Un problema classico

- Un uomo viaggia con un lupo, una pecora ed un cavolo
- Vuole attraversare un fiume
- Ha una barca che può contenere solo l'uomo più uno dei suoi possedimenti,
- Il lupo mangia la pecora se lasciati soli insieme
- La pecora mangia il cavolo se lasciati soli insieme
- Come può attraversare fiume senza perdite?

Soluzione come Stringa

Mosse possono essere rappresentate da simboli

- Uomo attraversa con lupo (*l*)
- Uomo attraversa con pecora (*p*)
- Uomo attraversa con cavolo (*c*)
- Uomo attraversa con niente (*n*)

Soluzione come Stringa

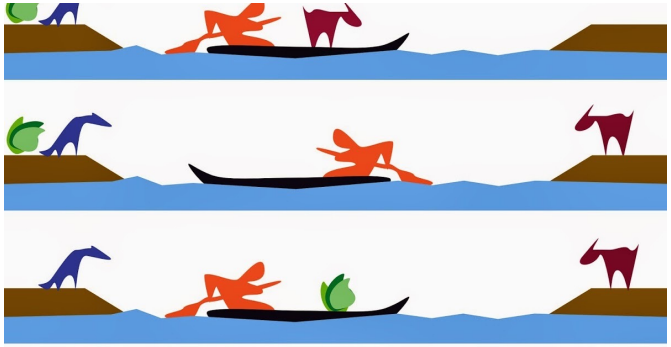
Mosse possono essere rappresentate da simboli

- Uomo attraversa con lupo (*l*)
- Uomo attraversa con pecora (*p*)
- Uomo attraversa con cavolo (*c*)
- Uomo attraversa con niente (*n*)

➔Sequenza mosse ⇔ stringa,

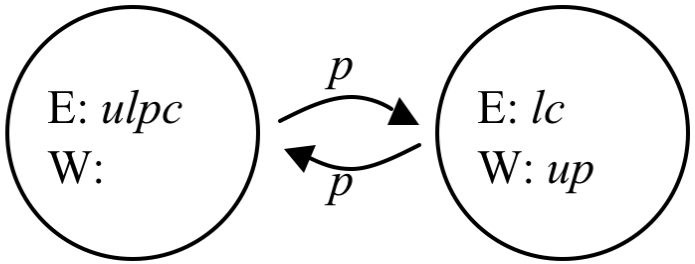
Es. soluzione *pncplnp*:

- **Prima attraversa *con pecora*,**
- **poi ritorna con *n*iente,**
- **poi attraversa con il cavolo , …**



Mosse ⇔transizioni di stato

- Ogni mossa porta da uno stato ad un’altro
- Es. Mossa p provoca transizione



Es. soluzione *pncplnp*:

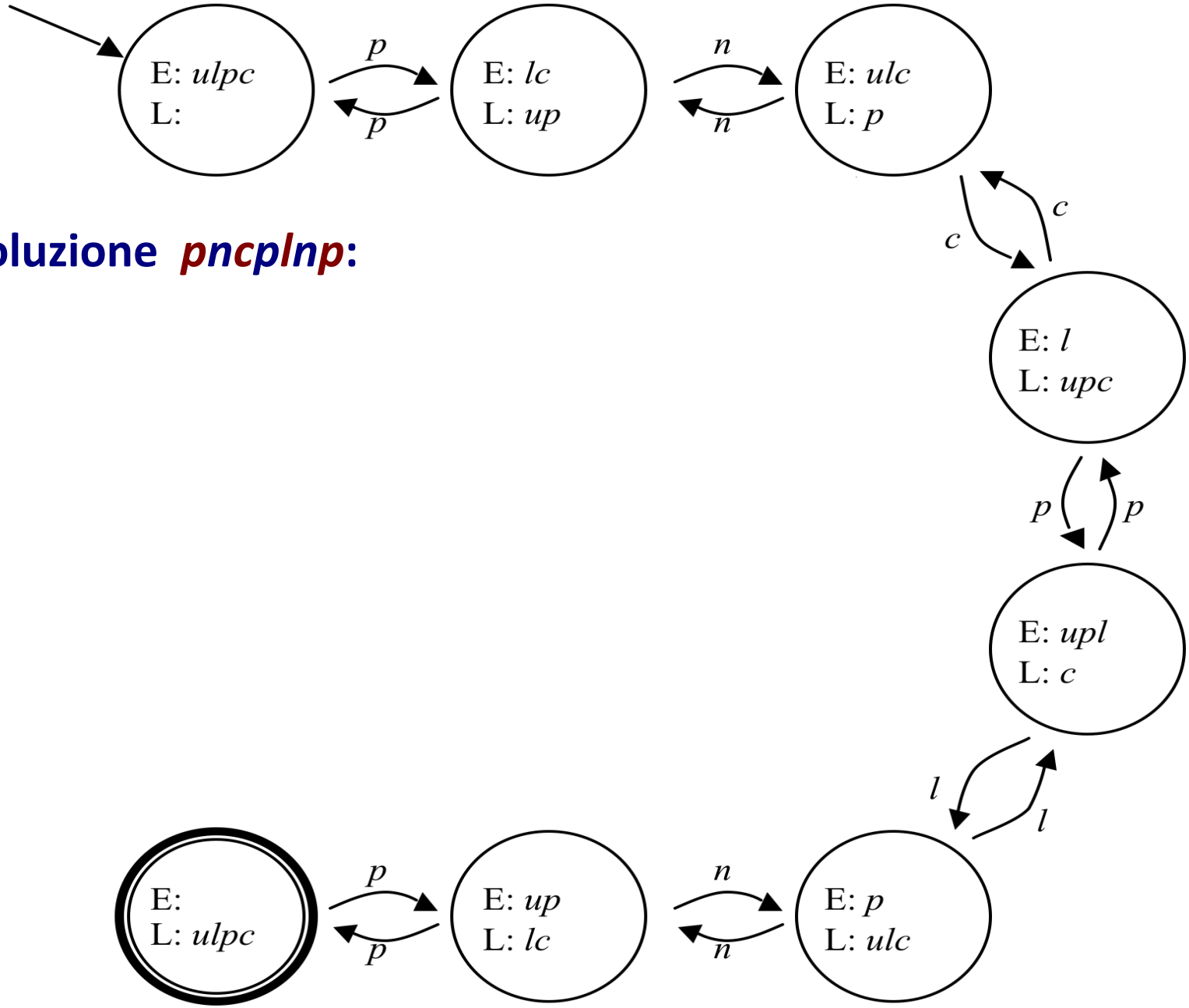
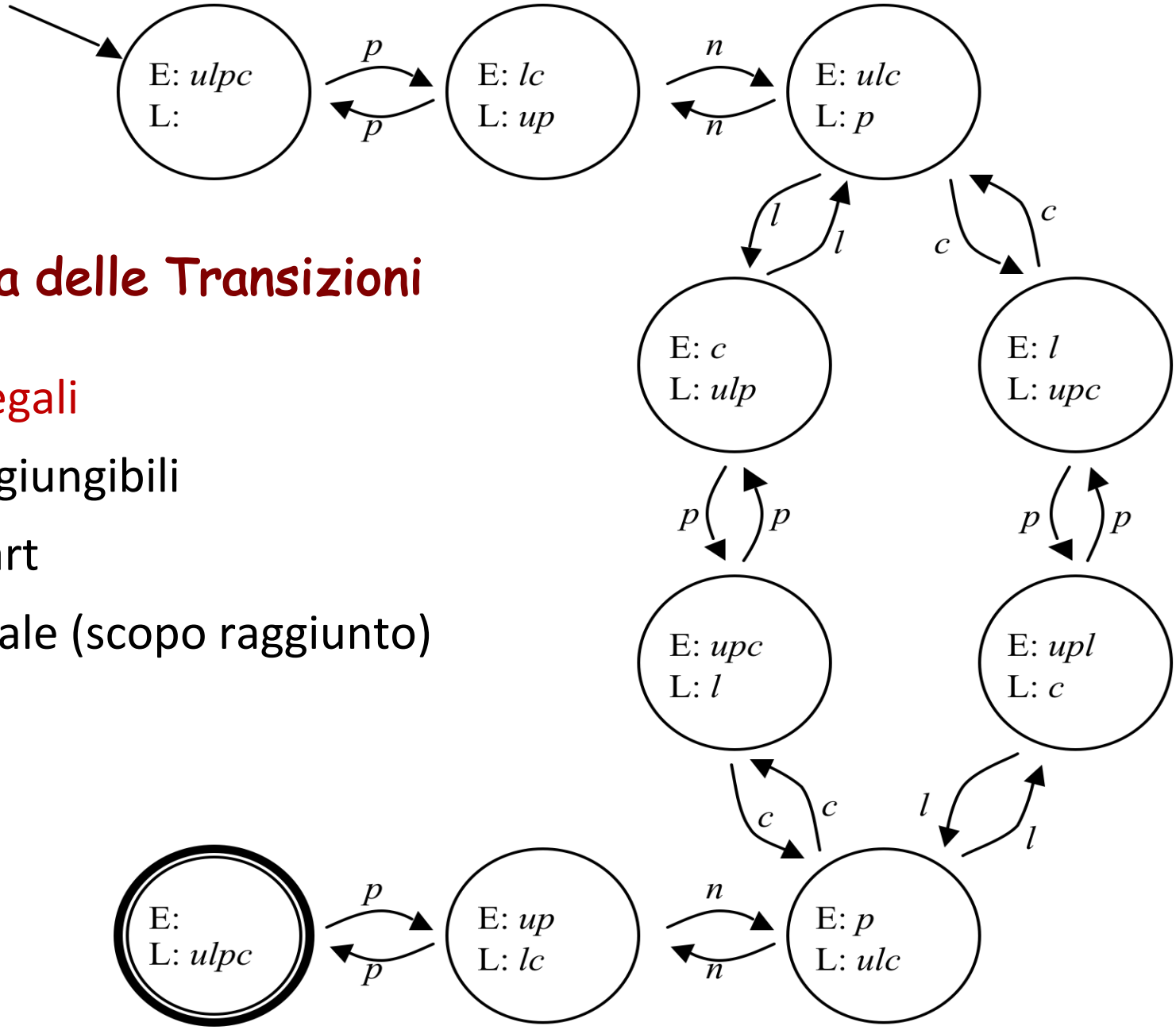
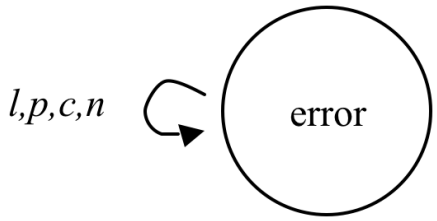


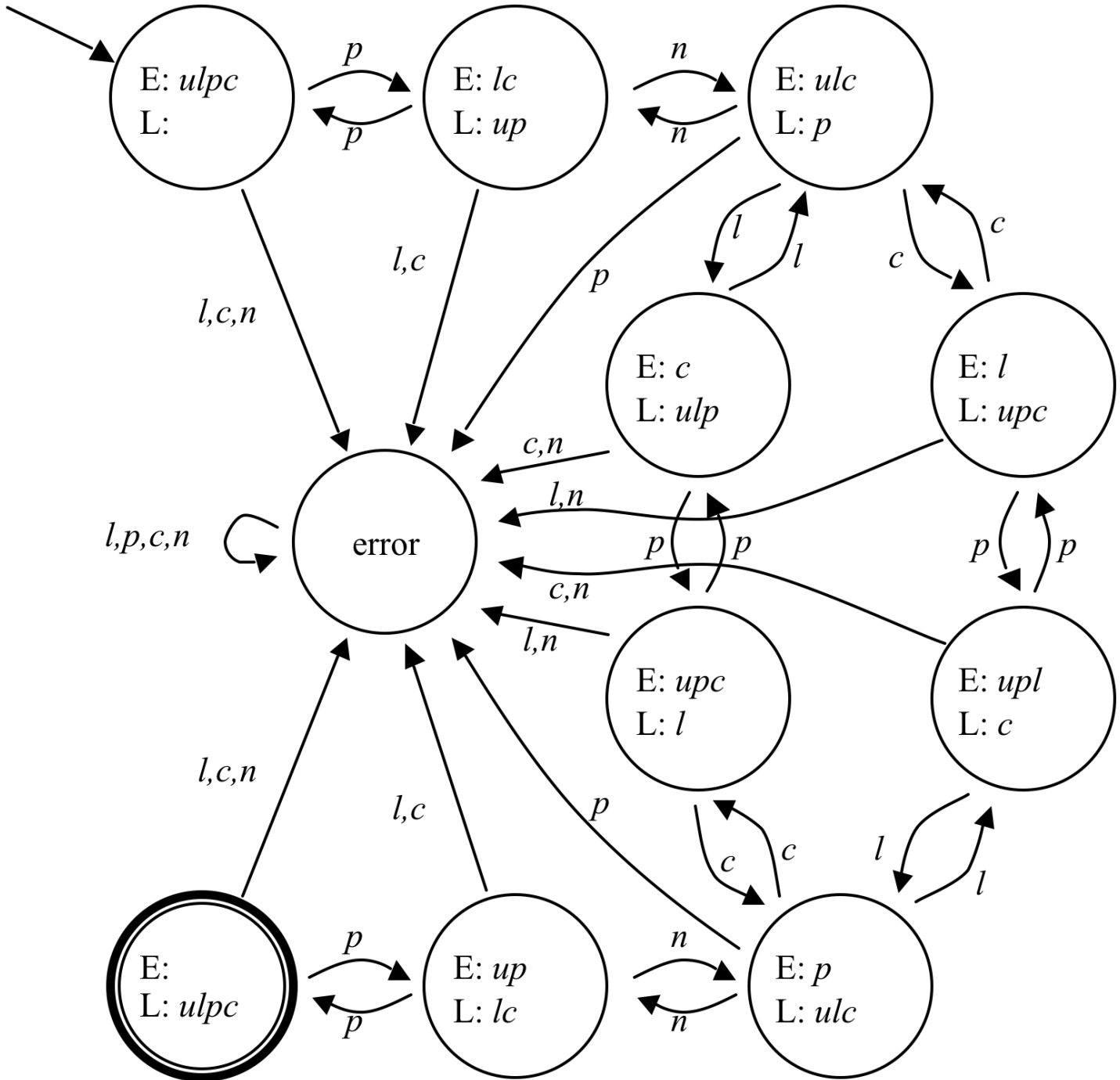
Diagramma delle Transizioni

- Mosse legali
- Stati raggiungibili
- Stato start
- Stato finale (scopo raggiunto)



- Alcune mosse non sono legali
- Usiamo stato addizionale




E: ulpc
L:
E: lc
L: up
E: ulc
L: p
E: c
L: ulp
E: l
L: upc
error
E: upc
L: l
E: upl
L: c
E:
L: ulpc
E: up
L: lc
E: p
L: ulc
p
p
n
n
l
l
c
c
l,c
l,c,n
p
c,n
l,n
l,p,c,n
p
p
p
p
c,n
l,n
l,c,n
l,c
p
c
c
l
l
p
p
n
n

Linguaggio delle soluzioni

- Sequenza mosse ⇔ Cammino ⇔ stringa $\boldsymbol{x} \in \{l,p,c,n\}^*$

- Sequenza di mosse è una soluzione

  ⇔

  stringa corrispondente è un cammino dallo stato iniziale allo stato finale del diagramma

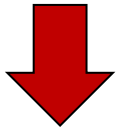
- diagramma definisce il ***linguaggio delle soluzioni***:

  $\{x \in \{l,p,c,n\}^*$ | iniziando in stato start e seguendo transizioni di x, terminano nello stato finale$\}$

- Nota: Linguaggio infinito

Nota: Dal Problema al Linguaggio

- **Problema logico**: trovare sequenza di mosse che permette di trasportare capra, cavolo e lupo



- Insieme delle stringhe sull'alfabeto {*l,p,c,n*}.

  Le stringhe corrispondono a sequenze di mosse.

  Tra esse cerchiamo una stringa che corrisponde ad una soluzione del problema.

  Linguaggio: Insieme delle stringhe corrispondenti ad una soluzione

- Nel diagramma esattamente una transizione per ogni stato e per ogni lettera nell'alfabeto Σ={n,l,c,p}

- Diagramma fornisce **procedura computazionale** per decidere se una stringa è una soluzione o meno:
  - Parti nello stato start
  - Segui una transizione per ogni simbolo input
  - Se alla fine arrivi in stato obiettivo accetta altrimenti rifiuta l'input

AUTOMI FINITI

Modello semplice di calcolatore avente una quantità finita di memoria
E' noto come **macchina a stati finiti** o **automa finito**.

Idea di base del funzionamento:

- Input= stringa w su un alfabeto Σ
- Legge i simboli di w da sinistra a destra.
- Dopo aver letto l'ultimo simbolo, l'automa indica se accetta o rifiuta la stringa input w

Alla base di importanti tipi di software, es.:

✓ Compilatori: realizzazione del **parser**, analisi lessicale

Il Parsing o analisi sintattica è il processo di analizzare una stringa in accordo ad una grammatica.

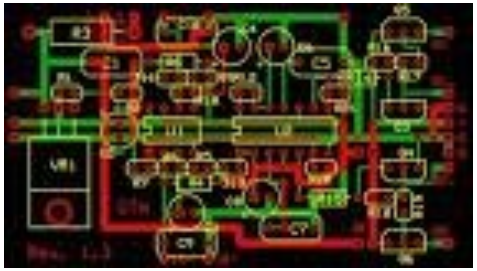
Un parser è un programma, di solito parte di un compilatore, che

- *riceve un input sotto forma di istruzioni di un programma (o comandi online interattivi, tag di codice, …) e*
- *lo divide in parti*
  (per esempio, i sostantivi (oggetti), verbi (metodi) e loro attributi o opzioni)
  che possono poi essere gestiti da altri programmi (per esempio, altri componenti di un compilatore).

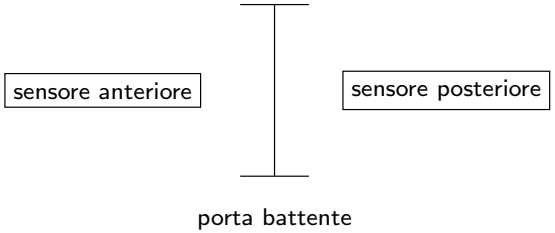
Un parser può anche verificare che è stato fornito tutto l'input necessario.

Alla base di importanti tipi di software, es.:

- ✓ Compilatori: realizzazione del parser, analisi lessicale
- ✓ Progettazione di circuiti digitali



Controllo porta automatica



Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Controllo porta automatica

Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Controllo porta automatica

Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

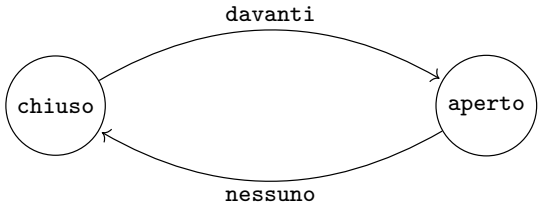
Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Regola:

- se la porta è chiusa, si apre solo se arriva una persona davanti;
- se la porta è aperta, si chiude solo se non c'è nessuno.

Controllo porta automatica



Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

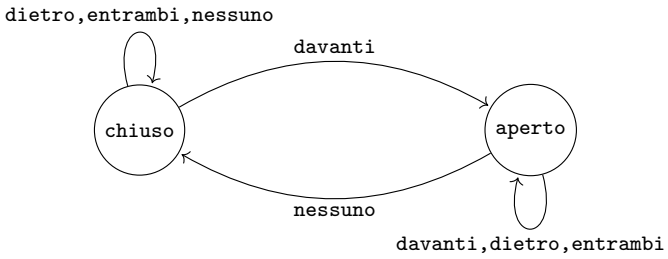
Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Regola:

- se la porta è chiusa, si apre solo se arriva una persona davanti;
- se la porta è aperta, si chiude solo se non c'è nessuno.

Controllo porta automatica



Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

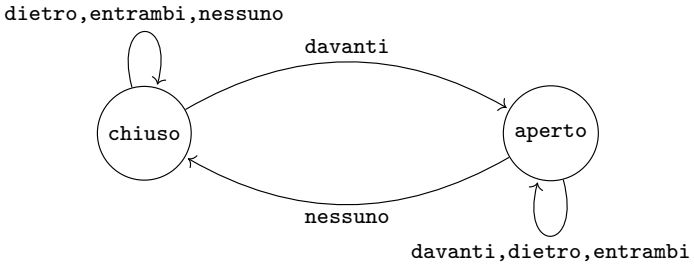
Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Regola:

- se la porta è chiusa, si apre solo se arriva una persona davanti;
- se la porta è aperta, si chiude solo se non c'è nessuno.

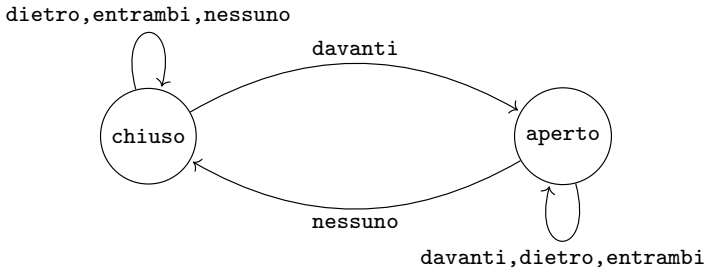
Controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati:
`chiuso`,

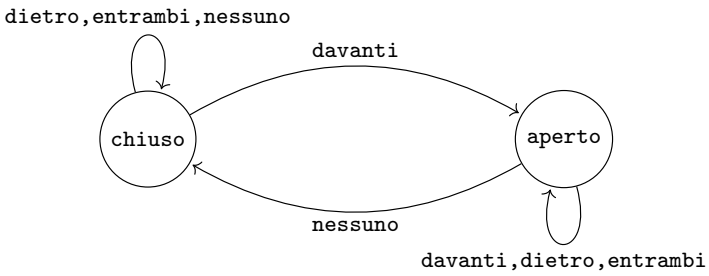
Controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati:
`chiuso`, `aperto`,

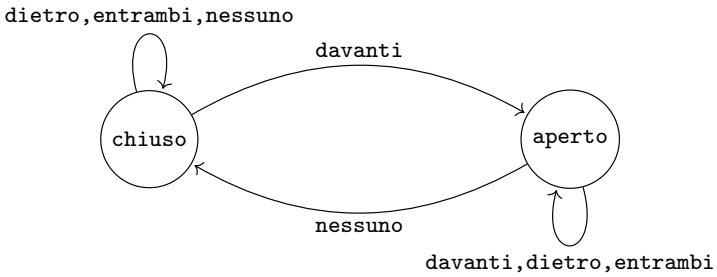
Controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati:
`chiuso`, `aperto`, `aperto`,

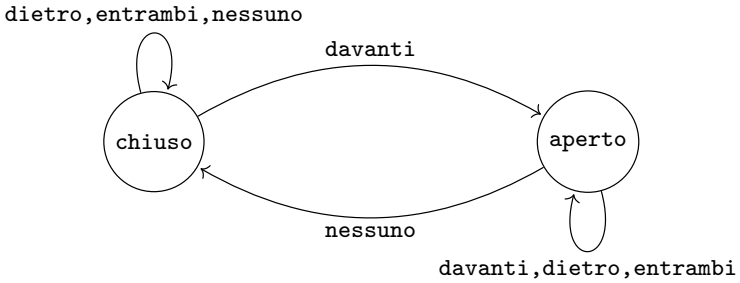
Controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati:
`chiuso`, `aperto`, `aperto`, `chiuso`,

Controllo porta automatica

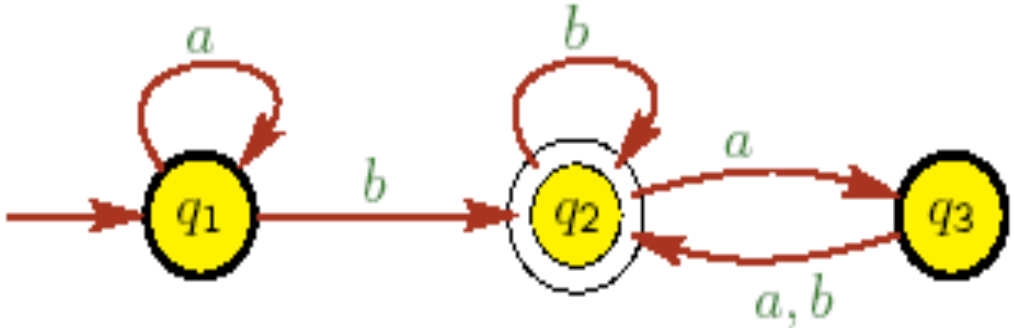


Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati:
`chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`, `aperto`, `chiuso`, `chiuso`, `chiuso`.

Automa finito deterministico (DFA)

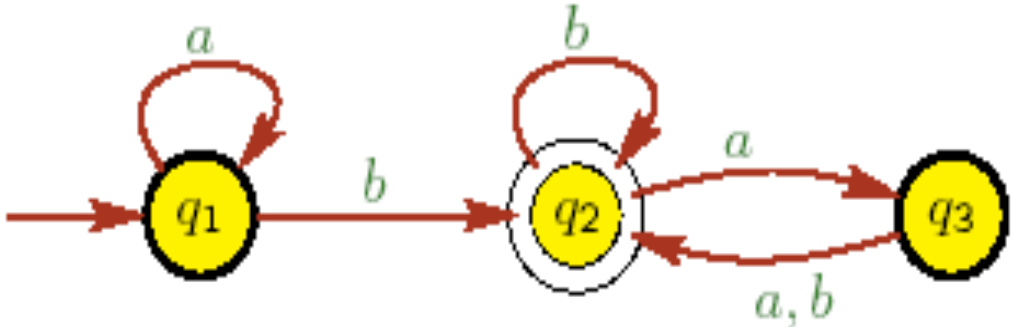
Es: DFA con alfabeto Σ ={a, b}:



Stringa arriva in input,
DFA legge 1 simbolo alla volta dal primo all'ultimo
Quindi DFA accetta o rifiuta la stringa

Automa finito deterministico (DFA)

Es: DFA con alfabeto Σ={a, b}:



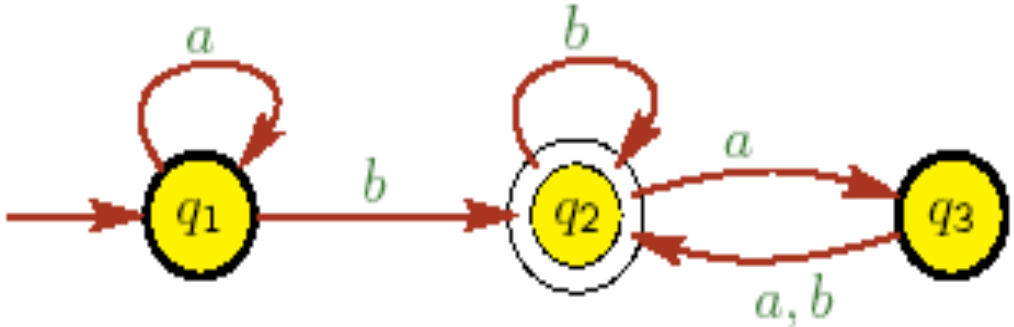
a b b a a

---

q_1 q_1 q_2 q_2 q_3 q_2

Automa finito deterministico (DFA)

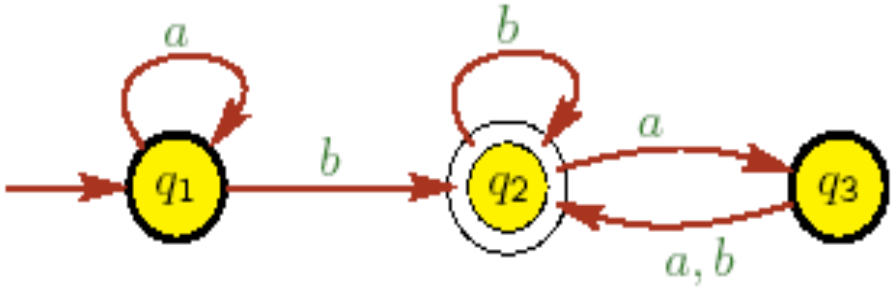
Es: DFA con alfabeto Σ={a, b}:



q_1, q_2, q_3 soni gli **stati**.

q_1 è lo **stato start** (ha freccia entrante da esterno)

q_2 è uno **stato accetta** (doppio cerchio)



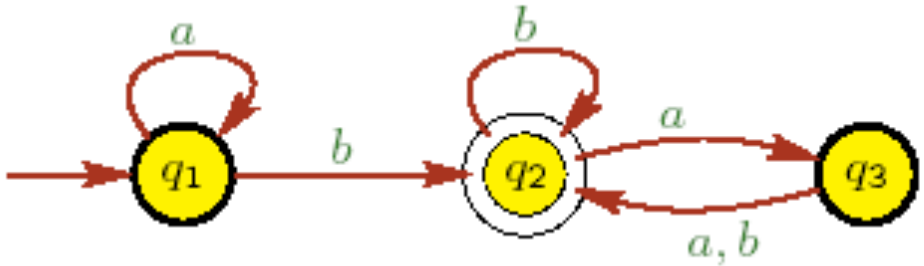
Archi dicono come muoversi quando l'automa si trova in uno stato e è letto un simbolo di Σ.

Dopo aver letto l'ultimo simbolo:

Se DFA è in uno stato accetta, allora stringa è **accettata,**
altrimenti è **rifiutata** .

Nell'esempio:

- ***abaa* è accettata**
- ***aba* è rifiutata**
- **ε è rifiutata**



Def. formale di DFA

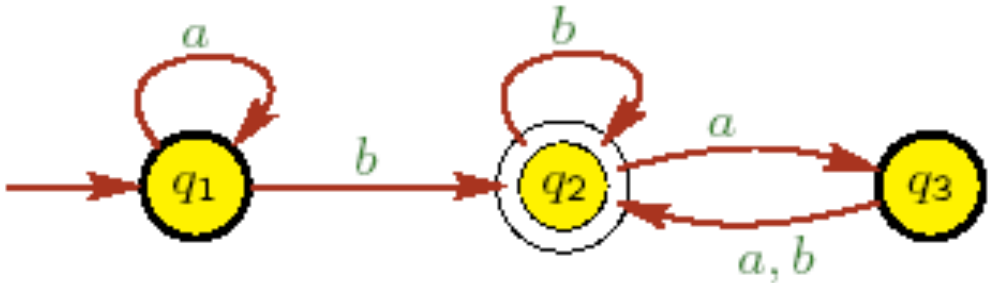
A **automa finito deterministico** (DFA) è 5-tupla

$M = (Q, \Sigma, f, q_1, F)$,

dove

1. Q è **insieme finito** di stati.
2. Σ è alfabeto, e il DFA processa stringhe **su** Σ.
3. $f : Q \times \Sigma \rightarrow Q$ è la funzione di transizione.
4. $q_1 \in Q$ è lo **stato start**.
5. F (sottoins. di Q) è l'insieme di **stati accetta** (o **finali**).

Nota: DFA chiamato anche semplicemente **automa finito (FA)**.



Funzione di transizione di DFA

funzione di transizione $f : Q \times \Sigma \rightarrow Q$:

per ogni stato e per ogni simbolo di input,

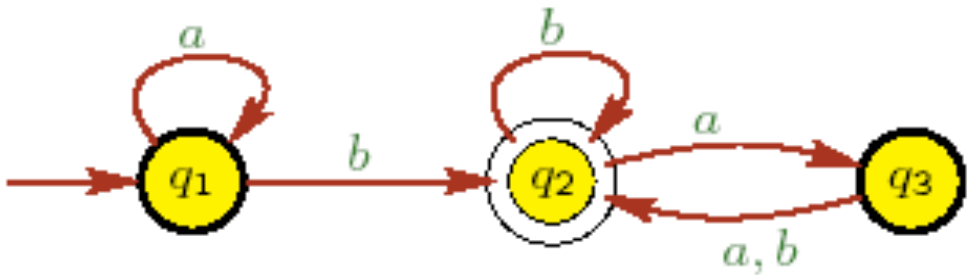
la funzione f dice in quale stato spostarsi.

Cioè, f(r, a) è lo stato che il DFA occupa quando trovandosi nello stato r legge a,

per stato $r \in Q$ ed ogni simbolo $a \in \Sigma$

Esiste esattamente un arco uscente da r con label a

quindi la macchina è **deterministica**.



Es. di DFA

M = (Q,Σ, f, q_1, F) con

Q = {q_1, q_2, q_3}

Σ={a, b}

f è descritta da

	a	b
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

q_1 è lo stato start

F = {q_2}.

Come funziona un DFA

- **Riceve in input stringhe di simboli su alfabeto A**

  Inizia nello stato start.

  Legge la stringa, un simbolo alla volta partendo da sinistra,

  il simbolo letto determina il successivo stato visitato.

- **Processo termina dopo lettura dell'ultimo simbolo**

  Dopo aver letto l'intera stringa input:

  Se DFA termina in stato accetta, allora stringa è accettata

  altrimenti , è rifiutata .

Definizione formale funzionamento

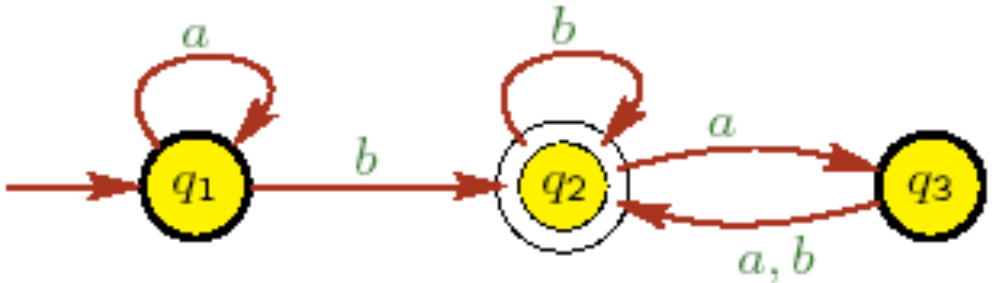
Sia $M = (Q, \Sigma, f, q_0, F)$ un DFA.

Consideriamo la stringa $w = w_1 w_2 \ldots w_n$ su Σ, dove ogni w_i in Σ.

M **accetta** w se esiste sequenza stati $r_0, r_1, \ldots, r_n$ di Q
tali che:

1. $r_0 = q_0$ (primo stato della sequenza è quello iniziale)
2. r_n in F (ultimo stato in sequenza è uno stato accetta)
3. $f(r_{i-1}, w_i) = r_i$, per ogni $i = 1, 2, \ldots, n$

(sequenza di stati corrisponde a transizioni valide per la stringa w).



ES.

abaa accettata

⇔

Esiste sequenza stati q_1,q_1,q_2,q_3,q_2

Linguaggio della Machina

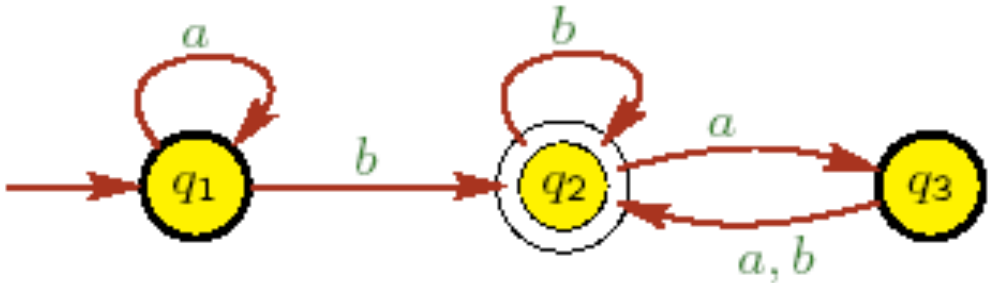
Def: Se L è l'insieme di tutte le stringhe che la machina M accetta, allora si dice che

- L è il **Linguaggio della machina** M, o analogamente
- M **riconosce** (o **accetta**) L

Scriviamo anche L(M) = L.

Def:

Un Linguaggio è **regolare** se è riconosciuto da qualche DFA.

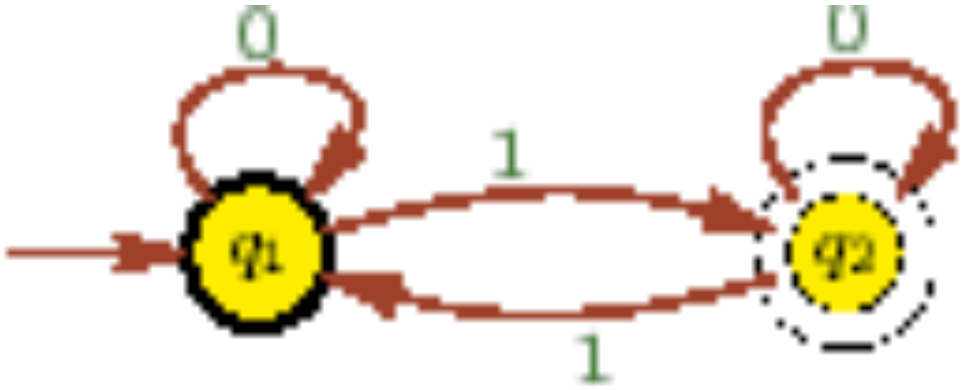


ES.

L(M) è l'insieme di tutte le stringhe sull'alfabeto {a,b} della forma

$\{a\}^*\{b\}\{b,aa,ab\}^*$

Es: Si consideri il seguente DFA M_1 con alfabeto $\Sigma=\{0, 1\}$:



010110 è accettata , ma 0101 è rifiutata .

$L(M_1)$ è il Linguaggio di stringhe su Σ in cui il numero totale di 1 è dispari.

Esercizio: dare DFA che riconosce il Linguaggio di stringhe su Σ con un numero pari di 1

Es: DFA M_2 con alfabeto $\Sigma=\{0, 1\}$:

Es: DFA M_2 con alfabeto $\Sigma=\{0, 1\}$:

$L(M_2)$ è Linguaggio su Σ delle stringhe di lunghezza 1, cioè

$$L(M_2) = \{w \text{ in } \Sigma^* \mid |w| = 1\} = \Sigma$$

Es: DFA M_2 con alfabeto $\Sigma=\{0, 1\}$:

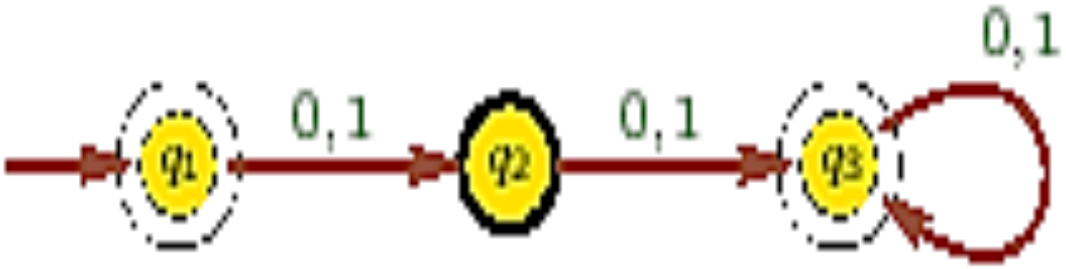
$L(M_2)$ è Linguaggio su Σ che ha lunghezza 1, cioè

$$L(M_2) = \{w \text{ in } \Sigma^* \mid |w| = 1\}$$

Si ricordi che $C(L(M_2))$, il complemento di $L(M_2)$, è l'insieme di stringhe su A che non sono in $L(M_2)$.

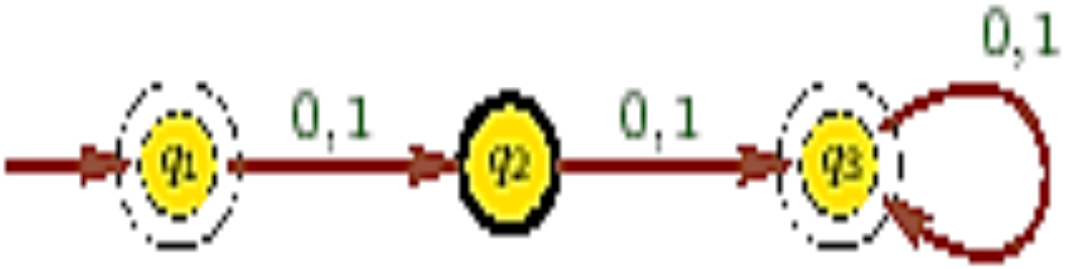
Domanda: DFA che riconosce $C(L(M_2))$?

Es: Si consideri il seguente DFA M_3 con alfabeto $\Sigma = \{0, 1\}$



$L(M_3)$ è il Linguaggio su Σ che **non** ha lunghezza 1,

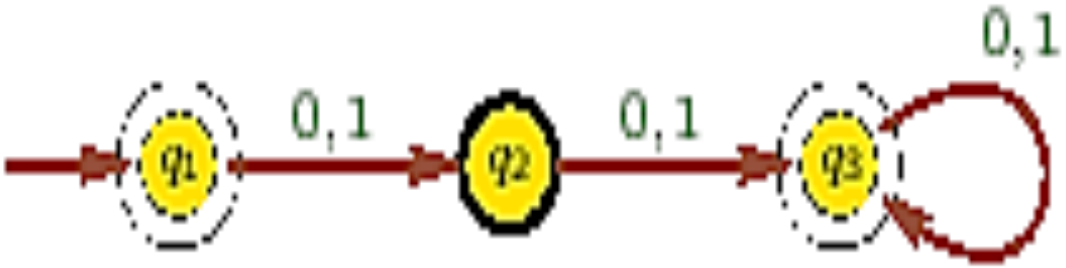
Es: Si consideri il seguente DFA M_3 con alfabeto $\Sigma = \{0, 1\}$



$L(M_3)$ è il Linguaggio su Σ che **non** ha lunghezza 1,

- Più di uno stato accetta.

Es: Si consideri il seguente DFA M_3 con alfabeto $\Sigma = \{0, 1\}$



$L(M_3)$ è il Linguaggio su Σ che **non** ha lunghezza 1,

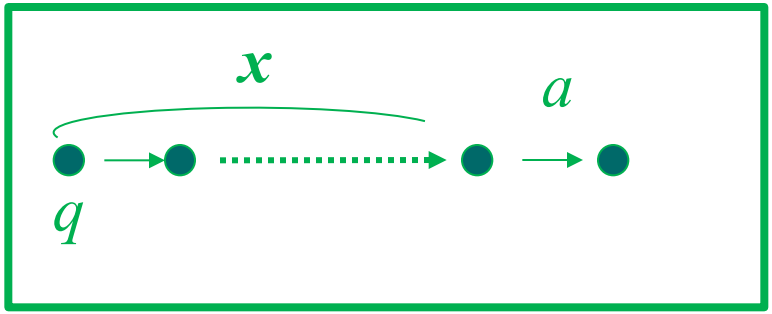
- Più di uno stato accetta.
- Stato start anche stato accetta.

DFA accetta ϵ ⇔ stato start è anche stato accetta.

Funzione di transizione estesa

La funzione di transizione f puo' essere estesa a f* che opera su stati e stringhe (invece che su stati e simboli)

$f^*(q,\varepsilon) = q$
$f^*(q,xa) = f(f^*(q, x), a)$

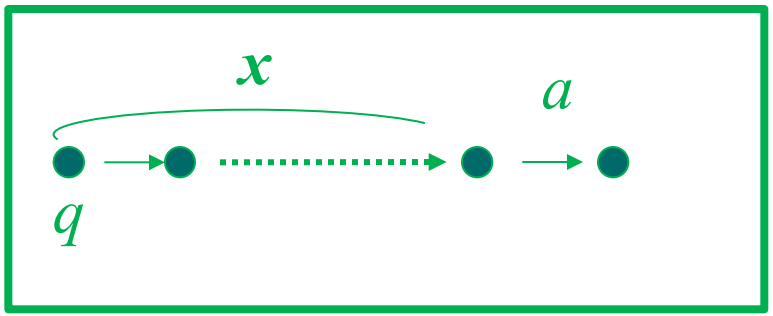


Funzione di transizione estesa

La funzione di transizione f puo' essere estesa a f* che opera su stati e stringhe (invece che su stati e simboli)

$f^*(q,\varepsilon) = q$

$f^*(q,xa) = f(f^*(q, x), a)$



$f^*(q_0, w) \in F \Leftrightarrow M$ accetta w

Il linguaggio accettato da M è quindi

$$L(M) = \{ w : f^*(q_0, w) \in F \}$$

ESERCIZI

Fornire DFA per i seguenti linguaggi sull'alfabeto {0, 1}:

1. Insieme di tutte le stringhe che terminano con 00

2. Insieme di tutte le stringhe con tre zeri consecutivi

3. Insieme delle stringhe con 011 come sottostringa

4. Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01

DFA per Complemento

In generale, dato DFA M per Linguaggio L,

possiamo costruire DFA M' per C(L) da M

- rendendo tutti gli stati accetta in M non-accetta in M',
- rendendo tutti stati non-accetta in M stati accetta in M'.

Più formalmente,

Se Linguaggio L su alfabeto Σ ha un DFA

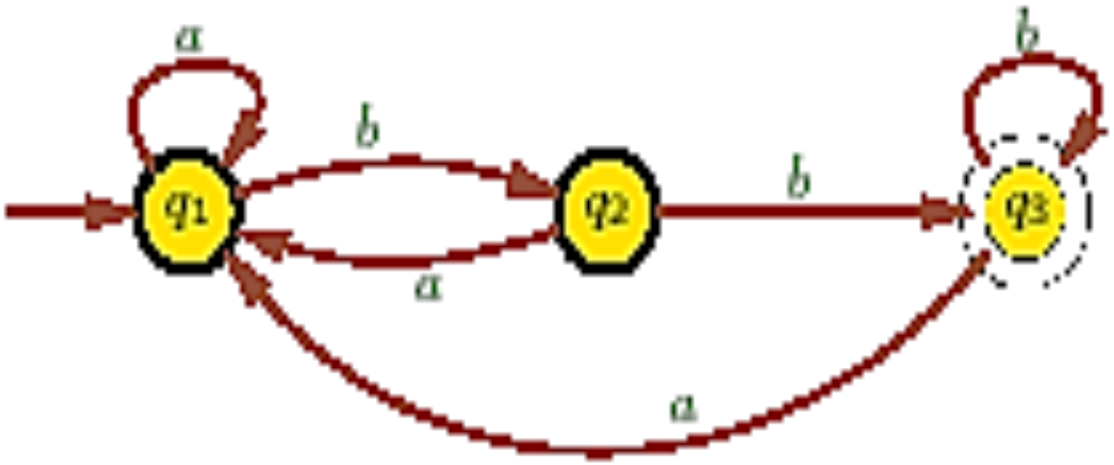
$M = (Q, \Sigma, f, q_1, F)$.

allora, DFA per il complemento di L è

$$M = (Q, \Sigma, f, q_1, Q-F).$$

Esercizio: Perchè funziona?

Es: Si consideri il seguente DFA M_4 con alfabeto $\Sigma=\{a, b\}$:

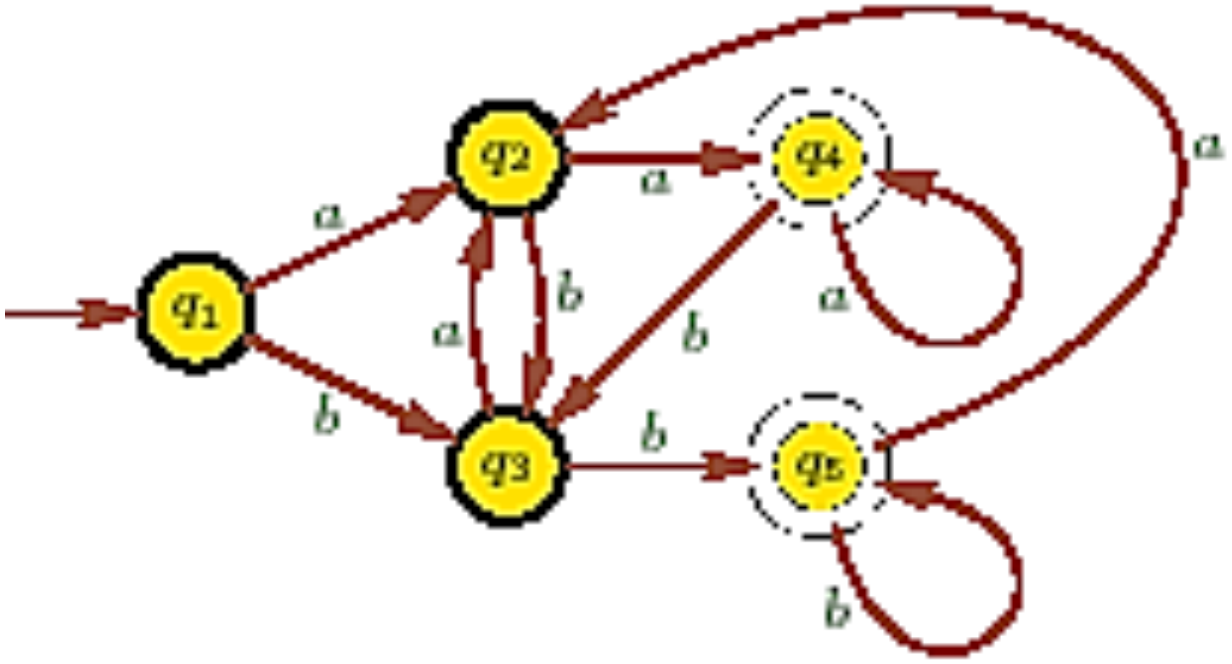


$L(M_4)$: Linguaggio di stringhe su Σ che terminano con bb,

Cioè

$L(M_4) = \{w \text{ in } \Sigma \mid w = sbb \text{ per qualche stringa } s\}$

Es: Si consideri il seguente DFA M_5 con alfabeto $\Sigma=\{a, b\}$



$L(M_5) = \{w \mid w = saa$ oppure $w = sbb$ per qualche stringa $s\}$.

Si consideri il seguente DFA M_6 con alfabeto $\Sigma=\{a,b\}$

Accetta tutte le possibili stringhe su Σ, cioè

$L(M_6) = \Sigma^*$

In generale, ogni DFA in cui tutti stati sono stato accetta riconosce il linguaggio Σ^*

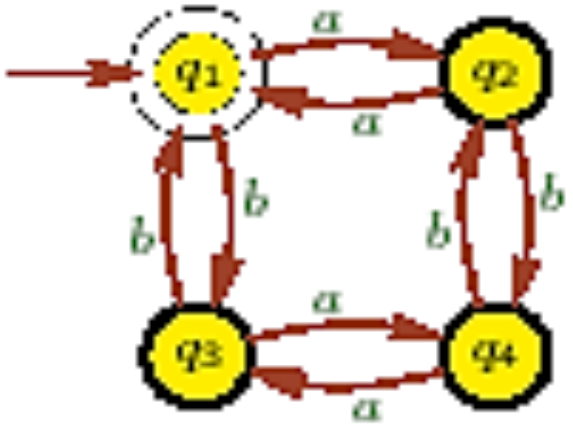
Si consideri il seguente DFA M6 con alfabeto Σ={a,b}

DFA non accetta stringhe su Σ, cioè

$L(M7) = \phi$

In generale, un DFA può non avere stati accetta

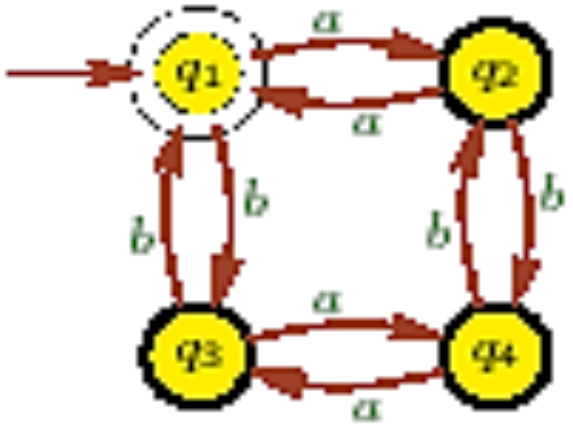
Es: Si consideri il seguente DFA M_8 con alfabeto $\Sigma=\{a, b\}$:



- Ogni *a* muove verso destra o sinistra.
- Ogni *b* muove verso l'alto o il basso
- DFA riconosce il Linguaggio di stringhe su Σ

…

Es: Si consideri il seguente DFA M_8 con alfabeto $\Sigma=\{a, b\}$:



- Ogni a muove verso destra o sinistra.
- Ogni b muove verso l'alto o il basso
- DFA riconosce il Linguaggio di stringhe su Σ con numero pari di a e numero pari di b.