

## Proprieta' dei linguaggi regolari

# Proprieta' dei Linguaggi regolari

- **Pumping Lemma**

Ogni linguaggio regolare soddisfa il Pumping Lemma. Per dimostrare che un linguaggio non e' regolare useremo il pumping lemma e arriveremo a una contraddizione.

- **Proprieta' di chiusura**

Se due linguaggi  $L$  e  $M$  sono regolari, anche i linguaggi  $L \cap M$ ,  $L \cup M$ ,  $\bar{L}$ ,  $L \setminus M$ ,  $L^R$ ,  $L^*$ ,  $L.M$  sono regolari

- **Tecniche di minimizzazione**

Possiamo risparmiare costruendo automi piu' piccoli.

## 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:

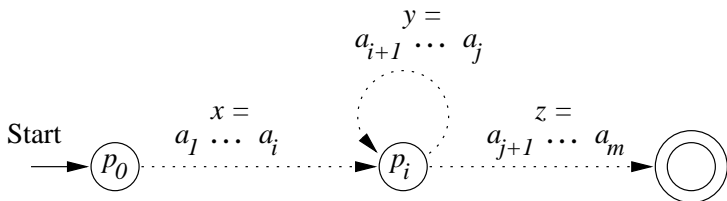
- 1  $y \neq \epsilon$
- 2  $|xy| \leq n$
- 3  $\forall k \geq 0, xy^kz \in L$

# Prova del Pumping Lemma

- Supponiamo che  $L$  sia regolare.
- Allora  $L$  è riconosciuto da un DFA  $A$  con, ad esempio,  $n$  stati.
- Sia  $w = a_1 a_2 \dots a_m \in L$ ,  $m \geq n$ .
- Per  $i = 0, \dots, n$  sia  $p_i = \hat{\delta}(q_0, a_1 a_2 \dots a_i)$   
 $p_i$  è lo stato in cui si trova  $A$  dopo aver letto  $i$  simboli.  
Si noti che  $p_0 = q_0$ .
- Dato che ci sono solo  $n$  stati in  $A$ , gli  $n + 1$  stati  $p_i$ , per  $i = 0, \dots, n$  non possono essere tutti distinti  
 $\Rightarrow \exists i < j : p_i = p_j$

Possiamo scomporre  $w = xyz$ :

- ①  $x = a_1 a_2 \cdots a_i$  ( $x$  porta da  $q_0$  a  $p_i$ ,  $x$  può essere vuota)
- ②  $y = a_{i+1} a_{i+2} \cdots a_j$  ( $y$  porta da  $p_i$  a  $p_j$ ,  $y$  non può essere vuota)
- ③  $z = a_{j+1} a_{j+2} \cdots a_m$  ( $z$  porta da  $p_j$  ad uno stato accettore,  $z$  può essere vuota)



Quindi anche  $xy^kz \in L$ , per ogni  $k \geq 0$ .

## Tecnica dei due giocatori

- Possiamo interpretare l'applicazione del Pumping Lemma come **gioco a due giocatori G1 e G2**
  - giocatore **G1** vuol dimostrare che  $L$  non è regolare
  - giocatore **G2** sceglie  $n$  (ma non deve comunicarlo a G1)
  - giocatore **G1** sceglie  $w \in L$ , che può dipendere da  $n$ , tale che  $|w| \geq n$
  - giocatore **G2** sceglie la partizione  $w = xyz$ , con  $|xy| \leq n$  e  $y \neq \epsilon$  (ma non deve comunicarla a G1)
  - giocatore **G1** sceglie  $k$ , che può dipendere da  $x, y, z$  e  $n$ , tale che  $xy^kz \notin L$ , così viola il Pumping lemma.
  - giocatore **G1** conclude che  $L$  non è regolare.

## Esercizio 1

- Provare con il Pumping Lemma che  $L$  non è regolare,  
 $L =$  linguaggio delle stringhe su  $\{0, 1\}$  con ugual numero di 0 e 1.
- Soluzione:
  - giocatore **G1** vuol dimostrare che  $L$  non è regolare
  - giocatore **G2** sceglie  $n$
  - giocatore **G1** sceglie  $w = 0^n 1^n \in L$ , con  $|w| \geq n$
  - giocatore **G2** sceglie la partizione  $w = xyz$  con  $|xy| \leq n$  e  $y \neq \epsilon$
  - giocatore **G1** sceglie  $k$  tale che  $xy^k z \notin L$ .  
**G1 sceglie  $k = 0$ .** G1 deve dimostrare che  $xy^0 z = xz \notin L$ .
    - Poiché  $|xy| \leq n$  allora, sia  $x$  che  $y$  sono composti solo da zeri poiché abbiamo scelto  $w = 0^n 1^n$ . Quindi  $xz = 0^{n-|y|} 1^n$ .
    - Poiché  $y \neq \epsilon$ , allora  $|y| \neq 0$ . Quindi  $n - |y| \neq n$ .
    - Perciò  $xz = 0^{n-|y|} 1^n \notin L$  poiché il numero di zeri è minore del numero di uni. G1 conclude che  **$L$  non è regolare.**

## Esercizio 2

- Provare con il Pumping Lemma che  $L$  non è regolare:  
 $L = \{vv^R, \text{ dove } v^R \text{ è la stringa } v \text{ rovesciata e } v \in \{0,1\}^*\}$
- Soluzione:
  - giocatore G1 vuol dimostrare che  $L$  non è regolare
  - giocatore G2 sceglie  $n$
  - giocatore G1 sceglie  $w = 0^n 1^n 1^n 0^n \in L$ , con  $|w| \geq n$
  - giocatore G2 sceglie la partizione  $w = xyz$  con  $|xy| \leq n$  e  $y \neq \epsilon$
  - giocatore G1 sceglie  $k = 0$  e deve dimostrare che  $xy^0z = xz \notin L$ .
    - Poiché  $|xy| \leq n$  allora sia  $x$  che  $y$  sono composti solo da zeri, poiché abbiamo scelto  $w = 0^n 1^n 1^n 0^n$ . Quindi  $xz = 0^{n-|y|} 1^n 1^n 0^n$ .
    - Poiché  $y \neq \epsilon$ , allora  $|y| \neq 0$ . Quindi  $n - |y| \neq n$ .
    - Perciò  $xz = 0^{n-|y|} 1^n 1^n 0^n \notin L$ , quindi  $L$  non è regolare.



## Esercizio 3

- Provare con il Pumping Lemma che il linguaggio delle palindromi di lunghezza dispari su alfabeto  $\{0, 1\}$  non è regolare.
- Soluzione:
  - giocatore G1 vuol dimostrare che  $L$  non è regolare
  - giocatore G2 sceglie  $n$
  - giocatore G1 sceglie  $w = 0^n 1 0^n \in L$ , con  $|w| \geq n$
  - giocatore G2 sceglie la partizione  $w = xyz$ ,  $|xy| \leq n$  e  $y \neq \epsilon$
  - giocatore G1 sceglie  $k = 0$  e deve dimostrare che  $xy^0z = xz \notin L$ .
    - Poiché  $|xy| \leq n$  allora sia  $x$  che  $y$  sono composti solo da zeri, poiché abbiamo scelto  $w = 0^n 1 0^n$ . Quindi  $\underline{xz = 0^{n-|y|} 1 0^n}$ .
    - Poiché  $y \neq \epsilon$ , allora  $|y| \neq 0$ . Quindi  $n - |y| \neq n$ .
    - Perciò  $\underline{xz = 0^{n-|y|} 1 0^n \notin L}$ , quindi  $L$  non è regolare.

# Esercizi

- Esercizi per casa
  - Dimostrare con il Pumping Lemma che il linguaggio  $L = \{v\bar{v}, \text{ dove } \bar{v} \text{ è la stringa } v \text{ in cui ogni } 0 \text{ e' sostituito con } 1 \text{ e ogni } 1 \text{ e' sostituito con } 0, v \in \{0, 1\}^*\}$  non è regolare.
  - Dimostrare con il Pumping Lemma che il linguaggio  $L = \{v1^n, v \in \{0, 1\}^* \text{ con } |v| = n\}$  non è regolare.
  - Esercizi delle dispense sul Pumping Lemma

## 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:*  $\bar{L}$
- *Differenza:*  $L \setminus M$
- *Inversione:*  $L^R = \{w^R : w \in L\}$
- *Chiusura:*  $L^*$ .
- *Concatenazione:*  $L.M$

## Chiusura rispetto a unione e complemento

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

**Prova.** Poiché  $L$  e  $M$  sono linguaggi regolari, sono descritti da espressioni regolari. Sia  $L = L(E)$  e  $M = L(F)$  dove  $E$  e  $F$  sono espressioni regolari. Allora  $L \cup M = L(E) \cup L(F) = L(E + F)$  per definizione dell'operatore  $+$  delle espressioni regolari.

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

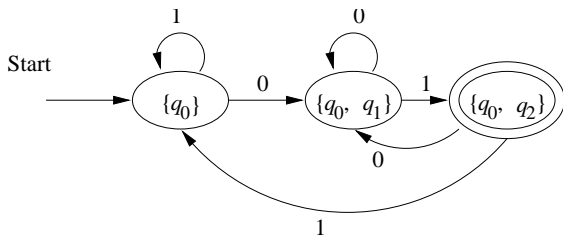
**Prova.** Sia  $L$  riconosciuto da un DFA

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

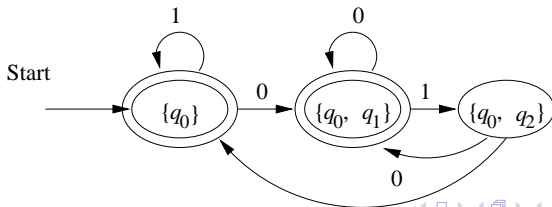
Sia  $B = (Q, \Sigma, \delta, q_0, Q \setminus F)$ . Allora  $L(B) = \bar{L}$ .

## Esempio

Sia  $L$  riconosciuto dal DFA qui sotto:



Allora  $\bar{L}$  e' riconosciuto da:



## Chiusura rispetto all'intersezione

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

**Prova 1.** 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.

## Chiusura rispetto alla differenza

**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.

## Chiusura rispetto al "reverse"

$L^R$  e' il linguaggio formato dalle inversioni di tutte le stringhe di  $L$ .  
L'inversione di una stringa  $a_1a_2\dots a_{n-1}a_n$  e' la stringa scritta al contrario  $a_na_{n-1}\dots a_2a_1$

**Teorema 4.11** Se  $L$  e' un linguaggio regolare, allora anche  $L^R$  e' 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).



## Esercizi

- Esercizio (Problema 2 - I compito 2013)  
Commentare la veridicità delle seguenti affermazioni, giustificando la risposta:
  - 1 L'intersezione di un linguaggio non regolare  $L_1$  e di un linguaggio regolare  $L_2$  può essere non regolare;
  - 2 L'intersezione di un linguaggio non regolare e di un linguaggio finito è sempre regolare;
  - 3 Ogni sottoinsieme di un linguaggio non regolare è non regolare.
- Esercizi per casa
  - Esercizi sulle proprietà dei linguaggi delle dispense

## Esercizio

- Esercizio (Problema 2 - I compito 2013)

Commentare la veridicità delle seguenti affermazioni:

- 1 L'intersezione di un linguaggio non regolare  $L_1$  e di un linguaggio regolare  $L_2$  può essere non regolare;  
Soluzione: VERO. Infatti, se ad esempio  $L_2 = \Sigma^*$ , allora  $L_1 \cap L_2 = L_1$  che è non regolare.
- 2 L'intersezione di un linguaggio non regolare e di un linguaggio finito è sempre regolare;  
Soluzione: VERO. Infatti l'intersezione di un qualunque linguaggio con un linguaggio finito è sempre un linguaggio finito e ogni linguaggio finito è regolare.
- 3 Ogni sottoinsieme di un linguaggio non regolare è non regolare.  
Soluzione: FALSO. Infatti  $\emptyset$  è sottoinsieme di tutti i linguaggi e quindi anche di quelli non regolari e  $\emptyset$  è regolare.

# Equivalenza e minimizzazione di automi

## Stati equivalenti

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

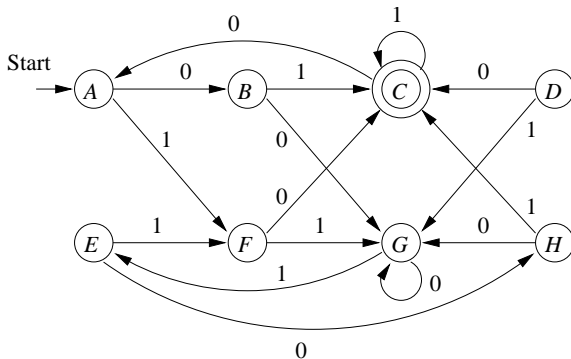
$$p \equiv q \Leftrightarrow \forall w \in \Sigma^* : \hat{\delta}(p, w) \in F \text{ se e solo se } \hat{\delta}(q, w) \in F$$

- 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

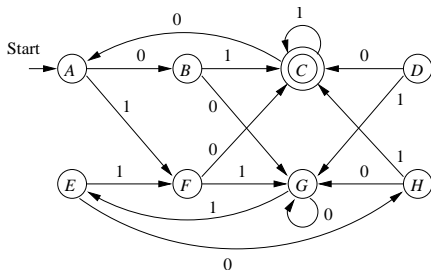
$$\exists w : \hat{\delta}(p, w) \in F \text{ e } \hat{\delta}(q, w) \notin F, \text{ o viceversa}$$

## Esempio



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

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

Cosa si può dire su  $A$  e  $E$ ?

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

$$\hat{\delta}(A, 0) = B \notin F, \quad \hat{\delta}(E, 0) = H \notin F$$

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

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

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

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

Conclusione:  $A \equiv E$

# Algoritmo induttivo

Possiamo calcolare coppie di stati distinguibili con il seguente metodo induttivo (**algoritmo riempi-tabella**):

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

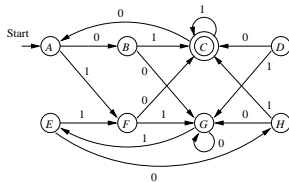
**Induzione:** Se  $\exists a \in \Sigma : \delta(p, a) \neq \delta(q, a)$ , allora  $p \neq q$ .

Esempio: Applichiamo l'algoritmo all'automa precedente:

<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>

*A B C D E F G*

# Algoritmo riempi-tabella sull'esempio



- Poiche'  $C \in F$  e  $A, B, D, E, F, H \notin F$ ,  
 $C \neq A$ ,  $C \neq B$ ,  $C \neq D$ ,  $C \neq E$ ,  $C \neq F$ ,  $C \neq G$ ,  $C \neq H$
- Poiche'  $C \neq H$ ,
  - visto che  $\delta(F, 0) = C$  e  $\delta(E, 0) = H$ , allora  $F \neq E$
  - visto che  $\delta(D, 0) = C$  e  $\delta(E, 0) = H$ , allora  $D \neq E$
- Poiche'  $F \neq E$ ,
  - visto che  $\delta(A, 1) = F$  e  $\delta(G, 1) = E$ , allora  $A \neq G$
  - visto che  $\delta(E, 1) = F$  e  $\delta(G, 1) = E$ , allora  $E \neq G$
- .... (Completarlo per casa)



## 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  $p \not\equiv q : \exists w : \hat{\delta}(p, w) \in F, \hat{\delta}(q, w) \notin F$ , 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  $s = \delta(q, a_1)$ .
- $r$  e  $s$  sono **distinti dalla stringa**  $a_2, \dots, a_n$ , poiché li porta rispettivamente in  $\hat{\delta}(p, w) \in F$  e  $\hat{\delta}(q, w) \notin F$ .
- Ma la stringa che distingue  $r$  e  $s$  è **più corta** di ogni stringa che distingue una coppia "sbagliata"
- Allora  $\{r, s\}$  **non** possono essere una coppia "sbagliata".
- Quindi **l'algoritmo riempi-tabella** deve avere scoperto 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 è vero.

L'algoritmo riempi tabella e' utile per

- Testare l'equivalenza di linguaggi regolari
- Costruire il DFA minimo

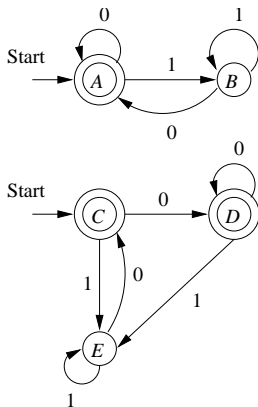
# Testare l'equivalenza di linguaggi regolari

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  $L(\epsilon + (0 + 1)^*0)$ .

## 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>

Poiché *A* e *C* sono equivalenti, i due automi sono equivalenti.

# Minimizzazione di DFA

## Algoritmo di minimizzazione di un DFA:

- 1 Per prima cosa si **eliminano gli stati irraggiungibili** dallo stato iniziale
- 2 Si applica **l'algoritmo riempi-tabella solo per gli stati rimanenti**: tali stati vengono ripartiti in blocchi in modo che tutti gli stati di uno stesso blocco siano tutti equivalenti e due stati in blocchi diversi non siano mai equivalenti

# Minimizzazione di DFA

- Possiamo usare l'algoritmo riempi-tabella per minimizzare un DFA mettendo insieme tutti gli stati equivalenti.  
Cioè **rimpiazzando**  $p$  con  $p/\equiv$ .
- Esempio: Il DFA di prima ha le seguenti classi di equivalenza:  $\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$ .
- Notare: affinché  $p/\equiv$  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

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

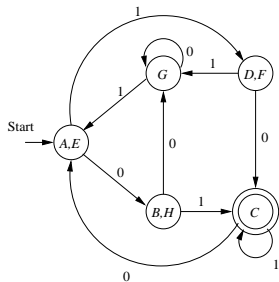
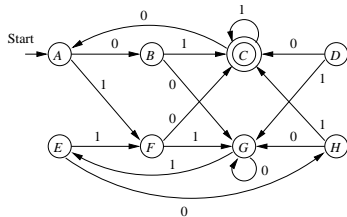
Se  $\delta(p, a) \not\equiv \delta(q, a)$ , allora l'algoritmo concluderebbe  $p \not\equiv q$ , quindi  $B$  e' ben definito.

Notare anche che  $F/\equiv$  contiene tutti e soli gli stati accettanti di  $A$ .

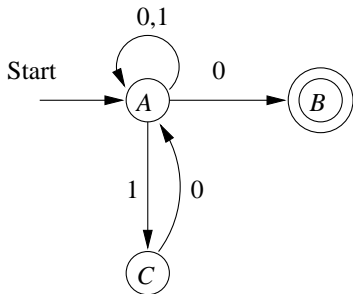
Il DFA minimo non puo' essere migliorato

# Esempio

Costruiamo il DFA minimo dell'automata seguente



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



rimuoviamo lo stato  $C$  e otteniamo un automa più piccolo  
Ma  $A \neq C$  e quindi l'algoritmo riempi-tabella non ci avrebbe  
consentito di rimuovere  $C$ .

# Esercizio

• Esercizio per casa: Problema 2 - Il appello 2013

- ① Dare la definizione di stati equivalenti.
- ② Costruire il **diagramma** di transizione dell'automa che minimizza l'automa seguente spiegando **sinteticamente** il procedimento seguito.

$Q = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, \delta, q_0, \{q_3, q_5\})$  dove  $\delta$  è descritta dalla seguente tabella di transizione:

	0	1
$\rightarrow q_0$	$q_1$	$q_3$
$q_1$	$q_0$	$q_3$
$q_2$	$q_1$	$q_4$
$*q_3$	$q_5$	$q_5$
$q_4$	$q_3$	$q_3$
$*q_5$	$q_5$	$q_5$