2.4 LR PARSERS

An efficient bottom-up syntax analysis technique that can be used to parse a large class of CFG is called LR(k) parsing. The 'L' is for left-to-right scanning of the input, the 'R' for constructing a rightmost derivation in reverse, and the 'k' for the number of input symbols. When 'k' is omitted, it is assumed to be 1.

Advantages of LR parsing:

- It recognizes virtually all programming language constructs for which CFG can be written.
- It is an efficient non-backtracking shift-reduce parsing method.
- A grammar that can be parsed using LR method is a proper superset of a grammar that can be parsed with predictive parser.
- It detects a syntactic error as soon as possible.

Drawbacks of LR method:

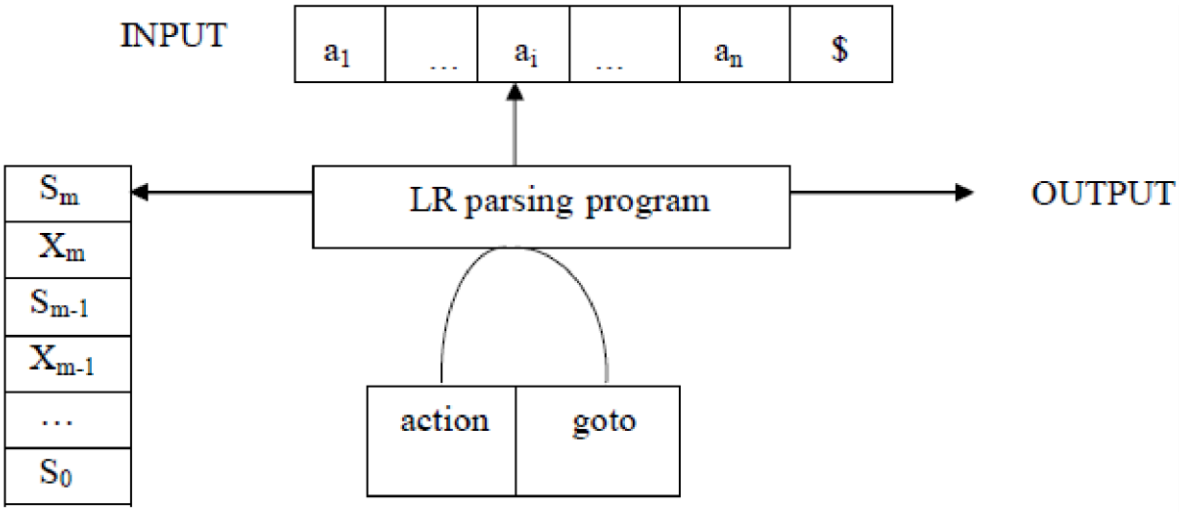
It is too much of work to construct a LR parser by hand for a programming language grammar. A specialized tool, called a LR parser generator, is needed. Example: YACC.

2.4.1 Types of LR parsing method:

1. SLR-Simple LR - Easiest to implement, least powerful.
2. CLR-Canonical LR - Most powerful, most expensive.
3. LALR-Look-Ahead LR - Intermediate in size and cost between the other two methods.

2.4.2 The LR parsing algorithm:

The schematic form of an LR parser is as follows:



It consists of : an input, an output, a stack, a driver program, and a parsing table that has two parts (action and goto).

- The driver program is the same for all LR parser.
- The parsing program reads characters from an input buffer one at a time.
- The program uses a stack to store a string of the form s0X1s1X2s2…Xmsm, where sm is on top. Each Xi is a grammar symbol and each si is a state.
- The parsing table consists of two parts : action and goto functions.

Action: The parsing program determines sm, the state currently on top of stack, and ai, the current input symbol. It then consults action[sm,ai] in the action table which can have one of four values :

1. shift s, where s is a state,
2. reduce by a grammar production A → β,
3. accept, and
4. error.

Goto: The function goto takes a state and grammar symbol as arguments and produces a state.

2.4.3 LR Parsing algorithm:

Input: An input string w and an LR parsing table with functions action and goto for grammar G.

Output: If w is in L(G), a bottom-up-parse for w; otherwise, an error indication.

Method: Initially, the parser has s0 on its stack, where s0 is the initial state, and w$ in the input buffer.

The parser then executes the following program:

```
set ip to point to the first input symbol of w$;
repeat forever begin
        let s be the state on top
                of the stack and
                a the symbol
                pointed to by ip;
        if action[s, a] = shift
           s' then begin push
           a then s' on top of
           the stack; advance
           ip to the next input
           symbol
        end
    else if action[s, a] = reduce
    A→ β then begin pop 2* | β |
    symbols off the stack; let s' be
    the state now on top of the
    stack; push A then goto[s', A]
    on top of the stack; output the
    production A→ β end
    else if action[s, a] = accept then
            return
    else error()
end
```