

Trasparenze del corso di

# Informatica Teorica

Parte Seconda

## Linguaggi e Grammatiche

1

corrispondenza tra  
linguaggi e problemi

## Linguaggi e informatica

- ubiquitari nelle applicazioni
  - compilatori ed interpreti
  - protocolli
  - sequenze di operazioni
  - interfacce
- paradigmatici nella teoria
  - molti importanti problemi teorici sono riconducibili a quello dell'appartenenza di una stringa ad un linguaggio

2

## Tre approcci diversi tra loro complementari

non molto utile  
all'atto pratico

- approccio insiemistico come un frasario
  - utile per determinare le proprietà elementari dei linguaggi
- approccio generativo
  - grammatiche formali regole per generare le stringhe
- approccio riconoscitivo
  - automi riconoscitori meccanismi che riconoscono le stringhe di un linguaggio

3

## Alfabeto

- un *alfabeto* è un insieme finito non vuoto di simboli (caratteri)
- esempi:
  - $\{\text{'M'}, \text{'C'}, \text{'L'}, \text{'X'}, \text{'I'}, \text{'V'}\}$
  - $\{\text{'0'}, \text{'1'}\}$
  - $\{\text{'0'}, \text{'1'}, \text{'2'}, \text{'3'}, \text{'4'}, \text{'5'}, \text{'6'}, \text{'7'}, \text{'8'}, \text{'9'}\}$
  - $\{\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}, \text{'e'}, \text{'f'}, \text{'g'}, \text{'h'}, \text{'i'}, \text{'l'}, \text{'m'}, \text{'n'}, \text{'o'}, \text{'p'}, \text{'q'}, \text{'r'}, \text{'s'}, \text{'t'}, \text{'u'}, \text{'v'}, \text{'z'}\}$

4

## Concatenazione

- dato un alfabeto  $\Sigma$ , definiamo l'operazione binaria *concatenazione* (denotata con “ $\circ$ ”)  

$$\mathbf{a} \circ \mathbf{b} = \mathbf{ab}$$
 con  $\mathbf{a}, \mathbf{b} \in \Sigma$
- indichiamo con  $\mathbf{a}^n$  la concatenazione di  $\mathbf{a}$  con se stessa  $n$  volte  
 esempio:  $\mathbf{a}^4 = \mathbf{a} \circ \mathbf{a} \circ \mathbf{a} \circ \mathbf{a} = \mathbf{aaaa}$
- l'operazione “ $\circ$ ” è associativa ma non commutativa
- dati  $\Sigma$  e “ $\circ$ ” definiamo  $\Sigma^+$  come l'insieme delle stringhe (parole) di lunghezza finita
- se a  $\Sigma^+$  aggiungiamo la stringa vuota  $\varepsilon = \text{“”}$  otteniamo  $\Sigma^*$

5

## Linguaggio

- un *linguaggio* è un sottoinsieme di  $\Sigma^*$
- $\Sigma^*$  è detto linguaggio universale
- il linguaggio vuoto  $\Lambda$  non contiene stringhe (nota che  $\Lambda$  coincide con l'insieme vuoto  $\emptyset$ )
- $\Lambda \neq \{\varepsilon\}$

6

## Operazioni sui linguaggi

$L_1$  e  $L_2$  linguaggi

- *unione*

$$L_1 \cup L_2 = \{x \in \Sigma^* \mid x \in L_1 \vee x \in L_2\}$$

tutte le stringhe di  $L_1$   
+ le stringhe di  $L_2$

$$L_1 \cup \Lambda = L_1$$

- *intersezione*

$$L_1 \cap L_2 = \{x \in \Sigma^* \mid x \in L_1 \wedge x \in L_2\}$$

tutte le stringhe che  
appartengono a entrambi  
i linguaggi  $L_1$  ed  $L_2$

$$L_1 \cap \Lambda = \Lambda$$

- *complementazione*

$$\underline{L}_1 = \{x \in \Sigma^* \mid x \notin L_1\}$$

tutte le stringhe che  
non appartengono a  $L_1$

7

## Operazioni sui linguaggi

$L_1$  e  $L_2$  linguaggi

- *concatenazione o prodotto*

$$L_1 \circ L_2 = \{x \in \Sigma^* \mid$$

$$\exists x_1 \in L_1 \wedge \exists x_2 \in L_2 \text{ tali che } x = x_1 \circ x_2\}$$

$$L_1 = \{a, b\}, L_2 = \{b, c\}$$

$$L_1 \circ L_2 = \{ac, ad, bc, bd, a, b\}$$

$$L \circ \{\epsilon\} = \{\epsilon\} \circ L = L$$

**esempio:**

$$L_1 = \{a^n \mid n \geq 1\}; L_2 = \{b^m \mid m \geq 1\}; L_1 \circ L_2 = \{a^n b^m \mid n, m \geq 1\}$$

- *potenza*  $L^h$  di un linguaggio  $L$

$$L^0 = \{\epsilon\}$$

$$L^h = L \circ L^{h-1}, \text{ per } h \geq 1$$

$$L = \{a, b\}, L^1 = L^0 L^1 = L$$

$$= L^0 L^0 = \{a, b\}$$

8

## Operatore di Kleene

- chiusura riflessiva e transitiva* di un linguaggio

$$L^* = \bigcup_{h=0}^{\infty} L^h$$

$$\varepsilon \in L^*$$

$$\Lambda^* = \{\varepsilon\}$$

**esempio:**  $L = \{aa\}$        $L^* = \{a^{2n} | n \geq 0\}$

- chiusura transitiva* (non riflessiva) di un linguaggio

$$L^+ = \bigcup_{h=1}^{\infty} L^h$$

**esempio:**  $L = \{aa\}$        $L^+ = \{a^{2n} | n \geq 1\}$

$$L^* = L^+ \cup \{\varepsilon\}$$

9

## Espressioni regolari

- è uno strumento per descrivere linguaggi (vedremo poi quali)
- dato un alfabeto  $\Sigma$ , si definisce *espressione regolare* ogni stringa  $r$

$$r \in (\Sigma \cup \{+, *, (, ), \circ, \emptyset\})^+ \quad \text{chiusura transitiva}$$

- tale che:

- $r = \emptyset$  oppure
- $r \in \Sigma$  oppure
- $r = (s+t)$  oppure  $r = (s \circ t)$  oppure  $r = s^*$ , con  $s$  e  $t$  espressioni regolari

semantica

espressione	linguaggio
$\emptyset$	$\Lambda$
$a \in \Sigma$	$\{a\}$
$(s+t)$	$L(s) \cup L(t)$
$(s \circ t)$	$L(s) \circ L(t)$
$s^*$	$L(s)^*$

un'espressione regolare può essere l'unione, la concatenazione o la chiusura transitiva e riflessiva di due espressioni regolari

10

$(ab+cd)^* = \{\varepsilon, ab, cd, abcd, abab, cdcd, cdab, \dots\}$   
in pratica tutte le combinazioni possibili

## Espressioni regolari

esempio:

$(a+b)^*$  rappresenta  $L = (\{a\} \cup \{b\})^*$

esempio:

$(a+b)^*a$  rappresenta  $L = \{x \mid x \in \{a,b\}^* \wedge \text{"x termina con a"}\}$

forma sintetica

$st$  è forma sintetica di  $s \circ t$

forma sintetica

espressioni sintetiche si ottengono definendo delle  
precedenze tra gli operatori:  $* > \circ > +$

esempio:

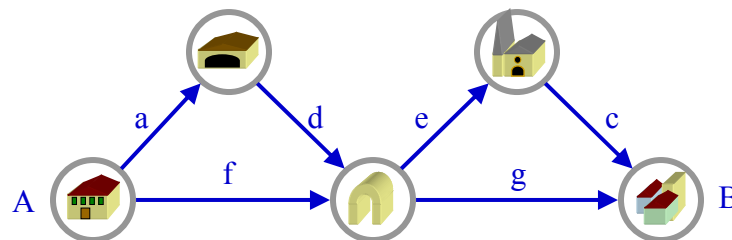
$(a+(b(cd))) = a+bcd$

i linguaggi rappresentabili con espressioni regolari sono una  
interessante sottoclasse

non si possono quindi  
rappresentare tutti i linguaggi

## Esercizio

data una mappa stradale, scrivere un'espressione regolare che  
definisca tutti i percorsi possibili tra A e B

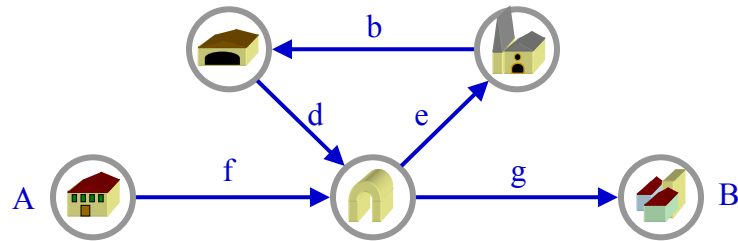


$$\begin{aligned}
 & adec + adg + fec + fg = \\
 & = ad(ec+g) + f(ec+g) = \\
 & = (ad+f)(ec+g)
 \end{aligned}$$

12

## Esercizio

data una mappa stradale, scrivere un'espressione regolare che definisca tutti i percorsi possibili tra A e B



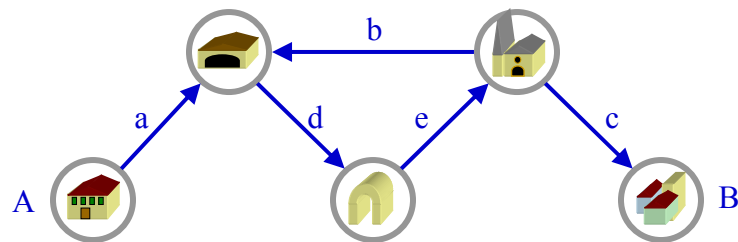
$$f(ebd)^*g$$

ebd lo faccio 0 o più volte

13

## Esercizio

data una mappa stradale, scrivere un'espressione regolare che definisca tutti i percorsi possibili tra A e B

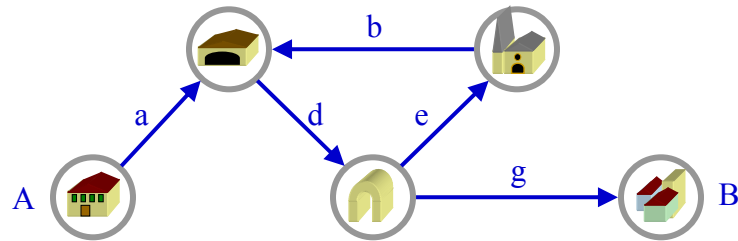


$$a(deb)^*dec = ad(ebd)^*ec$$

14

## Esercizio

data una mappa stradale, scrivere un'espressione regolare che definisca tutti i percorsi possibili tra A e B

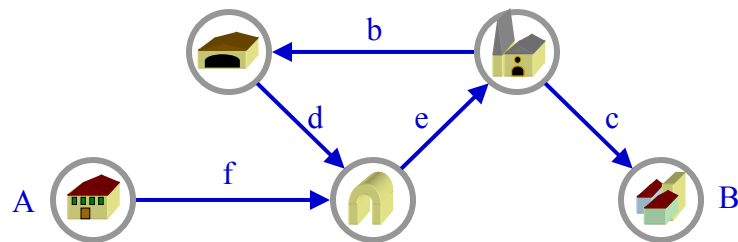


$$a(\text{deb})^* dg = ad(\text{deb})^* g$$

15

## Esercizio

data una mappa stradale, scrivere un'espressione regolare che definisca tutti i percorsi possibili tra A e B



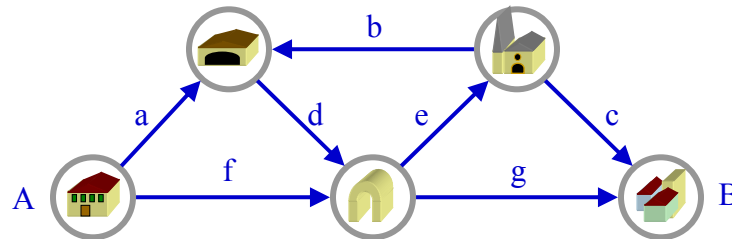
$$f(\text{edb})^* ec$$

16



## Esercizio

data una mappa stradale, scrivere un'espressione regolare che definisca tutti i percorsi possibili tra A e B



$$\begin{aligned}
 & f(ebd)^*g + f(ebd)^*ec + a(deb)^*dec + a(deb)^*dg = \\
 & = f(ebd)^*(g+ec) + a(deb)^*d(g+ec) = \\
 & = (f(ebd)^* + a(deb)^*d)(g+ec) = \\
 & = (f+ad)(ebd)^*(g+ec)
 \end{aligned}$$

17

## Cardinalità dei linguaggi

- dato un alfabeto  $\Sigma$  è possibile definire un ordinamento “lessicografico” per le stringhe di  $\Sigma^*$
- ciò implica l'esistenza di un algoritmo di enumerazione per  $\Sigma^*$
- quindi  $|\Sigma^*| = \aleph_0$
- un linguaggio definito su  $\Sigma$  è un sottoinsieme di  $\Sigma^*$
- l'insieme di tutti i linguaggi definiti su  $\Sigma$  è equinumeroso a  $P(\Sigma^*)$  e quindi ha cardinalità  $2^{\aleph_0}$
- ne segue che l'insieme di tutti i linguaggi definiti su  $\Sigma$  non è numerabile

18

## Cardinalità dei programmi

l'insieme dei programmi scritti in Java (per esempio) ha cardinalità  $\aleph_0$ , infatti posso facilmente enumerarli

**problema:**

dato un qualunque  $L \subseteq \Sigma^*$  esiste sempre un programma scritto in Java che, data una stringa  $x$  di  $\Sigma^*$ , decida se  $x$  appartiene ad  $L$ ?

**soluzione:**

semplici considerazioni sulla cardinalità evidenziano una risposta negativa

ci sono più problemi che soluzioni  
ci sono più problemi che programmi che li sappiano risolvere

19

## Le grammatiche formali

- approccio generativo alla descrizione dei linguaggi
- metodo di costruzione delle stringhe basato sulla riscrittura

1940 Post e Markof, riscrittura e derivazione di stringhe

1950 Chomsky, classificazione delle grammatiche nell'ambito degli studi sul linguaggio naturale

1960 Backus Naur Form per descrivere Algol

20

## Grammatiche formali

- grammatiche di Chomsky
- $\epsilon$ -produzioni
- riconoscimento di linguaggi

21

## Grammatiche di Chomsky

una *grammatica* è una quadrupla

$$G = \langle V_T, V_N, P, S \rangle$$

- $V_T \subseteq \Sigma$  è l'alfabeto (finito) di *simboli terminali*
- $V_N$  è un insieme (finito) di *simboli non terminali*, o *categorie sintattiche*, tale che  $V_N \cap \Sigma = \emptyset$
- $P$ , detto insieme delle *produzioni*, è una relazione binaria finita su

$$(V_T \cup V_N)^* \circ V_N \circ (V_T \cup V_N)^* \times (V_T \cup V_N)^*$$

forma sintetica

$\langle \alpha, \beta \rangle \in P$  si indica generalmente con  $\alpha \rightarrow \beta$

- $S \in V_N$  è l'*assioma*

22

## Esempio

una grammatica definisce implicitamente tutte le stringhe di terminali generabili a partire dall'assioma tramite una sequenza di riscritture

**esempio:**

$G = \langle \{a, b, c\}, \{S, B, C\}, P, S \rangle$ , con  $P$  composto da:

- |                      |                      |                      |
|----------------------|----------------------|----------------------|
| ① $S \rightarrow aS$ | ② $S \rightarrow B$  | ③ $B \rightarrow bB$ |
| ④ $B \rightarrow bC$ | ⑤ $C \rightarrow cC$ | ⑥ $C \rightarrow c$  |

genera  $L(G) = \{a^n b^m c^h \mid n \geq 0, m, h \geq 1\}$

forma sintetica

$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \dots \quad \alpha \rightarrow \beta_n$

viene anche indicato con

$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

23

## Linguaggio generato

forma sintetica

$V_T \cup V_N$  viene talvolta indicato con  $V$

- *derivazione diretta*: relazione su

$$(V^* \circ V_N \circ V^*) \times V^*$$

$\langle \varphi, \psi \rangle$  appartiene alla relazione (si scrive  $\varphi \Rightarrow \psi$ ) se

$\exists \alpha \in V^* \circ V_N \circ V^*$  ed  $\exists \beta, \gamma, \delta \in V^*$  t.c.  $\varphi = \gamma \alpha \delta$   $\psi = \gamma \beta \delta$  e  $\alpha \rightarrow \beta \in P$

$\varphi$  e  $\psi$  sono dette *forme di frase*

- *derivazione*: chiusura riflessiva e transitiva della derivazione diretta, si rappresenta con  $\Rightarrow^*$
- il *linguaggio generato* da  $G$  è  $L(G) = \{x \mid x \in V_T^* \wedge S \Rightarrow^* x\}$
- due *grammatiche*  $G_1$  e  $G_2$  sono *equivalenti* se  $L(G_1) = L(G_2)$

forma sintetica

talvolta  $\Rightarrow$  al posto di  $\Rightarrow^*$

24

## Grammatiche formali

**esempio:** generazione di  $\{a^n b^n c^n | n \geq 1\}$

grammatica  $G = \langle \{a, b, c\}, \{S, B, C, F, G\}, P, S \rangle$

con P composto da

- |                               |                       |
|-------------------------------|-----------------------|
| ① $S \rightarrow aSBC$        | ② $CB \rightarrow BC$ |
| ③ $SB \rightarrow bF$         | ④ $FB \rightarrow bF$ |
| ⑤ $FC \rightarrow cG$         | ⑥ $GC \rightarrow cG$ |
| ⑦ $G \rightarrow \varepsilon$ |                       |

per generare **aabbcc**

$S \Rightarrow$  ①  $aSBC \Rightarrow$  ①  $aaSBCBC \Rightarrow$  ②  $aaSBBCC$   
 $\Rightarrow$  ③  $aabFBCC \Rightarrow$  ④  $aabbFCC \Rightarrow$  ⑤  $aabbccGC$   
 $\Rightarrow$  ⑥  $aabbccG \Rightarrow$  ⑦  $aabbcc$

25

## Grammatiche formali

**osservazione:** non è detto che una sequenza di derivazioni porti ad una stringa del linguaggio

**esempio:**

la grammatica  $G = \langle \{a, b, c\}, \{S, A\}, P, S \rangle$  con

$S \rightarrow aSc \mid A$

$A \rightarrow bAc \mid \varepsilon$

genera  $\{a^n b^m c^{n+m} | n, m \geq 0\}$

**esempio:**

la grammatica  $G = \langle \{a, b, c\}, \{S, A\}, P, S \rangle$  con

$S \rightarrow Ab$

$A \rightarrow Sa$

genera  $\Lambda$

26

## Grammatiche di Chomsky

- di tipo 0, non limitate
- di tipo 1, context sensitive, contestuali
- di tipo 2, context free (CF), non contestuali
- di tipo 3, lineari destre (RL), regolari

27

## Grammatiche di Chomsky di tipo 0, non limitate

- sono le meno restrittive
- produzioni del tipo

$$\alpha \rightarrow \beta, \alpha \in V^* \circ V_N \circ V^*, \beta \in V^*$$

ammettono anche derivazioni che accorciano stringhe  
linguaggi di tipo 0

**esempio:**

il linguaggio  $\{a^n b^n | n \geq 1\}$  è di tipo 0 in quanto generato da

$$\begin{aligned} S &\rightarrow aAB & B &\rightarrow b \\ aA &\rightarrow aaAb & aAb &\rightarrow ab \\ aAA &\rightarrow aA \end{aligned}$$

28

## Grammatiche di Chomsky di tipo 1, context sensitive, contestuali

- produzioni che non riducano la lunghezza delle forme di frase

$$\alpha \rightarrow \beta, |\alpha| \leq |\beta|, \alpha \in V^* \circ V_N \circ V^*, \beta \in V^*$$

linguaggi di tipo 1

**esempio:**

il linguaggio  $\{a^n b^n c^n | n \geq 1\}$  è di tipo 0 in quanto generato da

$$\begin{aligned} S &\rightarrow aSBC & CB &\rightarrow BC \\ SB &\rightarrow bF & FB &\rightarrow bF \\ FC &\rightarrow cG & GC &\rightarrow cG \\ cG &\rightarrow c \end{aligned}$$

ma è anche di tipo 1, infatti è generato anche da

$$\begin{aligned} S &\rightarrow aSBc \mid aBc \\ cB &\rightarrow Bc \\ bB &\rightarrow bb \\ aB &\rightarrow ab \end{aligned}$$

29

## Generazione di stringhe di $a^n b^n c^n$

(1) $S \rightarrow aSBc \mid aBc$	(2) $cB \rightarrow Bc$
(3) $bB \rightarrow bb$	(4) $aB \rightarrow ab$

$$\begin{aligned} S &\Rightarrow aSBc && \Rightarrow aaa aBBBBccccc \\ &\Rightarrow aaSBcBc && \Rightarrow aaaa bBBBBccccc \\ &\Rightarrow aaaSBcBcBc && \Rightarrow aaaab bBBBBccccc \\ &\Rightarrow aaaaBcBcBcBc && \Rightarrow aaaabb bBBBBccccc \\ &\Rightarrow aaaaBcBcBcBcBc && \Rightarrow aaaabbb bBBBBccccc \\ &\Rightarrow aaaaBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBcBcBcBcBc && \\ &\Rightarrow aaaaBcBcBcBcBcBcBcBcBcBcBcBc && \end{aligned}$$

30

## Grammatiche di Chomsky di tipo 2, context free (CF), non contestuali

- produzioni del tipo

$$A \rightarrow \gamma, A \in V_N, \gamma \in V^+$$

linguaggi di tipo 2

**esempio:**

il linguaggio  $\{a^n b^n | n \geq 1\}$  è di tipo 0 in quanto generato da

$$S \rightarrow aAb$$

$$aA \rightarrow aaAb$$

$$A \rightarrow \varepsilon$$

ma è anche di tipo 2, infatti è generato anche da

$$S \rightarrow aSb \mid ab$$

31

## Esempi di linguaggi di tipo 2

linguaggio delle espressioni aritmetiche con la variabile  $i$  (come per esempio “ $i * i + (i * i + (i)) * i * i$ ”, oppure “ $((i + i) * i)$ ”).

L’assioma è  $E$ .

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow i \mid (E)$$

grammatica delle parentesi ben bilanciate (esempio “ $(( ( ( ) ) ) ) ( )$ ”)

$$S \rightarrow ( ) \mid SS \mid (S)$$

da quale sequenza di produzioni è generata “ $( ) ( ( ( ) ) )$ ”?

grammatica delle stringhe palindrome (esempio “elle”, “ereggere”)

32



## Grammatiche di Chomsky di tipo 3, lineari destre (RL), regolari

- produzioni del tipo

$$A \rightarrow \delta, A \in V_N, \delta \in (V_T^* V_N) \cup V_T$$

- linguaggi di tipo 3

**esempio:**

il linguaggio  $\{a^n b | n \geq 0\}$  è di tipo 3 in quanto generato da

$$S \rightarrow aS$$

$$S \rightarrow b$$

si possono anche definire grammatiche lineari sinistre (LL) con

$$A \rightarrow \delta, A \in V_N, \delta \in (V_N^* V_T) \cup V_T$$

**esempio:** il linguaggio  $\{a^n b | n \geq 0\}$  è anche generato da

$$S \rightarrow Tb | b$$

$$T \rightarrow a | Ta$$

**teorema:** i linguaggi generati da grammatiche LL e RL coincidono

33

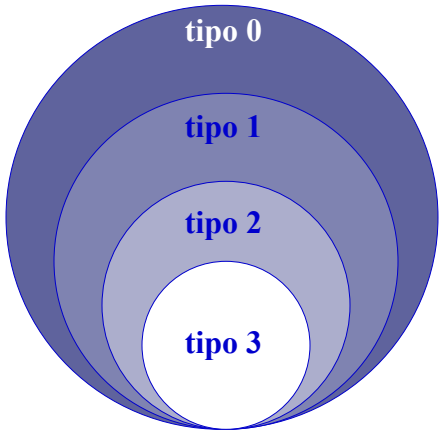
## Grammatiche di Chomsky

un linguaggio è *strettamente di tipo n* se esiste una grammatica di tipo n che lo genera e non esiste una grammatica di tipo  $m > n$  che lo genera

**esempio:** il linguaggio  $\{a^n b^n | n \geq 1\}$  è generato da una grammatica di tipo 2 e non è generato da nessuna grammatica di tipo 3

34

# Grammatiche di Chomsky



contenimento tra i linguaggi

35

# Grammatiche di Chomsky

tipo	produzioni	vincoli
tipo 0 non limitate	$\alpha \rightarrow \beta$	$\alpha \in V^* \circ V_N \circ V^*, \beta \in V^*$
tipo 1 contestuali	$\alpha \rightarrow \beta$	$ \alpha  \leq  \beta $ $\alpha \in V^* \circ V_N \circ V^*, \beta \in V^*$
tipo 2 non contestuali	$A \rightarrow \gamma$	$A \in V_N, \gamma \in V^+$
tipo 3 regolari	$A \rightarrow \delta$	$A \in V_N, \delta \in (V_T \circ V_N) \cup V_T$

quadro riassuntivo della classificazione delle grammatiche secondo Chomsky

36

## ε-produzioni

- con grammatiche di tipo 1, 2, 3 non è possibile generare  $\varepsilon$ 
  - per generare  $\varepsilon$  occorre una produzione  $\alpha \rightarrow \varepsilon$  che viene detta  $\varepsilon$ -produzione
  - per Chomsky tutti i linguaggi che contengono  $\varepsilon$ -produzioni sono linguaggi di tipo 0
- qual'è l'impatto sui corrispondenti linguaggi delle  $\varepsilon$ -produzioni nelle grammatiche?

37

## ε-produzioni in posizione arbitraria

esempio:

$G$  è una grammatica di tipo 0 con una sola produzione  $\alpha \rightarrow \beta$  con  $|\alpha| \geq |\beta|$ ; supponiamo sia  $AB \rightarrow C$ , si può costruire una  $G'$  di tipo 1 e con  $\varepsilon$ -produzioni equivalente a  $G$

$$\begin{cases} G' = \langle V_T, V_N \cup \{H\}, P', S \rangle \\ P' = P - \{AB \rightarrow C\} \cup \{AB \rightarrow CH, H \rightarrow \varepsilon\} \end{cases}$$

teorema:

data una grammatica di tipo 0 esiste una grammatica di tipo 1 con  $\varepsilon$ -produzioni equivalente

dimostrazione:

sia  $G = \langle V_T, V_N, P, S \rangle$  una grammatica di tipo 0, ricaviamo una grammatica equivalente

$G' = \langle V_T, V_N', P', S \rangle$  con  $V_N' = V_N \cup \{X\}$ ,  $X \notin V_N$

a  $P$  aggiungiamo  $X \rightarrow \varepsilon$  e modifichiamo tutte le  $\alpha \rightarrow \beta$  con  $|\alpha| \geq |\beta|$  in  $\alpha \rightarrow \beta X \dots X$  con  $X$  ripetuta  $|\alpha| - |\beta|$  volte

38

## ε-produzioni

- **teorema**  
ad ogni grammatica  $G = \langle V_T, V_N, P, S \rangle$  di tipo 1, 2 o 3 corrisponde una grammatica  $G' = \langle V_T, V_N \cup \{S'\}, P', S' \rangle$  tale che
  - $L(G') = L(G) \cup \{\epsilon\}$
  - $G'$  è dello stesso tipo (1, 2, o 3) di  $G$  a meno di ε-produzioni sull'assioma
- **dimostrazione**
  - sia  $G = \langle V_T, V_N, P, S \rangle$  la grammatica che genera  $L$
  - definiamo  $G' = \langle V_T, V_N \cup \{S'\}, P', S' \rangle$  che genera  $L' = L \cup \{\epsilon\}$
  - se  $G$  è di tipo 1 o 2,  $P' = P \cup \{S' \rightarrow \epsilon, S' \rightarrow S\}$
  - se  $G$  è di tipo 3,  $P' = P \cup \{S' \rightarrow \epsilon\} \cup \{S' \rightarrow \alpha \mid \text{per ogni } S \rightarrow \alpha\}$
- è facile dimostrare che, se l'assioma non compare mai a destra di qualche produzione, vale anche la corrispondenza opposta

39

## ε-produzioni e grammatiche regolari

le grammatiche di tipo 3 (right linear, RL, lineari destre, regolari)  
non modificano la loro natura se aggiungiamo ε-produzioni

**esempio:** la grammatica RL

$$\begin{aligned} S &\rightarrow bX \mid aB \\ B &\rightarrow cX \mid dX \\ X &\rightarrow \epsilon \end{aligned}$$

si può modificare in

$$\begin{aligned} S &\rightarrow b \mid aB \\ B &\rightarrow c \mid d \end{aligned}$$

40

## $\epsilon$ -produzioni e grammatiche regolari

**teorema:**

se  $L$  è il linguaggio definito da una grammatica regolare a cui sono state aggiunte  $\epsilon$ -produzioni, il linguaggio  $L' = L - \{\epsilon\}$  è regolare

**dimostrazione:**

aggiungiamo una  $\epsilon$ -produzione  $E \rightarrow \epsilon$  ad una grammatica regolare

- supponiamo, per semplicità, che l'assioma  $S$  non appaia mai a destra di una produzione, se  $E=S$  il teorema è dimostrato
- supponiamo che  $E$  compaia a destra delle  $k \geq 0$  produzioni  $A_i \rightarrow \alpha_i$
- eliminiamo la produzione  $E \rightarrow \epsilon$  e aggiungiamo  $k$  produzioni  $A_i \rightarrow \alpha_i'$  dove  $\alpha_i'$  è ricavato da  $\alpha_i$  sostituendo  $\epsilon$  alle  $E$

41

## $\epsilon$ -produzioni e grammatiche CF

le grammatiche di tipo 2 (context free, CF, non contestuali) non modificano la loro natura se aggiungiamo  $\epsilon$ -produzioni

**esempio:** nella grammatica CF

$$S \rightarrow AB \mid aB \mid B$$

$$A \rightarrow ab \mid aB$$

$$B \rightarrow cX \mid X$$

$$X \rightarrow \epsilon$$

la  $\epsilon$ -produzione può “migrare” verso l'assioma

$$S \rightarrow AB \mid aB \mid B$$

$$A \rightarrow ab \mid aB$$

$$B \rightarrow c \mid \epsilon$$

nuovamente

$$S \rightarrow AB \mid A \mid aB \mid a \mid B \mid \epsilon$$

$$A \rightarrow ab \mid aB \mid a$$

$$B \rightarrow c$$

42

## $\epsilon$ -produzioni e grammatiche CF

### teorema:

se  $L$  è il linguaggio definito da una grammatica CF a cui sono state aggiunte  $\epsilon$ -produzioni, il linguaggio  $L' = L - \{\epsilon\}$  è CF

### dimostrazione:

aggiungiamo una  $\epsilon$ -produzione  $E \rightarrow \epsilon$  ad una grammatica CF

- supponiamo, per semplicità, che l'assioma non appaia mai a destra di una produzione, se  $E$  è l'assioma il teorema è dimostrato
- supponiamo che  $E$  compaia a destra delle  $k \geq 0$  produzioni  $A_i \rightarrow \alpha_i$
- eliminiamo la produzione  $E \rightarrow \epsilon$  e aggiungiamo  $k$  produzioni  $A_i \rightarrow \alpha_i'$  dove  $\alpha_i'$  è ricavato da  $\alpha_i$  sostituendo  $\epsilon$  alle  $E$
- se non ci sono più  $\epsilon$ -produzioni o se ci sono solo sull'assioma il procedimento termina, altrimenti il procedimento si ripete per tutte le nuove  $\epsilon$ -produzioni
- la terminazione è garantita dalla finitezza di  $V_N$

43

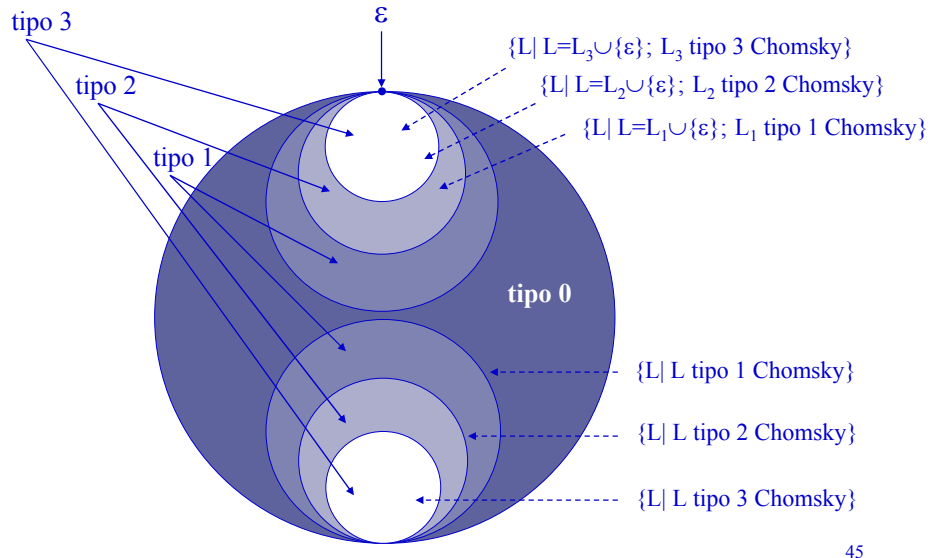
## $\epsilon$ -produzioni: ricapitolazione

coerentemente a quanto stabilito dai teoremi precedenti nel seguito ammetteremo, nelle diverse grammatiche, le seguenti  $\epsilon$ -produzioni:

tipo	$\epsilon$ -produzioni ammesse
0	tutte (per definizione)
1	solo sull'assioma quando quest'ultimo non compare mai a destra di una produzione
2	tutte
3	tutte

44

## Grammatiche con $\varepsilon$ -produzioni



45

## Esempi di grammatiche

- il linguaggio  $\{w \circ w \mid w \in (a+b)^*\}$
- è generato dalla grammatica contestuale

<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>
$S \rightarrow T \mid \varepsilon$ $T \rightarrow aAT \mid bBT \mid A_0a \mid B_0b$	$Aa \rightarrow aA$ $Ab \rightarrow bA$ $Ba \rightarrow aB$ $Bb \rightarrow bB$	$AA_0 \rightarrow A_0a$ $BA_0 \rightarrow A_0b$ $AB_0 \rightarrow B_0a$ $BB_0 \rightarrow B_0b$	$A_0 \rightarrow a$ $B_0 \rightarrow b$

le 1 generano insieme caratteri della prima e della seconda stringa;  $A_0$  ( $B_0$ ) è l'ultimo carattere della prima stringa

le 2 e le 3 separano la prima stringa dalla seconda; le 4 chiudono la generazione, se sono applicate troppo presto il processo diverge

46

## Esempi di grammatiche

- il linguaggio  $\{(x\#)^* \mid x = \text{permutazione di } (a,b,c)\}$  (che contiene per esempio le stringhe  $\varepsilon$ ,  $abc\#$ ,  $acb\#$ ,  $bac\#cab\#$ , ecc)

è generato dalla grammatica contestuale:

$$\begin{array}{lll} S \rightarrow S' \mid \varepsilon & AB \rightarrow BA & A \rightarrow a \\ S' \rightarrow ABC\# & AC \rightarrow CA & B \rightarrow b \\ S' \rightarrow ABC\#S' & BC \rightarrow CB & C \rightarrow c \end{array}$$

ma è generato anche dalla grammatica CF:

$$S \rightarrow E\#S \mid \varepsilon \quad E \rightarrow abc \mid acb \mid cba \mid cab \mid bac \mid bca$$

ed anche dalla grammatica regolare:

$$\begin{array}{lll} S \rightarrow aX \mid bY \mid cZ \mid \varepsilon & R \rightarrow \#S \\ X \rightarrow bX' \mid cX'' & Y \rightarrow aY' \mid cY'' & Z \rightarrow aZ' \mid bZ'' \\ X' \rightarrow cR & Y' \rightarrow cR & Z' \rightarrow bR \\ X'' \rightarrow bR & Y'' \rightarrow aR & Z'' \rightarrow aR \end{array}$$

47

## Linguaggi lineari

- esistono classificazioni dei linguaggi alternative a quella di Chomsky
- una grammatica è *lineare* quando la parte sinistra di ogni produzione è costituita da *esattamente un* non terminale e la parte destra contiene *al più un* non terminale
- le grammatiche di tipo 3 sono lineari
- esistono grammatiche CF lineari

**esempio:**

$$S \rightarrow aSb$$

$$S \rightarrow c$$

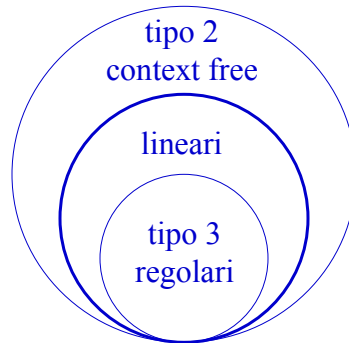
che genera  $\{a^n c b^n \mid n \geq 0\}$

- i linguaggi strettamente lineari sono un sottoinsieme di quelli strettamente CF e un soprainsieme di quelli regolari

48



## Grammatiche lineari e grammatiche di Chomsky



contenimento tra i linguaggi

49

## Forma normale di Backus

- la BNF è una notazione CF con alcuni accorgimenti sintattici che ne aumentano la leggibilità

esempio

```
<sequenza istruzioni> ::= <istruzione>;  
                        {<istruzione>;}
```

può essere riscritto:

$$Q \rightarrow I; \mid I; Q$$

```
<istruzione if> ::= if ( <condizione> )  
                  <istruzione> [else <istruzione>]
```

può essere riscritto:

$$F \rightarrow \text{if}(C)I \text{ else } I \mid \text{if}(C)I$$

50

## Riconoscimento dei linguaggi

### problema:

stabilire se una stringa appartiene ad un dato linguaggio

- esistono linguaggi a cui non corrisponde alcun algoritmo di decisione
- i linguaggi di tipo 3 sono riconosciuti da dispositivi con memoria costante in tempo lineare (automi a stati finiti)
- i linguaggi strettamente di tipo 2 sono riconosciuti da dispositivi non deterministici con pila in tempo lineare (automi a pila non deterministici)

51

## Riconoscimento dei linguaggi

- i linguaggi strettamente di tipo 1 sono riconosciuti da dispositivi non deterministici con memoria che cresce linearmente con la lunghezza della stringa da esaminare (automi non deterministici “linear bounded”)
- i linguaggi strettamente di tipo 0 sono riconosciuti da macchine di Turing con memoria e tempo illimitati
- è possibile che non esista un algoritmo di decisione ma un processo semidecisionale, in cui, se la stringa non fa parte del linguaggio non è detto che la computazione termini

52