COMPILER

SYNTAX ANALYSIS

BOTTOM UP PARSER

Top-down versus Bottom-up Parsing

- Top down:
  - Recursive descent parsing
  - LL(k) parsing
- Top to down and leftmost derivation
  - Expanding from starting symbol (top) to gradually derive the input string
- Can use a parsing table to decide which production to use next
- The power is limited
  - Many grammars are not LL(k)
  - Left recursion elimination and left factoring can help make many grammars LL(k), but after rewriting, the grammar can be very hard to comprehend
- Space efficient
- Easy to build the parse tree

Top-down versus Bottom-up Parsing

- Bottom up:
  - Also known as shift-reduce parsing
    - LR family
    - Precedence parsing
  - Shift: allow shifting input characters to the stack, waiting till a matching production can be determined
  - Reduce: once a matching production is determined, reduce
  - Follow the rightmost derivation, in a reversed way
    - Parse from bottom (the leaves of the parse tree) and work up to the starting symbol
  - Due to the added “shift”

    $\Rightarrow$ More powerful
    - Can handle left recursive grammars and grammars with left factors

    $\Rightarrow$ Less space efficient

Basic Idea

Construct Leaves first then move up to the root.

Example :

Let G be a grammar such that

E → E+T | T

T → T*F | F

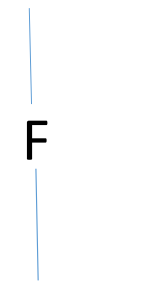
F → id

Let the input string be w = id * id

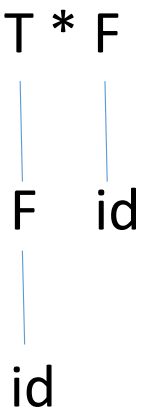
The steps to construct the tree will be as follows:

1st step:

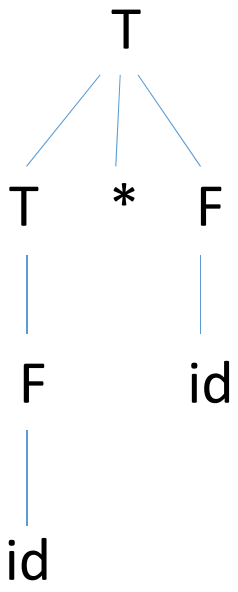
2nd step:



3rd step :



4th step :



Properties of Bottom Up Parsing

- A bottom-up parser begins with parse tree's leaves, and moves toward its root
- A bottom-up parser traces a rightmost derivation in reverse
- A bottom-up parser uses a grammar rule to replace the rule's RHS with its LHS
- (Fig. 4.5 & Fig. 4.6)

Reduction

Definition : Reverse process of derivation.

Ex. $\delta = \alpha\beta\omega$

and if $A \rightarrow \beta$

Then $\delta = \alpha A\omega$

Handle

Definition : A substring within a string which matches with the body of a production. It should comply with the rightmost derivation.

Ex. $S \rightarrow \delta$

$\delta \rightarrow \alpha\beta\omega$

$A \rightarrow \beta$

Then $\delta \rightarrow \alpha A\omega$

Thus β is a valid handle.

Handle Pruning

$\omega = \delta_0 \rightarrow \delta_1 \rightarrow \ldots\ldots \rightarrow S$

Challenges:

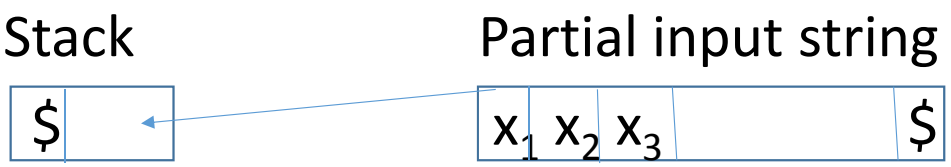
1) Which productions to use.
2) When to reduce.

Shift Reduce Parser

Input String	Handle	Production
id * id	id	F→id
F * id	F	T→F
T * id	T	F→T
T * F	F	F→id
T	T*F	T→T*F
E	T	E→T

Major Actions

1) Shift
2) Reduce



a) Find a handle from the string that u have in the stack
b) Reduce.