

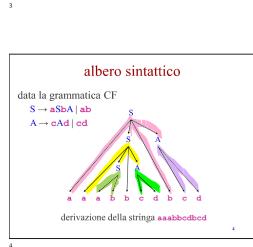
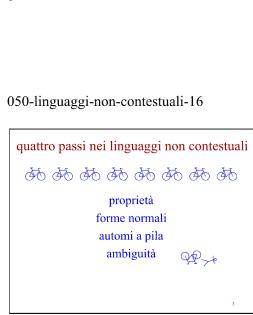
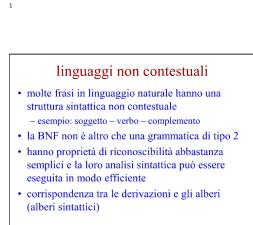
## 050-linguaggi-non-contestuali-16



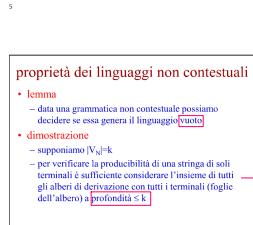
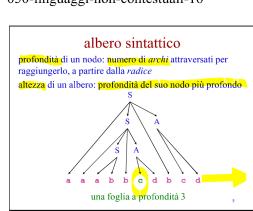
23/10

Linguaggi  
non...

Registrazione audio avviata: 09:40 mercoledì 23 ottobre 2024



## 050-linguaggi-non-contestuali-16



Proprietà dei linguaggi regolari:

- Chiusura (es: rispetto l'unione)
- È possibile stabilire se un linguaggio regolare è vuoto, infinito o finito

I CF sono chiusi rispetto l'unione? Sì

Per produrre stringhe che fanno parte di 2 linguaggi CF (unione) posso costruire un nuovo assioma  $S'$  che va in  $S$  e  $S' \rightarrow$  otengo ancora una grammatica CF. $S \rightarrow S' \mid S''$ 

Usa gli alberi per vedere se un linguaggio CF è infinito, vuoto, finito. (VEDI SOTTO)

Es: profondità 2

una foglia a profondità 3

## Dimostrazione:

## 1. Profondità degli alberi di derivazione:

- Consideriamo una grammatica con  $|V_N| = k$ , dove  $V_N$  è l'insieme dei simboli non terminali.
- Per verificare se il linguaggio generato dalla grammatica è vuoto, è sufficiente considerare gli alberi di derivazione con profondità  $\leq k$ .
- Se nessuno di questi alberi produce una stringa composta esclusivamente da terminali, il linguaggio è vuoto.

2. Perché la profondità  $k$  è sufficiente?

- Se un albero di derivazione genera una stringa terminale e ha profondità maggiore di  $k$ , esiste almeno un simbolo non terminale  $A$  che viene ripetuto lungo il cammino tra la radice e una foglia.
- Questa ripetizione permette di "potare" l'albero, eliminando il ciclo relativo a  $A$ , e ottenere un albero equivalente che genera la stessa stringa di terminali ma con una profondità minore o uguale a  $k$ .

## 3. Concetto di "potatura":

- Come mostrato nella seconda immagine, se un simbolo terminale  $y$  si trova a profondità  $> k$ , allora lungo il cammino dalla radice a  $y$  c'è un simbolo non terminale  $A$  ripetuto almeno due volte (a causa del principio del **pigeonhole**).
- Eliminando il sottografo che si ripete e mantenendo solo una copia del ciclo, possiamo produrre la stessa stringa con un albero di derivazione più "compatto".

## 4. Conclusioni:

- È sufficiente esaminare gli alberi con profondità  $\leq k$  per determinare se una stringa di terminali è produttibile.
- Se il simbolo iniziale  $S$  non riesce a generare alcuna stringa terminale entro questa profondità, allora il linguaggio generato dalla grammatica è vuoto.

## Sintesi del procedimento:

- L'algoritmo si basa sul fatto che, per una grammatica con  $|V_T| = k$ , ogni albero di derivazione può essere "ridotto" a uno di profondità  $\leq k$  senza perdere la capacità di generare stringhe terminali.
- Se nessun albero di profondità  $\leq k$  genera stringhe terminali, allora il linguaggio è vuoto.

Questo dimostra che il problema è decidibile e fornisce un criterio pratico per verificarlo.

## 050-linguaggi-non-contestuali-16



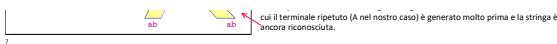
Pigeon principle

io so che il numero di non terminali = k  
Se c'è un terminale a profondità > k → lungo la strada un non terminale viene ripetuto 2 volte.

Allora questo albero è già stato ripotato, e genera un albero che ancora genera una stringa nel linguaggio.

Se c'è un terminale abbastanza profondo, vuol dire che per generarlo, a partire dalla radice, ho fatto un cammino molto lungo e per farlo un non terminale deve essere ripetuto più volte. A è ripetuto più volte (vedi figura)

Allora questo albero è anche in grado di generare un altro albero di derivazione in



**grammatica in forma ridotta**

- non contiene  $\epsilon$ -produzioni, salvo (eventualmente) sull'assiomà
- se l'assiomà contiene  $\epsilon$ -produzioni, allora non compare mai a destra in una produzione
- non contiene simboli inutili
  - cioè inutilizzabili per produrre terminali
- non contiene produzioni unitarie
  - esempio di produzione unitaria:  $A \rightarrow B$

- Per fare questi ragionamenti devo **creare una grammatica in forma ridotta** (4 proprietà)
- Restringo le epsilon produzioni al suo assiomà
  - Come per le grammatiche di tipo 1
  - Non terminali non raggiungibili
  - Da non terminale à non terminale

8 \*

4

19/23/2024

## 050-linguaggi-non-contestuali-16

**teorema – forma ridotta**

**teorema:** data una grammatica non contestuale esiste e si può costruire una grammatica equivalente in forma ridotta

2 grammatiche equivalenti: se generano lo stesso linguaggio

d'ora in poi considereremo solo grammatiche Context Free in forma ridotta

9 \*

**linguaggi non contestuali infiniti**

**teorema:** data una grammatica non contestuale  $G$ , è decidibile stabilire se  $L(G)$  è infinito

**dimostrazione:** consideriamo l'insieme di tutti gli alberi sintattici che generano stringhe terminali in cui la foglia con maggiore profondità ha profondità tra  $k$  e  $2k$ , con  $k$  uguale al numero dei non terminali

Se tale insieme è vuoto, allora il linguaggio è finito

altrimenti, se ne scelgono uno a caso, se è un terminali ripetuto 2 volte e l'albero di derivazione può essere ampliato indefinitamente, continuando a produrre stringhe di terminali; il linguaggio è infinito

10 \*

5

19/23/2024

## 050-linguaggi-non-contestuali-16

**dimostrazione – linguaggio infinito**

Considero gli alberi tra  $k$  e  $2k$ . Se c'è una stringa la dentro (tra gli alberi tra  $k$  e  $2k$ ), vuol dire che c'è un non terminale ripetuto almeno due volte. Allora è un linguaggio infinito, perché se posso ripetere il non terminale almeno 2 volte, posso farlo infinite volte.

Considero costruire infinite stringhe, tutte appartenenti al linguaggio. Allora il linguaggio è infinito.

La forma ridotta che ruolo svolge? Se ci fossero produzioni unitarie nel cammino da  $A$  ad  $A$ , le successive produzioni sarebbero sempre la stessa.

11 \*

6

**pumping lemma per i linguaggi CF**

**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

- $uvwx^n = z$
- $|w| \geq 1$
- $|wx|^n \leq n$
- $\forall i \geq 0 \quad uv^iwx^{n-i} \in L$

**dimostrazione:** analogie con le dimostrazioni dei teoremi precedenti

12 \*

6

19/23/2024

## 050-linguaggi-non-contestuali-16

**pumping lemma – intuizione**

13 \*

6

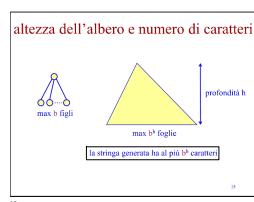
**pumping lemma – dimostrazione**

- sia  $G$  una grammatica in forma ridotta, tale che  $L(G)=L$  e sia  $b$  sia il massimo numero di simboli a destra in ciascuna produzione (assumiamo  $b \geq 2$ )
- in qualunque albero di derivazione per  $G$ :
  - un nodo può avere al massimo  $b$  figli
  - a profondità  $h$  dalla radice  $S$  ci sono al massimo  $b^h$  foglie
  - a profondità  $h$  da  $S$  ci sono al massimo  $b^h$  foglie
  - se l'altezza dell'albero è al più  $h$  la stringa generata ha al più  $b^h$  caratteri
  - se una stringa ha almeno  $b^{h+1}$  caratteri il suo albero deve essere alto almeno  $h+1$

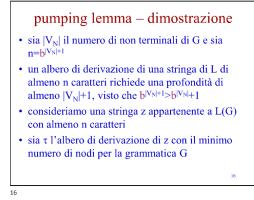
Dimostrazione:  
 $b > 2 \rightarrow$  se ci fosse un solo simbolo non terminale  $\rightarrow$  non sarebbe forma ridotta  
 $\rightarrow$  se ci fosse solo un simbolo terminale non ha senso (es  $S \rightarrow a$ , e poi che succede?)

14 \*

## 050-linguaggi-non-contestuali-16



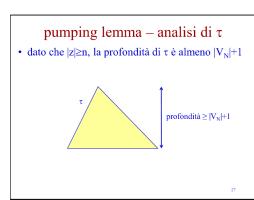
15

Ci prendiamo il numero di non terminali della grammatica  $|V_N|$ , che genera il nostro linguaggio  $L$ .

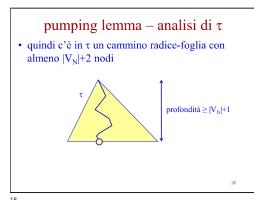
16

8

## 050-linguaggi-non-contestuali-16



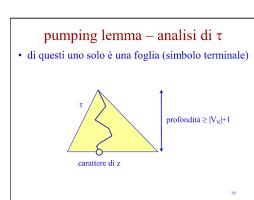
17



18

9

## 050-linguaggi-non-contestuali-16



19

10

## 050-linguaggi-non-contestuali-16



20

10



22

11

050-linguaggi-non-contestuali-16

**pumping lemma – analisi di  $\tau$  – recap**

- dato che  $|z| \geq n$  la profondità di  $\tau$  è almeno  $|V_N|+1$
- quindi c'è in  $\tau$  un cammino radice-foglia con almeno  $|V_N|+2$  nodi
- di questi uno solo è una foglia (simbolo terminale)
- quindi c'è in  $\tau$  un cammino radice-foglia con almeno  $|V_N|+1$  non terminali
- nel cammino c'è almeno un non terminale che si ripete
- sia  $R$  un non terminale con due occorrenze  $R_1$  e  $R_2$  fra i  $|V_N|+1$  non terminali più in basso

23

v

**pumping lemma – dimostrazione**

- suddividiamo  $z = uvwx$
- sia  $vwx$  la sottostringa di  $z$  generata dalla occorrenza  $R_1$  di  $R$  più vicina a  $S$
- sia  $w$  la sottostringa  $\underline{v}$  generata dalla occorrenza  $R_2$  di  $R$  più vicina a  $S$
- possiamo allora suddividere il sottoalbero radicato a  $R_2$  con il sottoalbero radicato a  $R_1$  ottenendo  $uvwx$  con  $i=1$  e rimuovere il sottoalbero radicato a  $R_1$  con il sottoalbero radicato a  $R_2$  ottenendo  $uv^kwx$  con  $i=0$

24

u

12

050-linguaggi-non-contestuali-16

**pumping lemma – dimostrazione**

- per dimostrare che  $|x| > 0$  è sufficiente mostrare che una tra  $v$  e  $x$  deve essere diversa da  $\epsilon$
- se fosse  $v=x=\epsilon$  si potrebbe sostituire il sottoalbero più grande con quello più piccolo, ottenendo ancora  $z$
- ma ciò contrasta con il fatto che  $\tau$  è l'albero di derivazione di  $z$  con il minimo numero di nodi

E quindi questa cosa non può accadere

25

v

**pumping lemma – dimostrazione**

- ora dimostriamo che  $|vwx| \leq n$
- ricordiamo che le due occorrenze  $R_1$  e  $R_2$  di  $R$  sono tra i  $|V_N|+1$  simboli non terminali più in basso nel cammino e scegliamo il cammino come quello più lungo di  $\tau$
- quindi l'albero che genera  $vwx$  è alto al massimo  $|V_N|+1$
- un albero di tale altezza genera una stringa di al massimo  $b^{|V_N|+1}$  caratteri; ma  $b^{|V_N|+1} = n$

26

w

13

050-linguaggi-non-contestuali-16

**il linguaggio  $a^n b^n c^n$** corollario del pumping lemma per i linguaggi non contestuali:  $\{a^n b^n c^n\}_{n \geq 1}$  non è di tipo 2

dimostrazione:

sia  $a^n b^n c^n = uvwx$   
se  $v$  (oppure  $x$ ) contiene almeno due simboli diversi (es.  $v=ab$ )allora la stringa  $uv^kwx$  contiene un'alternanza di simboli incompatibile con  $a^n b^n c^n$  (es.  $uabbcc$ )se  $v$  è composta da simboli tutti uguali fra loro (es.  $v=aa$ ) e  $x$  è composta da simboli tutti uguali fra loro (es.  $x=cc$ )allora  $uv^kwx$  contiene un numero diverso di  $a$ ,  $b$  e  $c$ 

27

Dimostrazione:  
Supponi per assurdo che esista una costante  $n$  tale che per ogni  $\tau$  che ha cardinalità maggiore o uguale a  $n$ , allora prendo la stringa  $a^n b^n c^n$ .

Immaginiamo di averne un'altra che si sposta su questa stringa:

Dobbiamo cercare  $uvwx$ . E possiamo dire che la porzione  $vwx$  è fatta di n caratteri.Nella stringa  $a^n b^n c^n$  la posizione esiste sia a tra b e c, se no non esiste.Se in  $vwx$  la  $v$  è stata usata nella prima parte della stringa,  $v$  e  $x$  sono composte di sole a, il bilanciamento è perduto, e le

porzioni di stringa da replicare NON possono stare la dentro.

Rimanere il caso in cui  $v$  sia a cavallo tra le a e le b, o tra le b e le c: supponiamo sia a cavallo tra le a e le b, ma comunque

le prenderò viene uno sbilanciamento con le c, e non siamo più all'interno del linguaggio.

Vale lo stesso con le b e le c.

Per  $a^n b^n$  vale? Si, è CF. Come scegliamo  $n$  per il linguaggio non contestuale?  
La stringa più piccola che fa parte del linguaggio è "abb", e quindi comunque le moltiplici rimango nel linguaggio.Questo perché  $n=1$ , "ab" andrebbe bene se  $n>0$ .Come si dimostra che  $L_1$  è di tipo 2? NON col PL, ma scrivendo una grammatica CF che lo genera:

S-&gt;ABC

A-&gt;aAb|epsilon

B-&gt;bBc|epsilon

C-&gt;cCc|epsilon

Invece per  $L_2$ :

S-&gt;ABx

A-&gt;aA|a

B-&gt;bM|epsilon

C-&gt;cC|epsilon

28

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

osservazione: dimostrazione della decidibilità dell'equivalenza tra linguaggi regolari ai fondi sulla chiusura dell'intersezione: tale tecnica non è estendibile ai linguaggi non contestuali

&gt;&gt; MA NON E' IN FORMA RIDOTTA. Ma non fa niente perché un teorema che lo dimostra che per ogni grammatica di tipo 2 ha una grammatica in forma ridotta

Come si dimostra che  $L_1$  è di tipo 2? NON col PL, ma scrivendo una grammatica CF che lo genera:

S-&gt;ABC

A-&gt;aAb|epsilon

B-&gt;bBc|epsilon

C-&gt;cCc|epsilon

Invece per  $L_2$ :

S-&gt;ABx

A-&gt;aA|a

B-&gt;bM|epsilon

C-&gt;cC|epsilon

29

## 050-linguaggi-non-contestuali-16

**chiusura dei linguaggi non contestuali**

teorema: i linguaggi non contestuali sono chiusi rispetto all'unione, alla concatenazione e alla iterazione

**dimostrazione:**

siano  $\langle \Sigma, V_{S_1}, P_1, S_1 \rangle < \Sigma, V_{S_2}, P_2, S_2 \rangle$  due linguaggi CF

- unione  
aggiungiamo  $S = S_1 \cup S_2$ , e usiamo  $S'$  come nuovo assioma  
otteniamo un linguaggio non contestuale che genera l'unione dei due linguaggi
- concatenazione  
aggiungiamo  $S = S_1 S_2$ , e usiamo  $S'$  come nuovo assioma
- iterazione  
aggiungiamo  $S = S^*$ , e usiamo  $S'$  come nuovo assioma

29

Linguaggi  
non...

Registrazione audio avviata: 09:07 mercoledì 30 ottobre 2024

- Unione:  
Genero una grammatica che non è in forma ridotta, ma c'è il teorema che mi dice che se c'è una grammatica c'è anche una in forma ridotta
- Concatenazione:  
Processo induttivo
- Iterazione:  
Ho una epilogo produzione, e non da fastidio (anche se per la forma ridotta non va bene)

Rispetto alla complementazione invece? Sono chiusi i linguaggi CF?  
NO, perché se fossero chiusi rispetto la complementazione potrei descrivere l'operatore di intersezione con de Morgan che sfrutta la chiusura dell'intersezione e dell'unione, che sarebbe un contraddizione.

**forme normali**

- una **forma normale** per una grammatica non contestuale consiste in un insieme di vincoli sulle sue produzioni tale che tutti i soli i linguaggi non contestuali ammettono una grammatica nella forma normale
- studieremo le seguenti
  - forma normale di Chomsky (CNF, Chomsky Normal Form)
    - quasi-CNF
    - forma normale di Greibach (GNF, Greibach Normal Form)

30

15

## 050-linguaggi-non-contestuali-16

**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 → BC oppure A → a

– una grammatica di tipo 2 è in **forma normale di quasi-Chomsky** (quasi-CNF) se tutte le produzioni sono del tipo

A → BC...Z oppure A → a

31

Sono del tipo:

- Se un non terminale a 2 non terminali, o da un non terminale a un terminale.  
Se faccio queste 2 produzioni, genero ancora tutti i linguaggi non contestuali (CF). Riduco molto le possibilità a dx della freccia, ma se mi limito a queste grammatiche genero comunque linguaggi non contestuali.
- > questa forma ci aiuta nelle dimostrazioni successive.  
La forma è:  
Da un non terminale ad una sequenza di non terminali, oppure da un non terminale ad un terminale.

31

**grammatica → CNF**

- **teorema:** data una grammatica G di tipo 2 tale che c'è L(G) esiste una grammatica G' in CNF con L(G)=L(G')
- **dimostrazione:**
  - portiamo la grammatica in forma ridotta
  - portiamo la grammatica in forma normale di quasi-Chomsky
  - portiamo la grammatica in forma normale di Chomsky

32

Sono del tipo:

- grammatica in forma ridotta
1. non contiene ε-produzioni, salvo (eventualmente) sull'assimone
  2. se l'assimone contiene ε-produzioni, allora non compare mai a destra in una produzione
  3. non contiene produzioni multiple
  4. non contiene produzioni unitarie
    - esempio di produzione unitaria: A → B

32

1

\*

**grammatica → quasi-CNF**

supponiamo che una grammatica in forma ridotta non sia in forma normale di quasi-Chomsky

- necessariamente ci sono anche produzioni del tipo p: A → β₁β₂...βₙ  
con n≥2, tali che per qualche i ∈ {1,...,n} si ha βᵢ ∈ V\_T
- per ogni βᵢ ∈ V\_T aggiungiamo un non terminale Yᵢ, sostituendo Yᵢ a βᵢ in p ed aggiungiamo la produzione Yᵢ → βᵢ

33

\*

**quasi-CNF → CNF**

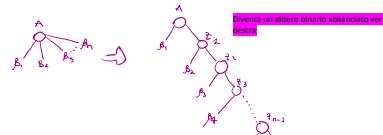
supponiamo che una grammatica in forma normale di quasi-Chomsky non sia in forma normale di Chomsky

- necessariamente ci sono produzioni del tipo p: A → β₁β₂...βₙ  
con n≥3, tali che per ogni i ∈ {1,...,n} si ha βᵢ ∈ V\_N
- aggiungiamo n-2 nuovi non terminali Z₁,...,Zₙ₋₂ e sostituiamo p: A → β₁β₂...βₙ con una catena di produzioni

A → Z₁  
Z₁ → β₁Z₂  
Z₂ → β₂Z₃  
...  
Zₙ₋₂ → βₙ₋₂βₙ

34

\*



17

## 050-linguaggi-non-contestuali-16

**esempio**

trasformazione in CNF della grammatica

S → aSb  
S → ab

sostituiamo S → ab; con

S → AB  
A → a  
B → b

sostituisci S → aSb con

S → ASB  
sostituisci S → ASB con

a → A

18 23 5824



35

Linguaggi  
non...**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  $A \rightarrow \beta_1$  con  $\beta \in V^*$ .

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

Registrazione audio avviata: 17:05 giovedì 21 ottobre 2024  
 **$\beta_1$**  può non essere proprio perché  $V^n$  dentro ha la stringa vuota.  
 Beta può contenere due non terminali, 3, 4 ecc..  
 Il primo carattere è un terminale

Le grammatiche regolari sono già in forma normale di Greibach in cui  **$\beta_1$**  è stringa vuota o un singolo non terminale.

»

18

36

18

## 050-linguaggi-non-contestuali-16

Linguaggi  
non...**teorema — grammatiche tipo 2 e GNF**

**teorema:** data una grammatica  $G$  di tipo 2 tale che  $L(G) = L(G')$  esiste una grammatica  $G'$  in GNF con  $L(G) = L(G')$

**dimostrazione:**

strategy: procedere

- prendiamo  $G$  in quasi-CNF
  - tutti i produttori solo del tipo  $A \rightarrow BCD$ , copiare  $A \rightarrow \beta$
  - trasformiamo la quasi-CNF in una particolare quasi-CNF con del vincolo aggiuntivo di precedenza
  - sostituzione
  - eseguiamo delle operazioni di **sostituzione** dei primi non terminali nella parte destra delle produzioni

»

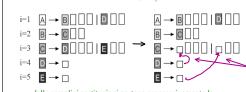
37

18

**intuizione della dimostrazione**

• supponiamo che nella nostra grammatica quasi-CNF

- i non terminali siano ordinati in una successione  $A_1, A_2, \dots, A_n$
- se una produzione ha la forma  $A_i \rightarrow \beta_j$  allora  $j < i \Rightarrow$  Si va da un non terminale più piccolo ad un non terminale più grande.



E a destra della freccia può avere non terminali? **No, perché è l'ultimo della**

A destra "E" può avere un terminale necessariamente. Allora al posto di E che compare nelle altre produzioni ci metto il terminale. Vale anche per D, per rispettare la gerarchia del non terminale più piccolo al più grande

»

38

19

## 050-linguaggi-non-contestuali-16

**forme normali per linguaggi di tipo 2****lemma (sostituzione):**

data una grammatica  $G$  le cui produzioni includono

$$\begin{aligned} A &\rightarrow \alpha_1 B \alpha_2 \\ B &\rightarrow \beta_1 \dots \beta_m \end{aligned}$$

la grammatica  $G'$  in cui la produzione

$$\begin{aligned} A &\rightarrow \alpha_1 B \alpha_2 \\ A &\rightarrow \alpha_1 \beta_1 \alpha_2 \dots \alpha_m \alpha_2 \\ (\text{le produzioni di } B \text{ sono invariate}) \end{aligned}$$

è equivalente a  $G$

**dimostrazione:** banale

»

39

19

**forme normali per linguaggi di tipo 2****lemma (eliminazione della ricorsione sinistra):**

data una grammatica  $G$  le cui produzioni includono

$$A \rightarrow \alpha_1 \dots \alpha_n \beta_1 \dots \beta_m$$

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

$$\begin{aligned} A &\rightarrow \alpha_1 \dots \alpha_n \beta_1 \dots \beta_m \\ A &\rightarrow \alpha_1 \beta_1 \dots \beta_m \beta_1 \dots \beta_m \\ B &\rightarrow \alpha_1 \beta_1 \dots \beta_m \beta_1 \dots \beta_m \end{aligned}$$

è equivalente a  $G$

»

40

20

## 050-linguaggi-non-contestuali-16

19

**eliminazione della ricorsione sinistra**

ogni derivazione in  $G$  del tipo

$$A \Rightarrow \alpha_1 \Rightarrow \alpha_1 \alpha_2 \Rightarrow \dots \Rightarrow \alpha_1 \alpha_{n_1} \dots \alpha_{n_1} \alpha_2 \Rightarrow \beta_1 \alpha_{n_1} \dots \alpha_{n_1} \alpha_2$$

è rimpiazzato in  $G'$  da

$$A \Rightarrow \beta_1 \beta_2 \dots \beta_m \alpha_{n_1} \dots \alpha_{n_1} \alpha_2 \Rightarrow \dots \Rightarrow \beta_1 \alpha_{n_1} \dots \alpha_{n_1} \alpha_2, \dots, \alpha_1 \alpha_2$$

è possibile effettuare anche il passaggio inverso

Registrazione audio avviata: 10:13 mercoledì 21 ottobre 2024

»

41

20

**dimostrazione — grammatica  $\rightarrow$  GNF**

se le produzioni **non** sono ordinate in modo tale che se  $A \rightarrow A_1 \beta$  è una produzione, allora  $i < j$

- stabiliamo arbitrariamente un ordinamento  $A_1, A_2, \dots, A_m$  sui non terminali di  $G$

• conviene scegliere un ordinamento che ha poche eccezioni

• ma questo che si vuole imporre

- eseguiamo delle trasformazioni che impongono progressivamente il vincolo

• tali trasformazioni possono introdurre nuovi non terminali che vengono messi in testa alla lista

Se ci riesco, applico poi il procedimento di sostituzione a cui abbiamo accennato prima

»

42

20

## 050-linguaggi-non-contestuali-16

**dimostrazione – grammatica  $\rightarrow$  GNF**

passo 1

- modifichiamo le produzioni in modo tale che se  $A_i \rightarrow A_j$  è una produzione, allora  $j \leq i$
- assumiamo che ciò sia stato già fatto per le produzioni  $A_i \rightarrow A_j$  con  $i < k$
- quindi possiamo usare le  $A_k$ -produzioni
- se  $A_k \rightarrow A_l$  è una produzione, allora usiamo l'operazione di sostituzione rimpiazzando  $A_k$  con le produzioni  $A_k \rightarrow A_l$
- ripetendo questa operazione al più  $k-1$  volte otteniamo produzioni nella forma  $A_k \rightarrow A_l \gamma$  per  $\gamma \in P_k$
- se  $\gamma \neq \epsilon$  applichiamo l'eliminazione della ricorsione a sinistra introducendo un nuovo terminale  $B_k$  in testa all'ordinamento delle categorie sintattiche

43

Così che si va dal più piccolo al più grande.  
Al termine prima di  $A_k$ . Dobbiamo modificare le produzioni in modo tale che si vada dal non terminale più piccolo al più grande.

\* -> j < k non va bene: allora faccio l'operazione di sostituzione....

**passo 1 – algoritmo**

```
for (k=1; k<n; k++)
    for (j=1; j<k; j++)
        per ogni produzione del tipo  $A_k A_k \alpha$ 
            rimuovi  $A_k A_k$ 
            per ogni produzione  $A_k A_k \alpha$ 
                applica  $A_k \beta$ 
        )
    per ogni produzione del tipo  $A_k A_k \alpha$ 
        rimuovi  $A_k A_k$ 
        applica  $A_k \beta$ 
        per ogni produzione del tipo  $A_k \beta$ 
            per ogni produzione  $A_k \beta \gamma$  dove  $\gamma \neq \epsilon$  e non inizia con  $A_k$ 
                applica  $A_k \beta \gamma$ 
)
)
```

44

22

## 050-linguaggi-non-contestuali-16

**considerazioni sul passo 1**

- ripetendo il procedimento abbiamo solo produzioni del tipo:
  - $A_i \rightarrow A_l \gamma$ , con  $j > i$
  - $A_i \rightarrow \gamma$ , con  $\gamma \in V_T$
  - $B_i \rightarrow \gamma$ , con  $\gamma \in (V_N \cup B_1, \dots, B_n)^*$

45

**dimostrazione – grammatica  $\rightarrow$  GNF**

passo 2

al termine del passo 1

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

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

46

23

## 050-linguaggi-non-contestuali-16

**dimostrazione – grammatica  $\rightarrow$  GNF**

considerazione sui non terminali  $B_i$

- 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  $B_i A_k \alpha$  se  $A_k \neq \epsilon$  e non è mai vuoto e non può iniziare con un  $B_j$
  - dunque se abbiamo messo le  $B_i$  in testa all'ordinamento non abbiamo creato nessuna dipendenza che contraddice l'ordinamento

47

48

**esempio: CNF  $\rightarrow$  GNF**

data la seguente grammatica in CNF

$$\begin{cases} S \rightarrow AB \mid b \\ A \rightarrow B \mid BS \\ B \rightarrow a \mid BA \mid AS \end{cases}$$

calcolarne una equivalente in GNF

- scelgono l'ordinamento:  $S, A, B$
- sostituiamo  $B \rightarrow AS$  con  $B \rightarrow BSS$  e  $B \rightarrow BSS'$
- ora l'insieme delle B-produzioni è il seguente:

$$B \rightarrow a \mid BSS \mid BSS'$$

48

Dobbiamo scegliere l'ordinamento dei non terminali. Scelta S.A.B.  
L'ultima produzione ha qualcosa che non va: c'è un ordinamento che non va bene ( $B \rightarrow AS$ ). Ricorda: ti interessa solo il primo simbolo che è un non terminale, dopo non ti interessa più cosa c'è.  
Nell'ultima produzione sostituisco  $AS$  con  $BSS$ . Però questa ultima produzione ancora non va bene.  
Devo eliminare la ricorsione a sinistra

49

24

## 050-linguaggi-non-contestuali-16

**esempio: passo 1**

- eliminiamo la ricorsione a sinistra con  $B'$ 
  - $B' \rightarrow BSS \mid BSS'$
  - $B' \rightarrow A \mid SS \mid AB \mid SS'$
  - $B' \rightarrow A \mid SS \mid AB \mid SS'$
- giungiamo quindi alla grammatica
 
$$\begin{cases} B' \rightarrow A \mid SS \mid AB \mid SS' \\ S \rightarrow AB \mid b \\ A \rightarrow b \mid BS \end{cases}$$

Ciò che è uscito fuori:  
Va tutto bene adesso.

Applico l'eliminazione della ricorsione a sinistra su  $B$ .  
Creo  $B'$

Applico l'eliminazione della ricorsione a sinistra su  $B$ .

