

**课程设计报告**

( 2017--2018年度第一学期)

名 称： 编译技术课程设计

题 目： LL(1)文法和预测分析方法

院 系： 控制与计算机工程学院

班 级： 信安1502班

学 号： 1151290210

学生姓名： 林端

指导教师： 夏宏

设计周数： 一周

成 绩：

日期：2018年 1 月15日

# 1 课程设计的目的和要求

## 1.1 课程设计的目的

本次设计的时间为1周，目的是通过使用高级语言实现部分算法加强对编译技术和理论的理解。设计的题目要求具有一定的规模，应涵盖本课程内容和实际应用相关的主要技术。

## 1.2 课程设计的要求

要求：

1. 文法使用产生式来定义；
2. 分别求出文法中每一个非终结符的FIRST 集、FOLLOW集和SELECT集；
3. 画出预测分析表；
4. 判定给定的文法是否是LL(1)文法；
5. 给定符号串判定是否是文法中的句子，分析过程用分析表格的方式打印出来。

# 2 系统描述

本次实验使用了基于Java语言的集成开发环境Eclipse进行编写，利用自顶向下的语法分析方法对一个文法进行判断，输入为多条产生式组成的文法G[S]，计算文法中各非终结符的FIRST集、FOLLOW集，根据FIRST集、FOLLOW集求出各产生式的SELECT集，分析的结果显示在界面中。

其次，根据上述所求FIRST集、FOLLOW集和SELECT集来构造预测分析表，并且能够对用户输入的一个字符串进行判断，判断其是否为该文法中的句子，将分析过程打印到界面。

## 2.1 LL（1）文法的描述

1、一个上下文无关文法是LL(1)文法的充分必要条件是：对每个非终结符A的两个不同产生式，A→α, A→β,满足SELECT(A→α)∩SELECT(A→β)= Φ（ 其中α，β不同时能推导出ε）。

2、LL(1)文法的含义：

第一个L即从左到右扫描输入串

第二个L即生成的是最左推导

1：向右看一个输入符号便可决定选择哪个产生式。

3、LL(1)文法的判别：当我们需选用自顶向下分析技术时，首先必须判别所给文法是否是LL(1)文法。因而我们对任给文法需计算FIRST、FOLLOW、SELECT集，进而判别文法是否为LL(1)文法。

## 2.2 FIRST集、FOLLOW集、SELECT集描述及算法

一、FIRST集

1、定义：

设G=(VT，VN，S，P)是上下文无关文法 ，FIRST(α)={a|α→aβ,a∈VT，α,β∈V\*}

特别的，若α→ε,则规定ε∈FIRST(α)．

2、根据定义求解FIRST集（对每一文法符号X∈VN 计算FIRST(X)）：

(1). 若X∈VT，则FIRST(X)＝{X}。

(2). 若X∈VN，且有产生式X→a…，a∈VT， 则 a∈FIRST(X)X→ε,则ε∈FIRST(X)。

(3). X→Y…是一个产生式且Y ∈VN 则把FIRST(Y)中的所有非空符号串ε元素都加入到FIRST(X)中。

(4).若X∈VN；Y1，Y2，…，Yi∈VN，且有产生式X→Y1 Y2 … Yn；当Y1 Y2 … Yn-1→ε时，则FIRST(Y1)、FIRST(Y2)、…、FIRST(Yn-1)的所有非空元素和FIRST(Yn) 包含在FIRST(X)中。

(5).当(4)中所有Yi→ε,(i=1,2,…n)，则

FIRST(X)=(FIRST(Y1)－{ε}）∪（FIRST(Y2)－ {ε}∪…∪（FIRST(Yn) －{ε}）∪{ε}

反复使用上述(2)～(5)步直到每个符号的FIRST集合不再增大为止。

二、FOLLOW集

1、定义：

设G=(VT，VN，S，P)是上下文无关文法，A∈VN，S是开始符号

需要注意的是，FOLLOW(A)集是针对非终结符A的，集合中的元素是终结符，是A的全部后跟符号的集合，当A是文法G的开始符(识别符)时，把‘#也加入其中’

2、根据定义求解FOLLOW集（对每一文法符号S∈VN 计算FOLLOW(S)）：

(1). 设S为文法中开始符号，把{#}加入FOLLOW(S)中(这里“#”为句子括号)。

(2). 若A→αBβ是一个产生式，则把FIRST(β)的非空元素加入FOLLOW(B)中。如果β→ε则把FOLLOW(A)也加入FOLLOW(B)中。

(3).反复使用(b)直到每个非终结符的FOLLOW集不再增大为止。

三、SELECT集

1、定义：

给定上下文无关文法的产生式A→α, A∈VN,α∈V\*：

若α不能推导出ε,则SELECT(A→α)=FIRST(α)

如果α能推导出ε则：SELECT(A→α)=（FIRST(α) –{ε}）∪FOLLOW(A)

需要注意的是，SELECT集是针对产生式而言的。

## 2.3预测分析表的构造

LL(1)分析表的作用是对当前非终极符和输入符号确定应该选择用哪个产生式进行推导。它的行对应文法的非终极符，列对应终极符，表中的值有两种：一是产生式的右部的字符串，一是null。若用M表示LL(1)分析表，则M可表示如下：

M: VN×VT🡪P∪{Error}

M(A, t) = A🡪α，当t∈select(A🡪α) ，否则

M(A, t) = Error

其中P表示所有产生式的集合。

## 2.4语法分析程序构造

LL(1)分析中X为符号栈栈顶元素，a为输入流当前字符，E为给定测试数据的开始符号，#为句子括号即输入串的括号。分析表用一个二位数组M表示，数组元素M[A，a]中的下标A表示非终结符，a为终结符或句子括号‘#’，二维数组中存放的是一条关于A 的产生式，表明当非终结符A向下推导时，面临输入符a时，所采用的候选产生式，当元素内容无产生式时，则表明用A 的左部向下推导时出现了不该出现的符号，因此元素内容转向出错处理的信息。

LL(1)分析过程主要包括以下四个动作：

替换：当X∈VN时选相应产生式右部β去替换X。此时X出栈，β逆序入栈。

匹配：当X∈VT时它与a进行匹配，其结果可能成功，也可能失败，如果成功则符号栈中将X退栈并将输入流指针向前移动一位，否则报错。

接受：当格局为（#，空#）时报告分析成功。

报错：出错后，停止分析。并给出相应的错误提示信息。

# 3 概要设计

## 3.1 概要设计

首先运行源程序，在界面相应文本区依次输入文法G[S]的产生式，显示到文本区域t1 (JTextArea类型) 中，对产生式无顺序要求，但不允许输入非法的产生式，例如空。输入完成后，进入文法分析阶段。

文法分析程序的功能是对文本区t1的所有字符串从头到尾逐一扫描，然后将读到的字符串（如S→AB）拆分成每个字符进行分析，“→”左侧为产生式的左部（非终结符Vn），右侧为产生式的右部，一般的产生式为二型文法，即左侧只允许输入一个大写字母，右侧输入的字符串中的非大写字母（小写字母a~z，各种符号）为终结符Vt。

LL（1）文法分析的过程如下：

(1) 计算FIRST集。

(2) 计算FOLLOW集。

(3) 计算SELECT集。

(4) 判定是否为LL(1)文法。

(5) 画出预测分析表。

在求解FIRST集时定义了如下数据结构：

String Vn[] = null; //所有非终结符集合

int Vnnum; //非终结符个数

int firstComplete[] = null; // 存储已判断过 first 的数据

char first[][] = null; // 存储最后 first 结果

first = new char[Vnnum][100];

String s[] = t1.getText().split("\n"); //依次读取文本框中的每条产生式

char yn\_null[] = null; // 存储能否推出空

String firstVn[] = null; // 进行first判断的Vn集合

在求解FOLLOW集时定义了如下数据结构：

int followComplete[] = null; // 存储已判断过 follow 的数据

char follow[][] = null; // 存储最后 follow 结果

follow = new char[Vnnum][100];

String followVn[] = null; // 存储已判断过 follow 的数据

在求解SELECT集时定义了如下的数据结构：

char select[][] = null; // 存储最后 select 结果

select = new char[P.size()][100];

//P.size()是产生式的数目，每个产生式都有一个select集

char Vt[] = new char[100]; //终结符集

在构造预测分析表示定义了如下数据结构：

Vector<String> P = null; //字符串型向量，用于填预测分析表

int LL = 0; // 标记是否为 LL（1）

String Vt\_table[] = null; // 储存 Vt

Object Vn\_data[][] = null; // 存储表达式数据

Vt\_table = new String[temp + 1];// 根据 select 和表达式生成预测分析表，temp为终结符个数，表头第一行一列做空

Vn\_data = new String[Vnnum][temp + 1]; //头为所有Vn

dtm = new DefaultTableModel(Vn\_data, Vt\_table);// 显示预测分析表

在对输入字符串进行分析时定义了如下数据结构：

char Sentence[] = new char[100]; // 剩余输入串

char Analyze[] = new char[100]; // 分析栈

int n\_Sentence; // 剩余输入串的字符下标

int n\_Analyze; // 分析栈的下标

int step; // 记录步骤

char n[] = new char[65]; // 存储要显示的一整行内容

## 3.2 系统用例图

添加产生式

构造预测分析表

LL（1）文法分析器

句子判断

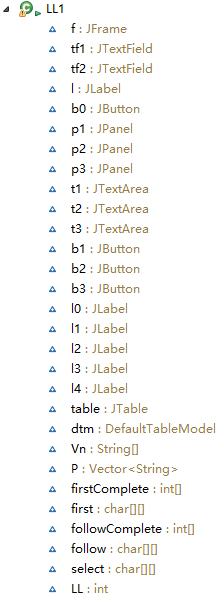
判断文法

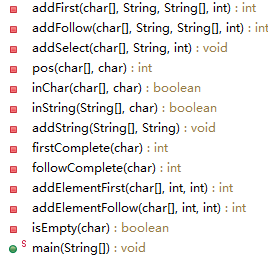
## 3.3 开发环境

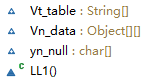
Windows 8系统基于Java语言的集成开发环境Eclipse。

# 4 详细设计

## 4.1 系统的类图









## 4.2 主要算法的流程图

（一）总体思路分析

给定一个正规文法G[S]，在LL(1)文法分析中，必须先计算出FIRST集和FOLLOW集，根据FIRST集和FOLLOW集来求SELECT集，通过判断同一个非终结符的SELECT集中是否有交集，或者Φ，来决定是否为LL(1)文法，最后构造预测分析表。求出预测分析表之后，再输入一个字符串，依据LL(1)分析程序输出字符串的分析过程，打印在文本区中。通过文本区显示该输入字符串是否为该G[S]的句子。



图4-1 功能模块分解图

（二）主程序流程图



图4-2 主程序流程图

（三）核心算法流程图

1.计算非终结符的First集的算法及流程：

（1）若X∈VT，则First（X）={X}。

（2）若X∈VN，且有产生式X→a……，则把a加入到First (X)中；若X→ε也是一条产生式，则把ε也加到First (X)中。

（3）若X→Y……是一个产生式且Y∈VN，则把First (Y)中的所有非ε-元素都加到First (X)中；若X→Y1Y2…Yk是一个产生式，Y1，…，Yi-1都是非终结符，而且，对于任何j，1≤j≤i-1，First (Yj)都含有ε（即Y1…Yi-1\* ε），则把First (Yj)中的所有非ε-元素都加到First (X)中；特别是，若所有的First (Yj)均含有ε，j=1,2,…，k，则把ε加到First (X)中。

连续使用上面的规则，直至每个集合First不再增大为止。



图4-3 计算FIRST集流程图

2.计算非终结符的Follow集：

Follow集合的具体构造算法如下：

（1）对于文法的开始符号S，置#于Follow(S)中；

（2）若A→αBβ是一个产生式，则把First(β)|{ε}加至Follow(B)中；

（3）若A→αB是一个产生式，或A→αBβ是一个产生式而β ε（即ε∈First(β)），则把Follow(A)加至Follow(B)中。

连续使用上面的规则，直至每个集合Follow不再增大为止。



图4-4 计算FOLLOW集流程图

3.预测分析控制程序的算法流程



图4-5 预测分析程序流程图

4.3 数据分析与定义

|  |  |
| --- | --- |
| **主类** | public class LL1 extends JFrame implements ActionListener {} |
| **界面变量** | JFrame f; // 总界面  JTextField tf1; // 产生式左部  JTextField tf2; // 产生式右部  JLabel l; // "→"  JButton b0; //添加产生式  JPanel p1, p2, p3; // 三个面板  JTextArea t1, t2, t3; //文法显示  //待输入句子  //文法分析及判断待输入句子是否为文法句型  JButton b1, b2, b3; // 判断文法 // 句子判断 //清空全部  JLabel l0, l1, l2, l3, l4; // 文字标签  JTable table; // 预测分析表  JScrollPane jp1 = new JScrollPane(t1); // 滚动  JScrollPane jp2 = new JScrollPane(t2);  JScrollPane jp3 = new JScrollPane(t3);  DefaultTableModel dtm; // 表格模型 |
| **其他变量** | int firstComplete[] = null; // 存储已判断过 first 的数据  char first[][] = null; // 存储最后 first 的结果  int followComplete[] = null; // 存储已判断过 follow 的数据  char follow[][] = null; // 存储最后 follow 的结果  char select[][] = null; // 存储最后 select 的结果  int LL = 0; // 标记是否为 LL（1）  String Vt\_table[] = null; // 储存分析表中所有终结符  Object Vn\_data[][] = null; // 存储表达式数据  char yn\_null[] = null; // 存储能否推出空  Vector<String> P = null; //字符串型向量，用于填预测分析表 |
| **子函数** | private int addFirst(char a[], String b, String firstVn[], int flag)  //添加first集  private int addFollow(char a[], String b, String followVn[], int flag)  //添加follow集  private void addSelect(char a[], String b, int flag)  //添加select集  private int pos(char Vt[], char x)  // x在终结符集的位置  private boolean inChar(char a[], char x)  //判断 x 是否在 字符串 a 中，在返回 false，不在返回 true  private boolean inString(String a[], char x)  // 判断字符 x 是否在字符串 a 中，在返回 false，不在返回 true  private void addString(String firstVn[], String x)  // 把x加入字符串组 firstVn[]的最后  private int firstComplete(char x)  // 判断x 是否已完成 first  private int followComplete(char x)  //判断 x是否已完成 follow  private int addElementFirst(char a[], int pos, int flag)  // 把相应终结符添加到first[]中, pos为该下标  private int addElementFollow(char a[], int pos, int flag)  // 把相应终结符添加到 follow[]中, pos为该下标  private boolean isEmpty(char x)  //判断该Vn的FIRST集合是否有空  public static void main(String[] args)  // 主函数执行 |

表4-1 数据分析与定义表

## 4.4 系统界面设计



图4-6 系统界面图

# 5 测试方法和测试结果

## 5.1 测试用例1

测试目的：是否能够正确生成FIRST、FOLLOW、SELECT集和预测分析表，是否能利用预测分析程序对输入的字符串进行分析，并正确生成栈的变化过程。

输入的文法如文法5-1所示。

（在输入文法时，用#代替我们熟知的空ε）

S→E

E→TA

A→+TA

A→#

T→FB

B→\*FB

B→#

F→(E)

F→i

文法5-1 测试文法1

计算的输出结果如图5-1所示。





图5-1 测试文法1的FIRST,FOLLOW和SELECT集

该文法的预测分析表如图5-2所示。



图5-2 预测分析表

输入字符串i\*i+i，输入串分析如图5-3所示。

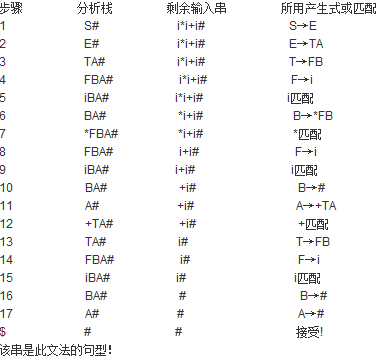


图5-3 输入串分析表

## 5.1 测试用例2

测试目的：对于不属于LL（1）的文法，是否能检测其文法的属性，是否正确生成FIRST、FOLLOW、SELECT集，并根据SELECT集含有交集得出不属于LL（1）文法的结论。

输入如文法5-2所示。该文法为书中示例。

（在输入文法时，用#代替我们熟知的空ε）

S→AB

S→bC

A→#

A→b

B→#

B→aD

C→AD

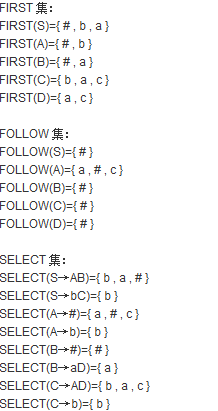
C→b

D→aS

D→c

文法5-2 测试文法2

计算的输出结果如图5-4所示。结论不属于LL（1）的文法，正确！



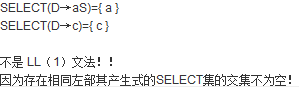


图5-4 测试文法2的FIRST,FOLLOW和SELECT集

# 结论和展望

## 结论

按照编译原理课程设计的题目要求，本次课程设计圆满的实现了题目的所有要求，如输入文法、计算FIRST集、FOLLOW集和SELECT集来构造预测分析表，并且能够对用户输入的一个字符串进行判断，判断其是否为该文法中的句子，处理分析栈，显示分析过程。也很好地按照课程设计的要求，对系统进行了多组数据测试，对不同类型的文法都进行了分析，测试结果正确，符合题目要求。

从算法的设计到最终的完成，难度最大的也是数据结构的建立和不同数据类型的处理，FIRST集、FOLLOW集和SELECT集的操作都需要创建许多的一维数组、二维数组的变量，变量多且复杂，在调试的时候有不小的难度。

其次，是众多子函数的设计和调用，递归嵌套可以说本次课程设计的核心就在于此。对非终结符和终结符的判断、添加和修改都需要在按键响应中调用这些子程序。起初的设计，考虑用简单的循环结构和选择结构。但是在编程的过程中发现许多功能不能单纯地依靠循环和判断，由于循环嵌套的不便和繁琐，才利用一个个子函数来代替原本大量的循环嵌套，也降低了分析的难度。递归的使用也使程序调试时更简便。

总体来说，系统运行正常，算法逻辑正确，递归函数的算法符合题目的要求。

## 展望

在为期一周的编译原理课程设计中，值得欣慰的是自己最终实现了LL（1）文法分析工具，从无到有的过程总是令人激动而且自豪的。在一步步编写程序的过程中，总是会遇到很多麻烦和困难，例如预期的输出结果和现实总有偏差，总是需要不断地检查漏洞和改进程序，使最后的成果是完善的、人性化的，也必须是正确的。编译原理课讲的更多是书本中的知识，和习题等，原本以为对编译原理的掌握地很好了，但是真正要将抽象的算法和流程落实在程序中，并不是一件容易的事。所谓实践出真知，唯有真正体验到编写算法的复杂和判断情况的限制条件时，真正面对一个个字符串和字符时，才会加深对编译的理解，可以说理解能力上升的一个层次。

本次课设让我对LL（1）文法有了十分清晰的认识，对递归的思想有了更深的认识，同时也温习了书中的知识点，对原理和过程也得到了进一步巩固。其次，对Java编程的熟练度也得到了进一步提高，以及在报告的设计和程序的调试中都受益匪浅。学以致用，融会贯通。当然，也存在一些小小的不足，比如代码还可以优化或简化，数组的设计还应更合理，界面还能设计地更加美观等。

## 学习编译技术课程的体会和对本门课程的评价

对我而言，编译技术是一门容易接受且很好掌握的课程，学习的过程中几乎没有难度，课后对老师所讲内容也常常举一反三，对待习题和练习也比较认真。我在期末得到了优异的成绩，也要十分感谢夏宏老师的教导。

通过这一学期的学习，我发现了编译原理是一门理论性很强的学科，就以本次涉及的LL（1）文法来说，几乎都是对具体问题的高度抽象。因此，学习时必然需要更多的时间来理解和掌握。我认为，编译原理课不仅仅在于其本身的理论知识，还在于为我们解决实际问题提供了更科学更严谨的思维方式和解决方法。不管是LL(1)还是LR（0）文法，我觉得理解算法的过程才是主要的，死记硬背是事倍功半的，认识不深，就无法在编程中写出完善而且鲁棒性强的代码。编译原理课程的结束并不意味着学习的结束，而是一个新的开始，从编译原理认识软件工程、认识开发设计，认识信息安全问题，融会贯通。作为一名计算机信息安全专业的学生来说，这些经历和知识都是十分宝贵的。

总体来说，学习的难度并不大，掌握地也不错。在学习编译的过程中收获了很多，也反思改进了不少。希望可以在今后的学习中知难而上，在编译学习中曾出现的问题要尽量避免，学会整体考虑整体规划，冷静面对，戒骄戒躁。

最后，衷心感谢夏宏老师一学期的辛苦教学。

# 参 考 文 献

[1] 张素琴. 编译原理(2版). 北京：清华大学出版社[M]，2015，75-96.

[2] 王素琴. Java语言程序设计实用教程. 北京：中国电力出版社[M]，2017，114-183.