22 February 2025

Parser:

- stream of tokens গুলো grammar rule follow করছে কিনা check করে, parse table and parse tree generate করে,

- যদি successfully parse tree বানাতে পারে then stream of tokens are correct.
- Tokens এর parsing hoy; code এর নয়,
- Parsing always left to right hoy.

Types of Parsers:

There are three types of parser,

- Universal Parser (Outdated)
- Top down parser also known as LL Parser
- Bottom up parser also known as LR parser

এর LL parser top down parser; এর LR parser bottom up parser.

$$\begin{tikzcd} L & L \arrow[r, "\text{Left to right parsing}"] & {} \\ & {} \arrow[r, "\text{left most derivative}"] & {} \end{tikzcd}$$

$$\begin{tikzcd} L & R \arrow[r, "\text{left to right parsing}"] & {} \\ & {} \arrow[r, "\text{right most derivative.}"] & {} \end{tikzcd}$$

Top down parser root node থেকে leaf node এ যায়;

Bottom up parser leaf node থেকে root node এ যায়.

• CFG stands for Context Free Grammar

8 production rules

✓start → factor
start → expression
✓expression → expression + term
✓term → term + factor
✓term → term * factor.
✓factor → id.
✓factor → num
✓term → factor

একটা CFG.

is factor reduced anywhere? if not, that means factor is the start state.

num is reduced in factor

একটা CFG এর 4 টা part থাকে,
(i) Terminals
(ii) Non Terminals
(iii) Start
(iv) Production Rules

এখানে,
Terminals: id, num, +, *
Non Terminals: expression, term, factor
Start:
Production rules: 8 production rule.

Production head $\boxed{\begin{array}{c}\rightarrow \\ : \\ =\end{array}}$ Production body

এই ৩ টার মধ্যে যেকোনো একটা থাকতে পারে

Production Rules

• একধরণের নিয়ম,
• Given CFG তে 6 টা production rule আছে।
• 2 টা portion থাকে
  ⇒ production head
  ⇒ Production body

→ বা : বা = এর বামপাশের portion হলো production head এবং ডানপাশের portion production body

$$\begin{tikzcd} \text{Production head} \arrow[r] \arrow[d] & \text{Production body} \arrow[d] \\ \text{non terminals} & \text{both terminals and non terminals} \end{tikzcd}$$

Terminals:

- Tokens
- Direct tokens যা আমরা lexer ও parser এ pars করছে।
- যারা ও terminal তারা কখনো production head এ থাকে না, তারা সবসময় production body তে থাকে।
- তাদের জন্য আলাদা কোনো rules define করা থাকে না।

Non terminals.

- Terminals বাদে বাকি সবই non terminals
- এদের জন্য আলাদা rules define করা থাকে।
- Non terminals production head এ থাকা জরুরী এবং
- Non terminals production body তে use হলে তা কোথাও না কোথাও production head এ আছে।

Start:

- Start is a special kind of non terminals
- It is basically your root node.
- এখান থেকে আর কোথাও যাওয়ার নাই,
- Start এর জন্য different multiple rules থাকতে পারে.
- Start যে সবসময় শুরু তে থাকবে not necessary মাঝখানেও থাকতে পারে.

exp → exp + term
exp → term
term → term * factor.

exp is the starting state here & exp can be the production body.

Start
|
Factor
|
num

letter দিয়ে express করতে পারি, token গুলো গুলো. letter দিয়ে express করতে পারবো না.

Start → factor
Start → expression
expression → expression + term
term → term + factor
term → factor.
term → term * factor.
factor → id
factor → number

S → F
S → E
E → E+T
T → T+F
T → F
T → T*F
F → id
F → number

S → F | E
E → E+T
T → T+F | F | T*F
F → id | number

optimized representation

more optimized representation

যদি আর production rule হতো... তাহলে or দিয়ে লিখব.

notational convention → go through book

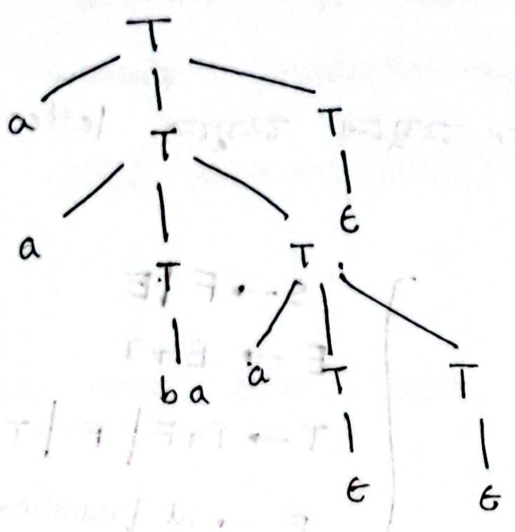
▨ $T \rightarrow aAT \mid aTT \mid \epsilon$

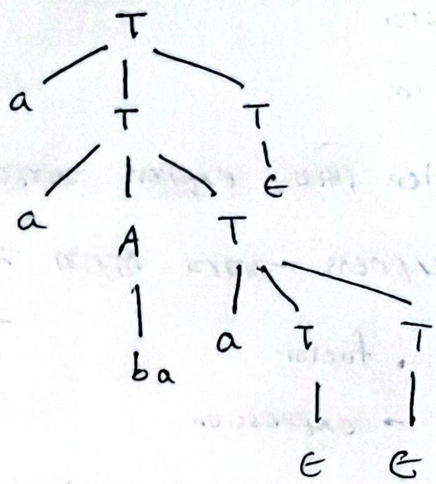
$A \rightarrow Tba \mid ba$

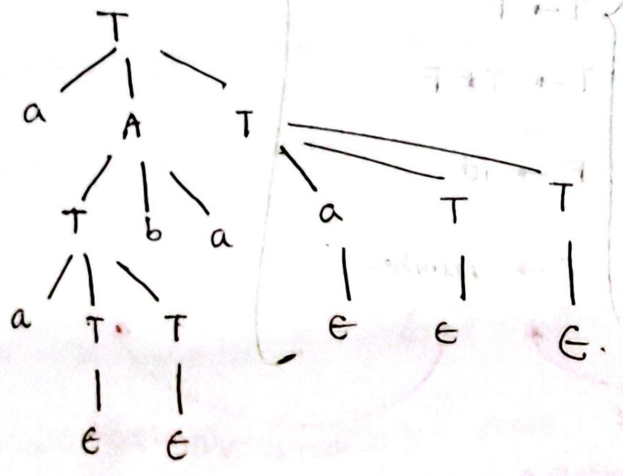
top down → rule থেকে start
bottom up → string থেকে start

check whether the word aabaa belong to the grammar or not

⇒







# 1টা input string বানানোর জন্য different parse tree থাকতে পারে,

# কোনোভাবেই যদি parse tree বানানো সম্ভব না হয়, তাহলে, given CFG দ্বারা ঐ input string build করা সম্ভব না বা ঐ input string given grammar follow করে না,

parsing এর জন্য buffer এবং stack use করা হয়.

Ambiguous:

একটি input string এর জন্য same grammar দিয়ে যদি different way তে parse tree generate করতে পারি; then that given grammar is ambiguous.

আর যদি শুধু একটাই tree হয় তাহলে non ambiguous.

Ambiguous Grammar:

For ambiguous grammar there exists more than one derivation for any word that belong to the grammar.

In case of non ambiguous grammar there exists only one derivation

Bottom up parsing:

Introduction to bottom up parsing

- A bottom up parser creates the parse tree of the given input starting from leave nodes toward the root node.
  - That means string theke start korbo instead of grammar.

▨

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid id$$

input string: id * id

Use bottom up parsing to generate the parse tree

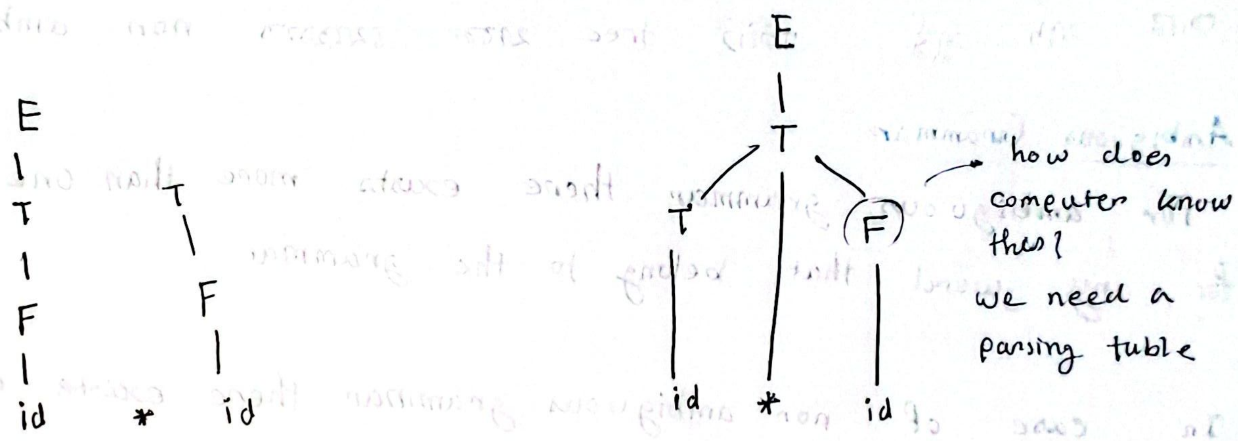
⇒

Terminals: id, +, *, (,)

Non Terminals: E, T, F

Start : E

Production Rules: 6টা



root node এ পৌঁছাইছি এবং মধ্যে আর কোনো input string বাকি নাই তখন বলে bottom up parsing done