Ambiguità
delle grammatiche acontestuali

Il concetto di ambiguità è stato introdotto indipendentemente nel 1962, da David G. Cantor, e da Robert W. Floyd entrambi nel loro articolo mettono in rilievo un'ambiguità dell'ALGOL60.

L'esordio dell'articolo di D.G. Cantor recita:

"Backus [1] has developed an elegant method of defining well-formed formulas for computer languages such as ALGOL"

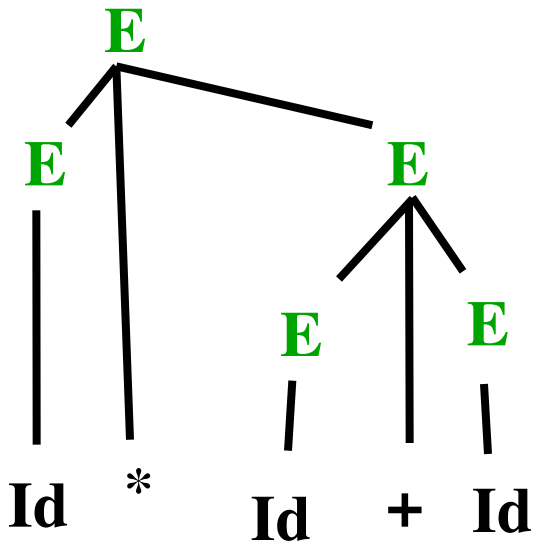
AMBIGUITÁ - Es. 1

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow Id$

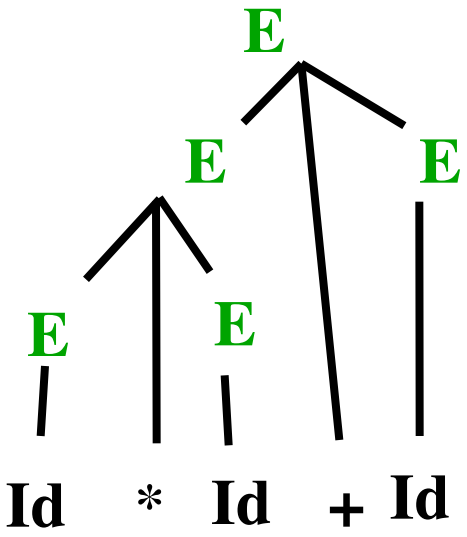


$E \Rightarrow E * E \Rightarrow Id * E$

$\Rightarrow Id * E+E \Rightarrow^* Id * Id + Id$

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

$\Rightarrow^* Id * Id + Id$



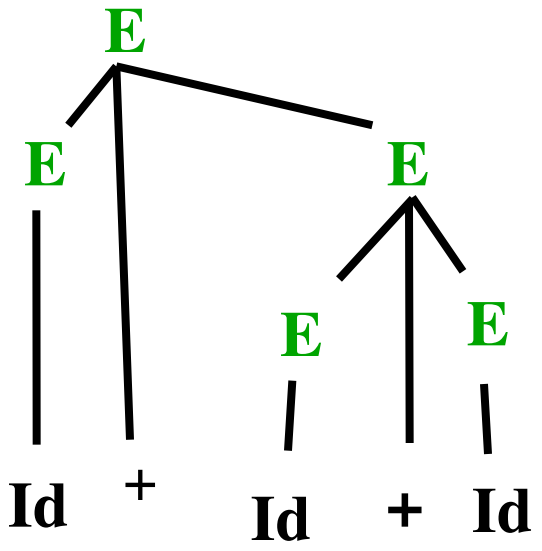
AMBIGUITÁ - Es. 2

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow Id$

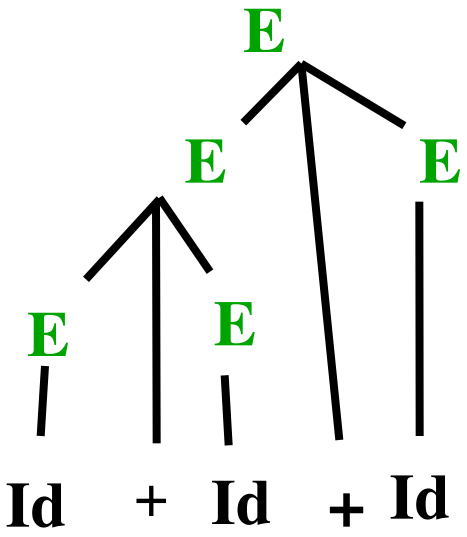


$E \Rightarrow E + E \Rightarrow Id + E$

$\Rightarrow Id + E + E \Rightarrow^* Id + Id + Id$

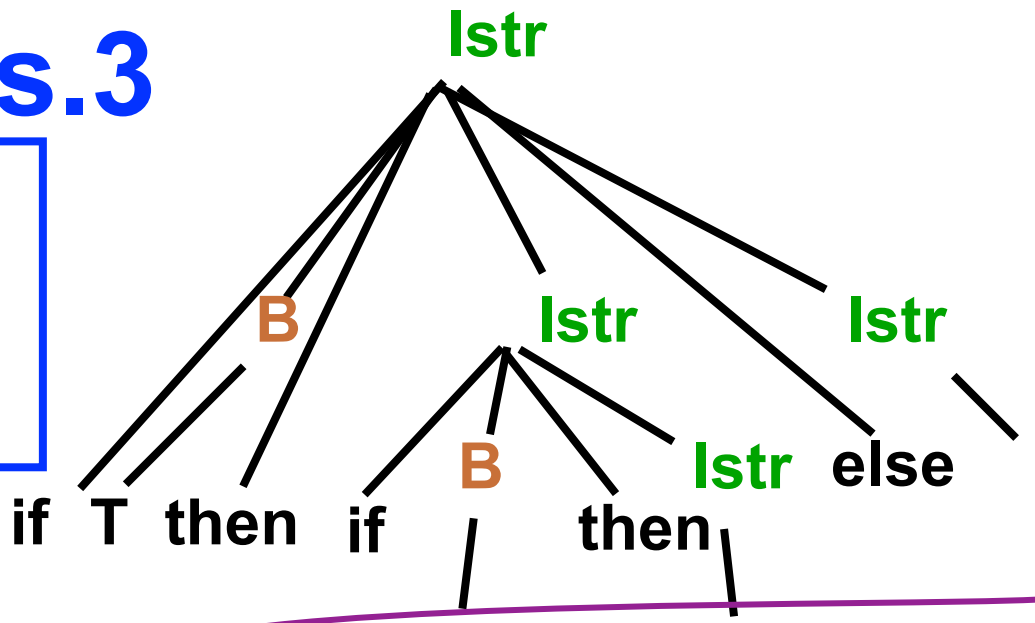
$E \Rightarrow E+E \Rightarrow E + E + E$

$\Rightarrow^* Id + Id + Id$

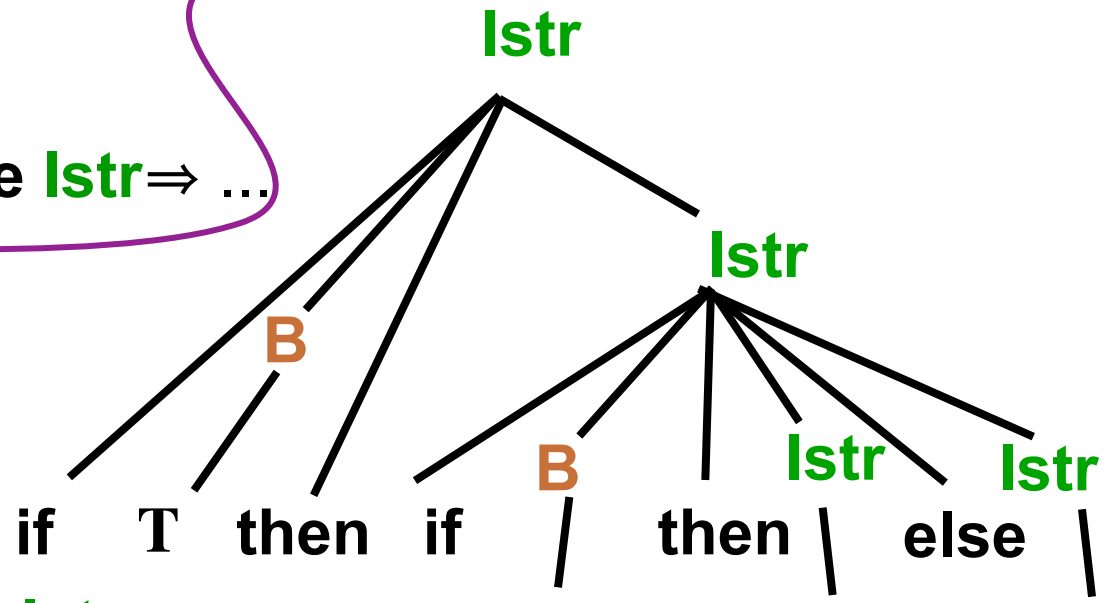


AMBIGUITÁ - Es.3

lstr→ if B then lstr
| if B then lstr else lstr
| Altra-lstr
B → T | F | ...



lstr⇒ if B then lstr else lstr ⇒
if T then lstr else lstr⇒
if T then if B then lstr else lstr⇒ ...



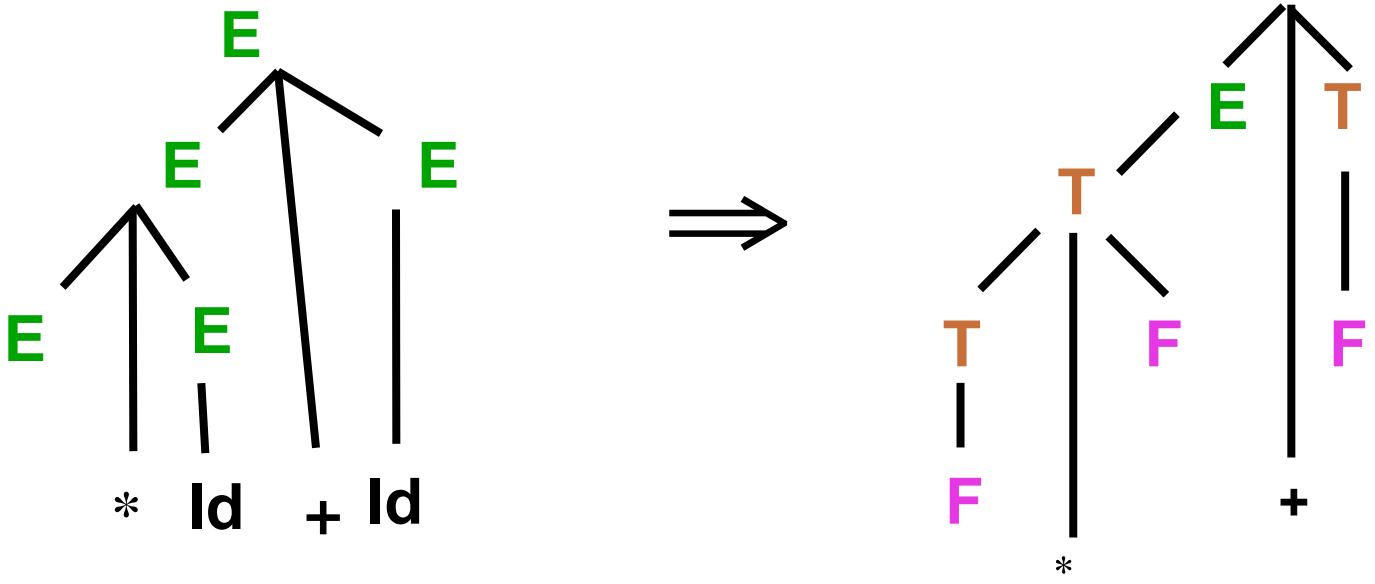
lstr⇒ if B then lstr ⇒
if T then lstr ⇒
if T then if B then lstr else lstr⇒ ...

AMBIGUITÁ: eliminazione ad hoc

$E \rightarrow E + E \mid E * E \mid E \rightarrow (E) \mid \mathrm{Id}$ $\Longrightarrow$

$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow (E) \mid \mathrm{Id}$

Modifichiamo le regole in modo da attribuire una precedenza più alta al prodotto rispetto alla somma.



AMBIGUITÁ: eliminazione ad hoc

```
Istr→ if B then Istr
      | if B then Istr else Istr
      | Altra-Istr
B → T | F | ...
```

⟹

```
Istr→  ITE  | IT | ...
IT →   if B then Istr
       | if B then ITE else IT
ITE → if B then ITE else ITE
       | Altra-Istr
B → T | F | ...
```

Modifichiamo le regole in modo da incorporare la regola dell'accoppiamento dell'else: l'*else* va accoppiato al più vicino *then* non ancora accoppiato.

AMBIGUITÁ - Es.3

$lstr \rightarrow I_{TE} \mid I_T \mid \ldots$

$I_T \rightarrow$ if B then lstr

$\mid$ if B then I_{TE} else I_T

$I_{TE} \rightarrow$ if B then I_{TE} else I_{TE}

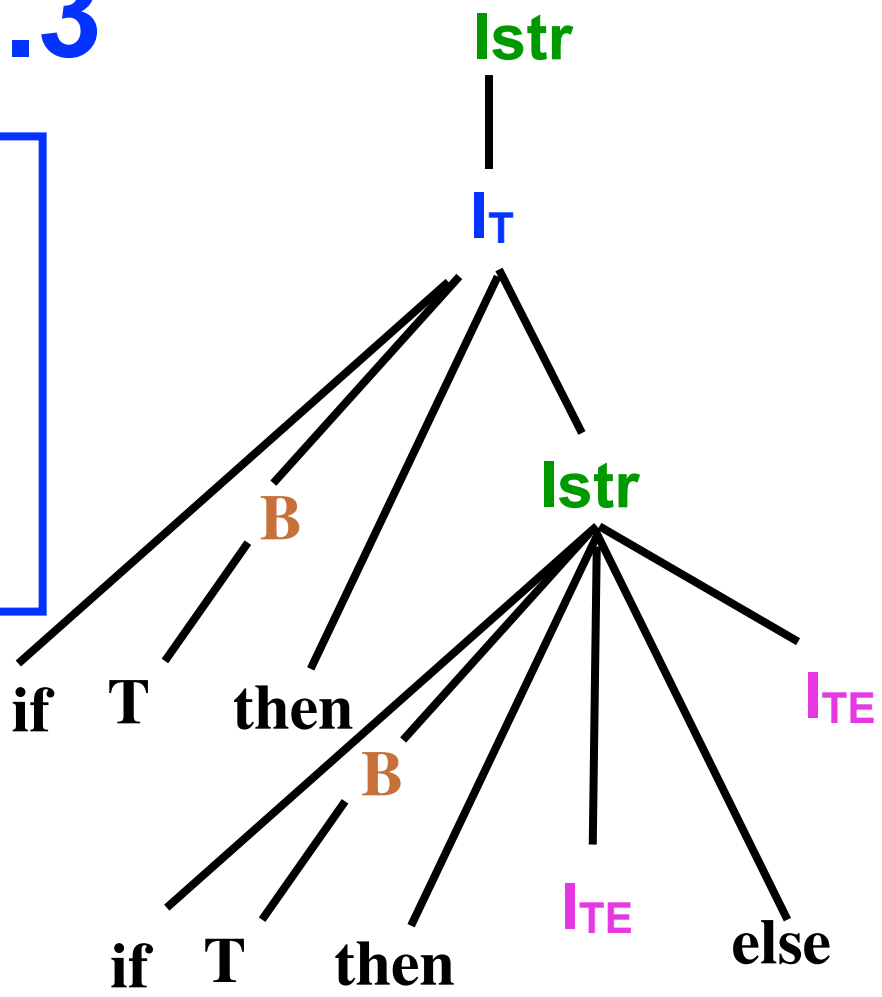
$\mid$ Altra-lstr

$B \rightarrow T \mid F \mid \ldots$

$lstr \Rightarrow I_T \Rightarrow$ if B then lstr $\Rightarrow$

if B then $I_{TE} \Rightarrow$

if B then I_{TE} else I_{TE}



AMBIGUITÁ: eliminazione ?

Disponiamo di un algoritmo che ricevendo in input una CFG, ne produce una equivalente non ambigua?

NO!

Anzi possiamo dimostrare che un tale algoritmo NON esiste.

Inoltre ci sono linguaggi che sono inerentemente ambigui, cioè tali da non ammettere una grammatica non ambigua. Per esempio:

$$L = \{a^i b^j c^k \mid i = j \text{ o } j = k \text{ e } i,j,k \geq 0\}$$

AMBIGUITÁ e DETERMINISMO

Se L è in L(DPDA) allora NON è inerentemente ambiguo.

La costruzione standard della grammatica da un PDA fornisce una grammatica non ambigua, se il PDA di partenza è deterministico.