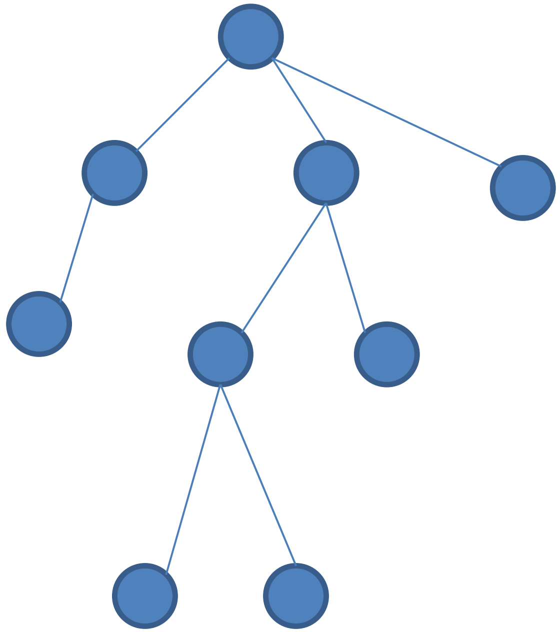
Lezione 2

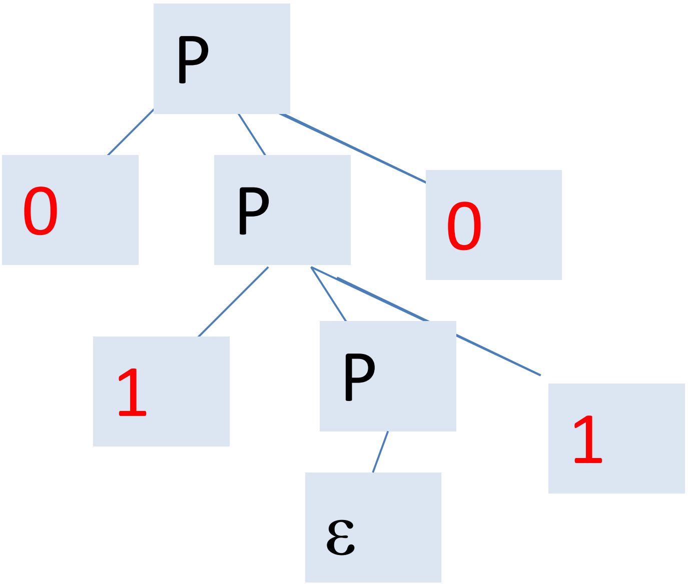
alberi sintattici



radice
nodo interno
foglia

padre
figli ordinati
discendente
frontiera

P -> ε | 0 | 1 | 0P0 | 1P1



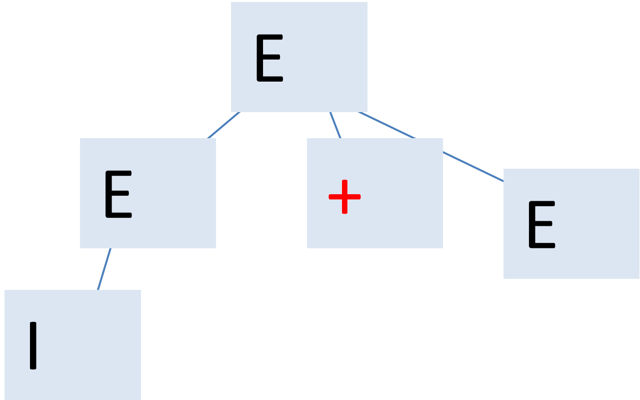
prodotto = 0110

alberi sintattici

data G=(V,T,R,S)

un albero sintattico di G soddisfa:

1) ciascun nodo interno è etichettato da una variabile
2) ciascuna foglia è etichettata da variabile o terminale o ε, in quest'ultimo caso deve essere l'unico figlio
3) se un nodo interno è etichettato A e i suoi figli (da sinistra a destra) sono X1...X2, allora A -> X1...Xn è in R.

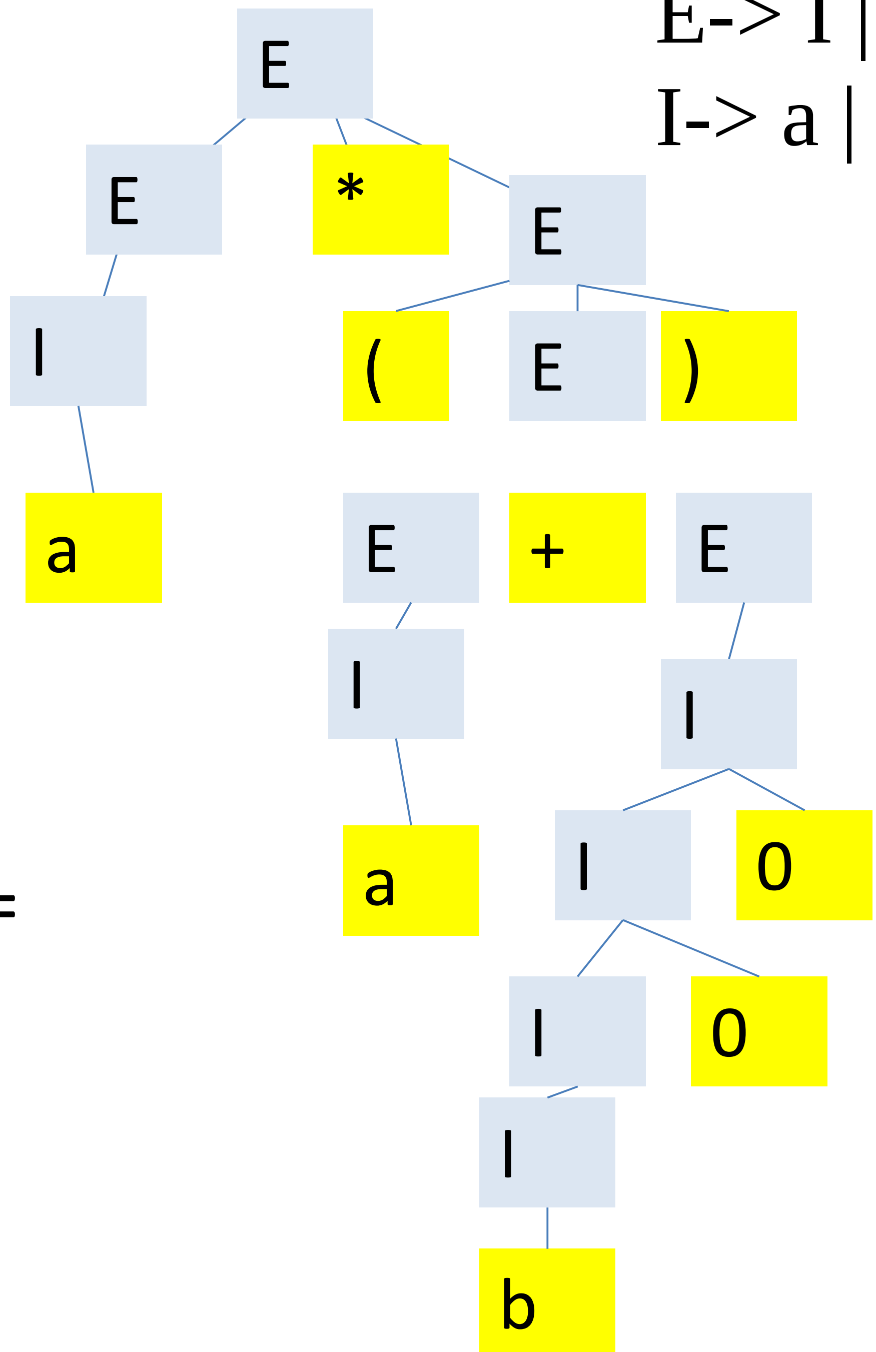


E-> I | E+E | E*E | (E)

I-> a | b | Ia | Ib | I0 | I1

E-> I | E+E | E*E | (E)

I-> a | b | Ia | Ib | I0 | I1



a*(a+b00) =
prodotto
dell'albero

abbiamo visto molti modi di caratterizzare il funzionamento delle grammatiche.

Sono:

--inferenza ricorsiva che stabilisce che w è nel linguaggio della variabile A

-- $A \Rightarrow^* w$

--$A \Rightarrow^{*lm} w$

--$A \Rightarrow^{*rm} w$

-- albero sintattico con radice A e prodotto w

sono tutte equivalenti


derivazione a destra
albero sintattico
inferenza ricorsiva
derivazione a sinistra
derivazione

Teorema 5.12 Sia G=(V,T,R,S) una CFG. Se la procedura di inferenza ricorsiva indica che la stringa terminale w è nel linguaggio della variabile A, allora esiste un albero sintattico con radice A e prodotto w.

Dimostrazione per induzione sul numero di passi dell'inferenza ricorsiva.
Base: 1 passo
deve esistere in R, A -> w e quindi esiste anche l'albero sintattico desiderato.

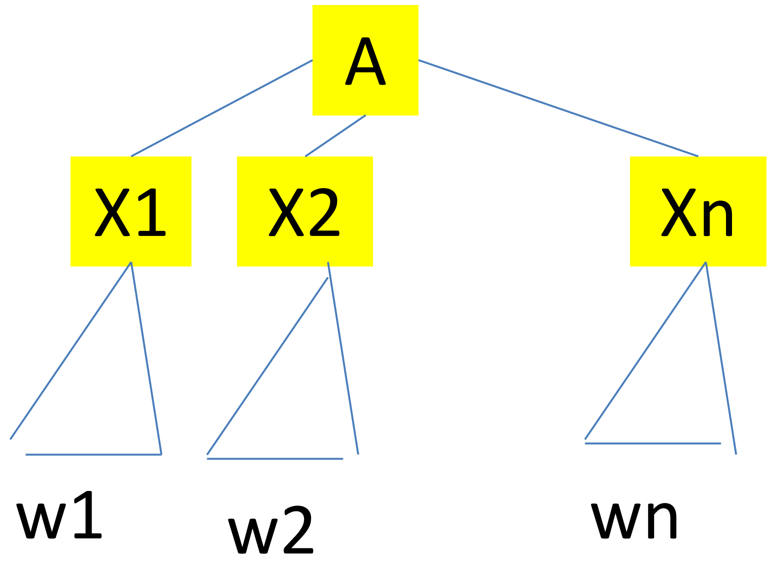
passo induttivo: supponiamo di aver usato n+1 passi di inferenza, e che l'enunciato sia valido per tutte le stringhe x e variabili B tali che l'appartenenza di x al linguaggio di B sia deducibile in n o meno passi di inferenza. Consideriamo l'ultimo passo dell'inferenza di w. Questo passo usa una produzione, diciamo A -> X1X2..Xn, dove ogni Xi è una variabile o un terminale. Possiamo scomporre w=w1w2..wn in modo che:

i) se Xi è terminale allora Xi=wi
ii) se Xi è variabile allora wi appartiene al ling. di Xi in al più n passi

l’ipotesi induttiva applicata a wi e Xi implica che esiste un albero sintattico con radice Xi e prodotto wi.

=>

costruiamo un albero sintattico:



con radice A e prodotto w1w2...wn=w

<u>inferenza</u> <u>ricorsiva</u> di a*(a+b00)

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

etichetta	stringa	V	Prod	stringhe usate
(i)	a	I	5	_
(ii)	b	I	6	_
(iii)	b0	I	9	(ii)
(iv)	b00	I	9	(iii)
(v)	a	E	1	(i)
(vi)	b00	E	1	(iv)
(vii)	a + b00	E	2	(v), (vi)
(viii)	(a +b00)	E	4	(vii)
(ix)	a * (a+b00)	E	3	(v),(viii)

dagli alberi alle derivazioni

serve una osservazione:
nella grammatica delle espressioni,
$E \Rightarrow I \Rightarrow Ib \Rightarrow ab$

per ogni coppia di stringhe α e β vale
$\alpha\ E\ \beta \Rightarrow \alpha\ I\ \beta \Rightarrow \alpha\ Ib\ \beta \Rightarrow \alpha\ ab\ \beta$

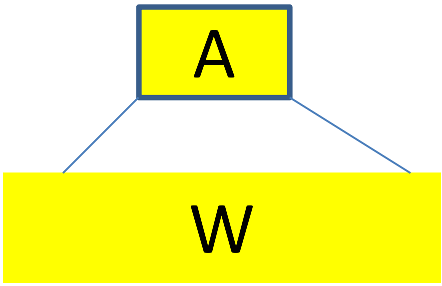
$E+(E) \Rightarrow E+(I) \Rightarrow E+(Ib) \Rightarrow E+(ab)$

insomma la derivazione è <u>libera dal contesto</u>

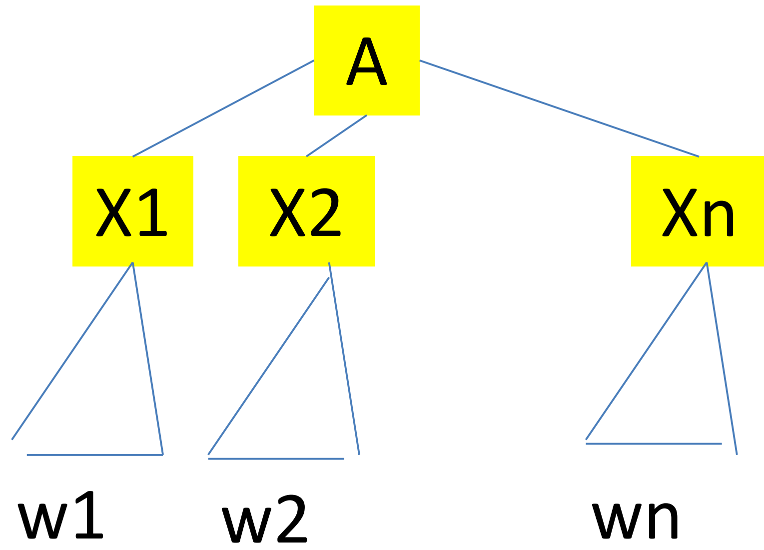
Teorema 5.14. Sia G=(V,T,R,S) una CFG e supponiamo che esista un albero sintattico con radice A e prodotto w in T*. Allora esiste una derivazione leftmost $A \Rightarrow^{*lm} w$

Dimostrazione: induzione sull'altezza dell'albero.

Base: altezza 1, consiste di 1 sola produzione di G, ovvio che $A \Rightarrow^{lm} w$



passo induttivo: un albero di altezza n con n>1, ha la forma, con w=w1w2..wn



dove la produzione A->X1X2..Xn è in R e,

--se Xi è terminale allora wi=Xi

--se Xi è variabile, allora è radice di un sottoalbero di altezza <n e prodotto wi,

per ipotesi induttiva $Xi =>^{*lm} wi$
e quindi per la libertà di contesto:

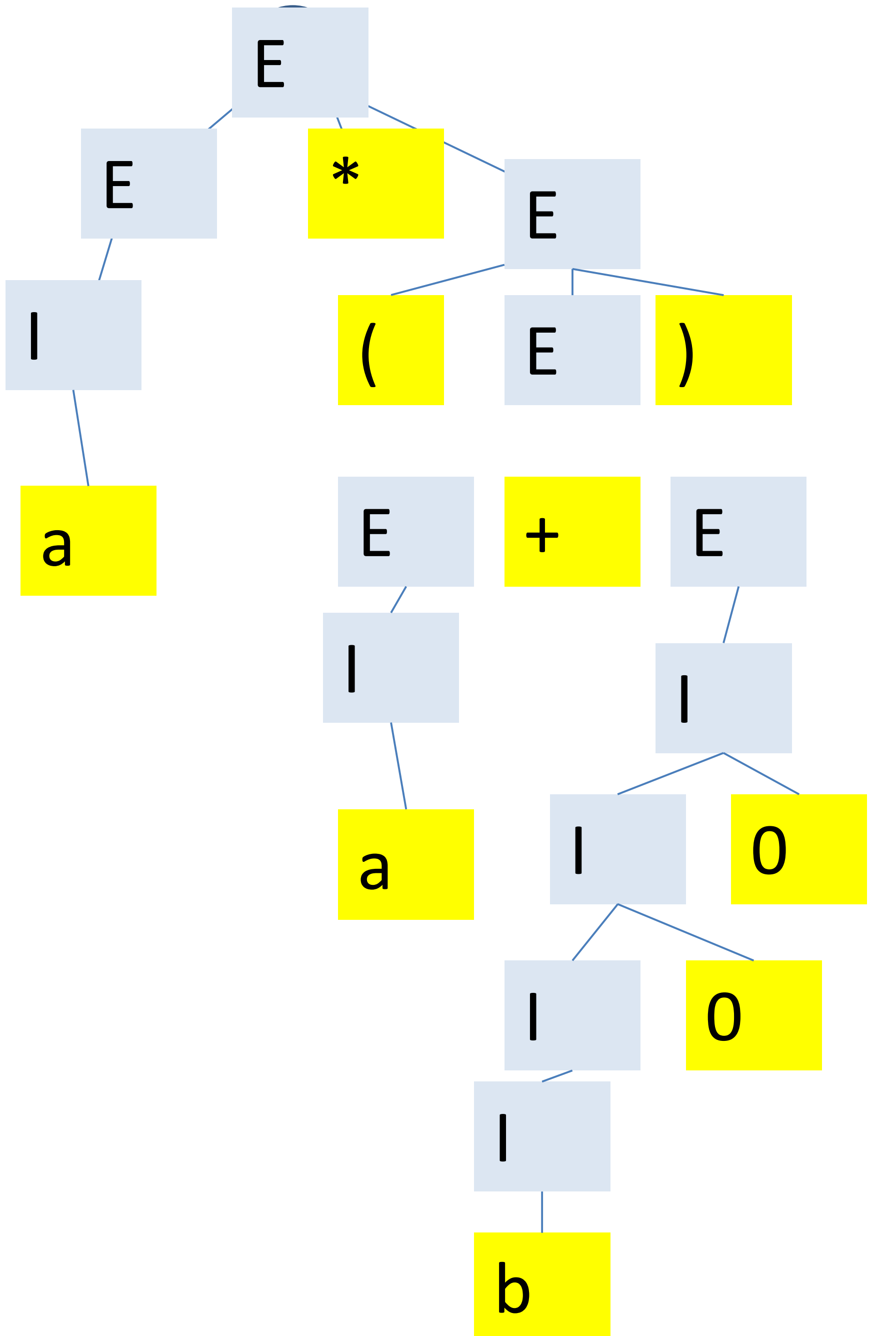
$A=>^{*lm} X1X2...Xn =>^{*lm} w1X2..Xn =>^{*lm}$
$w1w2...Xn$

rispettando l'ordine leftmost

formalmente serve un'induzione su i in [1..n] d1: $E \Rightarrow^{lm} I \Rightarrow^{lm} a$

d2: $E \Rightarrow^{lm} (E)$
$\Rightarrow^{lm} (E+E) \Rightarrow^{lm} (I+E) \Rightarrow^{lm}$
$(a+E) \Rightarrow^{lm} (a+I) \Rightarrow^{lm} (a+I0)$
$\Rightarrow^{lm} (a+I00)$
$\Rightarrow^{lm} (a+b00)$

da cui
$A \Rightarrow^{lm} E*E \Rightarrow^{lm} d1 * E \Rightarrow^{lm}$
$d1 * d2$



è ovvio che esiste un analogo teorema per le derivazioni rightmost

Manca un passo per chiudere il cerchio:
dalle derivazioni alle inferenze ricorsive

osserviamo che : $A \Rightarrow^{lm} X1X2..Xn \Rightarrow^{*lm}$ w1w2..wn=w
è facile estrarre derivazioni
$Xi \Rightarrow^{*lm} wi$

osserva che se wi=a allora la derivazione è $a \Rightarrow^{*lm} a$

Teorema 5.18
Sia G=(V,T,R,S) e supponiamo che esista una derivazione lm $A \Rightarrow^{*lm} w$ con w in T*,allora esiste una inferenza ricorsiva che determina che w è nel linguaggio di A

Dimostrazione: per induzione sulla lunghezza della derivazione $A \Rightarrow^{*lm} w$

Base: lunghezza 1, esiste A-> w in R con w in T*, allora w è nel linguaggio di A

passo induttivo: supponiamo che esista una derivazione lm di n+1 passi e che l'enunciato valga per ogni derivazione di n o meno passi.

La derivazione comincia con una produzione di R,

A = >X1X2...Xn =>*^{lm} w

allora possiamo scomporre w = w1w2..wn in modo tale che

--se Xi è terminale, allora Xi=>*^{lm} wi=Xi

--se Xi è variabile allora Xi =>*^{lm} wi ed avrà al più n passi

per ipotesi induttiva, esiste un'inferenza induttiva che prova che wi è nel linguaggio di Xi, quindi da

A -> X1X2...Xn si dimostra che w è nel linguaggio di A

esercizio 5.2.1

Applicazioni delle CFG

--parsing

--document type definition DTD che descrive i tag ammessi

parsing:
il problema del bilanciamento delle parentesi
(()), (()(())) sono ben bilanciate, (((o ()) non sono bilanciate

G_{bal}=({B},{(,)},R,B), con R uguale a :

B->BB | (B) | ε

è facile dimostrare che non è un linguaggio regolare

anche begin-end e anche altre parentesi

nei linguaggi ci sono anche costrutti che richiedono che ci possano essere più aperte (if) che chiuse (else)

Cond -> if (Exp) Cond |if (Exp) Cond else Cond

S -> iS | iSeS | ε

ei non va, anche iee non va, mentre ie e iiiie vanno

Vedi esercizio 5.4.2

Yacc è un parser generator: da una grammatica CF genera automaticamente un parser per essa, cioè un programma che data una stringa cerca di costruire un albero sintattico della grammatica che genera la stringa.

--se riesce allora stringa è ok

--se no stringa ha errori sintattici

```
Exp : Id
     | Exp ‘+’ Exp {azioni da fare quando la si usa}
     | Exp ‘*’ Exp {...}
     | ‘(‘ Exp ‘)’     {.....}
     ;
Id    : ‘a’   {.....}
```

vedrete qualcosa di più nel progettino di fine aprile col Prof. Bresolin

Linguaggi di Markup
HTML e XML

DTD = Document Type Definition

<!DOCTYPE nome-della-DTD[
 elenco di definizioni di elementi
]>

<!ELEMENT nome-elemento(descrizione dell’elemento)>
le descrizioni sono espressioni regolari

```
<!DOCTYPE PcSpecs [
	<!ELEMENT PCS (PC*)>
	<!ELEMENT PC (MODEL,PRICE,PROCESSOR,RAM,DISK+)>
	<!ELEMENT MODEL (#PCDATA)>
	<!ELEMENT PROCESSOR (MANF,MODEL,SPEED)>
	<!ELEMENT MANF (#PCDATA)>
	<!ELEMENT DISK (HARDDISK | CD | DVD)>
	<!ELEMENT HARDDISK(MANF, MODEL,SIZE)>
	<!ELEMENT CD (SPEED)>
.....
]>
```

documento conforme alla DTD precedente

```
<PCS>
    <PC>
        <MODEL>4560</MODEL>
        <PRICE>$2295</PRICE>
        <PROCESSOR>
            <MANF>Intel</MANF>
            <MODEL>Pentium</MODEL>
            <SPEED>800MHz</SPEED>
        </PROCESSOR>
        <RAM>256</RAM>
        <DISK><HARDDISK>
            <MANF>Maxtor</MANF>
            <MODEL>Diamond</MODEL>
            <SIZE>30.5Gb</SIZE>
        </HARDDISK></DISK>
        <DISK><CD>
            <SPEED>32x</SPEED>
        </CD></DISK>
    </PC>
    <PC>
        ...
    </PC>
</PCS>
```

il DTD è una CFG (o quasi)

<!ELEMENT PROCESSOR (MANF,MODEL,SPEED)>
Processor -> Manf Model Speed

<!ELEMENT DISK (HARDDISK | CD | DVD)>
Disk -> Harddisk | Cd | Dvd

espressioni regolari

<!ELEMENT PC (MODEL,PRICE,PROCESSOR,RAM,DISK+)>

Pc -> Model Price Processor Ram Disks
Disks -> Disk | Disk Disks

Per trasformare espressioni regolari in CFG esercizio 5.1.3