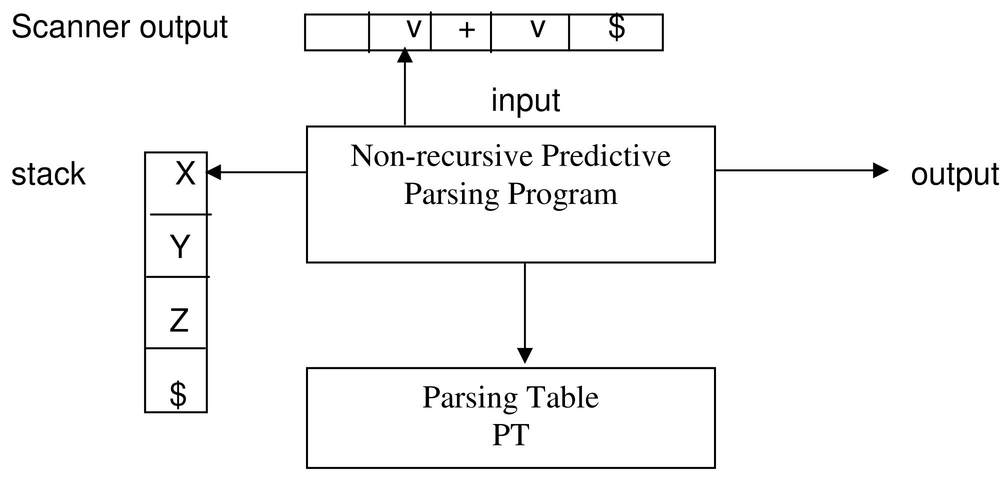
Non-Recursive Predictive Parsing
Article # 15

In recursive descent parsing, a nonterminal is implemented as a function call and the right-hand side of its production as the body of the function. Since the body may consist of other nonterminals, then it will call other functions. The system stack is used to pass parameters to and from the functions.

Instead of using the system stack, it is possible for the parser to maintain its own stack and integrate it more closely with the parsing process. In addition, since the main task of the function calls is to match the lookahead token, matching can now be done explicitly using grammar symbols (representing productions) on the stack. Instead of recursive function calls, a table is now used (analogous to the lexical transition table) to guide the parser to a matching token.

Structure diagram of Nonrecursive Predictive Parser



The components are:

- input token stream terminated with $
- stack of grammar tokens with the $ at the bottom
- parsing table (called frequently transition table) showing next production (c.f. next state)
- the parsing table is two-dimensional array PT[*A,a*], where *A* is a nonterminal and *a* is a terminal defined in the language grammar.

Initially the stack contains the start symbol of the grammar on top of $.

Operation

The program considers X, the symbol at the top of the stack, and a, the current input symbol. The subsequent action can be one of:

1. if X = a = $ the parsing is complete.
2. if X = a ≠ $, the parser pops X off the stack and advances the input pointer to the next input symbol.
3. if X is a nonterminal the program consults the table M[A,a] to lookup the production with which to replace X on the stack or else give an error. If X→UVW is the selected production, then X on the stack is replaced by UVW with U at the top of the stack.

Therefore we need a parsing table for the grammar (just as we needed a transition table for the regular expression). The table entries are determined by the FIRST and FOLLOW sets of the grammar. The FIRST set determines a production if the input token can be matched to the FIRST set. The FOLLOW set determines a production if the production derives ε and therefore will disappear but can match to a following production that contains the token.

Building Predictive Parsing Tables

The following algorithm can be used to construct a predictive parser table for a language **L(G)** defined by a grammar G.

BPPT1. For each production A -> α of the grammar G, do steps 2 and 3.
BPPT2. For each terminal *a* in the FIRST(α), add α to PT[A,a].
BPPT3. If ε is in FIRST(α), add ε to PT[A,b] for each terminal *b* in FOLLOW(A).