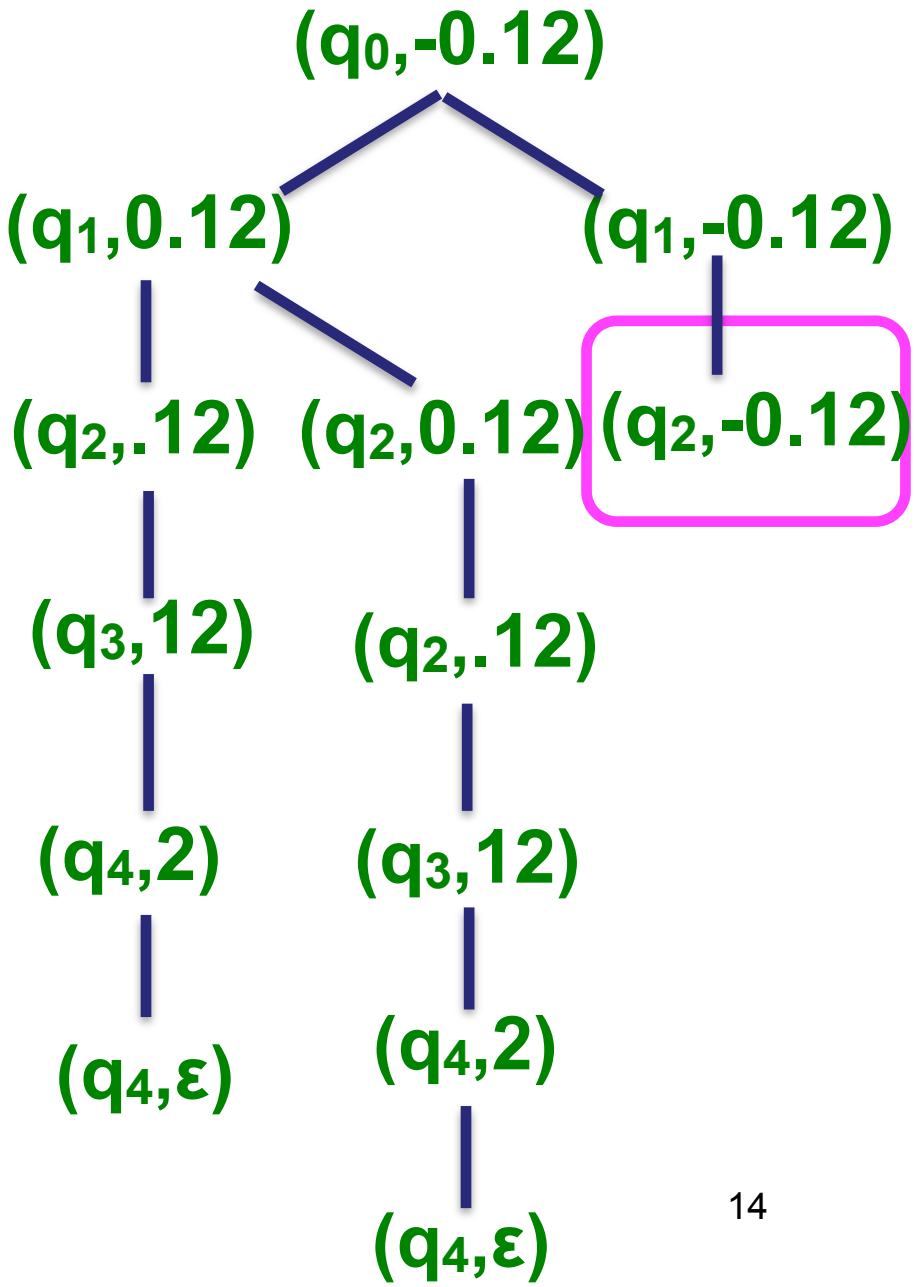
Esempio di NFA con ε-mosse



Il linguaggio accettato da questo NFA è quello dei numeri decimali, nella notazione anglosassone

Esempio di computazione su -0.12





Invece la parola +-0.1 non è accettata perché la funzione di transizione dà l'insieme vuoto sulla coppia (q_1,-)

NON DETERMINISMO E REGOLARI

Nel caso degli automi a stati finiti il non determinismo non aumenta il potere computazionale, detta $\mathcal{L}$(NFA) la classe dei linguaggi accettati dagli automi finiti non deterministici:

TEOREMA $\mathcal{L}$(DFA) = $\mathcal{L}$(NFA)

CONFRONTO TRA NFA e DFA

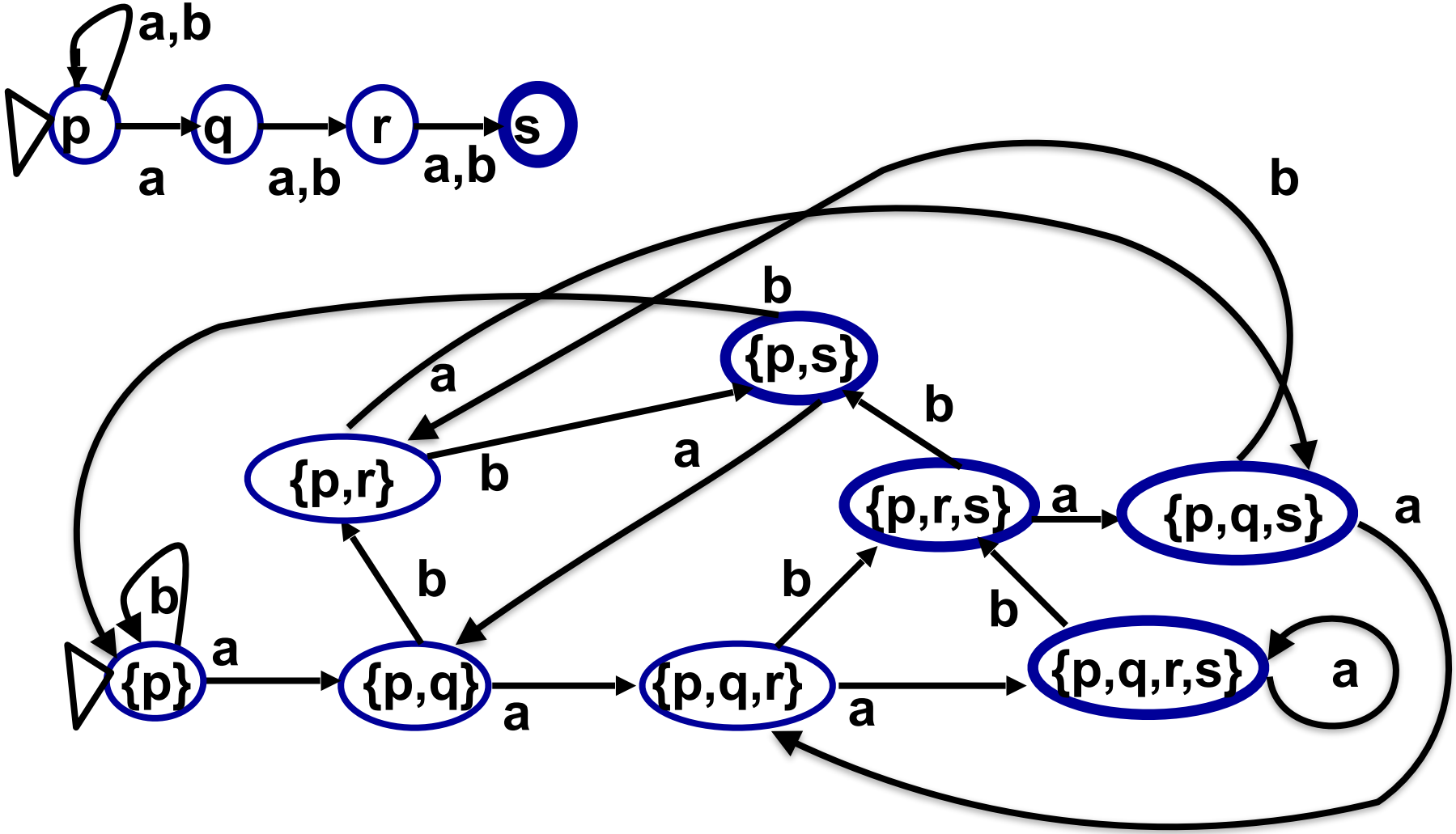
Ogni DFA può essere visto come un caso particolare di NFA e quindi

$$\mathcal{L}(\text{DFA}) \subseteq \mathcal{L}(\text{NFA})$$

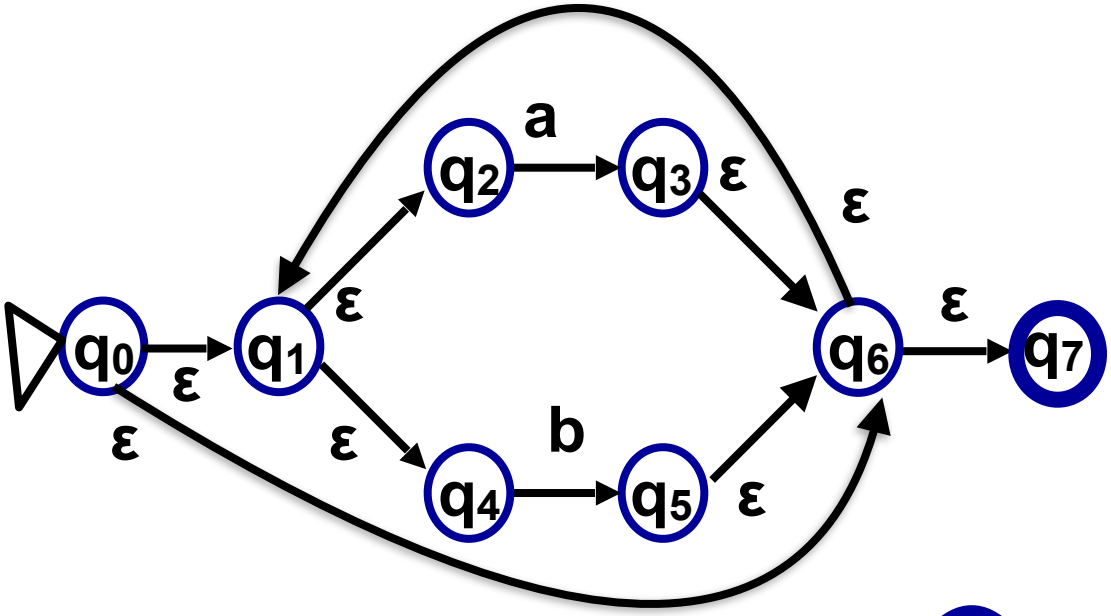
Ma possiamo anche dimostrare che per ogni NFA M, esiste un DFA M1 equivalente a M, cioè tale che L(M)=L(M1) e cioè che

$$\mathcal{L}(\text{NFA}) \subseteq \mathcal{L}(\text{DFA})$$

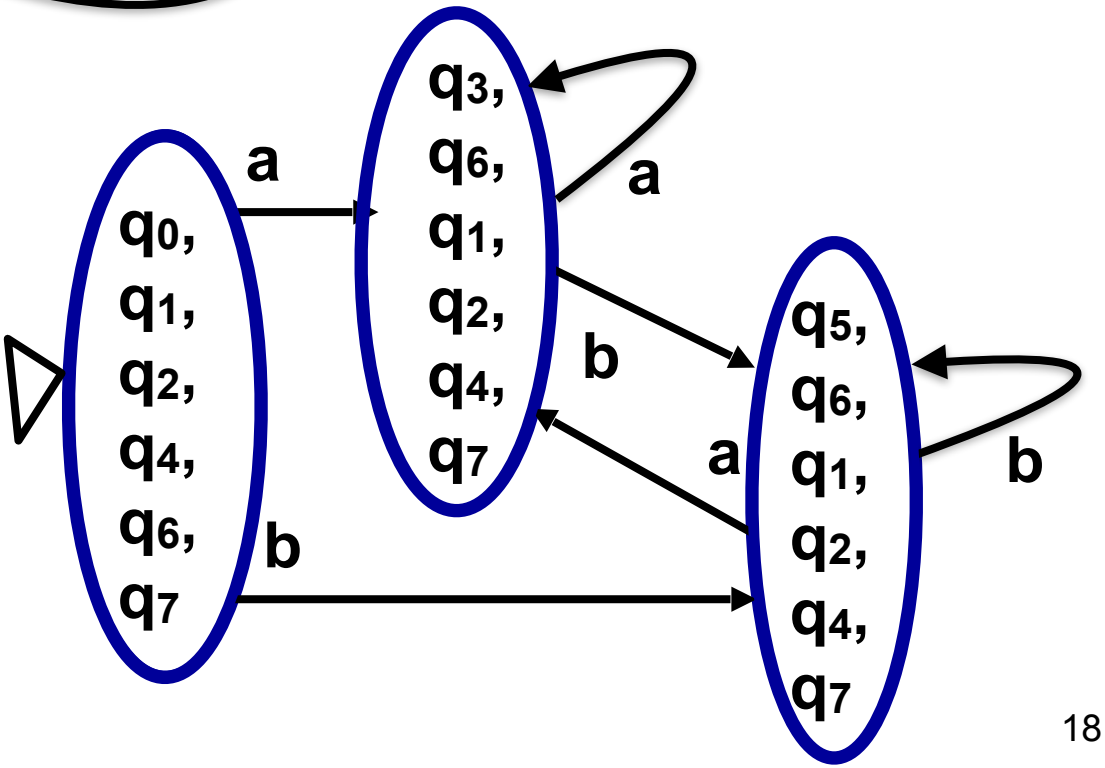
DIMOSTRAZIONE: L'IDEA



ESEMPIO



ε-closure(q)= l'insieme degli stati raggiungibili da q con solo ε-mosse.
Qui le parentesi graffe sono omesse



Algoritmo di costruzione di un DFA equivalente a un NFA

Algoritmo DFAeqNFA
input un NFA con ε-mosse $M=(\Sigma,Q,\delta,q_0,F)$
output un DFA $M'=(\Sigma,Q',\delta',q_0',F')$ equivalente
Inizializza Q' all'insieme che contiene
ε-closure(q_0) come unico elemento non marcato.
while c'è uno stato non marcato T in Q' do
{marca T;
for ogni simbolo input a in Σ do
{$V = \cup_{q \in T} \delta(q,a)$);
$V' = \cup_{q \in V}$ ε-closure(q);
if V' non appartiene a Q' then
aggiungi V' come stato non marcato in Q';
poni $\delta'(T,a)= V'$
}
}

Costruzione di un DFA equivalente a un NFA: conclusione

Con l'algoritmo DFAeqNFA abbiamo costruito l'insieme degli stati e la funzione di transizione del DFA $M'=(\Sigma,Q',\delta',q_0',F')$ equivalente al dato NFA $M=(\Sigma,Q,\delta,q_0,F)$.

Lo stato iniziale di M',q_0', è l'insieme degli stati raggiungibili da quello iniziale di M utilizzando solo ε-mosse, cioè q_0' = ε-closure(q_0).

Gli stati finali di M' sono gli stati di Q' che contengono uno stato finale di M

OPERAZIONI SUI LINGUAGGI

$\mathcal{L}$(DFA) è chiusa rispetto alle operazioni di concatenazione e stella di Kleene:

Proveremo quindi che

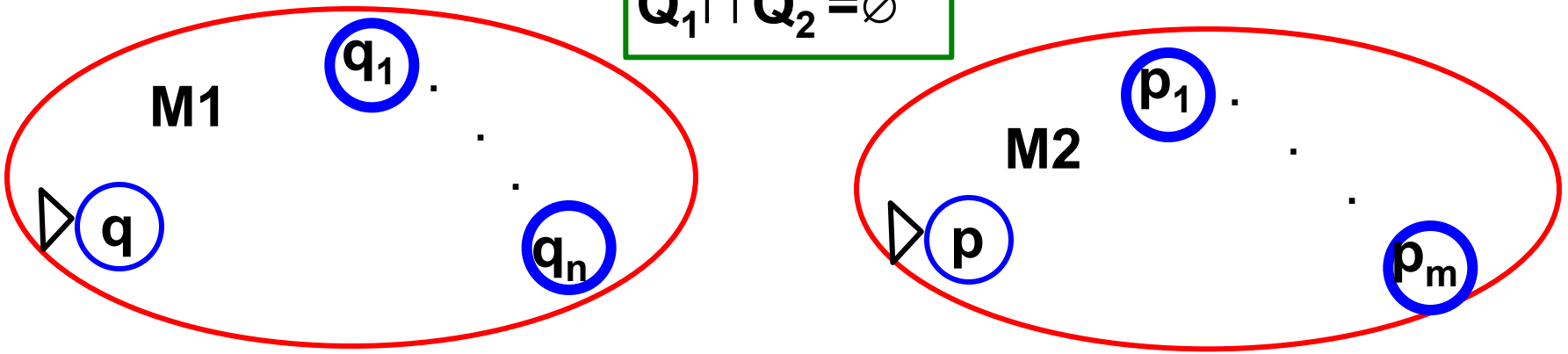
$L, L' \in \mathcal{L}(DFA) \Rightarrow LL' \in \mathcal{L}(DFA)$

$L \in \mathcal{L}(DFA) \Rightarrow L^* \in \mathcal{L}(DFA)$

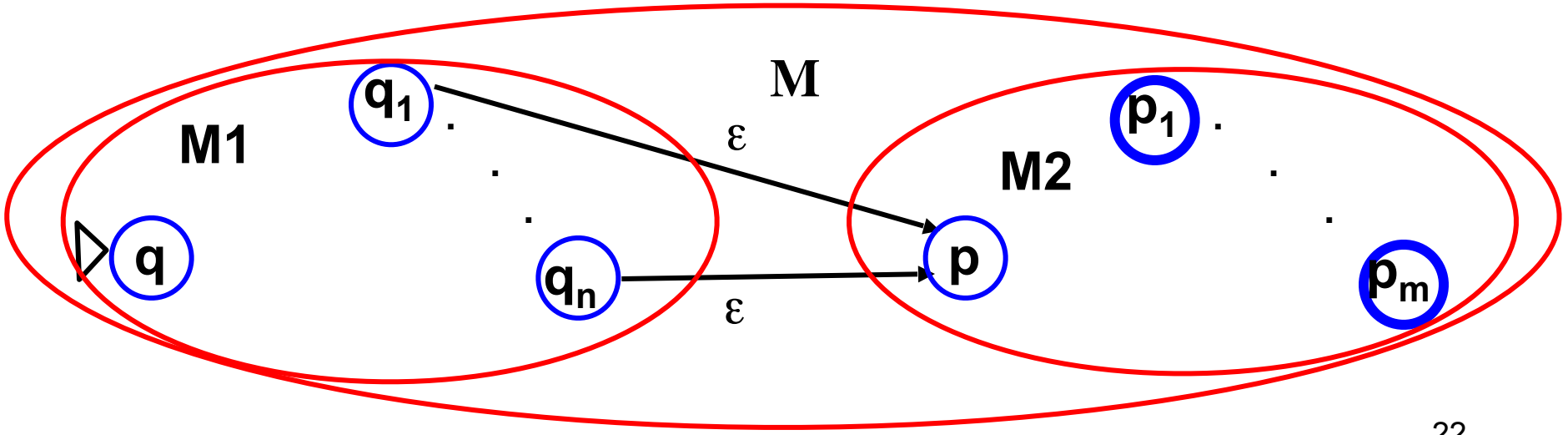
Prodotto, M1 e M2 sono DFA

$M1 = (Q_1, \Sigma, \delta_1, q, F_1=\{q_1,\ldots,q_n\})$ $M2 = (Q_2, \Sigma, \delta_2, p, F_2=\{p_1,\ldots,p_m\})$

$Q_1 \cap Q_2 = \varnothing$



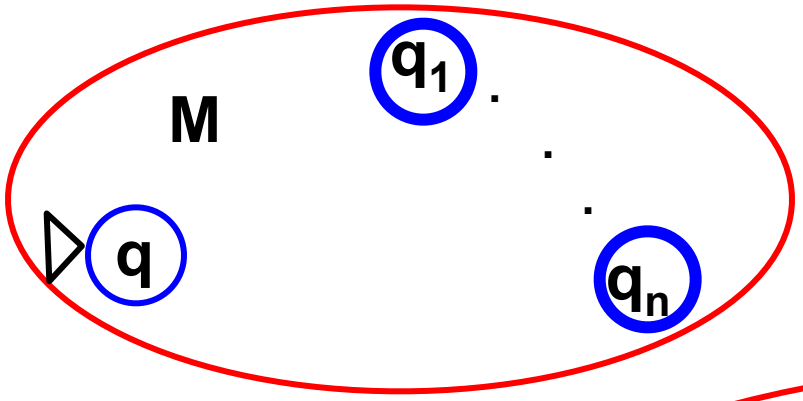
$M = (Q_1 \cup Q_2, \Sigma, \delta, q, F=F_2)$



M è un NFA tale che L(M) = L(M1)L(M2)

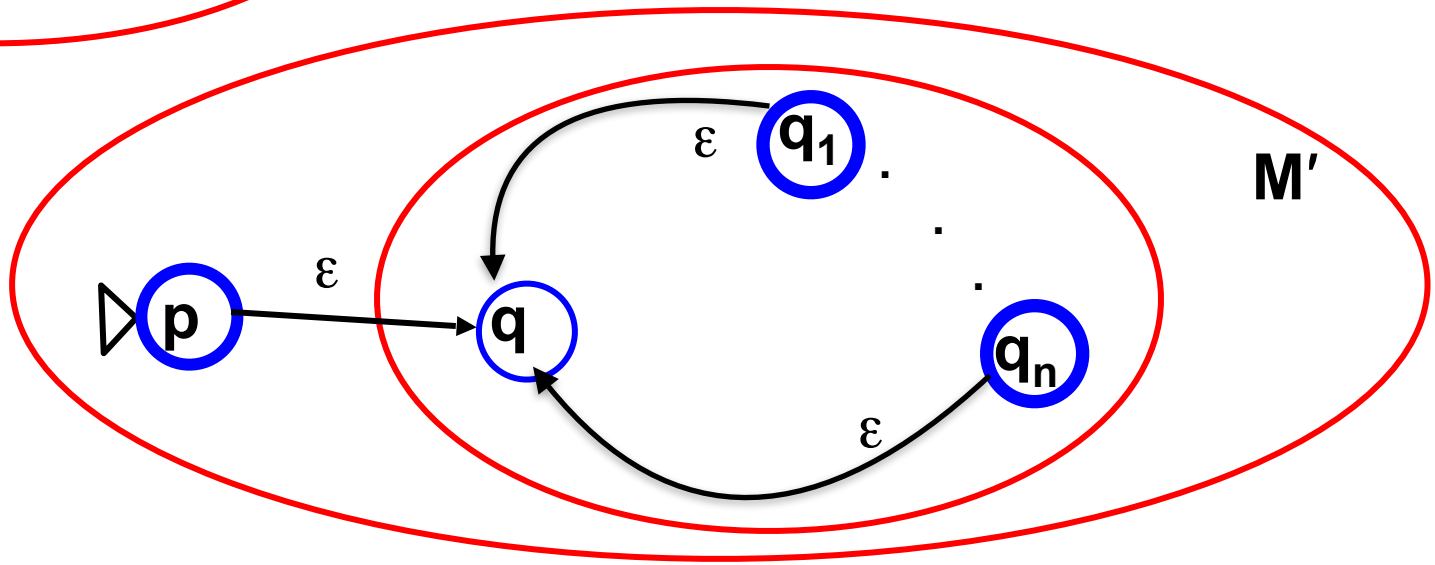
STELLA DI KLEENE, M è un DFA

$M = (Q,\Sigma,\delta,q,F=\{q_1,\ldots,q_n\})$



$p \notin Q$

$M' = (Q \cup \{p\},\Sigma,\delta',p,F'=F \cup \{p\})$



M′ è un NFA tale che L(M′) = L(M)*

CURIOSITÀ

1. **Perchè è necessario aggiungere un nuovo stato nella costruzione dell'automa per la chiusura rispetto alla stella di Kleene?**
2. **In quale caso potrei evitare di introdurre nuove epsilon mosse nella costruzione per la chiusura rispetto al prodotto?**
3. **La classe dei linguaggi regolari è chiusa rispetto alla differenza tra linguaggi?**

   (la differenza tra due linguaggi L ed L' è così definita $L-L' = \{x \mid x \in L \text{ e } x \notin L'\}$)