Equivalenza e minimizzazione di automi

Stati equivalenti

Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA, e $\{p, q\} \subseteq Q$. Definiamo

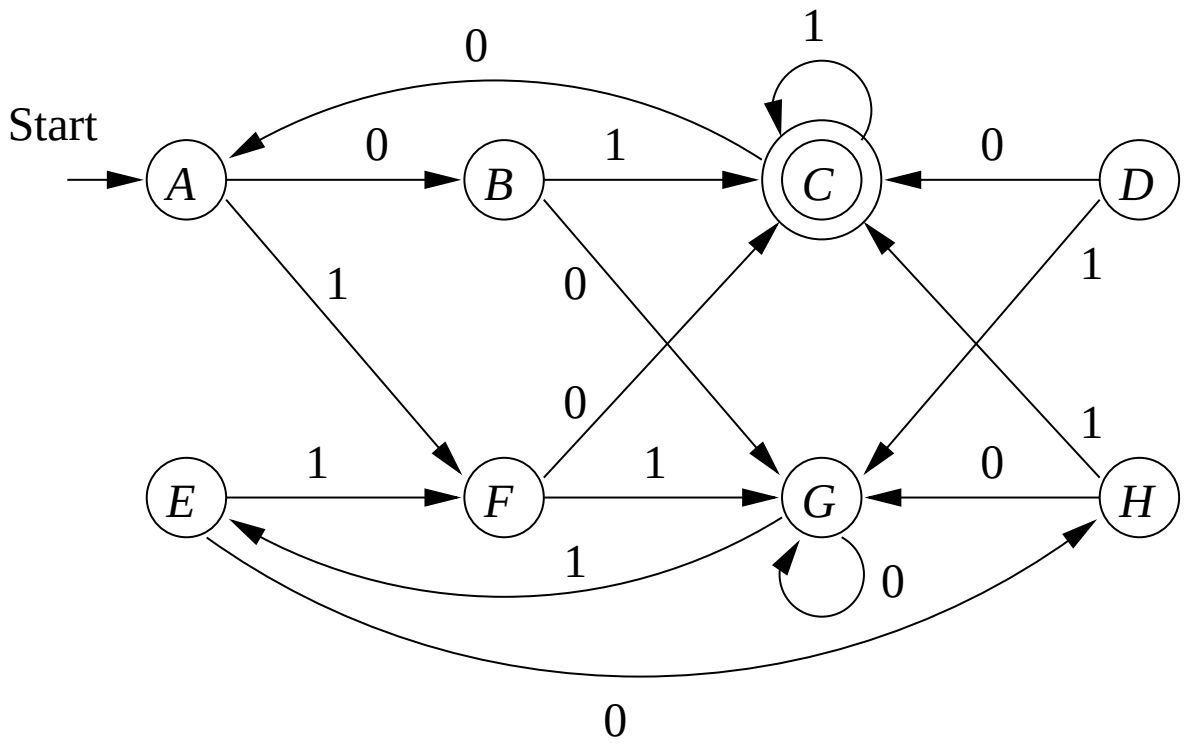
$$p \equiv q \;\Leftrightarrow\; \forall w \in \Sigma^* \;:\; \hat{\delta}(p, w) \in F \text{ se e solo se } \hat{\delta}(q, w) \in F$$

- Se $p \equiv q$ diciamo che p e q sono *equivalenti*
- Se $p \not\equiv q$ diciamo che p e q sono *distinguibili*

In altre parole: p e q sono distinguibili se e solo se

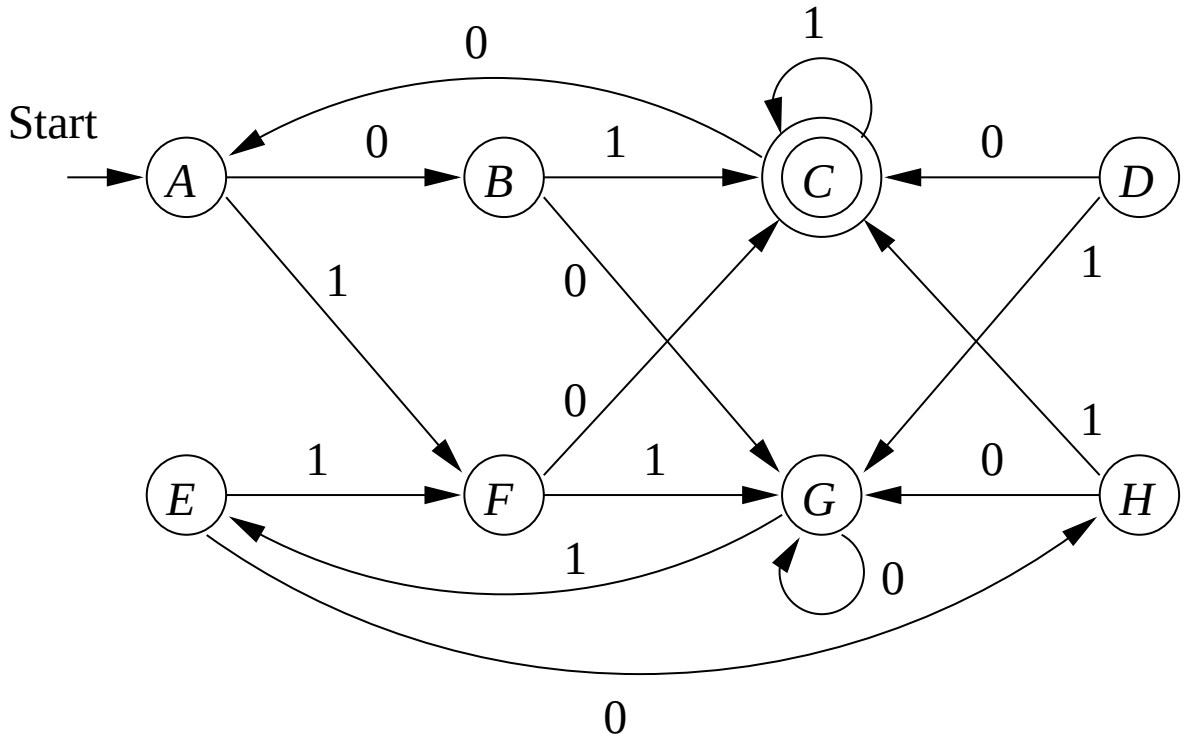
$$\exists w : \hat{\delta}(p, w) \in F \text{ e } \hat{\delta}(q, w) \notin F, \text{ o viceversa}$$

Esempio



$\hat{\delta}(C, \epsilon) \in F, \hat{\delta}(G, \epsilon) \notin F \Rightarrow C \not\equiv G$
$\hat{\delta}(A, 01) = C \in F, \hat{\delta}(G, 01) = E \notin F \Rightarrow A \not\equiv G$

Cosa si puo' dire su A e E?



$\hat{\delta}(A, \epsilon) = A \notin F, \hat{\delta}(E, \epsilon) = E \notin F$
$\hat{\delta}(A, 1) = F = \hat{\delta}(E, 1)$
Quindi $\hat{\delta}(A, 1x) = \hat{\delta}(E, 1x) = \hat{\delta}(F, x)$
$\hat{\delta}(A, 00) = G = \hat{\delta}(E, 00)$
$\hat{\delta}(A, 01) = C = \hat{\delta}(E, 01)$
Conclusione: $A \equiv E$.

Algoritmo induttivo

Possiamo calcolare coppie di stati distinguibili con il seguente metodo induttivo (*algoritmo di riempimento di una tavola*):
Base: Se $p \in F$ e $q \notin F$, allora $p \not\equiv q$.
Induzione: Se $\exists a \in \Sigma : \delta(p, a) \not\equiv \delta(q, a)$, allora $p \not\equiv q$.

Esempio: Applichiamo l'algoritmo ad A:

B	x					
C	x	x				
D	x	x	x			
E		x	x	x		
F	x	x	x		x	
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Correttezza dell'algoritmo

Teorema 4.20: Se p e q non sono distinguibili dall'algoritmo, allora $p \equiv q$.

Prova: Supponiamo per assurdo che esista una *coppia "sbagliata"* $\{p, q\}$, tale che

1. $\exists w : \hat{\delta}(p, w) \in F, \hat{\delta}(q, w) \notin F$, o viceversa.
2. L'algoritmo non distingue tra p e q.

Sia $w = a_1 a_2 \cdots a_n$ la stringa piu' corta che identifica la coppia "sbagliata" $\{p, q\}$.
Allora $w \neq \epsilon$ perche' altrimenti l'algoritmo distinguerebbe p da q (caso base). Quindi $n \geq 1$.

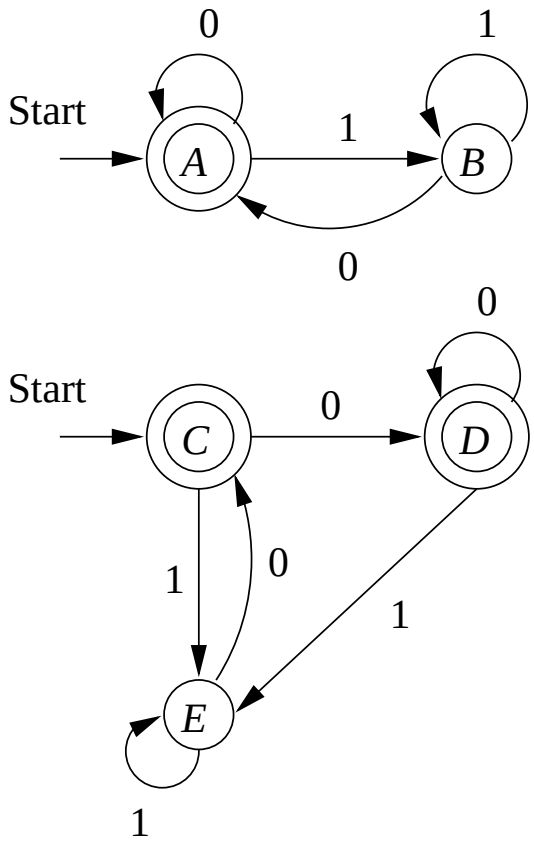
- Consideriamo gli stati $r = \delta(p, a_1)$ e $s = \delta(q, a_1)$.
- Allora $\{r, s\}$ non puo' essere una coppia sbagliata perche' $\{r, s\}$ srebbe identificata da una stringa piu' corta di w.
- Quindi, l'algoritmo deve aver scoperto nel caso base che r and s sono distinguibili.
- Ma allora l'algoritmo distinguerebbe p da q nella parte induttiva.
- Quindi non ci sono coppie "sbagliate" e il teorema e' vero.

Testare l'equivalenza di linguaggi regolari

Siano L e M linguaggi regolari (descritti in qualche forma).
Per testare se $L = M$

1. convertiamo sia L che M in DFA.
2. Immaginiamo il DFA che e' l'unione dei due DFA (non importa se ha due stati iniziali)
3. Se l'algoritmo dice che i due stati iniziali sono distinguibili, allora $L \neq M$, altrimenti $L = M$.

Esempio



Possiamo vedere che entrambi i DFA accettano $L(\epsilon + (\mathbf{0} + \mathbf{1})^*\mathbf{0})$.

Esempio

Il risultato dell'algoritmo e'

B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

Quindi i due automi sono equivalenti.

Minimizzazione di DFA

- Possiamo usare l'algoritmo per minimizzare un DFA mettendo insieme tutti gli stati equivalenti. Cioe' rimpiazzando p by $p/_{\equiv}$.
- Esempio: Il DFA di prima ha le seguenti classi di equivalenza: $\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$.
- Il DFA unione di prima ha le seguenti classi di equivalenza: $\{\{A, C, D\}, \{B, E\}\}$.
- Notare: affinche' $p/_{\equiv}$ sia una *classe di equivalenza*, la relazione $\equiv$ deve essere una *relazione di equivalenza* (riflessiva, simmetrica, e transitiva).

Transitivita'

Teorema 4.23: Se $p \equiv q$ e $q \equiv r$, allora $p \equiv r$.

Prova: Supponiamo per assurdo che $p \not\equiv r$.

- Allora $\exists w$ tale che $\hat{\delta}(p, w) \in F$ e $\hat{\delta}(r, w) \notin F$, o viceversa.
- Lo stato $\hat{\delta}(q, w)$ e' o di accettazione o no.
- *Caso 1*: $\hat{\delta}(q, w)$ e' di accettazione. Allora $q \not\equiv r$.
- *Caso 2*: $\hat{\delta}(q, w)$ non e' di accettazione. Allora $p \not\equiv q$.
- Il caso contrario puo' essere provato simmetricamente.
- Quindi deve essere $p \equiv r$.

Minimizzazione di automi

Per minimizzare un DFA $A = (Q, \Sigma, \delta, q_0, F)$ costruiamo un DFA $B = (Q/_{\equiv}, \Sigma, \gamma, q_0/_{\equiv}, F/_{\equiv})$, dove

$$\gamma(p/_{\equiv}, a) = \delta(p, a)/_{\equiv}$$

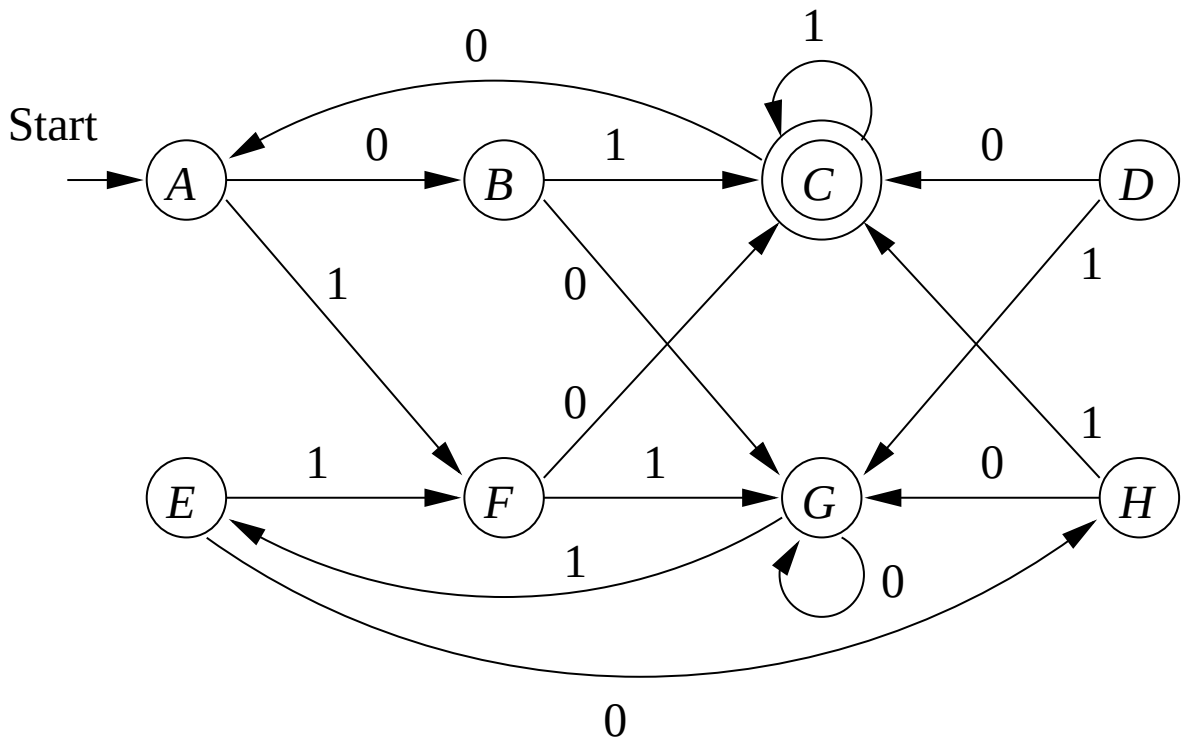
Affinche' B sia ben definito, dobbiamo mostrare che

$$\text{Se } p \equiv q \text{ allora } \delta(p, a) \equiv \delta(q, a)$$

Se $\delta(p, a) \not\equiv \delta(q, a)$, allora l'algoritmo concluderebbe $p \not\equiv q$, quindi B e' ben definito. Notare anche che $F/_{\equiv}$ contiene tutti e soli gli stati accettanti di A.

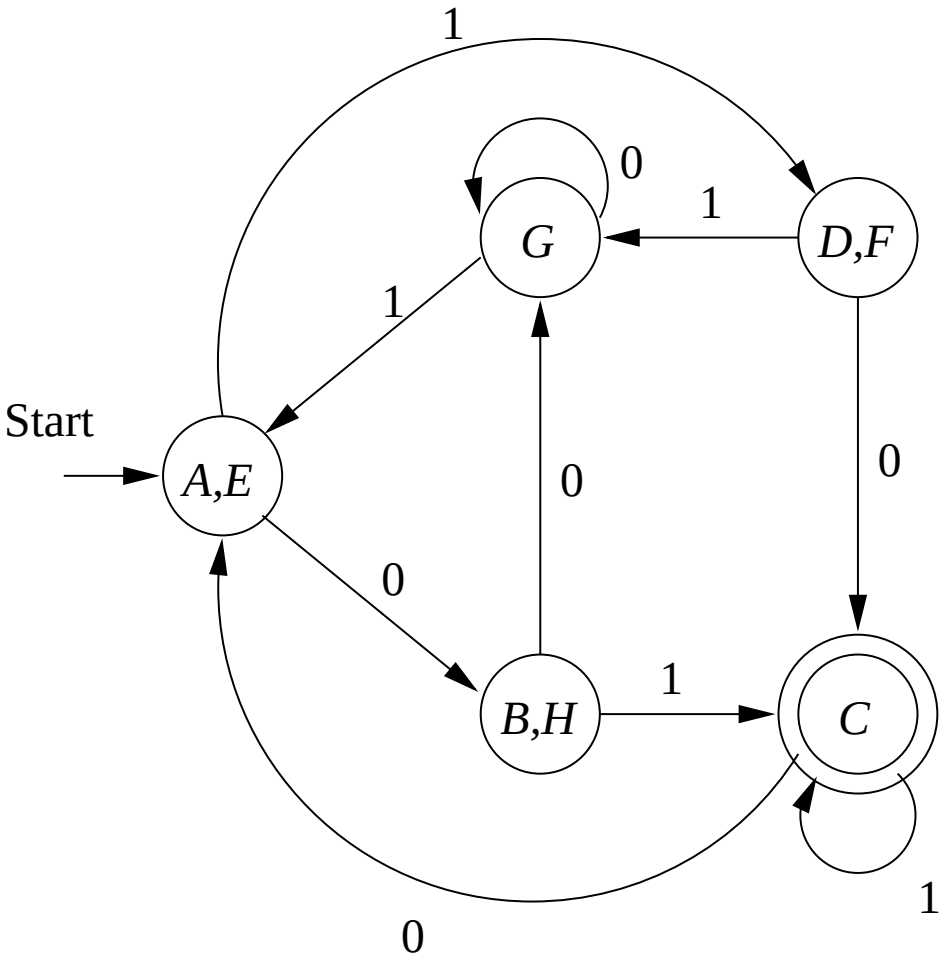
Esempio

Possiamo minimizzare

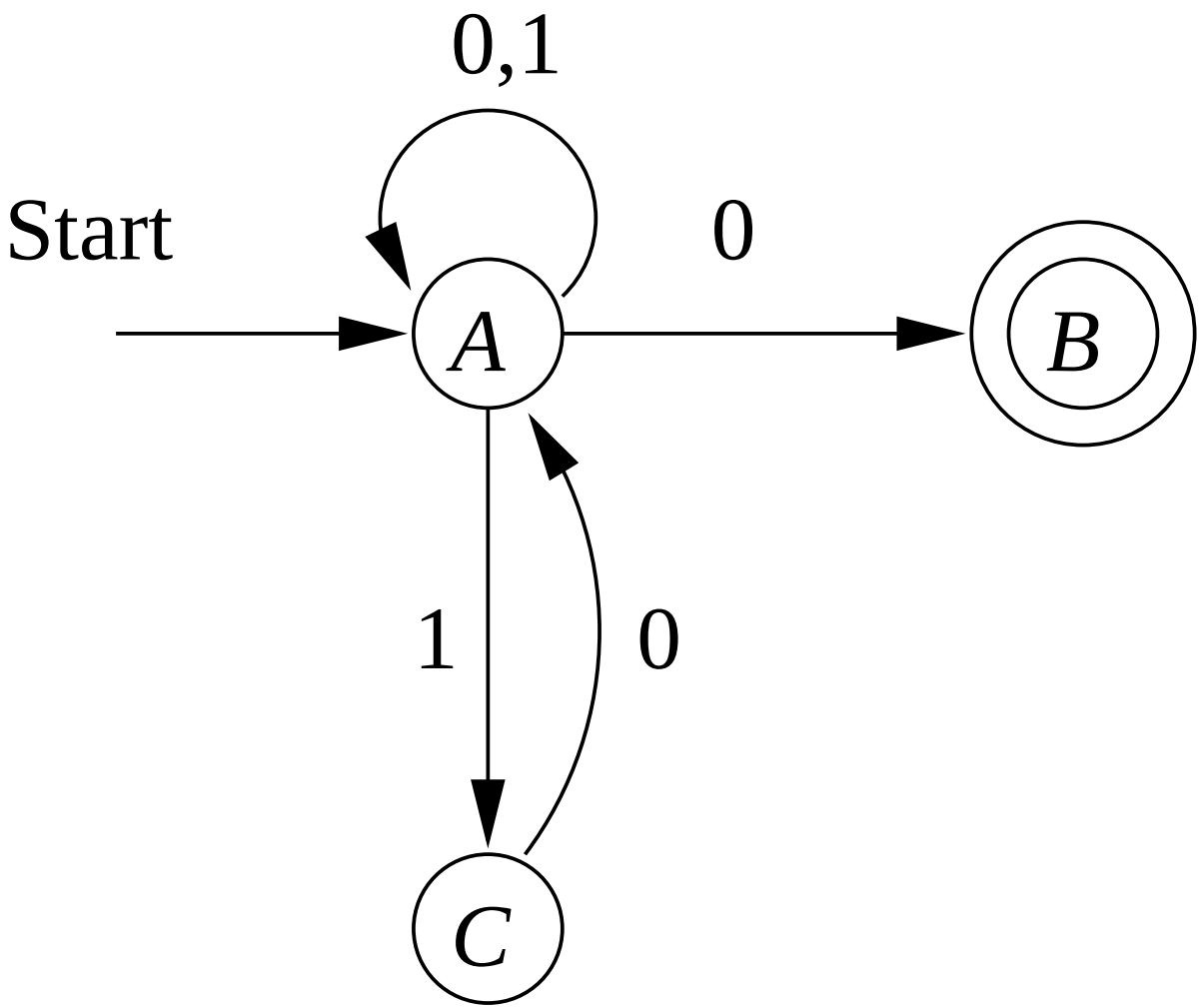


Esempio

Otteniamo:



Notare: Non possiamo applicare l'algoritmo a NFA.
Per esempio, per minimizzare



rimuoviamo lo stato C.
Ma $A \not\equiv C$.

Perche' non si puo' migliorare il DFA minimizzato

- Sia B il DFA minimizzato ottenuto applicando l'algoritmo al DFA A.
- Sappiamo gia' che $L(A) = L(B)$.
- Potrebbe esistere un DFA C, con $L(C) = L(B)$ e meno stati di B?
- Applichiamo l'algoritmo a B "unito con" C.
- Dato che $L(B) = L(C)$, abbiamo $q_0^B \equiv q_0^C$.
- Inoltre, $\delta(q_0^B, a) \equiv \delta(q_0^C, a)$, per ogni a.

- Per ogni stato p in B esiste almeno uno stato q in C, tale che $p \equiv q$.
- **Prova**:
  - Non ci sono stati inaccessibili, quindi $p = \hat{\delta}(q_0^B, a_1 a_2 \cdots a_k)$, per una qualche stringa $a_1 a_2 \cdots a_k$.
  - Allora $q = \hat{\delta}(q_0^C, a_1 a_2 \cdots a_k)$, e $p \equiv q$.
  - Dato che C ha meno stati di B, ci devono essere due stati r e s di B tali che $r \equiv t \equiv s$, per qualche stato t di C.
  - Ma allora $r \equiv s$ che e' una contraddizione, dato che B e' stato costruito dall'algoritmo.