Automi e Linguaggi Formali

5. Proprietà di decisione dei linguaggi regolari

Davide Bresolin

a.a. 2018/19

Università degli Studi di Padova

Primo compitino

Venerdì 12 Aprile – ore 12:30

Aule LuM250 e P200

- La lista su uniweb è aperta
- Si chiude Mercoledì 10 aprile
- Giovedì 11 verrà pubblicata sul moodle la ripartizione tra le due aule degli iscritti

Proprietà dei linguaggi regolari

- **Proprietà di chiusura.** Come costruire automi da componenti usando delle operazioni, ad esempio dati L e M possiamo costruire un automa per $L \cap M$.
- **Pumping Lemma.** Ogni linguaggio regolare soddisfa il pumping lemma. Se qualcuno vi presenta un falso linguaggio regolare, l'uso del pumping lemma mostrerà una contraddizione.
- **Proprietà di decisione.** Analisi computazionale di automi, cioè quanto costa controllare varie proprietà, come l'equivalenza di due automi.

Proprietà di chiusura

Università degli Studi di Padova

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

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

Theorem (Pumping Lemma per Linguaggi Regolari)

Sia L un linguaggio regolare. Allora

- *esiste una lunghezza $h \geq 0$ tale che*
- *ogni parola $w \in L$ di lunghezza $|w| \geq h$*
- *può essere spezzata in $w = xyz$ tale che:*
  1. $y \neq \varepsilon$ *(il secondo pezzo è non vuoto)*
  2. $|xy| \leq h$ *(i primi due pezzi sono lunghi al max h)*
  3. $\forall k \geq 0$, $xy^kz \in L$ *(possiamo "pompare" y rimanendo in L)*

Pumping Lemma: esercizi

2. Il linguaggio $L_{rev} = \{ww^R : w \in \{a, b\}^*\}$ è regolare?

2 Il linguaggio $L_{rev} = \{ww^R : w \in \{a,b\}^*\}$ è regolare?

No, L_{rev} non è regolare:

- supponiamo per assurdo che lo sia
- sia h la lunghezza data dal Pumping Lemma
- consideriamo la parola $w = a^h bba^h$
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq h$:
  $w = \underbrace{aaa\dots}_{x}\underbrace{\dots a}_{y}\underbrace{abbaaa\dots aaa}_{z}$
- poiché $|xy| \leq h$, le stringhe x e y sono fatte solo di a
- per il Pumping lemma, anche $xy^0z = xz \in L_{rev}$, ma non la posso spezzare in $ww^R \Rightarrow$ assurdo

Pumping Lemma: esercizi

4. Il linguaggio $L_p = \{1^p : p \text{ è primo}\}$ è regolare?

4 Il linguaggio $L_p = \{1^p : p \text{ è primo}\}$ è regolare?

No, L_p non è regolare:

- supponiamo per assurdo che lo sia
- sia h la lunghezza data dal Pumping Lemma
- consideriamo una parola $w = 1^p$ con p primo e $p > h + 2$
- sia $w = xyz$ una suddivsione di w tale che $y \neq \varepsilon$ e $|xy| \leq h$:

$$w = \underbrace{11\dots11}_{x}\underbrace{11\dots1}_{y}\underbrace{111\dots11}_{z}$$

- ...

Pumping Lemma: esercizi

- ...
- sia $|y| = m$: allora $|xz| = p - m$
- per il Pumping lemma, anche $v = xy^{p-m}z \in L_p$
- allora $|v| = m(p-m) + p - m = (p-m)(m+1)$ si può scomporre in due fattori
- poiché $y \neq \varepsilon$, allora $|y| = m > 0$ e $m + 1 > 1$
- anche $p - m > 1$ perché abbiamo scelto $p > h + 2$ e $m \leq h$ perché $|xy| \leq h$
- i due fattori sono entrambi maggiori di 1 e quindi $|v|$ non è un numero primo
- $v \notin L_{rev}$, assurdo

9 Il linguaggio $L_{exp} = \{1^{2^n} : n \geq 0\}$ è regolare?

Think: **due minuti** per pensare da soli a come risolvere l'esercizio

Pair: **cinque minuti** per arrivare ad una soluzione condivisa con il vostro compagno di banco

Share: **cinque minuti** per spiegare la vostra soluzione ad un'altra coppia di studenti.

Proprietà di decisione

Proprietà di decisione

- Convertire tra diverse rappresentazioni dei linguaggio regolari
- Il linguaggio L è vuoto?
- La parola w appartiene al linguaggio L?
- Due descrizioni definiscono lo stesso linguaggio?

Da NFA a DFA

- Supponiamo che l'ε-NFA di input abbia n stati.
- Per calcolare $\mathrm{ECLOSE}(p)$ seguiamo al più n^2 transizioni.
- Lo facciamo per n stati, quindi in totale sono n^3 passi.
- Il DFA ha 2^n stati, per ogni stato S e ogni simbolo $a \in \Sigma$ calcoliamo $\delta_D(S, a)$ in n^3 passi.
- In totale abbiamo $O(n^3 2^n)$ passi.
- Se calcoliamo δ solo per gli stati raggiungibili, dobbiamo calcolare $\delta_D(S, a)$ solo s volte, dove s è il numero di stati raggiungibili. In totale: $O(n^3 s)$ passi.

Da DFA a NFA

- Facile: basta mettere le parentesi graffe attorno agli stati nella tabella di transizione
- Numero di passi necessari: $O(n)$, se n è il numero di stati del DFA di input

Da FA a espressione regolare

- Supponiamo che l'FA di input abbia n stati ed un solo stato finale
- La procedura deve eliminare tutti gli n stati escluso quello finale e quello iniziale
- Uno stato che viene eliminato va sostituito con $O(n^2)$ transizioni
- Ogni transizione è etichettata con quattro espressioni provenienti dall'iterazione precedente
- Dobbiamo quindi calcolare n^3 cose di grandezza fino a 4^n. Totale: $O(n^3 4^n)$

Da espressione regolare a FA

- Supponiamo che l'espressione regolare di input sia lunga n simboli
- Possiamo costruire un albero che rappresenta l'espressione in n passi
- Possiamo costruire l'ε-NFA equivalente applicando le regole ad ogni nodo dell'albero
- Ogni regola aggiunge al massimo due stati e quattro transizioni all'automa
- Totale: O(n) stati e transizioni nell'ε-NFA

Controllare se un linguaggio è vuoto (1)

Università degli Studi di Padova

- $L(A) \neq \emptyset$ se e solo se esiste uno stato finale raggiungibile dallo stato iniziale.

Controllare se un linguaggio è vuoto (1)

- $L(A) \neq \emptyset$ se e solo se esiste uno stato finale raggiungibile dallo stato iniziale.

Base: lo stato iniziale è raggiungibile dallo stato iniziale

Induzione: se q è raggiungibile dallo stato iniziale e c'è una transizione da q a p con un qualsiasi simbolo (compreso ε), allora anche p è raggiungibile dallo stato iniziale

Controllare se un linguaggio è vuoto (1)

- $L(A) \neq \emptyset$ se e solo se esiste uno stato finale raggiungibile dallo stato iniziale.

  Base: lo stato iniziale è raggiungibile dallo stato iniziale

  Induzione: se q è raggiungibile dallo stato iniziale e c'è una transizione da q a p con un qualsiasi simbolo (compreso ε), allora anche p è raggiungibile dallo stato iniziale

- Se nel corso della procedura incontriamo uno stato finale:
  $\Rightarrow$ No, il linguaggio non è vuoto
- Se la procedura termina senza trovare stati finali:
  $\Rightarrow$ Si, il linguaggio è vuoto
- Se l'automa ha n stati, la procedura impiega un numero di passi $O(n^2)$

Controllare se un linguaggio è vuoto (2)

- In alternativa possiamo analizzare l'espressione regolare $\mathbf{E}$ e vedere se $L(\mathbf{E}) = \emptyset$ ragionando per casi:
  - $\mathbf{E} = \mathbf{F} + \mathbf{G}$. Allora $L(\mathbf{E})$ è vuoto se e solo se sia $L(\mathbf{F})$ e $L(\mathbf{G})$ sono vuoti;
  - $\mathbf{E} = \mathbf{F}.\mathbf{G}$. Allora $L(\mathbf{E})$ è vuoto se e solo $L(\mathbf{F})$ è vuoto oppure $L(\mathbf{G})$ è vuoto;
  - $\mathbf{E} = \mathbf{F}^*$. Allora $L(\mathbf{E})$ non è vuoto perche $\varepsilon \in L(\mathbf{E})$
  - $\mathbf{E} = \varepsilon$. Allora $L(\mathbf{E})$ non è vuoto perche $\varepsilon \in L(\mathbf{E})$
  - $\mathbf{E} = \mathbf{a}$. Allora $L(\mathbf{E})$ non è vuoto perche $a \in L(\mathbf{E})$
  - $\mathbf{E} = \emptyset$. Allora $L(\mathbf{E})$ è vuoto

Controllare l'appartenenza

- Per controllare se $w \in L(A)$ dobbiamo simulare A su w:
  - se $|w| = n$ dobbiamo fare n passi di simulazione
  - se A è un DFA ogni passo costa $O(1)$

    $\Rightarrow$ Totale: $O(n)$
  - se A è un NFA con s stati ogni passo costa $O(s^2)$

    $\Rightarrow$ Totale: $O(ns^2)$
  - se A è un ε-NFA con s stati ogni passo costa $O(s^2)$

    $\Rightarrow$ Totale: $O(ns^2)$

Controllare l'appartenenza

- Per controllare se $w \in L(A)$ dobbiamo simulare A su w:
  - se $|w| = n$ dobbiamo fare n passi di simulazione
  - se A è un DFA ogni passo costa $O(1)$

    $\Rightarrow$ Totale: $O(n)$
  - se A è un NFA con s stati ogni passo costa $O(s^2)$

    $\Rightarrow$ Totale: $O(ns^2)$
  - se A è un ε-NFA con s stati ogni passo costa $O(s^2)$

    $\Rightarrow$ Totale: $O(ns^2)$
- Se L è rappresentato con un'espressione regolare **E**, prima convertiamo **E** in ε-NFA e poi simuliamo w su questo automa.

Equivalenza e minimizzazione di automi

Equivalenza tra linguaggi regolari

Problema:

Dati due linguaggi regolari L e M (descritti in qualche forma), stabilire se $L = M$

- possiamo ipotizzare che sia L che M siano DFA
- come facciamo a stabilire se i due automi sono equivalenti?

Stati equivalenti

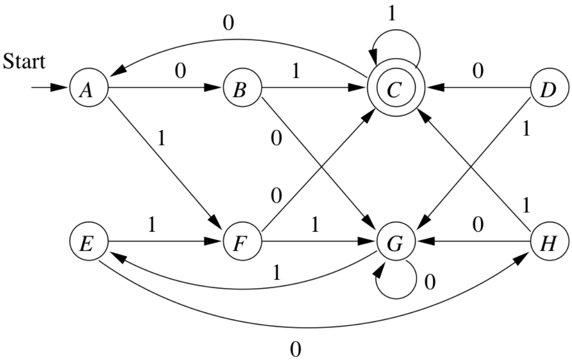
Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA, e siano $p, q \in Q$ due stati. Diciamo che

$$p \equiv q \text{ se e solo se per ogni parola } w \in \Sigma^*,$$
$$\hat{\delta}(p, w) \in F \text{ se e solo se } \hat{\delta}(q, w) \in F$$

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

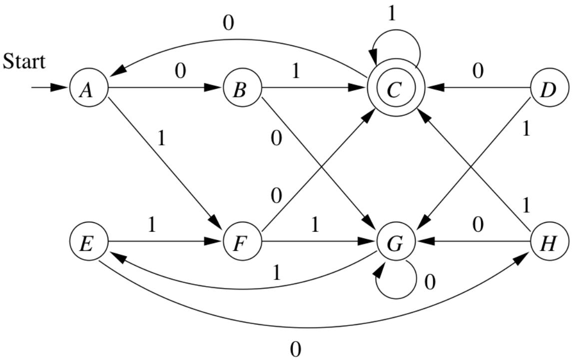
In altre parole: p e q sono distinguibili se e solo se esiste una parola w tale che $\hat{\delta}(p, w) \in F$ e $\hat{\delta}(q, w) \notin F$, o viceversa

Esempio



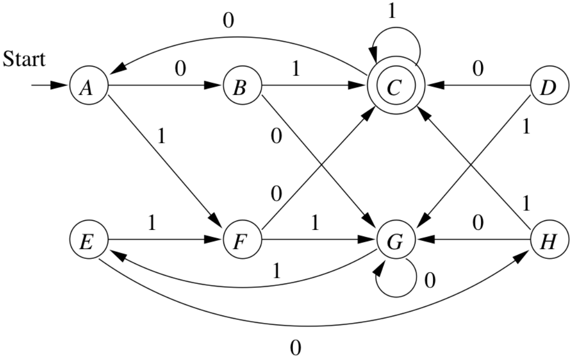
- C e G

Esempio



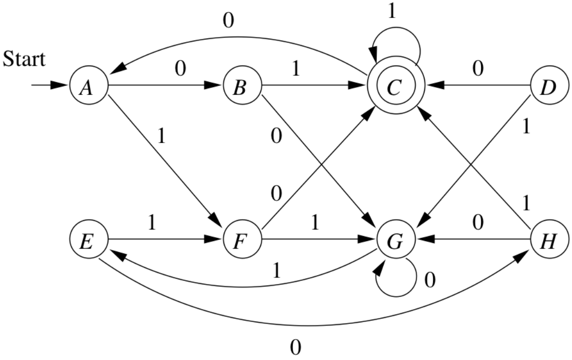
- C e G sono distinguibili: $\hat{\delta}(C, \varepsilon) \in F$, $\hat{\delta}(G, \varepsilon) \notin F$

Esempio

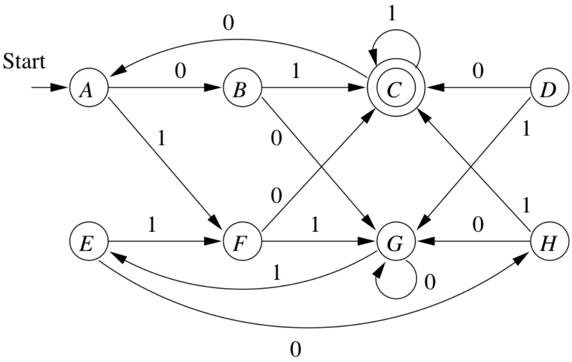


- C e G sono distinguibili: $\hat{\delta}(C, \varepsilon) \in F$, $\hat{\delta}(G, \varepsilon) \notin F$
- A e G

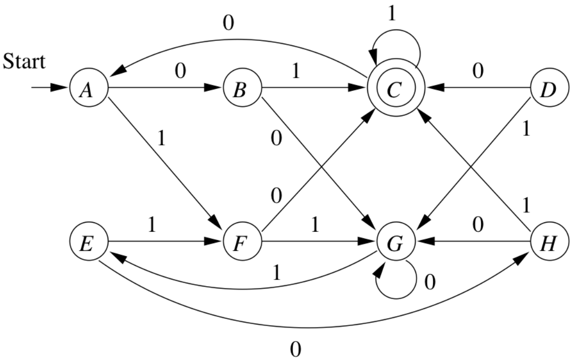
Esempio



- C e G sono distinguibili: $\hat{\delta}(C, \varepsilon) \in F$, $\hat{\delta}(G, \varepsilon) \notin F$
- A e G sono distinguibili: $\hat{\delta}(A, 01) = C \in F$, $\hat{\delta}(G, 01) = E \notin F$

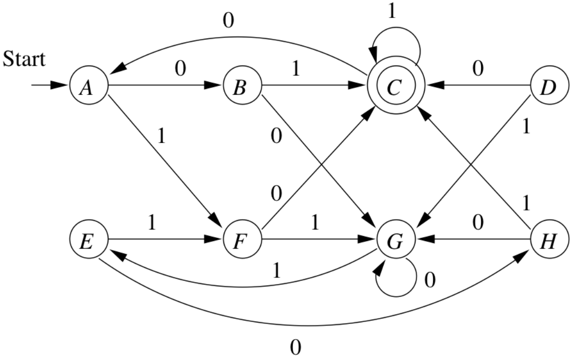


- A ed E sono ...



- A ed E sono ...

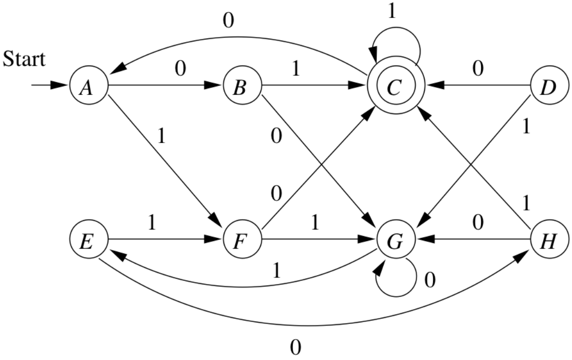
  $\hat{\delta}(A, \varepsilon) = A \notin F$, $\hat{\delta}(E, \varepsilon) = E \notin F$
  $\hat{\delta}(A, 1) = F = \hat{\delta}(E, 1)$
  Quindi $\hat{\delta}(A, 1x) = \hat{\delta}(E, 1x)$



- A ed E sono ...

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

$\hat{\delta}(A, 0) = B \notin F$, $\hat{\delta}(E, 0) = H \notin F$
$\hat{\delta}(A, 00) = G = \hat{\delta}(E, 00)$
$\hat{\delta}(A, 01) = C = \hat{\delta}(E, 01)$



- A ed E sono ... equivalenti!

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

$\hat{\delta}(A, 0) = B \notin F$, $\hat{\delta}(E, 0) = H \notin F$
$\hat{\delta}(A, 00) = G = \hat{\delta}(E, 00)$
$\hat{\delta}(A, 01) = C = \hat{\delta}(E, 01)$

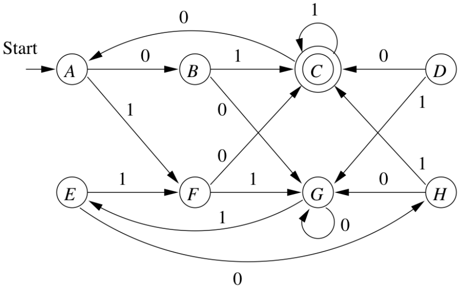
Algoritmo riempi-tabella

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

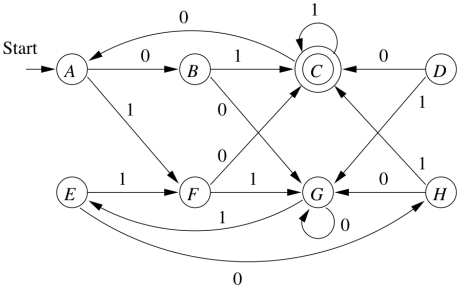
Base: Se $p \in F$ e $q \notin F$, allora $p \not\equiv q$

Induzione: Se trovo un simbolo $a \in \Sigma$ tale che $\delta(p, a) \not\equiv \delta(q, a)$, allora $p \not\equiv q$.

Esempio: Applichiamo l'algoritmo riempi-tabella all'automa precedente:



Esempio: Applichiamo l'algoritmo riempi-tabella all'automa precedente:



	A	B	C	D	E	F	G
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

Correttezza dell'algoritmo

Theorem

Se p e q non sono distinguibili dall'algoritmo, allora $p \equiv q$

Dimostrazione:

Correttezza dell'algoritmo

Università degli Studi di Padova

Theorem

Se p e q non sono distinguibili dall'algoritmo, allora $p \equiv q$

Dimostrazione:

- Supponiamo per assurdo che esista una coppia "sbagliata" p, q:
  1. esiste una parola w tale che $\hat{\delta}(q, w) \in F$ e $\hat{\delta}(p, w) \notin F$, o viceversa
  2. l'algoritmo non distingue tra p e q
- Sia $w = a_1 a_2 \dots a_n$ la parola più corta che identifica una coppia sbagliata p, q
- Ragioniamo per induzione su n.
- Base: $n = 0$ e $w = \varepsilon$. Assurdo perché l'algoritmo avrebbe distingo p e q immediatamente.

Correttezza dell'algoritmo

...

- Induzione: $n \geq 1$. Consideriamo gli stati $r = \delta(p, a_1)$ e $s = \delta(q, a_1)$
- Allora r, s non può essere una coppia "sbagliata" perché sarebbe identificata da una stringa più corta di w.
- Gli stati r ed s sono distinti dalla parola $a_2 a_3 \ldots \_n$: quindi l'algoritmo deve scoperto che sono distinguibili
- Ma allora la parte induttiva dell'algoritmo procede e distingue p da q
- Quindi non ci sono coppie "sbagliate" ed il teorema è dimostrato.

Testare l'equivalenza

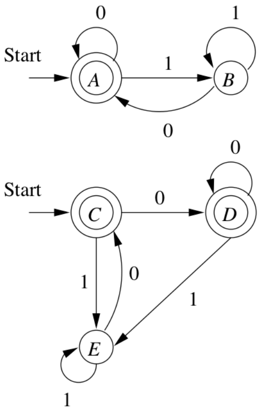
Problema:

Dati due linguaggi regolari L e M (descritti in qualche forma), stabilire se $L = M$

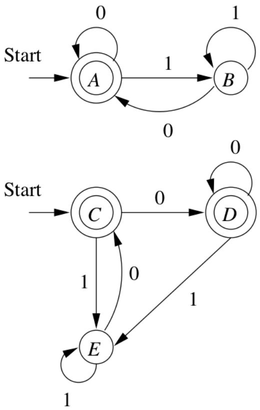
Soluzione:

1. convertiamo sia L che M in DFA.
2. Consideriamo un DFA i cui stati sono l'unione degli stati dei due DFA (non importa se ha due stati iniziali)
3. Se l'algoritmo riempi-tabella dice che i due stati iniziali sono distinguibili, allora $L \neq M$, altrimenti $L = M$.

Esempio



Esempio



Il risultato dell'algoritmo è

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

I due automi sono **equivalenti**

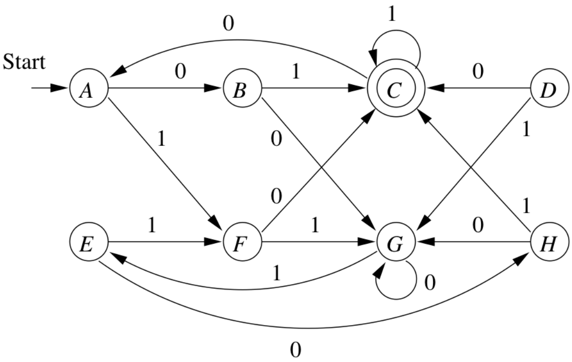
Minimizzazione di un DFA

1. Elimino tutti gli stati non raggiungibili dallo stato iniziale
2. Con l'algoritmo determino le coppie di stati equivalenti
3. Ripartisco gli stati in blocchi in modo che:
   - gli stati in uno stesso blocco siano tutti equivalenti
   - due stati in due blocchi diversi non sono mai equivalenti
4. Il DFA minimo ha i blocchi come stati
5. Sia S un blocco e a un simbolo:
   - allora deve esistere un blocco T tale che, per ogni stato $q \in S$, $\delta(q, a)$ appartiene a T

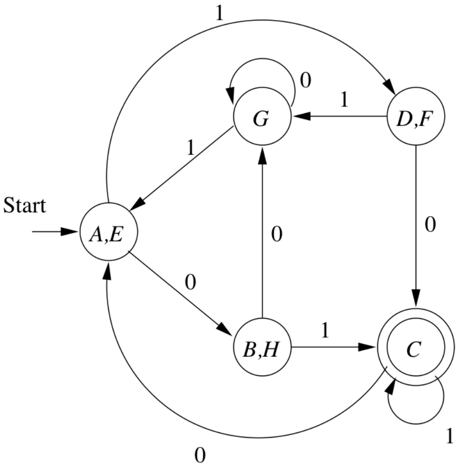
   La funzione di transizione γ del DFA minimo è $\gamma(S, a) = T$
6. I blocchi che contengono stati finali sono stati finali del DFA minimo
7. Il blocco che contiene lo stato iniziale è lo stato iniziale del DFA minimo

Esempio

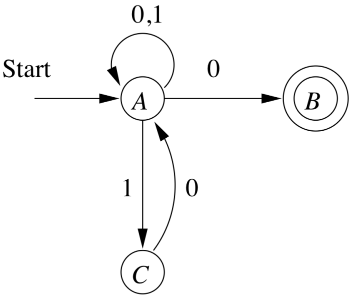
Minimizziamo



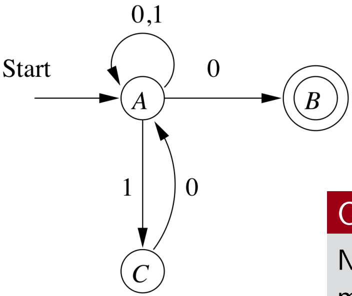
Risultato:



Domanda: cosa succede se proviamo a minimizzare un NFA?



Domanda: cosa succede se proviamo a minimizzare un NFA?



- L'algoritmo ci dice che non ci sono stati equivalenti
- Ma se rimuoviamo lo stato C otteniamo un automa equivalente e più piccolo!

Conclusione:

Non possiamo usare l'algoritmo di minimizzazione con gli NFA!

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 "unione degli stati con" C.
- Dato che $L(B) = L(C)$, abbiamo che gli stati iniziali sono equivalenti: $q_0^B \equiv q_0^C$.
- Inoltre, $\delta(q_0^B, a) \equiv \delta(q_0^C, a)$, per ogni a.

Perché non si può migliorare il DFA minimizzato

- 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 $p = \hat{\delta}(q_0^B, w)$, per una qualche stringa w .
  - Allora $q = \hat{\delta}(q_0^C, w)$, 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.

Esercizi

Create un DFA che riconosca i seguenti linguaggi e minimizzatelo:

- Tutte le stringhe sull'alfabeto $\{a, b, c\}$
- Tutte le stringhe sull'alfabeto $\{0, 1\}$ con un numero pari di 0 e un numero dispari di 1
- Tutte le stringhe sull'alfabeto $\{0, 1\}$ dove 0 e 1 hanno la stessa parità
- Tutte le stringhe sull'alfabeto $\{0, 1\}$ con 1 in terzultima posizione