College Of Engineering Trivandrum

Compiler Design Lab

CS431

Submitted By:
Neethu S
S6 CSE Roll No:42

TVE18CS043

Department of Computer Science

December 3, 2021

Contents

Cycle 2 Experiment 5

1 Shift Reduce Parser

1.1 Aim

Construct a Shift Reduce parser for a given grammar. Read the formal definition of the context-free grammar describing the language in the form: (V, T, P, S) as input to your program. Find the equivalent Shift Reduce parser program for the same. Also write code for parsing any input string from the user, using the Shift Reduce parser program constructed.

1.2 Theory

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom-up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of the shift-reduce parser is the LR parser.

- At the shift action, the current symbol in the input string is pushed to a stack.
- At each reduction, the symbols will replaced by the non-terminals. The symbol is the right side of the production and non-terminal is the left side of the production.

1.3 Algorithm

```
1. Start
2. Get the input expression and store it in the input buffer.
3. Read the data from the input buffer one at the time
4. Using stack and push & pop operation shift and reduce symbols with respect
to production rules available.
5. Continue the process till symbol shift and production rule reduce reaches
the start symbol.
6. Display the Stack Implementation table with corresponding Stack actions
with input symbols.
7. Stop
```

1.4 Program & Output

```
// Shift Reduce Parser

#include <stdio.h>
#include <string.h>

int k = 0, z = 0, i = 0, j = 0, c = 0;
char a[16], ac[20], stk[15], act[10];

void check();
```

```
int main()
{
    printf("GRAMMAR is \n E->E+E \n E->E*E \n E->(E) \n E->id\n");

    printf("\nEnter input string: ");
    scanf("%s", a);

    c = strlen(a);
    strcpy(act, "SHIFT->");

    printf("\nStack \t  Input \t Action");

    for (k = 0, i = 0; j < c; k++, i++, j++)
    {
        if (a[j] == 'i' && a[j + 1] == 'd')
        {
            stk[i] = a[j];
            stk[i + 1] = a[j + 1];
            stk[i + 2] = '\0';
            a[j] = ' ';
            a[j + 1] = ' ';
            printf("\n$%s\t%s$\t%sid", stk, a, act);
            check();
        }

        else
        {
            stk[i] = a[j];
            stk[i + 1] = '\0';
            a[j] = ' ';
            printf("\n$%s\t%s$\t%ssymbols", stk, a, act);
            check();
        }
    }

    if (stk[0] == 'E' && stk[1] == '\0')
        printf("Accept\n");
    else
        printf("Reject\n");
}

void check()
{
    strcpy(ac, "REDUCE TO E");

    for (z = 0; z < c; z++)
        if (stk[z] == 'i' && stk[z + 1] == 'd')
        {
            stk[z] = 'E';
            stk[z + 1] = '\0';
            printf("\n$%s\t%s$\t%s", stk, a, ac);
            j++;
```

```
        }

    for (z = 0; z < c; z++)
        if (stk[z] == 'E' && stk[z + 1] == '+' && stk[z + 2] == 'E')
        {
            stk[z] = 'E';
            stk[z + 1] = '\0';
            stk[z + 2] = '\0';
            printf("\n$%s\t%s$\t%s", stk, a, ac);
            i = i - 2;
        }

    for (z = 0; z < c; z++)
        if (stk[z] == 'E' && stk[z + 1] == '*' && stk[z + 2] == 'E')
        {
            stk[z] = 'E';
            stk[z + 1] = '\0';
            stk[z + 1] = '\0';
            printf("\n$%s\t%s$\t%s", stk, a, ac);
            i = i - 2;
        }

    for (z = 0; z < c; z++)
        if (stk[z] == '(' && stk[z + 1] == 'E' && stk[z + 2] == ')')
        {
            stk[z] = 'E';
            stk[z + 1] = '\0';
            stk[z + 1] = '\0';
            printf("\n$%s\t%s$\t%s", stk, a, ac);
            i = i - 2;
        }

    printf("\n");
}
```

1.5 Output

```
neethu@neethu-Inspiron-15-3567:~/CD-Lab$ cc exp10.c
neethu@neethu-Inspiron-15-3567:~/CD-Lab$ ./a.out
GRAMMAR is
 E->E+E
 E->E*E
 E->(E)
 E->id

Enter input string: id+id*id

Stack     Input          Action
$id       +id*id$       SHIFT->id
$E        +id*id$       REDUCE TO E

$E+        id*id$       SHIFT->symbols

$E+id        *id$       SHIFT->id
$E+E         *id$       REDUCE TO E
$E           *id$       REDUCE TO E

$E*           id$       SHIFT->symbols

$E*id           $       SHIFT->id
$E*E            $       REDUCE TO E
$E              $       REDUCE TO E
Accept
neethu@neethu-Inspiron-15-3567:~/CD-Lab$ ./a.out
GRAMMAR is
 E->E+E
 E->E*E
 E->(E)
 E->id

Enter input string: i*i

Stack     Input          Action
$i       *i$    SHIFT->symbols

$i*       i$    SHIFT->symbols

$i*i       $    SHIFT->symbols
Reject
neethu@neethu-Inspiron-15-3567:~/CD-Lab$
```

1.6 Result

Implemented the program for shift reducer parser for a given grammar using C language in Ubuntu 20.04 with kernel and the above outputs were obtained.