程序设计 2 语法分析程序的设计与实现

实验报告

班级：-------- 姓名：--------- 学号：--------

一、 实验题目与要求

题目：语法分析程序的设计与实现。

实验内容：编写语法分析程序，实现对算术表达式的语法分析。要求所分析算术表达式由如下的文法产生。

$$E \to E+T \mid E-T \mid T$$

$$T \to T*F \mid T/F \mid F$$

$$F \to id \mid (E) \mid num$$

实验要求：在对输入表达式进行分析的过程中，输出所采用的产生式。

方法 1：编写递归调用程序实现自顶向下的分析。

方法 2：编写 LL(1)语法分析程序，要求如下。

(1) 编程实现算法 4.2，为给定文法自动构造预测分析表。

(2) 编程实现算法 4.1，构造 LL(1)预测分析程序。

方法 3：编写语法分析程序实现自底向上的分析，要求如下。

(1) 构造识别所有活前缀的 DFA。

(2) 构造 LR 分析表。

(3) 编程实现算法 4.3，构造 LR 分析程序。

方法 4：利用 YACC 自动生成语法分析程序，调用 LEX 自动生成的词法分析程序。

二、 实验分析

本次实验使用了方法 4，即使用 YACC 语法分析程序生成器，其中词法分析调用了 LEX 词法分析程序。通过定义翻译规则，YACC 使用 LALR 文法进行规约。通过 LEX 将读取到的字符串分析为符号串，再对符号串按照定义的表达式进行规约，判断输入的字符串是否符合表达式定义。

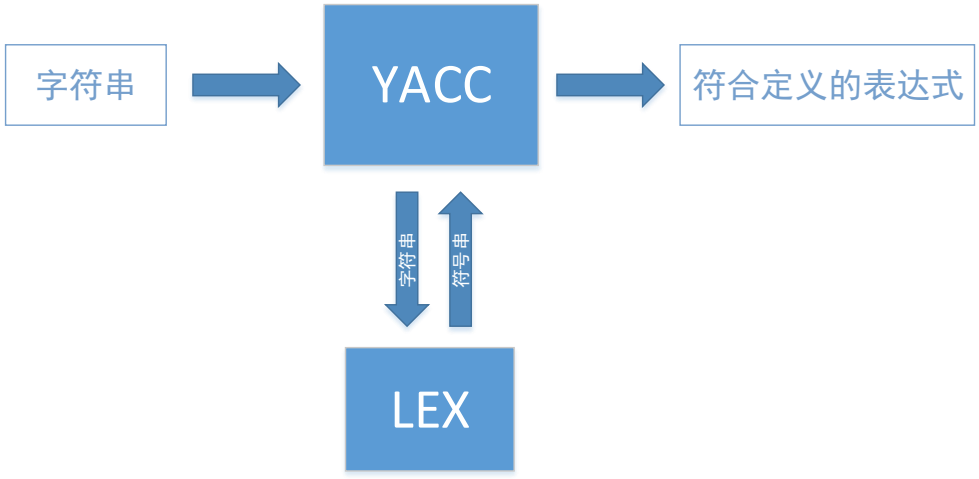

图 1 YACC 和 LEX 的关系

为了更好地理解这一文法，首先先分析 LR 文法并构造 LR 文法分析表，通过将 LR 文法中的相同状态合并即可得到 LALR 文法。

1. 构造扩展文法

$$E' \to E$$

$$E \to E+T \mid E-T \mid T$$

$$T \to T*F \mid T/F \mid F$$

$$F \to id \mid (E) \mid num$$

2. FIRST 和 FOLLOW 集

表 1 FIRST 和 FOLLOW 集

	E	T	F
FIRST	id, (, num	id, (, num	id, (, num
FOLLOW	$,), +, -	$,), +, -, *, /	$,), +, -, *, /

3. 构造识别所有活前缀的 DFA

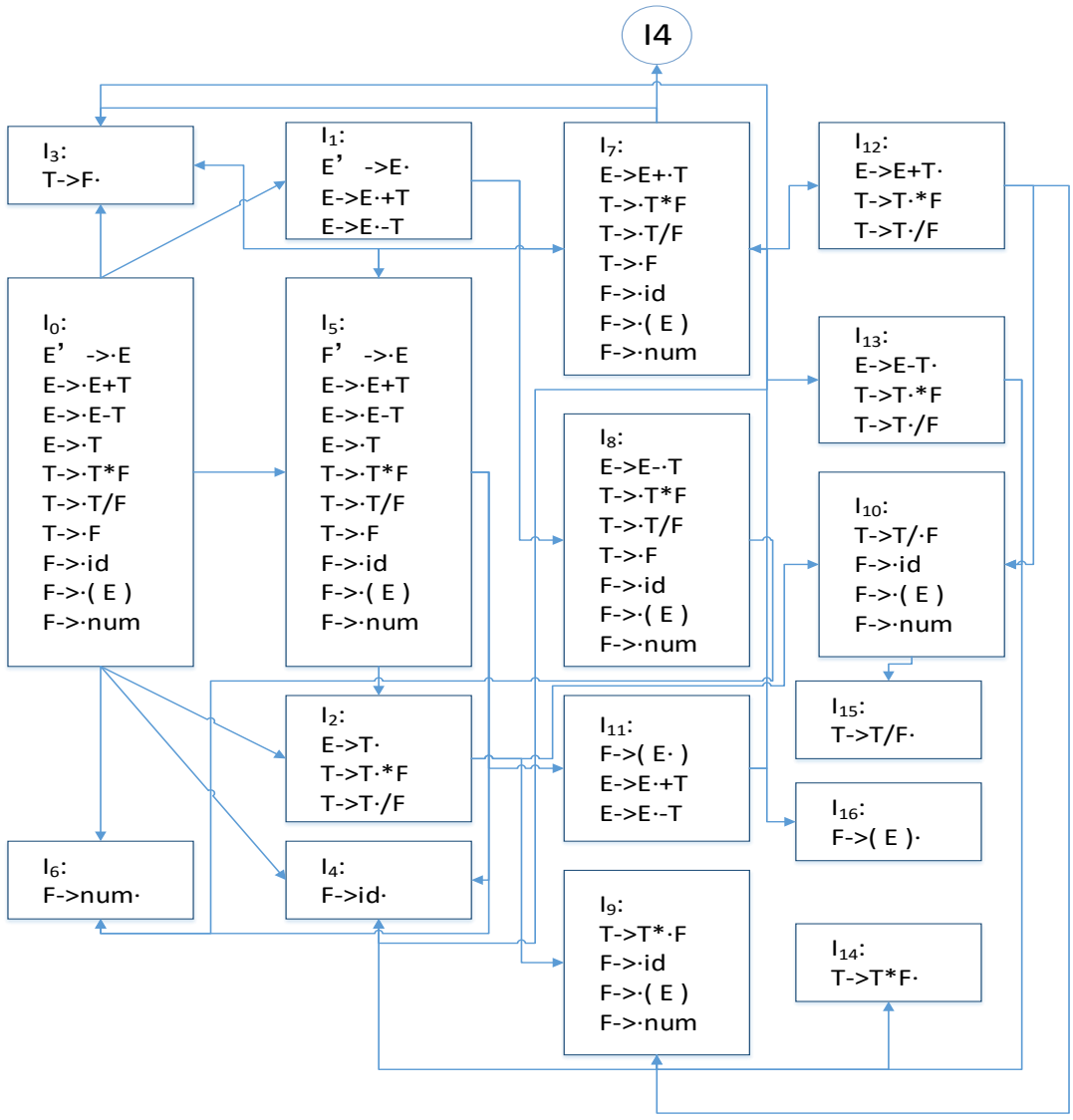

图 2 DFA 自动机

4. 构造 LR 分析表

表 2 LR 分析表

状态	action									goto		
	+	-	*	/	id	num	(	)	$	E	T	F
0					s4	s6	s5			1	2	3
1	s7	s8							acc			
2	r3	r3	s9	s10				r3	r3			
3	r6	r6	r6	r6				r6	r6			
4	r7	r7	r7	r7				r7	r7			
5					s4	s6	s5			11	2	3
6	r9	r9	r9	r9				r9	r9			
7					s4	s6	s5				12	3
8					s4	s6	s5				13	3
9					s4	s6	s5					14
10					s4	s6	s5					15
11	s7	s8						s16				16
12	r1	r1	s9	s10				r1	r1			
13	r2	r2	s9	s10				r2	r2			
14	r4	r4	r4	r4				r4	r4			
15	r5	r5	r5	r5				r5	r5			
16	r8	r8	r8	r8				r8	r8			

可以看出 YACC 的作用即在于帮助生成 LALR 的 DFA 自动机和分析表，按照分析表进行规约和移入，从而完成语法分析。

当存在二义性时，LALR 算法将会出现语法分析动作冲突。对于 YACC 来说，默认在解决规约/规约冲突时，选择规则中前面那个冲突进行分析；在解决移入/规约冲突时总是选择移入。而在这个文法中，+和-、*和/具有相同的优先级，因此使用%left ‘+’ ‘-’ 来定义其优先性和左结合性。

在文法中出现了 id，而 id 的值是未知的，分析 id 是没有问题的，然而这对表达式

求值造成了阻碍，因此在 lex 词法分析中建立了一个符号表，当遇到符号表里没有的符号时，提示用户输入符号的值并加入符号表中，在之后的分析中就会自动查找符号表并求值了。符号表采用数组形式，每一个 id 对应一个 num，相当于 Python 中的字典，只不过需要自己去写查找接口。

三、 源程序

YACC 程序：

```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define YYSTYPE double /*double type for YACC stack*/

int yylex(void);
void yyerror(char * s);
int yywrap(void);

%}

%token NUM
%token ID
%left '+' '-'
%left '*' '/'
%right UMINUS
%%


lines : lines expr '\n' {printf("Result: %f\n",$2);}
    |lines '\n'
    |      /* empty */
    |error '\n' {yyerror("Retry please.");
                yyerrok;}
    |'-' expr %prec UMINUS {$$ = -$2;printf("E->-E\n");}
    ;
expr : expr '+' term {$$ = $1 + $3;printf("E->E+T\n");}
    |expr '-' term {$$ = $1 - $3;printf("E->E-T\n");}
    |term {printf("E->T\n");}
    ;
term : term '*' form {$$ = $1 * $3;printf("T->T*F\n");}
    |term '/' form {$$ = $1 / $3;printf("T->T/F\n");}
    |form {printf("T->F\n");}
    ;
```

```
form : ID {printf("F->id\n");}
    |'(' expr ')' {$$ = $2;printf("F->(E)\n");}
    |NUM {printf("F->num\n");}
    ;

%%

#include "lex.yy.c"

void yyerror(char * s){
    printf("%s\n",s);
}

int main (){
    return yyparse();
}
```

LEX 程序：

```
%option noyywrap
%{
#include "first.tab.h"
char* id[100];
double num[100];
int count=0;
extern double yylval;

double install_id();
int yywarp (void);

%}
space       [' ']*
wrap        [\n]
letter      [A-Za-z]
digit       [0-9]
id          {letter}({letter}|{digit})*
num         {digit}+(\.{digit}+)?


%%
{id}          {yylval=install_id();return (ID);}
{num}         {yylval=atof(yytext);return (NUM);}
{space}       {printf(" ");}
{wrap}        {return '\n';}
```

```
.             {return yytext[0];}
%%

double install_id(){
    int i=0;
    for(i=0;i<count;i++){
        if(strcmp(yytext,id[i])==0){
            return num[i];
        }
    }
    printf("Found an unassigned id:\n%s = ",yytext);
    id[count]=(char*)malloc(sizeof(char)*yyleng);
    strcpy(id[count],yytext);
    scanf("%lf", &num[count]);
    count++;
    return num[count-1];
}

int yywarp (void){
    return 1;
}
```

四、 验证结果

输入：(5+6)/(9-2)
输出：F->num

```
T->F
E->T
F->num
T->F
E->E+T
F->(E)
T->F
F->num
T->F
E->T
F->num
T->F
E->E-T
F->(E)
T->T/F
E->T
Result: 1.571429
```

输入：5+8/6*(7+4)
输出：F->num
T->F
E->T
F->num
T->F
F->num
T->T/F
F->num
T->F
E->T
F->num
T->F
E->E+T
F->(E)
T->T*F
E->E+T
Result: 19.666667

输入错误表达式时：
输入：(5+7)/5+
输出：F->num
T->F
E->T
F->num
T->F
E->E+T
F->(E)
T->F
F->num
T->T/F
E->T
syntax error

输入：5+8/(5+46/5
输出：F->num
T->F
E->T
F->num
T->F
F->num
T->F
E->T
F->num

```
T->F
F->num
T->T/F
E->E+T
syntax error
```

出现 id 符号时：

输入：a+b/2

```
Found an unassigned id:
a = 5
F->id
T->F
E->T
Found an unassigned id:
b = 4
F->id
T->F
F->num
T->T/F
E->E+T
Result: 7.000000
```

输入：a+(6/b+3)

```
F->id
T->F
E->T
F->num
T->F
F->id
T->T/F
E->T
F->num
T->F
E->E+T
F->(E)
T->F
E->E+T
Result: 9.500000
```

（a 和 b 已经记录在符号表中，因此直接求值）

五、 实验总结

这次实验通过编写语法分析程序以完成对算术表达式的语法分析。经测试，设计出的程序能够准确识别表达式设定的语法，并对错误表达式进行反馈。这次实验加强了我

们对于 LALR 文法的理解，同时由于 YACC 中输出语法实际上是通过在翻译方案中设计输出语句进行输出，因此这次实验也涉及到了语法制导翻译的部分知识。设计出的程序还有值得改进的地方，例如在遇到错误表达式时进行一定的错误恢复并提示错误类型、扩充更多表达式等等。