Il linguaggio di una grammatica

- Se $G(V, T, P, S)$ è una CFG, allora il **linguaggio di** G è

$$L(G) = \{w \in T^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

  cioè l'insieme delle stringhe su T^* derivabili dal simbolo iniziale.
- Se G è una CFG, chiameremo $L(G)$ un **linguaggio libero da contesto**.
- Esempio: $L(G_{pal})$ è un linguaggio libero da contesto.
- Il linguaggio è visto come l'insieme delle stringhe generate dalla grammatica (approccio *generativo-sintetico*), mentre finora come l'abbiamo visto come l'insieme delle stringhe riconosciute o accettate dagli automi (approccio *riconoscitivo-analitico*).

Teorema 5.7:

$$L(G_{pal}) = \{w \in \{0,1\}^* : w = w^R\}$$

Dimostrazione: (direzione $\supseteq$) Supponiamo $w = w^R$. Mostriamo per induzione su $|w|$ che $w \in L(G_{pal})$.

- **Base:** $|w| = 0$, o $|w| = 1$. Allora w è $\epsilon, 0$, o 1. Dato che $P \to \epsilon, P \to 0$, e $P \to 1$ sono produzioni, concludiamo che $P \overset{*}{\underset{G}{\Rightarrow}} w$ in tutti i casi base.
- **Induzione:** Supponiamo $|w| \geq 2$. Dato che $w = w^R$, abbiamo $w = 0x0$, o $w = 1x1$, e $x = x^R$.
  Se $w = 0x0$ sappiamo che per l'ipotesi induttiva $P \overset{*}{\Rightarrow} x$.
  Allora
  $$P \Rightarrow 0P0 \overset{*}{\Rightarrow} 0x0 = w$$
  Quindi $w \in L(G_{pal})$.
  Il caso di $w = 1x1$ è simile.

(direzione $\subseteq$) Supponiamo che $w \in L(G_{pal})$ e dobbiamo mostrare che $w = w^R$.
Dato che $w \in L(G_{pal})$, abbiamo $P \stackrel{*}{\Rightarrow} w$.
Faremo un'induzione sulla lunghezza di $\stackrel{*}{\Rightarrow}$.

- **Base:** La derivazione $P \stackrel{*}{\Rightarrow} w$ ha 1 passo.
  Allora w deve essere $\epsilon, 0$, o 1, tutte palindromi.
- **Induzione:** Sia $n \geq 1$, e supponiamo che la derivazione abbia $n+1$ passi e che l'enunciato sia vero per tutte le derivazioni di n passi (se $P \stackrel{*}{\Rightarrow} x$ in n passi, allora $x = x^R$). Allora una derivazione di $n+1$ passi deve essere del tipo

$$w = 0x0 \stackrel{*}{\Leftarrow} 0P0 \leftarrow P \text{ oppure } w = 1x1 \stackrel{*}{\Leftarrow} 1P1 \leftarrow P$$

  dove la seconda derivazione di x ha n passi. Infatti $n+1 > 1$ e le produzioni $P \Rightarrow 0P0$ e $P \Rightarrow 1P1$ sono le uniche che permettono passi aggiuntivi.
  Per l'ipotesi induttiva, x è palindroma. Lo sarà quindi anche la stringa w.

Forme sentenziali

- Sia $G = (V, T, P, S)$ una CFG, e $\alpha \in (V \cup T)^*$.
- Se $S \overset{*}{\Rightarrow} \alpha$ diciamo che α è una **forma sentenziale** (*sentential form*).
- Se $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$ diciamo che α è una **forma sentenziale sinistra**,
- Se $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$ diciamo che α è una **forma sentenziale destra**
- Nota: $L(G)$ contiene le forme sentenziali (o *sentence*) che sono in T^*.

Esempi

- Prendiamo la G delle espressioni. Allora $E * (I + E)$ è una forma sentenziale perché

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

  Questa derivazione non è né a sinistra né a destra

- $a * E$ è una forma sentenziale sinistra, perché

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

- $E * (E + E)$ è una forma sentenziale destra, perché

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

Alberi sintattici

- 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 rappresentazione alternativa alle derivazioni e alle inferenze ricorsive.
- 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.

Costruzione di un albero sintattico

Sia $G = (V, T, P, S)$ una CFG. Un albero è un **albero sintattico** per G se:

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

$$X_1 X_2 \ldots X_k$$

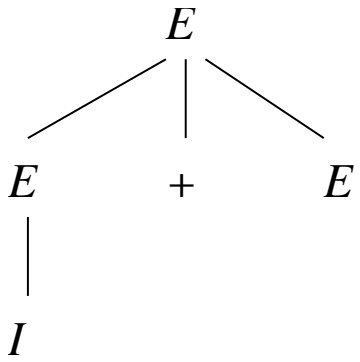
allora $A \to X_1 X_2 \ldots X_k \in P$.

Esempio

Nella grammatica

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

il seguente è un albero sintattico:



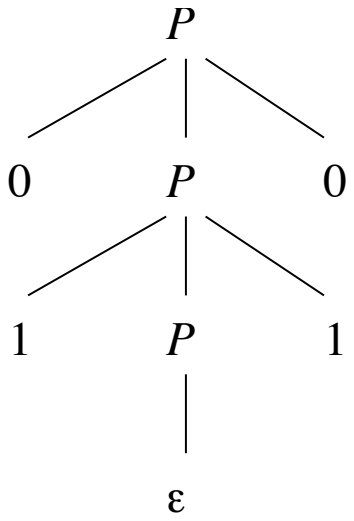
Questo albero sintattico mostra la derivazione $E \overset{*}{\Rightarrow} I + E$

Esempio

Nella grammatica

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

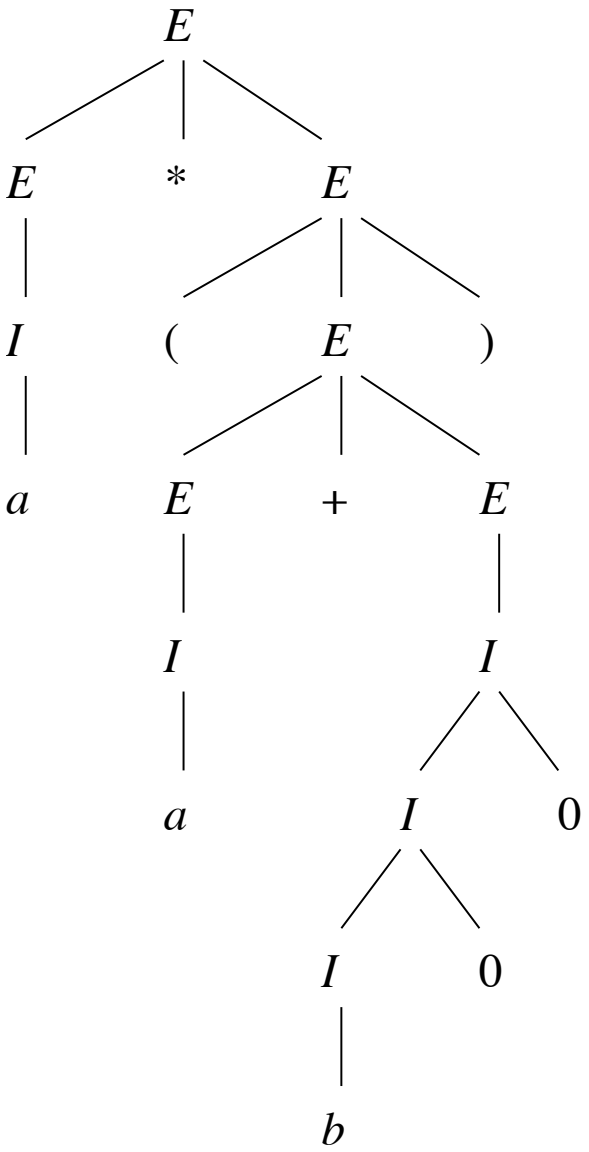
il seguente è un albero sintattico:



Il prodotto di un albero sintattico

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

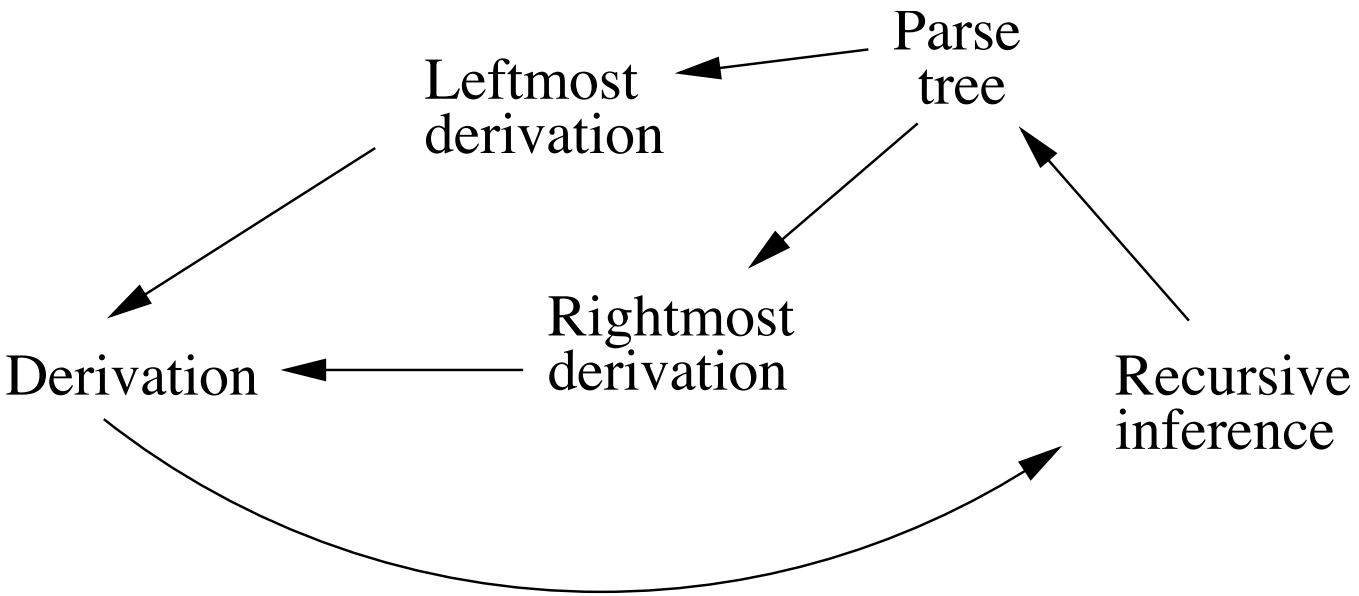
Esempio



Sia $G = (V, T, P, S)$ una CFG, e $A \in V$. I seguenti sono equivalenti:

1. Possiamo determinare per inferenza ricorsiva che w è nel linguaggio di A
2. $A \stackrel{*}{\Rightarrow} w$
3. $A \underset{lm}{\stackrel{*}{\Rightarrow}} w$, e $A \underset{rm}{\stackrel{*}{\Rightarrow}} w$
4. C'è un albero sintattico di G con radice A e prodotto w.

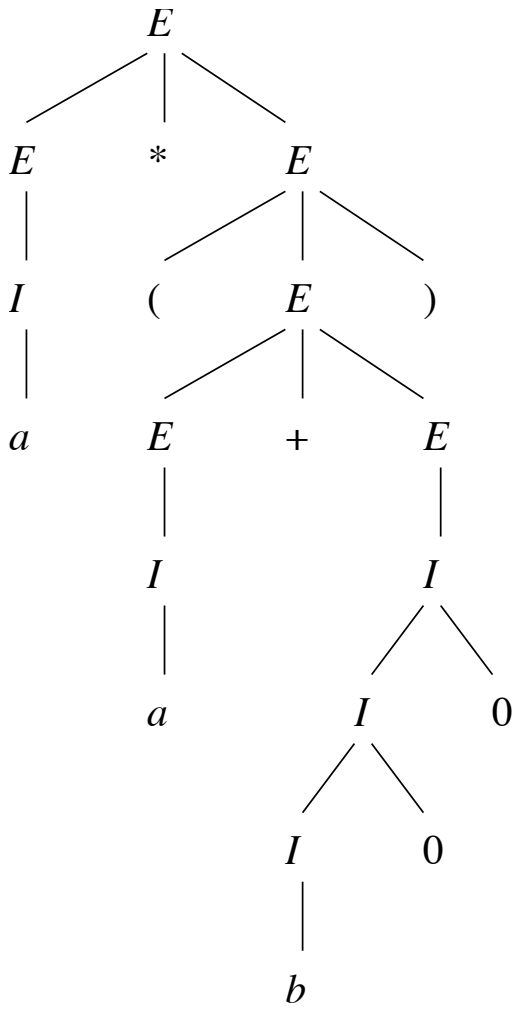
Per provare l'equivalenza, usiamo il seguente piano.



- Dalle inferenze ricorsive agli alberi: partiamo dall'ultimo passo e costruiamo sottoalberi, poi passiamo ai passi precedenti fino al primo.
- Dagli alberi alle derivazioni a sinistra: seguo l'albero da sinistra a destra.
- Dalle derivazioni alle inferenze ricorsive: vado all'indietro nella derivazione.

Esempio

Costruiamo la derivazione a sinistra per l'albero



Supponiamo di aver induttivamente costruito la derivazione a sinistra

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

corrispondente al sottoalbero più a sinistra, e la derivazione a sinistra

$$E \underset{lm}{\Rightarrow} (E) \underset{lm}{\Rightarrow} (E + E) \underset{lm}{\Rightarrow} (I + E) \underset{lm}{\Rightarrow} (a + E) \underset{lm}{\Rightarrow}$$

$$(a + I) \underset{lm}{\Rightarrow} (a + I0) \underset{lm}{\Rightarrow} (a + I00) \underset{lm}{\Rightarrow} (a + b00)$$

corrispondente al sottoalbero più a destra.

Per la derivazione corrispondente all'intero albero, iniziamo con $E \underset{lm}{\Rightarrow} E * E$ e espandiamo la prima E con la prima derivazione e la seconda E con la seconda derivazione:

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow}$$
$$I * E \underset{lm}{\Rightarrow} a * E \underset{lm}{\Rightarrow}$$
$$a * (E) \underset{lm}{\Rightarrow} a * (E + E) \underset{lm}{\Rightarrow}$$
$$a * (I + E) \underset{lm}{\Rightarrow} a * (a + E) \underset{lm}{\Rightarrow}$$
$$a * (a + I) \underset{lm}{\Rightarrow} a * (a + I0) \underset{lm}{\Rightarrow}$$
$$a * (a + I00) \underset{lm}{\Rightarrow} a * (a + b00)$$

Ambiguità in Grammatiche e Linguaggi

Nella grammatica

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

$\cdots$

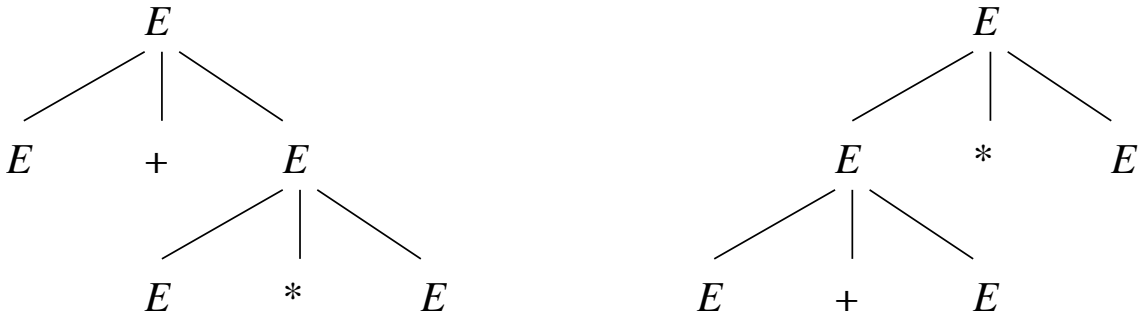
la forma sentenziale $E + E * E$ ha due derivazioni:

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

e

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

Questo ci dà due alberi sintattici:



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

la stringa $a + b$ ha varie derivazioni:

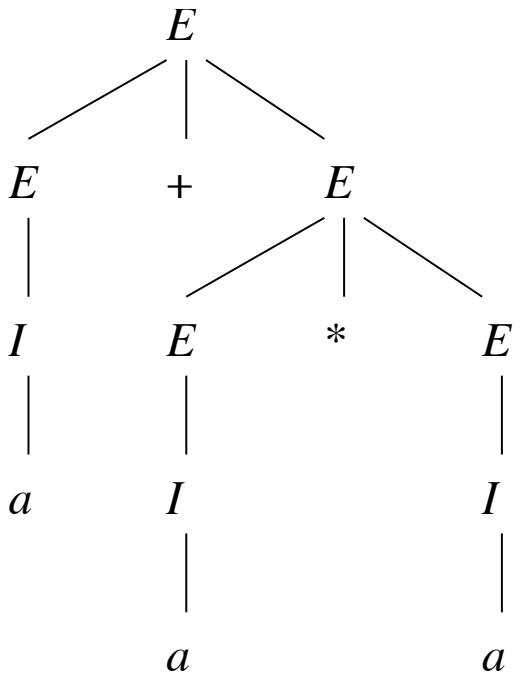
$$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$

e

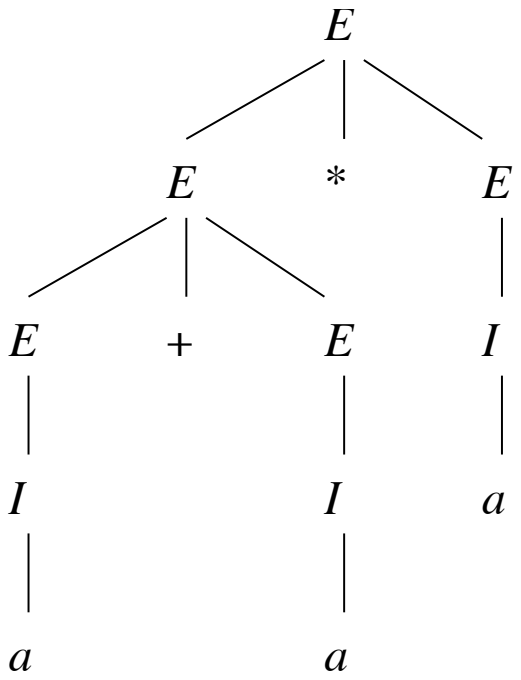
$$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$$

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

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:



(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

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

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

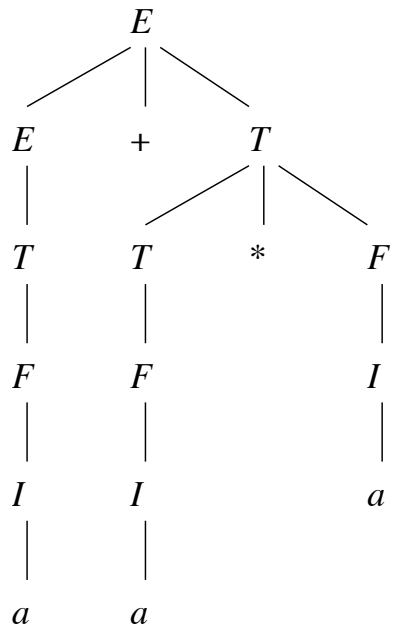
  - Non c'è precedenza tra * e +
  - Non c'è raggruppamento di sequenze di operatori: $E + E + E$ è inteso come $E + (E + E)$ o come $(E + E) + E$?

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

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

1. $I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
2. $F \to I \mid (E)$
3. $T \to F \mid T * F$
4. $E \to T \mid E + T$



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

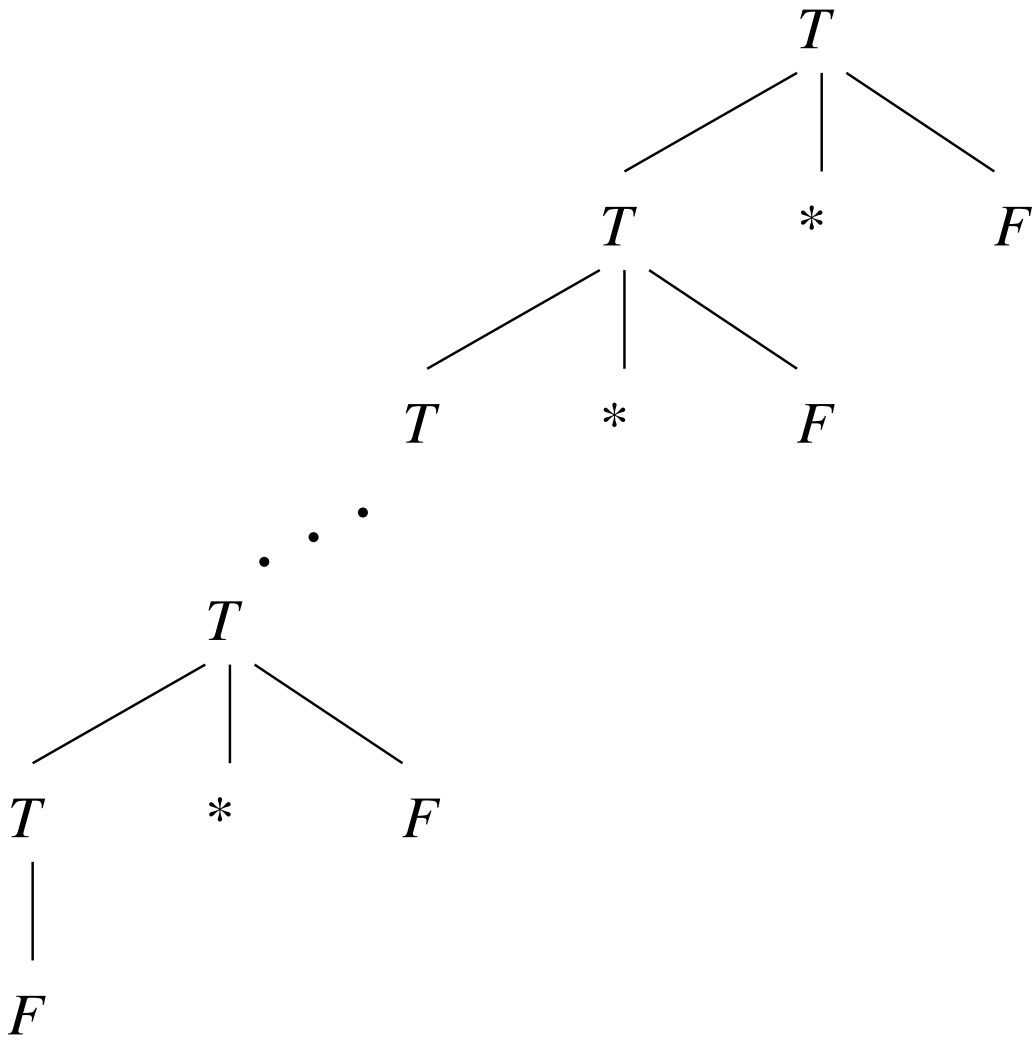
$$f_1 * f_2 * \cdots * f_{n-1} * f_n$$

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

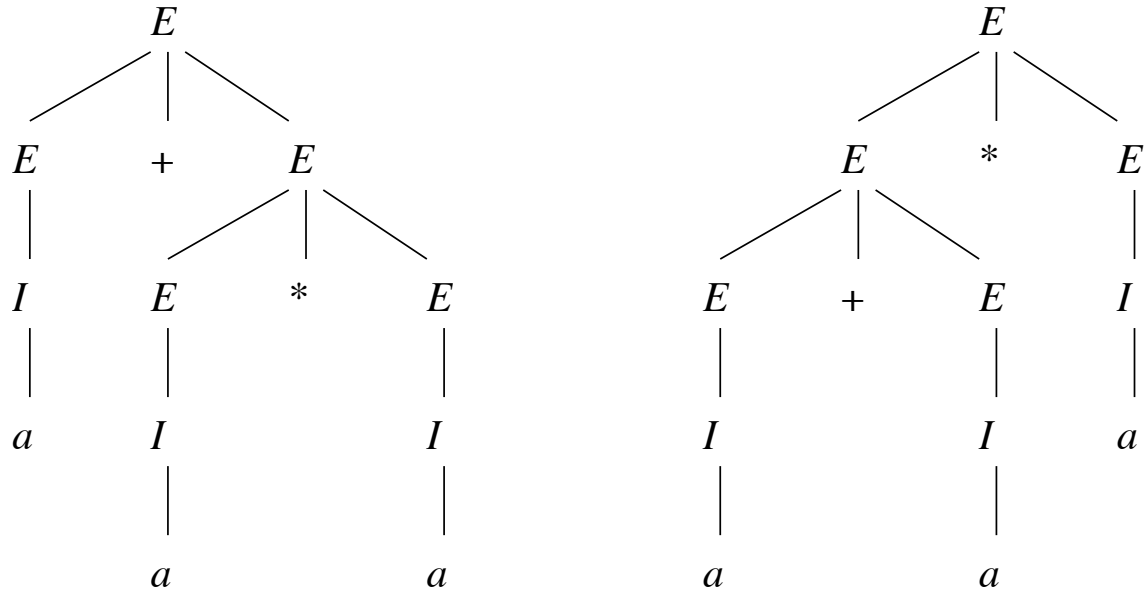
$$t_1 + t_2 + \cdots + t_{n-1} + t_n$$

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.


T
T * F
T * F
T
T * F
F

Derivazioni a sinistra e ambiguità

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



(a) (b)

danno luogo a due derivazioni:

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

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, 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: $A \to X_1 X_2 \cdots X_k$ e $B \to Y_1 Y_2 \cdots Y_m$. 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

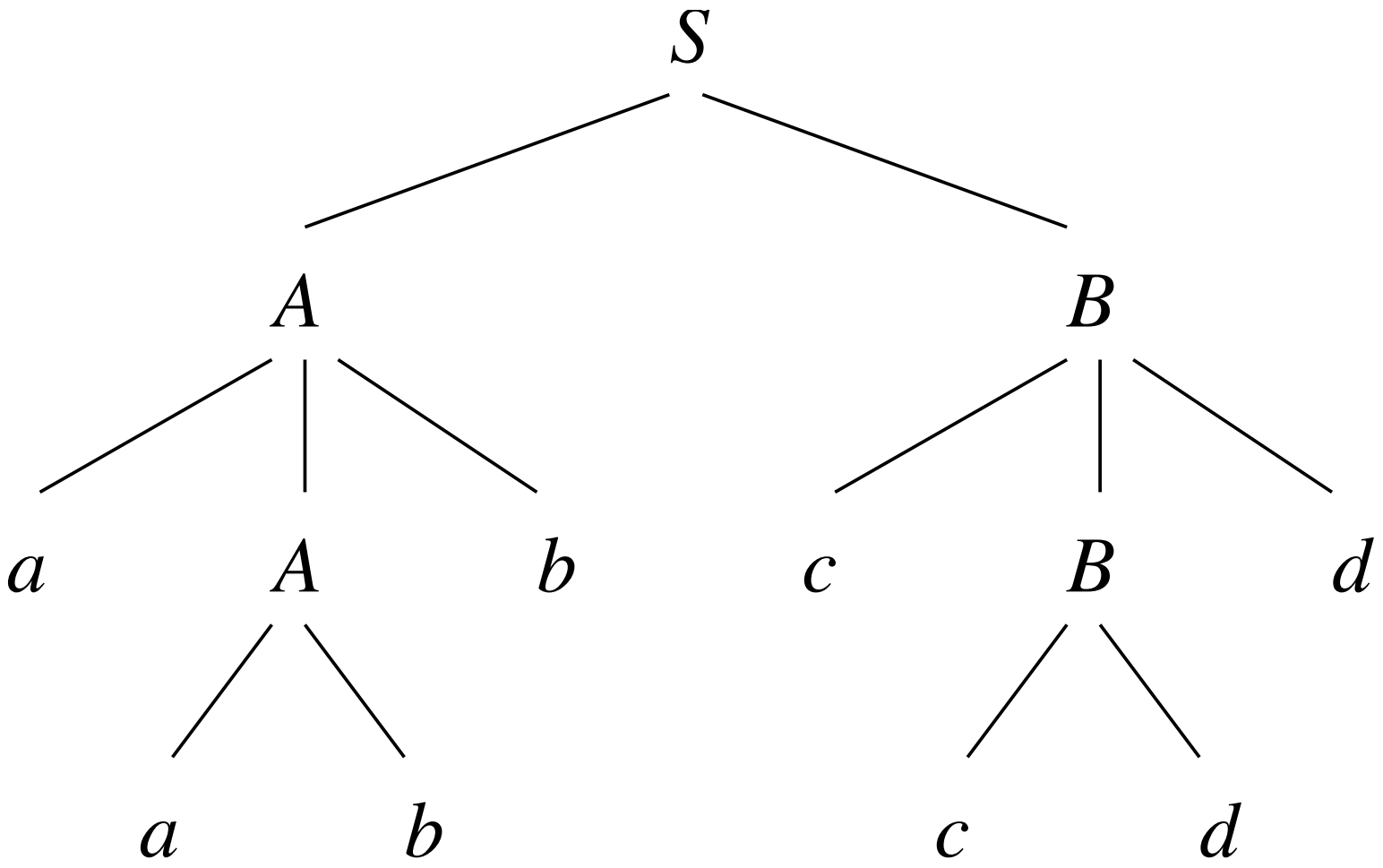
Un CFL L è **inerentemente ambiguo** se **tutte** le grammatiche per L sono ambigue.
Esempio: Consideriamo $L =$

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

Una grammatica per L è

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

Guardiamo la struttura sintattica della stringa *aabbccdd*.



Vediamo che ci sono due derivazioni a sinistra:

$$S \underset{lm}{\Rightarrow} AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aabbB \underset{lm}{\Rightarrow} aabbcBd \underset{lm}{\Rightarrow} aabbccdd$$

e

$$S \underset{lm}{\Rightarrow} C \underset{lm}{\Rightarrow} aCd \underset{lm}{\Rightarrow} aaDdd \underset{lm}{\Rightarrow} aabDcdd \underset{lm}{\Rightarrow} aabbccdd$$

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.
- Non è vero il viceversa. Esistono infatti linguaggi liberi che non sono regolari (Ad es: L_{01}).
- Un linguaggio regolare può essere espresso da una grammatica libera, ma anche da una grammatica *regolare*. In una grammatica **regolare**, le produzioni hanno un vincolo non presente nelle grammatiche libere, cioè sono della forma:

$$S \rightarrow v$$

dove S è un simbolo non terminale in e v è una stringa composta da un terminale o da un terminale e un non terminale.
Esempio: grammatica regolare che genera $\mathbf{a}^*$.

$$S \rightarrow \epsilon | aS$$

Linguaggi regolari e grammatiche

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

Attenzione: ciò che otteniamo è una grammatica **libera** per il corrispondente linguaggio regolare.

Esempio

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

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

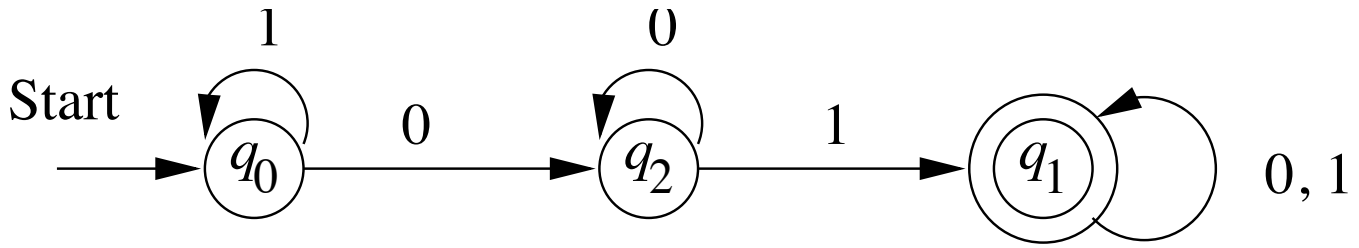
Da automa a grammatica

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

Attenzione: ciò che otteniamo è una grammatica **regolare** per il corrispondente linguaggio regolare.

Esempio

Automa:



Grammatica:

$$Q_0 \to 1Q_0 \mid 0Q_2$$
$$Q_2 \to 0Q_2 \mid 1Q_1$$
$$Q_1 \to 0Q_1 \mid 1Q_1 \mid \epsilon$$

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

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

Esercizi sulle grammatiche

Ideare la grammatica libera per generare i seguenti linguaggi:

- $\{0^n 1^n \ n \geq 1\}$
- 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.