姓名：张振阳

学号：20142100014

备注：testcase中文法主要来源于同学的方法，因此会有较大的相同之处

实现思路：

1. 首先从文件中读入input.bnf 读入成功后就对其进行处理，首先是对文法中存在“|”的处理，将其分开成两部分，然后是根据空格分割，同时获取文本中的非终结符号
2. 获取文法中的所有非终结符号的firstpos
3. 获取文法中的所有非终结符号的followpos
4. 结合firstpos和followpos判断该文法是否为正确的LL（1）文法
5. 如果是正确的文法就制作分析表
6. 根据分析表进行tok的判断

主要代码及说明：

private static final Map<String,ArrayList<ArrayList<String>>> GrammarInput=new LinkedHashMap();

private static final Map<String,ArrayList<String>> firstpos=new LinkedHashMap();

private static final ArrayList<String> nonTermSymbol=new ArrayList();

private static final Map<String,ArrayList<String>> followpos=new LinkedHashMap();

private static final Map<String,Map<String,ArrayList<String>>> analtsistable=new LinkedHashMap<>();

//数据结构

GrammarInput用于保存文法读入后的保存

firstpos用于保存文法的firstpos

nonTermSymbol 用于保存非终结符号

followpos 用于保存followpos

analtsistable 用于保存分析表

public static void main(String[] args) {

String root=args[0];

String filename=root+"/input.bnf";

readfile(filename);

getfirstpos();

getfollowpos();

boolean judge=false;

judge=judgeLL();

if(judge){

maketable();

File folderFile=new File(args[0]);

String [] listfileStrings=folderFile.list();

for(int i=0;i<listfileStrings.length;i++){

System.out.println(listfileStrings[i]);

if(analysis(root+"/"+listfileStrings[i])){

System.out.println("yes");

}

else{

System.out.println("no");

}

}

}

}

}

主函数：主要用于调用各个函数以及最后结果的输出

public static void readfile(String filename){

File file= new File(filename);

BufferedReader reader=null;

try{

//System.out.println("Readfile");

reader=new BufferedReader(new FileReader(file));

String tempstr=null;

int line=1;

while((tempstr=reader.readLine())!=null){

String[] input=tempstr.split(" ::=");

String leftword=input[0];

nonTermSymbol.add(leftword);

String[] rightword=input[1].split("\\|");

int len=rightword.length;

ArrayList<ArrayList<String>> arraylist1=new ArrayList<ArrayList<String>>();

for(int i=0;i<len;i++){

String[] rightword1=rightword[i].trim().split(" ");

ArrayList<String> arraylist2=new ArrayList();

for(int j=0;j<rightword1.length;j++){

arraylist2.add(rightword1[j]);

}

arraylist1.add(arraylist2);

}

GrammarInput.put(leftword, arraylist1);

}

}catch(IOException e){

e.printStackTrace();

}finally{

if(reader!=null){

try{

reader.close();

}catch(IOException e){

}

}

}

}

读取bnf文件的函数功能 读入后先对“|”进行分析，如果存在就分为两部分，然后根据“ ”进行分析，分成各个元素

public static void getfirstpos(){

for(int i=0;i<nonTermSymbol.size();i++){

Stack<String> remmerberfirst=new Stack();

String nonstr=nonTermSymbol.get(i);

remmerberfirst.push(nonstr);

ArrayList<String> first=new ArrayList();

while(!remmerberfirst.empty()){

String stackstr=remmerberfirst.pop();

ArrayList<ArrayList<String>> list1=GrammarInput.get(stackstr);

int list1len=list1.size();

for(int j=0;j<list1len;j++){

ArrayList<String> list2=list1.get(j);

String list2first=list2.get(0);

if(nonTermSymbol.contains(list2first)){

if(remmerberfirst.indexOf(list2first)==-1){

remmerberfirst.push(list2first);

}

}

else{

first.add(list2first);

}

}

}

firstpos.put(nonstr, first);

}

}

获取文法的firstpos 将某个要求的非终结符号先入栈，然后对其进行处理，如果能推出非终结符号就将推出的非终结符号入栈，为终结符号就直接加入到firstpos数组中，然后重复对栈的操作

public static void getfollowpos(){

ArrayList<ArrayList<String>> templist = new ArrayList<ArrayList<String>>();

for(int i=0;i<nonTermSymbol.size();i++){

ArrayList<String> tempunit = new ArrayList<String>();

templist.add(tempunit);

}

templist.get(0).add("$");

for(int i=0;i<nonTermSymbol.size();i++){

ArrayList<ArrayList<String>> group = GrammarInput.get(nonTermSymbol.get(i));

for(int j=0;j<group.size();j++){

ArrayList<String> unit = group.get(j);

for(int k=0;k<unit.size()-1;k++){

if(inNotEnd(unit.get(k))){

int n = 1;

boolean flag;

do {

flag = false;

if(inNotEnd(unit.get(k+n))){

ArrayList<String> strlist = firstpos.get(unit.get(k+n));

for(int m=0;m<strlist.size();m++){

if(strlist.get(m).equals("\"\"")){

flag = true;

}

else {

templist.get(getNotEndPosition(unit.get(k)))

.add(strlist.get(m));

}

}

n++;

}

else {

templist.get(getNotEndPosition(unit.get(k))).add(unit.get(k+n));

}

}while (flag&&((k+n)<unit.size()));

}

}

}

}

for(int i=0;i<nonTermSymbol.size();i++){

ArrayList<ArrayList<String>> group = GrammarInput.get(nonTermSymbol.get(i));

for(int j=0;j<group.size();j++){

ArrayList<String> unit = group.get(j);

int n = 1;

boolean flag;

do {

flag = false;

if(inNotEnd(unit.get(unit.size()-n))){

if(firstpos.get(unit.get(unit.size()-n)).contains("\"\""))

flag = true;

if(!unit.get(unit.size()-n).equals(nonTermSymbol.get(i))){

templist.get(getNotEndPosition(unit.get(unit.size()-n)))

.addAll(templist.get(getNotEndPosition(nonTermSymbol.get(i))));

}

n++;

}

}while (flag&&((unit.size()-n)>=0));

}

}

for(int i=0;i<nonTermSymbol.size();i++){

followpos.put(nonTermSymbol.get(i),removeSame(templist.get(i)));

}

}

private static ArrayList<String> removeSame(ArrayList<String> list){

HashSet<String> set = new HashSet<String>(list);

ArrayList<String> temp = new ArrayList<String>(set);

return temp;

}

private static boolean inNotEnd(String s){

for(int i=0;i<nonTermSymbol.size();i++){

if(s.equals(nonTermSymbol.get(i))){

return true;

}

}

return false;

}

private static int getNotEndPosition(String s){

for(int i=0;i<nonTermSymbol.size();i++){

if(s.equals(nonTermSymbol.get(i))){

return i;

}

}

return -1;

}

获取followpos的功能函数，主要是根据书本推荐的方法进行处理

public static boolean judgeLL(){

boolean check=true;

loop:for(String key1:GrammarInput.keySet()){

ArrayList<ArrayList<String>> list1=GrammarInput.get(key1);

int list1len=list1.size();

if(list1len>1){

for(int i=0;i<list1len;i++){

ArrayList<String> list2=list1.get(i);

for(int j=0;j<list1len;j++){

if(i==j)continue;

else{

ArrayList<String> list3=list1.get(j);

String list2first=list2.get(0);

String list3first=list3.get(0);

if(nonTermSymbol.contains(list2first)&&nonTermSymbol.contains(list3first)){

ArrayList<String> list4=firstpos.get(list2first);

ArrayList<String> list5=firstpos.get(list3first);

ArrayList<String> list6=new ArrayList<>();

list6.addAll(list4);

list6.retainAll(list5);

if(list6.size()>0){

check=false;

break loop;

}

else{

if(list4.contains("\"\"")){

ArrayList<String> list7=followpos.get(key1);

ArrayList<String> list8=new ArrayList<>();

list8.addAll(list5);

list8.retainAll(list7);

if(list8.size()>0){

check=false;

break loop;

}

}

if(list5.contains("\"\"")){

ArrayList<String> list7=followpos.get(key1);

ArrayList<String> list8=new ArrayList<>();

list8.addAll(list4);

list8.retainAll(list7);

if(list8.size()>0){

check=false;

break loop;

}

}

}

}

else if(nonTermSymbol.contains(list2first)&&!nonTermSymbol.contains(list3first)){

ArrayList<String> list4=firstpos.get(list2first);

if(list4.contains(list3first)){

check=false;

break loop;

}

else{

if(list4.contains("\"\"")){

ArrayList<String> list5=followpos.get(key1);

if(list5.contains(list3first)){

check=false;

break loop;

}

}

else if(list3first.equals("\"\"")){

ArrayList<String> list5=followpos.get(key1);

ArrayList<String> list6=new ArrayList<>();

list6.addAll(list4);

list6.retainAll(list5);

if(list6.size()>0){

check=false;

break loop;

}

}

}

}

else if(!nonTermSymbol.contains(list2first)&&nonTermSymbol.contains(list3first)){

ArrayList<String> list4=firstpos.get(list3first);

if(list4.contains(list2first)){

check=false;

break loop;

}

else{

if(list4.contains("\"\"")){

ArrayList<String> list5=followpos.get(key1);

if(list5.contains(list2first)){

check=false;

break loop;

}

}

else if(list2first.equals("\"\"")){

ArrayList<String> list5=followpos.get(key1);

ArrayList<String> list6=new ArrayList<>();

list6.addAll(list4);

list6.retainAll(list5);

if(list6.size()>0){

check=false;

break loop;

}

}

}

}

else if(!nonTermSymbol.contains(list2first)&&!nonTermSymbol.contains(list3first)){

if(list2first.equals(list3first)){

check=false;

break loop;

}

else{

if(list2first.equals("\"\"")){

ArrayList<String> list4=followpos.get(key1);

if(list4.contains(list3first)){

check=false;

break loop;

}

}

else if(list3first.equals("\"\"")){

ArrayList<String> list4=followpos.get(key1);

if(list4.contains(list2first)){

check=false;

break loop;

}

}

}

}

}

}

}

}

}

if(check==true){

System.out.println("LL1");

}

else{

System.out.println("noLL1");

}

return check;

}

对文法是否为LL1文法的判断主要是对书本推荐的方法进行代码化。然后代码中要求对四种情况进行处理

public static void maketable(){

for(String key1:GrammarInput.keySet()){

ArrayList<ArrayList<String>> list1=GrammarInput.get(key1);

int list1len=list1.size();

String inkey1=key1;

Map<String,ArrayList<String>> map=new LinkedHashMap<>();

loop1:for(int i=0;i<list1len;i++){

ArrayList<String> list2=list1.get(i);

int list2len=list2.size();

boolean nullexist=false;

loop2:for(int j=0;j<list2len;j++){

String list2str=list2.get(j);

if(list2str.equals("\"\"")){

nullexist=true;

continue;

}

else if(!nonTermSymbol.contains(list2str)&&!list2str.equals("\"\"")){

map.put(list2str, list2);

break loop2;

}

else if(nonTermSymbol.contains(list2str)){

ArrayList<String> list3=firstpos.get(list2str);

int list3len=list3.size();

int list3num=0;

boolean list3null=false;

loop3:for(list3num=0;list3num<list3len;list3num++){

String tem=list3.get(list3num);

if(tem.equals("\"\"")){

nullexist=true;

list3null=true;

continue;

}

else{

map.put(tem, list2);

}

}

if(!list3null){

break loop2;

}

}

}

if(nullexist){

ArrayList<String> list4=followpos.get(key1);

for(int j=0;j<list4.size();j++){

String list4str=list4.get(j);

map.put(list4str, list2);

}

}

}

analtsistable.put(key1,map);

}

}

构建分析表的文法，将书本介绍的方法进行代码化

private static boolean analysis(String path){

Stack<String> stack = new Stack<String>();

stack.push(nonTermSymbol.get(0));

try{

String file = path;

BufferedReader f=new BufferedReader(new InputStreamReader(new FileInputStream(file),"UTF-8"));

String s;

for (s=f.readLine( ); s!=null; s=f.readLine( )){

while (true){

if (stack.empty())return false;

if(s.equals(stack.peek())){

stack.pop();

break;

}

Map<String,ArrayList<String>> map = analtsistable.get(stack.peek());

ArrayList<String> array = map.get(s);

if(array==null){

return false;

}

else {

stack.pop();

for(int i=array.size()-1;i>=0;i--){

if(!array.get(i).equals("\"\"")){

stack.push(array.get(i));

}

}

}

}

}

f.close();

while (!stack.empty()){

Map<String,ArrayList<String>> map = analtsistable.get(stack.peek());

ArrayList<String> array = map.get("$");

if(array==null){

return false;

}

else {

stack.pop();

for(int i=array.size()-1;i>=0;i--){

if(!array.get(i).equals("\"\"")){

stack.push(array.get(i));

}

}

}

}

}

catch(Exception e){

e.printStackTrace();

}

return true;

}

对文法进行分析