# linguaggi non contestuali

# di tipo 2

# context free

# CF



## linguaggi non contestuali

- molte frasi in linguaggio naturale hanno una struttura sintattica non contestuale
  - esempio: soggetto – verbo – complemento
- la BNF non è altro che una grammatica di tipo 2
- hanno proprietà di riconoscibilità abbastanza semplici e la loro analisi sintattica può essere eseguita in modo efficiente
- corrispondenza tra le derivazioni e gli alberi (alberi sintattici)

# quattro passi nei linguaggi non contestuali

proprietà

forme normali

automi a pila

ambiguità

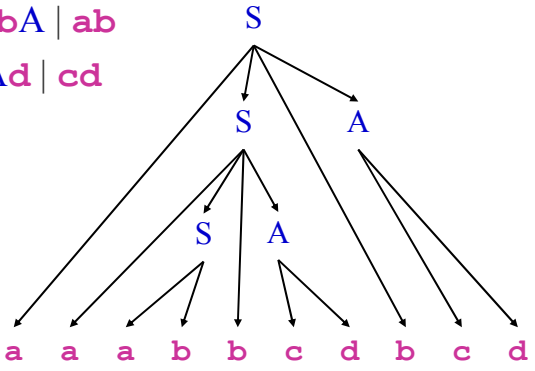


# albero sintattico

data la grammatica

S → **a**S**b**A | **ab**

A → **c**A**d** | **cd**




derivazione della stringa **aaabbcdbcd**

## proprietà dei linguaggi non contestuali

lemma:

data una grammatica non contestuale possiamo decidere se essa genera il linguaggio vuoto



## proprietà dei linguaggi non contestuali

lemma:

data una grammatica non contestuale possiamo decidere se essa genera il linguaggio vuoto

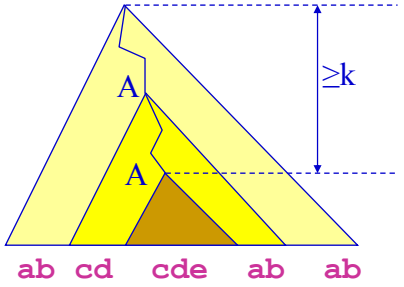
dimostrazione:

supponiamo $|V_N|=k$

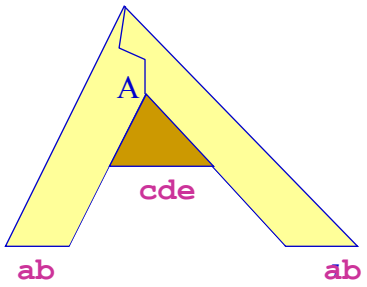
per verificare la producibilità di una stringa di soli terminali è sufficiente considerare l'insieme di tutti gli alberi di derivazione con tutti i terminali a profondità $\leq k$

# dimostrazione – linguaggio vuoto

infatti, se una stringa di soli terminali è generata con almeno un terminale **y** a profondità > k, allora sul cammino da **y** alla radice c'è un nonterminale A ripetuto due volte




l'albero può quindi essere "potato", e dopo la potatura produce ancora una stringa di soli terminali




# grammatica in forma ridotta

1. non contiene ε-produzioni, salvo (eventualmente) sull'assioma
2. se l'assioma contiene ε-produzioni, allora non compare mai a destra in una produzione
3. non contiene simboli inutili (inutilizzabili per produrre terminali)
4. non contiene produzioni unitarie

   esempio: A → B

# teorema – forma ridotta

teorema:

data una grammatica non contestuale esiste una grammatica equivalente in forma ridotta



# teorema – forma ridotta

teorema:

data una grammatica non contestuale esiste una grammatica equivalente in forma ridotta

dimostrazione:

1. e 2. rimozione delle ε-produzioni che non sono sull'assioma:
   tecnica mostrata in precedenza

3. eliminazione dei simboli inutili:
   perché un simbolo A sia utile è necessario
   (a) che da A siano generabili stringhe di terminali
   (b) che A sia generabile da S in produzioni senza simboli inutili

la condizione (a) è verificabile con il lemma precedente, trattando A come assioma

la condizione (b) è verificabile iterativamente da S

# teorema – forma ridotta

4. eliminazione delle produzioni unitarie:

se esiste una derivazione $U \Rightarrow V$
allora per ogni $V \rightarrow \alpha$ aggiungiamo $U \rightarrow \alpha$ e quindi
eliminiamo le produzioni unitarie



# ordine delle semplificazioni

attenzione ad applicare le semplificazioni nell'ordine giusto
esempio: riduciamo la grammatica

$S \rightarrow AB|a$
$A \rightarrow a$

eliminazione dei simboli non derivabili

$S \rightarrow AB|a$
$A \rightarrow a$

eliminazione dei simboli non fecondi

$S \rightarrow a$
$A \rightarrow a$ (ora occorrerebbe di nuovo eliminare i non derivabili)

strategia: eliminiamo prima i non fecondi e poi i non derivabili

$S \rightarrow a$
$A \rightarrow a$

e quindi

$S \rightarrow a$

# esempio – forma ridotta

riduciamo la grammatica

| | |
|---|---|
| S → aUVb\|TZ | V → aY |
| Z → aZ | Y → bY\|b |
| U → bU\|b | W → cWd\|cd |
| V → W | T → tT\|tz |

eliminazione dei simboli non fecondi

| | |
|---|---|
| S → aUVb~~\|TZ~~ | V → aY |
| ~~Z → aZ~~ | Y → bY\|b |
| U → bU\|b | W → cWd\|cd |
| V → W | T → tT\|tz |

eliminazione dei simboli non derivabili

| | |
|---|---|
| S → aUVb | V → aY |
| U → bU\|b | Y → bY\|b |
| V → W | W → cWd\|cd |
| | ~~T → tT\|tz~~ |

eliminazione delle produzioni unitarie

| | |
|---|---|
| S → aUVb\|aUWb | V → aY |
| U → bU\|b | Y → bY\|b |
| ~~V → W~~ | W → cWd\|cd |



# linguaggi non contestuali infiniti

teorema:

data una grammatica non contestuale G, è decidibile stabilire se L(G) è infinito

# linguaggi non contestuali infiniti

teorema:

data una grammatica non contestuale G, è decidibile stabilire se L(G) è infinito

dimostrazione:

basta considerare l'insieme di tutte le derivazioni che generano stringhe terminali in cui il ramo più lungo ha lunghezza tra k e 2k

se tale insieme è vuoto, allora il linguaggio è finito

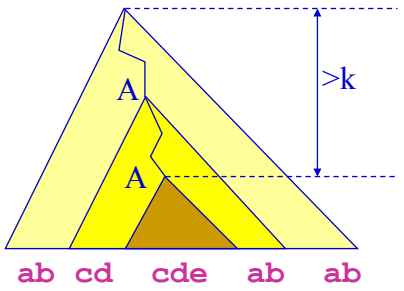
se no, tali derivazioni hanno almeno un non terminale ripetuto due volte

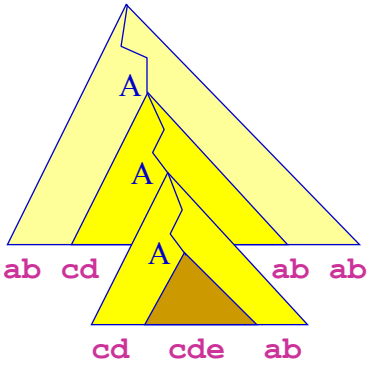
allora l'albero di derivazione può essere ampliato indefinitivamente continuando a produrre stringhe terminali

allora il linguaggio è infinito



# dimostrazione – linguaggio infinito



# pumping lemma – non contestuali

teorema:

se L è un linguaggio non contestuale allora esiste una costante n tale che se $z \in L$ e $|z| \geq n$ allora esistono u,v,w,x,y tali che

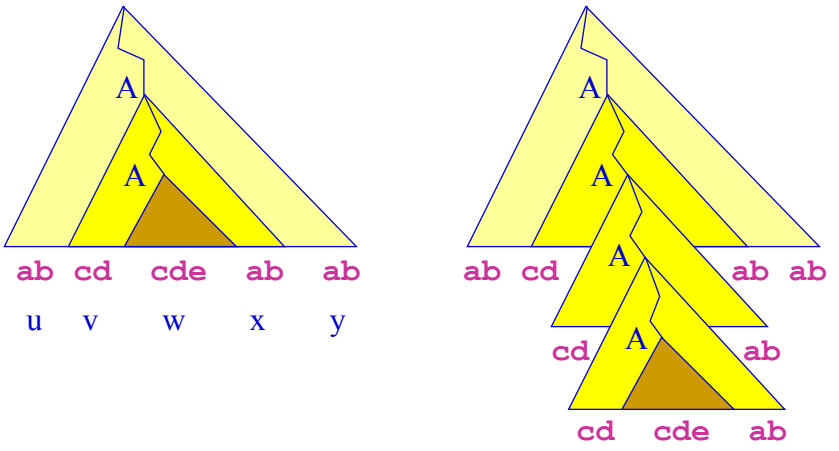
1. $uvwxy=z$
2. $|vx| \geq 1$
3. $|vwx| \leq n$
4. $\forall\ i \geq 0\ uv^iwx^iy \in L$

dimostrazione:

analoga a quella dei teoremi precedenti



# dimostrazione – pumping lemma




generazione della stringa $uv^iwx^iy$




# il linguaggio $a^nb^nc^n$

corollario del pumping lemma per i linguaggi non contestuali: $\{a^nb^nc^n|n\geq 1\}$ non è di tipo 2

dimostrazione:

sia $a^nb^nc^n=uvwxy=z$

se v (oppure x) contiene almeno due simboli diversi (es. v=**ab**)

allora la stringa $uv^2wx^2y$ contiene un'alternanza di simboli incompatibile con $a^nb^nc^n$ (es. u**abab**w$x^2$y)

se v è composta da simboli tutti uguali fra loro (es. v=**aa**) e x è composta da simboli tutti uguali fra loro (es. v=**cc**)

allora $uv^2wx^2y$ contiene un numero diverso di a, b e c

# intersezione di linguaggi non contestuali

corollario del pumping lemma per i linguaggi non contestuali: i linguaggi non contestuali non sono chiusi rispetto all'intersezione

dimostrazione:

- il linguaggio $L_1=\{a^n b^n c^m | m,n \geq 1\}$ è di tipo 2
- il linguaggio $L_2=\{a^m b^n c^n | m,n \geq 1\}$ è di tipo 2
- il linguaggio intersezione di $L_1$ ed $L_2$ è $\{a^n b^n c^n | n \geq 1\}$ che non è di tipo 2

osservazione: la dimostrazione della decidibilità dell'equivalenza tra linguaggi regolari si fonda sulla chiusura dell'intersezione

tale tecnica non è estendibile ai linguaggi non contestuali



# chiusura dei linguaggi non contestuali

teorema:

i linguaggi non contestuali sono chiusi rispetto all'unione, alla concatenazione ed alla iterazione

# chiusura dei linguaggi non contestuali

teorema:

i linguaggi non contestuali sono chiusi rispetto all'unione, alla concatenazione ed alla iterazione

dimostrazione:

siano $<\Sigma,V_{N1},P_1,S_1>$ e $<\Sigma,V_{N2},P_2,S_2>$ due linguaggi non contestuali

- unione

  aggiungiamo $S' \rightarrow S_1|S_2$ e usiamo $S'$ come nuovo assioma, otteniamo un linguaggio non contestuale che genera l'unione dei due linguaggi

- concatenazione

  aggiungiamo $S' \rightarrow S_1S_2$ e usiamo $S'$ come nuovo assioma

- iterazione

  aggiungiamo $S' \rightarrow S_1S'|\varepsilon$ e usiamo $S'$ come nuovo assioma



# forma normale di chomsky

una grammatica di tipo 2 è in forma normale di chomsky (*Chomsky Normal Form*, CNF) se tutte le produzioni sono del tipo

$$A \rightarrow BC \text{ oppure } A \rightarrow a$$

teorema:

data una grammatica G di tipo 2 tale che $\varepsilon \notin L(G)$ esiste una grammatica $G'$ in CNF con $L(G)=L(G')$

# forma normale di chomsky

teorema:

data una grammatica G di tipo 2 tale che $\varepsilon \notin L(G)$
esiste una grammatica G′ in CNF con L(G)=L(G′)

dimostrazione:

portiamo la grammatica in forma ridotta

per ogni produzione p: $A \rightarrow \beta_1\beta_2\ldots\beta_n$
se p non è in CNF sono possibili due casi

(caso 1) per ogni i $\beta_i \in V_N$ $\{1,\ldots,n\}$ $(n\geq 3)$

(caso 2) per qualche i $\beta_i \in V_T$ $\{1,\ldots,n\}$ $(n\geq 2)$



# dimostrazione – grammatica → CNF

(caso 1) per ogni i $\beta_i \in V_N$ $\{1,\ldots,n\}$ $(n\geq 3)$

aggiungiamo n-2 nuovi non terminali $Z_1,\ldots,Z_{n-2}$

e sostituiamo p: $A \rightarrow \beta_1\beta_2\ldots\beta_n$ con una catena di produzioni

$A \rightarrow \beta_1 Z_1$
$Z_1 \rightarrow \beta_2 Z_2$
$Z_2 \rightarrow \beta_3 Z_3$
…
$Z_{n-2} \rightarrow \beta_{n-1}\beta_n$

(caso 2) per qualche i $\beta_i \in V_T$ $\{1,\ldots,n\}$ $(n\geq 2)$

per ogni $\beta_i \in V_T$ aggiungiamo un non terminale $\underline{Z_i}$, sostituiamo $\underline{Z_i}$ a $\beta_i$ in p ed aggiungiamo la produzione $\underline{Z_i} \rightarrow \beta_i$

se n=2, allora non è necessario lavorare ulteriormente sulla produzione, altrimenti ci riconduciamo al (caso 1)

# esempio

trasformazione in CNF della grammatica

S→**a**S**b**

S→**a**b

sostituiamo S→ab con

S→AB

A→**a**

B→**b**

sostituiamo S→**a**S**b** con

S→ASB

sostituiamo S→ASB con

S→AZ

Z→SB



# forma normale di greibach

una grammatica di tipo 2 è in *forma normale di greibach* (*Greibach Normal Form*, GNF) se tutte le produzioni sono del tipo A→aβ con $\beta \in V_N^*$

la forma normale di Greibach ha applicazioni all'analisi sintattica dei linguaggi

# forme normali per linguaggi di tipo 2

lemma (sostituzione):

data una grammatica G le cui produzioni includono

$A \rightarrow \alpha_1 B \alpha_2$ $(\alpha_1, \alpha_2 \in V^*)$

$B \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$

ed in cui B non appare al primo membro in nessun'altra produzione

la grammatica G′ in cui la produzione

$A \rightarrow \alpha_1 B \alpha_2$ è sostituita con

$A \rightarrow \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \ldots | \alpha_1 \beta_n \alpha_2$

è equivalente a G

dimostrazione: banale



# forme normali per linguaggi di tipo 2

lemma (eliminazione ricorsione sinistra):

data una grammatica G le cui produzioni includono

$A \rightarrow A\alpha_1 | \ldots | A\alpha_m | \beta_1 | \ldots | \beta_n$

ed in cui A non appare al primo membro in nessun'altra produzione e nessuna delle $\beta_i$ inizia con A

la grammatica G′ in cui la produzione

$A \rightarrow A\alpha_1 | \ldots | A\alpha_m | \beta_1 | \ldots | \beta_n$ è sostituita con

$A \rightarrow \beta_1 B | \ldots | \beta_n B | \beta_n | \ldots | \beta_n$ e

$B \rightarrow \alpha_1 B | \ldots | \alpha_m B | \alpha_1 | \ldots | \alpha_m$

è equivalente a G

## dimostrazione – elim. ricorsione sinistra

ogni derivazione in G del tipo

$A \Rightarrow A\alpha_1$
$\Rightarrow A\alpha_2\alpha_1$
…
$\Rightarrow A\alpha_m \ldots \alpha_2\alpha_1$
$\Rightarrow \beta_1\alpha_m \ldots \alpha_2\alpha_1$

è rimpiazzata in G′ da

$A \Rightarrow \beta_j B$
$\Rightarrow \beta_j\alpha_m B$
$\Rightarrow \beta_j\alpha_m\alpha_{m-1} B$
…
$\Rightarrow \beta_j\alpha_m \ldots \alpha_2 B$
$\Rightarrow \beta_j\alpha_m \ldots \alpha_2\alpha_1$

è possibile effettuare anche il passaggio inverso



## teorema – grammatiche tipo 2 e GNF

teorema:

data una grammatica G di tipo 2 tale che $\varepsilon \notin L(G)$ esiste una grammatica G′ in GNF con $L(G)=L(G')$

# teorema – grammatiche tipo 2 e GNF

teorema:

data una grammatica G di tipo 2 tale che $\varepsilon \notin L(G)$
esiste una grammatica G′ in GNF con $L(G)=L(G')$

dimostrazione: 

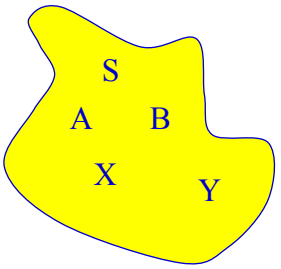
strategia generale:

- supponiamo che G sia in CNF (tutte le produzioni sono del tipo $A \rightarrow BC$ oppure $A \rightarrow a$)
- stabiliamo arbitrariamente un ordinamento $A_1,\ldots,A_m$ sui non terminali di G
- eseguiamo, varie volte, le operazioni di sostituzione ed eliminazione della ricorsione sinistra



# dimostrazione – grammatica → GNF

- stabiliamo arbitrariamente un ordinamento $A_1,\ldots,A_m$ sui non terminali di G




noi siamo i non terminali di G

S
A
B
Y
X

ora ci ordiniamo

# dimostrazione – grammatica → GNF

passo 1

- modifichiamo le produzioni in modo tale che se $A_i \rightarrow A_j\gamma$ è una produzione, allora $i<j$
- assumiamo che ciò sia stato già fatto per le produzioni $A_i \rightarrow A_j\gamma$ con $1 \leq i < k$
- quindi modifichiamo le $A_k$-produzioni
- se $A_k \rightarrow A_j\gamma$ è una produzione con $j<k$ usiamo l'operazione di sostituzione rimpiazzando $A_j$ con le produzioni $A_j \rightarrow \beta$
- ripetendo l'operazione al più k-1 volte otteniamo produzioni nella forma $A_k \rightarrow A_p\gamma$ $p \geq k$
- se $p=k$ applichiamo l'eliminazione della ricorsione a sinistra introducendo un nuovo terminale $B_k$



# passo 1 – algoritmo

```
for (k=1; k≤m; k++) {
  for(j=1; j<k; j++) {
    per ogni produzione del tipo Ak→Ajα {
      per ogni Aj→β
        aggiungi Ak→βα e rimuovi Ak→Ajα
    }
    per ogni produzione del tipo Ak→Akα {
      aggiungi Bk→α e Bk→αBk
      rimuovi Ak→Akα
    }
    per ogni produzione del tipo Ak→β {
      // dove β non inizia con Ak
      aggiungi Ak→βBk
    }
  }
}
```



# considerazioni sul passo 1

- ripetendo il procedimento abbiamo solo produzioni del tipo:
  1. $A_i \rightarrow A_j\gamma$, con $j>i$
  2. $A_i \rightarrow \mathbf{a}\gamma$, con $\mathbf{a} \in V_T$
  3. $B_i \rightarrow \gamma$, con $\gamma \in (V_N \cup \{B_1,\ldots,B_q\})^+$

# dimostrazione – grammatica → GNF

passo 2

al termine del passo 1

- il simbolo più a sinistra della parte destra di ogni $A_m$-produzione è un terminale, infatti m è il numero più alto
- il simbolo più a sinistra della parte destra di ogni $A_{m-1}$-produzione è un terminale oppure è $A_m$
  - se è $A_m$ allora possiamo usare l'operazione di sostituzione e rimpiazzarlo con le $A_m$-produzioni

ripetiamo lo stesso procedimento per ogni $A_i$ a ritroso fino a $A_1$



# dimostrazione – grammatica → GNF

passo 3

- esaminiamo le produzioni dei nuovi non terminali $B_i$
- il fatto che G sia in CNF ci garantisce che ogni produzione abbia a destra o un terminale o due non terminali
- quindi, quando effettuiamo il passo:

  per ogni produzione del tipo $A_k \rightarrow A_k\alpha$

  aggiungi $B_k \rightarrow \alpha$ e $B_k \rightarrow \alpha B_k$

  $\alpha$ non è mai vuoto e non può iniziare con un $B_i$
- quindi, per far comparire un terminale all'inizio della parte destra, possiamo riapplicare alle $B_i$-produzioni lo stesso algoritmo applicato alle $A_i$-produzioni

# esempio

data la seguente grammatica in CNF, calcolarne una equivalente in GNF

S→AB
S→b
A→b
A→BS
B→a
B→BA
B→AS

ordinamento: S,A,B

sostituiamo B→AS con B→bS e B→BSS

ora l'insieme delle B-produzioni è il seguente

B→a|bS|BA|BSS



# esempio – CNF → GNF

eliminiamo la ricorsione a sinistra con B′

B→a|bS|aB′|bSB′
B′→A|SS|AB′|SSB′

giungiamo quindi alla grammatica

S→AB
S→b
A→b
A→BS
B→a|bS|aB′|bSB′
B′→A|SS|AB′|SSB′

che rispetta l'ordinamento imposto dal passo 1

# esempio – CNF → GNF

ora possiamo sostituire (passo 2)

A→BS

con

A→aS|bSS|aB′S|bSB′S

quindi sostituiamo

S→AB

con

S→aSB|bSSB|aB′SB|bSB′SB|bB

ora possiamo sostituire (passo 3)

B'→A|SS|AB′|SSB′

con

B'→aS|bSS|aB′S|bSB′S|b|
aSBS|bSSBS|aB′SBS|bSB′SBS|bBS|bS|
aSB′|bSSB′|aB′SB′|bSB′SB′|bB′|
aSBSB′|bSSBSB′|aB′SBSB′|bSB′SBSB′|
bBSB′|bSB′