中国矿业大学计算机学院

实验一：词法分析

课程名称 编译原理及实现

报告时间 2019.10.23

学生姓名 李治远

学　　号 07172757

专　　业 计算机科学与技术

任课教师 张辰

1、实验目的

1） 学会针对 DFA 转换图实现相应的高级语言源程序。

2） 深刻领会状态转换图的含义，逐步理解有限自动机。

3） 掌握手工生成词法分析器的方法，了解词法分析器的内部工作原理。

2、实验内容

C 语言的编译程序的词法分析部分实现。

从左到右扫描每行该语言源程序的符号，拼成单词，换成统一的内部表示（token）送给语法分析程序。

为了简化程序的编写，有具体的要求如下：

1） 空白符仅仅是空格、回车符、制表符。

2） 代码是自由格式。

3） 注释应放在花括号之内，并且不允许嵌套

3、状态转换图

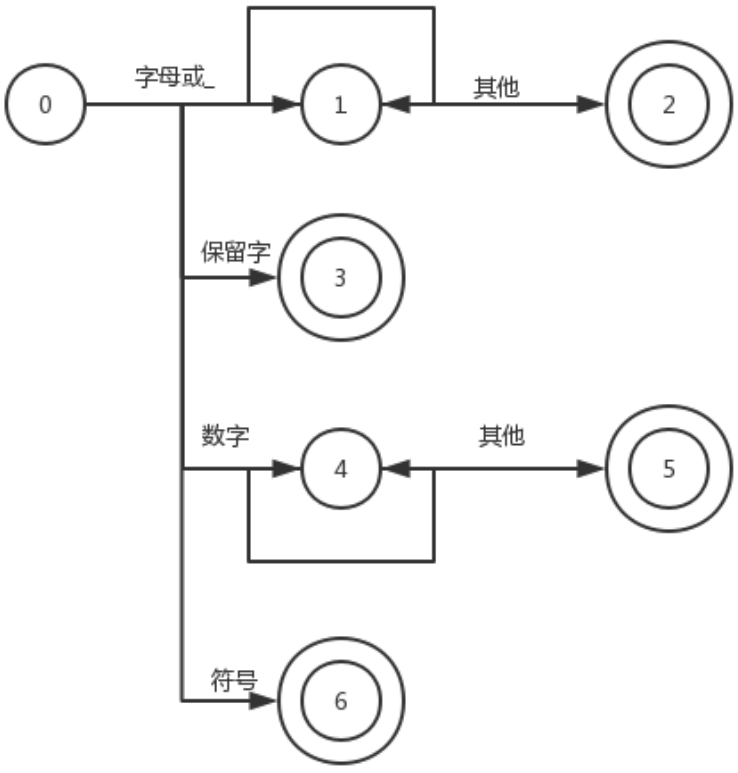

图 1：标识符、数字、字符等状态转换图

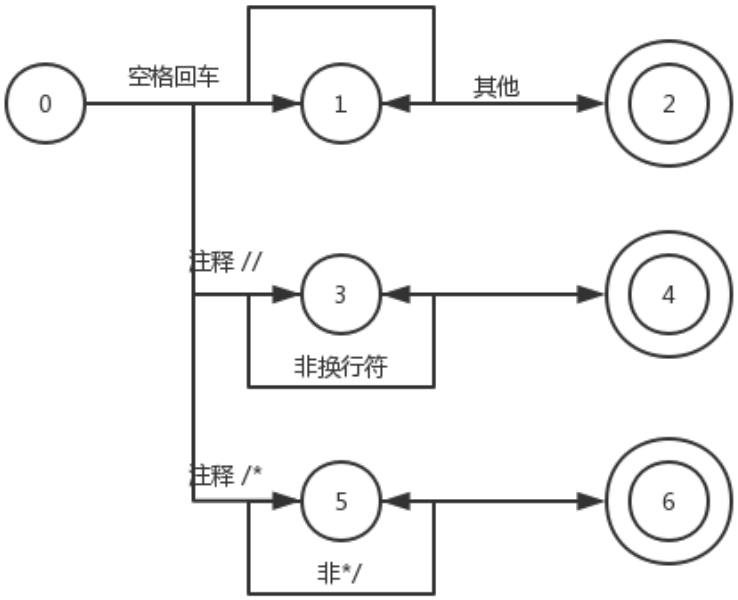

图 2：空白间隔符、换行符与注释（//、/* */）的状态转换图

4、输入输出测试

4.1 输入样例：

```
#include <studio.h>
int main(int argc, char const *argv[])
{
    char *str = "String123",c = 'a';
    /*
    printf("NULL\n");
    */
    int floatnum = 123.456;
    // 不做词法处理
    if(6.4ab <= 3.2E-1){
        int x = 0x454aEf;
        float y = 0347l;
        print("Yes ");}
    return 0;
}
```

4.2 输出结果：

表 1：输出结果

(7,#)	**(2,argv)**	**(2,floatnum)**	**(7,;)**
(3,include)	(7,[)	(7,=)	(3,float)
(7,<)	(7,])	-5,123.46	(2,y)
(2,studio)	(7,))	(7,;)	(7,=)
(7,,)	(7,{)	(3,if)	(5,0347l)
(2,h)	(3,char)	(7,()	(7,;)
(7,>)	(7,*)	(5,6.4)	(2,print)
(3,int)	(2,str)	(2,ab)	(7,()
(3,main)	(7,=)	(7,<=)	(9,Yes)
(7,()	(9,String123)	(5,3.2E-1)	(7,))
(3,int)	(7,,)	(7,))	(7,;)
(2,argc)	(2,c)	(7,{)	(7,})
(7,,)	(7,=)	(3,int)	(3,return)
(3,char)	(10,a)	(2,x)	(5,0)
(3,const)	(7,;)	(7,=)	(7,;)
(7,*)	(3,int)	(5,0x454aEf)	(7,})

注：2 为标识符、3 为保留字（关键字）、5 为数字（科学计数法、八进制、十进制与十六进制）7 为特殊符号，9 为字符串，10 为字符。

5、结语

词法分析程序设计步骤：1、确定词法分析器的接口；2、确定单词分类和单词结构；3、构造每一类单词的正规式以及对应的 NFA；对各类单词对应的 NFA 进行合并，构成一个能识别该语言所有单词的等价 DFA。

经过这次的编译原理实验，我更加理解了词法分析器是怎样的一个工作原理，根据编写的正规式和 DFA 状态转换图，通过不断读取输入的字符串，使用 GetToken()函数来获取不同的词。这使我对状态转换图的掌握大大加深，虽然说，有部分功能不是很完善，但是，基本功能都进行了实现，也锻炼了我的编程能力。

中国矿业大学计算机学院

实验二：递归下降语法分析器设计

课程名称 编译原理及实现

报告时间 2019.10.23

学生姓名 李治远

学　　号 07172757

专　　业 计算机科学与技术

任课教师 张辰

1、实验目的

1) 加深对递归下降分析法一种自顶向下的语法分析方法的理解。
2) 根据文法的产生式规则消除左递归，提取公共左因子构造出相应的递归下降分析器。

2、实验内容

根据课堂讲授的形式化算法，编制程序实现递归下降分析器，能对常见的语句进行分析。

3、消除左递归和左公共因子

左递归

将 A -> Aα | β 转换为

A -> β A'

A' -> α A'

左公共因子

将 S → aB1|aB2|aB3|aB4|...|aBn|y 转换为

S → aS'|y

S'→ B1|B2|B3|...|Bn

```
program -> block
block -> { stmts }
stmts -> stmt stmts
stmts -> null
stmt -> id = expr;
stmt -> if(bool) stmt else stmt
stmt -> if(bool) stmt
stmt -> while( bool ) stmt
stmt -> do stmt while(bool)
stmt -> break
stmt -> block
bool -> expr bool1
bool1 -> <= expr
bool1 -> < expr
bool1 -> >= expr
bool1 -> expr > expr
bool1 -> null
expr -> term expr1
expr1 -> + term expr1
expr1 -> - term expr1
expr1 -> null
term -> factor term1
term1 -> * factor term1
term1 -> / factor term1
term1 -> null
factor -> ( expr )
factor -> id
factor -> num
```

4、输入输出测试

4.1 输入样例：

```
{
    i = 2;
    while(i <= 100)
    {
        sum = sum + i;
        i = i + 2;
    }
}
```

4.2 输出结果：

```
program -> block
block -> { stmts }
stmts -> stmt stmts
stmt -> id = expr;
expr -> term expr1
term -> factor term1
factor -> num
term1 -> null
expr1 -> null
stmts -> stmt stmts
stmt -> while( bool ) stmt
bool -> expr bool1
expr -> term expr1
term -> factor term1
factor -> id
term1 -> null
expr1 -> null
bool1 -> <= expr
expr -> term expr1
term -> factor term1
factor -> num
term1 -> null
expr1 -> null
stmt -> block
block -> { stmts }
stmts -> stmt stmts
stmt -> id = expr;
expr -> term expr1
term -> factor term1
factor -> id
term1 -> null
expr1 -> + term expr1
term -> factor term1
factor -> id
term1 -> null
expr1 -> null
stmts -> stmt stmts
stmt -> id = expr;
expr -> term expr1
term -> factor term1
factor -> id
term1 -> null
expr1 -> + term expr1
term -> factor term1
factor -> num
term1 -> null
expr1 -> null
stmts -> null
```

```
stmts -> null
```

5、结语

递归下降分析器的基本思想是：文法的每个非终结符对应一个子程序（函数），在程序（函数）中实现对该终结符所在产生式的右部语法成分的识别，分析过程是按产生式规则自顶向下一层一层调用相关子程序（函数）来完成。

首先观察语法存在左递归和左公共因子，含有回溯，所以需要消除直接左递归和间接左递归。改造文法后才能自顶向下语法分析，然后根据输入串中的当前输入符号可以准确选择一个产生式进行匹配，因此对改写后的文法中的非终结符分别构造其对应的函数，实现对产生式右部匹配。

通过这个实验，我理解了“递归”和“下降”的含义，“递归”就比如是函数的递归调用，而“下降”的含义是，函数的调用顺序是按产生式规则自顶向下来完成的。如果文法含有左递归和左公因子，递归下降分析的时候就会产生回溯，将无法成功分析，所以一定需要先修改文法，才能使用递归下降分析。

中国矿业大学计算机学院

实验三：LR（k）分析器设计

课程名称 编译原理及实现

报告时间 2019.10.23

学生姓名 李治远

学　　号 07172757

专　　业 计算机科学与技术

任课教师 张辰

1、实验目的

1) 掌握有限自动机这一数学模型的结构和理论，并深刻理解下推自动机在 LR 分析法中的应用（即 LR 分析器）。

2) 掌握 LR 分析法的思想，学会特定分析表的构造方法，利用给出的分析表进行 LR 分析。

2、实验内容

根据课堂讲授的形式化算法，编制程序实现对以下语法进行自底向上语法分析的 LR 分析器，设计分析表，对给出的输入语句进行语法分析，判断是否符合相应的文法要求。

3、识别文法所有可归前缀 DFA

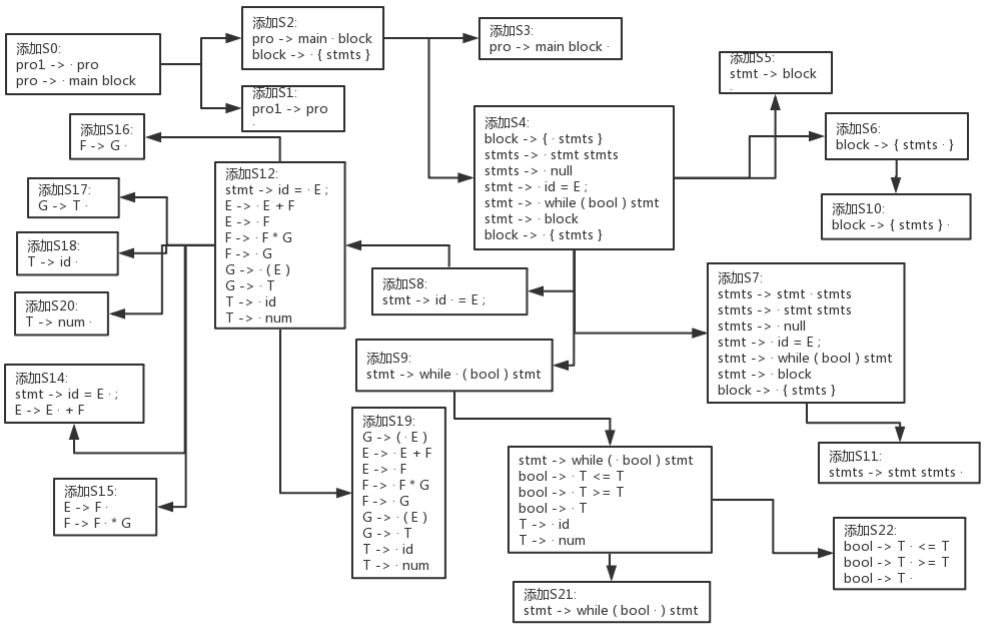

图 1：识别文法所有可归前缀 DFA

4、输入输出测试

4.1 输入样例：

```
{
    i = 2;
    while(i <= 100)
```

```
        {
            sum = sum + i;
            i = i + 2;
        }
    }
```

4.2 输出结果：

表 1：识别文法的 SLR（1）分析表

	main	{	}	id	=	(	)	+	*	>=	;	<=	num	while	#	pro1	pro	block	stmts	stmt	E	bool	F	G	T
0	s2																1								
1															acc										
2		s4																3							
3															r1										
4		s4	r4	s8										s9				5	6	7					
5		r7	r7	r7										r7											
6			s10																						
7		s4	r4	s8										s9				5	11	7					
8					s12																				
9						s13																			
10		r2	r2	r2										r2	r2										
11			r3																						
12				s18		s19							s20								14		15	16	17
13				s18									s20									21			22
14								s23			s24														
15							r9	r9	s25		r9														
16							r11	r11	r11		r11														
17							r13	r13	r13		r13														
18							r17	r17	r17	r17	r17	r17													
19				s18		s19							s20								26		15	16	17
20							r18	r18	r18	r18	r18	r18													
21							s27																		
22							r16			s28		s29													
23				s18		s19							s20										30	16	17
24		r5	r5	r5										r5											
25				s18		s19							s20											31	17
26							s32	s23																	
27		s4		s8										s9				5		33					
28				s18									s20												34
29				s18									s20												35
30							r8	r8	s25		r8														
31							r10	r10	r10		r10														
32							r12	r12	r12		r12														
33		r6	r6	r6										r6											
34							r15																		
35							r14																		

表 2：输入串分析过程

栈中状态	栈中符号	输入符号串	分析步骤
0	#	main { id = num ; while …= id + num ; } } #	s2 移进 main，状态转至 2
0 2	# main	{ id = num ; while (i…= id + num ; } } #	s4 移进{，状态转至 4
0 2 4	# main {	id = num ; while (i…id = id + num ; } } #	s8 移进 id，状态转至 8
...	...	…	…
0 2 4 6	# main { stmts	} #	s10 移进}，状态转至 10
0 2 4 6 10	# main { stmts }	#	r2 用第 2 产生式规约
0 2 3	# main block	#	r1 用第 1 产生式规约
0 1	# pro	#	acc success

5、结语

LR 分析法中，L 指自左向右扫描符号串，R 指分析过程中是最右推导的逆过程，是一种规范归约过程。算法：初始化，将初始状态 S0 压入状态栈，输入串左界符#压入文法符号栈；循环执行下面步骤，直至分析成功或者报错；根据

当前状态栈顶状态 Si 以及当前输入符号 ai 查询动作，进行移进、规约、成功或者报错。

经过 LR 分析实验，我对其分析过程的理解更加透彻，因为原文法产生的项目集规范族中存在“移进-规约”冲突所以构造 SLR(1)分析表，求规约项目的 FOLLOW 集合，和移进项目的集合，两个集合的交集为空，所以 SLR(1)分析表可以构造。若是出现“规约-规约”冲突，则分别求其 FOLLOW 集合，若交集为空，则构造 SLR(1)分析表，否则继续分析，是否能构造 LR(1)分析表。

中国矿业大学计算机学院

实验四：中间代码生成器设计

课程名称 编译原理及实现

报告时间 2019.10.23

学生姓名 李治远

学　　号 07172757

专　　业 计算机科学与技术

任课教师 张辰

本实验任务是在词法分析、语法分析和语义分析程序的基础上，将 C 子集源代码翻译为中间代码。理论上中间代码在编译器的内部表示可以选用树形结构（抽象语法树）或者线形结构（三地址代码）等形式，本实验要求输出四元式。

1、实验目的

(1) 熟悉各种中间代码表示的方式，比较它们之间的优缺点；
(2) 掌握语法树到中间代码的转换线性处理方法；
(3) 设计符合源语言和目标语言得到综合平衡的中间语言；
(4) 属性文法和语法制导翻译法进行语义翻译。

2、实验内容

根据课堂讲授的形式化，编制程序实现一个中间代码生成器，该程序能够使用前面的词法分析器和语法分析器，完成语法树到中间代码的转换。

3、翻译模式构造

3.1 赋值语句翻译

S->id=E	{p=lookup(id.name); if !p=null then emit(p=E.place) else error}
E->E1+E2	{E.place=newtemp; Emit(E.place=E1.place+E2.place)}
E->id	{p=lookup(id.name); if != null then E.place=p else error}

3.2 控制流语句翻译

S->while M1(E)M2 S1	{backpatch(S1.nextlist,M1.quad); Backpatch(E.truelist,M2.quad); S.nextlist=E.falselist; Emit(goto M1.quad)}
M->ε	{M.quad=nextquad}

4、输入输出测试

4.1 输入样例：

```
{
    i = 2;
```

```
        while(i <= 100)
        {
            sum = sum + i;
            i = i + 2;
        }
    }
```

4.2 输出结果：

语义分析结果(四元式):

```
(=,2,_,i)
(j<=,i,100,#)
(+,sum,i,T1)
(=,T1,_,sum)
(+,i,2,T2)
(=,T2,_,i)
```

```
100 i=2
101 if i<=100 goto 102
102 goto 107
103 T1+sum
104 sum=T1
105 T2+i
106 i=T2
107 other
```

开始生成中间代码

5、结语

这是编译实验的最后一个，因为时间和能力的原因，只完成了要求语法的中间代码代码生成，根据翻译模式进行语句分析和回填标号，总得来说，逻辑有一些模糊，但是还可以。

通过这四次的编译实验，先不说编译知识掌握了多少，编程能力是锻炼了很多，当然，对于编译实验的一些原理理解也更加深刻，现在理论课程也要结束了，但是编译的了解才刚刚入门，以后仍需要继续学习编译的理论和实验，真正掌握编译。