语法分析程序说明文档

141250149 吴秦月

目录

1. 实验目的

对输入语句进行语法分析，加深对语法分析原理的理解。

2. 内容描述

在上次实验写的词法分析程序的基础上，对其产生的 token 进行语法分析，使用的是 LL(1)方法，输出产生式序列

3. 思路方法

设计自定义文法->提取公共左因子，消除左递归，消除二义性->求 first，follow->构造预测分析表-->编写程序

4. 假设

1.假设只包含 id（变量），num（常数），＝，;,(,),{,},+,*,||,==,if,else,while,$
2.假设代码中 if 后都有 else。
3.假设只支持赋值语句，ifelse 条件语句，while 循环语句

5. 相关分析过程描述

1.自定义的文法
S->id=A|if(C){S}else{S}|while(C){S}
A->TB
B->+TB|ε
T->FG
G->*FG| ε
F->(A)|num|id
C->DE
E->||DE| ε

D->(C)|id==num

2.提取公共左因子，消除左递归，消除二义性后

S->id=A　S->if(C){S}else{S}　S-> while(C){S}　A->TB

B->+TB　B->ε　T->FG　G->*FG　G-> ε　F->(A)　F->num　F->id　C->DE

E->||DE　E-> ε　D->(C)　D->id==num

3.求 first，follow

state	first	follow
S	Id,if,while	},$
A	(,num,id	),$
B	+	),$
C	(,id	)
D	(,id	||
E	||	)
F	(,num,id	*
G	*	),$
T	(,num,id	),$

4.构造预测分析表

	num	+	id	*	if	||	else	==
S			S->id=A		S->if(C){S}else{S}			
A	A->TB		A->TB					
B		B->+TB						
T	T->FG		T->FG					
G		G-> ε		G->*FG				
F	F->num		F->id					
C			C->DE					
E								E->||DE
D			D->id==num					

	+	(	)	{	}	while	;	$
S						S-> while(C){S}		
A		A->TB						
B	B->+TB		B->ε				B->ε	B->ε
T		T->FG						
G	G-> ε						G-> ε	G-> ε
F		F->(A)	G-> ε					
C		C->DE						
E			E-> ε					E-> ε
D		D->(C)						

6. 重要数据结构

1.初始有$的状态栈

```
public class Stack {
    private ArrayList<Token> stack;

    public Stack(){
        stack = new ArrayList<Token>();
        stack.add(new Token(Token.END,"$"));
    }

    public void push(Token t){
        stack.add(t);
//      print();
    }

    public void pop() { stack.remove(stack.size() - 1); }

    public Token get() { return stack.get(stack.size() - 1); }

    private void print(){
        for(int i=stack.size()-1;i>=0;i--){
            System.out.println(stack.get(i));
        }
        System.out.println();
    }
}
```

2.存放词法分析完后的 token 队列，最后加上$

```
public class Queue {
    private ArrayList<Token> line;

    public Queue(ArrayList<Token> list){
        this.line = list;
        this.line.add(new Token(Token.END,"$"));
    }

    public Token get() { return line.get(0); }

    public void dequeue(){
//      Token ret = line.get(0);
        line.remove(0);
//      return ret;
    }

    public void enqueue(Token token) { this.line.add(token); }
}
```

3.存放产生式的数组

```
private static String[] generations = {
        "S->id=A;", "S->if(C){S}else{S}", "S->while(C){S}", "A->TB",
        "B->+TB", "B->ε", "T->FG", "G->*FG", "G->ε", "F->(A)", "F->num",
        "F->id", "C->DE", "E->||DE", "E->ε", "D->(C)", "D->id==num"};
```

4.预测分析表，－1 表示该处为空，数字代表产生式数组里的下标

```
private static int[][] table = {
//   id   =   ;   if  (   )   {   } else while +  * num   ||  ==  $
        {0, -1, -1, 1,  -1, -1, -1, -1, -1, 2,  -1, -1, -1, -1, -1, -1},//S
        {3, -1, -1, -1, 3,  -1, -1, -1, -1, -1, -1, -1, 3,  -1, -1, -1},//A
        {-1,-1, 5,  -1, -1, 5,  -1, -1, -1, -1, 4,  -1, -1, -1, -1, 5},//B
        {6, -1, -1, -1, 6,  -1, -1, -1, -1, -1, -1, -1, 6,  -1, -1, -1},//T
        {-1,-1, 8,  -1, -1, 8,  -1, -1, -1, -1, 8,  7,  -1, -1, -1, 8},//G
        {11,-1, -1, -1, 9,  -1, -1, -1, -1, -1, -1, -1, 10, -1, -1, -1},//F
        {12,-1, -1, -1, 12, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},//C
        {-1,-1, -1, -1, -1, 14, -1, -1, -1, -1, -1, -1, -1, 13, -1, 14},//E
        {16,-1, -1, -1, 15, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}//D
};
```

7. 核心算法

有一个显示维护的栈和存放词法分析完后的token队列。

每次读出栈的一个元素t1然后读出队列的一个元素t2，如果t1是非终结符，则结合t2,通过查预测分析表找到相应产生式，将产生式右部压入栈中。如果t1是终结符，若t1==t2，则匹配成功弹出。

循环上述过程，直到遇到$为止，过程中如果有查表失败或终结符不匹配则输出错误报告。

```
Token t1 = null;
Token t2 = null;

while(queue.get().getCode()!=Token.END){
    t1 = stack.get();
    t2 = queue.get();
    if(t1.getCode()>99){//非终结符
        if(!generate(t1, t2.getCode())){
            System.out.println("Error1!");
            return;
        }
    }
    else{//终结符
        if(t1.getCode()==t2.getCode()){//匹配成功
            stack.pop();
            queue.dequeue();
        }
        else{//否则报错
            System.out.println("Error2!");
            return;
        }
    }
}
```

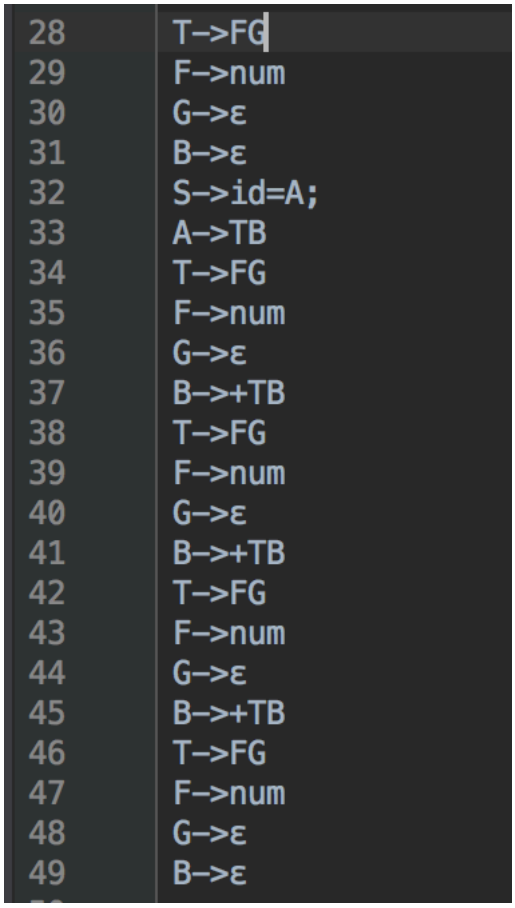
8. 运行截图

input:

```
while(a==11||b==11){
if(c==11){
a=2*(2+3)+1;
}
else{
b=1+2+3+5;
}
}
```

output:

```
1   S->while(C){S}
2   C->DE
3   D->id==num
4   E->||DE
5   D->id==num
6   E->ε
7   S->if(C){S}else{S}
8   C->DE
9   D->id==num
10  E->ε
11  S->id=A;
12  A->TB
13  T->FG
14  F->num
15  G->*FG
16  F->(A)
17  A->TB
18  T->FG
19  F->num
20  G->ε
21  B->+TB
22  T->FG
23  F->num
24  G->ε
25  B->ε
26  G->ε
27  B->+TB
```



9. 问题与解决

一开始直接在预测分析表里写产生式，但是有点多一直错而且看起来也不好看，后来用了表驱动之后就既看起来好看也不容易错

10. 感受与总结

经过自己动手查找资料、编写简单的语法分析程序，有助于对语法分析过程和方法有更深入的理解，受益匪浅！