《编译原理》课程实验报告二

——一个简易语法分析器的实现

殷迪 131250021

Motivation/Aim

本实验通过一个简易的语法分析器的实现，进一步加深对《编译原理》课程中语法分析理论的理解，增强动手程序设计能力。分析可以采用自上而下的 LL(1)分析法，自下而上的 SLR(1)和 LR(1)分析法。本次试验采用的是 LR(1)分析法，加深对它的理解。

Content description

利用 Java 语言实现一个 C 语言子集文法的语法分析器，输入为前一次实验对一段程序进行词法分析后获得的 token 序列，输出是自底向上语法分析中 LR（1）分析的移入规约序列，序列中也显示了栈的内容，并且包含了错误处理和提示。

Ideas/Methods

本程序可以通过输入文法，自动构建 LR(1)分析表进行分析。

程序设计的主要思路如下：

1. 根据输入的文法 G 初始化辅助的数据结构，包括终结符集，非终结符集，非终结符的 first 集（用于产生 LR(1)分析表中的预测符）等。
2. 根据输入的文法 G，构造出 LR(1)状态机（项目集）。
3. 根据构造出的状态机（项目集），生成分析表的 GOTO 和 ACTION 部分，构造出 LR(1)分析表。
4. 根据 LR(1)分析表，对输入的字符串进行分析
5. 得到并输出分析结果
6. 如果输入的文法有二义性规约冲突，或者输入的字符串不合法会有错误提示。

Assumptions

从词法分析器获取的 token 序列全部符合文法的要求，即全部为给定的文法以及文法符号。

Related FA descriptions

本程序所使用的自动机全部为程序从文本读取产生式后自动生成，并不需要给出 FA 的示意图，但是本报告后面会给出自动构建的各个状态。

Description of important Data Structures

产生式、终结符、非终结符集

```
public class CFG {

    public static String emp = "ε";

    public static String end = "$";

    public static TreeSet<String> keywords = new TreeSet<String>();//保留字集

    public static TreeSet<String> VN = new TreeSet<String>();//非终结符集
    public static TreeSet<String> VT = new TreeSet<String>();//终结符集
    public static ArrayList<Derivation> F = new ArrayList<Derivation>();//产生式集

    public static HashMap<String,TreeSet<String> > firstMap = new HashMap<String,TreeSet<String> >();//first
    public static HashMap<String,TreeSet<String> > followMap = new HashMap<String,TreeSet<String> >();//follow
```

全部为 static 变量，在程序编译链接时自动从文件中读取、解析并存储进对应的集合，在后续分析过程中随时可以进行调用和查询。

产生式

Left 为产生式左端的内容，ArrayList<String>中存储的是产生式右端的内容，按照产生式中的顺序进行排列。

```
public class Derivation {

    public String left;
    public ArrayList<String> list = new ArrayList<String>();
```

用于进行 LR（1）分析的产生式

其中 d 为一个产生式，lr 为 LR（1）分析式右端的那个终结符，index 指向点的位置。

```
public class LRDerivation implements java.lang.Cloneable{

    public Derivation d;
    public String lr;
    public int index;
```

DFA 中的单个状态

Id 是状态的编号，set 是状态中所包含的所有 LR(1)分析式。

```
public class DFAState {

    public int id ;
    public ArrayList<LRDerivation> set = new ArrayList<LRDerivation>();
```

DFA

```
public class DFA {

    public ArrayList<DFAState> states = new ArrayList<DFAState>();
```

Description of core Algorithms

计算一个文法符号的 first 集合

```
/**
 * 计算所有符号的first集合
 * 中间需要若干步推导的使用一个递归方法解决问题
 */
private static void addFirst(){
    //将所有的终结符的first都设为本身
    Iterator<String> iterVT = VT.iterator();
    while(iterVT.hasNext()){
        String vt = iterVT.next();
        firstMap.put(vt,new TreeSet<String>());
        firstMap.get(vt).add(vt);
    }
    //计算所有非终结符的first集合
    Iterator<String> iterVN = VN.iterator();
    while(iterVN.hasNext()){
        String vn = iterVN.next();
        firstMap.put(vn, new TreeSet<String>());//因为后续操作没有交叉涉及firstMap，所以不必分成两个while循环，合成一起即可
        int dSize = F.size();
        for(int i = 0;i < dSize;i++){
            Derivation d = F.get(i);
            if(d.left.equals(vn)){//其实可以到后面抽象成一个方法获取，这里懒得改了
                if(VT.contains(d.list.get(0))){//如果是产生式右端第一个文法符号是一个终结符，则直接添加
                    firstMap.get(vn).add(d.list.get(0));
                } else {//如果产生式右端第一个文法符号是个非终结符，则需要进行递归查找
                    firstMap.get(vn).addAll(findFirst(d.list.get(0)));
                }
            }
        }
    }
}
```

这里的 addFirst 方法的执行流程是：首先在 HashMap<String,TreeSet<String>>类型的用于记录各个文法符号对应 first 集合的记录，然后开始逐个遍历文法符号。如果文法符号是一个终结符，则直接将文法符号本身加入到相应的集合中；如果是一个非终结符，则开始遍历以该终结符为左端的产生式，如果产生式右端第一个文法符号是一个终结符，则将此文法符号加入到相应的集合中，如果是一个非终结符，则需要调用下面的递归过程获取可以推倒得出的终结符加入到 first 中。

```
/**
 * 一个用于查找first的递归函数
 * @param vn
 * @return
 */
private static TreeSet<String> findFirst(String vn){
    TreeSet<String> set = new TreeSet<String>();
    for(Derivation d:F){
        if(d.left.equals(vn)){
            if(VT.contains(d.list.get(0))){//如果是个终结符，则直接加入
                set.add(d.list.get(0));
            } else {
                if(!vn.equals(d.list.get(0))){//去除类似于E->E*E这样的左递归，从而有效避免栈溢出
                    set.addAll(findFirst(d.list.get(0)));//再次递归
                }
            }
        }
    }
    return set;
}
```

以上为根据推倒进行 first 计算的递归方法，通过遍历非终结符为左端的产生式进行获取。如果产生式对应右端是一个终结符，则直接加入，递归结束；如果是一个非终结符，则需要进行进一步递归。值得注意的是，对于类似 E->E*E 这样存在左递归的产生式，如果不加以处理，这里的递归会无穷无尽的进行下去，所以在这里要加入一个限制，即存在左递归，则递归调用过程立即结束，这样可以有效地避免出现无限的递归调用导致 StackOverflowException。

构建一个用于产生语法分析表的 DFA

```
private void createDFA(){
    this.dfa = new DFA();
    DFAState state0 = new DFAState(0);
    state0.addNewDerivation(new LRDerivation(getDerivation("S'").get(0),"$",0));//首先加入S'->·S,$
    for(int i = 0;i < state0.set.size();i++){
        LRDerivation lrd = state0.set.get(i);
        if(lrd.index < lrd.d.list.size()){
            String A = lrd.d.list.get(lrd.index);//获取·后面的文法符号
            String b = null;//求跟A的一级+a
            if(lrd.index==lrd.d.list.size()-1){
                b = lrd.lr;
            } else {
                b = lrd.d.list.get(lrd.index+1);
            }
            if(CFG.VN.contains(A)){
                ArrayList<String> firstB = first(b);
                ArrayList<Derivation> dA = getDerivation(A);
                for(int j=0,length1=dA.size();j<length1;j++){
                    for(int k=0,length2=firstB.size();k<length2;k++){
                        LRDerivation lrd1 = new LRDerivation(dA.get(j),firstB.get(k),0);
                        state0.addNewDerivation(lrd1);
                    }
                }
            }
        }
    }
    dfa.states.add(state0);
    //state0建立成功后开始递归建立其他的状态
    ArrayList<String> gotoPath = state0.getGotoPath();
    for(String path:gotoPath){
        ArrayList<LRDerivation> list = state0.getLRDs(path);//直接通过路径得到下一个状态的情况
        addState(0,path,list);//开始进行递归，建立用于分析的DFA
    }
}
```

构建一个 DFA 仍然是一个递归的过程，因为涉及到要将每个 LR（1）分析式中的点右端第一个文法符号的产生式找出来进行遍历并添加到相关的状态中，还有另一点很重要，就是我们需要通过获取 path 从而进入下一个状态，这两点共同决定了 DFA 的构建必然是一个递归的过程。

在上一个 createDFA（）方法中我们先利用产生式 S’->S 构建一个状态，然后获取可能的进入下一个状态的路径和对应的各个路径的进入下一个状态的产生式后，开始调用下面 addState 的递归方法。该递归方法的递归结束条件有两个：一个是所有的产生式的点都到达了产生式的末尾，还有一个是检测到已经有一个内容完全相同的状态被添加。

```
/**
 * 通过输入一个从上一个状态传下来的LR产生式的list获取下一个状态，
 * 如果该状态已经存在，则不作任何操作，跳出递归，如果该状态不存在，则加入该状态，继续进行递归
 * @param list
 * @param lastState 上一个状态的编号
 */
private void addState(int lastState,String path,ArrayList<LRDerivation> list){
    DFAState temp = new DFAState(0);
    for(int i = 0;i < list.size();i++){
        list.get(i).index++;
        temp.addNewDerivation(list.get(i));
    }
    for(int i = 0;i < temp.set.size();i++){
        if(temp.set.get(i).d.list.size() != temp.set.get(i).index){
            String A = temp.set.get(i).d.list.get(temp.set.get(i).index);
            String B = null;
            if(temp.set.get(i).index+1 == temp.set.get(i).d.list.size()){
                B = temp.set.get(i).lr;
            } else {
                B = temp.set.get(i).d.list.get(temp.set.get(i).index+1);
            }
            ArrayList<Derivation> dA = getDerivation(A);
            ArrayList<String> firstB = first(B);
            for(int j = 0;j < dA.size();j++){
                for(int k = 0;k < firstB.size();k++){
                    LRDerivation lrd = new LRDerivation(dA.get(j),firstB.get(k),0);
                    if(!temp.contains(lrd)){
                        temp.addNewDerivation(lrd);
                    }
                }
            }
        }
    }
    for(int i = 0;i < dfa.states.size();i++){
        if(dfa.states.get(i).equalTo(temp)){
            gotoStart.add(lastState);
            gotoEnd.add(i);
            gotoPath.add(path);

            return;
        }
    }
    temp.id = dfa.states.size();
    dfa.states.add(temp);
    gotoStart.add(lastState);
    gotoEnd.add(temp.id);
    gotoPath.add(path);
    ArrayList<String> gotoPath = temp.getGotoPath();
    for(String p:gotoPath){
        ArrayList<LRDerivation> l = temp.getLRDs(p);//直接通过路径传到下一个状态的情况
        addState(temp.id,p,l);
    }
}
```

语法分析表的构建

构建该分析表的过程中最复杂的问题莫过于获取 Action 和 Goto，把已经生成的 DFA 的状态一个个拿过来分析必然会是个繁琐的过程，所以在构建 DFA 的过程中我已经将所有的路径信息记录在下面三个容器中。

```
private ArrayList<Integer> gotoStart = new ArrayList<Integer>();
private ArrayList<Integer> gotoEnd = new ArrayList<Integer>();
private ArrayList<String> gotoPath = new ArrayList<String>();
/**
```

然后下面给出填充分析表的代码。其实就是通过这三个容器获取的信息，来对语法分析表进行填充，过程比较简单，这里不再赘述。

```
private void createAnalyzeTable(){
    for(int i = 0;i < gotoTable.length;i++){
        for(int j = 0;j < gotoTable[0].length;j++){
            gotoTable[i][j] = -1;
        }
    }
    for(int i = 0;i < actionTable.length;i++){
        for(int j = 0;j < actionTable[0].length;j++){
            actionTable[i][j] = AnalyzeTable.error;
        }
    }
    //先管语法分析表的goto部分
    int gotoCount = this.gotoStart.size();
    for(int i = 0;i < gotoCount;i++){
        int start = gotoStart.get(i);
        int end = gotoEnd.get(i);
        String path = gotoPath.get(i);
        int pathIndex = gotoIndex(path);
        this.gotoTable[start][pathIndex] = end;
    }
    //先管语法分析表的action部分
    int stateCount = dfa.states.size();
    for(int i = 0;i < stateCount;i++){
        DFAState state = dfa.get(i);//获取dfa的单个状态
        for(LRDerivation lrd:state.set){//对每一个进行分析
            if(lrd.index == lrd.d.list.size()){
                if(!lrd.d.left.equals("S'")){
                    int derivationIndex = derivationIndex(lrd.d);
                    String value = "r"+derivationIndex;
                    actionTable[i][actionIndex(lrd.lr)] = value;//设为规约
                } else {
                    actionTable[i][actionIndex("$")] = AnalyzeTable.acc;//设为接受
                }
            } else {
                String next = lrd.d.list.get(lrd.index);//获取·后面的文法符号
                if(CFG.VT.contains(next)){//必须是一个终结符号
                    if(gotoTable[i][gotoIndex(next)] != -1){
                        actionTable[i][actionIndex(next)] = "s"+gotoTable[i][gotoIndex(next)];
                    }
                }
            }
        }
    }
}
```

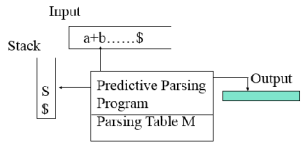
利用语法分析表中的 Action 和 Goto 进行语法分析

在这个过程中，我们需要用到下列数据结构，对应于语法分析器的模型

```
public class SyntaxParser {

    private LexicalAnalyzer lex;//词法分析器
    private ArrayList<Token> tokenList;//从词法分析器获得的所有token,相当于模型中的输入
    private int length;//tokenlist的长度
    private int index;//现在所指的token位置

    private AnalyzeTable table;//构造的语法分析表
    private Stack<Integer> stateStack;//用于存储相应的状态
```



分析过程如下，具体思想和书本所述的算法思想基本一致，此处不再赘述。

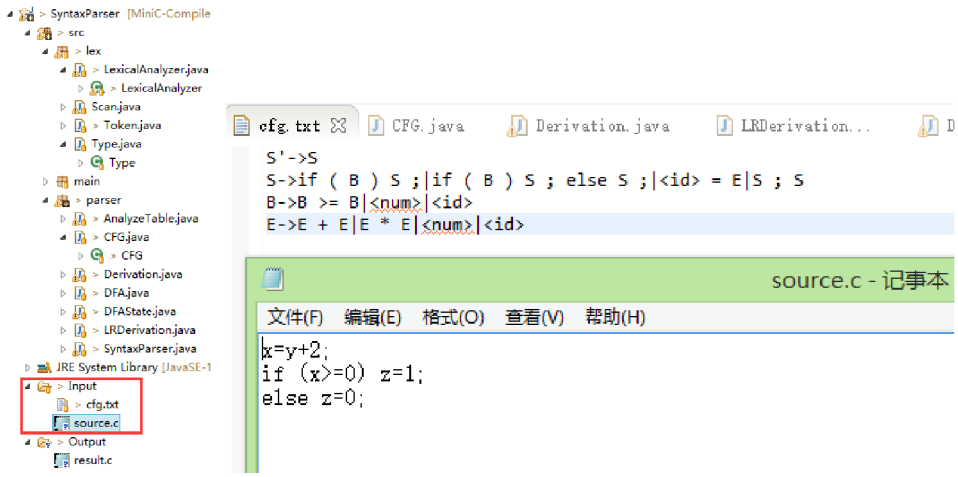
```
public void analyze(){
    while(true){
        Token token = readToken();
        int valueType = token.type;
        String value = getValue(valueType);
        int state = stateStack.lastElement();
        String action = table.ACTION(state, value);
        System.out.println(action);
        if(action.startsWith("s")){
            int newState = Integer.parseInt(action.substring(1));
            stateStack.push(newState);
            System.out.print("移入"+"\t");
            System.out.print("状态表:"+stateStack.toString()+"\t");
            System.out.print("输入:");
            printInput();
            System.out.println();
            System.out.println();
        } else if(action.startsWith("r")){
            Derivation derivation = CFG.F.get(Integer.parseInt(action.substring(1)));
            int r = derivation.list.size();
            index--;
            for(int i = 0;i < r;i++){
                stateStack.pop();
            }
            int s = table.GOTO(stateStack.lastElement(), derivation.left);
            stateStack.push(s);
            System.out.print("规约"+"\t");
            System.out.print("状态表:"+stateStack.toString()+"\t");
            System.out.print("输入:");
            printInput();
            System.out.println();

        } else if(action.equals(AnalyzeTable.acc)){
            System.out.print("语法分析完成"+"\t");
            System.out.print("状态表:"+stateStack.toString()+"\t");
            System.out.print("输入:");
            printInput();
            System.out.println();
            return;
        } else {
            error();
            return;
        }

    }
}
```

Use cases on running

这里的测试就用翟老师上次在群里举的那个例子，输入的文法和测试用的程序写在这里：



下面是运行 SyntaxParser 类中的 main 方法后的输出结果：（略长，希望能耐心拖完。。。）
（系统构建的自动机）

```
I0:
S'-> ·S ,$
S-> ·if ( B ) S ; ,$
S-> ·if ( B ) S ; else S ; ,$
S-> ·<id> = E ,$
S-> ·S ; S ,$
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I1:
S'-> S· ,$
S-> S ·; S ,$
S-> S ·; S ,;
I2:
S-> S ; ·S ,$
S-> S ; ·S ,;
S-> ·if ( B ) S ; ,$
S-> ·if ( B ) S ; else S ; ,$
S-> ·<id> = E ,$
S-> ·S ; S ,$
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I3:
S-> S ; S· ,$
S-> S ; S· ,;
S-> S ·; S ,$
S-> S ·; S ,;
I4:
S-> if ·( B ) S ; ,$
S-> if ·( B ) S ; else S ; ,$
S-> if ·( B ) S ; ,;
S-> if ·( B ) S ; else S ; ,;
I5:
S-> if ( ·B ) S ; ,$
S-> if ( ·B ) S ; else S ; ,$
S-> if ( ·B ) S ; ,;
S-> if ( ·B ) S ; else S ; ,;
B-> ·B >= B ,)
B-> ·<num> ,)
```

```
B-> ·<id> ,)
B-> ·B >= B ,>=
B-> ·<num> ,>=
B-> ·<id> ,>=
I6:
S-> if ( B ·) S ; ,$
S-> if ( B ·) S ; else S ; ,$
S-> if ( B ·) S ; ,;
S-> if ( B ·) S ; else S ; ,;
B-> B ·>= B ,)
B-> B ·>= B ,>=
I7:
S-> if ( B ) ·S ; ,$
S-> if ( B ) ·S ; else S ; ,$
S-> if ( B ) ·S ; ,;
S-> if ( B ) ·S ; else S ; ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I8:
S-> if ( B ) S ·; ,$
S-> if ( B ) S ·; else S ; ,$
S-> if ( B ) S ·; ,;
S-> if ( B ) S ·; else S ; ,;
S-> S ·; S ,;
I9:
S-> if ( B ) S ;· ,$
S-> if ( B ) S ; ·else S ; ,$
S-> if ( B ) S ;· ,;
S-> if ( B ) S ; ·else S ; ,;
S-> S ; ·S ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I10:
S-> if ( B ) S ; else ·S ; ,$
S-> if ( B ) S ; else ·S ; ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I11:
```

```
S-> if ( B ) S ; else S ·; ,$
S-> if ( B ) S ; else S ·; ,;
S-> S ·; S ,;
I12:
S-> if ( B ) S ; else S ;· ,$
S-> if ( B ) S ; else S ;· ,;
S-> S ; ·S ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I13:
S-> S ; S· ,;
S-> S ·; S ,;
I14:
S-> S ; ·S ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I15:
S-> if ·( B ) S ; ,;
S-> if ·( B ) S ; else S ; ,;
I16:
S-> if ( ·B ) S ; ,;
S-> if ( ·B ) S ; else S ; ,;
B-> ·B >= B ,)
B-> ·<num> ,)
B-> ·<id> ,)
B-> ·B >= B ,>=
B-> ·<num> ,>=
B-> ·<id> ,>=
I17:
S-> if ( B ·) S ; ,;
S-> if ( B ·) S ; else S ; ,;
B-> B ·>= B ,)
B-> B ·>= B ,>=
I18:
S-> if ( B ) ·S ; ,;
S-> if ( B ) ·S ; else S ; ,;
S-> ·if ( B ) S ; ,;
S-> ·if ( B ) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
```

I19:
S-> if (B) S ·; ,;
S-> if (B) S ·; else S ; ,;
S-> S ·; S ,;
I20:
S-> if (B) S ;· ,;
S-> if (B) S ; ·else S ; ,;
S-> S ; ·S ,;
S-> ·if (B) S ; ,;
S-> ·if (B) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I21:
S-> if (B) S ; else ·S ; ,;
S-> ·if (B) S ; ,;
S-> ·if (B) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I22:
S-> if (B) S ; else S ·; ,;
S-> S ·; S ,;
I23:
S-> if (B) S ; else S ;· ,;
S-> S ; ·S ,;
S-> ·if (B) S ; ,;
S-> ·if (B) S ; else S ; ,;
S-> ·<id> = E ,;
S-> ·S ; S ,;
I24:
S-> <id> ·= E ,;
I25:
S-> <id> = ·E ,;
E-> ·E + E ,;
E-> ·E * E ,;
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+
E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*

I26:
S-> <id> = E· ,;
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I27:
E-> E + ·E ,;
E-> E + ·E ,+
E-> E + ·E ,*
E-> ·E + E ,;
E-> ·E * E ,;
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+
E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*
I28:
E-> E + E· ,;
E-> E + E· ,+
E-> E + E· ,*
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I29:
E-> E * ·E ,;
E-> E * ·E ,+
E-> E * ·E ,*
E-> ·E + E ,;
E-> ·E * E ,;
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+

E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*
I30:
E-> E * E· ,;
E-> E * E· ,+
E-> E * E· ,*
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I31:
E-> <num>· ,;
E-> <num>· ,+
E-> <num>· ,*
I32:
E-> <id>· ,;
E-> <id>· ,+
E-> <id>· ,*
I33:
B-> B >= ·B ,)
B-> B >= ·B ,>=
B-> ·B >= B ,)
B-> ·<num> ,)
B-> ·<id> ,)
B-> ·B >= B ,>=
B-> ·<num> ,>=
B-> ·<id> ,>=
I34:
B-> B >= B· ,)
B-> B >= B· ,>=
B-> B ·>= B ,)
B-> B ·>= B ,>=
I35:
B-> <num>· ,)
B-> <num>· ,>=
I36:
B-> <id>· ,)
B-> <id>· ,>=

I37:
S-> <id> ·= E ,$
S-> <id> ·= E ,;
I38:
S-> <id> = ·E ,$
S-> <id> = ·E ,;
E-> ·E + E ,$
E-> ·E * E ,$
E-> ·<num> ,$
E-> ·<id> ,$
E-> ·E + E ,;
E-> ·E * E ,;
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+
E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*
I39:
S-> <id> = E· ,$
S-> <id> = E· ,;
E-> E ·+ E ,$
E-> E ·* E ,$
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I40:
E-> E + ·E ,$
E-> E + ·E ,;
E-> E + ·E ,+
E-> E + ·E ,*
E-> ·E + E ,$
E-> ·E * E ,$
E-> ·<num> ,$
E-> ·<id> ,$
E-> ·E + E ,;
E-> ·E * E ,;

```
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+
E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*
I41:
E-> E + E· ,$
E-> E + E· ,;
E-> E + E· ,+
E-> E + E· ,*
E-> E ·+ E ,$
E-> E ·* E ,$
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I42:
E-> E * ·E ,$
E-> E * ·E ,;
E-> E * ·E ,+
E-> E * ·E ,*
E-> ·E + E ,$
E-> ·E * E ,$
E-> ·<num> ,$
E-> ·<id> ,$
E-> ·E + E ,;
E-> ·E * E ,;
E-> ·<num> ,;
E-> ·<id> ,;
E-> ·E + E ,+
E-> ·E * E ,+
E-> ·<num> ,+
E-> ·<id> ,+
E-> ·E + E ,*
E-> ·E * E ,*
E-> ·<num> ,*
E-> ·<id> ,*
```

I43:
E-> E * E· ,$
E-> E * E· ,;
E-> E * E· ,+
E-> E * E· ,*
E-> E ·+ E ,$
E-> E ·* E ,$
E-> E ·+ E ,;
E-> E ·* E ,;
E-> E ·+ E ,+
E-> E ·* E ,+
E-> E ·+ E ,*
E-> E ·* E ,*
I44:
E-> <num>· ,$
E-> <num>· ,;
E-> <num>· ,+
E-> <num>· ,*
I45:
E-> <id>· ,$
E-> <id>· ,;
E-> <id>· ,+
E-> <id>· ,*

（分析表，好大……由于一行太长了，所以复制粘贴后会变形。。。）

	(	)	*	+	;	<id>	<num>	=	>=	else	if	$	(	)	*	+
	;	<id>	<num>	=	>=	else	if	B	E	S	S'					
0	X	X	X	X	X	s37	X	X	X	X	s4	X	X	X	X	X	X	37	X
	X	X	X	4	X	X	1	X											
1	X	X	X	X	s2	X	X	X	X	X	X	acc	X	X	X	X	2	X	X
	X	X	X	X	X	X	X	X											
2	X	X	X	X	X	s37	X	X	X	X	s4	X	X	X	X	X	X	37	X
	X	X	X	4	X	X	3	X											
3	X	X	X	X	s2	X	X	X	X	X	X	r4	X	X	X	X	2	X	X
	X	X	X	X	X	X	X	X											
4	s5	X	X	X	X	X	X	X	X	X	X	X	5	X	X	X	X	X	X
	X	X	X	X	X	X	X	X											
5	X	X	X	X	X	s36	s35	X	X	X	X	X	X	X	X	X	X	36	35
	X	X	X	X	6	X	X	X											
6	X	s7	X	X	X	X	X	X	s33	X	X	X	X	7	X	X	X	X	X
	X	33	X	X	X	X	X	X											
7	X	X	X	X	X	s24	X	X	X	X	s15	X	X	X	X	X	X	24	X
	X	X	X	15	X	X	8	X											
8	X	X	X	X	s9	X	X	X	X	X	X	X	X	X	X	X	9	X	X

```
    X   X   X   X   X   X   X   X
9   X   X   X   X   r1  s24 X   X   X   s10 s15 r1  X   X   X   X   X   24  X
    X   X   10  15  X   X   13  X
10  X   X   X   X   X   s24 X   X   X   X   s15 X   X   X   X   X   X   24  X
    X   X   X   15  X   X   11  X
11  X   X   X   X   s12 X   X   X   X   X   X   X   X   X   X   X   12  X   X
    X   X   X   X   X   X   X   X
12  X   X   X   X   r2  s24 X   X   X   X   s15 r2  X   X   X   X   X   24  X
    X   X   X   15  X   X   13  X
13  X   X   X   X   s14 X   X   X   X   X   X   X   X   X   X   X   14  X   X
    X   X   X   X   X   X   X   X
14  X   X   X   X   X   s24 X   X   X   X   s15 X   X   X   X   X   X   24  X
    X   X   X   15  X   X   13  X
15  s16 X   X   X   X   X   X   X   X   X   X   X   16  X   X   X   X   X   X
    X   X   X   X   X   X   X   X
16  X   X   X   X   X   s36 s35 X   X   X   X   X   X   X   X   X   X   36  35
    X   X   X   X   17  X   X   X
17  X   s18 X   X   X   X   X   X   s33 X   X   X   X   18  X   X   X   X   X
    X   33  X   X   X   X   X   X
18  X   X   X   X   X   s24 X   X   X   X   s15 X   X   X   X   X   X   24  X
    X   X   X   15  X   X   19  X
19  X   X   X   X   s20 X   X   X   X   X   X   X   X   X   X   X   20  X   X
    X   X   X   X   X   X   X   X
20  X   X   X   X   r1  s24 X   X   X   s21 s15 X   X   X   X   X   X   24  X
    X   X   21  15  X   X   13  X
21  X   X   X   X   X   s24 X   X   X   X   s15 X   X   X   X   X   X   24  X
    X   X   X   15  X   X   22  X
22  X   X   X   X   s23 X   X   X   X   X   X   X   X   X   X   X   23  X   X
    X   X   X   X   X   X   X   X
23  X   X   X   X   r2  s24 X   X   X   X   s15 X   X   X   X   X   X   24  X
    X   X   X   15  X   X   13  X
24  X   X   X   X   X   X   X   s25 X   X   X   X   X   X   X   X   X   X   X
    25  X   X   X   X   X   X   X
25  X   X   X   X   X   s32 s31 X   X   X   X   X   X   X   X   X   X   32  31
    X   X   X   X   X   26  X   X
26  X   X   s29 s27 r3  X   X   X   X   X   X   X   X   X   X   29  27  X   X   X
    X   X   X   X   X   X   X   X
27  X   X   X   X   X   s32 s31 X   X   X   X   X   X   X   X   X   X   32  31
    X   X   X   X   X   28  X   X
28  X   X   s29 s27 r8  X   X   X   X   X   X   X   X   X   X   29  27  X   X   X
    X   X   X   X   X   X   X   X
29  X   X   X   X   X   s32 s31 X   X   X   X   X   X   X   X   X   X   32  31
    X   X   X   X   X   30  X   X
30  X   X   s29 s27 r9  X   X   X   X   X   X   X   X   X   X   29  27  X   X   X
```

```
    X   X   X   X   X   X   X   X
31  X   X   r10 r10 r10 X   X   X   X   X   X   X   X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
32  X   X   r11 r11 r11 X   X   X   X   X   X   X   X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
33  X   X   X   X   X   s36 s35 X   X   X   X   X   X   X   X   X   X   36  35
    X   X   X   X   34  X   X   X
34  X   r5  X   X   X   X   X   X   s33 X   X   X   X   X   X   X   X   X   X
    X   33  X   X   X   X   X   X
35  X   r6  X   X   X   X   X   X   r6  X   X   X   X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
36  X   r7  X   X   X   X   X   X   r7  X   X   X   X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
37  X   X   X   X   X   X   X   s38 X   X   X   X   X   X   X   X   X   X   X
    38  X   X   X   X   X   X   X
38  X   X   X   X   X   s45 s44 X   X   X   X   X   X   X   X   X   X   45  44
    X   X   X   X   X   39  X   X
39  X   X   s42 s40 r3  X   X   X   X   X   X   r3  X   X   42  40  X   X   X
    X   X   X   X   X   X   X   X
40  X   X   X   X   X   s45 s44 X   X   X   X   X   X   X   X   X   X   45  44
    X   X   X   X   X   41  X   X
41  X   X   s42 s40 r8  X   X   X   X   X   X   r8  X   X   42  40  X   X   X
    X   X   X   X   X   X   X   X
42  X   X   X   X   X   s45 s44 X   X   X   X   X   X   X   X   X   X   45  44
    X   X   X   X   X   43  X   X
43  X   X   s42 s40 r9  X   X   X   X   X   X   r9  X   X   42  40  X   X   X
    X   X   X   X   X   X   X   X
44  X   X   r10 r10 r10 X   X   X   X   X   X   r10 X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
45  X   X   r11 r11 r11 X   X   X   X   X   X   r11 X   X   X   X   X   X   X
    X   X   X   X   X   X   X   X
```

（分析过程）

s37

移入状态表:[0, 37] 输入:= y + 2 ; if (x >= 0) z = 1 ; else z = 0 ; $

s38

移入状态表:[0, 37, 38] 输入:y + 2 ; if (x >= 0) z = 1 ; else z = 0 ; $

s45

移入状态表:[0, 37, 38, 45] 输入:+ 2 ; if (x >= 0) z = 1 ; else z = 0 ; $

r11

规约状态表:[0, 37, 38, 39] 输入:+ 2 ; if (x >= 0) z = 1 ; else z = 0 ; $

```
s40
移入状态表:[0, 37, 38, 39, 40] 输入:2 ; if ( x >= 0 ) z = 1 ; else z = 0 ; $

s44
移入状态表:[0, 37, 38, 39, 40, 44] 输入:; if ( x >= 0 ) z = 1 ; else z = 0 ;
$

r10
规约状态表:[0, 37, 38, 39, 40, 41] 输入:; if ( x >= 0 ) z = 1 ; else z = 0 ;
$
r8
规约状态表:[0, 37, 38, 39] 输入:; if ( x >= 0 ) z = 1 ; else z = 0 ; $
r3
规约状态表:[0, 1]   输入:; if ( x >= 0 ) z = 1 ; else z = 0 ; $
s2
移入状态表:[0, 1, 2]    输入:if ( x >= 0 ) z = 1 ; else z = 0 ; $

s4
移入状态表:[0, 1, 2, 4] 输入:( x >= 0 ) z = 1 ; else z = 0 ; $

s5
移入状态表:[0, 1, 2, 4, 5] 输入:x >= 0 ) z = 1 ; else z = 0 ; $

s36
移入状态表:[0, 1, 2, 4, 5, 36] 输入:>= 0 ) z = 1 ; else z = 0 ; $

r7
规约状态表:[0, 1, 2, 4, 5, 6]  输入:>= 0 ) z = 1 ; else z = 0 ; $
s33
移入状态表:[0, 1, 2, 4, 5, 6, 33]  输入:0 ) z = 1 ; else z = 0 ; $

s35
移入状态表:[0, 1, 2, 4, 5, 6, 33, 35]  输入:) z = 1 ; else z = 0 ; $

r6
规约状态表:[0, 1, 2, 4, 5, 6, 33, 34]  输入:) z = 1 ; else z = 0 ; $
r5
规约状态表:[0, 1, 2, 4, 5, 6]  输入:) z = 1 ; else z = 0 ; $
s7
移入状态表:[0, 1, 2, 4, 5, 6, 7]   输入:z = 1 ; else z = 0 ; $

s24
移入状态表:[0, 1, 2, 4, 5, 6, 7, 24]   输入:= 1 ; else z = 0 ; $
```

s25
移入状态表:[0, 1, 2, 4, 5, 6, 7, 24, 25] 输入:1 ; else z = 0 ; $

s31
移入状态表:[0, 1, 2, 4, 5, 6, 7, 24, 25, 31] 输入:; else z = 0 ; $

r10
规约状态表:[0, 1, 2, 4, 5, 6, 7, 24, 25, 26] 输入:; else z = 0 ; $
r3
规约状态表:[0, 1, 2, 4, 5, 6, 7, 8]输入:; else z = 0 ; $
s9
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9] 输入:else z = 0 ; $

s10
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10] 输入:z = 0 ; $

s24
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 24] 输入:= 0 ; $

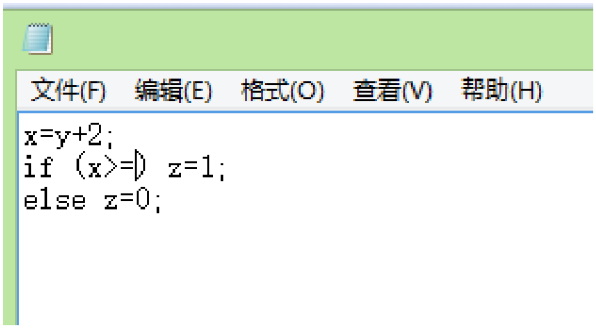
s25
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 24, 25] 输入:0 ; $

s31
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 24, 25, 31]输入:; $

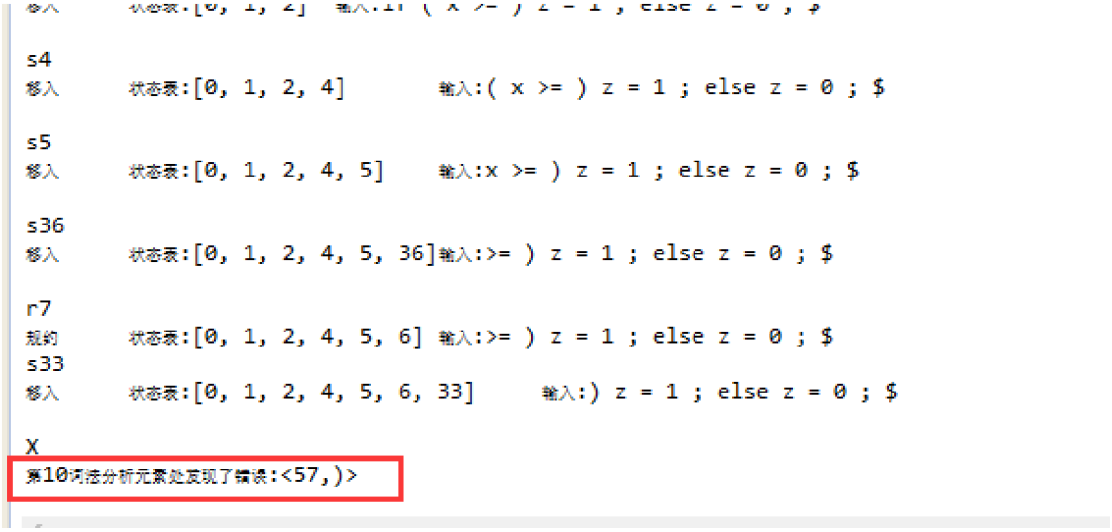
r10
规约状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 24, 25, 26]输入:; $
r3
规约状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11] 输入:; $
s12
移入状态表:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12] 输入:$

r2
规约状态表:[0, 1, 2, 3] 输入:$
r4
规约状态表:[0, 1] 输入:$
acc
语法分析完成 状态表:[0, 1] 输入:

关于报错，改个东西试试……



当读取到）对应的那个 token 时，问题来了……



Problems occurred and related solutions

刚刚开始对自动机的自动构建无从下手,后面通过自己建立数据结构一步步解决问题,成功构建了自动机。

Your feelings and comments

通过一个实际的语法分析器的具体实现，考虑了很多进行语法分析的过程中的细节，巩固了对课堂上所讲解的相关编译理论的掌握，加深了对语法分析过程的理解，增强了动手进行程序设计的能力，为以后自行实现一个完整的编译器和研究高级程序设计语言的内部原理打下了扎实的基础。