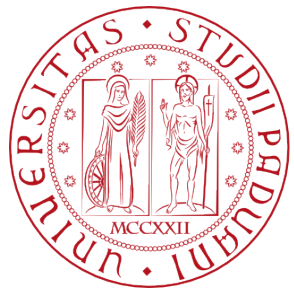
Automi e Linguaggi Formali

Grammatiche libere da contesto
– Alberi sintattici –

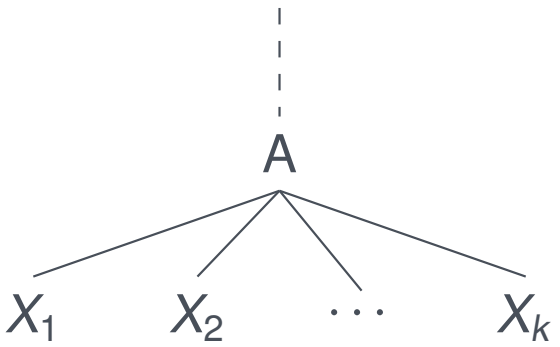


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

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

Costruzione di un albero sintattico

- Sia CFG $G = (V, T, P, S)$
- Gli **alberi sintattici** per G soddisfano le seguenti condizioni:
  1. Ogni nodo interno e' etichettato con una variabile in V
  2. Ogni foglia e' etichettata con un simbolo in $V \cup T \cup \{\epsilon\}$.
     (Una 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 $X_1, X_2, \ldots, X_k$, allora
     $$A \to X_1 X_2 \ldots X_k \in P$$

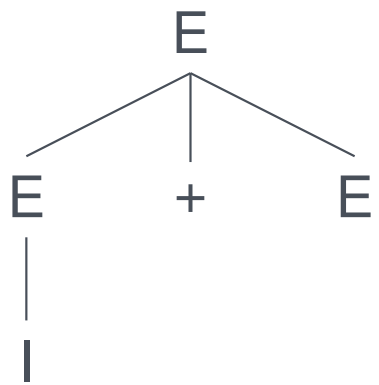


$$\Longrightarrow \qquad P \ni A \to X_1 X_2 \cdots X_k$$

Esempio

- Grammatica delle espressioni $G_{PL} = (\{E, I\}, T, P, E)$
- L'insieme delle produzioni P e' definito
  - $E \rightarrow I \mid E + E \mid E \times E \mid (E)$
  - $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

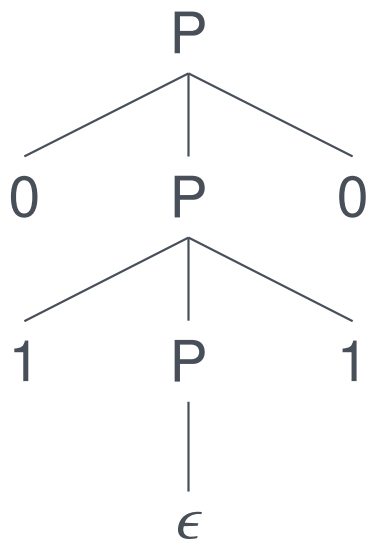
- Albero sintattico per $E \overset{*}{\Rightarrow} I + E$ in G_{PL}



Esempio - 2

- Grammatica delle palindrome $G_{pal} = (\{P\}, \{0, 1\}, A, P)$
- L'insieme delle produzioni P e' definito
  - $P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 0P1$

- Albero sintattico per $P \overset{*}{\Rightarrow} 0110$ in G_{pal}



Il prodotto di un albero sintattico

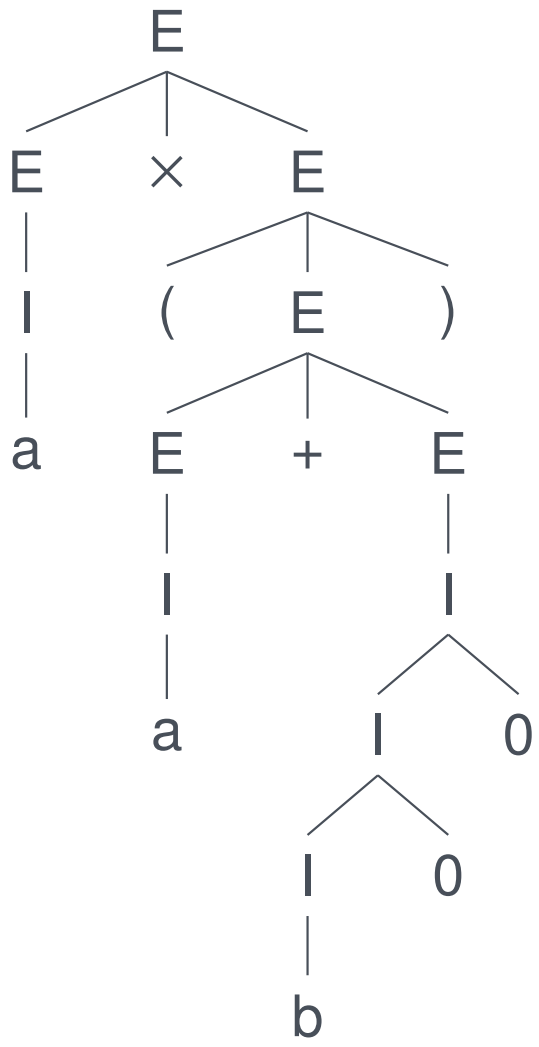
- Il **prodotto** di un albero sintattico e' la stringa ottenuta dal *concatenamento* delle foglie da sinistra a destra
  - E' una stringa *derivabile* dalla variabile radice

- Sono rilevanti gli alberi sintattici in cui:
  1. Il prodotto e' una *stringa terminale*
  2. La radice e' etichettata dal *simbolo iniziale*

- L'insieme dei prodotti di questi alberi sintattici rappresenta esattamente il linguaggio della grammatica

Esempio

- $G_{PL} = (\{E, I\}, T, P, E)$
  - Root e' E (iniziale)
  - Foglie sono *terminali*
- Prodotto $a \times (a + b00)$



Equivalenza tra inferenza, derivazioni e alberi

- Data una CFG $G = (V, T, P, S)$ e una variabile $A \in V$, i seguenti enunciati sono equivalenti:
  1. Per inferenza ricorsiva si puo' stabilire che $w \in L(A)$
  2. $A \overset{*}{\Rightarrow} w$
  3. $A \underset{lm}{\overset{*}{\Rightarrow}} w$ e $A \underset{rm}{\overset{*}{\Rightarrow}} w$
  4. Esiste un albero sintattico con radice A e' prodotto w.
- Per provare l'equivalenza, usiamo il seguente piano

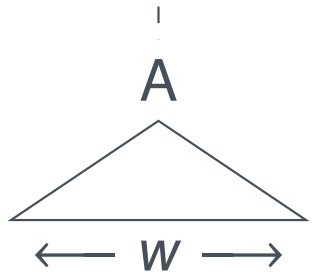
- 2–4 valgono per qualsiasi forma sentenziale

Da inferenze ad alberi

Th. 5.12 *Sia $G = (V, T, P, S)$ una CFG. Se la procedura di inferenza ricorsiva indica che la stringa terminale W e'nel linguaggio della variabile A, allora esiste un albero sintattico con radice A e prodotto w.*

Prova: per induzione sul numero di passi dell'inferenza

Base (un solo passo) $\rightarrow$ solo *base* della procedura di inferenza
Esiste una produzione $A \rightarrow w$ (incluso caso $w = \varepsilon$)



Da inferenze ad alberi - 2

Induz. (n+1 passi) $\rightarrow$ per ip.induttiva vale per inferenze di n passi

Consideriamo l'ultimo passo dell'inferenza

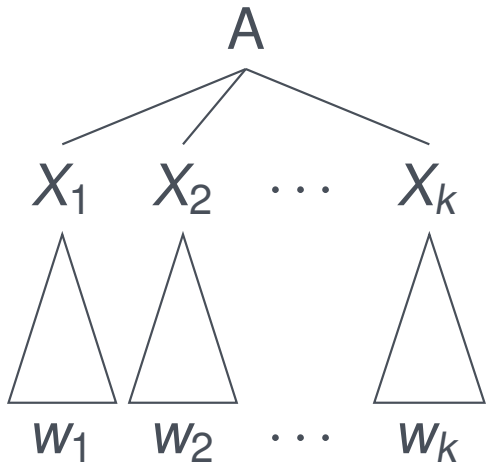
$$A \rightarrow X_1 X_2 \cdots X_k \quad dove\ X_i \in \{V \cup T\}$$

Scomponiamo w in $w_1 w_2 \cdots w_k$:

- Se X_i e' terminale allora $w_i = X_i$
- Se X_i e' variabile allora $w_i \in L(X_i)$ (in n passi)

Possiamo costruire un albero con radice A e prodotto w

- Prodotto determinato dalla concatenazione dei prodotti dei sub-tree



Da alberi a derivazioni

- Costruire una derivazione (lm o rm)
- *Incorporazione* di derivazioni
- Se esiste una derivazione

$$E \Rightarrow I \Rightarrow Ib \Rightarrow ab$$

  allora vale anche

$$\alpha E\beta \Rightarrow \alpha I\beta \Rightarrow \alpha Ib\beta \Rightarrow \alpha ab\beta \quad \forall \alpha,\ \beta$$

- In altre parole, possiamo applicare le stesse produzioni *indipendentemente dal contesto* (*context-free*)
- Permette di passare da albero a derivazione

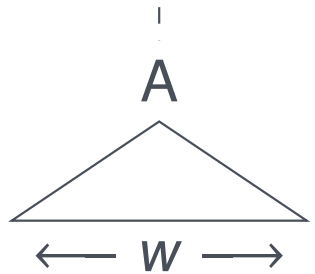
Da alberi a derivazioni - 2

Th. 5.14: *Sia $G = (V, T, P, S)$ una CFG, e supponiamo che essista un albero sintattico con radice etichettata con la variabile A e prodotto w con $w \in T^*$. Allora esiste una derivazione a sinistra $A \underset{lm}{\overset{*}{\Rightarrow}} w$ (e anche a destra) in G.*

Prova: Per induzione sull'altezza dell'albero.

Base (Altezza 1) Albero di altezza minima

- Una radice e figli che formano w (ds sx. a dx.)
- Deve esistere una produzione $A \to w$
- Quindi esiste anche una derivazione a sx $A \underset{lm}{\Rightarrow} w$ di un solo passo di w da A

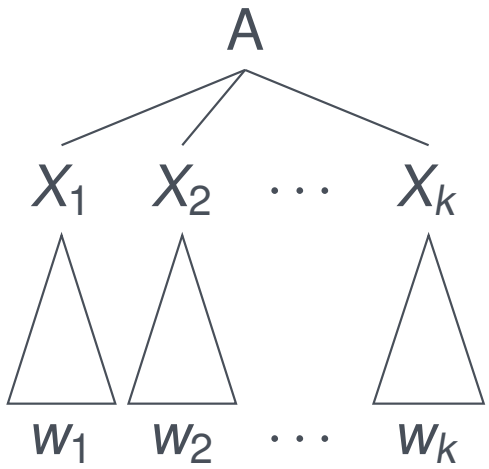


Da alberi a derivazioni - 3

Induz. (Altezza n>1)

Una radice A con figli etichettati $X_1 X_2 \cdots X_k$ *dove* $X_i \in \{V \cup T\}$

- Se X_i e' terminale allora $w_i = X_i$
- Se X_i e' variabile allora X_i e' radice di sottolabero con prodotto w_i (e di altezza n)



- Costruiamo una derivazione a sx. partendo da $A \underset{lm}{\Rightarrow} X_1 X_2 \cdots X_k$ e proviamo

  $\forall i \in 1, 2, \ldots, k \; A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \cdots w_i X_{i+1} X_{i+2} \cdots X_k$

- Per induzione su i $\quad A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \cdots w_{i-1} X_i X_{i+1} \cdots X_k$

  i=0 $\quad A \underset{lm}{\Rightarrow} X_1 X_2 \cdots X_k$ (dato)

  i≥1 Se X_i e' terminale $\rightarrow A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \cdots w_i X_{i+1} \cdots X_k$

  Altrimenti $A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \cdots w_{i-1} \alpha X_{i+1} \cdots X_k$ con $X_i \underset{lm}{\Rightarrow} \alpha_1 \cdots \underset{lm}{\Rightarrow} w_i$

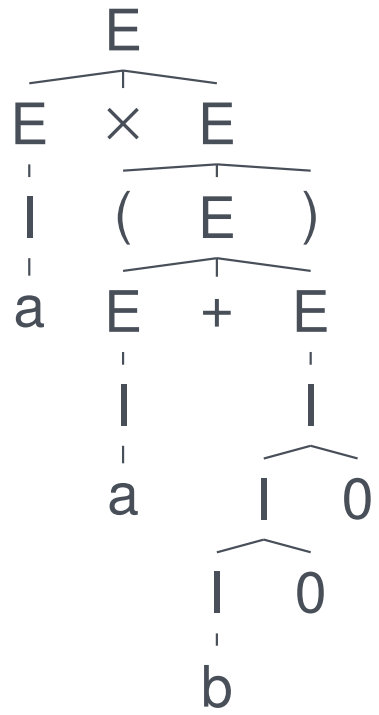
  i=k Abbiamo terminato

Esempio

- Costruiamo la derivazione a sinistra ($\underset{lm}{\Rightarrow}$) per l'albero

  - Supponiamo di aver induttivamente costruito la derivazione a sinistra

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



e la derivazione a sinistra

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

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

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

Esempio - 2

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

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

Applicazioni delle CFG

- Chomsky → descrivere il *linguaggio naturale* ✘
  - *Terminali*: horse, dog, cat, saw, heard, the
  - *Variabili*: <sentence>, <subject>, <verb>, <object>
  - *Variabile iniziale* : <sentence>
  - *Regole di produzione*:

<sentence>	⇒	<subject> <verb> <object>
<subject>	⇒	the horse
<subject>	⇒	the dog
<subject>	⇒	the cat
<object>	⇒	the horse
<object>	⇒	the dog
<object>	⇒	the cat
<verb>	⇒	saw
<verb>	⇒	heard

  - *Sentenziali* (solo terminali): ”the horse saw the dog”, ”the dog heard the cat”, ”the cat saw the horse”

Applicazioni delle CFG - 2

- Descrivere *concetti ricorsivi* in informatica ✓
  - Descrizione di linguaggi di programmazione
  - Trasformazione automatica da CFG a Parser
  - XML (DTD) per la descrizione di tag legali e relazioni
- CFG (abbreviata) per il linguaggio C
  - *Terminali*: `if do while for switch break continue typedef struct return main int long char float double void static ;( ) a b c A B C 0 1 2 + * - / _ # include += ++ ...`
  - *Variabili*: `<stmt> <expression> <C source file> <identifier> <digit> <nondigit> <identifier> <selection-stmt> <loop-stmt>`
  - *Variabile iniziale*: `<C source file>`

Applicazioni delle CFG - 3

- *Regole di produzione*:

```
<identifier>   ⇒  <nondigit>
               ⇒  <identifier> <nondigit>
               ⇒  <identifier> <digit>
  <nondigit>   ⇒  a|b|...|Y|Z|_
    <digit>    ⇒  0|1|2|3|4|5|6|7|8|9
 <expression>  ⇒  <identifier>|<constant>
               ⇒  <cond-expression>|<assign-expression>
<cond-expression>  ⇒  <expression> > <expression>
               ⇒  <expression> != <expression>
<assign-expression>  ⇒  <expression> = <expression>
               ⇒  <expression> += <expression>
      <stmt>   ⇒  <select-stmt>|<loop-stmt>
               ⇒  <compound-stmt>|<express-stmt>
<select-stmt>  ⇒  if (<expression>)
               ⇒  if (<expression>)<stmt> else <stmt>
  <loop-stmt>  ⇒  while (<expression>) <stmt>
               ⇒  do <stmt> while (<expression>)
<express-stmt>  ⇒  <expression> ;
```