

语法分析实验报告

141250019 崔浩



2016-11-4

南京大学软件学院

目录

[1. 目标 2](#_Toc466051484)

[2. 内容概述 2](#_Toc466051485)

[3. 假设与依赖 2](#_Toc466051486)

[4. 思路与方法 2](#_Toc466051487)

[5. 主要数据结构 3](#_Toc466051488)

[5.1 上下文无关文法 3](#_Toc466051489)

[5.2 分析表单元 3](#_Toc466051490)

[5.3 分析表 3](#_Toc466051491)

[6. 核心算法 4](#_Toc466051492)

[7. 输入定义 5](#_Toc466051493)

[7.1 上下文无关文法 5](#_Toc466051494)

[7.2 SLR分析表 5](#_Toc466051495)

[8. 测试输入输出 6](#_Toc466051496)

[9. 总结与收获 7](#_Toc466051497)

# 目标

本次实验的目的是对编译器语法分析的过程进行模拟，这次实验我选择使用SLR方法来进行语法分析，为了更有实用性，我将上下文无关文法与SLR分析表设置为自定义的文件输入，只要文法输入正确并且与输入的分析表正确对应，该语法分析程序就可以给出对应的分析过程。

# 内容概述

本文档描述了语法分析器实验的实验内容、实验方案和实验结果。包括重要的数据结构和核心算法，上下文无关文法和SLR分析表输入格式定义，示例的输入示例和输出结果截图。

# 假设与依赖

1. 用户需要确保输入的文法不具有二义性
2. 输入的文法和分析表中，终结符和非终结符只能由一个字符组成。（不是一个字符的可以用一个字符简化表达式，如if表示为i）
3. 确保输入的文法和分析表正确，并且相互对应，且输入格式符合文档后面定义的要求。

# 思路与方法

1. 读取分析表，建立状态与ACTION和GOTO的映射
2. 读取上下文无关文法
3. 初始化状态栈、符号栈、控制器
4. 读取输入的字符，根据状态查询分析表，若为ACTION-SHIFT，将字符和shift状态压栈，读入下一个字符。若为ACTION-REDUCE，则读取相应的文法，依次将文法对应的字符和状态出栈，并根据当前状态查询GOTO，将文法左部与GOTO所指的状态压栈。若为ACCEPT，程序结束。
5. 在任何一步查表返回分析表枚举类型为NULL，则输出错误，停止分析。
6. 程序的输出为控制台输出，输出每一步符号栈、状态栈以及shift和reduce的情况。

# 主要数据结构

## 上下文无关文法

**public class** CFG {  
 **private** String **left**;  
 **private** String **right**;  
 ……   
}

left为等号左边的部分，right为等号右边的部分

## 分析表单元

**public class** PPTUnit {  
  
 **private** PPTType **type**;  
 **private int num**;  
  
}

type指示表的单元格的类型，num指示该类型对应的跳转数字

**public enum** PPTType {  
 ***NULL***,***REDUCE***,***SHIFT***,***ACCEPT***}

## 分析表

**public class** PPT {  
 **private** String **ACTION**[] = {};  
 **private** String **GOTO**[] = {};  
 **private** PPTUnit **ACTION\_DATA**[][];  
 **private int GOTO\_DATA**[][];  
 **public int** getGoto(**int** state, String tag) {  
 **return GOTO\_DATA**[state][Arrays.*asList*(**GOTO**).indexOf(tag)];  
 }  
  
 **public** PPTUnit getAction(**int** state, String tag) {  
 **return ACTION\_DATA**[state][Arrays.*asList*(**ACTION**).indexOf(tag)];  
 }  
}

action和goto用于存放表头，表头是在输入字符时查询将字符转为表中的列数，方便查询。

Action\_data和goto\_data表存放了分析表中action和goto的数据。

# 核心算法

Controller类是该程序的核心算法部分，其中startAnalyze方法为核心算法：

**public void** startAnalyze() {  
 **stateStack**.push(0);  
 **tagStack**.push(**"$"**);  
 **try** {  
 Scanner scanner = **new** Scanner(**new** FileInputStream(**"input/test.txt"**));  
 loop:**while**(scanner.hasNext()) {  
 String line = scanner.nextLine();  
 **if** (line.isEmpty()) **continue**;  
 **for** (**int** i = 0; i < line.length(); i++) {  
 **int** state = **stateStack**.peek();  
 String tag = line.charAt(i)+**""**;  
 PPTUnit unit = **ppt**.getAction(state,tag);  
 **switch** (unit.getType()) {  
 **case *SHIFT***:  
 **stateStack**.push(unit.getNum());  
 **tagStack**.push(tag);  
 System.***out***.print(**"shift "**+unit.getNum()+**" "**);  
 **break**;  
 **case *REDUCE***:  
 i--;  
 CFG cfg = **cfgList**.get(unit.getNum());  
 **for** (**int** j = cfg.getRight().length()-1; j >= 0; j--) {  
 **stateStack**.pop();  
 **tagStack**.pop();  
 }  
 **tagStack**.push(cfg.getLeft());  
 **stateStack**.push(**ppt**.getGoto(**stateStack**.peek(),cfg.getLeft()));  
 System.***out***.print(**"reduce "**+cfg+**" "**);  
 **break**;  
 **case *ACCEPT***:  
 System.***out***.println(**"parse succeed!"**);  
 **break** loop;  
 **case *NULL***:  
 System.***out***.println(**"Unknown Error!"**);  
 **break** loop;  
 }  
 printStack();  
 }  
 }  
 } **catch** (FileNotFoundException e) {  
 System.***out***.println(**"Can't find test file."**);  
 }  
}

# 输入定义

## 上下文无关文法

* 上下文无关文法文件需要放在input文件夹中，并命名为CFG.txt
* 每行一句文法，文法的字符之间不能有空格（除非空格是文法的组成部分）
* 推导用“=”符号表示

示例输入：

S=E

E=E+T

E=T

T=T\*F

T=F

F=(E)

F=i

## SLR分析表

* 分析表文件需要放在input文件夹中，并命名为table.txt
* 第一行输入状态数量，第二行输入ACTION数量和列名，第三行输入GOTO数量和列名
* 分两段输入ACTION与GOTO的数据，并用%%%%分隔两段。
* 使用%分隔列与数据

示例输入：

12

6:i,+,\*,(,),$

3:E,T,F

%%%%

0:i%S5|(%S4

1:+%S6|$%$

2:+%R2|\*%S7|)%R2|$%R2

3:+%R4|\*%R4|)%R4|$%R4

4:i%S5|(%S4

5:+%R6|\*%R6|)%R6|$%R6

6:i%S5|(%S4

7:i%S5|(%S4

8:+%S6|)%S11

9:+%R1|\*%S7|)%R1|$%R1

10:+%R3|\*%R3|)%R3|$%R3

11:+%R5|\*%R5|)%R5|$%R5

%%%%

0:1%E|2%T|3%F

4:8%E|2%T|3%F

6:9%T|3%F

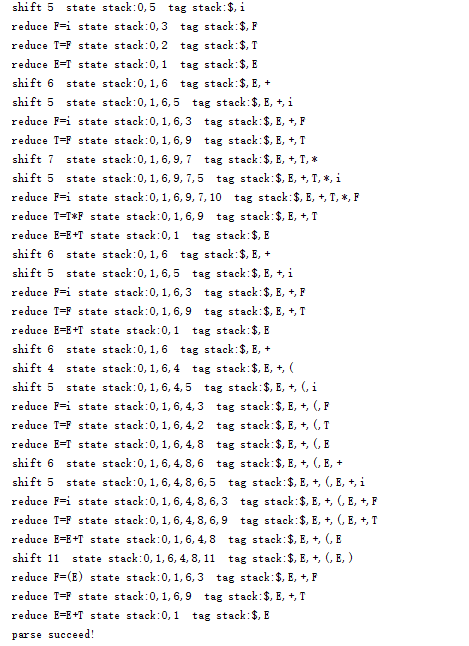
7:10%F

# 测试输入输出

使用上面定义的文法和分析表作为输入，分析以下测试输入：

i+i\*i+i+(i+i)$

可以看到每一步的分析结果：



# 总结与收获

因为使用的SLR方法，所以使用的输入示例是较为简单的语法描述，较复杂的语法描述有太多的状态。但是只要定义好文法和分析表，本程序可以分析较为复杂的语法。通过这次实验我对LR方法有了更深入的认识，体会到了分析器的思想和原理，同时也提高了自己的编程水平。