

# Equivalenza e minimizzazione

**Problema:** due descrizioni di linguaggi regolari denotano lo stesso linguaggio?

- vedremo un algoritmo che permette di stabilire se due FA definiscono lo stesso linguaggio, sono equivalenti
- una conseguenza importante, è l'esistenza di un modo per minimizzare un automa, ovvero per trovare l'automa minimo (con il minor numero di stati) equivalente
- l'automa minimo è unico a meno di ridenominazione degli stati

# Stati equivalenti

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

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

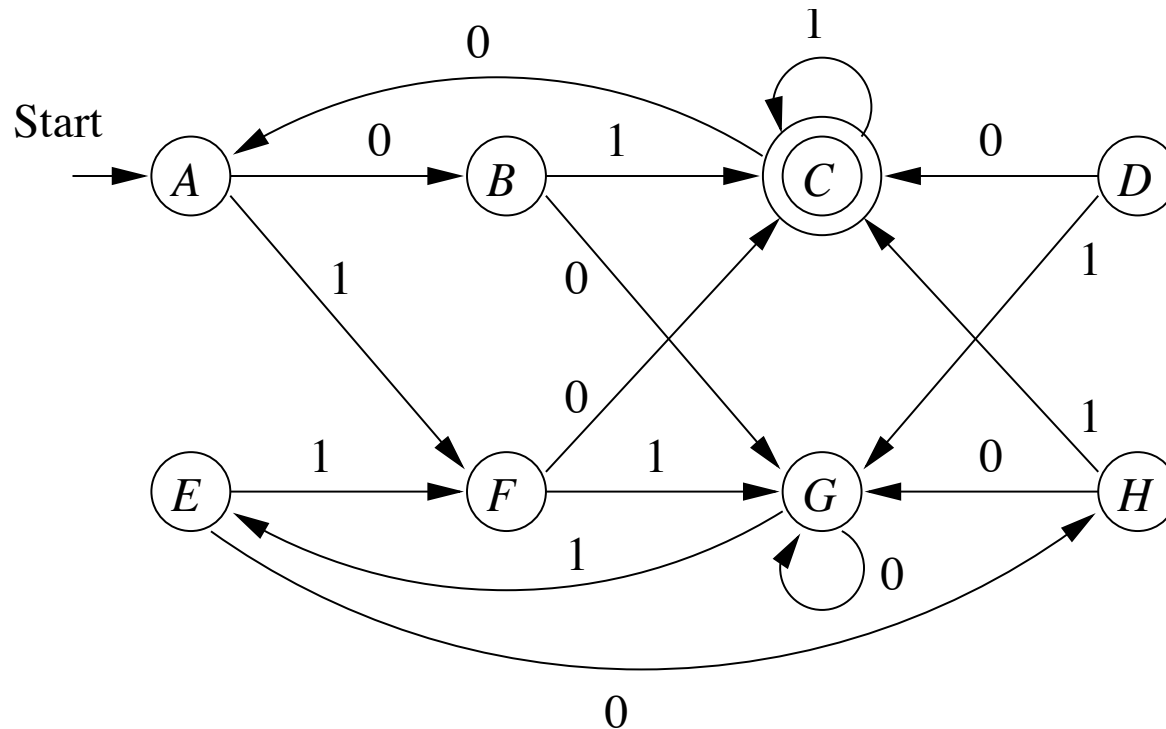
- Se  $p \equiv q$  diciamo che  $p$  e  $q$  sono *equivalenti*
- Se  $p \not\equiv q$  diciamo che  $p$  e  $q$  sono *distinguibili*

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

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

Ovvero se e solo se esiste almeno una stringa  $w$  che li differenzia, ovvero l'automa si muove da  $p$  in uno stato accettante e da  $q$  in uno stato non accettante (o viceversa).

# Esempio



$$\hat{\delta}(C, \epsilon) \in F, \hat{\delta}(G, \epsilon) \notin F \Rightarrow C \neq G$$

$$\hat{\delta}(A, 01) = C \in F, \hat{\delta}(G, 01) = E \notin F \Rightarrow A \neq G$$

## Stati distinguibili

Alcuni stati distinguibili:

- $C \neq G$  infatti  $C$  è accettante e  $G$  no. Formalmente, la stringa che li distingue è la stringa vuota,

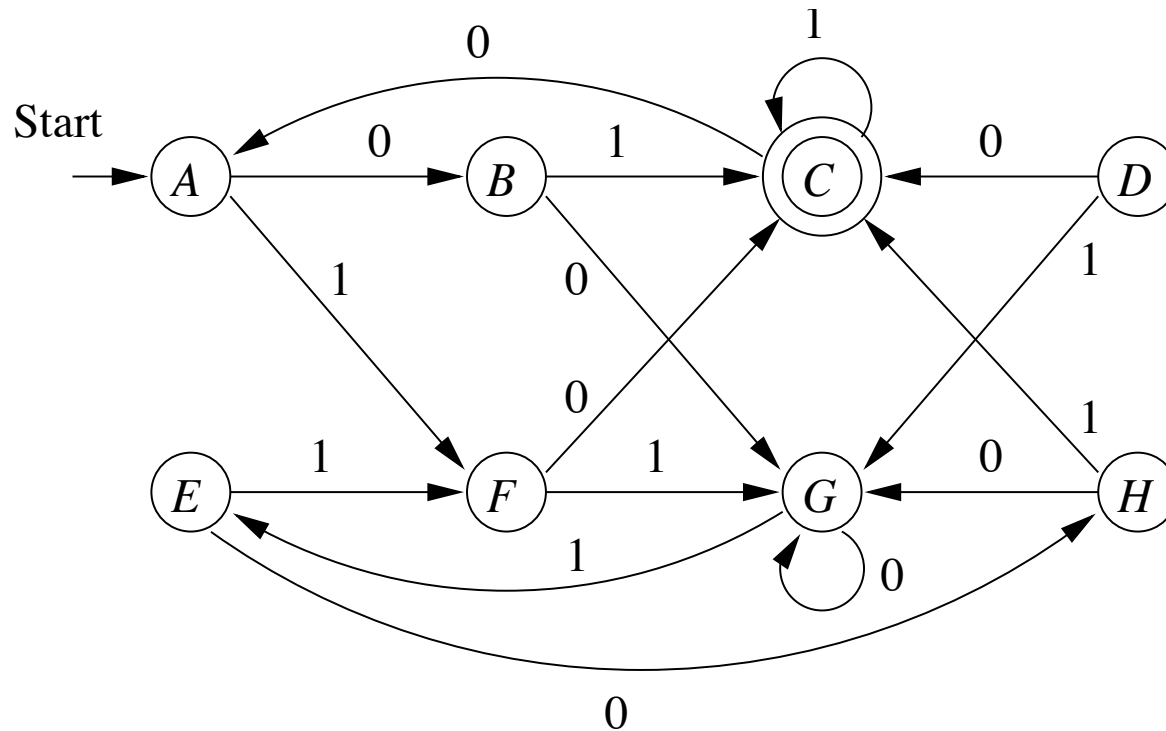
$$\hat{\delta}(C, \epsilon) \in F, \hat{\delta}(G, \epsilon) \notin F$$

- Consideriamo  $A$  e  $G$ . Non sono distinti da  $\epsilon$  (entrambi non sono accettanti). Non sono distinti da  $0$  perché porta  $A$  in  $B$  e  $G$  in se stesso, ma entrambi non sono accettanti. Analogamente non sono distinti da  $1$  perché porta  $A$  in  $F$  e  $G$  in  $E$ , ma entrambi non sono accettanti. Abbiamo tuttavia:

$$\hat{\delta}(A, 01) = C \in F, \hat{\delta}(G, 01) = E \notin F \Rightarrow A \neq G$$

Dato che  $C$  è accettante e  $E$  non lo è, la stringa  $01$  distingue  $A$  da  $G$ , ovvero  $A \neq G$ .

# Cosa si può dire su $A$ e $E$ ?



$$\hat{\delta}(A, \epsilon) = A \notin F, \hat{\delta}(E, \epsilon) = E \notin F$$

$$\hat{\delta}(A, 1) = F = \hat{\delta}(E, 1)$$

$$\text{Quindi } \hat{\delta}(A, 1x) = \hat{\delta}(E, 1x) = \hat{\delta}(F, x)$$

$$\hat{\delta}(A, 00) = G = \hat{\delta}(E, 00)$$

$$\hat{\delta}(A, 01) = C = \hat{\delta}(E, 01)$$

Conclusione:  $A = E$

# Algoritmo induttivo

Possiamo calcolare *tutte e sole* le coppie di stati distinguibili con il seguente metodo induttivo (**algoritmo di riempimento-tabella**):

Formalmente, calcoliamo la minima relazione  $R_A \subseteq Q \times Q$  tale che

**Base:** Se  $p \in F$  e  $q \notin F$ , allora  $(p, q) \in R_A$ , ovvero  $p \neq q$ .

**Induzione:** Se  $\exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in R_A$ , ovvero  $\delta(p, a) \neq \delta(q, a)$ , allora  $(p, q) \in R_A$ , ovvero  $p \neq q$ .

# Algoritmo induttivo

Applichiamo l'algoritmo ad A:

<i>B</i>	<i>x</i>						
<i>C</i>	<i>x</i>	<i>x</i>					
<i>D</i>	<i>x</i>	<i>x</i>	<i>x</i>				
<i>E</i>		<i>x</i>	<i>x</i>	<i>x</i>			
<i>F</i>	<i>x</i>	<i>x</i>	<i>x</i>		<i>x</i>		
<i>G</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	
<i>H</i>	<i>x</i>		<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

Le coppie *marcate* nella tabella stanno in  $R_A$  (sono distinguibili).

Le coppie  $(E, A)$ ,  $(B, H)$ ,  $(F, D) \notin R_A$  sono le uniche che risultano non distinguibili, equivalenti.

## Algoritmo: dettagli

- 1 Marcare tutte le coppie  $(q_i, q_j)$ .  $q_i \in F \wedge q_j \notin F$  o viceversa: dato che  $C$  è l'unico stato accettante, l'algoritmo parte con  $(C, q) \in R_A$  per qualsiasi altro stato  $q$
- 2 Marcare ogni coppia non ancora marcata  $(q_i, q_j)$  se  $(\delta(q_i, a), \delta(q_j, a))$  ( $a \in \Sigma$ ) è già marcata: marchiamo gli stati  $(p, q)$  che si muovono con  $a$  in una coppia di stati del punto 1. Ad es. la coppia  $(E, F)$  viene aggiunta perché si muove in  $(H, C)$  con input 0. Analogamente, vengono aggiunte tutte le altre coppie, ad eccezione di  $(A, G)$  ed  $(E, G)$ .
- 3 Ripetere il passo precedente fino a che non si possono marcare ulteriori stati: al passo successivo si aggiungono le coppie  $(A, G)$  ed  $(E, G)$ . Infatti in entrambi i casi i due stati si muovono, leggendo 1, negli stati  $(E, F)$ , che sono stati aggiunti al passo 2.
- 4 Le coppie rimaste  $(q_i, q_j) \notin R_A$  sono di stati **indistinguibili**: non ci sono altre coppie da aggiungere, rimangono fuori solo





## Variante al modo di procedere (Algoritmo di Hopcroft)

- 1 Si inizia proponendo una prima partizione: gli stati in  $F$  e quelli  $Q \setminus F$ , in modo da non avere nella partizione finale nessuna classe con stati finale e non finali.
- 2 Poi iterativamente si affina questa partizione, fino ad arrivare alle classe finali: per ogni stato  $q$  di una classe si osserva dove si arriva con  $0$ : se non si rimane nella stessa classe di  $q$ , allora  $q$  forma una nuova classe insieme agli stati che si comportano come  $q$ , ovvero che su  $0$  vanno nello stesso stato raggiunto da  $q$  su  $0$ . Si ripete il procedimento su 1. A questo punto si procede con la intersezione delle partizioni ottenute e si ripete il procedimento a partire dalle nuove classi.

## Altro modo di procedere (2)

- 1 Dividiamo gli stati in  $\{C\}$  e  $\{A, B, D, E, F, G, H\}$ .
- 2 Osserviamo che con 0: da  $A$  si va in  $B$  (stessa classe), da  $B$  si va in  $G$  (stessa classe), mentre da  $D$  si va in  $C$  (l'altra classe), da  $E$  si va in  $H$  (stessa classe), da  $F$  si va in  $C$  (altra classe, come  $G$ ), da  $G$  si va in  $G$  (stessa classe), infine da  $H$  si va in  $G$  (stessa classe). Nuova partizione della classe degli stati non finali:  $\{A, B, E, G, H\}$  e  $\{D, F\}$ .
- 3 Osserviamo che con 1: da  $A$  si va in  $F$  (stessa classe), da  $B$  si va in  $C$  (altra classe), mentre da  $D$  si va in  $G$  (stessa classe), da  $E$  si va in  $F$  (stessa classe), da  $F$  si va in  $G$  (stessa classe), da  $G$  si va in  $E$  (stessa classe), infine da  $H$  si va in  $C$  (altra classe). Nuova partizione della classe degli stati non finali:  $\{A, D, E, F, G\}$  e  $\{B, H\}$ .
- 4 Intersecando otteniamo:  $\{A, E, G\}$ ,  $\{B, H\}$ ,  $\{C\}$ ,  $\{D, F\}$ .
- 5 Ripetiamo. In particolare osserviamo che  $A$  ed  $E$  si comportano nello stesso modo, e diversamente da  $G$ . Alla fine

# Correttezza dell'algoritmo

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

$$(p, q) \notin R_A \text{ sse } p \equiv q$$

**Dimostrazione:** 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  $q$ , ovvero  $(p, q) \notin R_A$ .

Sia  $w = a_1 a_2 \cdots a_n$  la stringa più corta che identifica la coppia "sbagliata"  $\{p, q\}$ , ovvero  $\hat{\delta}(p, w) \in F$  and  $\hat{\delta}(q, w) \notin F$ .

Allora  $w \neq \epsilon$  perché altrimenti l'algoritmo distinguerebbe  $p$  da  $q$  (caso base). Quindi  $n \geq 1$  (deve esserci almeno un simbolo  $a_1$ ).

- Consideriamo gli stati  $r = \delta(p, a_1)$  e  $s = \delta(q, a_1)$
- Notiamo che

$$\hat{\delta}(p, w) = \hat{\delta}(r, a_2 \cdots a_n) \in F \text{ and } \hat{\delta}(q, w) = \hat{\delta}(s, a_2 \cdots a_n) \notin F$$

Allora  $\{r, s\}$  non può essere una coppia sbagliata perché  $\{r, s\}$  sarebbe identificata da una stringa più corta di  $w$ .

- Quindi, l'algoritmo deve aver scoperto nel caso base che  $r$  and  $s$  sono distinguibili, ovvero  $(r, s) \in R_A$ .
- Ma allora l'algoritmo distinguerebbe  $p$  da  $q$  nella parte induttiva. Contraddizione.
- Quindi non ci sono coppie "sbagliate" e il teorema è vero.

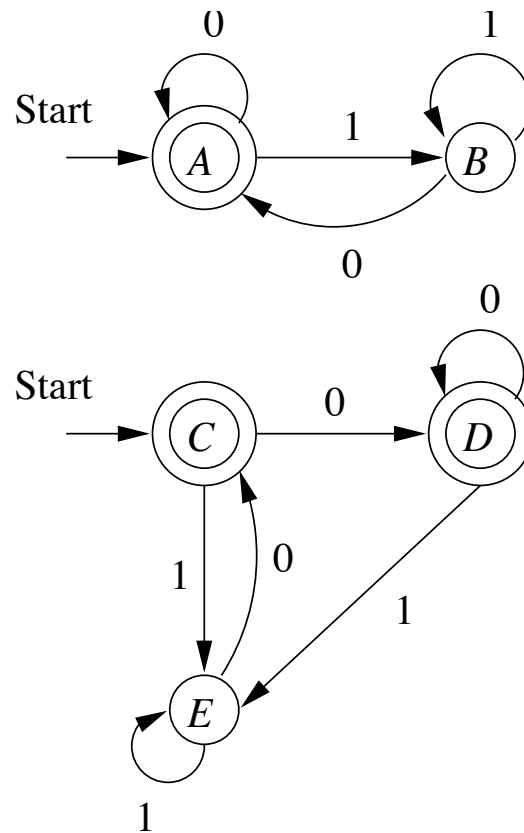
# Testare l'equivalenza di linguaggi regolari

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 sia 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 è

<i>B</i>	$x$			
<i>C</i>		$x$		
<i>D</i>		$x$		
<i>E</i>	$x$		$x$	$x$
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>

La coppia di stati iniziali  $(A, C)$  non è marcata. 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  $p/\equiv$ .
- 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).



# Classi di equivalenza

<i>B</i>	<i>x</i>						
<i>C</i>	<i>x</i>	<i>x</i>					
<i>D</i>	<i>x</i>	<i>x</i>	<i>x</i>				
<i>E</i>		<i>x</i>	<i>x</i>	<i>x</i>			
<i>F</i>	<i>x</i>	<i>x</i>	<i>x</i>		<i>x</i>		
<i>G</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	
<i>H</i>	<i>x</i>		<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

Il primo DFA esaminato ha le seguenti classi di equivalenza:

$$\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$$

# Transitività

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

**Dimostrazione:** 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  $\hat{\delta}(q, w)$  o è di accettazione o no.
- *Caso 1:*  $\hat{\delta}(q, w)$  è di accettazione. Allora  $q \not\equiv r$ .
- *Caso 2:*  $\hat{\delta}(q, w)$  non è di accettazione. Allora  $p \not\equiv q$ .
- Il caso contrario può essere dimostrato 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

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

ovvero le mosse della classe di equivalenza dello stato  $p$ , per ogni  $a \in \Sigma$  sono la classe di equivalenza trovata tramite  $\delta$  per ogni stato  $p \in p/\equiv$ .

**Nota** Affinché  $B$  sia ben definito, dobbiamo mostrare che

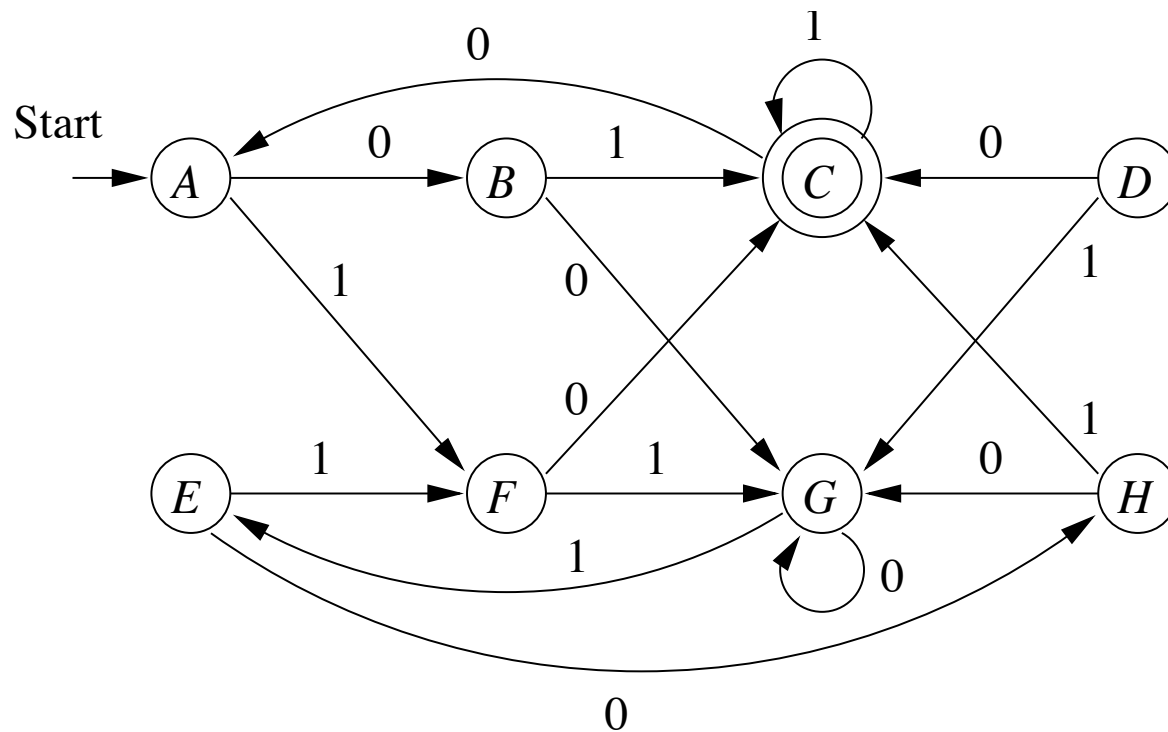
$$\text{Se } p \equiv q \text{ allora } \delta(p, a) \equiv \delta(q, a)$$

Se  $\delta(p, a) \not\equiv \delta(q, a)$ , allora l'algoritmo concluderebbe  $p \not\equiv q$ , quindi  $B$  è ben definito.

Analogamente, si può mostrare che  $F/\equiv$  contiene tutti e soli gli stati accettanti di  $A$ .

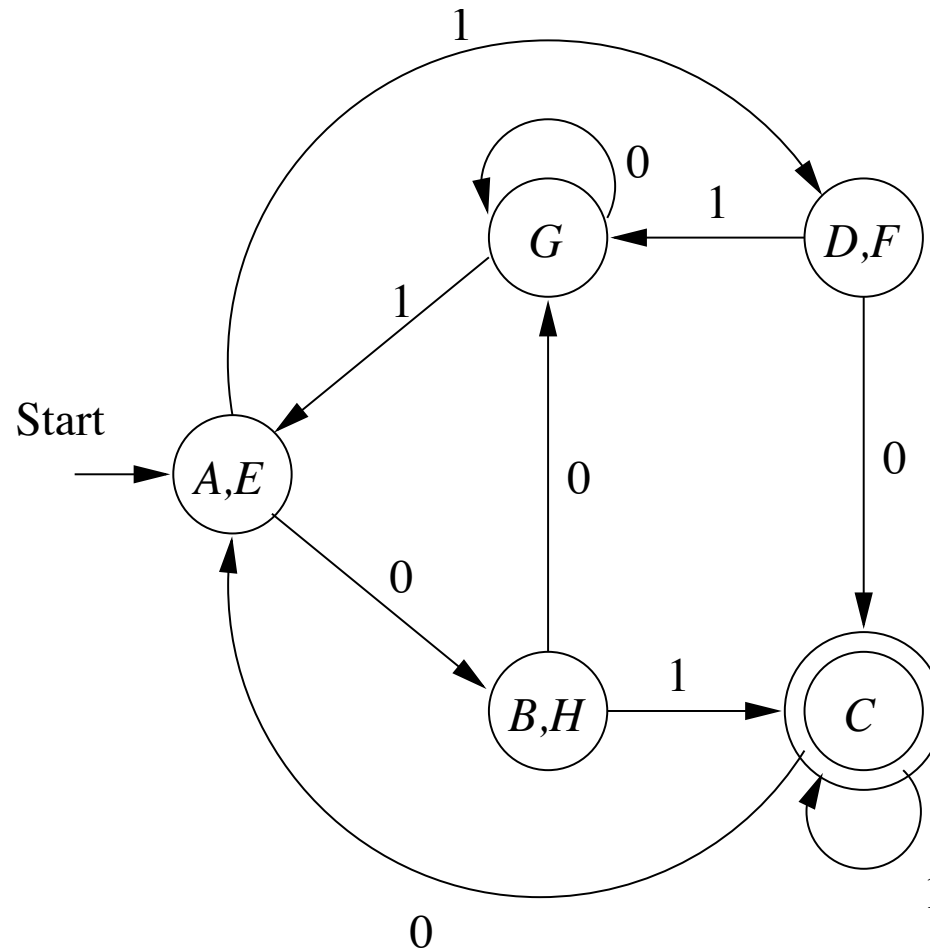
# Esempio

Possiamo minimizzare



# Esempio

Otteniamo:



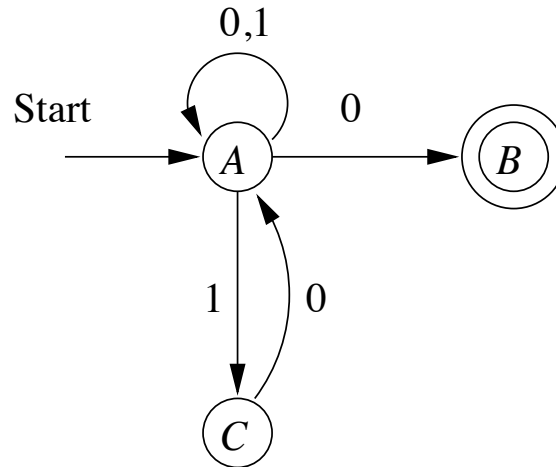
# Commenti

- Gli stati sono le classi di equivalenza

$$\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$$

- lo stato iniziale è la classe di equivalenza dello stato iniziale  $A$ , ovvero  $\{A, E\}$
- gli stati accettanti sono le classi di equivalenza dell'unico stato accettante  $C$ , ovvero  $\{C\}$
- per ogni classe di equivalenza e per ogni simbolo, tutta la classe si muove in modo uniforme. Esempio:  $\{A, E\}$  leggendo 0 si muove in  $\{B, H\}$ .

**Nota:** Non possiamo applicare l'algoritmo agli NFA. Proviamo a minimizzare, ad esempio, l'NFA che riconosce il linguaggio  $(0 + 1)^*(10)^*(0 + 1)^*0$



- Intuitivamente possiamo rimuovere lo stato  $C$  e le sue transizioni.
- L'algoritmo tuttavia non riporterebbe nessuna coppia di stati equivalenti
- $B$  è finale e distinguibile da  $A$  e  $C$
- Si può togliere lo stato  $C$  e il ciclo, ma  $A \not\equiv C$ , quindi l'automa non è minimo.

# Perché non si può migliorare il DFA minimizzato

- Sia  $B$  il DFA minimizzato ottenuto applicando l'algoritmo al DFA  $A$ .
- Sappiamo già 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  $p \equiv q$ .
- **Dimostrazione:**
  - 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 è una contraddizione, dato che  $B$  è stato costruito dall'algoritmo.