UNIVERSITÀ
DEGLI STUDI
DI TORINO

Proprietà dei linguaggi regolari

a.a. 2018-2019

Riferimenti bibliografici

Automi, Linguaggi e Calcolabilità
J. E. Hopcroft, R. Motwani, J. D. Ullman.

Proprietà dei linguaggi regolari [cap. 4]

4.1 Dimostrare che un linguaggio non è regolare

4.1.1 Il pumping lemma per i linguaggi regolari

4.2 Proprietà di chiusura dei linguaggi regolari (pag.124)

Compilatori: principi, tecniche e strumenti
A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman

Analisi lessicale [cap.3]

3.9.6 Minimizzazione del numero degli stati di un DFA

Proprietà dei linguaggi regolari

Proprietà Pumping.

Ogni linguaggio regolare soddisfa una proprietà caratteristica, chiamata pumping.

In presenza di un falso linguaggio regolare, l'uso di tale proprietà permette di ottenere una contraddizione.

Proprietà di chiusura.

Operazioni che applicate a linguaggi regolari forniscono linguaggi regolari, cioè operazioni rispetto alle quali la classe dei linguaggi regolari è chiusa.

Automa minimo.

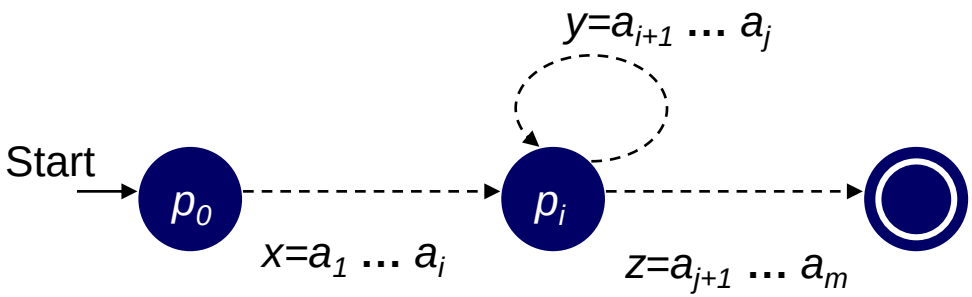
Per ogni linguaggio regolare esiste un DFA che lo riconosce con un numero di stati minimo e tale DFA è unico.

Proprietà Pumping

Sia *L* un linguaggio regolare e sia *n* il numero di stati di un automa deterministico a stati finiti che lo riconosce.

Si consideri una stringa w di lunghezza *m*, maggiore o uguale a *n* accettata dall'automa, se esiste, cioè una stringa di *L.*

Poichè m > n, durante l'esame della stringa almeno uno stato deve occorrere due volte.



Possiamo scrivere w come la concatenazione delle tre stringhe:

$x = a_1a_2 \ldots a_i$ $y = a_{i+1}a_{i+2} \ldots a_j$ $z = a_{j+1}a_{j+2} \ldots a_m$

$w = xyz$

Quindi tutte le stringhe $xy^kz \in L$, per ogni $k \geq 0$.

Pumping Lemma

Teorema

Sia L un linguaggio regolare.

Allora $\exists n$, che dipende solo dal linguaggio, tale che $\forall w \in L$, $|w| \geq n$, w si può scrivere come la concatenazione di tre sottostringhe xyz tali che:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. $\forall k \geq 0$, $xy^k z \in L$

Proprietà di chiusura

Siano L e M due linguaggi regolari.

Allora i seguenti linguaggi sono regolari:

- Unione: $L \cup M$
- Concatenazione: $L.M$
- Chiusura: L^*
- Complemento: $\overline{L}$
- Differenza: $L - M$
- Inversione: $L^R = \{w^R \mid w \in L\}$
- Intersezione: $L \cap M$

Proprietà di chiusura

La chiusura rispetto a unione, concatenazione e chiusura di Kleene discende direttamente dalla definizione di espressione regolare.

La chiusura rispetto alle operazioni di complemento, differenza, inversione e intersezione può essere facilmente provata ad esempio dimostrando che, dati due automi deterministici che riconoscano i linguaggi L e M, rispettivamente, è possibile costruire automi a stati finiti che riconoscano:

$\overline{L}$

$L^R = \{w^R \mid w \in L\}$

$L \cap M$

$L - M$

Proprietà di chiusura

Esempio: Chiusura per complementazione

Sia $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ deterministico con $L(A) = M$

Costruiamo $A_c = \langle Q_c, \Sigma, \delta_c, q_c, F_c \rangle$ in modo che: $L(A_c) = \Sigma^* - M$

$Q_c = Q$

$q_c = q_0$

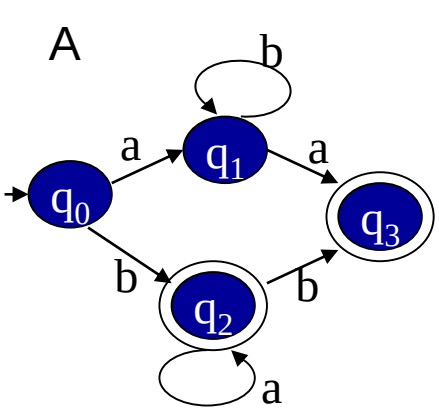
$\forall q \in Q, \forall a \in \Sigma : \quad \delta_c(q, a) = \delta(q, a)$

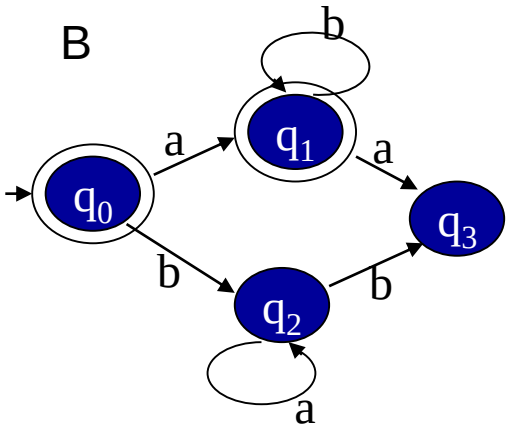
$F_c = Q - F$

N.B.: l'automa A deve essere completato con lo stato di errore (funzione di transizione totale su Q x Σ)

Proprietà di chiusura

Esempio: Chiusura per complementazione

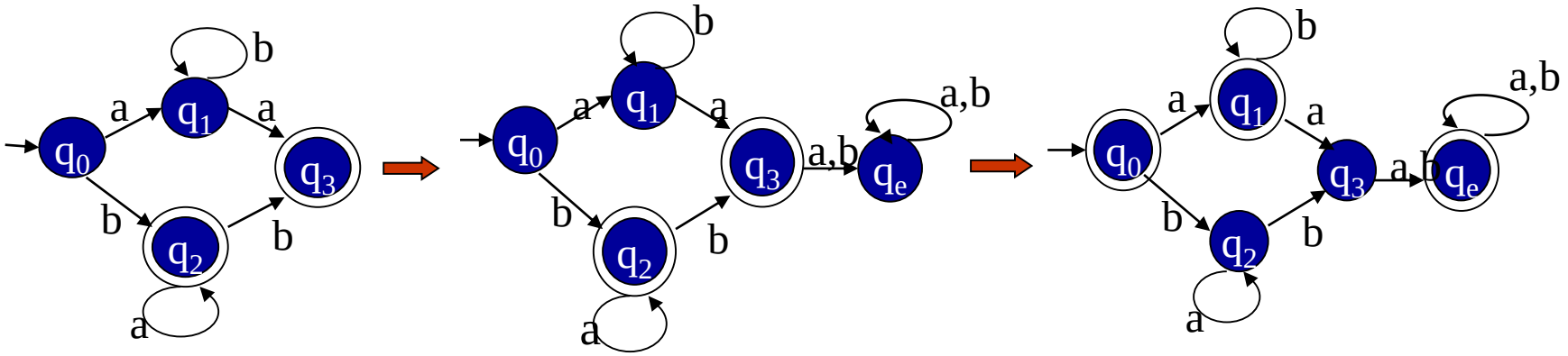




bba $\in$ L((a+b)* - L(A))

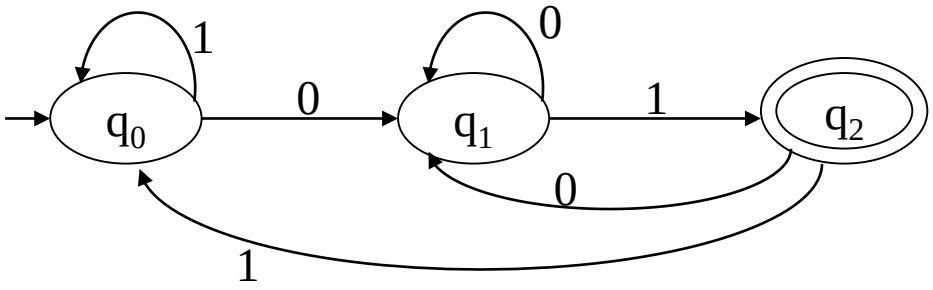
bba $\notin$ L(B)

L'automa costruito non riconosce il linguaggio complemento perché nell'automa di partenza manca lo stato d'errore.

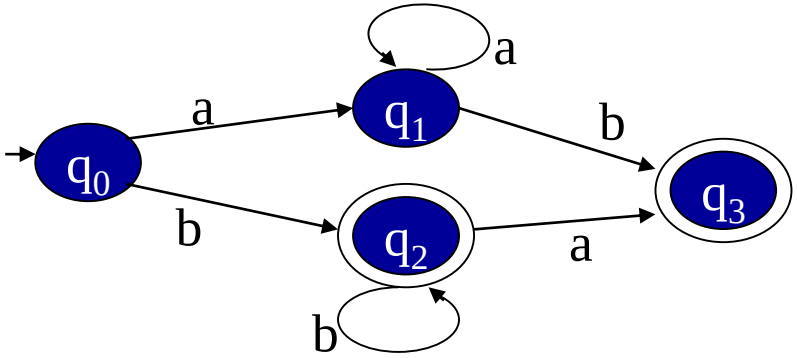


Esercizi

- Costruire un automa che riconosca il complemento del linguaggio riconosciuto dal seguente automa:

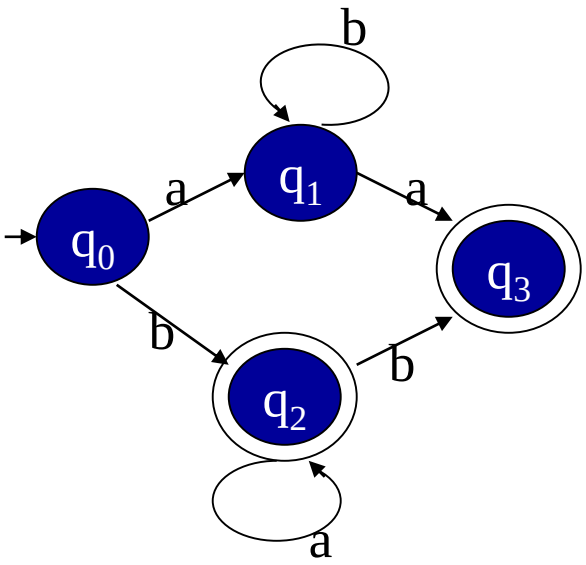


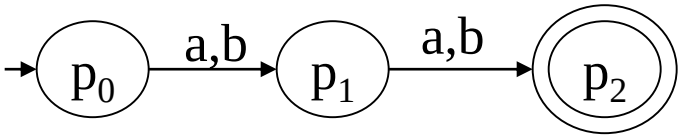
- Costruire un automa che riconosca il linguaggio inverso del linguaggio riconosciuto dal seguente automa:



Esercizi

- Costruire un automa che riconosca l'intersezione e un automa che riconosca la differenza dei linguaggi riconosciuti dai seguenti automi:

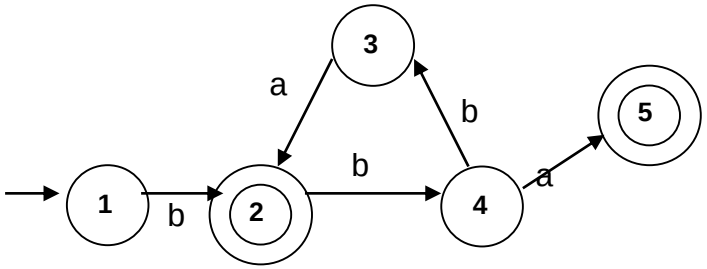


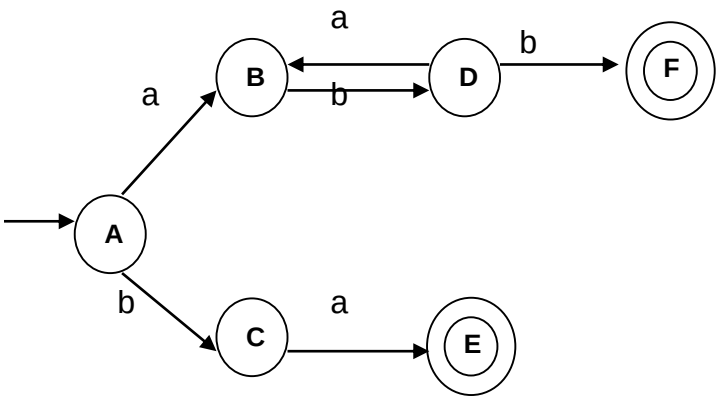


Esercizi

- Fornire un'espressione regolare che denoti il linguaggio
  $L = \{w / w \in a^*b^* \text{ e } |w| = 2i+1, i \geq 0\}$.
  e costruire un automa che lo riconosca.

- Costruire automi che riconoscano rispettivamente l'intersezione, l'unione, la concatenazione, l'inversione, la differenza, il complemento e la chiusura dei linguaggi riconosciuti dai seguenti automi:

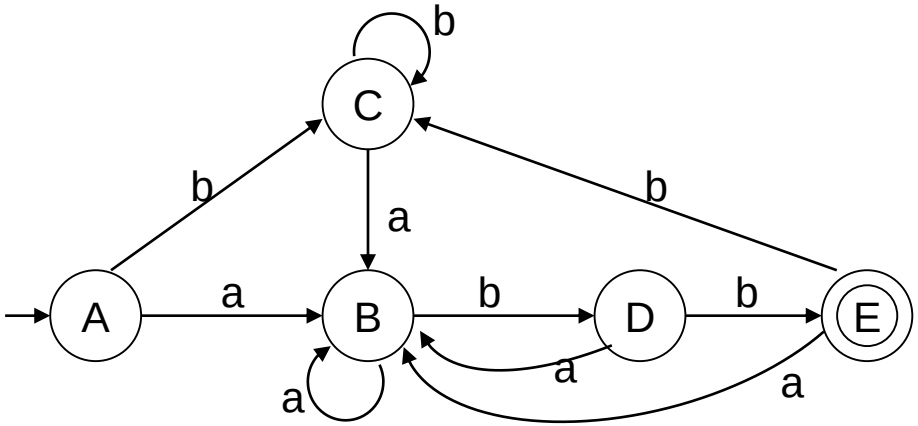


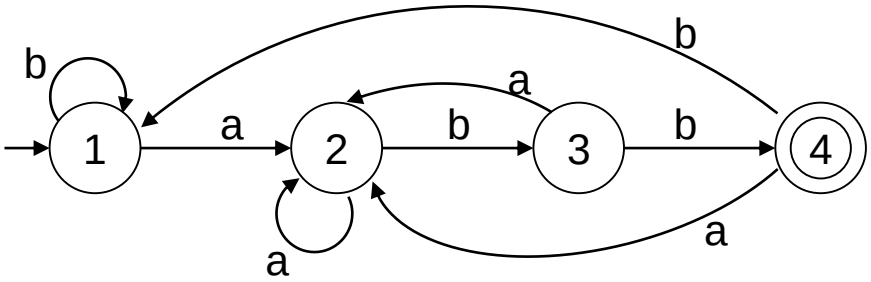


Uguaglianza di automi finiti

Due automi sono *uguali a meno dei nomi* se uno può essere trasformato nell'altro modificando solo i nomi degli stati.

I due automi deterministici seguenti sono equivalenti (entrambi riconoscono il linguaggio (a + b)*abb, ma non uguali.





Automa minimo

Dato un automa a stati finiti *deterministico* D possiamo costruire l'automa a stati finiti "più piccolo" (con il minimo numero di stati) equivalente a D, unico a meno del nome degli stati, eliminando gli stati "inutili" e "fondendo" gli stati "indistinguibili".

Uno stato *s* dell'automa D è *utile* se esiste una stringa $x = x_1x_2$ tale che:

$$\hat{\delta}(q_0, x_1) = s \quad \text{e} \quad \hat{\delta}(s, x_2) \in F,$$

cioè se *s* è raggiungibile dallo stato iniziale e da *s* si può raggiungere uno stato finale.
Uno stato *s* risulta pertanto *inutile* se nel diagramma di transizione non esiste nessun cammino da q_0 a *s* e/o non esiste nessun cammino da *s* a uno stato finale.

Stati indistinguibili

Sia D = <S, Σ, δ, s_0,F> un automa finito deterministico senza stati non raggiungibili dallo stato iniziale e *s* e *t* siano due suoi stati.

1. $x \in \Sigma^*$ *distingue* tra *s* e *t* se uno e uno solo tra gli stati $\hat{\delta}(s, x)$ e $\hat{\delta}(t, x)$ è finale. (*x* distingue *s* da *t* se esattamente uno degli stati raggiungibili da *s* e da *t* mediante un percorso etichettato con la stringa *x* è di accettazione).

2. *s* e *t* sono *indistinguibili* se e solo se nessuna stringa (di nessuna lunghezza) distingue *s* da *t*.

Possiamo cercare gli stati distinguibili considerando le stringhe in ordine di lunghezza.

Stati indistinguibili

Osservazioni:

1. La stringa vuota ε distingue ogni stato finale da ogni stato non finale, per definizione. Pertanto partizioniamo inizialmente l'insieme degli stati nei due sottinsiemi F e S-F.
2. Per le stringhe di lunghezza 1, due stati *s* e *t* sono distinguibili se, per almeno un simbolo dell'alfabeto *a*, uno e uno solo tra gli stati $\delta(s,a)$ e $\delta(t,a)$ appartiene a F, cioè $\delta(s,a)$ e $\delta(t,a)$ sono in gruppi diversi della partizione precedente.
3. Se $\delta(s,a) = s'$ e $\delta(t,a) = t'$ e s' e t' sono distinti da una stringa *x* (ad esempio $\hat{\delta}(s',x) \in F$ e $\hat{\delta}(t',x) \notin F$), allora *s* e *t* sono distinti dalla stringa *ax*.

Se si ha la partizione che raccoglie gli stati in gruppi di stati distinguibili da stringhe di lunghezza $\leq$ n, due stati *s* e *t* nello stesso gruppo saranno distinguibili da stringhe di lunghezza $\leq$ n+1 se gli stati $\delta(s,a)$ e $\delta(t,a)$ appartengono a gruppi diversi della partizione precedente.

Stati indistinguibili

Possiamo costruire una sequenza di partizioni degli stati:

$$\Pi_0 = (S\text{-}F, F),\ \Pi_1, \Pi_2, \ldots, \Pi_i, \ldots$$

tali che i gruppi della partizione Π_i contengano gli stati non distinguibili da stringhe di lunghezza $\leq$ i.

Quando fermarsi?

È stato dimostrato che se $\Pi_i = \Pi_{i+1}$, la partizione degli stati non cambia più, cioè $\Pi_i = \Pi_{i+1} = \Pi_{i+2} = \ldots = \Pi_{i+k} = \ldots$
Allora Π_i (o Π_{i+1}) è la partizione finale Π_{final}, che contiene i gruppi degli stati indistinguibili.

Quanti passi possono servire?

Si può dimostrare che, se l'automa ha n stati, al massimo

$$\Pi_{final} = \Pi_{n-2}$$

Algoritmo di minimizzazione

1. L'algoritmo parte dalla partizione $\Pi = \Pi_0$ che distingue gli stati finali da quelli non finali e costruisce progressivamente partizioni i cui gruppi sono insiemi di stati non ancora identificati come distinguibili.
2. Due stati appartenenti ad insiemi diversi di una partizione sono invece già stati identificati come distinguibili.
3. Quando la partizione non può più essere modificata spezzando un gruppo in gruppi più piccoli, l'algoritmo costruisce il DFA minimo prendendo come stati i gruppi della partizione stessa (Π_{final}).
4. Il procedimento fondamentale consiste nel considerare un generico gruppo $\{s_1, s_2, \ldots, s_n\}$ e verificare per ogni coppia di stati s_i e s_j, se almeno un simbolo dell'alfabeto di input *a* li "distingue" nel senso che le transizioni da s_i e s_j con il simbolo *a* portano in stati che sono in gruppi diversi della partizione precedente.

Algoritmo di minimizzazione

Input: un automa finito deterministico D = <S, Σ, δ, s_0, F>.
Output: un automa deterministico D' = <S', Σ, δ', s'_0, F'> equivalente a D con il minimo numero di stati.

Algoritmo:

1) Sia Π la partizione iniziale costituita dai due gruppi di stati F e S – F.
2) Applicare l'algoritmo per costruire una nuova partizione Π_{new}.
3) Se $\Pi_{new} = \Pi$, allora $\Pi_{final} = \Pi_{new}$ e si procede con il passo (4), altrimenti ripetere il passo (2) con Π_{new} al posto di Π.
4) Costruire l'automa D' = <S', Σ, δ', s'_0, F'> nel modo seguente:
   - Per ogni gruppo di Π_{final} scegliere uno stato come rappresentativo del gruppo. S' è l'insieme degli stati rappresentativi dei diversi gruppi.
   - δ': $\delta'(s, a) = t$ se $\delta(s, a) = r$ e r è nel gruppo di cui t è il rappresentante.
   - s'_0 è lo stato che rappresenta il gruppo che contiene lo stato iniziale di D, s_0
   - F' = $\{s \mid s \in F\}$

Algoritmo di minimizzazione

Costruzione della nuova partizione Π_{new}:

Inizializza Π_{new} a Π;
for ogni gruppo G di Π {
    suddividi G in sottogruppi tali che due stati *s* e *t* appartengano allo stesso sottogruppo se e solo se per ogni *a* $\in \Sigma$ gli stati *s* e *t* hanno transizioni per il simbolo *a* verso stati appartenenti allo stesso gruppo di Π;
    sostituisci G in Π_{new} con il nuovo insieme di sottogruppi;
}

Nota: ogni gruppo contiene tutti stati finali o tutti stati non finali poiché si parte con i due gruppi F e Q-F e il procedimento forma unicamente sottogruppi di gruppi preesistenti.

Automa minimo

Teorema

Sia *B* il DFA ottenuto applicando l'algoritmo di minimizzazione al DFA *A*.

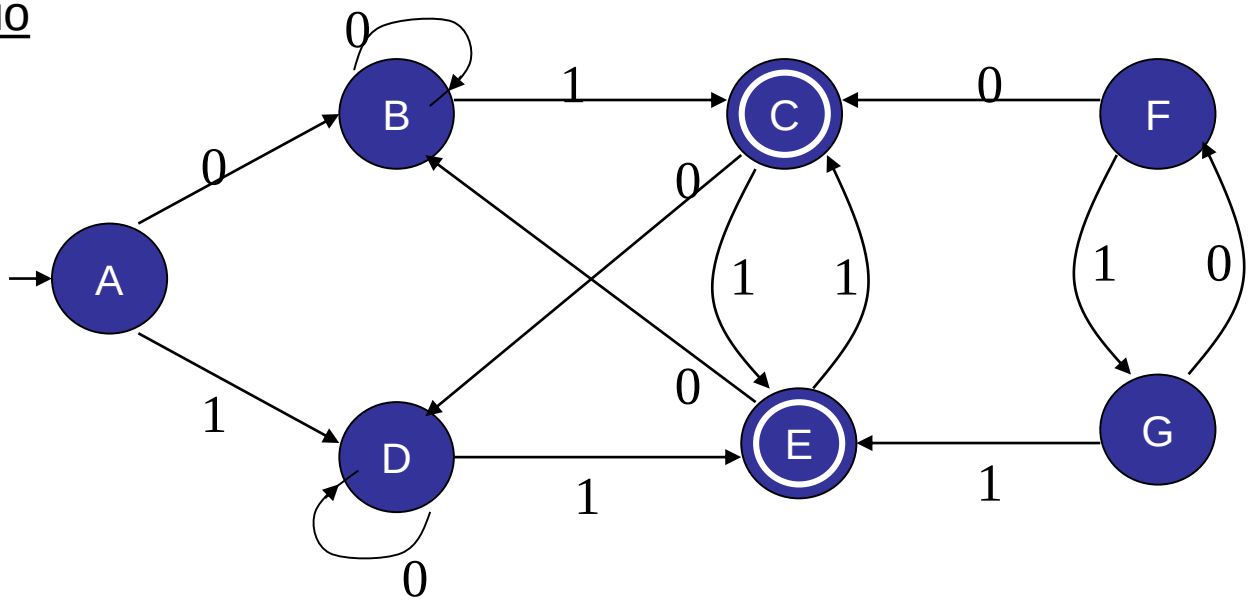
Allora:

1. L(*B*) = L(*A*).
2. L'automa *B* è minimo nel senso che non esiste nessun automa equivalente ad *A* con un numero di stati inferiore al numero di stati di B.
3. L'automa minimo è unico, a meno del nome degli stati.

Nota: Un modo semplice per verificare se due automi finiti accettano lo stesso linguaggio è quello di minimizzarli. I due automi sono equivalenti se e solo se gli automi minimi ottenuti sono uguali a meno dei nomi degli stati (isomorfi).

Automa minimo

Esempio



F e G sono stati non raggiungibili dallo stato iniziale e perciò li eliminiamo

Π_0 = {A, B, D}, {C, E}

Π_1 = {A}, {B, D}, {C, E}

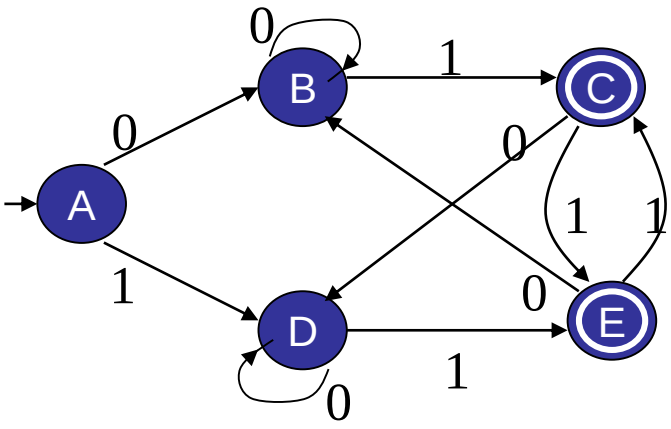
Π_2 = {A}, {B, D}, {C, E}

Automa minimo

Esempio

L'automa minimo è:

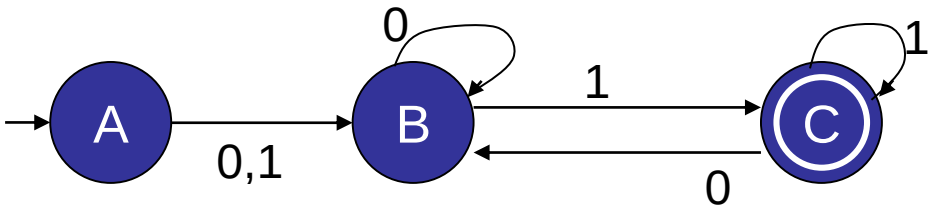
< {A, B, C}, {0, 1}, δ, A, {C}>



δ(A, 0) = B δ(A, 1) = B

δ(B, 0) = B δ(B, 1) = C

δ(C, 0) = B δ(C, 1) = C

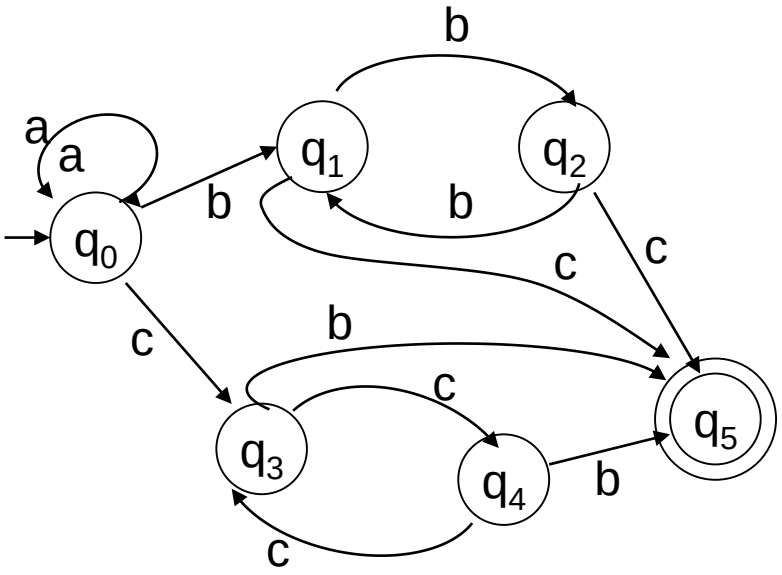


Automa minimo

Esempio

$< \{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \delta, q_0, \{q_5\} >$

δ	a	b	c
$\rightarrow q_0$	q_0	q_1	q_3
q_1	q_{err}	q_2	q_5
q_2	q_{err}	q_1	q_5
q_3	q_{err}	q_5	q_4
q_4	q_{err}	q_5	q_3
$*q_5$	q_{err}	q_{err}	q_{err}



Automa minimo

Esempio

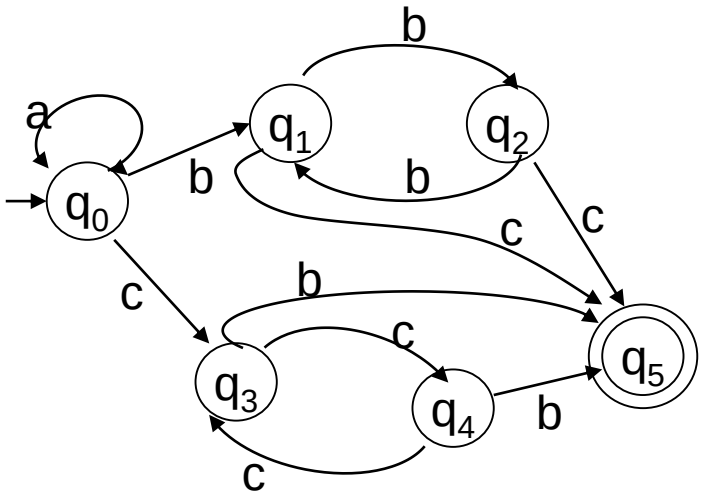
δ	a	b	c
→ q_0	q_0	q_1	q_3
q_1	q_{err}	q_2	q_5
q_2	q_{err}	q_1	q_5
q_3	q_{err}	q_5	q_4
q_4	q_{err}	q_5	q_3
*q_5	q_{err}	q_{err}	q_{err}

$\Pi_0 = \{q_0, q_1, q_2, q_3, q_4, q_{err}\}, \{q_5\}$

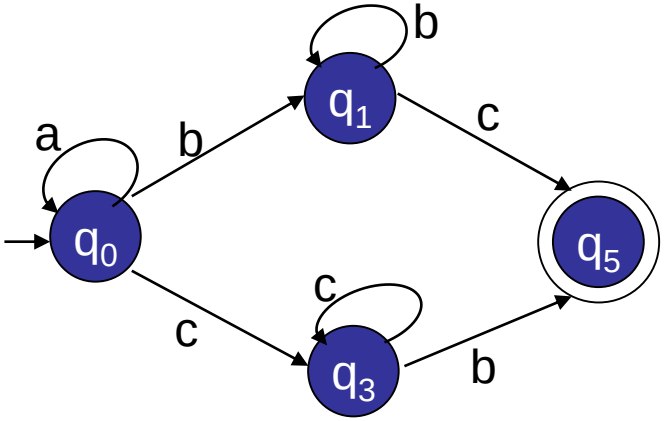
$\Pi_1 = \{q_0, q_{err}\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5\}$

$\Pi_2 = \{q_0\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5\}, \{q_{err}\}$

$\Pi_3 = \{q_0\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5\}, \{q_{err}\}$



Automa minimo



N.B. Lo stato d'errore è sempre distinguibile da tutti gli altri stati (perché?), pertanto nella minimizzazione può essere ignorato.

Esercizi

Per ognuno dei seguenti automi costruire l'automa minimo equivalente;