AUTOMI A STATI FINITI

Stati

- Un FSA (*Finite State Automaton*) ha un insieme finito di stati
  - Un sistema con un numero limitato di configurazioni
  - (in italiano anche: AF)
- Esempi
  - {On, Off},
  - {1,2,3,4, …,k}
  - {canali TV}
  - …
- Gli stati sono rappresentati come segue:

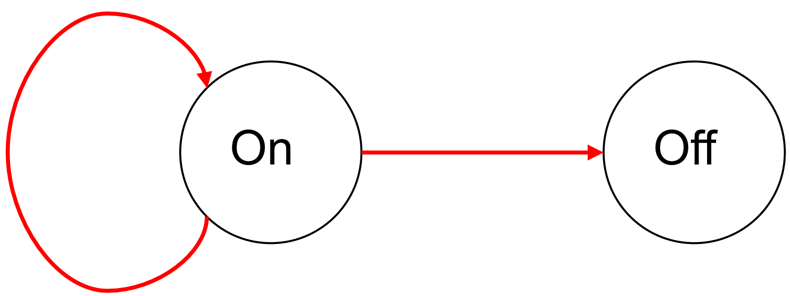
Ingresso (input)

- Un FSA è definito su un alfabeto
- I simboli dell’alfabeto rappresentano l’ingresso del sistema
- Esempi
  - {a, b, c}
  - {0, 1}

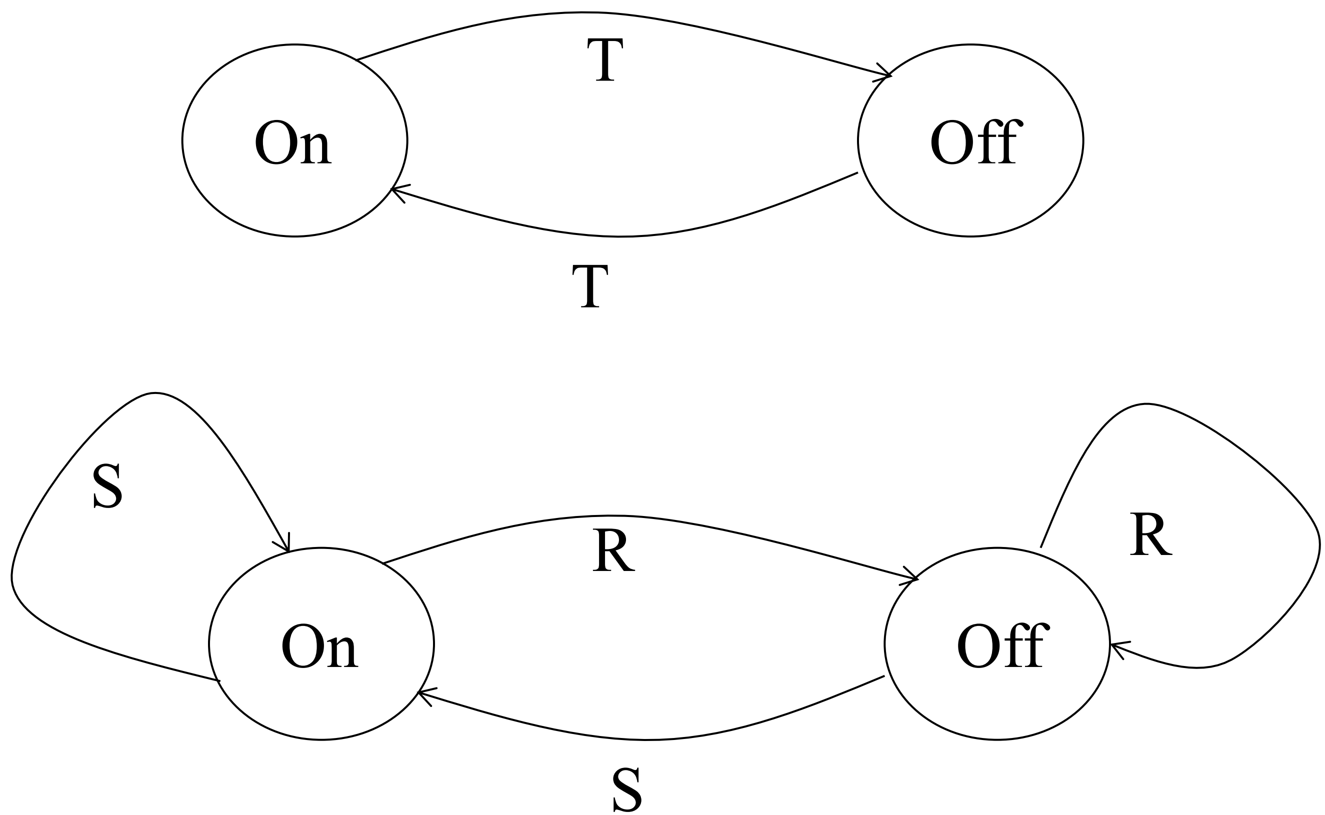
  Ma anche nomi più complessi di simboli, se servono a rendere più comprensibile lo scenario rappresentato:
  - {switch_on, switch_off}
  - {incoming==0, 0<incoming<=10, incoming>10}

Transizioni tra stati

- Quando si riceve un ingresso, il sistema cambia il proprio stato
- Il passaggio da uno stato all'altro avviene tramite <u>transizioni</u>
- Una transizione è rappresentata mediante frecce:



Semplici esempi (flip-flop)



FSA

- Gli FSA sono il più semplice modello di computazione
- Molti utili dispositivi possono essere modellati tramite FSA

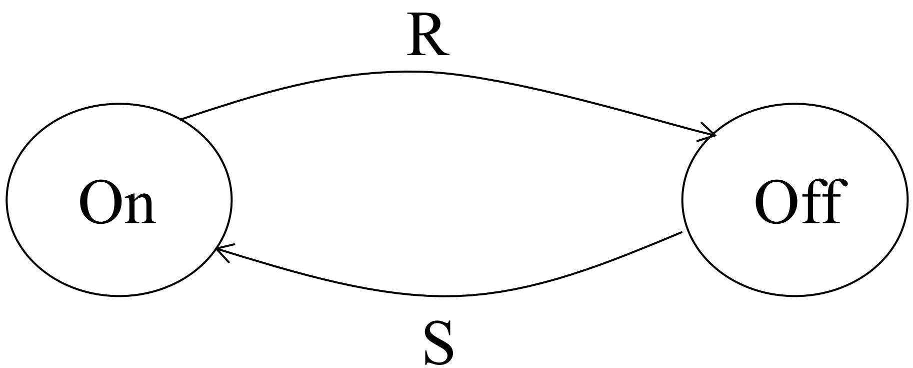
… hanno però alcune limitazioni

Formalmente

- Un FSA è una tripla <Q, A, δ>, dove
  - Q è un insieme finito di stati
  - A è l'alfabeto di ingresso
  - δ è una funzione di transizione (che può essere parziale), data da δ: $Q \times A \rightarrow Q$
- Nota

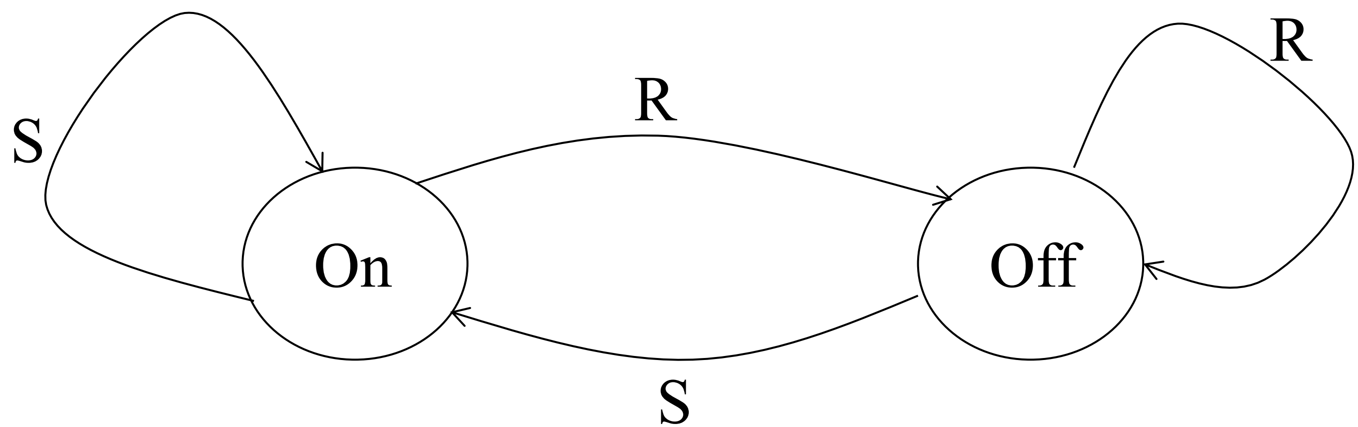
  se la funzione è parziale, allora non tutte le transizioni da tutti i possibili stati per tutti i possibili elementi dell'alfabeto sono definite

Funzione di transizione parziale o totale



Un FSA con una funzione di transizione totale è detto <u>completo</u>

Funzione di transizione parziale o totale



Un FSA con una funzione di transizione totale è detto <u>completo</u>

Riconoscimento di linguaggi

- Per poter usare gli FSA per riconoscere linguaggi, è importante identificare:
  - Le condizioni iniziali del sistema
  - Gli stati finali ammissibili
- Esempio:
  - La luce dev'essere spenta all'inizio e alla fine

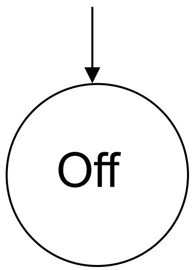
Elementi

- Gli elementi del modello sono
  - Stati
  - Transizioni
  - Ingresso

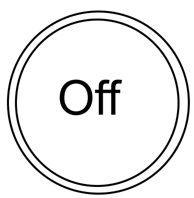
  e anche
  - Stato iniziale
  - Stati finali

Rappresentazione grafica

- Stato iniziale



- Stato finale



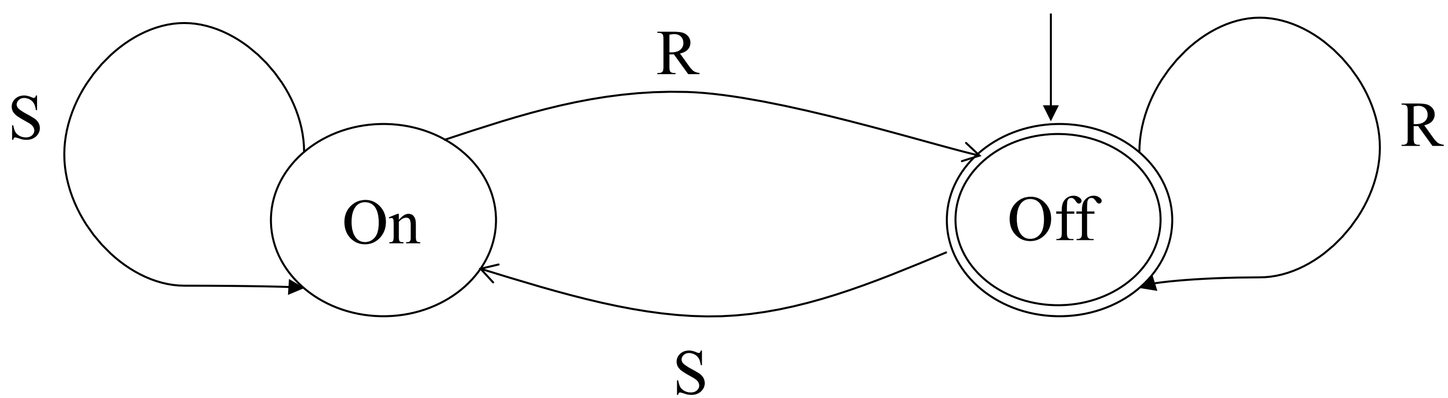
Formalmente

- Un FSA è una tupla <Q, A, δ , q_0, F>, dove
  - Q è un insieme finito di stati
  - A è l'alfabeto di ingresso
  - δ è una funzione di transizione (parziale), data da

    $\delta: Q \times A \to Q$
  - $q_0 \in Q$ è detto lo <u>stato iniziale</u>
  - $F \subseteq Q$ è l'insieme di <u>stati finali</u>

Sequenza di mosse

- Una <u>sequenza di mosse</u> inizia da uno stato iniziale ed è di *accettazione* se raggiunge uno degli stati finali

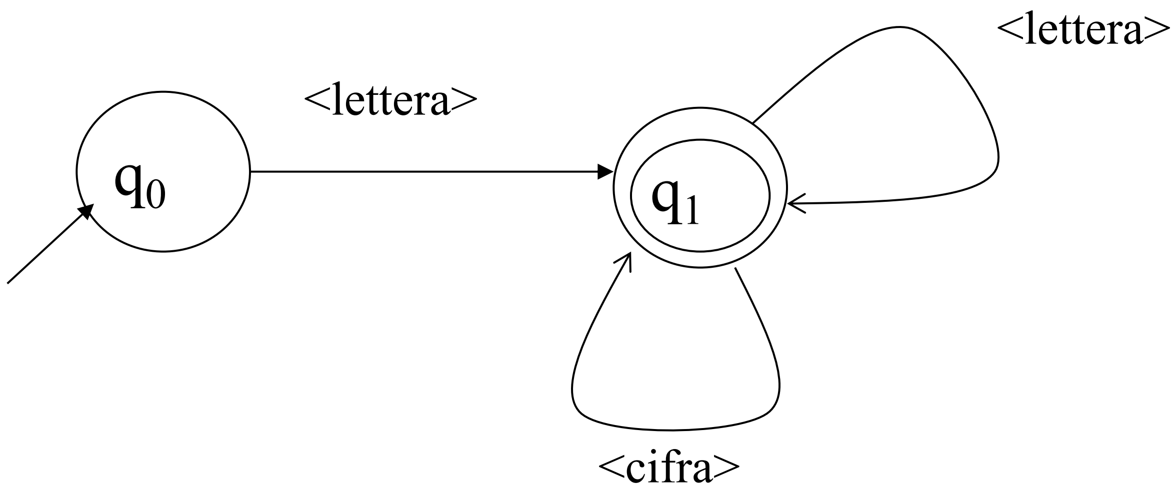


Formalmente

- Sequenza di mosse:
  - $\delta^*: Q \times A^* \rightarrow Q$
- δ^* è definita induttivamente da δ
  - $\delta^*(q,\varepsilon) = q$
  - $\delta^*(q,yi) = \delta(\delta^*(q,y), i)$
- Stato iniziale: $q_0 \in Q$
- Stati finali (o di accettazione): $F \subseteq Q$
- $\forall x \, (x \in L \leftrightarrow \delta^* (q_0,x) \in F)$

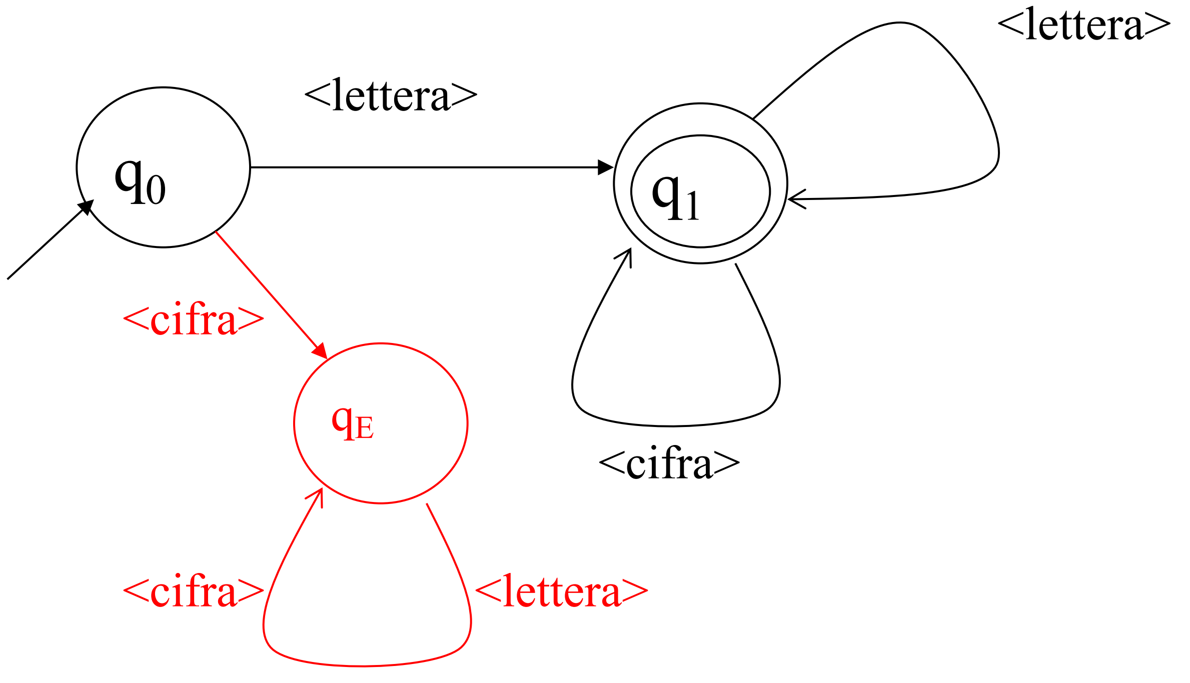
Esempio pratico

- Riconoscimento degli identificatori del Pascal



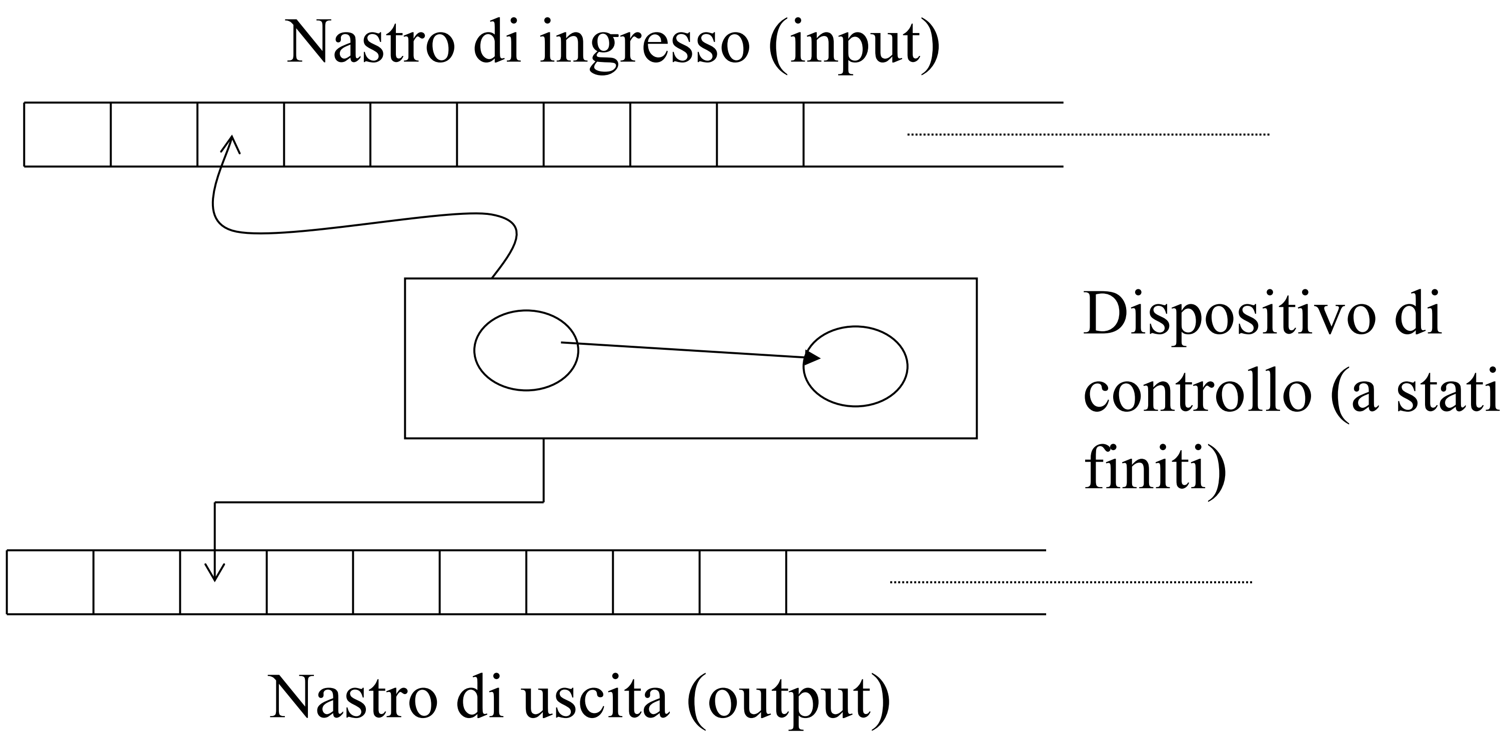
Esempio pratico

- Riconoscimento degli identificatori del Pascal



TRASDUTTORI A STATI FINITI

Automi come traduttori di linguaggi



Un FST (finite state transducer) è un FSA che lavora su due nastri.

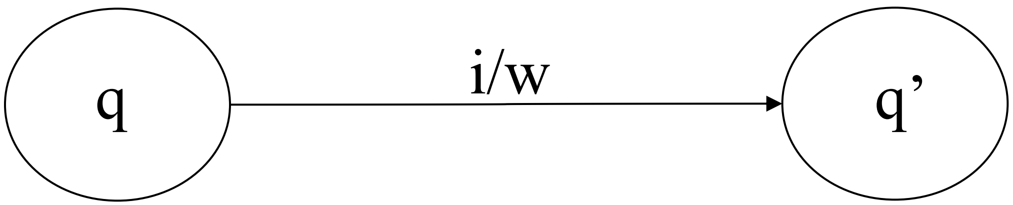
→ è una sorta di «macchina traduttrice».

L’idea

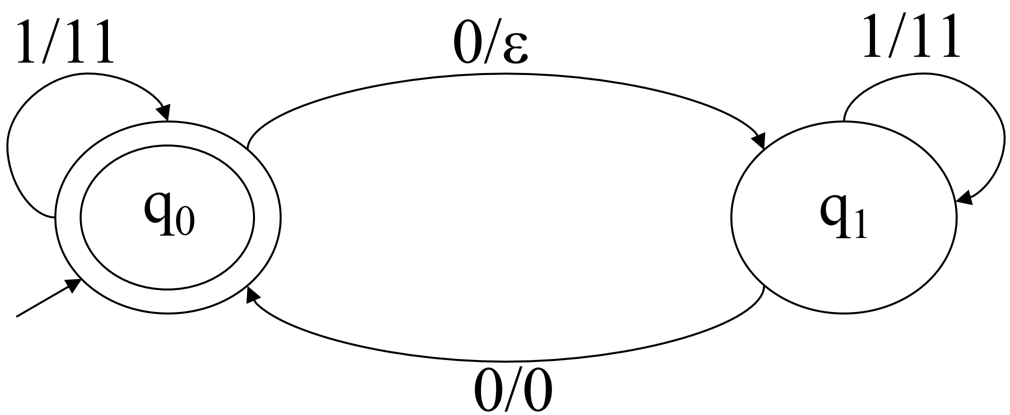
- $y = \tau(x)$
  - x: stringa d’ingresso
  - y: uscita
  - τ: funzione da L_1 a L_2
- Esempi:
  - τ_1 le occorrenze di “1” sono raddoppiate (1 --> 11)
  - τ_2 ‘a’ è scambiato con ‘b’ (a <---> b):
- Ma anche
  - Compressione di file
  - Compilazione da linguaggi di alto livello a linguaggi oggetto
  - Traduzione da inglese a italiano

Informalmente

- Transizioni con uscita



- Esempio: τ dimezza il numero di "0" e raddoppia il numero di "1"



Formalmente

- Un trasduttore a stati finiti (FST) è una tupla $T = \langle Q, I, \delta, q_0, F, O, \eta \rangle$
  - $\langle Q, I, \delta, q_0, F \rangle$: esattamente come gli accettori
  - O: alfabeto di uscita
  - $\eta : Q \times I \rightarrow O^*$
- Nota: la condizione di accettazione resta la stessa degli accettori
  - La traduzione è eseguita solo su stringhe accettate

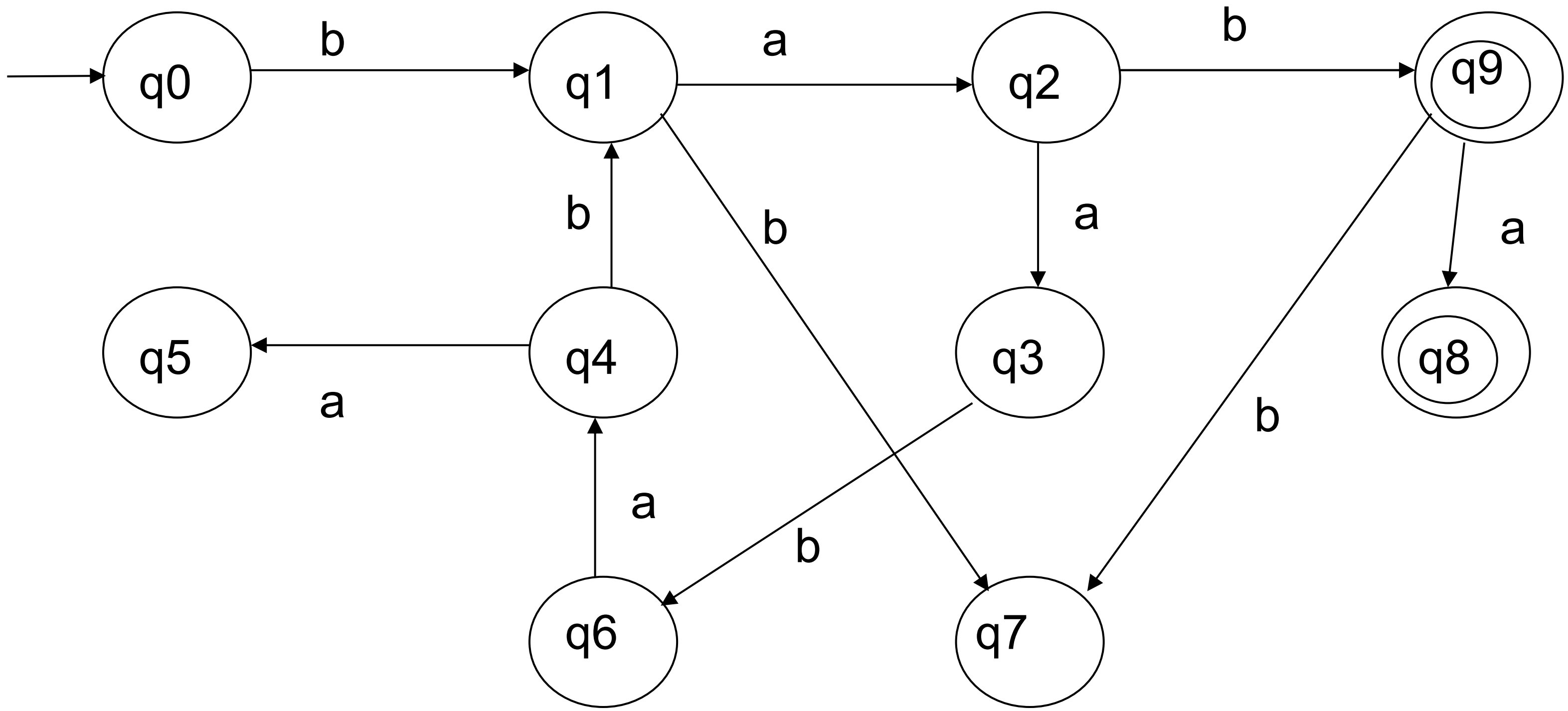
Traduzione di una stringa

- Come per la δ, definiamo η^* induttivamente
  - $\eta^*(q,\varepsilon) = \varepsilon$
  - $\eta^*(q,y.i) = \eta^*(q,y).\eta(\delta^*(q,y), i)$
- Nota: $\eta^*: Q \times I^* \rightarrow O^*$

$\forall x\ (\tau(x) = \eta^*(q_0,x)$ iff $\delta^*(q_0,x) \in F)$

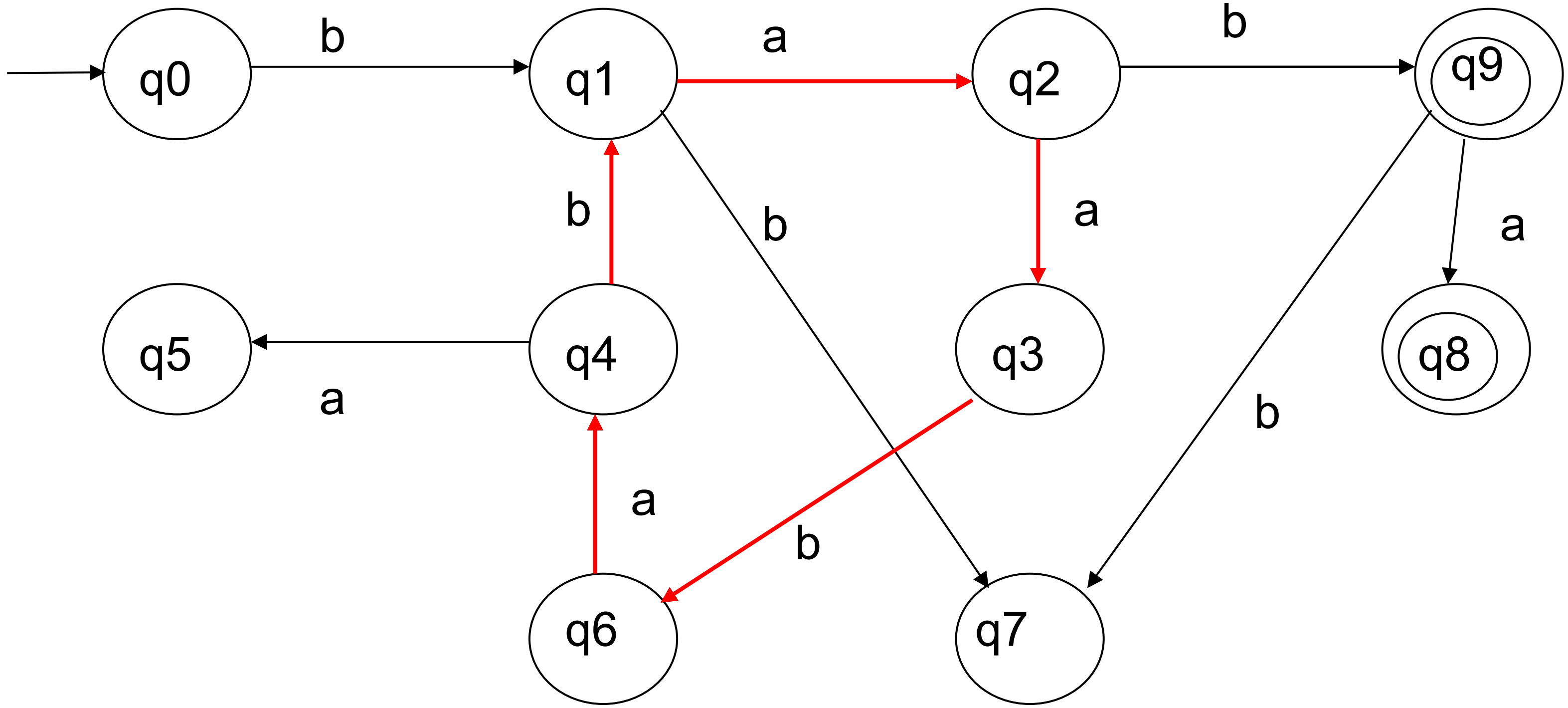
PUMPING LEMMA

Cicli



Cicli

C’è un ciclo: q1 ----aabab---> q1



Se si attraversa un ciclo una volta, lo si può attraversare anche 2, 3, …, n volte

Più formalmente

- Se $x \in L$ e $|x| \geq |Q|$, allora esistono uno stato $q \in Q$ e una stringa $w \in I^+$ tali che:
  - $x = ywz$
  - $\delta^*(q,w) = q$

- Perciò vale anche quanto segue:
  - $\forall n \geq 0 \quad yw^n z \in L$

Questo è il *Pumping Lemma* (si può “pompare” w)

Conseguenze del pumping lemma

- $L = \emptyset$? $\exists x \in L \leftrightarrow \exists y \in L \;\; |y| < |Q|$:
  Basta "togliere tutti i cicli" dall'FSA che accetta x
- $|L| = \infty$? Verifica in modo analogo se $\exists x \in L \;\; |Q| <= |x| < 2|Q|$
- Nota che *in generale* sapere come rispondere alla domanda "$x \in L$?" per una generica x, *non* implica sapere come rispondere alle altre domande
  - Funziona per gli FSA, ma…

Impatto in pratica

- Ci interessa un linguaggio di programmazione che consiste di… 0 programmi corretti?
- Ci interessa un linguaggio di programmazione in cui si può solo scrivere un numero finito di programmi?
- …

Una conseguenza negativa del pumping lemma

- Il linguaggio $L = \{a^n b^n \mid n > 0\}$ è riconosciuto da qualche FSA?
- Assumiamo che lo sia. Allora:
- Consideriamo $x = a^m b^m$, $m > |Q|$ e applichiamo il P.L.
- Casi possibili:
  - $x = ywz$, $w = a^k$, $k > 0$ ====> $a^{m + r \cdot k} b^m \in L, \forall r$: NO
  - $x = ywz$, $w = b^k$, $k > 0$ ====> idem
  - $x = ywz$, $w = a^k b^s$, $k, s > 0$ ====> $a^{m-k} a^k b^s a^k b^s b^{m-s} \in L$: NO

Intuitivamente

- Per “contare” un numero *n* arbitrariamente grande, ci serve una memoria infinita!
- A rigore, ogni computer è un FSA, ma… è un livello errato di astrazione: numero di stati intrattabile!

  (è come studiare ogni singola molecola per descrivere il volo di un aereo)
- Importanza di una nozione astratta di infinito
- Dall’esempio giocattolo $\{a^n b^n\}$ ad altri casi concreti:
  - La verifica del buon bilanciamento delle parentesi (tipicamente usato nei linguaggi di programmazione) non si può fare con memoria finita
- Ci servono quindi modelli più potenti

OPERAZIONI SUGLI FSA

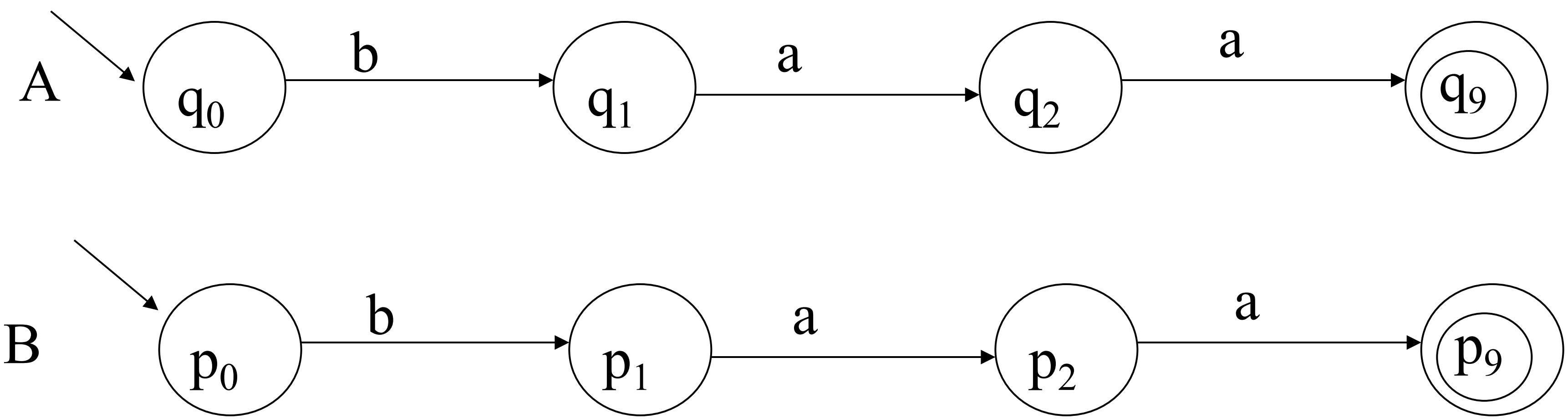
Chiusura in matematica

- Un insieme S è <u>chiuso</u> rispetto ad una operazione OP se, quando OP è applicata agli elementi di S, il risultato è ancora un elemento di S
- Esempi:
  - I numeri naturali sono chiusi rispetto alla somma (ma non rispetto alla sottrazione)
  - Gli interi sono chiusi rispetto a somma, sottrazione e moltiplicazione (ma non divisione)
  - Razionali…
  - Reali…
  - …

Chiusura per i linguaggi

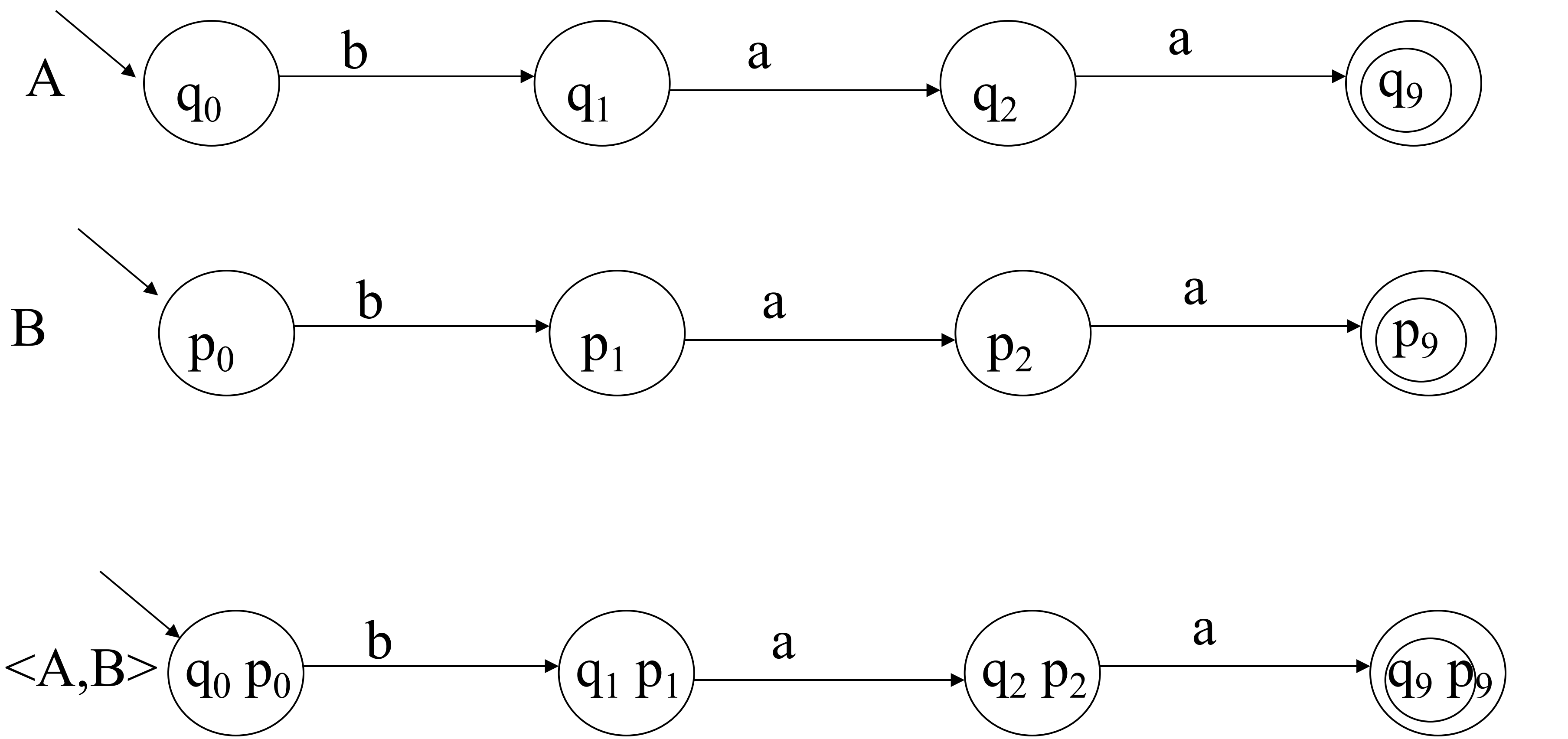
- $\mathcal{L} = \{L_i\}$: <u>famiglia</u> di linguaggi
- $\mathcal{L}$ è chiuso rispetto all'operazione OP se e solo se, per ogni L_1 , $L_2 \in \mathcal{L}$, L_1 OP $L_2 \in \mathcal{L}$.
- $\mathcal{R}$: linguaggi regolari (riconosciuti da FSA)
- <u>$\mathcal{R}$ è chiuso rispetto alle operazioni insiemistiche, alla concatenazione, a "*"</u>, …

Intersezione

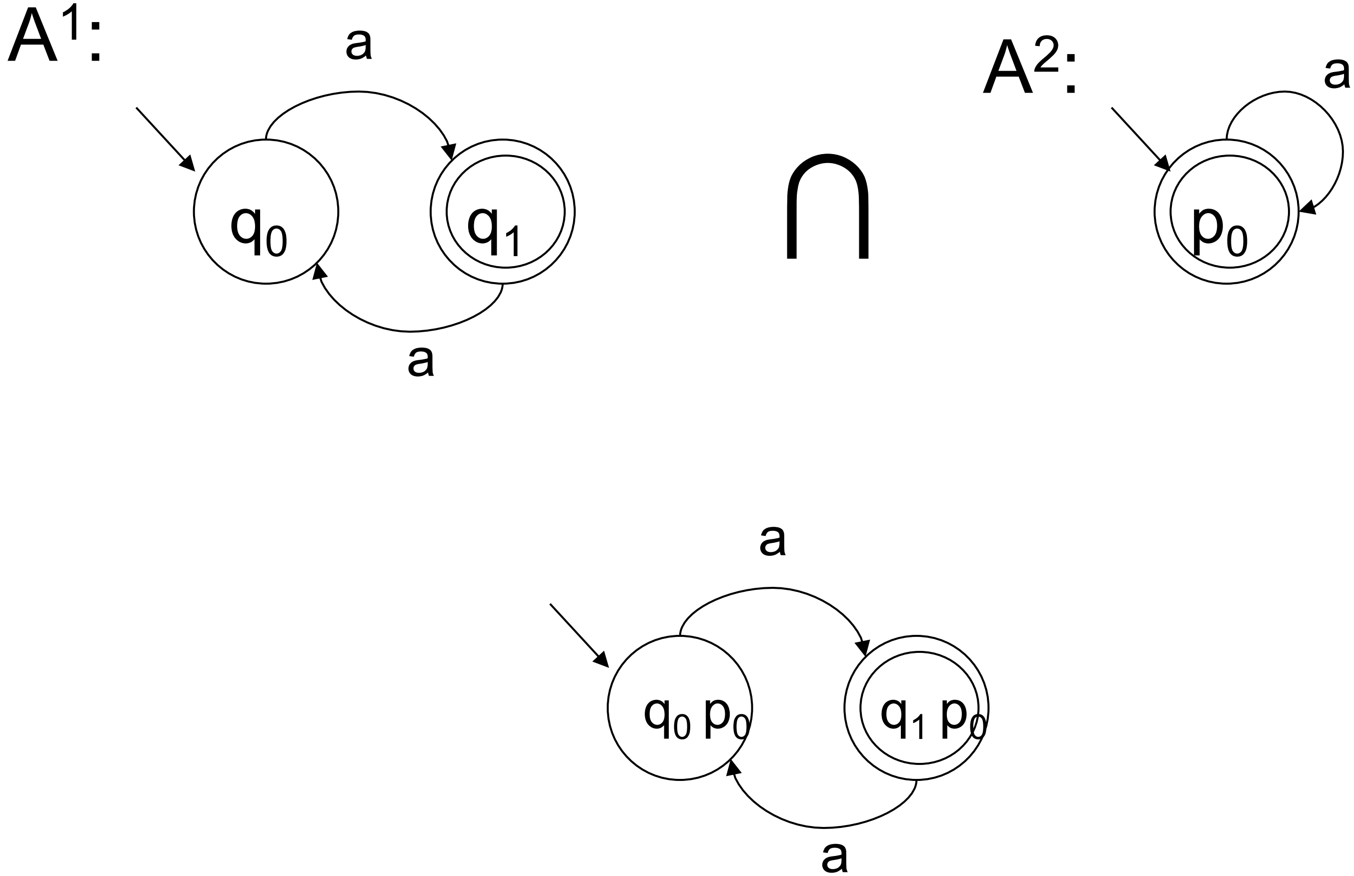


Intersezione

L'"esecuzione parallela" di A e B può essere simulata "accoppiandoli"



Esempio



Formalmente

- Dati
  - $A^1 = <Q^1, I, \delta^1, q_0^1, F^1>$
  - $A^2 = <Q^2, I, \delta^2, q_0^2, F^2>$

  $< A^1, A^2 > = <Q^1 \times Q^2, I, \delta, <q_0^1, q_0^2>, F^1 \times F^2 >$
  - $\delta(<q^1,q^2>, i) = <\delta^1(q^1, i), \delta^2(q^2, i)>$
- Si può mostrare (mediante semplice induzione) che

  $L(< A^1,A^2>) = L(A^1) \cap L(A^2)$
- Si può fare lo stesso per l'unione?

Unione

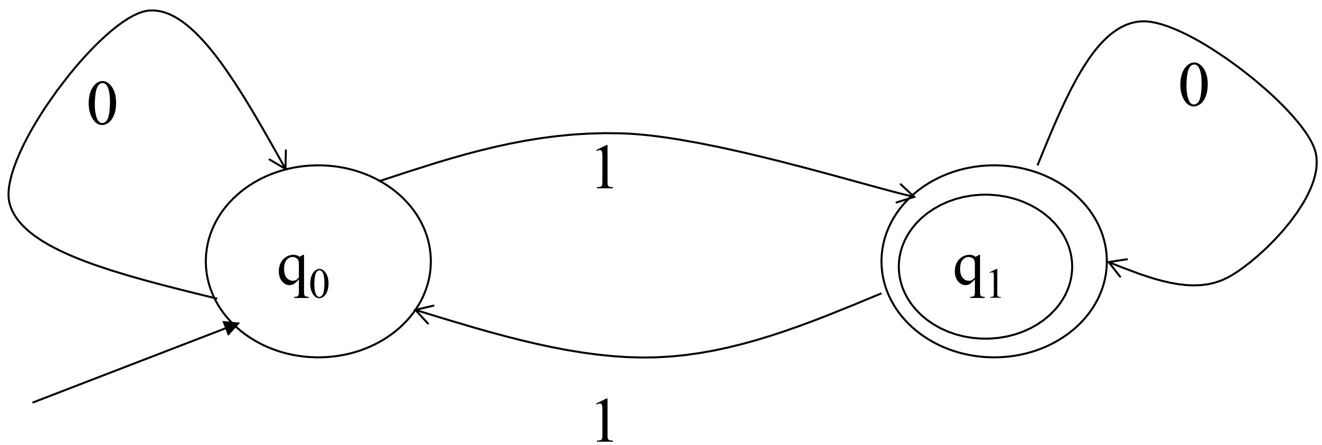
- L'unione è costruita analogamente
- Dati
  - $A^1 = <Q^1, I, \delta^1, q_0{}^1, F^1>$
  - $A^2 = <Q^2, I, \delta^2, q_0{}^2, F^2>$

$<A1, A2> = <Q^1 x Q^2, I, \delta, <q_0{}^1, q_0{}^2>, F^1 x Q^2 \cup Q^1 x F^2>$

  - $\delta(<q^1, q^2>, i) = <\delta^1(q^1, i), \delta^2(q^2, i)>$

Complemento (1)

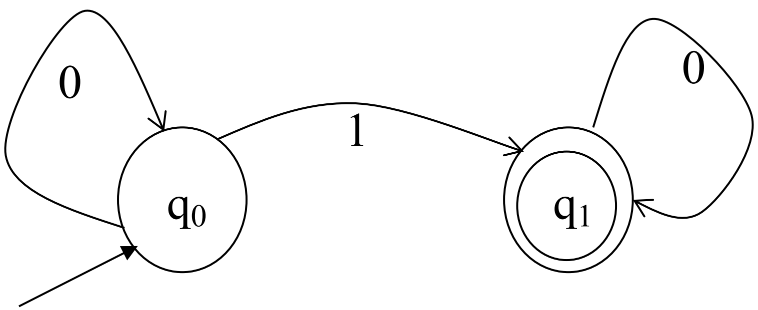
- Idea di base $F^c = Q-F$



… ma in generale la funzione di transizione è parziale!

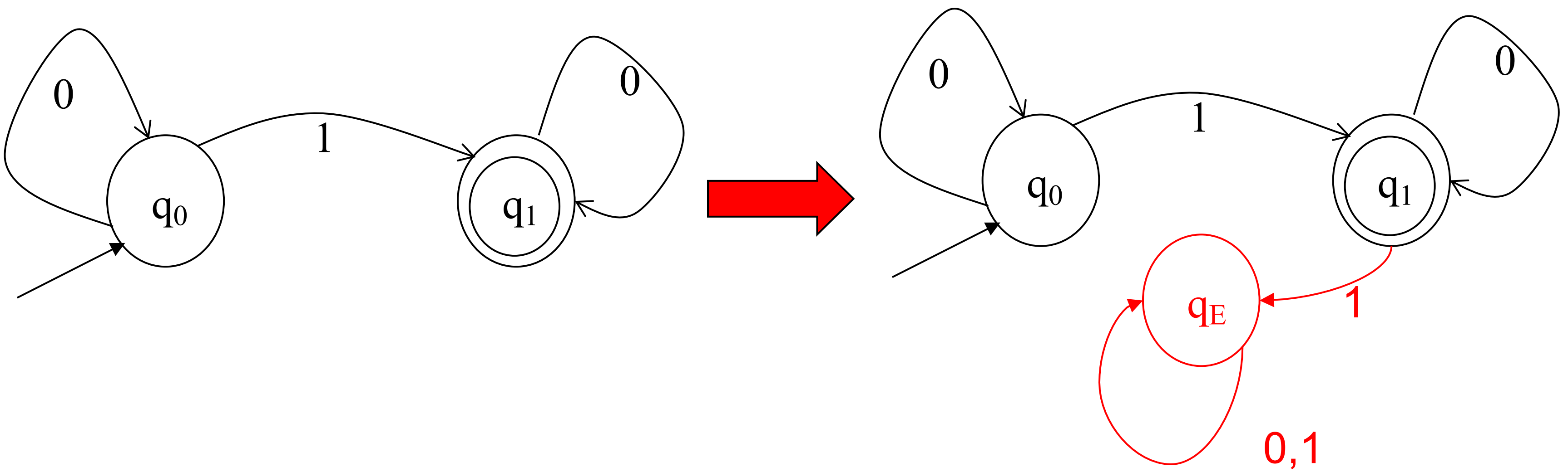
Complemento (2)

- Prima di scambiare stati finali e non finali è necessario completare l'FSA



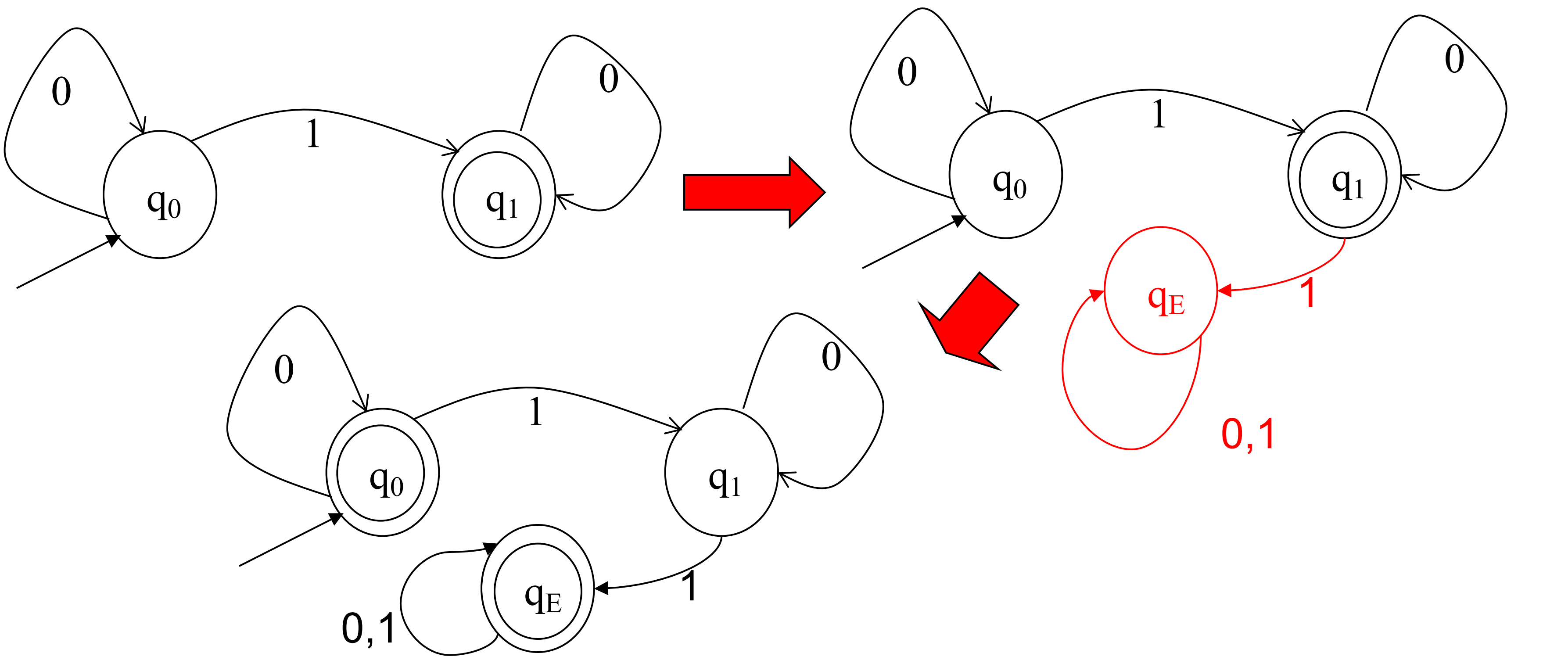
Complemento (2)

- Prima di scambiare stati finali e non finali è necessario completare l'FSA



Complemento (2)

- Prima di scambiare stati finali e non finali è necessario completare l'FSA



Ancora unione

- Un’altra possibilità è di esprimere l’unione in funzione della complementazione e dell’intersezione, mediante le leggi di De Morgan:

  $A \cup B = \neg(\neg A \cap \neg B)$

Filosofia del complemento

- Se scandisco l’intera stringa d’ingresso, allora basta “scambiare il sì con il no” (F con Q-F)
- Se non raggiungo la fine della stringa, allora lo scambio di F con Q-F non funziona
- Nel caso di FSA c’è un trucco semplice (completare l’FSA)
- In generale, però, non possiamo considerare equivalenti la risposta negativa a una domanda e la risposta positiva alla domanda opposta!