Automi e Linguaggi Formali

Grammatiche libere da contesto
– Ambiguita' –

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

A.A. 2014-2015
Enrico Mezzetti
emezzett@math.unipd.it

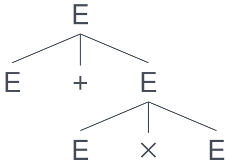
Ambiguita' in Grammatiche e Linguaggi

- Non tutte le CFG generano strutture univoche
  - Diverse regole applicabili $\rightarrow$ diversi alberi sintattici
- E.g., nella CFG delle espressioni G_{PL}

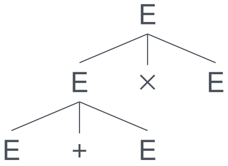
$$E \rightarrow I \mid E + E \mid E \times E \mid (E)$$

  - La forma sentenziale $E + E \times E$ ha due derivazioni

$$E \Rightarrow E + E \Rightarrow E + E \times E$$



$$E \Rightarrow E \times E \Rightarrow E + E \times E$$



Ambiguita' in Grammatiche e Linguaggi - 2

- Esistenza di derivazioni distinte $\not\Rightarrow$ ambiguita'
  - Molteplici **alberi sintattici** $\Rightarrow$ ambiguita'
- E.g., indicatori nella grammatica delle Espressioni G_{PL}

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

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

  - La stringa $a + b$ ha varie derivazioni

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

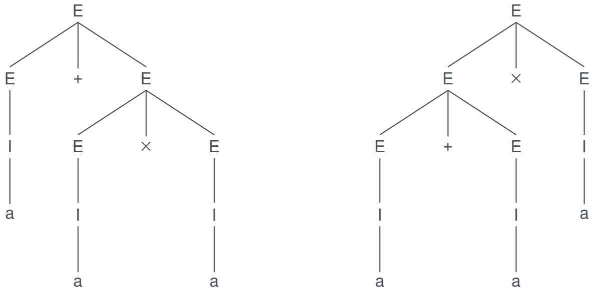
e

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

  - L'albero sintattico e' *univoco*
    $\Rightarrow$ la struttura di $a + b$ non e' ambigua

Ambiguita' in Grammatiche e Linguaggi - 2

- Una CFG $G = (V, T, P, S)$ e' **ambigua** se esiste almeno una stringa $w \in T^*$ che ha piu' di un albero sintattico.
  - Se ogni stringa in $L(G)$ ha solo un albero sintattico la CFG e' **non ambigua**
- E.g., la stringa terminale $a + a \times a$ ha due alberi sintattici



Rimuovere l'ambiguita' dalle grammatiche

- ☺ A volte possiamo rimuovere l'ambiguita'
- ☹ Ma non c'e' nessun algoritmo per farlo in modo sistematico
- ☹ ...e alcuni CFL hanno solo CFG ambigue
- Due cause principali di ambiguita'

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

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

  - Non c'e' precedenza tra operatori diversi ($\times$ e $+$)
  - Non c'e' precedenza tra sequenze dello stesso operatore: $E + E + E$ e' inteso come $E + (E + E)$ o come $(E + E) + E$?
    - Per associativita' sono lecite entrambe

Rimuovere l'ambiguita' dalle grammatiche - 2

- Come rimuovere l'ambiguita' (ove possibile)
- Introduciamo variabili aggiuntive (*fattori* e *termini*)
  - Rappresentare espressioni con lo stesso grado di "forza di legamento"

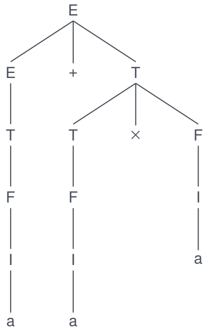
1. **Fattore:** espressione che non puo' essere spezzata da un $\times$ o un $+$ adiacente.
   - In G_{PL} *identificatori* ed *espressione racchiuse tra parentesi*
2. **Termine:** espressione che non puo' essere spezzata da un $+$.
   - E.g., $a \times b$ puo' essere spezzata da $a1 \times a \times b \equiv (a1 \times a) \times b$
     Non puo' essere spezzata da $+$
     In G_{PL}, $a1 + a \times b \equiv a1 + (a \times b)$
3. Il resto sono semplici espressioni che possono essere spezzate con $\times$ o $+$.

Esempio

- Usiamo F per i fattori, T per i termini, e E per le espressioni.

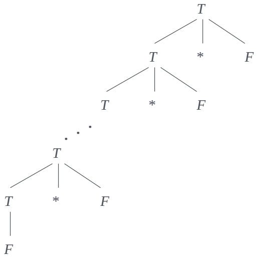
1. $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
2. $F \rightarrow I \mid (E)$
3. $T \rightarrow F \mid T * F$
4. $E \rightarrow T \mid E + T$

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

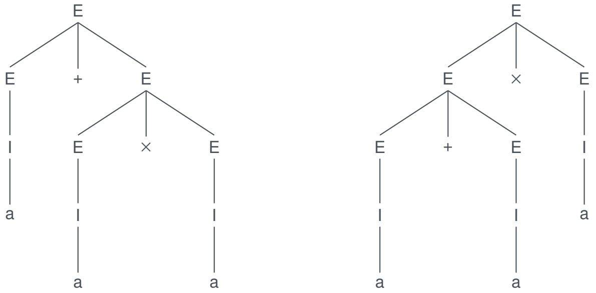


Esempio - 2

- Osservazioni sulla *non-ambiguita'*
  - Qualsiasi stringa da T e' una sequenza di fattori F legati da $\times$
    F e' un identificatore I o (E), per qualche espressione E.
  - L'unico albero per una sequenza $f_1 \times f_2 \times \cdots \times f_{n-1} \times f_n$ e' quello che contiene $f_1 \times f_2 \times \cdots \times f_{n-1}$ come termine e f_n come fattore
  - Un'espressione e' una sequenza di termini T legati da $+$
    Una sequenza $t_1 + t_2 + \cdots + t_{n-1} + t_n$ di termini t_i puo' 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 ambiguita'



- Alberi sintattici per $a + a \times a$ e derivazioni a sx

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

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

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

Prova:

Solo se: Se due alberi sintattici sono diversi, hanno un nodo dove sono state usate due diverse produzioni:

$$A \rightarrow X_1 X_2 \cdots X_k \text{ e } B \rightarrow Y_1 Y_2 \cdots Y_m$$

Le corrispondenti derivazioni a sinistra useranno queste diverse produzioni e quindi saranno distinte.

Se: Per come costruiamo un albero da una derivazione, e' chiaro che due derivazioni distinte generano due alberi distinti.

Ambiguita' inerente

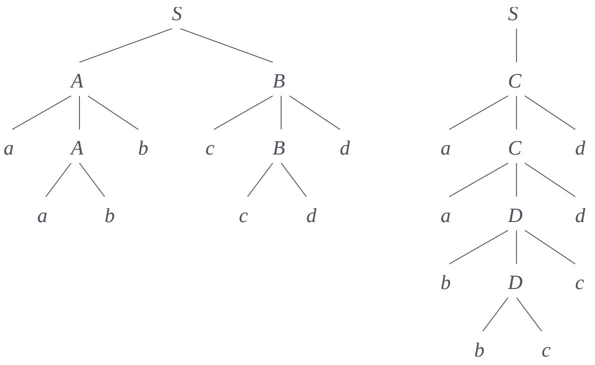
- Un CFL L e' *inerentemente ambiguo* se **tutte** le grammatiche per L sono ambigue.
- Esempio

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

  - Grammatica intuitiva usa inisemi di separazione. Una grammatica per L e'

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

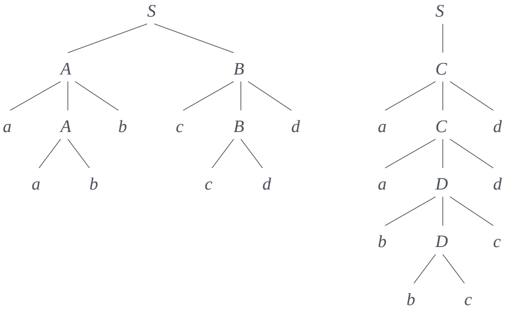
- Osserviamo la struttura sintattica della stringa *aabbccdd*



(a) (b)

$$S \rightarrow AB \mid C$$
$$A \rightarrow aAb \mid ab$$
$$B \rightarrow cBd \mid cd$$
$$C \rightarrow aCd \mid aDd$$
$$D \rightarrow bDc \mid bc$$

- Esistono due derivazioni a sx. distinte



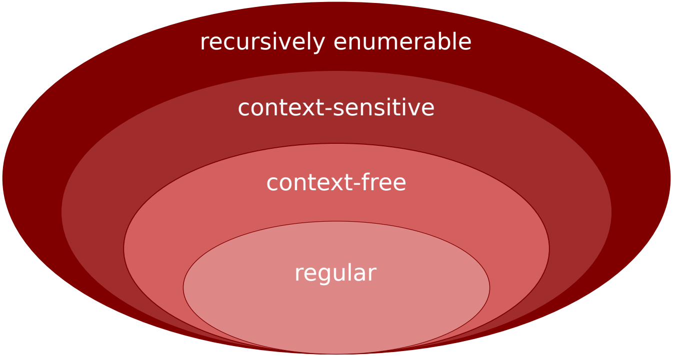
(a) (b)

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

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

- Puo' essere provato che **ogni** grammatica per L si comporta come questa
  - Il linguaggio L e' quindi inerentemente ambiguo

Linguaggi regolari e CFG



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

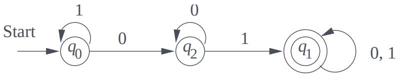
Da espressione regolare a grammatica

- 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: $\mathtt{0}^*\mathtt{1}(\mathtt{0} + \mathtt{1})^*$
  - Produzioni:

$$
\begin{aligned}
S &\rightarrow ABC \\
A &\rightarrow 0A \mid \epsilon \\
B &\rightarrow 1 \\
C &\rightarrow DC \mid \epsilon \\
D &\rightarrow 0 \mid 1
\end{aligned}
$$

Da automa a grammatica

- Pe rogni elemento nella quintupla di un FA
  - 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



  - Grammatica:

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

  - Derivazione per 1101 (accettata dall'automa)

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