

Grammatiche libere da contesto

- Abbiamo visto che **multi linguaggi non sono regolari**.
Consideriamo allora classi piu' grandi di linguaggi
- I **Linguaggi Liberi da Contesto** (CFL) sono stati usati nello studio dei linguaggi naturali dal 1950, e nello studio dei compilatori dal 1960
- Le **grammatiche libere da contesto** (CFG) sono una notazione formale per definire linguaggi definiti da regole ricorsive
- Studieremo:
 - CFG
 - CFL
 - alberi sintattici
 - automi a pila
 - proprieta' di chiusura dei CFL

- Consideriamo $L_{pal} = \{w \in \Sigma^* : w = w^R\}$
- Per esempio: otto $\in L_{pal}$, madamimadam $\in L_{pal}$.
- L_{pal} non e' regolare.
 - Sia $\Sigma = \{0, 1\}$. Dimostriamo grazie al **Pumping lemma** che L_{pal} non e' regolare
 - Il giocatore G1 vuole dimostrare che L_{pal} non e' regolare.
 - Il giocatore G2 sceglie n .
 - Il giocatore G1 sceglie $w = 0^n 1 0^n \in L_{pal}$, $|w| \geq n$.
 - Il giocatore G2 sceglie la partizione $w = xyz$ tale che $y \neq \epsilon$ e $|xy| \leq n$.
 - Il giocatore G1 sceglie $k = 0$ e dimostra che $xz \notin L_{pal}$ poiche', avendo scelto $w = 0^n 1 0^n$, il numero dei primi zeri in xz e' strettamente minore del numero dei secondi zeri essendo $y \neq \epsilon$, $|xy| \leq n$.

Esempio informale di CFG

- $L_{pal} = \{w \in \Sigma^* : w = w^R\}$
- Definiamo L_{pal} induttivamente:
 - **Base:** ϵ , 0, e 1 sono palindromi.
 - **Induzione:** Se w e' una palindroma, anche $0w0$ e $1w1$ lo sono.
 - Nessun'altra stringa e' una palindroma.
- Le CFG sono un modo formale per def. linguaggi come L_{pal} .

1. $P \rightarrow \epsilon$
2. $P \rightarrow 0$
3. $P \rightarrow 1$
4. $P \rightarrow 0P0$
5. $P \rightarrow 1P1$

- 0 e 1 sono *terminali*
- P e' una *variabile* (o *nonterminale*, o *categoria sintattica*)
- P e' in questa grammatica anche il *simbolo iniziale*.
- 1-5 sono *produzioni* (o *regole*)

Una *grammatica libera da contesto* e' una quadrupla

$$G = (V, T, P, S)$$

dove

- V e' un insieme finito di **variabili**
- T e' un insieme finito di **terminali**
- P e' un insieme finito di **produzioni** della forma $A \rightarrow \alpha$, dove A e' una variabile e $\alpha \in (V \cup T)^*$
- S e' una variabile chiamata il **simbolo iniziale**

- Esempio: $G_{pal} = (\{P\}, \{0, 1\}, A, P)$,
dove $A = \{P \rightarrow \epsilon, P \rightarrow 0, P \rightarrow 1, P \rightarrow 0P0, P \rightarrow 1P1\}$.

A volte raggruppiamo le produzioni con la stessa testa:

$$A = \{P \rightarrow \epsilon | 0 | 1 | 0P0 | 1P1\}.$$

Esempio: Espressioni (semplici) in un tipico ling. di programmazione:

- Gli operatori sono $+$ e $*$
- Gli operandi sono identificatori, cioè stringhe in $L((\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b} + \mathbf{0} + \mathbf{1})^*)$

Esempio: Espressioni (semplici) in un tipico ling. di programmazione: gli operatori sono $+$ e $*$ e gli operandi sono identificatori, cioè stringhe in $L((\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b} + \mathbf{0} + \mathbf{1})^*)$

Usiamo la grammatica $G = (\{E, I\}, T, P, E)$ dove $T = \{+, *, (,), a, b, 0, 1\}$ e P e' il seguente insieme di produzioni:

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

- **Derivazioni:** per definire linguaggio di una CFG G , $L(G)$
- $L(G)$: stringhe di terminali che si possono **derivare** a partire dal simbolo iniziale
- Sia $G = (V, T, P, S)$ una CFG
sia $A \in V$, siano $\alpha, \beta \in (V \cup T)^*$,
sia $A \rightarrow \gamma \in P$
Allora scriviamo

$$\alpha A \beta \xRightarrow{G} \alpha \gamma \beta$$

o, se e' ovvia la G , $\alpha A \beta \Rightarrow \alpha \gamma \beta$ e
diciamo che **da** $\alpha A \beta$ **si deriva** $\alpha \gamma \beta$.

- Definiamo $\xRightarrow{*}$ la chiusura riflessiva e transitiva di \Rightarrow , cioe':
 - **Base:** Sia $\alpha \in (V \cup T)^*$. Allora $\alpha \xRightarrow{*} \alpha$.
 - **Induzione:** Se $\alpha \xRightarrow{*} \beta$, e $\beta \Rightarrow \gamma$, allora $\alpha \xRightarrow{*} \gamma$.

Esempio visto in precedenza

Esempio: Definire la grammatica delle espressioni (semplici) in un tipico ling. di programmazione: **gli operatori sono + e *** e **gli operandi sono** identificatori, cioè stringhe **in $L((a + b)(a + b + 0 + 1)^*)$**

Usiamo la **grammatica** $G = (\{E, I\}, T, P, E)$ dove $T = \{+, *, (,), a, b, 0, 1\}$ e P e' il seguente insieme di produzioni:

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

- Esercizio: Determinare una derivazione di $a * (a + b00)$ da E nella grammatica delle espressioni:

$$\begin{aligned}
 E &\Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow \\
 a * (E + E) &\Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow \\
 a * (a + I0) &\Rightarrow a * (a + I00) \Rightarrow a * (a + b00)
 \end{aligned}$$

- Ad ogni passo potremmo avere varie regole tra cui scegliere, ad esempio
 - $I * E \Rightarrow a * E \Rightarrow a * (E)$, oppure
 - $I * E \Rightarrow I * (E) \Rightarrow a * (E)$
- Non tutte le scelte portano a derivazioni di una particolare stringa, per esempio

$$E \Rightarrow E + E$$

non ci fa derivare $a * (a + b00)$

Derivazioni a sinistra e a destra

- **Derivazione a sinistra** \Rightarrow_{lm} : rimpiazza sempre la variabile piu' a sinistra con il corpo di una delle sue regole.
- **Derivazione a destra** \Rightarrow_{rm} : rimpiazza sempre la variabile piu' a destra con il corpo di una delle sue regole.
- Der. a sinistra: quella del lucido precedente.
- Der. a destra: quella seguente.

Esercizio per casa: Determinare una derivazione a destra di $a * (a + b00)$ da E nella grammatica delle espressioni:

$$\begin{aligned} E &\Rightarrow_{rm} E * E \Rightarrow_{rm} \\ E * (E) &\Rightarrow_{rm} E * (E + E) \Rightarrow_{rm} E * (E + I) \Rightarrow_{rm} E * (E + I0) \\ &\Rightarrow_{rm} E * (E + I00) \Rightarrow_{rm} E * (E + b00) \Rightarrow_{rm} E * (I + b00) \\ &\Rightarrow_{rm} E * (a + b00) \Rightarrow_{rm} I * (a + b00) \Rightarrow_{rm} a * (a + b00) \end{aligned}$$

Possiamo concludere che $E \xRightarrow{*}_{rm} a * (a + b00)$

- Se $G = (V, T, P, S)$ e' una CFG, allora il **linguaggio di G** e'

$$L(G) = \{w \in T^* : S \xrightarrow[G]{*} w\}$$

cioe' l'insieme delle stringhe di terminali derivabili dal simbolo iniziale.

- Se G e' una CFG, chiameremo $L(G)$ un **linguaggio libero da contesto**

Esempio: $L(G_{pal})$ e' un linguaggio libero da contesto

$G_{pal} = (\{P\}, \{0, 1\}, A, P)$, dove $A = \{P \rightarrow \epsilon | 0|1|0P0|1P1\}$

Teorema 5.7: Sia $L_{pal} = \{w \in \{0, 1\}^* : w = w^R\}$.

Allora $L(G_{pal}) = L_{pal}$.

- Sia $G = (V, T, P, S)$ una CFG

Sia $\alpha \in (V \cup T)^*$

Se $S \xRightarrow{*} \alpha$ diciamo che α e' una **forma sentenziale**.

Se $S \xRightarrow{lm} \alpha$ diciamo che α e' una forma sentenziale sinistra

Se $S \xRightarrow{rm} \alpha$ diciamo che α e' una forma sentenziale destra

- Nota: $L(G)$ contiene le forme sentenziali che sono in T^* .

- Prendiamo la G delle espressioni.

$E * (I + E)$ e' una forma sentenziale perche'

$$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (I + E)$$

Questa derivazione non e' ne' a sinistra ne' a destra

- $a * E$ e' una forma sentenziale sinistra, perche'

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow} I * E \underset{lm}{\Rightarrow} a * E$$

- $E * (E + E)$ e' una forma sentenziale destra, perche'

$$E \underset{rm}{\Rightarrow} E * E \underset{rm}{\Rightarrow} E * (E) \underset{rm}{\Rightarrow} E * (E + E)$$

- Descrivere una CFG che genera il linguaggio seguente.
Per ogni variabile descrivere l'insieme delle stringhe generate.

① $L = \{0^n 1^n, n \geq 0\}$

② $L = \{0^n 1^n, n \geq 1\}$

③ $L = \{a^i b^j c^k, j \neq k\}$ (II compitino 2013)

- Descrivere il linguaggio generato dalla seguente CFG
 $G = (\{S, Z\}, \{0, 1\}, P, S)$, dove $P = \{S \rightarrow 0S1 \mid 0Z1, Z \rightarrow 0Z \mid \epsilon\}$

- Esercizi per casa

- Descrivere una CFG che genera il linguaggio seguente: Per ogni variabile descrivere l'insieme delle stringhe generate.
 - ① $L = \{w \in \{0, 1\}^* \mid w = (0 + 1)^* 1\}$
 - ② $L = \{w \in \{0, 1, 2\}^+ \mid w = x2x', x, x' \in (0 + 1)^*, x' = x^R\}$.
 $x' = x^R$ significa che x' è la stringa di x rovesciata, per esempio se $x = 011$ allora $x' = 110$ (III appello 2013).

- Se $w \in L(G)$, per una CFG, allora w ha un **albero sintattico**, che ci dice la struttura (sintattica) di w
- w potrebbe essere un programma, una query SQL, un documento XML, ...
- Gli alberi sintattici sono una **racpresentazione grafica alternativa alle derivazioni**
- Ci possono essere diversi alberi sintattici per la stessa stringa
- Idealmente ci dovrebbe essere solo un albero sintattico (la "vera" struttura), cioè il linguaggio dovrebbe essere non ambiguo
- Sfortunatamente, non sempre possiamo rimuovere l'ambiguità

Sia $G = (V, T, P, S)$ una CFG

Un albero e' un **albero sintattico** per G se:

- 1 Ogni nodo interno e' etichettato con una variabile in V
- 2 Ogni foglia e' etichettata con un simbolo in $V \cup T \cup \{\epsilon\}$ Ogni foglia etichettata con ϵ e' l'unico figlio del suo genitore
- 3 Se un nodo interno e' etichettato A , e i suoi figli (da sinistra a destra) sono etichettati

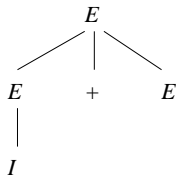
$$X_1, X_2, \dots, X_k,$$

allora $A \rightarrow X_1 X_2 \dots X_k \in P$.

Nella grammatica

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
- ⋮

il seguente e' un albero sintattico:

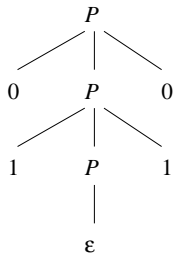


Questo albero sintattico mostra la derivazione $E \xRightarrow{*} I + E$

Nella grammatica

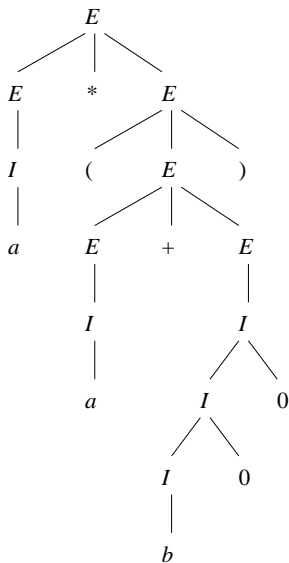
1. $P \rightarrow \epsilon$
2. $P \rightarrow 0$
3. $P \rightarrow 1$
4. $P \rightarrow 0P0$
5. $P \rightarrow 1P1$

il seguente e' un albero sintattico:



Mostra la derivazione $P \xRightarrow{*} 0110$.

- Il **prodotto** di un albero sintattico e' la stringa di foglie da sinistra a destra
 - Sono importanti quegli alberi sintattici dove:
 - ① Il prodotto e' una stringa terminale
 - ② La radice e' etichettata dal simbolo iniziale
- L'insieme dei prodotti di questi alberi sintattici e' il linguaggio della grammatica



Il prodotto e' $a * (a + b00)$

Sia $G = (V, T, P, S)$ una CFG e $A \in V$

I seguenti sono equivalenti:

① $A \xRightarrow{*} w$

② $A \xRightarrow[*]{lm} w$

③ $A \xRightarrow[*]{rm} w$

④ C'è un **albero sintattico** di G con **radice** A e **prodotto** w

Nella grammatica

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
- ...

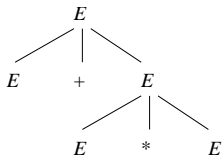
la forma sentenziale $E + E * E$ ha due derivazioni:

$$E \Rightarrow E + E \Rightarrow E + E * E$$

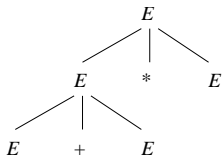
e

$$E \Rightarrow E * E \Rightarrow E + E * E$$

Questo ci da' due alberi sintattici:



(a)



(b)

L'esistenza di **varie derivazioni** di per se non e' pericolosa, e'
l'esistenza di **vari alberi sintattici** che rovina la grammatica.

Esempio: Nella stessa grammatica

5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

la stringa $a + b$ ha varie derivazioni:

$$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$

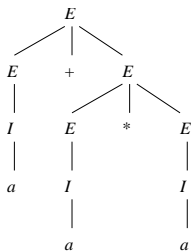
e

$$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$$

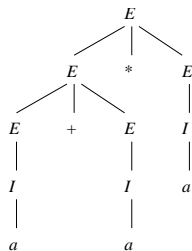
Pero' il loro albero sintattico e' lo stesso, e la struttura di $a + b$ e'
quindi non ambigua

Ambiguità in Grammatiche e Linguaggi

- Sia $G = (V, T, P, S)$ una CFG
 G è **ambigua** se esiste una stringa w in T^* che ha più di un albero sintattico con radice S e prodotto w
- Se ogni stringa in $L(G)$ ha al più un albero sintattico, G è detta **non ambigua**
- Esempio: La stringa terminale $a + a * a$ ha due alberi sintattici:



(a)



(b)

Rimuovere l'ambiguità' dalle grammatiche

- Buone notizie: a volte possiamo rimuovere l'ambiguità'
- Cattive notizie: non c'è nessun algoritmo per farlo in modo sistematico
- Ancora cattive notizie: alcuni CFL hanno solo CFG ambigue
- Studiamo la grammatica

$$E \rightarrow I \mid E + E \mid E * E \mid (E)$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

- Non c'è precedenza tra * e +
- Non c'è raggruppamento di sequenze di operatori: $E + E + E$ è inteso come $E + (E + E)$ o come $(E + E) + E$?

Soluzione: Introduciamo piu' variabili, ognuna che rappresenta espressioni con lo stesso grado di "forza di legamento".

Consideriamo la seguente grammatica:

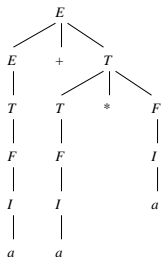
$$1. I \rightarrow a \mid b \mid la \mid lb \mid l0 \mid l1$$

$$2. F \rightarrow I \mid (E)$$

$$3. T \rightarrow F \mid T * F$$

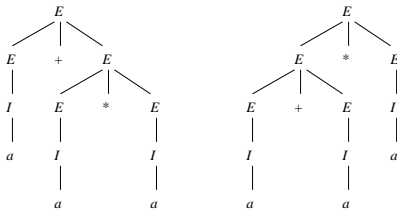
$$4. E \rightarrow T \mid E + T$$

Ora l'unico albero sintattico per $a + a * a$ e':



Derivazioni a sinistra e ambiguità'

I due alberi sintattici per $a + a * a$ nella grammatica ambigua



danno luogo a due derivazioni a sinistra:

$$\begin{aligned} E &\Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + E * E \\ &\quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \\ &\Rightarrow a + I * E \Rightarrow a + a * E \Rightarrow a + a * I \Rightarrow \underline{a + a * a} \\ &\quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \end{aligned}$$

e

$$\begin{aligned} E &\Rightarrow E * E \Rightarrow E + E * E \Rightarrow I + E * E \Rightarrow a + E * E \\ &\quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \\ &\Rightarrow a + I * E \Rightarrow a + a * E \Rightarrow a + a * I \Rightarrow \underline{a + a * a} \\ &\quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \end{aligned}$$

In generale:

- Un albero sintattico, ma molte derivazioni
- Molte derivazioni a sinistra implica molti alberi sintattici
- Molte derivazioni a destra implica molti alberi sintattici

Teorema 5.29:

Data una CFG $G = (V, T, P, S)$,

una stringa $w \in T^*$ ha **due distinti alberi sintattici** se e solo se w ha **due distinte derivazioni a sinistra** da S

Un CFL L e' **inerentemente ambiguo** se **tutte** le grammatiche per L sono ambigue

- Esercizio:

La CFG con produzioni $S \rightarrow aS|aSbS|\epsilon$ e' ambigua?

Provarlo formalmente.

Se la CGF sopra e' ambigua, trovare una CFG non ambigua per il linguaggio da essa generato.

- Esercizio (II compitino 2013):

Dimostrare con un esempio che la grammatica con le seguenti produzioni è ambigua (suggerimento: è sufficiente una stringa di lunghezza 4).

$$S \rightarrow aSbS|bSaS|\epsilon$$

- Un linguaggio regolare e' anche libero da contesto
- Da una espressione regolare
si puo' ottenere una grammatica che genera lo stesso linguaggio
- Da un automa a stati finiti,
si puo' ottenere una grammatica che genera lo stesso linguaggio

Per induzione sulla struttura della espressione regolare:

- se $E = a$, allora produzione $S \rightarrow a$
- se $E = \epsilon$, allora produzione $S \rightarrow \epsilon$
- se $E = F + G$, allora produzione $S \rightarrow F \mid G$
- se $E = FG$, allora produzione $S \rightarrow FG$
- se $E = F^*$, allora produzione $S \rightarrow FS \mid \epsilon$

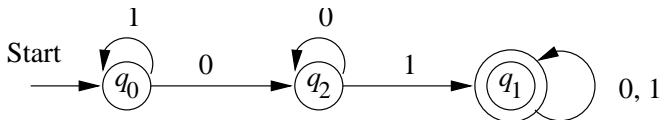
Espressione regolare: $0^*1(0 + 1)^*$

Grammatica:

$$S \rightarrow ABC$$
$$A \rightarrow 0A \mid \epsilon$$
$$B \rightarrow 1$$
$$C \rightarrow DC \mid \epsilon$$
$$D \rightarrow 0 \mid 1$$

- Una variabile per ogni stato
- Simbolo iniziale = stato iniziale
- Per ogni transizione da stato s a stato p con simbolo a ($\delta(s, a) = p$), produzione $S \rightarrow aP$
- Se p stato finale, allora produzione $P \rightarrow \epsilon$

Automa:



Grammatica:

$$Q_0 \rightarrow 1Q_0 \mid 0Q_2$$

$$Q_2 \rightarrow 0Q_2 \mid 1Q_1$$

$$Q_1 \rightarrow 0Q_1 \mid 1Q_1 \mid \epsilon$$

La stringa 1101 e' accettata dall'automa. Nella grammatica, ha la derivazione:

$$Q_0 \Rightarrow 1Q_0 \Rightarrow 11Q_0 \Rightarrow 110Q_2 \Rightarrow 1101Q_1 \Rightarrow 1101$$