Proprietà dei Linguaggi Regolari (Capitolo 4)

• *Pumping Lemma.* Ogni linguaggio regolare soddisfa il pumping lemma. Se qualcuno vi presenta un linguaggio che non soddisfa il pumping lemma, allora il linguaggio non è regolare.

• *Proprietà di chiusura.* Come costruire automi da componenti usando delle operazioni, ad esempio dati L e M possiamo costruire un automa per $L \cap M$.

• *Proprietà di decisione.* Analisi computazionale di automi, cioè quando due automi sono equivalenti.

• *Tecniche di minimizzazione.* Possiamo risparmiare costruendo automi più piccoli.

Il Pumping Lemma, informalmente

Supponiamo che $L_{01} = \{0^n 1^n : n \geq 1\}$ sia regolare.

Allora deve essere accettato da un qualche DFA A, con, ad esempio, k stati.

Supponiamo che A legga 0^k. Avrà le seguenti transizioni:

ϵ	p_0
0	p_1
00	p_2
$\ldots$	$\ldots$
0^k	p_k

$\Rightarrow \exists i < j : p_i = p_j$.
Chiamiamo q questo stato.

Adesso possiamo ingannare A:

Se $\hat{\delta}(q, 1^i) \in F$ l'automa accetterà, sbagliando, $0^j 1^i$.

Se $\hat{\delta}(q, 1^i) \notin F$ l'automa rifiuterà, sbagliando, $0^i 1^i$.

Quindi L_{01} non può essere regolare.

- Generaliziamo questo ragionamento.

Il Pumping Lemma

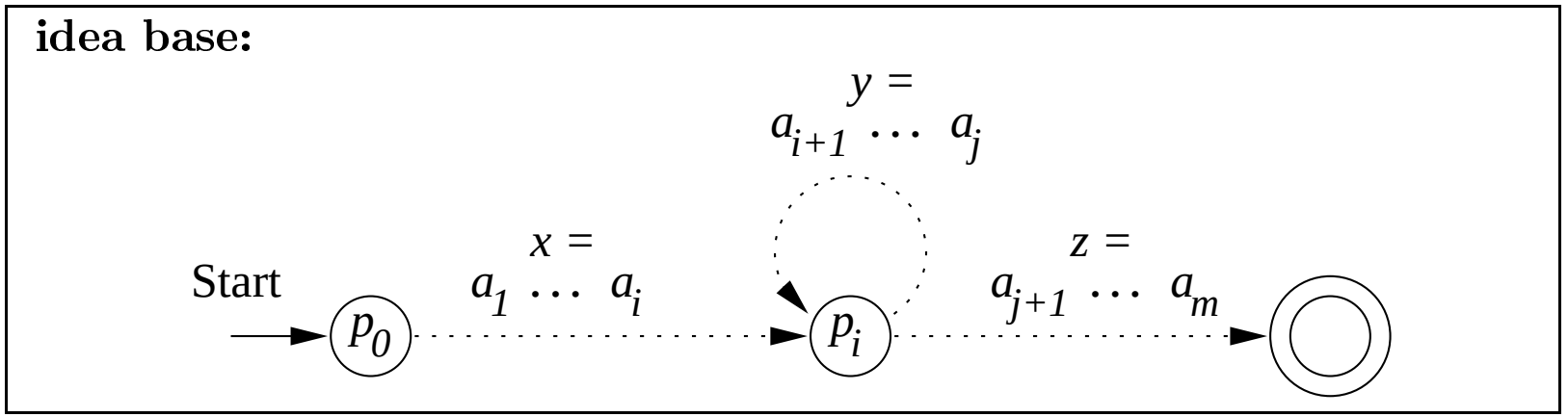
Teorema 4.1. *Il Pumping Lemma per Linguaggi Regolari.*

Sia L un linguaggio regolare.

Allora $\exists n, \forall w \in L : |w| \geq n \Rightarrow w = xyz$ tale che:

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

idea base:



Il Pumping Lemma: Esempio

Sia L_{eq} il linguaggio delle stringhe con ugual numero di zeri e di uni.

Supponiamo che L_{eq} sia regolare. Allora $w = 0^n 1^n \in L$.

Per il pumping lemma, $w = xyz$, $|xy| \leq n$, $y \neq \epsilon$ e $xy^k z \in L_{eq}$

$$w = \underbrace{000\cdots}_{x}\underbrace{\cdots 0}_{y}\underbrace{0111\cdots 11}_{z}$$

In particolare, $xz \in L_{eq}$, ma xz ha meno zeri di uni.

Proprietà di chiusura dei linguaggi regolari

Siano L e M due linguaggi regolari. Allora i seguenti linguaggi sono regolari:

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

Proprietà di chiusura dei linguaggi regolari

Teorema 4.4. Per ogni coppia di linguaggi regolari L e M, $L \cup M$ è regolare.

Prova. Sia $L = L(E)$ e $M = L(F)$. Allora $L(E + F) = L \cup M$ per definizione.

Teorema 4.5. Se L è un linguaggio regolare su Σ, allora anche $\overline{L} = \Sigma^* \setminus L$ è regolare.

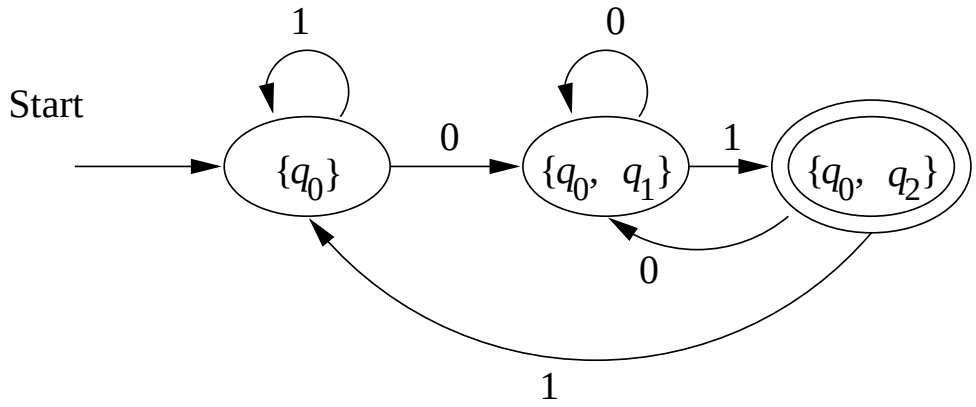
Prova. Sia L riconosciuto da un DFA

$$A = (Q, \Sigma, \delta, q_0, F).$$

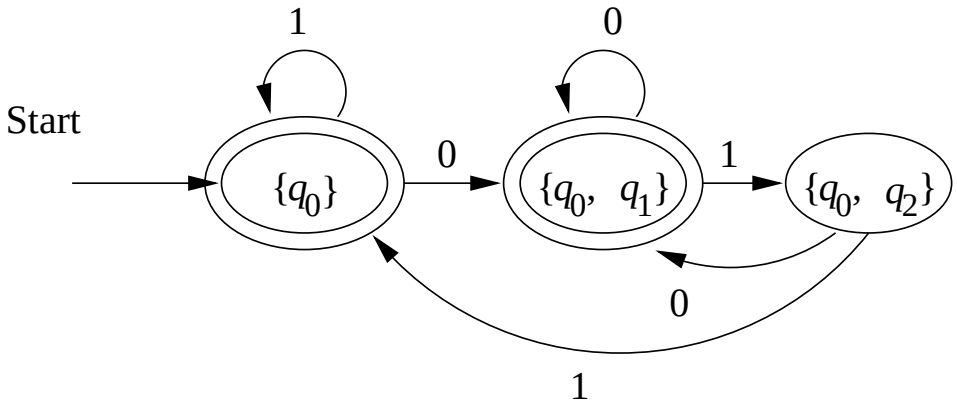
Sia $B = (Q, \Sigma, \delta, q_0, Q \setminus F)$. Allora $L(B) = \overline{L}$.

Esempio:

Sia L riconosciuto dal DFA qui sotto:



Allora $\overline{L}$ è riconosciuto da:



Teorema 4.8. Se L e M sono regolari, allora anche $L \cap M$ è regolare.

Prova. Per la legge di DeMorgan, $L \cap M = \overline{\overline{L} \cup \overline{M}}$. Sappiamo già che i linguaggi regolari sono chiusi sotto il complemento e l'unione.

Teorema 4.8. Se L e M sono regolari, allora anche $L \cap M$ è regolare.

Prova (idea). Sia L il linguaggio di

$$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$$

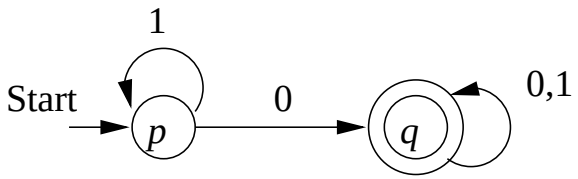
e M il linguaggio di

$$A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$$

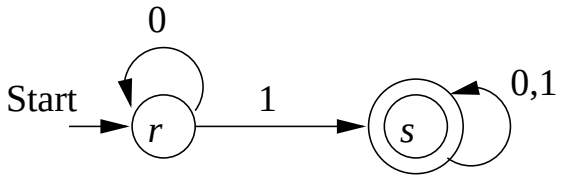
Assumiamo senza perdita di generalità che entrambi gli automi siano deterministici.

Costruiremo un automa che simula A_L e A_M in parallelo, e accetta se e solo se sia A_L che A_M accettano.

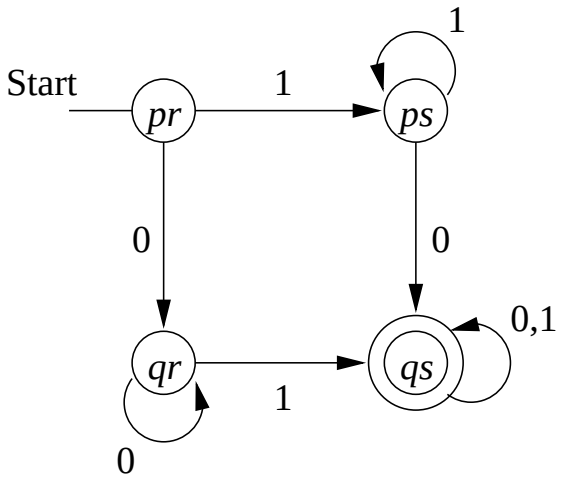
Se A_L va dallo stato p allo stato q leggendo 0, e A_M rimane nello stato r leggendo 0, allora $A_{L\cap M}$ andrà dallo stato (p, r) allo stato (q, r) leggendo 0.



(a)



(b)



(c)

Teorema 4.10. Se L e M sono linguaggi regolari, allora anche $L \setminus M$ è regolare.

Prova. Osserviamo che $L \setminus M = L \cap \overline{M}$. Sappiamo già che i linguaggi regolari sono chiusi sotto il complemento e l'intersezione.

Teorema 4.11. Se L è un linguaggio regolare, allora anche L^R è regolare.

Prova 1: Sia L riconosciuto da un FA A. Modifichiamo A per renderlo un FA per L^R:

1. Giriamo tutti gli archi.
2. Rendiamo il vecchio stato iniziale l'unico stato finale.
3. Creiamo un nuovo stato iniziale p_0, con $\delta(p_0, \epsilon) = F$ (i vecchi stati finali).

Teorema 4.11. Se L è un linguaggio regolare, allora anche L^R è regolare.

Prova 2: Sia L descritto da un'espressione regolare E. Costruiremo un'espressione regolare E^R, tale che $L(E^R) = (L(E))^R$.

Procediamo per induzione strutturale su E.

Base: Se E è $\boldsymbol{\epsilon}$, $\boldsymbol{\emptyset}$, o $\boldsymbol{a}$, allora $E^R = E$.

Induzione:

1. $E = F + G$. Allora $E^R = F^R + G^R$
2. $E = F.G$. Allora $E^R = G^R.F^R$
3. $E = F^*$. Allora $E^R = (F^R)^*$

$$L(E^R) = (L(E))^R$$

Equivalenza e Minimizzazione di Automi

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

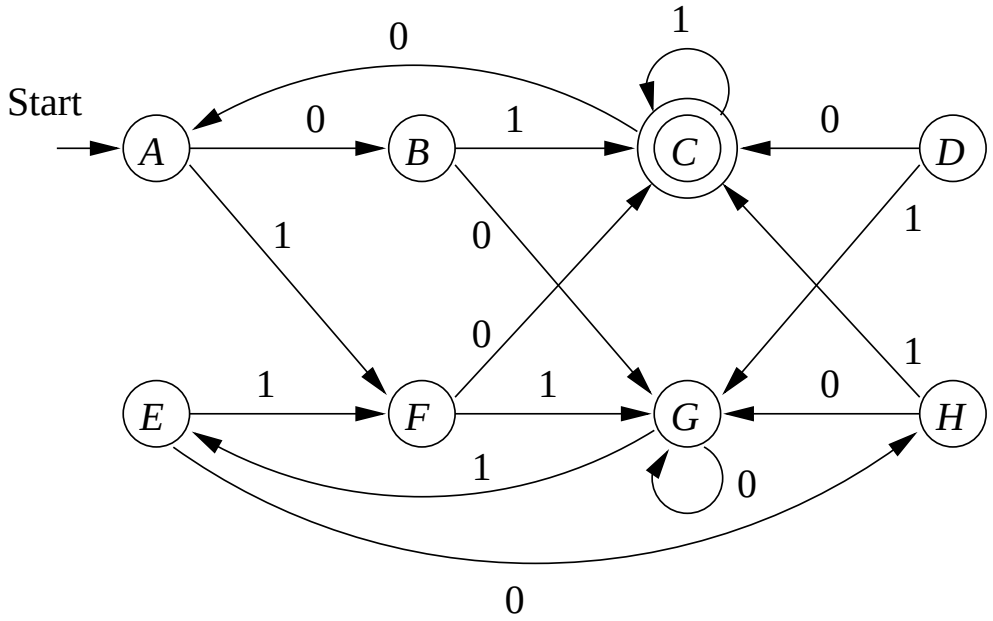
$$p \equiv q \;\Leftrightarrow\; \forall w \in \Sigma^* \;:\; \hat{\delta}(p, w) \in F \text{ se e solo se } \hat{\delta}(q, w) \in F$$

- 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

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

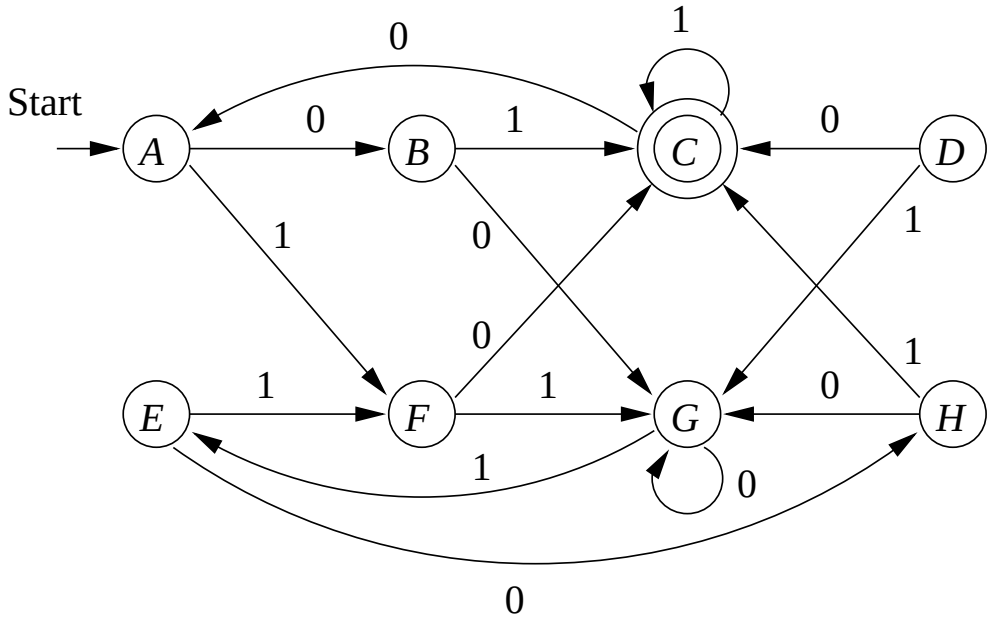
Esempio:



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

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

Cosa si può 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: $A \equiv E$.

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*

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

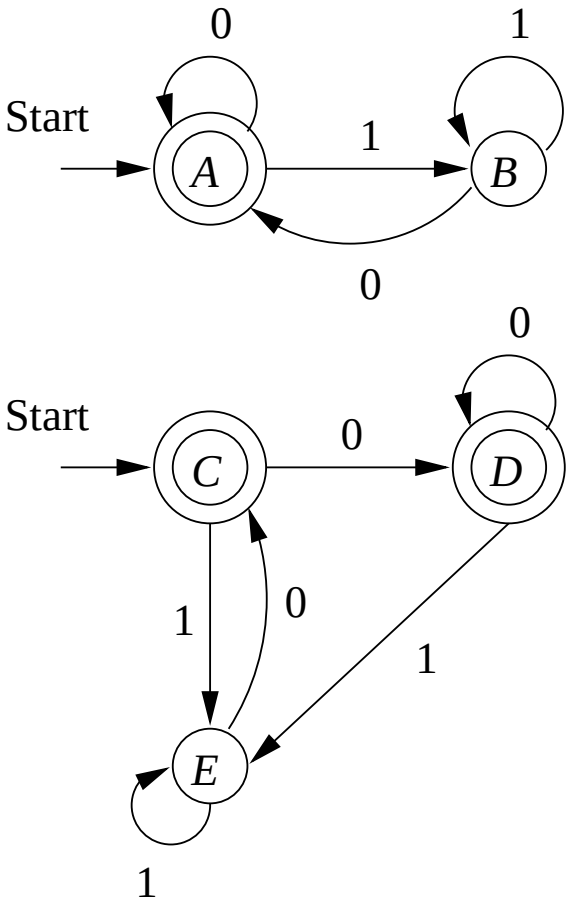
Testare l'equivalenza di linguaggi regolari

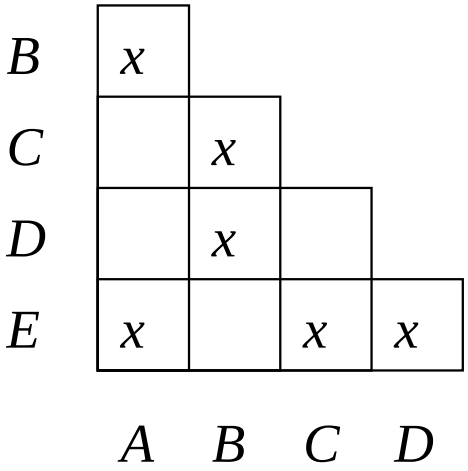
Siano L e M linguaggi regolari (descritti in qualche forma).

Per testare se $L = M$

1. convertiamo sia L che M in DFA.
2. Immaginiamo il DFA che è 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 $L(\epsilon + (\mathbf{0} + \mathbf{1})^*\mathbf{0})$. Quindi i due automi sono equivalenti.

Minimizzazione di DFA

Possiamo usare l'algoritmo per minimizzare un DFA mettendo insieme tutti gli stati equivalenti. Cioè 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: affinché $p/_{\equiv}$ sia una *classe di equivalenza*, la relazione $\equiv$ deve essere una *relazione di equivalenza* (riflessiva, simmetrica, e transitiva).

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

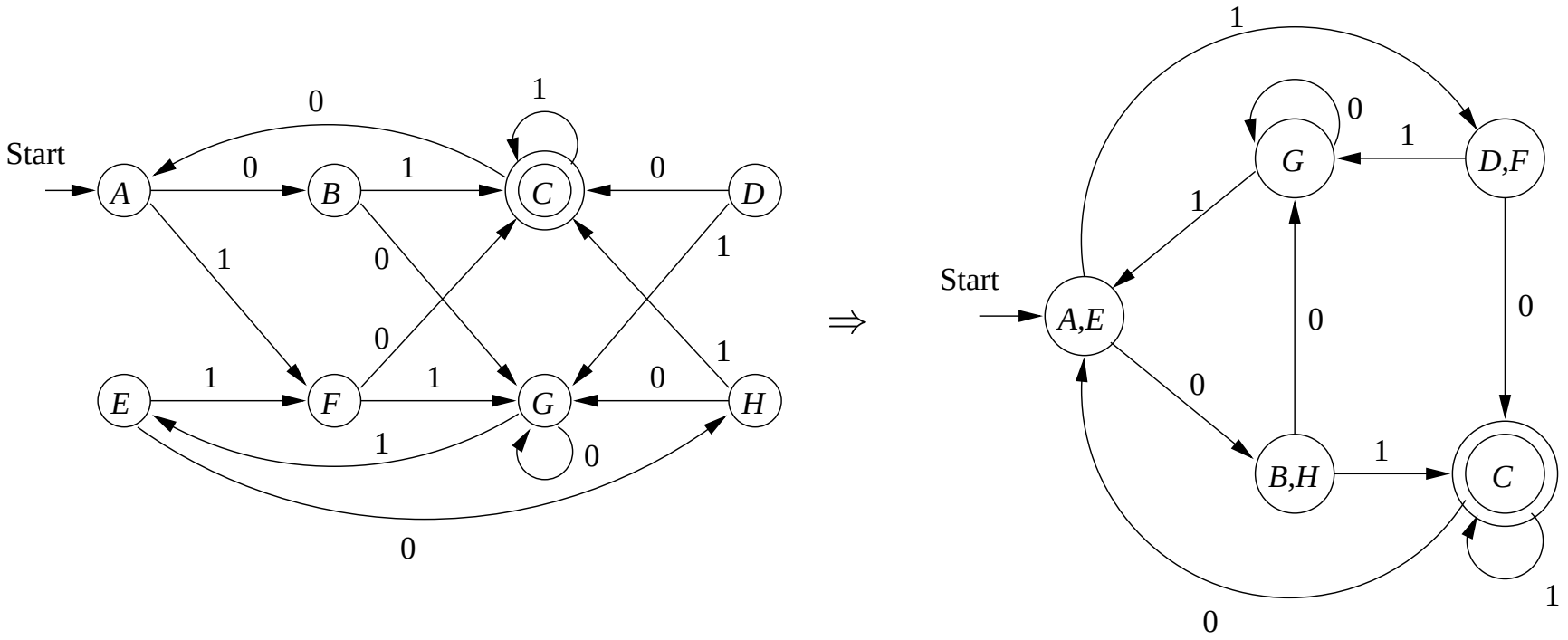
$$\gamma(p/_{\equiv}, a) = \delta(p, a)/_{\equiv}$$

Affinché B sia ben definito, dobbiamo mostrare che

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

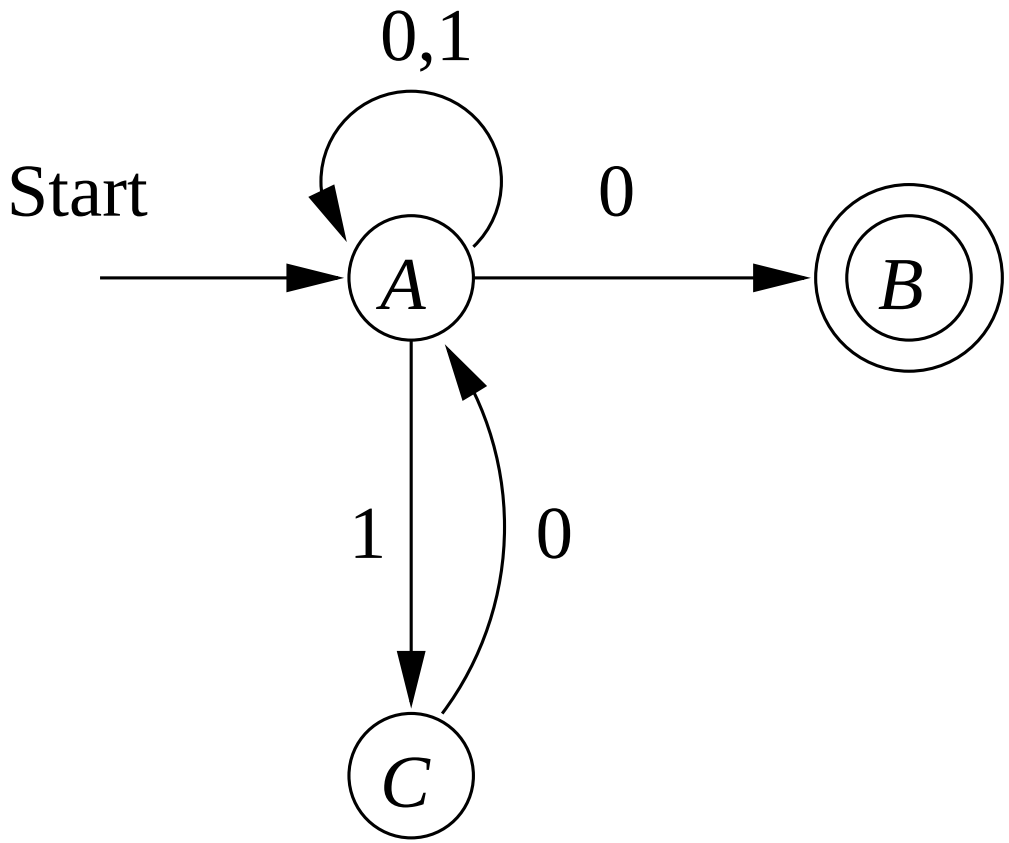
Se $\delta(p, a) \not\equiv \delta(q, a)$, allora l'algoritmo concluderebbe $p \not\equiv q$, quindi B è ben definito. Notare anche che $F/_{\equiv}$ contiene tutti e soli gli stati accettanti di A.

Esempio:



Notare: Non possiamo applicare l'algoritmo a NFA.

Per esempio, per minimizzare



rimuoviamo lo stato C.

Ma $A \not\equiv C$.