7 Context Free Grammars

Wednesday, September 18, 2024 12:13 PM

Why? Because RegEx are not powerful enough.
for programming languages.
([ ] { }) ¿ ? BEGIN ... END.

Another approach for specification is needed.

- GRAMMARS:

DEF $G = \langle N, T, S, P \rangle$

N: Set of non-terminal symbols
T: Set of terminal symbols. Σ
$N \cap T = \emptyset$ N,T are disjoint
$S \in N$: The start symbol
P: a set of "production Rules"
rules of the form:
$A \to \alpha$ where $A \in N$
$\alpha \in (N \cup T)^*$

i.e. α is a string of terminals and non terminals.
Note α could be the empty string.

- In a production $A \to \alpha$ (A: head, α: body)
- a shorthand $A \to \alpha_1$, $A \to \alpha_2$, $A \to \alpha_3$ } $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$

E.G.

N: {S, H, B}
T: {x, q, m}
S: S

P:
$S \to xHqm$
$S \to HB$
$H \to qm$
$H \to Bx$
$B \to mm$

How does a language specifies a grammar?

Intuition: production rules are rules to rewrite strings.

e.g.

$xHqmBx \xrightarrow{H \to qm} xqmqmBx$

- DEFs
  - Sentence :- a string of terminals and non-terminals
  - Application of rule R into sentence β

    $R: A \to \alpha$

    replacing one occurance of A in β by α
  - Derivation :- a sequence of sentences each obtained from the previous one by the application of a rule.
  - The language specified by a grammar G :-

    $w \in L$ iff there exists a derivation from S to w, by application of rules in the grammar.

    $(w \in T^*)$

E.G.

$N: \{S, H, B\}$

$T: \{x, q, m\}$

$S: S$

$P:$
- $S \to xHqm$
- $S \to HB$
- $H \to qm$
- $H \to Bx$
- $B \to mm$

"mmxmm" $\in L$ ✓

$S \to HB \to Bx B \to mmxB \to mmxmm$ ✓

$S \to xHqm \to xqmqm$ ✓

$S \to HB \to qmB \to qmmm$ ✓

E.G

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

$T = \{ ( ) a b [ ] \}$

$S = S$

P
- $S \to AS \mid \lambda$
- $A \to BA \mid C \mid D \mid \lambda$
- $B \to a \mid b$

time a ([ )] ...

S = S

B → a | b
C → (S)
D → [S]

} every time a (is used a) is also introduced.

E.g:
a(b)a[(b)]
S → AS → BAS → a AS → aCS → a(S)S → a(AS)S → a(BAS)S →
a(bAS)S → a(b)S → a(b)AS → a(b)BAS → a(b)aAS →
a(b)aDS → a(b)a[S]S → a(b)a[AS]S → a(b)a[CS]S →
a(b)a[(S)S]S → .....

- MORE ON DERIVATIONS:

DEF: Leftmost derivation:- a derivation where the left-most non-terminal is the one re-written

Rightmost derivation:- a derivation where the rightmost non-terminal is the one re-written

E.g S → SAS | x | y | z
A → a | b

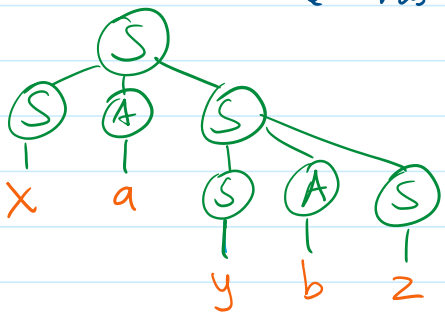
prove: xaybz ∈ L

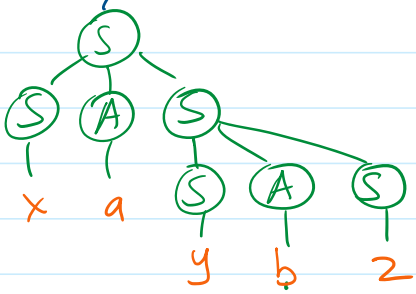
Leftmost:-
S
SAS
xAS
xaS
xaSAS
xayAS
xaybS
xaybz

Rightmost:-
S
SAS
SASAS
SASAz
SASbz
SAybz
Saybz
xaybz

DEF Parse tree:- a tree 🌳 representation of a derivation
- root:- the start symbol

Parse tree: - a tree representation of a derivation
- root - the start symbol
- leafs - terminal symbols
- intermediate nodes - non-terminals.
- A node has it's rewritten symbol as its children.

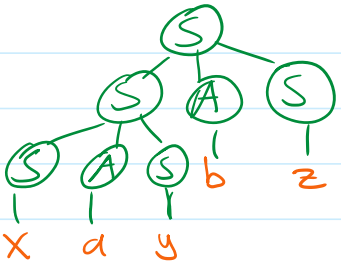




- AMBIGUITY:

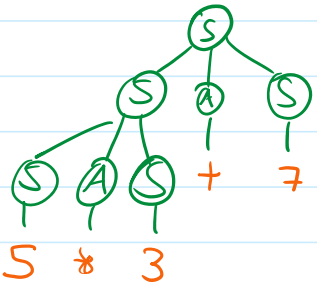
A gramma is called ambiguous when:
there exists a word w in the language that has
- more than one left-most derivations
or
- more than one right-most derivations
or
- more than one distinct parse-trees

E.G. A different parse tree:

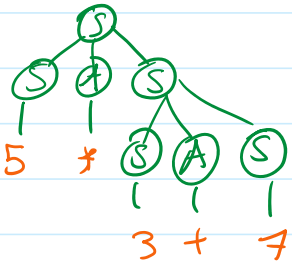


E.g $S \to SAS \mid 5 \mid 3 \mid 7$
$A \to + \mid *$

$w = 5 * 3 + 7$



This tree is preferable.



E.G. "The dangling else" – Algol '60

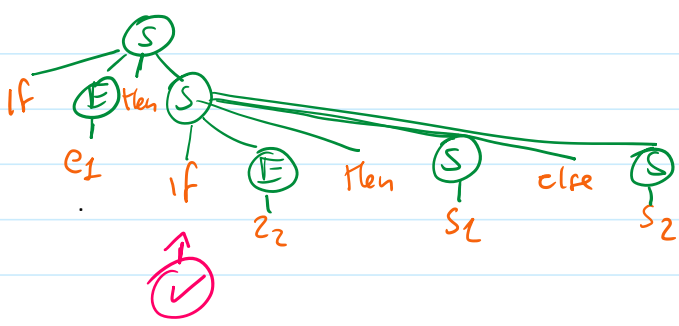
$S \to I \mid P \mid s$

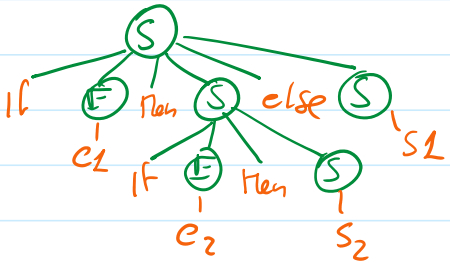
$I \to$ if E then S | if E then S else S

$E \to e$

w = if e_1 then if e_2 then s_1 else s_2

```
if e1 then                    if e1 then
  if e2 then   <-?->            |  if e2 then
  |  s1                       |     s1
  else                        else
     s2                          s2
```





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

EOF