

Ambiguità in Grammatiche e Linguaggi

Nella grammatica

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

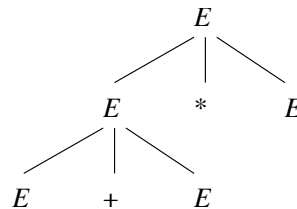
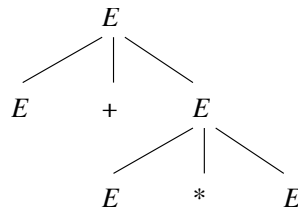
la forma sentenziale $E + E * E$ ha due derivazioni:

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

e

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

Questo ci dà due alberi sintattici:



Grammatiche libere da contesto

L'esistenza di **varie derivazioni** di per sé non è pericolosa, è 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

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

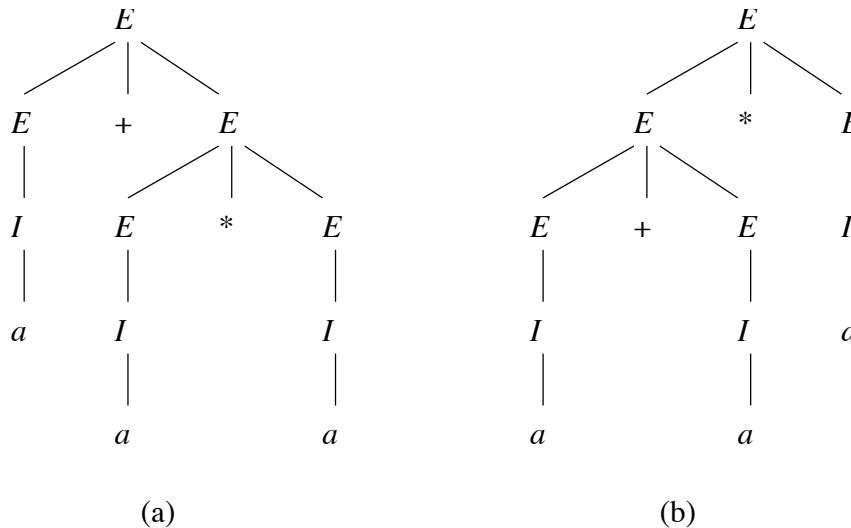
Però il loro albero sintattico è lo stesso, e la struttura di $a + b$ è quindi non ambigua.

Grammatiche libere da contesto

Definizione: Sia $G = (V, T, P, S)$ una CFG. Diciamo che G è **ambigua** se esiste una stringa in T^* che ha più di un albero sintattico.

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:



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 $(E + E) + E$?

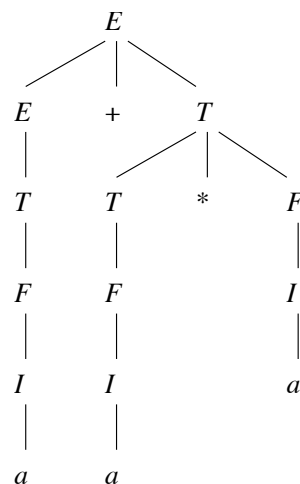
Soluzione: Introduciamo più variabili, ognuna che rappresenta espressioni con lo stesso grado di "forza di legame"

- 1 Un *fattore* è un'espressione che non può essere spezzata da un $*$ o un $+$ adiacente. I nostri fattori sono:
 - 1 Identificatori
 - 2 Un'espressione racchiusa tra parentesi.
- 2 Un *termine* è un'espressione che non può essere spezzata da un $+$. Ad esempio, $a * b$ può essere spezzata da $a1*$ o $*a1$. Non può essere spezzata da $+$, perché ad esempio $a1 + a * b$ è (secondo le regole di precedenza) lo stesso di $a1 + (a * b)$, e $a * b + a1$ è lo stesso di $(a * b) + a1$.
- 3 Il resto sono *espressioni*, cioè possono essere spezzate con $*$ o $+$.

Usiamo F per i fattori, T per i termini, e E per le espressioni. Consideriamo la seguente grammatica:

1. $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
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 $a + a * a$ è:



Perché la nuova grammatica non è ambigua?

- Un fattore è o un identificatore o (E) , per qualche espressione E .
- L'unico albero sintattico per una sequenza

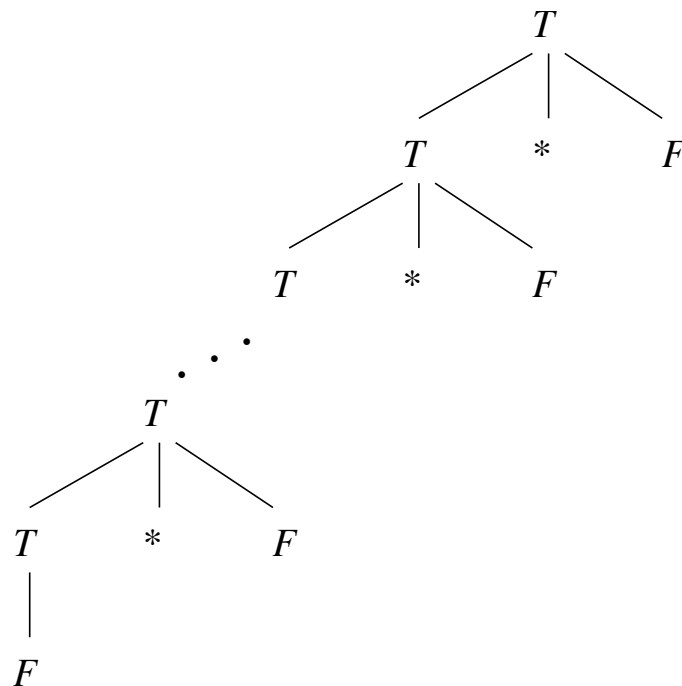
$$f_1 * f_2 * \cdots * f_{n-1} * f_n$$

di fattori è quello che dà $f_1 * f_2 * \cdots * f_{n-1}$ come termine e f_n come fattore, come nell'albero del prossimo lucido.

- Un'espressione è una sequenza

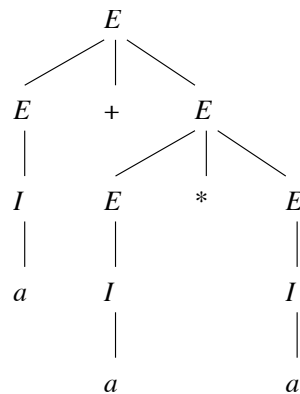
$$t_1 + t_2 + \cdots + t_{n-1} + t_n$$

di termini t_i . Può essere solo raggruppata con $t_1 + t_2 + \cdots + t_{n-1}$ come un'espressione e t_n come un termine.

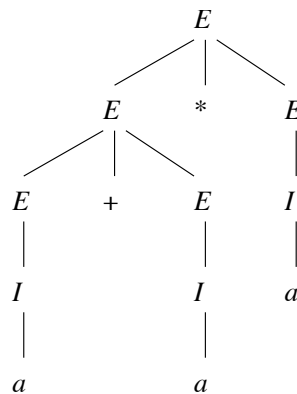


Derivazioni a sinistra e ambiguità

I due alberi sintattici per $a + a * a$



(a)



(b)

danno luogo a due derivazioni:

• $E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + E * E$
 $\quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm}$
 $\Rightarrow a + I * E \Rightarrow a + a * E \Rightarrow a + a * I \Rightarrow a + a * a$
 $\quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm}$
• $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow I + E * E \Rightarrow a + E * E$
 $\quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm}$
 $\Rightarrow a + I * E \Rightarrow a + a * E \Rightarrow a + a * I \Rightarrow a + a * a$
 $\quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm} \quad \quad \quad \text{lm}$

Grammatiche libere da contesto

In generale:

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

Grammatiche libere da contesto

Teorema 5.29: Data una CFG G , una stringa terminale w ha due distinti alberi sintattici se e solo se w ha due distinte derivazioni a sinistra dal simbolo iniziale.

Dimostrazione:

- (Solo se.) Se due alberi sintattici sono diversi, hanno un nodo dove sono state usate due diverse produzioni:
 $A \rightarrow X_1 X_2 \cdots X_k$ e $B \rightarrow Y_1 Y_2 \cdots Y_m$. Le corrispondenti derivazioni a sinistra useranno queste diverse produzioni e quindi saranno distinte.
- (Se.) Per come costruiamo un albero da una derivazione, è chiaro che due derivazioni distinte generano due alberi distinti.

Ambiguità inerente

Un CFL L è **inerentemente ambiguo** se **tutte** le grammatiche per L sono ambigue.

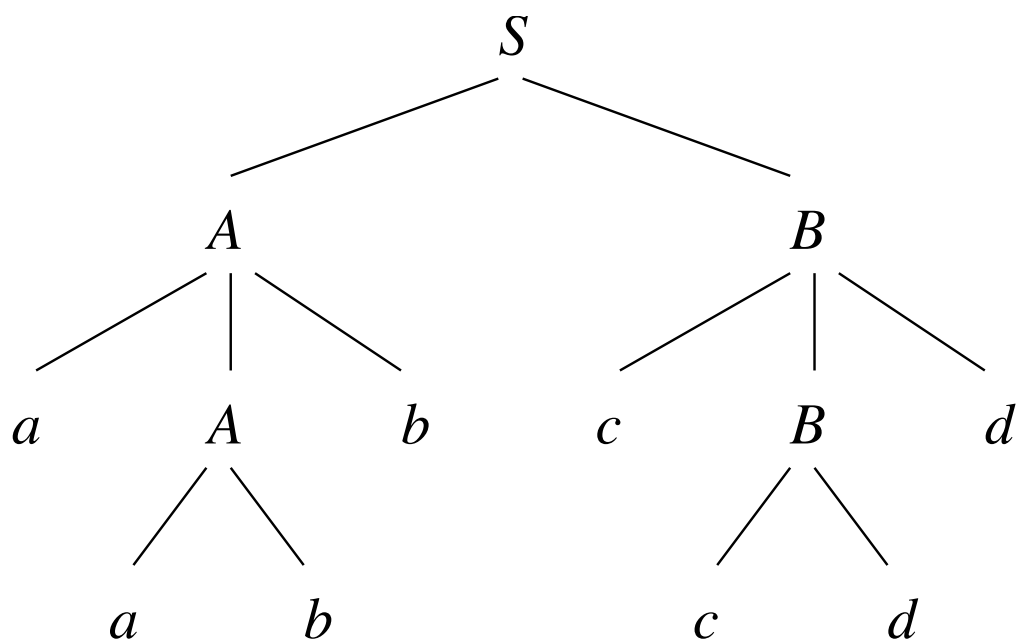
Esempio: Consideriamo $L =$

$$\{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}.$$

Una grammatica per L è

$$\begin{aligned} S &\rightarrow AB \mid C \\ A &\rightarrow aAb \mid ab \\ B &\rightarrow cBd \mid cd \\ C &\rightarrow aCd \mid aDd \\ D &\rightarrow bDc \mid bc \end{aligned}$$

Guardiamo la struttura sintattica della stringa *aabbccdd*.



Vediamo che ci sono due derivazioni a sinistra:

$$S \xRightarrow{lm} AB \xRightarrow{lm} aAbB \xRightarrow{lm} aabbB \xRightarrow{lm} aabbcBd \xRightarrow{lm} aabbccdd$$

e

$$S \xRightarrow{lm} C \xRightarrow{lm} aCd \xRightarrow{lm} aaDdd \xRightarrow{lm} aabDcdd \xRightarrow{lm} aabbccdd$$

Può essere provato che **ogni** grammatica per L si comporta come questa. Il linguaggio L è quindi inerentemente ambiguo.

Linguaggi regolari e grammatiche

- Un linguaggio regolare è anche libero da contesto.
- Da una espressione regolare, o da un automa, si può ottenere una grammatica che genera lo stesso linguaggio.

Da espressione regolare a grammatica

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 $S \rightarrow FS \mid \epsilon$

Esempio

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

Grammatica:

$$S \rightarrow ABC$$

$$A \rightarrow 0A \mid \epsilon$$

$$B \rightarrow 1$$

$$C \rightarrow DC \mid \epsilon$$

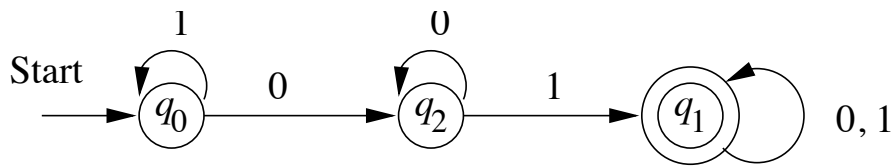
$$D \rightarrow 0 \mid 1$$

Da automa a grammatica

- Un simbolo non-terminale per ogni stato.
- Simbolo iniziale = stato iniziale.
- Per ogni transizione da stato s a stato p con simbolo a , produzione $S \rightarrow aP$.
- Se p stato finale, allora produzione $P \rightarrow \epsilon$

Esempio

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 è accettata dall'automa. Nella grammatica, ha la derivazione:

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

Navigation icons: back, forward, search, etc.

Grammatiche libere da contesto

Esercizi su minimizzazione e sulle grammatiche

- Sia L il linguaggio dato dall'intersezione dei seguenti linguaggi:
 - $L_1 = \{w \in \{0,1\}^* \mid \text{in } w \text{ ci siano almeno due } 0 \text{ consecutivi}\}$
 - $L_2 = \{w \in \{0,1\}^* \mid \text{in } w \text{ non ci siano due } 1 \text{ consecutivi}\}$

Costruire l'automa D per riconoscere L e minimizzarlo se necessario.

- Ideare la grammatica libera per generare i seguenti linguaggi:
 - $\{0^n 1^n \mid n \geq 1\}$
 - L'insieme di tutte le stringhe in $\{0,1\}^*$ tali che il numero di 0 sia il doppio del numero di 1.
 - L'insieme delle stringhe di parentesi bilanciate.

Navigation icons: back, forward, search, etc.

Grammatiche libere da contesto