<u>LR PARSERS</u>: -

Construction of efficient Bottom-Up Parsers for a large class of Context-Free Grammars. Such Bottom Up Parsers are called LR Parsers.

LR parsers can be used to parse a large class of Context-Free Grammars. The technique is called LR(k) parsing.

L denotes that input sequence is processed from left to right

R denotes that the right most derivation is performed

K denotes that atmost K symbols of the sequence are used to make a decision.

<u>Features of LR Parsers</u>: -

* LR Parsers can be constructed to recognize virtually all programming constructs for which CFG can be written

* The LR Parsing method is move general than operator precedence or any of the other common shift reduce techniques.

* LR Parsers can detect syntactic errors as soon as it is possible to do so on a left to right scan of the input.

* LR Parsers can handle all languages recognizable by LL(1)

*LR Parsers can handle a large class of CF languages.

<u>Drawbacks of LR Parser</u>:-

Too much work has to be done to implement an LR Parser manually for a typical programming language grammar.

<u>LR Parser consists of two parts</u>.

(i) A driver routine

(ii) The parsing table changes from one parser to another

<u>LR Parsing Algorithm:</u>

LR Parsers consists of an input, an output, a stack, a driver program and a parsing table that has two functions.

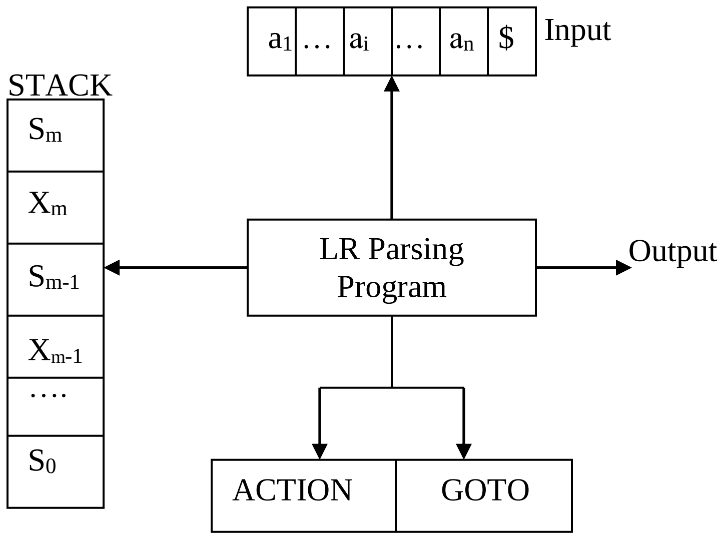
1. ACTION 2. GOTO

The driver program is same for all LR Parsers. Only the parsing table changes from one parser to another parser. The parsing program reads character from an input buffer one at a time. The program uses a STACK to store a string of the form $S_0X_1S_1X_2S_2……X_mS_m$, where Sm is on top. Each Si is a symbol called STATE and each Xi is a grammar symbol.

The function ACTION takes a state and input symbol as arguments and produces one of four values.

1. Shift S where S is a state
2. Reduce by a Grammar production
3. Accept and
4. Error

The function GOTO takes a state and Grammar symbol as arguments and produces as a state.



<u>Different LR Parsers Techniques</u>:-

There are three techniques for constructing an LR Parsing table for a Grammar.

1. Simple LR Parsing (SLR)
    * Easy to implement
    * Fails to produce a table for certain Grammars
2. Canonical LR parsing (CLR)
    * Most Powerful
    * Very Expensive to implement
3. Look Ahead LR Parsing (LALR Parsing)
    * It is intermediate in power between the SLR and the Canonical LR Methods.

References

1. J. Tremblay, P.G. Sorenson,"The Theory and Practice of Compiler Writing “, McGRAW-HILL,1985.
2. W.M. Waite, L.R. Carter,"An Introduction to Compiler Construction",Harper Collins,New york,1993
3. A.W. Appel,"Modern Compiler Implementation in, CambridgeUniversity Press,1998
4. Internet Papers

5. Aho, R. Sethi, J.D. Ullman," Compilers- Principles, Techniques and Tools"Addison-Weseley, 2007