TOP DOWN PARSERS

Parses an input string starting from start symbol and through a series of derivations reaches out to string (if it can be parsed)

Recursive-Descent Parsing - is a form of top-down parsing that may require backtracking

• Consists of a set of procedures, one for each nonterminal

PREDICTIVE PARSER - Special case of recursive-descent parsing that does not require backtracking
• Lookahead symbol unambiguously determines which production rule to use

• Advantage is that the algorithm is simple and the parser can be constructed by hand

Requirement of Grammar (Top Down Predictive Parser)

LL(1) Grammars - A grammars free from Left recursion and backtracking, Thus does not cause duplicate entries into parsing table

First L stands for left-to-right scan, second L stands for leftmost derivation

- There is one lookahead token
- In LL(k), k stands for k lookahead tokens
- Predictive parsers accept LL(k) grammars

Two major problems of top down parser

1. Backtracking
2. Left recursion

Any CFG grammar is not suitable to be used for Top-down parser. If problems as stated above persists in the CFG, It should be removed first. The resultant grammar is called LL(1).

Parsing Table Construction

It requires computations of FIRST AND FOLLOW SETS

FIRST Set

1. If X is a terminal, then FIRST $X = \{X\}$

2. If $X \rightarrow \epsilon$ is a production, then $\epsilon \in$ FIRST(X)

3. If X is a nonterminal and $X \rightarrow Y1Y2 \ldots Yk$ is a production then,

   I. Everything in FIRST($Y1$) is in FIRST X

   II. If for some i, $a \in$ FIRST(Yi) and $\forall 1 \leq j < i$, $\epsilon \in$ FIRST(Yj), then $a \in$ FIRST(X)

   III. If $\epsilon \in$ FIRST($Y1 \ldots Yk$), then $\epsilon \in$ FIRST(X)

FOLLOW Set

Defined as the set of terminals that can immediately follow X, that is, $t \in$ FOLLOW(X), if there is any derivation containing Xt

1. \$ is in FOLLOW(S) where S is the start symbol and \$ is the end marker

2. If there is a production $A \rightarrow \alpha B\beta$, then everything in FIRST(β) except ϵ is in FOLLOW(B)

3. If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B\beta$ where FIRST(β) contains ϵ, then everything in FOLLOW(A) is in FOLLOW(B)

Predictive Parsing Table Construction Algorithm

Input: Grammar G

Algorithm:

1. For each production $A \rightarrow \alpha$ in G,
2. For each terminal a in FIRST α , add $A \rightarrow \alpha$ to $M[A, a]$
3. If ϵ is in FIRST α , then for each terminal b in FOLLOW(A), add $A \rightarrow \alpha$ to M $[A, b]$
4. If ϵ is in FIRST α and \$ is in FOLLOW(A), add $A \rightarrow \alpha$ to $M[A, \$]$
5. No production in $M[A, a]$ indicates error

Predictive Parsing Algorithm

Input: String w and parsing table M for grammar G

Algorithm: Let a be the first symbol in w

Let X be the symbol at the top of the stack

while $X \neq \$$:

if $X == a$:

Pop the stack and advance the input

else if X is a Non terminal or $M[X, a]$ is an error entry:

error

else if M $[X, a] == X \rightarrow Y1Y2 \ldots Yk$:

output the production

pop the stack

push $YkYk-1 \ldots Y1$ onto the stack

$X \leftarrow$ top stack symbol

Example:

E -> E +T| T

T -> T*F | F

F - > (E) | id

Grammar has left recursion. So after modifications

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow * FT' \mid \epsilon$

$F \rightarrow ( ) \mid id$