合肥工業大學

系统软件综合设计报告
编译原理分册

设计题目 30

学生姓名

学　　号

专业班级 计算机科学与技术

指导教师

完成日期 2020.8.22

报告目录

一、设计目的及设计要求

设计目的

为了进一步掌握对编译原理这门课程所学知识以及将这些知识运用在项目中。

因为当时实验的时候只完成了直接左递归的功能实现而间接左递归并未实现，因此这次通过自己的研究写了一个可以消除间接和直接左递归的程序出来

设计要求

构造一程序，实现教材 P.70 消除左递归算法。对于用

第 7 页户任意输入的文法 G，输出一个无左递归的等价文法，可显示输出，或输出到指定文件中

二、开发环境描述

windows 10 家庭中文版

处理器:Intel i7-8750H

内存:24GB

开发软件: IntelliJ IDEA 2019.1.3 x64

使用语言: Java

三、设计内容、主要算法描述

设计内容

一个可以消除所有左递归（包括直接左递归和间接左递归）的程序

主要算法描述

※题目 30

●直接左递归的检测与消除

实际上对于一个文法来说，直接左递归的检测是相对较容易的，比方说G->Gab,因为直接左递归的产生式首字符和非终结符是一致的,利用这个特点就可以实现直接左递归的检测,也就是通过比较每条规则中产生式的首字符和非终结符即可

而直接左递归的消除，可根据以下规则进行，初始产生式为

$$A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_n | \beta_1 | \beta_2 | \ldots | \beta_n$$

引进一个新的非终结符号比如说①，将左递归改写为右递归，改写后如下所示

$$A \to \beta_1 ① | \beta_2 ① | \ldots | \beta_n ①$$
$$① \to \alpha_1 ① | \alpha_2 ① | \ldots | \alpha_n ① | \varepsilon$$

规则制定完之后，就可以进行算法的实现了，因为本人实现的数据结构是一个二维数组,每一个子数组的第一项就是非终结符,而后面紧跟着的才是产生式,大体如下图所示

G → Ab|Bb|ab
A→ Aba|Ba
B→ Abd|+ab

part array	0	1	2	3
0	G	Ab	Bb	ab
1	A	Aba	Ba	
2	B	Abd	+ab	

Example: part[0][0] = "G"， part[1][1] = "Aba"

因此一旦检测到直接左递归后，就开始执行直接左递归消除函数，首先找出其中除了非终结符之外其他所有第一个字符是非终结符的项，将这些项尾部添加新非终结符后填入数组新的一行中，并且新的一行中要加入一个空字符ε。新的一行的第一个元素填入新引入的非终结符。随后将剩下的项全部在末尾添加新非终结符（因为数组是字符串形式所以可以自由修改，修改完如下图所示

G → Ab|Bb|ab
A→ Ba①
B→ Abd|+ab
①→ba①|ε

part array	0	1	2	3
0	G	Ab	Bb	ab
1	A	Ba①		
2	B	Abd	+ab	
3	①	ba①	ε	

之后只要一直顺序扫描下去就可以了，因为根据规则来判定，当前扫描到的产生式是不包含直接左递归的，因此只需要对后续的产生式进行扫描就可以了

●间接左递归的检测与消除

接左递归相对于直接左递归来说无论是检测还是消除都更加困难，这也是当

初为什么我无法在实验中实现的原因，间接左递归无法用直接扫描检测，诸如一个文法，A→Bb|b, B→Ab|ab 来说，这其中就包含间接左递归，因为将第一个产生式中的 B 替换为 Ab 后就出现了直接左递归，而单纯通过一次扫描或者比较首字符肯定是不可行的

后来通过对间接左递归的原理的理解以及上网查询了相关资料，本人自己写出了一个算法用于检测间接左递归

范例文法如下

$$A \rightarrow aB$$
$$A \rightarrow Bb$$
$$B \rightarrow Ac$$
$$B \rightarrow d$$

①第一次拿到文法的时候先对文法进行预处理，随后先进行一遍直接左递归的消除，这里因为不存在直接左递归，因此我们直接得到处理后的文法

$$A \rightarrow aB \mid Bb$$
$$B \rightarrow Ac \mid d$$

②然后将所有非终结符都标上序号，这里使用的是 HashMap

{A=1, B=2}

随后对每一个产生式都进行一个递归搜索，具体做法是找出这个产生式中所有非终结符开头的项，随后依次从这些非终结符所对应的产生式中找出新的对应的非终结符，以此类推，将先前搜索到的所有非终结符都加入到一个 Set 中（该数据结构可以有效避免元素重复，并且本次功能中并不要求顺序），扫描完成后得到如下 Set 数组

[A, B]

[A, B]

也就是说 A,B 对应的两个产生式都能同时到达 A 和 B，这是一个很明显的间接左递归

③待所有递归搜索都结束后（因为当前文法不包含），遍历该 Set 数组，只要有一个 Set 中出现了和对应标号的非终结符相同的非终结符，就代表该非终结符对应的产生式存在间接左递归，本人将这些都添加到一个函数中，并将遍历得到的第一个有间接左递归的标号保存，因为这边两个都包含左递归，自然就保存了标号为 1 的产生式

④随后从末尾往前直到第一个有间接左递归的标号进行扫描，每次将第一个非终结符进行存储，作为主终结符 MAIN，并且扫描该产生式中的非终结符（客终结符，为 COM），并且将二者的顺序进行对比，一旦发现前者小于后者（MAIN<COM），则表示找到了直接左递归产生的根源，此时将该产生式标号和非终结符对应的序号进行保存

我们可以看到这个产生式 $B \to Ac \mid d$，其中主终结符 B（标号为 2）大于客终结符 A（标号为 1），因此要对这个式子进行处理

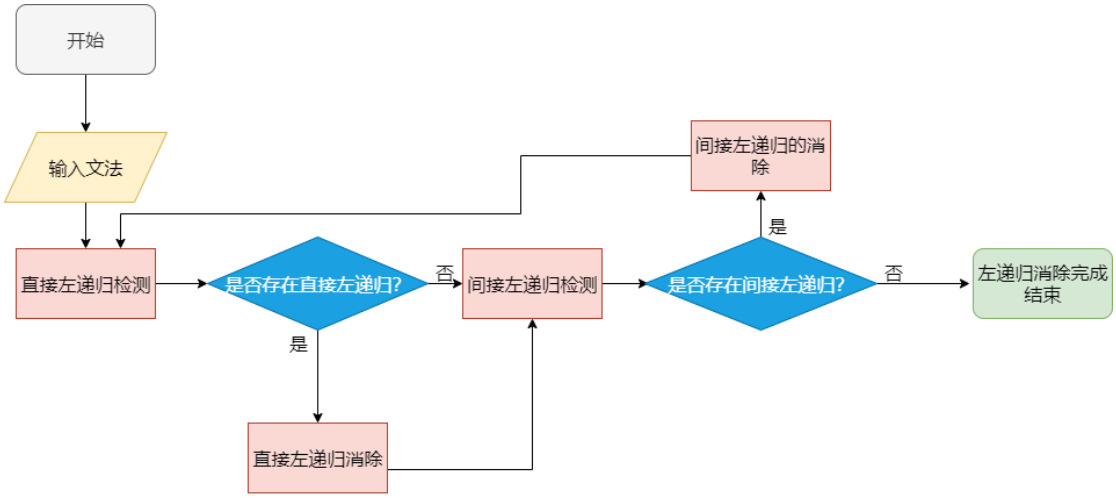
⑤按照间接左递归的消除规则进行消除即可，同样是引入一个新的非终结符，随后对问题产生式 $A \to aB \mid Bb$ 进行扫描，将其改写为

$$A \to aB① \mid db①$$
$$① \to cb① \mid \varepsilon$$

⑥用相同的方法消除直接左递归后进行扫描，扫描出来有问题继续同样操作，

直到不出现问题为止完成递归的消除

需要注意的是这里首先对靠前的产生式进行修改，这样可以确保扫描所对应的编号前面都是无需处理的产生式，从这个方法中可以看出，直接子递归的操作作为一个子函数已经融入间接左递归中了，因此只要对整个文法进行一次间接左递归的操作就会自动完成对直接左递归的检测与消除，算法流程图如下



题目30： 左递归的消除算法流程图

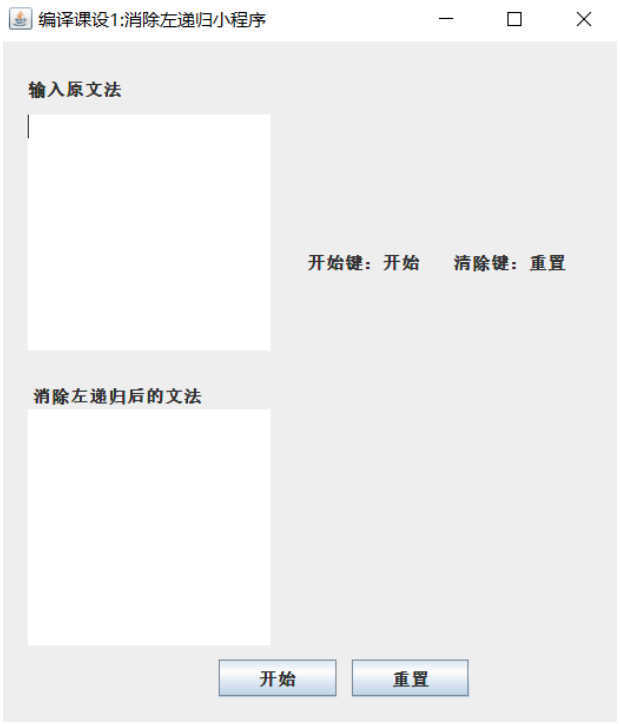
四、设计的输入和输出形式

设计输入：原始文法（字符串），在 GUI 中专门提供一个输入框可以输入完整的文法，单个产生式的格式为 A->Bab，通过->连接，非终结符限定为 26 个大写字母

设计输出：消除左递归后的文法（字符串），在 GUI 中专门提供一个框可以进行输出，输出格式与输入格式相同，因为消除左递归而产生的新终结符用圆圈+数字的特殊字符比如说①，②表示

五、程序运行（测试、模拟）的结果（屏幕拷贝、生成结果的打印输出）

①程序 UI 界面如下



②首先针对包含直接左递归的文法进行左递归消除操作如下

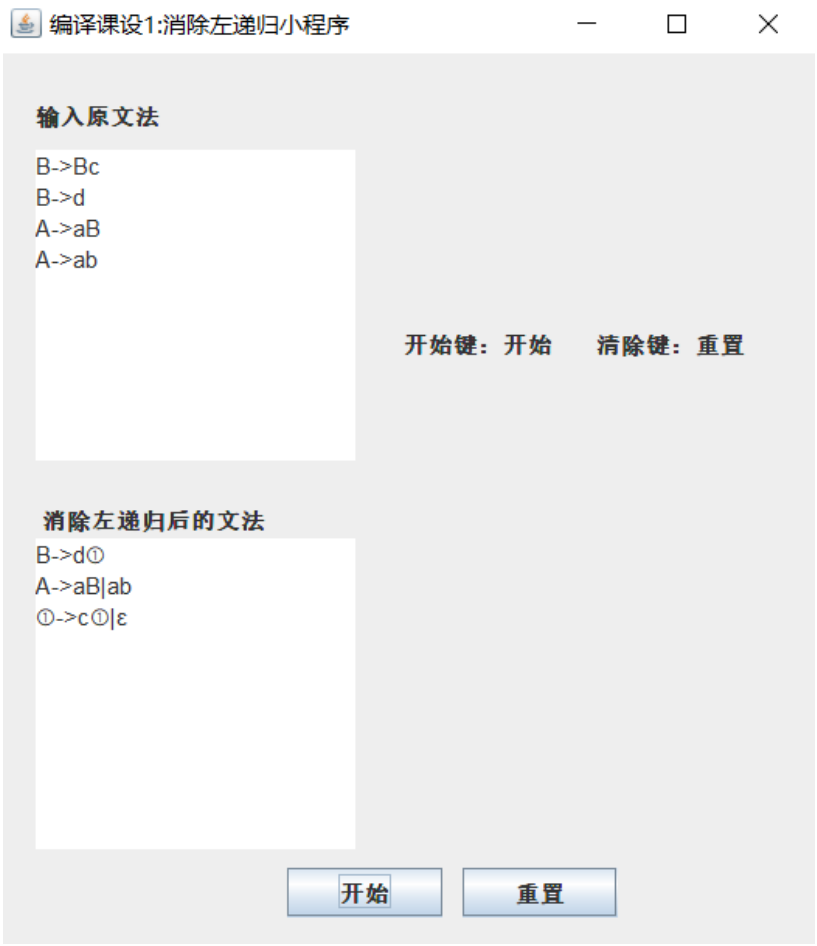
输入文法:

B- > Bc
B- > d
A- > aB
A- > ab

输出结果:

B- > d①
A- > aB | ab
①- > c① | ε

成功消除该文法的直接左递归，运行界面如下



③其次针对包含间接左递归的文法进行左递归消除操作如下

输入文法:

S- > Qc | c
Q- > Rb | b
R- > Sa | a

输出结果:

S- > abc① | bc① | c①
Q- > Sab | ab | b
R- > Sa | a
①- > abc① | ε

成功消除该文法的间接左递归，运行界面如下

③最后为了验证算法的严谨性，本人特地找了一个不包含任何左递归的文法来处理，看是否会执行操作

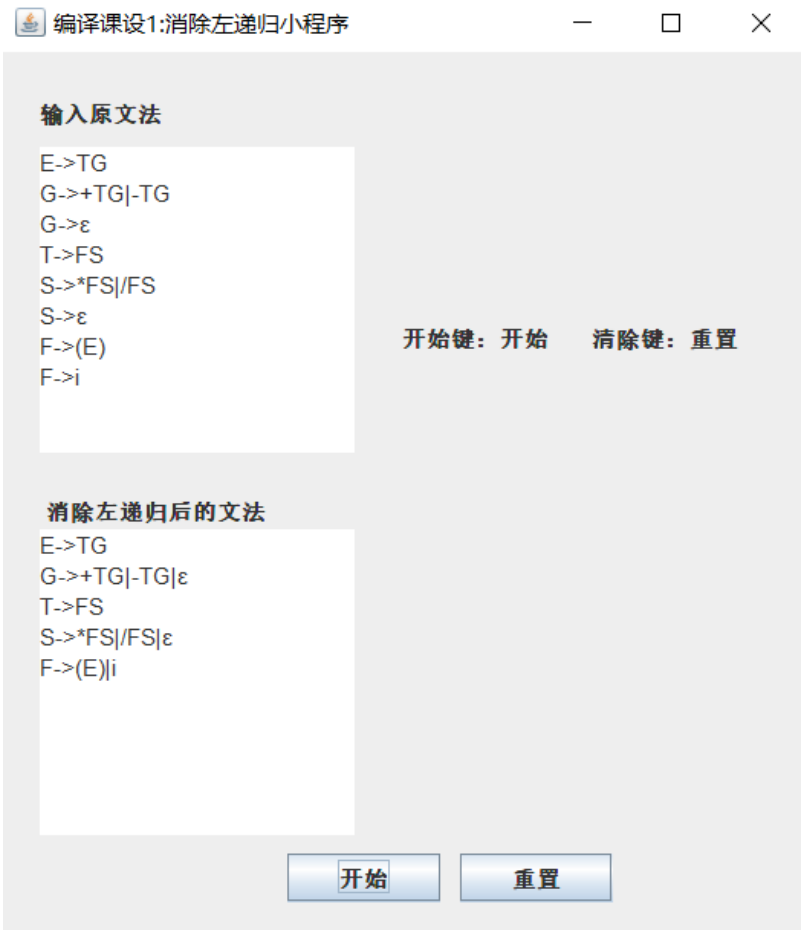
输入文法:

E- > TG
G- > +TG | -TG
G- > ε
T- > FS
S- > *FS | /FS
S- > ε
F- > (E)
F- > i

输出结果:

E- > TG
G- > +TG | -TG | ε
T- > FS
S- > *FS | /FS | ε
F- > (E) | i

程序只是将该文法中相同非终结符产生的产生式归纳到了一起，并没有产生新的非终结符之类的操作，验证了本消除左递归算法的正确性，能有效的检测并消除直接，间接左递归，并且还不对不含左递归的文法有影响，运行界面如下



六、总结（体会）

本次题目设计进一步加深了本人对于消除左递归思路的理解，并且在独自思考构想出消除间接左递归的检测和消除算法中也进一步强化了对编译中语法过程分析的掌握和 java 程序的编写能力

实际上除了参考了教材上的基本思路之外，本人还参考了著名的编译原理巨作龙书，上面的很多内容和思路在我编写代码的时候都起到了十分重要的作用，

也让我对编译原理的很多细节有了初步的掌握

七、源程序清单（部分核心代码）作为报告的附件。

路径下的 src 文件夹包含四个源代码文件

GUI.java：项目的 GUI 界面代码实现

LeftDelete.java 项目的功能函数定于与实现

Main.java 项目主函数

test.java 用于进行一些测试