

## **GRAMMARS II**

- Parse Trees
- Parsing
- Ambiguity
- Languages
- Exercises – 5.6

## Parse Trees

The derivation of a sentence can be visualized as a tree – the parse tree. The root of this tree is the start symbol, and the leaves of the tree are the terminals which form a sentence.

e.g.  $G = ( T, N, S, P )$

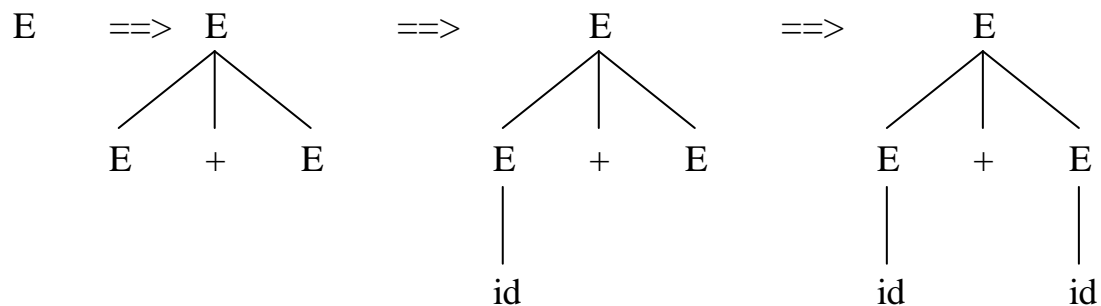
$T = \{ id, + \}$

$N = \{ E \}$

$S = E$

$P: E \implies E + E \mid id$

$E \implies E + E \implies id + E \implies id + id$



note: the final tree does not show the order of steps

- derivation is the process of rewriting the start symbol into the sentence
- parsing is the reverse...performed by building the parse tree
- how?

## Left-most and Right-most Derivations

e.g.  $G = (T, N, S, P)$

$T = \{ a, b, c, d, e \}$

$N = \{ S, A, B \}$

P:  $S \implies aABe$

$A \implies Abc \mid b$

$B \implies d$

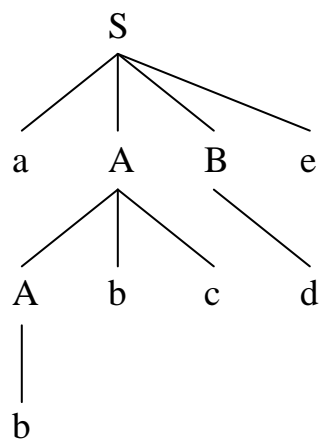
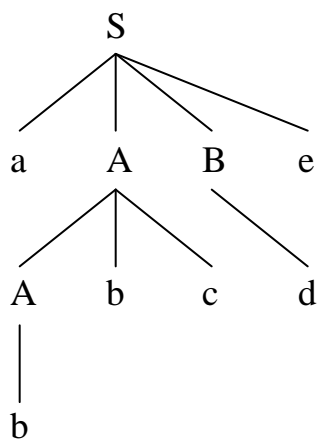
is abcde valid?

left-most derivation (always expand left-most non-terminal)

$S \implies aABe \implies aAbcBe \implies abbcBe \implies abcde$

right-most derivation (always expand right-most non-terminal)

$S \implies aABe \implies aAde \implies aAbcde \implies abcde$



- same parse tree for both derivations!

## Ambiguous Grammars

$$G = ( T, N, S, P )$$

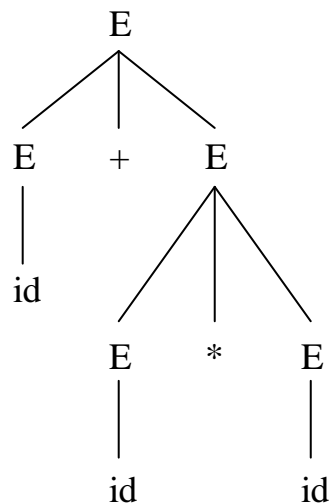
$$T = \{ \text{id}, +, * \} \quad N = \{ E \} \quad S = E$$

$$P: \quad E \implies E + E \mid E * E \mid \text{id}$$

id + id \* id

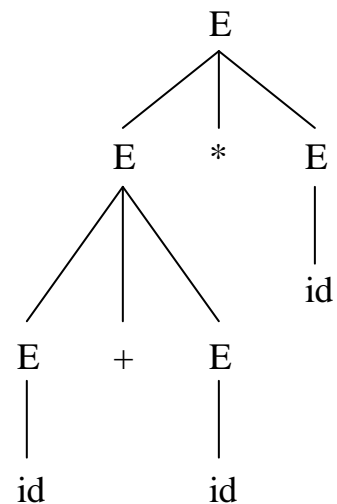
### left-most derivation

$$\begin{aligned} E &\implies E + E \\ &\implies \text{id} + E \\ &\implies \text{id} + E * E \\ &\implies \text{id} + \text{id} * E \\ &\implies \text{id} + \text{id} * \text{id} \end{aligned}$$



### right-most derivation

$$\begin{aligned} E &\implies E * E \\ &\implies E * \text{id} \\ &\implies E + E * \text{id} \\ &\implies E + \text{id} * \text{id} \\ &\implies \text{id} + \text{id} * \text{id} \end{aligned}$$



- more than one parse tree for a sentence
  - the grammar is ambiguous!
- fix: introduce more non-terminals

## Languages

- If we know the type of sentences that can appear in a language
  - can we develop a grammar for it?
- identify terminals
- specify a start symbol
- add non-terminals and production rules as necessary

## Example

- design a grammar for the language in which all sentences consist of lower-case letters separated by commas ( “,” )

$$T = \{ a, b, c, \dots z, \text{“,”} \}$$

$$N = \{ S, A, \text{ch} \}$$

$$P: \quad S \implies A \mid \epsilon$$

$$A \implies \text{ch} \mid \text{ch}, A$$

$$\text{ch} \implies a \mid b \mid c \mid \dots \mid z$$

$$x, y, z$$

$$\begin{aligned} S \implies A \implies \text{ch}, A \implies x, A \implies x, \text{ch}, A \implies x, y, A \implies x, y, \text{ch} \\ \implies x, y, z \end{aligned}$$

The above grammar appears to work (used example, not formal proof)



## Example

- Write a grammar for a valid if statement in JAVA

$G = ( T, N, S, P )$

$T = \{ \text{if, else, (, ), BOOLEXPR, STMT} \}$

$N = \{ \text{IF, ELSEIF, ELSE, BLOCK, MULTISTMT} \}$

$S = S$

P:  $S \Rightarrow \text{IF ELSEIF ELSE}$

$\text{IF} \Rightarrow \text{if ( BOOLEXPR ) BLOCK}$

$\text{ELSEIF} \Rightarrow \text{else if (BOOLEXPR ) BLOCK ELSEIF} \mid \epsilon$

$\text{ELSE} \Rightarrow \text{else BLOCK} \mid \epsilon$

$\text{BLOCK} \Rightarrow \text{STMT}$

$\text{BLOCK} \Rightarrow \{ \text{MULTISTMT} \}$

$\text{MULTISTMT} \Rightarrow \text{STMT MULTISTMT} \mid \epsilon$

- Does our grammar work?
- Does the JAVA grammar always work?