7 Context Free Grammars

Monday, September 18, 2023 11:04 AM

Why? Because programming Languages cannot be specified using reg-ex.

([] {¿?})

- GRAMMARS

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

N: set of non-terminal symbols. ↔ disjoint.
T: set of terminal symbols Σ
$S \in N$: the "start" non-terminal
P : a set of production rules.
rules of the form

$A \to \alpha$ where $A \in N$
$\alpha \in (N \cup T)^*$
↑ is a string made with terminals and non-terminals

Note α can be the empty string.

In a production rule $A \to \alpha$
A: head, α: body

shorthand

$A \to \alpha_1$
$A \to \alpha_2$
$A \to \alpha_3$
$\Rightarrow A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$

E.G.

$N = \{A, B, C\}$
$T = \{a, b\}$
$S = A$

$P =$
$A \to aBbA$
$A \to a$
$B \to bb \mid Cb$
$C \to ab \mid bAb$

Intuitively: a production rule is rule to rewrite a string.

e.g. $B \to bb$

$aAbBbC \Rightarrow aAbbbbC$

Definitions:

- Sentence :- a string of terminals and non-terminals in $(T \cup N)^*$
- Application of rule P :-
  Rule $A \to \alpha$, sentence s
  replacing an occurance of A in s with α
- Derivation
  A sequence of sentences each obtained from the previous one, by application of a rule.
- The language specified by a Grammar :-
  w (a string made of terminals)
  $w \in L$ iff there is a derivation from S to w

using the rules of the grammar.

E.G.

$N = \{A, B, C\}$
$T = \{a, b\}$
$S = A$

P =
1 $A \to aBbA$
2 $A \to a$
3 $B \to bb \mid Cb$
4 $C \to ab \mid bAb$

$A \xrightarrow{1} aBbA \xrightarrow{3} abbbA \xrightarrow{2} abbba$

abbba $\in L$!!!

EG #2.

$N = \{S, AT, BC\}$
$T = \{red, black, blue\}$
$S = S$

P
1 $S \to red \mid AT$
2 $AT \to S\ black\ BC$
3 $BC \to blue \mid blue\ AT$

$S \to AT \to S\ black\ BC \to S\ black\ blue \to red\ black\ blue$
$S \to red.$

Choices.

$L\{$ red, red black blue, ...

- MORE ON DERIVATIONS

DEF.- Leftmost Derivation
A derivation where the leftmost non-terminal is rewritten.

Rightmost Derivation
A derivation where the rightmost non-terminal is rewritten.

E.G.

$S \to SAS \mid x \mid y \mid z$
$A \to a \mid b$

Prove : $xaybz \in L$

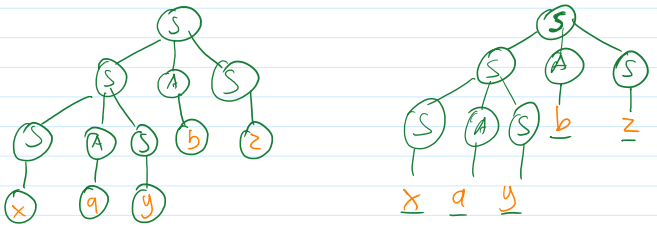
Leftmost:
S
SAS
SASAS
xASAS
xaSAS
xayAS
xaybS
xaybz

Rightmost:
S
SAS
SAz
Sbz
SASbz
SAybz
Saybz
xaybz.

S
SAS
SASAS

- Parse Tree:-
A tree representation of a derivation:
  - root: the start symbol
  - leafs: terminal symbols
  - intermediate nodes: non terminal symbols.

- leafs : terminal symbols
- intermediate nodes: non terminal symbols



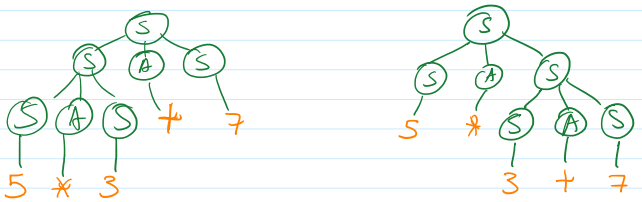
- AMBIGUITY IN GRAMMARS

A Grammar is called ambiguous when:
there exists a word w in the language that has
- more than one leftmost derivations
- more than one rightmost derivations
- more than one distinct parse trees.

E.G.

$S \to SAS \mid 5 \mid 3 \mid 7$
$A \to + \mid *$

w = 5 * 3 + 7

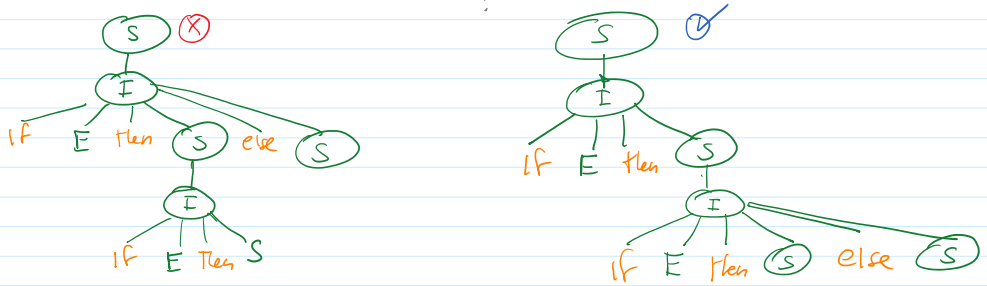


↑ preferable.

E.G. "the dangling else" – Algol-60

$S \to I \mid P$
$I \to$ if E then S | if E then S_1 else S_2

if E_1 then if E_2 then S_1 else S_2.



Note: an 'else' statement is paired with the closest "then"

-EOF-