

030-linguaggi-regolari-26

Automata, Languages and Computing

Linguaggi Regolari
Generati da grammatiche di Chomsky di tipo 3

1

Linguaggi regolari

- sono i linguaggi generati da grammatiche di Chomsky di tipo 3
- vari elementi sintattici di base dei linguaggi di programmazione sono regolari (es. identificatori)
- godono di interessanti proprietà algebriche
- rapporti con espressioni regolari

2

1

030-linguaggi-regolari-26

Lungo viaggio nei linguaggi regolari

automi a stati finiti
relazioni tra automi e linguaggi regolari
“pumping lemma”
chiusura dei linguaggi regolari
espressioni regolari e linguaggi regolari
decidibilità e linguaggi regolari
teorema di Myhill-Nerode

3

Parliamo di automi a stati finiti: il suo obiettivo è riconoscere linguaggi; Pumping Lemma--> si usa per dimostrare che se un linguaggio non è regolare;

La proprietà di chiusura dal punto di vista algebrico vuol dire che: ho 2 linguaggi regolari, se ne calcolo l'unione viene ancora regolare? Se sì, vuol dire che i linguaggi regolari sono chiusi rispetto l'unione;

Un problema è decidibile se esiste un algoritmo che lo risolve

Automi a stati finiti (ASF)

- sono il tipo più semplice di macchina per

riconoscere linguaggi

- dispositivi che leggono la stringa di input da un nastro unidirezionale e la elaborano usando una memoria limitata
- elaborazione un passo alla volta
- ad ogni passo: lettura di un carattere, spostamento della testina in avanti, aggiornamento dello stato

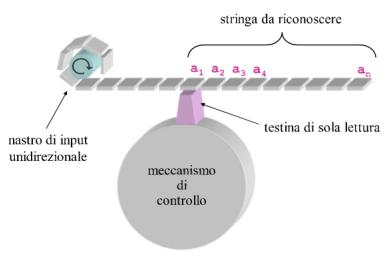
4

2

10/15/2023

030-linguaggi-regolari-26

Schema di automa a stati finiti



5

Meccanismo: una stringa scritta su un nastro deve essere riconosciuta da un automa per vedere se appartiene ad un linguaggio. Viene letto carattere per carattere e il nastro si sposta in una sola direzione

5

Automi a stati finiti (ASF)

un automa a stati finiti è una quintupla

$$A = \langle \Sigma, K, \delta, q_0, F \rangle$$

$$\Sigma = \{\sigma_1, \dots, \sigma_n\}$$

insieme di caratteri

$$K = \{q_0, \dots, q_m\}$$

insieme finito non vuoto di stati

$$F \subseteq K$$

insieme di stati finali

$$q_0 \in K$$

stato iniziale

$$\delta : K \times \Sigma \rightarrow K$$

funzione totale di transizione che determina lo stato successivo

può essere descritta tramite una tabella di transizione o un diagramma di stato

K è uno stato, sigma è un carattere. Da questi l'automa va in un altro stato K.
La funzione di transizione è totale

6

3

10/15/2023

030-linguaggi-regolari-26

Esempio di ASF

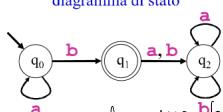
$$A = \langle \Sigma, K, \delta, q_0, F \rangle$$

$$\Sigma = \{a, b\} \quad K = \{q_0, q_1, q_2\} \quad F = \{q_1\}$$

tabella di transizione

δ	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	\sim	\sim

diagramma di stato



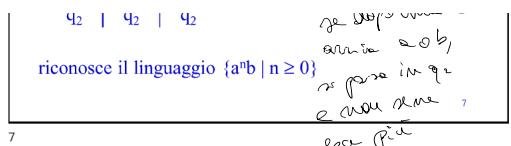
CONVENZIONALMENTE

Se l'automa, dopo aver letto tutta la stringa si ferma in uno stato finale (F), allora ha riconosciuto la stringa.

Altamente NO.

"Se stai nello stato q_0 e leggi a vai nello stato q_1 ; se sei nello stato q_0 e leggi b vai in q_2 ... ecc."

Lo stato finale è q_1 . se F fosse vuoto, l'automa non riconoscerebbe alcuna stringa.



È un linguaggio di tipo 3. per dimostrarlo devo usare una grammatica che genera il linguaggio.
 $S \rightarrow aS|b \rightarrow$ grammatica regolare ---> linguaggio regolare.

Linguaggio riconosciuto da un ASF

estendiamo la funzione di transizione alle stringhe:

$$\delta : K \times \Sigma^* \rightarrow K$$

$$\begin{cases} \delta(q, \varepsilon) = q \\ \delta(q, xa) = \delta(\delta(q, x), a) \end{cases} \quad \text{con } x \in \Sigma^* \text{ ed } a \in \Sigma$$

linguaggio riconosciuto da un automa A:

$$L(A) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}$$

Delta segnato è la funzione di transizione estesa alle stringhe. Non gli diamo uno stato e un carattere, ma uno stato, una stringa e dopo la freccia--> ci dice dove si trova l'automa. MI DICE LO STATO IN CUI MI TROVO, LETTI n CARATTERI

- Se sono in uno stato e non leggo rimango nello stato stesso (caso base):
- Se leggo una sequenza di caratteri (stringa) : vai nello stato della lettura di x e fai come ultimo passaggio a minuscolo

$L(A) \rightarrow$ IL LINGUAGGIO RICONOSCIUTO è l'insieme delle stringhe x appartenenti a sigma asteriscato tale che se parto da q_0 e le leggo tutte con l'automa la mia computazione finisce in uno stato di F (stato finale) attraverso la funzione di transizione.

$L(G) \rightarrow$ ho una definizione analoga ma con le produzioni e parto dall'assioma.

8

4

Domande:

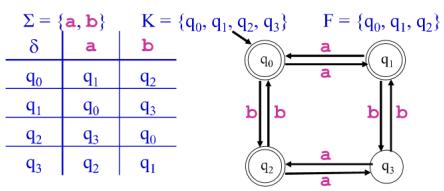
- Se al posto di q_0 inizio da q_5 ? Sono le stringhe che farebbero parte del linguaggio se q_5 appartenesse allo stato iniziale.
- Se al posto del simbolo "appartiene", avessimo "non appartiene"--> il linguaggio riconosciuto sarebbe il complemento.

10/15/2023

030-linguaggi-regolari-26

Esempio

automa che riconosce il linguaggio delle parole che contengono un numero pari (anche zero) di a o un numero pari (anche zero) di b



Se la computazione si ferma in $q_0 \rightarrow$ abbiamo un numero pari di a e pari di b;
In $q_1 \rightarrow$ pari b dispari a

"ab" non è accettata come stringa.

Se levo q_0 dagli stati finali che succede? --> Se ho un numero pari di a e di b la stringa non è riconosciuta

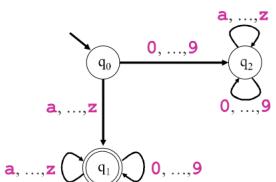
Il complemento del linguaggio invece? Le stringhe che non appartengono al linguaggio sono: tutte quelle che non hanno un numero pari di b e nemmeno di a, ovvero a dispari b dispari.
Se finisco in q_3 ho l'automa complemento.

9

9

Esempio

automa che riconosce gli identificatori (stringhe alfano numeriche che cominciano con una lettera)



Stringa che inizia per forza con una lettera altrimenti
Non è un identificatore.

Può invece terminare con un numero o una lettera.

10

5

10/15/2023

**Automi a stati finiti
non deterministici (ASFND)**

un **automa a stati finiti non deterministico** è una quintupla

$$A = \langle \Sigma, K, \delta_N, q_0, F \rangle$$

$\Sigma = \{\sigma_1, \dots, \sigma_n\}$ alfabeto di input
 $K = \{q_0, \dots, q_m\}$ insieme finito non vuoto di stati
 $F \subseteq K$ insieme di stati finali
 $q_0 \in K$ stato iniziale
 $\delta_N : K \times \Sigma \rightarrow P(K)$ funzione totale di transizione che determina l'insieme (eventualmente vuoto) degli stati successivi

11

Non deterministico == l'automa fa **più cose contemporaneamente**

Con la **funzione di transizione** se sono in uno stato e leggo sigma arrivo in $P(K) \rightarrow$ l'insieme delle parti di K .

La funzione di transizione non determinista solo uno stato successivo ma un insieme di stati successivi, anche vuoto.

La fdt non porta ad un insieme degli stati ma $P(K)$, l'insieme delle parti dell'insieme K

Linguaggio riconosciuto da un ASFND

estendiamo la funzione di transizione alle stringhe:

$$\delta_N : K \times \Sigma^* \rightarrow P(K)$$

$$\begin{cases} \delta_N(q, \varepsilon) = \{q\} \\ \delta_N(q, xa) = \bigcup_{p \in \delta_N(q, x)} \delta_N(p, a) \text{ con } x \in \Sigma^*, a \in \Sigma, p \in K \end{cases}$$

linguaggio riconosciuto da un automa A_N :

$$L(A_N) = \{x \in \Sigma^* \mid \delta_N(q_0, x) \cap F \neq \emptyset\}$$

↓
per ogni stringa la stringa

12

6

La parte induttiva: parto da q , leggo x e mi trovo in un insieme di stati. Per ciascuno degli stati, vedo **dove mi porta p quando leggo a**. tutti gli stati dove arrivo ne devo fare l'unione per metterli insieme.

$L(A_N) \rightarrow$ la stringa appartiene se almeno uno degli stati è finale.

10/15/2023

030-linguaggi-regolari-26

Esempio

ASFND che riconosce le stringhe che terminano con **bb o ba o baa**

Un automa non deterministico accetta una stringa quando si ferma se tra gli insiemi degli stati ci sta almeno uno finale.

13

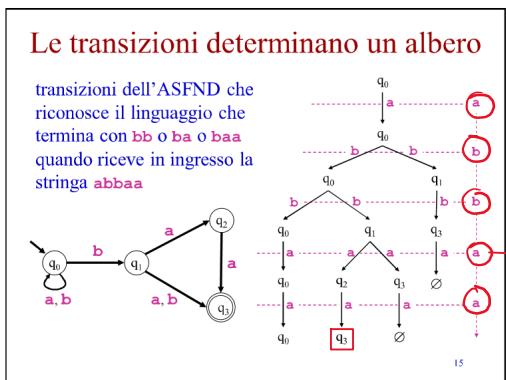
Un automa non deterministico accetta una stringa quando si ferma se tra gli insiemi degli stati ci sta almeno uno finale.

Osservazioni sugli ASFND

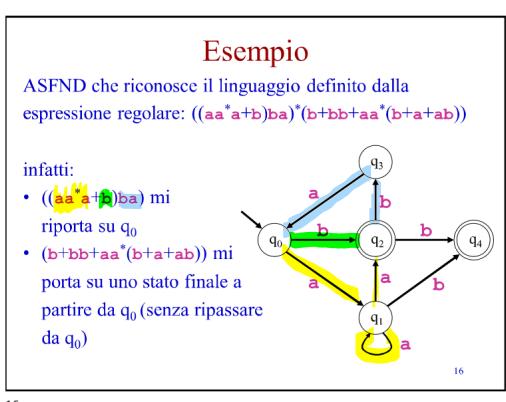
- attenzione a non confondere il **non determinismo** degli automi con altre accezioni del termine (per esempio aspetti probabilistici 🎲)
- il non determinismo degli automi consente di rappresentare una computazione come un **albero** nello spazio degli stati



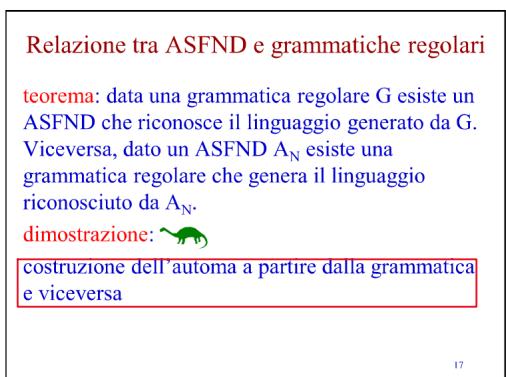
030-linguaggi-regolari-26



Oss: La computazione di un automa deterministico è un cammino

Se ho letto solo **abba** la fdt mi deve dire che sto in **q0,q2,q3**

030-linguaggi-regolari-26



Grammatiche regolari e automi non deterministici, sono la stessa cosa \rightarrow linguaggi di tipo 3: **grammatiche** li generano, gli automi li riconoscono

un linguaggio è regolare se e solo se esiste un automa a stati finiti che lo riconosce

Grammatiche regolari e produzioni elementari

- una **produzione elementare** ha la forma $A \rightarrow a$
– dove $A \in V_N$ e $a \in V_T$

• possiamo limitarci a dimostrare il teorema per le grammatiche regolari che **non hanno produzioni elementari**

– hanno solo produzioni della forma $A \rightarrow aA$ e $A \rightarrow \epsilon$

- infatti una produzione $A \rightarrow a$ può sempre sostituita con $A \rightarrow aF$ e $F \rightarrow \epsilon$

– dove F è una nuova categoria sintattica introdotta appositamente

18

Se ho una produzione $A \rightarrow a$, la modifco come sta scritto sotto e funziona ugualmente. Così escludiamo le produzioni **che vanno da un non terminale ad un terminale secco**

18

9

10/15/2023

030-linguaggi-regolari-26

Dimostrazione (grammatica \rightarrow ASFND)

data $G = \langle V_T, V_N, P, S \rangle$ priva di produzioni elementari costruiamo $A_N = \langle \Sigma, K, \delta_N, q_0, F \rangle$ come segue:

$$\begin{aligned} \Sigma &= V_T \\ K &= \{q_i | i \in V_N\} \xrightarrow{\text{per ogni non terminale gli faccio uno stato}} \text{corrispondenza tra stato} \\ q_0 &= q_S \xrightarrow{\text{corrispondenza tra stato iniziale e assioma}} \\ F &= \{q_B | B \rightarrow \epsilon \in P\} \\ \delta_N(q_B, a) &= \{q_C | B \rightarrow aC \in P\} \end{aligned}$$

in generale l'automa A_N **non è deterministico**

19

$$\begin{aligned} &\left(\alpha \left(\beta \beta + \epsilon \right) \right)^* \alpha \\ &\alpha \left(\left(\beta \beta + \epsilon \right) \alpha \right)^* \\ &\left(\alpha \beta \right)^* \alpha = \alpha \left(\alpha \beta \right)^* \end{aligned}$$

Dimostrazione (grammatica \rightarrow ASFND)

dimostriamo che

1. esiste una derivazione $S \Rightarrow \epsilon$ di G se e solo se q_S è finale

2. esiste una derivazione $S \Rightarrow xZ$ di G se e solo se $q_Z \in \delta_N(q_S, x)$ e se q_Z è finale allora $S \Rightarrow x$

la 1 è banale (per costruzione)

la 2 è dimostrabile per induzione:

- passo base: $|x|=1$
- passo induttivo: $x=y\alpha$ con $|y|=n \geq 1$

passo base: $|x|=1$

per costruzione $S \Rightarrow aZ$ se e solo se $q_Z \in \delta_N(q_S, a)$

x è una sequenza di terminali.
 Z è un non terminale. Il non terminale è sempre alla fine.

$S \Rightarrow xZ$: le derivazioni/stringhe sono fatte tutte così.

Passo base -->Stringa composta da 1 carattere

Passo induttivo--> una stringa con più caratteri

20

10

10/15/2023

030-linguaggi-regolari-26

Dimostrazione (grammatica \rightarrow ASFND)

passo induttivo: $x=y\alpha$ con $|y|=n \geq 1$ y è più piccolo di x .

HP induttiva: esiste una derivazione $S \Rightarrow yW$ di G
se e solo se $q_W \in \delta_N(q_S, y)$
e se q_W è finale allora $S \Rightarrow y$

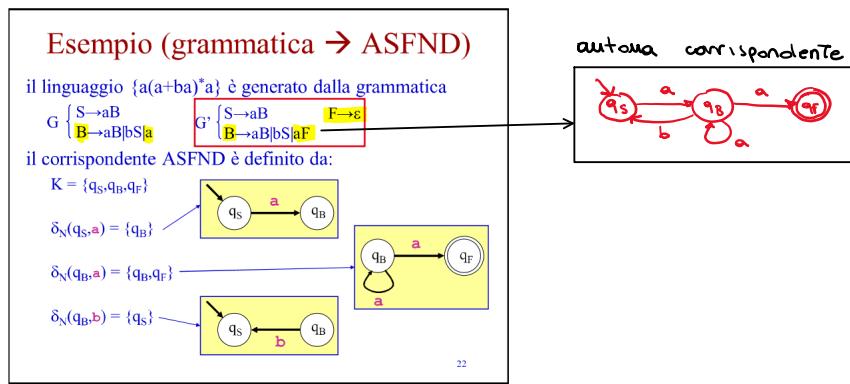
TH induttiva: esiste una derivazione $S \Rightarrow xZ$ di G
 se e solo se $q_Z \in \delta_N(q_S, x)$
 e se q_Z è finale allora $S \Rightarrow x$

dimostrazione: una derivazione $S \Rightarrow yW \Rightarrow yAZ$ esiste se e solo se
 $q_W \in \delta_N(q_S, y)$ e $q_Z \in \delta_N(q_W, A)$ e ciò è vero se e solo se
 $q_Z \in \bigcup_{p \in \delta_N(q_S, y)} \delta_N(p, A)$ cioè se $q_Z \in \delta_N(q_S, x)$

Si può mostrare che esiste una derivazione $S \Rightarrow x$ di G
 se e solo se $\delta_N(q_S, x) \cap F \neq \emptyset$

21

21

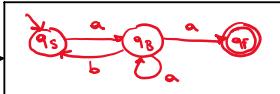


22

22

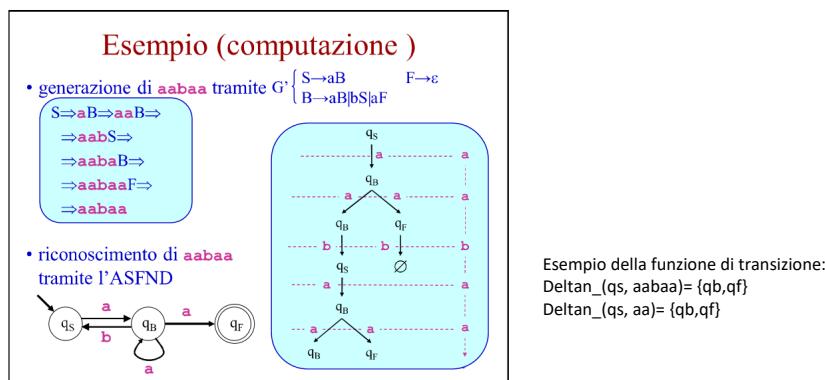
11

automa corrispondente:



10/15/2023

030-linguaggi-regolari-26



23

23



24

24

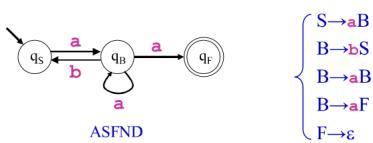
12

030-linguaggi-regolari-26

Esempio (ASFND → grammatica)

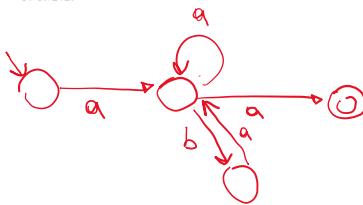
dato l'ASFND che riconosce il linguaggio
 $\{a(a+ba)^*a\}$ trovare la grammatica corrispondente

$$\begin{array}{l} S \rightarrow aB \\ B \rightarrow bS | aB | \epsilon \\ F \rightarrow \epsilon \end{array}$$

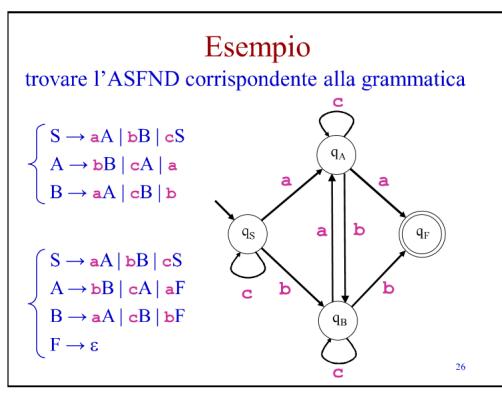


$$\left\{ \begin{array}{l} S \rightarrow aB \\ B \rightarrow bS \\ B \rightarrow aB \\ B \rightarrow aF \\ F \rightarrow \epsilon \end{array} \right.$$

25



25



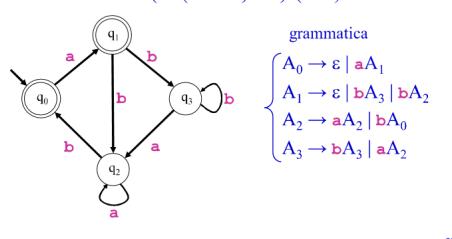
26

13

030-linguaggi-regolari-26

Esempio

ASFND che riconosce
 $(ab(b^*a+\epsilon)a^*b)^*(a+\epsilon)$



grammatica

$$\left\{ \begin{array}{l} A_0 \rightarrow \epsilon | aA_1 \\ A_1 \rightarrow \epsilon | bA_3 | bA_2 \\ A_2 \rightarrow aA_2 | bA_0 \\ A_3 \rightarrow bA_3 | aA_2 \end{array} \right.$$

27

27

Relazione tra ASF e ASFND

sono computazionalmente più potenti gli ASF o gli ASFND?
teorema: dato un ASF che riconosce il linguaggio L, esiste un ASFND che riconosce L. Viceversa, dato un ASFND che riconosce il linguaggio L',

Asfnd --> la computazione da un albero.

La parte deterministica, è quella che descrive l'insieme di stati come un unico stato.

Costruisco un automa deterministico per il quale esistano degli stati che sono equivalenti a quelli del non deterministico (mettendo insieme più stati in un unico

esiste un ASF che riconosce L' .

dimostrazione:

la **simulazione** di un ASF con un ASFND è banale, i due automi sostanzialmente coincidono
la simulazione di un ASFND con un ASF sfrutta la finitezza di $P(K)$

stato es: q_1, q_2, q_3 sono uno stato nel deterministico.)

K è finito e allora anche l'insieme delle parti è finito.

28

14

10/15/2023

030-linguaggi-regolari-26

Dimostrazione ($\text{ASFND} \rightarrow \text{ASF}$)

dato $A_N = <\Sigma, K, \delta_N, q_0, F>$

costruiamo $A' = <\Sigma', K', \delta', q'_0, F'>$

come segue:

$$\Sigma' = \Sigma$$

K' è in corrispondenza biunivoca con 2^K

i nomi degli stati di K' sono i sottoinsiemi in $[]$

$$K' = \{[q_{i_1}, \dots, q_{i_k}] \mid \{q_{i_1}, \dots, q_{i_k}\} \in 2^K\}$$

$$q'_0 = [q_0]$$

$$F' = \{[q_{i_1}, \dots, q_{i_k}] \mid \{q_{i_1}, \dots, q_{i_k}\} \in 2^K \wedge \{q_{i_1}, \dots, q_{i_k}\} \cap F \neq \emptyset\}$$

Dato un automa non deterministico, costruiamo il deterministico:

Gli alfabeti coincidono, lavoriamo sullo stesso alfabeto.

K' : gli stati di k' sono in corrispondenza biunivoca con 2^k . I nomi sono i sottoinsiemi specificati tra parentesi quadre. Questo è il nome che noi diamo al nuovo stato deterministico. Questo meccanismo è descritto come:

Tra gli insiemi degli stati che appartengono a 2^k faccio corrispondere tra parentesi quadre l'insieme di stati del deterministico.

Lo stato iniziale è lo stesso.

l'insieme degli stati finali --> sono gli stati tali che l'insieme contiene al suo interno **almeno un elemento di F (del non deterministico, che è punto di partenza della costruzione).**

29

29

Dimostrazione ($\text{ASFND} \rightarrow \text{ASF}$)

infine definiamo $\delta'([q_{i_1}, \dots, q_{i_k}], a) = [q_{j_1}, \dots, q_{j_m}]$
dove $\{q_{j_1}, \dots, q_{j_m}\} = \delta_N(q_{i_1}, a) \cup \dots \cup \delta_N(q_{i_k}, a)$

Bisogna costruire la fdt del deterministico: l'automa deterministico deve andare in uno stato che li rappresenti tutti, per questo si fa l'unione.

dobbiamo ora mostrare che i due automi A_N ed A' si comportano allo stesso modo

resta cioè da mostrare che

$$\delta'([q_0], x) = q_i \text{ con } q_i \in F' \text{ e } q_i = [q_{j_1}, \dots, q_{j_m}]$$

se e solo se

$$\delta_N(q_0, x) = \{q_{j_1}, \dots, q_{j_m}\} \text{ con } \{q_{j_1}, \dots, q_{j_m}\} \cap F \neq \emptyset$$

ciò è vero per costruzione

Si comportano allo stesso modo == riconoscono lo stesso linguaggio.

30

30

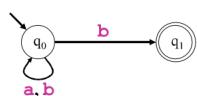
15

10/15/2023

030-linguaggi-regolari-26

Esempio

costruire l'ASF corrispondente all'ASFND seguente:

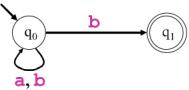


che accetta le stringhe di $\{a,b\}^*$ che terminano con b

una strategia per la costruzione della funzione di transizione dell'ASF è quella di visitare gli stati a partire da $[q_0]$ introducendo nuovi stati (e relative transizioni) non

Stati dell'automa deterministico corrispondente:
[], [q0], [q1], [q0 q1]

Esempio (continua)



δ'	a	b
$[q_0]$	$[q_0]$	$[q_0, q_1]$

$$\delta'([q_0], \textcolor{red}{a}) = [\delta_N(q_0, \textcolor{red}{a})] = [q_0]$$

$$\delta'([q_0], \textcolor{red}{b}) = [\delta_N(q_0), \textcolor{red}{b}] = [q_0, q_1]$$

nuovo stato
da introdurre

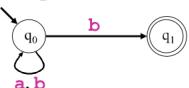
Insieme di stati dove arrivo nel non deterministico. Che mi determinano gli stati nuovi del deterministico

32

16

030-linguaggi-regolari-26

Esempio (continua)



δ'	a	b
$[q_0]$	$[q_0]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0]$	$[q_0, q_1]$

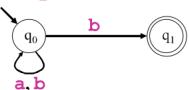
$\delta'([q_0, q_1], a) = [\delta_N(q_0, a)] \cup [\delta_N(q_1, a)] = [q_0, \emptyset] = [q_0]$

[q0, q1] è un
nuovo stato e devo
vedere che
succede quando
leggo a e b

33

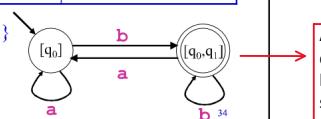
33

Esempio (continua)



δ'	a	b
$[q_0]$	$[q_0]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0]$	$[q_0, q_1]$

quindi $K' = \{[q_0], [q_0, q_1]\}$
 dato che $F = \{q_1\}$
 abbiamo $F' = \{[q_0, q_1]\}$



Automa deterministico corrispondente. Potenzialmente l'insieme [] e [q1] esistono ma non sono raggiungibili quindi non li prendo in esame in questo caso.

34

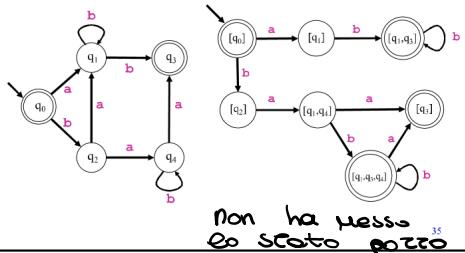
17

10/15/2023

030-linguaggi-regolari-26

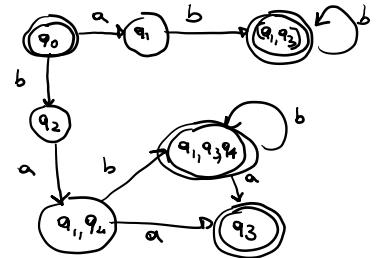
Esempio

costruzione di un ASF con 8 stati (compreso lo stato pozzo) a partire da un ASFND con 5 stati



35

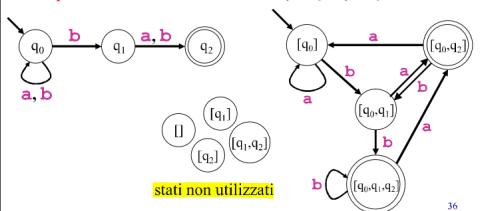
	a	b	
q_0	$[q_2] \checkmark$	$[q_2] \checkmark$	✓
q_1	$[]$	$[q_4, q_5] \checkmark$	✓
q_2	$[q_1, q_2] \checkmark$	$[]$	✓
q_3	$[]$	$[q_1, q_3] \checkmark$	✓
q_4	$[q_1, q_4] \checkmark$	$[q_3] \checkmark$	✓
q_5	$[q_1, q_3, q_4] \checkmark$	$[q_1, q_3, q_4]$	
q_6	$[q_5]$	$[q_1, q_3, q_4]$	
q_7	$[]$	$[]$	



Osservazione

con il metodo di costruzione introdotto, l'ASF può (ma non è detto) avere un numero esponenziale di stati rispetto all'ASFND di partenza

esempio: automi che riconoscono $(a+b)^*b(a+b)$



36

18

Sia A un automa a stati finiti non deterministico con 5 stati. Quali delle seguenti affermazioni su A sono vere.

- a. $L(A)$ è un linguaggio context free
- b. A ha necessariamente 5 stati finali
- c. Esiste un automa a stati finiti deterministico che riconosce $L(A)$ e che ha meno di 70 stati ✓
- d. $L(A)$ è un linguaggio regolare
- e. Esiste un automa a stati finiti deterministico che riconosce $L(A)$ ✓
- f. Esattamente uno degli stati di A è chiamato stato iniziale; quando la computazione inizia, A si trova in quello stato ✓

Risposta parzialmente esatta.

Hai selezionato correttamente 3.

Le risposte corrette sono:

Esiste un automa a stati finiti deterministico che riconosce $L(A)$,

Esiste un automa a stati finiti deterministico che riconosce $L(A)$ e che ha meno di 70 stati,

Esattamente uno degli stati di A è chiamato stato iniziale; quando la computazione inizia, A si trova in quello stato,

$L(A)$ è un linguaggio regolare,

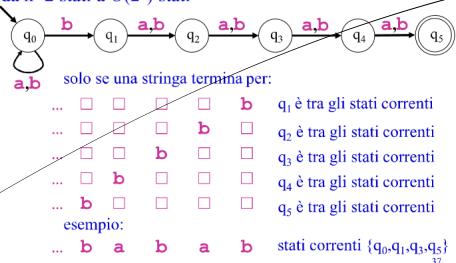
$L(A)$ è un linguaggio context free

?

030-linguaggi-regolari-26

Osservazione

generalizzando a stringhe del tipo $(a+b)^*b(a+b)^k$, si passa da $k+2$ stati a $O(2^k)$ stati



37

10/15/2023