1 Linguaggi e grammatiche

lunedì 23 settembre 2024 14:08



02-linguaggi-e-grammatiche-26

# Linguaggi
# e
# Grammatiche

1

1

## Linguaggi e informatica

- ubiquitari nelle applicazioni
  - linguaggi di programmazione
    - compilatori e interpreti
  - linguaggi per intefacce
    - specifica di sequenze di operazioni
  - linguaggi di comunicazione
    - protocolli per il dialogo tra computer
  - large language models
    - esempio di applicazione chatgpt

2

2





02-linguaggi-e-grammatiche-26

## Linguaggi e informatica

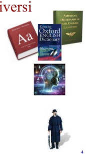
- paradigmatici nella teoria
  - molti importanti problemi teorici sono riconducibili a quello della verifica dell'appartenenza di una frase (*stringa di caratteri*) a un linguaggio

3

3

## Approcci diversi

- approccio insiemistico
  - utile per determinare le proprietà elementari dei linguaggi
  - il linguaggio viene descritto usando una notazione insiemistica



4

4

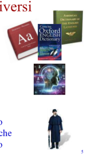




02-linguaggi-e-grammatiche-26

## Approcci diversi

- approccio generativo
  - grammatiche formali
  - il linguaggio è specificato dalle regole di una grammatica
  - seguendo le regole si generano tutte le frasi che appartengono al linguaggio ed esclusivamente le frasi che appartengono al linguaggio

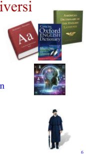


5

5

## Approcci diversi

- approccio riconoscitivo
  - automi riconoscitori
  - il linguaggio è composto esclusivamente dalle frasi che sono riconosciute da un automa (es. un software opportunamente programmato); le frasi che non sono riconosciute non fanno parte del linguaggio



6

6



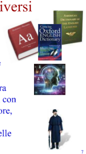


02-linguaggi-e-grammatiche-26

## Approcci diversi

Questo è l'approccio più complicato

- approccio statistico
  - es. large language models
  - a un automa sono mostrate molte frasi del linguaggio
  - l'automa riconosce o genera come parte del linguaggio, con una certa probabilità d'errore, le frasi che sono statisticamente simili a quelle che gli sono state proposte



7

7

## Concetti matematici di base

- Insiemi
- Relazioni
- Funzioni

8







## Insiemi

- consideriamo insiemi *finiti* e insiemi *infiniti*
- $|A|$ = cardinalità dell'insieme (finito) A
- alcuni insiemi infiniti di numeri:

| | | | |
|---|---|---|---|
| N | naturali (contiene lo zero) | Q | razionali relativi |
| $N^+$ | naturali positivi | $Q^+$ | razionali positivi |
| | | $Q^-$ | razionali negativi |
| Z | interi relativi | | |
| $Z^+$ | interi positivi | R | reali |
| $Z^-$ | interi negativi | $R^+$ | reali positivi |
| | | $R^-$ | reali negativi |

9

Naturali e Reali sono insiemi infiniti, ma quali sono di più?
Lo vedremo.

9

## Sottoinsiemi e insiemi uguali

- dati due insiemi A e B, se
  $x \in B \Rightarrow x \in A$
  allora B è *sottoinsieme* di A, e si scrive $B \subseteq A$
- ogni insieme è sottoinsieme di se stesso
- l'insieme vuoto $\varnothing$ è sottoinsieme di ogni insieme
- se A e B sono finiti, allora $B \subseteq A \Rightarrow |B| \leq |A|$
- A e B *insiemi uguali*
  $A=B \Leftrightarrow (x \in A \Leftrightarrow x \in B)$
  si può scrivere anche
  $A=B \Leftrightarrow (A \subseteq B \wedge B \subseteq A)$
- A è *sottoinsieme proprio* di B ($A \subset B$) se
  $(A \subseteq B) \wedge (A \neq B)$

10

10



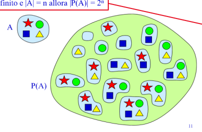




## Insieme delle parti

l'insieme dei sottoinsiemi di A è detto l'*insieme delle parti* di A e si indica con P(A) o $2^A$

$\boxed{\text{se A è finito e } |A| = n \text{ allora } |P(A)| = 2^n}$




11

11

C'è anche l'insieme vuoto poiché
l'insieme vuoto
È sottoinsieme di qualunque
insieme.

Si può dimostrare per induzione.
Caso base, e poi per n elementi e per n+1 elementi.
Chiedi a chat

### Teorema:

Se $A$ è un insieme finito di cardinalità $|A| = n$, allora la cardinalità dell'insieme delle parti di $A$, $P(A)$, è $|P(A)| = 2^n$.

### Dimostrazione per induzione:

**Passo 1: Caso base** ($n = 0$)

Quando $A = \emptyset$, cioè $A$ è l'insieme vuoto, sappiamo che l'unico sottoinsieme di $\emptyset$ è l'insieme vuoto stesso:

$$P(\emptyset) = \{\emptyset\}$$

Quindi, la cardinalità dell'insieme delle parti è:

$$|P(\emptyset)| = 1$$

Ora verifichiamo che la formula $2^n$ sia valida per $n = 0$:

$$2^0 = 1$$

Il caso base è verificato.

**Passo 2: Passo induttivo**

Supponiamo che il teorema sia vero per un insieme $A$ di cardinalità $|A| = n$. In altre parole, assumiamo l'ipotesi induttiva:

$$|P(A)| = 2^n$$

Ora dobbiamo dimostrare che la proprietà vale anche per un insieme $A'$ di cardinalità $n + 1$, dove $A' = A \cup \{x\}$ e $x$ è un nuovo elemento che non appartiene a $A$. La cardinalità di $A'$ è quindi:

$$|A'| = n + 1$$

L'insieme delle parti di $A'$, $P(A')$, è l'insieme di tutti i sottoinsiemi di $A'$. Notiamo che ogni sottoinsieme di $A'$ può essere scritto in uno dei due seguenti modi:

1. Un sottoinsieme che **non contiene** $x$.
2. Un sottoinsieme che **contiene** $x$.

Ora vediamo come questi sottoinsiemi sono legati all'insieme $A$:

- **Sottoinsiemi che non contengono** $x$: Questi sono esattamente i sottoinsiemi di $A$. Quindi, il numero di tali sottoinsiemi è $|P(A)| = 2^n$ per ipotesi induttiva.
- **Sottoinsiemi che contengono** $x$: Per ogni sottoinsieme di $A$, possiamo ottenere un nuovo sottoinsieme di $A'$ aggiungendo $x$ a quel sottoinsieme. Poiché ci sono $2^n$ sottoinsiemi di $A$, ci sono anche $2^n$ sottoinsiemi di $A'$ che contengono $x$.

In totale, il numero di sottoinsiemi di $A'$ è la somma dei sottoinsiemi che non contengono $x$ e di quelli che contengono $x$:

$$|P(A')| = 2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$$

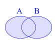
---

**Passo 3: Conclusione**

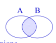
Abbiamo dimostrato che, se l'ipotesi è vera per un insieme di cardinalità $n$, allora è vera anche per un insieme di cardinalità $n + 1$. Poiché abbiamo verificato il caso base e il passo induttivo, per il principio di induzione matematica, il teorema ro per ogni $n \in \mathbb{N}$.

## Operazioni tra insiemi

- *unione* $C = A \cup B$
  - se A e B sono finiti $|C| \leq |A|+|B|$
  - commutativa e associativa




- *intersezione* $C=A \cap B$
  - se A e B sono finiti $|C| \leq \min\{|A|, |B|\}$
  - commutativa e associativa
  - l'intersezione è distributiva rispetto all'unione




Si fa riferimento a 2 operatori (vedi wiki).

- *partizione* di A
  - insieme di n sottoinsiemi di A tali che
    $A_1 \cup A_2 \cup \ldots \cup A_n = A$
    $i \neq j \Rightarrow A_i \cap A_j = \varnothing$


12

12







Questa è una delle proprietà fondamentali delle operazioni sugli insiemi.

### Dimostrazione:

**Passo 1: Dimostrazione di $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$**

Dobbiamo dimostrare che ogni elemento appartenente al lato sinistro dell'equazione appartiene anche al lato destro.

- Sia $x \in A \cap (B \cup C)$. Per la definizione di intersezione e unione, questo significa che:
  - $x \in A$
  - $x \in B \cup C$ (cioè $x$ appartiene a $B$ o a $C$)
- Ora consideriamo i due casi che derivano da $x \in B \cup C$:
  - **Caso 1**: Se $x \in B$, allora $x \in A \cap B$. Di conseguenza, $x \in (A \cap B) \cup (A \cap C)$.
  - **Caso 2**: Se $x \in C$, allora $x \in A \cap C$. Anche in questo caso, $x \in (A \cap B) \cup (A \cap C)$.

In entrambi i casi, abbiamo dimostrato che $x \in (A \cap B) \cup (A \cap C)$.

Quindi, ogni elemento di $A \cap (B \cup C)$ appartiene anche a $(A \cap B) \cup (A \cap C)$, cioè:

$$A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$$

**Passo 2: Dimostrazione di $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$**

Ora dobbiamo dimostrare che ogni elemento appartenente al lato destro dell'equazione appartiene anche al lato sinistro.

- Sia $x \in (A \cap B) \cup (A \cap C)$. Per la definizione di unione, questo significa che $x \in A \cap B$ **oppure** $x \in A \cap C$.
- Consideriamo i due casi:
  - **Caso 1**: Se $x \in A \cap B$, allora $x \in A$ e $x \in B$, quindi $x \in B \cup C$ (poiché $x \in B$). Di conseguenza, $x \in A \cap (B \cup C)$.
  - **Caso 2**: Se $x \in A \cap C$, allora $x \in A$ e $x \in C$, quindi $x \in B \cup C$ (poiché $x \in C$). Anche in questo caso, $x \in A \cap (B \cup C)$.

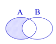
In entrambi i casi, $x \in A \cap (B \cup C)$.

Quindi, ogni elemento di $(A \cap B) \cup (A \cap C)$ appartiene anche a $A \cap (B \cup C)$, cioè:

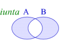
$$(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$$

## Operazioni tra insiemi

- *complemento* di B rispetto ad A
  $C = A-B = \{x | x \in A \wedge x \notin B\}$




- *differenza simmetrica* o *somma disgiunta*
  $A+B=A \cup B-(A \cap B)$




- *prodotto cartesiano* $C=A \times B$
  $C= \{\langle x,y \rangle | x \in A \wedge y \in B\}$
  - insieme di tutte le possibili coppie ordinate
  - il prodotto cartesiano è associativo ma non commutativo

13

13

## Relazioni

- siano $A_1, A_2, \ldots, A_n$ n insiemi
  (non necessariamente distinti)
- una *relazione n-aria* è un sottoinsieme di

$$A_1 \times A_2 \times \ldots \times A_n$$

$$R \subseteq A_1 \times A_2 \times \ldots \times A_n$$

Prodotto
cartesiano
di n insiemi

esempio:
- la relazione "minore di" definita sui naturali è l'insieme
  $R \subseteq N \times N = N^2$, dove $R=\{\langle x,y \rangle | x<y\}$

14

14

## Relazione d'ordine

C'è un ordine tra gli elementi dell'insieme. Ma non è un ordine Totale: elementi--> {a,b,c,d} = A.
se facessi A*A --> nell'insieme ci sarebbe: {<a,a>, <a,b>, ....}
- R= {<a,b>, <a,c>} --> questa è una relazione sull'insieme A? si perché è un Sottoinsieme ma non d'ordine perché non vale la proprietà riflessiva.
- R={<a,b>, <a,c>, bd, cd, aa, bb, cc, dd } --> si è una relazione ma manca la transitività
- R={<a,b>, <a,c>, bd, cb,ba, aa, bb, cc, dd, ad } --> è una relazione, è antisimmetrica, è transitiva --> allora è una relazione d'ordine.

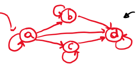
In questo ordinamento viene prima b o c? non lo sappiamo

- $R \subseteq A^2 = A \times A$ è una *relazione d'ordine* se valgono le seguenti proprietà:
  1. *riflessività*
     $\langle x,x\rangle \in R$
  2. *antisimmetria*
     $\langle x,y\rangle \in R \wedge \langle y,x\rangle \in R \Rightarrow x=y$
  3. *transitività*
     $\langle x,y\rangle \in R \wedge \langle y,z\rangle \in R \Rightarrow \langle x,z\rangle \in R$

un insieme su cui è definita una relazione d'ordine si dice parzialmente ordinato o *poset* ("partially ordered set")

esempio: la relazione "≤" è una relazione d'ordine su N

15



rapp. grafica della relazione

tra b e c non sai chi viene prima → partial ordered set → insieme parzialmente ordinato

15

- Altro Esempio:
relat. "≤" sui naturali:

→ posso non disegnarli perchè sono scontati

- Solo "<" non è relat d'ordine → manca la riflessività

## Relazione d'ordine totale:

se so dire cosa c'è prima e cosa c'è dopo un elemento

- una relazione d'ordine $R \subseteq A^2$ è detta *totale* se
  $\langle x,y\rangle \in A^2 \Rightarrow \langle x,y\rangle \in R \vee \langle y,x\rangle \in R$

esempio:
la relazione "≤" è una relazione d'ordine totale su N
$1 \leq 2 \leq 3 \leq 4 \leq 5 \leq 6 \leq 7 \leq 8 \ldots$

16

16







## Relazione di equivalenza

- $R \subseteq A^2 = A \times A$ è una *relazione di equivalenza* se valgono le seguenti proprietà:
  1. *riflessività*
     $\langle x,x\rangle \in R$
  2. *simmetria*
     $\langle x,y\rangle \in R \Rightarrow \langle y,x\rangle \in R$
  3. *transitività*
     $\langle x,y\rangle \in R \wedge \langle y,z\rangle \in R \Rightarrow \langle x,z\rangle \in R$

esempio: la relazione "=" è una relazione di equivalenza su R

17

17

Es. insieme di 3 elementi:
A = {a, b, c}

così è di equivalenza

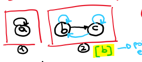
## Relazione di equivalenza

- un insieme A su cui è definita una relazione di equivalenza R si può partizionare in sottoinsiemi massimali di equivalenza, detti *classi di equivalenza*
- l'insieme delle classi di equivalenza di A è detto *insieme quoziente* e si denota A/R
- un elemento di A/R si denota con [a]
- il numero di classi di A/R si chiama *indice* di R → è 2 in questo caso.

18

18

classi di equivalenza:
(vale la rifl, simm e transitività)



① ② [b] → potevo mettere anche [c] è uguale

una classe è massimale se non posso espandere di più
l'insieme quoziente ha 2 elementi: ① e ②







## Esempio di relazione di equivalenza

- consideriamo la relazione $E_k$ su N
  $n \equiv_k m$
  se esistono q, q', r (con r<k) tali che
  $n=qk+r$ e $m=q'k+r$
- $E_k$ è una relazione di equivalenza
- le sue classi sono le classi resto rispetto alla divisione per k

19

19

Molti problemi teorici sono riconducibili a quello della verifica dell'appartenenza di una frase a un linguaggio. Per farlo, si possono seguire diversi approcci
- Insiemistico: descrizione del linguaggio tramite notazione insiemistica
- Generativo: si specificano le regole del linguaggio, e seguendo le regole si generato tutte ed esclusivamente le frasi che appartengono al linguaggio
- Riconoscitivo: il linguaggio è composto dalle frasi che sono riconosciute da un automa, quelle non riconosciute no
- Statistico: a un automa sono mostrate molte frasi di un linguaggio e l'automa genera o riconosce **con una certa probabilità di errore** le frasi che sono statisticamente simili a quelle proposte

Insiemi: finiti e infiniti (naturali, razionali, reali, interi)
Sottoinsieme proprio se non coincide con l'altro insieme

Insieme delle parti: insieme di tutti i sotto insiemi. La sua cardinalità è 2^(cardinalità dell'insieme di partenza), dimostrato per induzione a partire dall'insieme vuoto, n = 0.
l'operazione di intersezione è distributiva rispetto all'unione

Relazione d'ordine: riflessiva, anti simmetrica e transitiva
totale: vale in entrambe le "direzioni" <a,b> e <b,a>
ad esempio, <= (e non solo < perché manca la rifl)
Relazione d'equivalenza: riflessiva, simmetrica e transitiva

le relazioni sono insiemi e posso definirne delle operazioni:

## Operazioni su relazioni

- *unione*
  $R_1 \cup R_2 = \{\langle x,y\rangle \mid \langle x,y\rangle \in R_1 \vee \langle x,y\rangle \in R_2\}$
- *complementazione*
  $\underline{R} = \{\langle x,y\rangle \mid \langle x,y\rangle \notin R\}$
- *chiusura transitiva*
  $R^+ = \{\langle x,y\rangle \mid \exists y_1, \ldots, y_n \in A,\ n \geq 2,\ y_1=x,\ y_n=y$ tali che $\langle y_i, y_{i+1}\rangle \in R,\ i=1, \ldots, n-1\}$
- *chiusura transitiva e riflessiva*
  $R^* = R^+ \cup \{\langle x,x\rangle \mid x \in A\}$

20

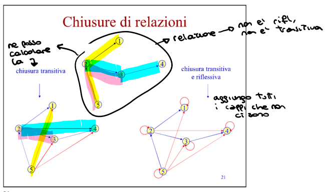
20







## Chiusure di relazioni




21

## Funzioni

$$R \subseteq X_1 \times \ldots \times X_n$$

è una *relazione funzionale* se

$$\forall \langle x_1, \ldots, x_{n-1} \rangle \in X_1 \times \ldots \times X_{n-1}$$

esiste al più un elemento $x_n \in X_n$ tale che

$$\langle x_1, \ldots, x_{n-1}, x_n \rangle \in R$$

si chiama *funzione* la legge che associa $\langle x_1, \ldots, x_{n-1} \rangle$ ad $x_n$

$$f(x_1, \ldots, x_{n-1}) = x_n$$

$$f: X_1 \times \ldots \times X_{n-1} \to X_n$$

$X_1 \times \ldots \times X_{n-1}$ è il *tipo* della funzione

22

22







## Funzioni: dominio codominio

dom(f) = *dominio* di f

sottoinsieme di $X_1 \times \ldots \times X_{n-1}$

$dom(f) = \{\langle x_1, \ldots, x_{n-1} \rangle \in X_1 \times \ldots \times X_{n-1} \mid \exists\, x_n \in X_n \;\; f(x_1, \ldots, x_{n-1}) = x_n\}$

cod(f) = *codominio* di f

sottoinsieme di $X_n$

$cod(f) = \{x_n \in X_n \mid \exists \langle x_1, \ldots, x_{n-1} \rangle \in X_1 \times \ldots \times X_{n-1} \;\; f(x_1, \ldots, x_{n-1}) = x_n\}$

23

23

## Funzioni: fibra

dato un $x_n$

$f^{-1}(x_n)$ = *controimmagine* o *fibra* di $x_n$

sottoinsieme di $X_1 \times \ldots \times X_{n-1}$

$f^{-1}(x_n) = \{\langle x_1, \ldots, x_{n-1} \rangle \in X_1 \times \ldots \times X_{n-1} \mid \langle x_1, \ldots, x_{n-1} \rangle \in dom(f) \wedge f(x_1, \ldots, x_{n-1}) = x_n\}$

24

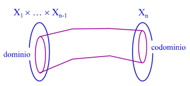
24







## Funzione




25

25

Es:

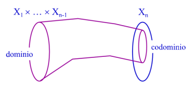
$A = \{a, b, c\}$

$A \times A$

$R = \{\langle a, b \rangle, \langle a, c \rangle\}$ → R è una funzione? No → 1 el dom → 1 el codom.

## Funzione totale

- una funzione f è *totale* se $dom(f) = X_1, \ldots, X_{n-1}$




26

26

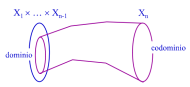






## Funzione suriettiva

- una funzione f è *suriettiva* se $cod(f) = X_n$

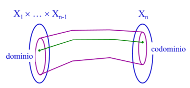



27

27

## Funzione iniettiva

- una funzione f è *iniettiva* se $|f^{-1}(x_n)|=1$

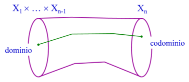



29

28

## Funzione biiettiva

- una funzione f è *biiettiva* (biiezione) se è iniettiva, suriettiva e totale




29

29

## Alfabeto

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

30

30





## Concatenazione

- dato un alfabeto $\Sigma$, definiamo l'operazione binaria *concatenazione* (denotata con "$\circ$")
  $a \circ b = ab$ con $a, b \in \Sigma$
- indichiamo con $a^n$ la concatenazione di a con se stessa n volte
  esempio: $a^4 = a \circ a \circ a \circ a = aaaa$
- 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 $\varepsilon$ = "" otteniamo $\Sigma^*$

31

La concatenazione viene usata non solo per 2 caratteri ma anche per stringhe.
Si può anche non scrivere "°".
Stringa(parola) e frase  sono sinonimi.

--> è associativa, ma non commutativa perché a°b è diverso da b°a.
Sigma è l'alfabeto.
Stringa vuota--> stringa che non contiene nessun carattere: ""

31

## 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 $\varnothing$)
  – attenzione:

$$\boxed{\begin{array}{c}\Lambda \equiv \varnothing \\ \Lambda \neq \{\varepsilon\}\end{array}}$$

32

Un linguaggio è un sottoinsieme di sigma* e contiene stringhe di lunghezza finita ma in numero finito o infinito
Un linguaggio è un sottoinsieme di sigma asteriscato.
Un linguaggio contiene stringhe di lunghezza infinita? No, tutte le stringhe di cui parliamo hanno lunghezza finita.

Data la definizione, i nostri linguaggi hanno un insieme finito di stringhe o infinito di stringhe? --> può essere sia finito che infinito. Se infinito, per esempio:

$\Sigma = \{a, b\} \longrightarrow \Sigma^+ = \{a, b, ab, ba, \ldots\ldots\}$
(su questo alfabeto)

Sigma + ha un insieme infinito di stringhe.
Sigma*= sigma+ $\cup$ {epsilon (stringa vuota)}.
Può essere sia infinito (come sopra in rosso) che finito perché i sottoinsiemi possono avere cardinalità finita.

32



Un linguaggio che contiene solo la stringa vuota non è vuoto ma ha cardinalità =1.



## Operazioni sui linguaggi

$L_1$ e $L_2$ linguaggi

- *unione*
  $L_1 \cup L_2 = \{x \in \Sigma^* \mid x \in L_1 \vee x \in L_2\}$
  $L_1 \cup \Lambda = L_1$
- *intersezione*
  $L_1 \cap L_2 = \{x \in \Sigma^* \mid x \in L_1 \wedge x \in L_2\}$
  $L_1 \cap \Lambda = \Lambda$
- *complementazione*
  $\underline{L_1} = \{x \in \Sigma^* \mid x \notin L_1\}$

33

Unione: metto insieme le stringhe dei due linguaggi. Se fai l'unione con il linguaggio vuoto riottengo lo stesso linguaggio;

Intersezione: solo le stringhe in comune.

Complementazione di un linguaggio: il riferimento è sigma*, la complementazione è fatta rispetto a sigma*:


il complemento di L1, è l'insieme delle stringhe x che appartengono a sigma* ma che non appartengono a L1: --> parte rossa

33

## Operazioni sui linguaggi

$L_1$ e $L_2$ linguaggi

- *concatenazione* o *prodotto*
  $L_1 \circ L_2 = \{x \in \Sigma^* \mid$
  $\exists x_1 \in L_1 \wedge \exists x_2 \in L_2$ tali che $x = x_1 \circ x_2\}$
  $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$
  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
  $L^0 = \{\varepsilon\}$
  $L^h = L \circ L^{h-1}$, per $h \geq 1$

34

Concatenazione tra linguaggi: prendi i due linguaggi e quando li concateni prendi qualunque stringa di L1 e di L2 e le appiccichi tra loro, in tutti i possibili modi.
Così ottengo la concatenazione dei due linguaggi.

a^n --> linguaggio di stringhe che contiene solo a;
b^m --> uguale a sopra ma contiene solo b;
Se calcolo la loro concatenazione: ottengo stringhe che hanno prima tutte a e poi tutte b. la stringa "abb" c'è? Si. "ba" invece non c'è perché la concatenazione non è commutativa

Potenza di un linguaggio: concatenazione di un linguaggio con se stesso per n volte. L^0 è un linguaggio contenente solo la stringa vuota.
L^h --> concateno le stringhe del linguaggio L, h volte

34





## 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} $L^* = \{a^{2n} \mid n \geq 0\}$
- *chiusura transitiva* (non riflessiva) di un linguaggio
  $L^+ = \bigcup_{h=1}^{\infty} L^h$
  esempio: L={aa} $L^+ = \{a^{2n} \mid n \geq 1\}$
  $L^* = L^+ \cup \{\varepsilon\}$

35

Operatore di Kleene:
L* è l'unione di un linguaggio con se stesso infinite volte compreso lo 0.
(Epsilon gli appartiene a L*).

L={aa} --> linguaggio che contiene una sola stringa, ovvero "aa".
Se uso Kleene: L* = a^2n, concateno a tante volte, ma in numero pari.

Chiusura transitiva--> uguale alla precedente ma h parte da 1 invece che da 0.
Quindi non ho la stringa vuota.

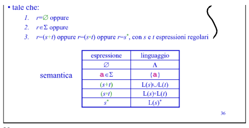
35

## Espressioni regolari

- è uno strumento per descrivere linguaggi (vedremo nel seguito quali)
- dato un alfabeto $\Sigma$, si definisce *espressione regolare* ogni stringa *r*
  $r \in (\Sigma \cup \{+, *, (, ), \circ, \varnothing\})^+$

Espressioni regolari: sono strumenti per descrivere i linguaggi.

Le stringhe che sono espressioni regolari sono definite in questo modo.




Questi caratteri in verde sono simboli dell'alfabeto.

Queste stringhe hanno un significato (semantica): una espressione regolare (stringa del linguaggio) descrive un linguaggio.
 - Se l'espressione è un singolo carattere -->viene denotato il linguaggio che contiene 1 sola stringa che contiene un solo carattere. E così via…

36





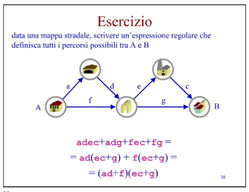

Primo esempio: (a+b)* è una espressione regolare?
a è una espressione regolare, e anche b --> lo è anche la loro somma --> lo è anche (a+b)*.
Il linguaggio rappresentato da questa espressione regolare è: a descrive il linguaggio che contiene una sola stringa "a", così come il linguaggio per "b". Se faccio la somma ottengo un linguaggio che contiene 2 stringhe: "a" e "b" poiché ho fatto l'unione.
(a+b)* --> Sono stringhe create in modo qualunque con la combinazione di a e b. è un linguaggio infinito

Secondo esempio: (a+b)*a è infinito come linguaggio.
Ma con le espressioni regolari posso definire linguaggi finiti? Si:
(a+b)a --> è finito e ne fanno parte { aa, ba } che sono solo due stringhe

Il linguaggio ha due stringhe {a, bcd}

37

Sottoclasse perché: il linguaggio è un sottoinsieme di sigma * e di sottoinsiemi ne puoi fare tanti. Per ciascun sottoinsieme di sigma* esiste sempre una espressione regolare che lo rappresenta? NO.




$(a^*b^*)^* =$

ab
ab
ab
aaab

Adec + fg + fec+adg

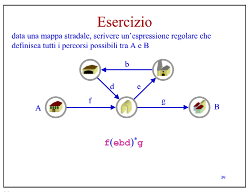
F(g+ec) + ad(ec+g)

(Ad+f )(g+ec) --> è una espressione regolare che descrive il linguaggio regolare dei percorsi.
Posso chiedermi : il percorso x porta da A a B? si, se x è una stringa del linguaggio

38

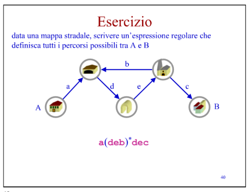







F(ebd)*g
Qui ho cammini infiniti ma iniziano tutti con la f e finiscono tutti con la g.
Qui viene descritto un linguaggio con un numero infinito di stringhe che iniziano con f e terminano con g.

39

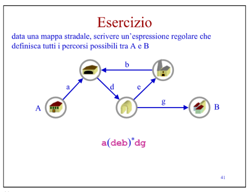



A(de+debde*)c
A(de(bde)*)c, levo parentesi:
Ade(bde)*c

40

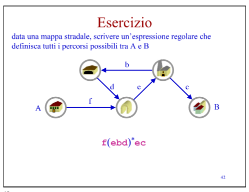







A(debd*+d)g
Raggruppo:
Ad(ebd*)g

41

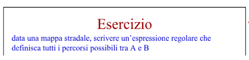



F(e+ ebde)c
F(e(bde)*)c
Fe(bde)*c

42

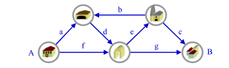







- espr reg su a,b che contengono esatt 1! volta il caratt b:

$a^*b\,a^*$ ✓




$f(ebd)^*g + f(ebd)^*ec + a(deb)^*dec + a(deb)^*dg =$
$= f(ebd)^*(g+ec) + a(deb)^*d(g+ec) =$
$= (f(ebd)^*+a(deb)^*d)\ (g+ec) =$
$= (f+ad)(ebd)^*(g+ec)$

43

43

- che contengono esattamente 2 b che non sono consecutive

$a^*b\,a^*b\,a^*$

-

Scrivi l'espressione regolare che descrive il linguaggio sull'alfabeto {a,b} di tutte le stringhe che iniziano con a e finiscono con b.

Risposta: a(a+b)*b

La risposta corretta è : a(a+b)*b

Scrivi l'espressione regolare che descrive il linguaggio sull'alfabeto {a,b} di tutte le stringhe che contengono almeno una b.

Risposta: (a+b)*b(a+b)*

## 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

44

44

Nelle grammatiche formali le stringhe sono descritte con metodi "di riscrittura".





## Grammatiche formali

- grammatiche di Chomsky
- ε-produzioni
- riconoscimento di linguaggi

45

45

## Grammatiche di Chomsky

una *grammatica* è una quadrupla

$G=<V_T, V_N, P, S>$

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

$$\underbrace{(V_T\cup V_N)^* \circ V_N \circ (V_T\cup V_N)^*}_{a} \times \underbrace{(V_T\cup V_N)^*}_{b}$$

forma sintetica
$<\alpha,\beta>\in P$ si indica generalmente con $\alpha\to\beta$

- $S\in V_N$ è *l'assioma*

46

46

La grammatica di Chomsky è una quadrupla, ho bisogno di 4 elementi:
Vt--> alfabeto di riferimento (alfabeto finito di simboli, ovvero i simboli terminali);
Vn-> insieme dei simboli non terminali (strumenti utilizzati nelle fasi intermedie nel processo generativo delle stringhe)
Questi due devono essere diversi (Vt e Vn)

P--> l'insieme delle produzioni, una produzione è una relazione binaria definita come nella slide qua a sx. quella espressione vuol dire:
- X (== prodotto cartesiano) ci dice le coppie come sono fatte : il primo elemento della coppia viene da a ed il secondo da b.

L'operatore di kleene mi dice che posso mettere ciò che mi pare.
P è un insieme di coppie <alfa,beta> rappresentate così: alfa-->beta.
Alfa--> ciò che c'è a sx della coppia;
beta--> ciò che sta a destra della coppia
S--> punto di partenza del processo generativo.





## Esempio

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

esempio:

G=<{a, b, c}, {S, B, C}, P, S>, con P composto da:

❶ $S\to aS$ ❷ $S\to B$ ❸ $B\to bB$
❹ $B\to bC$ ❺ $C\to cC$ ❻ $C\to c$

genera $L(G) = \{a^nb^mc^h \mid n\geq0,\ m,\ h\geq1\}$

forma sintetica
$\alpha\to\beta_1 \quad \alpha\to\beta_2 \quad \ldots \quad \alpha\to\beta_n$
viene anche indicato con
$\alpha\to\beta_1 \mid \beta_2 \mid\ldots\mid \beta_n$

47

47

Il processo generativo parte dall'assioma, applico le produzioni, la stringa cambia aspetto fino a divenire una stringa di soli terminali. Se così diventa solo di terminali, allora appartiene alla grammatica G.
L(G) --> "questo è il linguaggio generato dalla grammatica G".

Espressione regolare che descriva esattamente il linguaggio L(G):
a*bb*cc*

Processo generativo: parto da S(assioma) e faccio susseguire le produzioni:
S--> aS (oppure -->B) ---> aB--->abB-->abbC---> abbc
punto di arrivo: una stringa composta da soli terminali--> è una stringa del linguaggio.
abbc --> composta da soli terminali, se la voglio modificare devo trovare una produzione da applicare. A sx c'è obbligatoriamente un non terminale.
Con la grammatica di Chomsky alla fine quando ho trovato una stringa di terminali, essa non può essere modificata ulteriormente.

## Linguaggio generato

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

- *derivazione diretta*: relazione su $(V^*\circ V_N\circ V^*) \times V^*$

$<\varphi,\psi>$ 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 $\alpha\to\beta\in P$

$\varphi$ 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 $\Rightarrow^*$

48

48

Linguaggio generato.
Derivazione diretta--> relazione binaria in cui ho due parti.

Passo da una stringa composta da terminali e non terminali ad un'altra se una produzione me lo consente.
nel processo generativo troviamo le stringhe intermedie che sono chiamate le "forme di frase".
Posso scrivere S-->* abbc (la freccia con l'asterisco mi contiene tutte le varie stringhe intermedie).Posso scrivere anche se sto in un passaggio intermedio: as -->*abbC
Due grammatiche sono equivalenti se i loro linguaggi sono equivalenti tra loro.
Dato un linguaggio esso può essere generato da più grammatiche equivalenti.





## Grammatiche formali

esempio: generazione di $\{a^nb^nc^n \mid n\geq1\}$

grammatica G=<{a, b, c},{S,B,C,F,G},P,S>

con P composto da

❶ $S\to aSBC$ ❷ $CB\to BC$
❸ $SB\to bF$ ❹ $FB\to bF$
❺ $FC\to cG$ ❻ $GC\to cG$
❼ $G\to\varepsilon$

per generare aabbcc

$S \Rightarrow^{❶} aSBC \Rightarrow^{❶} aaSBCBC \Rightarrow^{❷} aaSBBCC$
$\Rightarrow^{❸} aabFBCC \Rightarrow^{❹} aabbFCC \Rightarrow^{❺} aabbcGC$
$\Rightarrow^{❻} aabbccG \Rightarrow^{❼} aabbcc$

49

49

a, b, c devono essere generata dello stesso numero. Quale è l'espressione regolare che descrive questo linguaggio? No, non esistono espressioni regolari che lo generino.

S-->aSBC-->aaSBCBC-->aaSBBCC--> ..... aabbcc oppure aaabbbccc ....

Se le produzioni le applico in una certa sequenza genero le stringhe del linguaggio; ma se le uso in modo sbagliato trovo forme di frase dalle quali il processo generativo non può continuare.
Ma ciò non mi dice che la grammatica sia sbagliata, essa non mi da garanzia che applicando le produzioni alla rinfusa arrivo ad una stringa del linguaggio.
Queste grammatiche sono fatte di produzioni che devo usare con intelligenza e usarle attentamente per generare le stringhe del linguaggio stesso.

Le stringhe che non hanno solo terminali non fanno parte del linguaggio e quindi se le genero non fa niente perché non gli appartengono.
Esempio di grammatica che non produce stringhe, ovvero che produce il linguaggio vuoto:
S->aS

## Grammatiche formali

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

esempio:

la grammatica G=<{a, b, c}, {S, A}, P, S> con

$S\to aSc \mid A$
$A\to bAc \mid \varepsilon$

genera $\{a^nb^mc^{n+m} \mid n,m\geq0\}$

Parto dall'assioma e genero una a a dx e una c a sx e ne ho tante a sx quante a destra. Ogni volta che applico la prima produzione, nella forma di frase che costruisco ho sempre 1 terminale.

$S\Rightarrow aSc \Rightarrow aaScc \Rightarrow aaAcc \Rightarrow aabAccc \Rightarrow$

esempio:
la grammatica G=<{a, b, c}, {S, A}, P, S> con
$S \to Ab$
$A \to Sa$
genera Λ

50

50

$\Rightarrow \underbrace{aa}_{2}\underbrace{b}_{+1}\underbrace{ccc}_{=3}$

Da S-->* aabccc (derivazione transitiva e riflessiva a partire dall'assioma).
Questa è una grammatica che ha sempre un modo per passare ad una forma di frase alla successiva finchè non arrivo ad una stringa di soli terminali.
Secondo esempio: da qui non genererò mai nessuna stringa, qualsiasi cosa faccio genero il linguaggio vuoto.





## 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

51

51

## Grammatiche di Chomsky di tipo 0, non limitate

- sono le meno restrittive
- produzioni del tipo

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

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
$S \to aAB$ $\quad B \to b$
$aA \to aaAb$ $\quad aAb \to ab$
$aAA \to aA$

52

52

Tipo 0--> le meno restrittive: hanno produzioni con a sx come vuoi basta che ci sia un non terminale e a dx quello che ti pare.
Oss: queste grammatiche ammettono anche derivazioni che accorciano le forme di frase:
ABC-->CD: la forma di frase si è accorciata

Questa grammatica ammette produzioni in grado di accorciare forme di frase.
L'ultima produzione non la userò mai perché a partire dall'assioma non troverò mai due AA insieme. Infatti non è detto che tutte le produzioni sono utili. Posso trovare produzioni che però non userò.
Queste produzioni generano quel linguaggio? si, è un linguaggio che si può generare dalla grammatica di tipo 0--> ovvero il linguaggio di tipo 0.





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

- produzioni che non riducano la lunghezza delle forme di frase

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

linguaggi di tipo 1
esempio:
il linguaggio $\{a^n b^n c^n | n \geq 1\}$ è di tipo 0 in quanto generato da
$S \to aSBC$ $\quad CB \to BC$
$SB \to bF$ $\quad FB \to bF$
$FC \to cG$ $\quad GC \to cG$
$cG \to c$
ma è anche di tipo 1, infatti è generato anche da
$S \to aSBc \mid aBc$
$cB \to Bc$
$bB \to bb$
$aB \to ab$

53

53

Tipo 1: qui le produzioni hanno a sx e a dx come prima ma la cardinalità di sx deve essere minore o uguale di quella di dx. le forme di frase NON possono essere accorciate, al massimo rimangono della stessa lunghezza.

Perché "contestuali"? Perché rimpiazzo non un pezzettino ma un contesto, una sequenza di terminali o non, delle forme di frase.

Un linguaggio può essere generato da due distinte grammatiche di tipo diverso.
Un linguaggio di tipo 0 non è detto che sia di tipo 1
Mentre se è di tipo 1 è anche di tipo 0.

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

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

$$\begin{aligned} S &\Rightarrow aSBc \\ &\Rightarrow aaSBcBc \\ &\Rightarrow aaaSBcBcBc \\ &\Rightarrow aaaaBcBcBcBc \\ &\Rightarrow aaaaBcBcBBcc \\ &\Rightarrow aaaaBcBBcBcc \\ &\Rightarrow aaaaBcBBBccc \\ &\Rightarrow aaaaBBcBBccc \\ &\Rightarrow aaaaBBBcBccc \\ &\Rightarrow aaaaBBBBcccc \\ &\Rightarrow aaaabBBBcccc \\ &\Rightarrow aaaabbBBcccc \\ &\Rightarrow aaaabbbBcccc \\ &\Rightarrow aaaabbbbcccc \end{aligned}$$

54

54





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

- produzioni del tipo

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

linguaggi di tipo 2
esempio:
il linguaggio $\{a^n b^n | n \geq 1\}$ è di tipo 0 in quanto generato da
$S \to aAb$
$aA \to aaAb$
$A \to \varepsilon$
ma è anche di tipo 2, infatti è generato anche da
$S \to aSb \mid ab$

55

55

Tipo 2: grammatiche dei linguaggi di programmazione. Caratteristica: a dx le produzioni possono avere qualunque cosa, ma a sx solo 1 simbolo non terminale.
V+ --> vuol dire che a dx qualcosa ci deve essere, non ci sta la stringa vuota.
Dato un linguaggio se c'è una grammatica di tipo 2 che lo genera allora è di tipo 2, e anche 1 e 0.

## 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.
$E \to E+T \mid T$
$T \to T*F \mid F$
$F \to i \mid (E)$

grammatica delle parentesi ben bilanciate (esempio "((()))) ()")
$S \to () \mid SS \mid (S)$
da quale sequenza di produzioni è generata "()((()()))" ?

grammatica delle stringhe palindrome (esempio "elle", "ereggere")

56

56

Esempi:
Il linguaggio delle espressioni aritmetiche è di tipo 2.

S--> SS--> S(S)--> S((S))-->S((SS)) --> ()((SS))--> ()((()()))

Come si generano? Quando genero, lo faccio al centro e se devo generare qualunque lettera lo faccio sia a dx che sx





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

- produzioni del tipo

$$A \to \delta,\ A \in V_N,\ \delta \in (V_T \circ V_N) \cup V_T$$

- linguaggi di tipo 3

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

Tipo 3:
Le più semplici:
A sx un solo non terminale e a dx o terminale o  terminale seguito da non terminale.
Sono le grammatiche più restrittive, chiamate regolari

$S \rightarrow b$

si possono anche definire grammatiche lineari sinistre (LL) con

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

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

$S \rightarrow Tb \mid b$

$T \rightarrow a \mid Ta$

teorema: i linguaggi generati da grammatiche LL e RL coincidono

57

Teorema: non ce le frega nu cazz

57

## 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

58

58

Quindi: dato che ciò è vero--> questo è un linguaggio strettamente di tipo 2

ESERCIZIO

Sia G una grammatica di tipo 0 e sia x una forma di frase derivabile da S. Sicuramente non esiste in G una produzione che, applicata a x, consenta di derivare da x una forma di frase y tale che |x|<|y|. -->FALSO

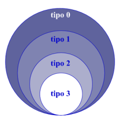
Considera una grammatica G. Supponi che usando G, a partire dall'assioma S, sia possibile derivare una forma di frase x composta da soli terminali. Ne segue che x non appartiene al linguaggio L(G). --> FALSO

In una grammatica di tipo 3 nella parte sinistra di ogni produzione c'è sempre un solo non terminale -->VERO





## Grammatiche di Chomsky




contenimento tra i linguaggi

59

59

Dato un linguaggio c'è sempre una grammatica di tipo 0 che lo genera? Non si sa

Qui dentro in ogni parte c'è un linguaggio, un insieme di stringhe.

Se è di tipo 1 è anche di tipo 0

## Grammatiche di Chomsky

| tipo | produzioni | vincoli |
|---|---|---|
| tipo 0<br>non limitate | $\alpha \rightarrow \beta$ | $\alpha \in V^* \circ V_N \circ V^*$, $\beta \in V^*$ |
| tipo 1<br>contestuali | $\alpha \rightarrow \beta$ | $\lvert\alpha\rvert \leq \lvert\beta\rvert$<br>$\alpha \in V^* \circ V_N \circ V^*$, $\beta \in V^*$ |
| tipo 2<br>non contestuali | $A \rightarrow \gamma$ | $A \in V_N$, $\gamma \in V^+$ |
| tipo 3<br>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

60

60





## ε-produzioni

- con grammatiche di tipo 1, 2, 3 non è possibile generare la stringa vuota ε
  - per generare ε occorre una produzione $\alpha \rightarrow \varepsilon$ che viene detta ε-produzione
  - per Chomsky tutti i linguaggi che contengono ε-produzioni sono linguaggi di tipo 0
- qual è l'impatto sui corrispondenti linguaggi delle ε-produzioni nelle grammatiche?
  - se ammettiamo ε-produzioni dobbiamo fare attenzione, altrimenti rischiamo di snaturare la gerarchia di Chomsky

61

61

## ε-produzioni: variazione della gerarchia

con le seguenti modifiche, i linguaggi generati dale diverse tipologie di grammatiche rimangono inalterati, salvo per la possibilità di generare la stringa vuota

| tipo | ε-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 |

62

62





## Esempi di grammatiche

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

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

- 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

63

63

abbabb appartiene al linguaggio

## Esempi di grammatiche

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

  è generato dalla grammatica contestuale:

  $S \rightarrow S' \mid \varepsilon$ $\quad AB \rightarrow BA$ $\quad A \rightarrow a$
  $S' \rightarrow ABC\#$ $\quad AC \rightarrow CA$ $\quad B \rightarrow b$
  $S' \rightarrow ABC\#S'$ $\quad BC \rightarrow CB$ $\quad C \rightarrow c$
- 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:

  $S \rightarrow aX \mid bY \mid cZ \mid \varepsilon$ $\quad R \rightarrow \#S$

  $X \rightarrow bX' \mid cX''$ $\quad Y \rightarrow aY' \mid cY''$ $\quad Z \rightarrow aZ' \mid bZ''$

$X'' \to bR$ $Y'' \to aR$ $Z'' \to aR$ 64

64





## 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:

$Q \to I\,;\ |\ I\,;Q$

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

può essere riscritto:

$F \to \texttt{if}\,(\,C\,)\,I\ \texttt{else}\ I\ |\ \texttt{if}\,(\,C\,)\,I$

65

65

## Riconoscimento dei linguaggi

problema: stabilire se una stringa appartiene a 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)

66

66





## 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

67

67