

**课程设计报告**

( 2022--2023年度第一学期)

名 称： 编译技术课程设计

题 目： LR(0)分析法研究

院 系： 控制与计算机工程

班 级： 计算2001

学 号： 120201080329

学生姓名： 王心然

指导教师： 齐林海

设计周数： 一周

成 绩：

日期：2022年 12 月31日

目录

[1 课程设计目的和要求 4](#_Toc128998176)

[1.1 课程设计目的 4](#_Toc128998177)

[1.2 课程设计的要求 4](#_Toc128998178)

[2 系统分析 4](#_Toc128998179)

[2.1 LR(k)分析法 4](#_Toc128998180)

[2.2 LR(0)分析法的实现 5](#_Toc128998181)

[3 概要设计 5](#_Toc128998182)

[3.1 概要设计 5](#_Toc128998183)

[3.2 系统用例图 6](#_Toc128998184)

[3.3 系统用例 6](#_Toc128998185)

[3.4 开发环境 7](#_Toc128998186)

[4 详细设计 8](#_Toc128998187)

[4.1 系统的类图 8](#_Toc128998188)

[4.2 主要算法的流程图 9](#_Toc128998189)

[4.3 数据分析与定义 10](#_Toc128998190)

[4.4 系统界面设计 11](#_Toc128998191)

[5 测试方法与测试结果 11](#_Toc128998192)

[5.1 测试用例1 12](#_Toc128998193)

[5.2 测试用例2 13](#_Toc128998194)

[5.3 测试用例3 15](#_Toc128998195)

[6 结论和展望 15](#_Toc128998196)

[6.1 结论 15](#_Toc128998197)

[6.2 不足之处 16](#_Toc128998198)

[6.3 收获 16](#_Toc128998199)

[6.4 学习体会和教学评价 16](#_Toc128998200)

[7 参考文献 17](#_Toc128998201)

# 课程设计目的和要求

## 课程设计目的

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。

## 课程设计的要求

1. 文法使用产生式来定义。
   1. 用大写字母和小写字母分别表示非终结符和终结符、产生式使用->。
   2. 文法中的空字符串统一使用@表示；
2. 给出项目集规范族（即所有的状态）。
3. 给出状态间的关系。
4. 给出LR(0)分析表。
5. 给定符号串判定它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。

# 系统分析

本次实验使用Windows 10的PyCharm软件与VSCode软件，利用LR(0)分析法实现对LR(0)文法的分析，并对输入符号串是否属于文法的句子进行判断，分析过程将以图形化界面的形式展示。

程序基于Python语言实现，通过PyQt实现界面的开发和设计，产生式的输入通过界面对产生式所在文档进行选择，所需分析的句子同理，分析过程和结果的展示图片由Graphviz和Matplotlib实现，并由PyQt进行展示。

## LR(k)分析法

LR分析法是目前编译程序的语法分析中最常用而且有效的自下而上的分析技术，它能适用于绝大多数上下文无关文法语言分析。

所谓LR(k)分析，是指从左至右扫描输入符号串并进行自下而上的语法分析，且在分析的每一步，只须根据分析栈当前已移进和规约出的全部文法符号，再向前查看k个输入符号，就能确定适合于文法规则的句柄是否已在分析栈顶形成，从而可以立即确定当前的分析动作。

与算符优先文法相比，LR分析法对文法的限制较少，使用起来更加灵活。

常见的LR分析法有四种：LR(0)、SLR(1)、LR(1)、LALR(1)。本次课设介绍和展示的是LR(0)分析法。

## LR(0)分析法的实现

1. 改写为拓广文法。
2. 列出所有项目。
3. 写出所有项目集规范族。
4. 作出LR(0)分析表。
5. 对一些指定的输入串进行分析。

# 概要设计

## 概要设计

首先，创建一个txt文档，并在文档中输入需要进行的LR(0)文法产生式。然后，通过图形化界面选择该txt文件的路径，并利用open函数打开该文件，对该文件中的文法产生式进行分析，并将分析所生成的DFA图用Graphviz生成图片，将分析所生成的ACTION—GOTO表用Matplotlib生成图片，并将二者用图形化界面展示出来。之后，创建一个txt文档，并在文档中输入需要进行分析的字符串。最后，通过图形化界面选择该txt文件的路径，并利用open函数打开该文件，对该文件中的字符串进行LR(0)分析。将分析过程以表格的形式展示出来，并将表格的图片用图形化界面展示出来。

## 系统用例图

系统用例图如[图3-1](#系统用例图)所示。

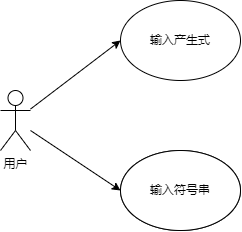


图3-1 系统用例图

## 系统用例

输入产生式用例如表3-1所示。

表3-1 输入产生式用例

|  |  |
| --- | --- |
| **用例名称** | 输入文法的产生式 |
| **简述** | 通过导入txt文件输入文法产生式 |
| **角色** | 用户 |
| **前置条件** | 打开初始界面 |
| **基本流** | 1. 用户在初始界面中点击“导入文法产生式” 2. 选择对应的txt文件，点击“确定” 3. 系统循环读取txt文件每一行 4. 系统将导入的产生式进行LR(0)分析，并将分析结果显示在“DFA”和“ACTION GOTO TABLE”中   【用例结束】 |
| **后置条件** | 导入成功 |
| **特殊需求** | 只能导入txt文件 |

输入符号串用例如表3-1所示。

表3-1 输入产生式用例

|  |  |
| --- | --- |
| **用例名称** | 输入所需判断的符号串 |
| **简述** | 通过导入txt文件输入符号串 |
| **角色** | 用户 |
| **前置条件** | 构建LR(0)分析表 |
| **基本流** | 1. 用户在初始界面中点击“导入符号串” 2. 选择对应的txt文件，点击“确定” 3. 系统读取txt文件中的符号串 4. 系统将导入的符号串进行分析，并将分析结果显示在“ANALYSIS TABLE”中   【用例结束】 |
| **后置条件** | 导入成功 |
| **特殊需求** | 只能导入txt文件 |

## 开发环境

软件开发环境为Pycharm和VSCode，如[图3-2](#开发环境1)、[图3-3](#开发环境2)所示。

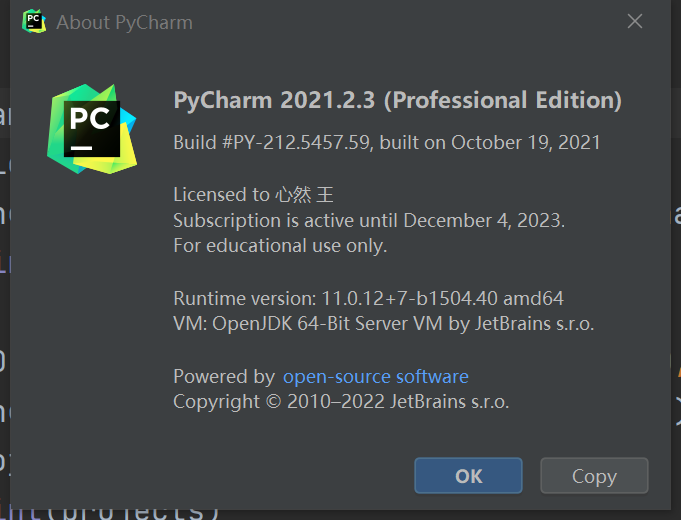


图3-2 开发环境1



图3-2 开发环境2

# 详细设计

## 系统的类图

系统的类图如图4-1所示。

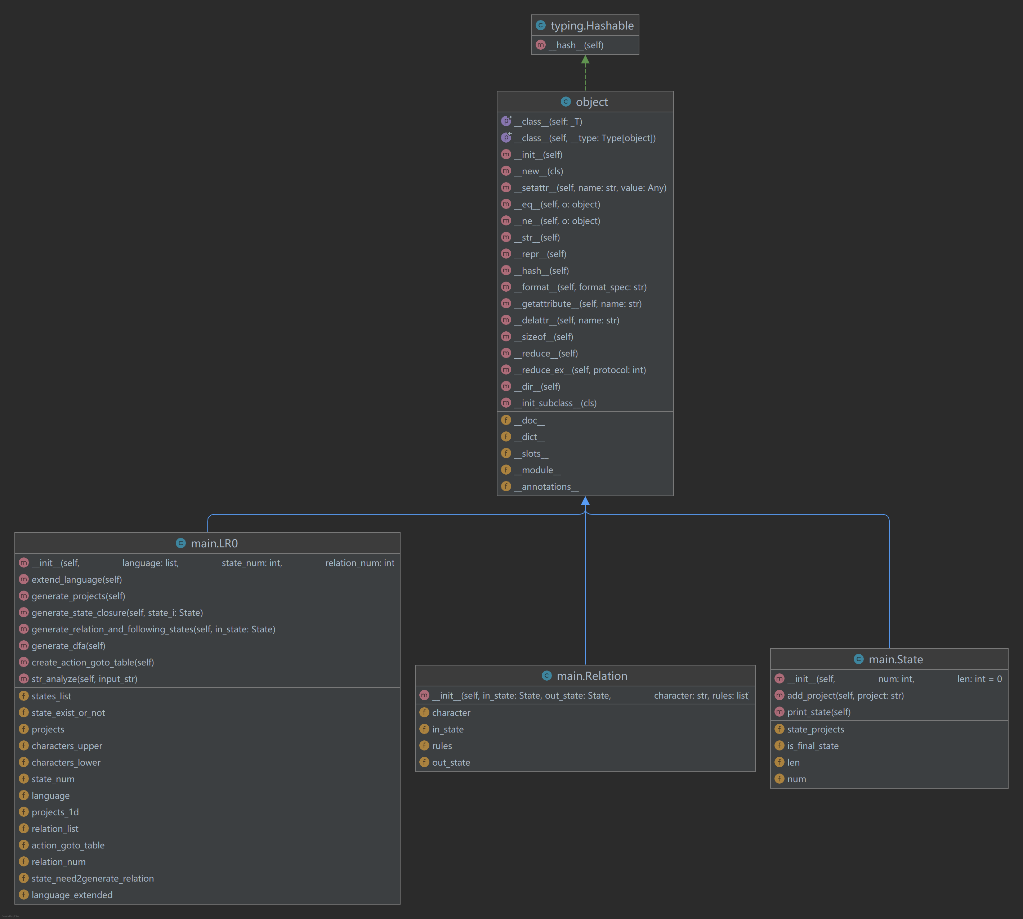
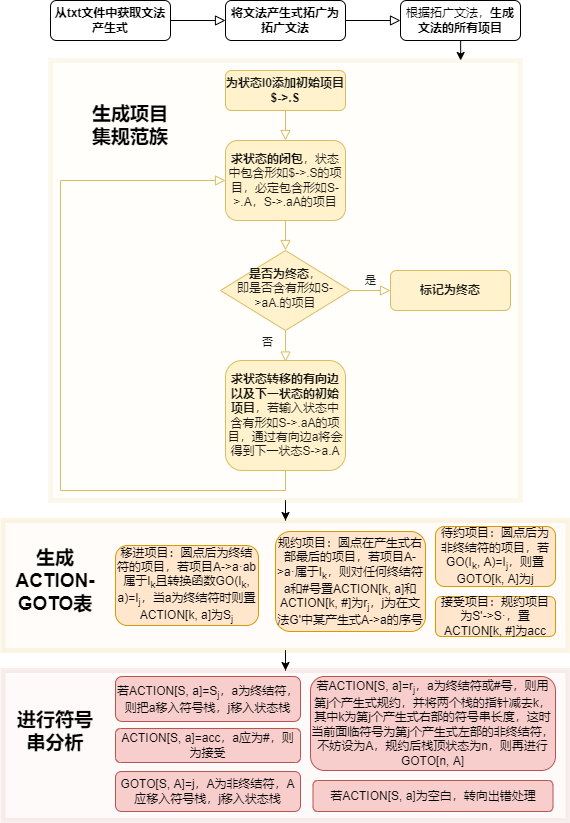


图4-1 系统的类图

## 主要算法的流程图

主要算法的流程图如图4-2所示。

图4-2 主要算法流程图

## 数据分析与定义

表4-3-1 State数据

|  |  |  |
| --- | --- | --- |
| State | | |
| 数据名称 | 数据类型 | 数据含义 |
| num | int | 状态标号 |
| len | int | 状态长度 |
| state\_projects | list | 状态所含的项目集 |
| is\_final\_state | bool | 是否为终态 |

表4-3-2 Relation数据

|  |  |  |
| --- | --- | --- |
| Relation | | |
| 数据名称 | 数据类型 | 数据含义 |
| in\_state | State | 输入状态 |
| out\_state | State | 输出状态 |
| character | str | 转换的字符 |
| rules | str | 转换的产生式 |

表4-3-3 LR0数据

|  |  |  |
| --- | --- | --- |
| LR0 | | |
| 数据名称 | 数据类型 | 数据含义 |
| language | str | 文法产生式，由txt文件转化而来 |
| state\_num | int | 状态数量 |
| relation\_num | int | 转化边的数量 |
| language\_extended | list | 拓广文法 |
| projects | list | 由拓广文法生成的项目 |
| projects\_1d | list | 将projects展为一维列表 |
| states\_list | list | 状态列表 |
| relation\_list | list | 转化边列表 |
| state\_exist\_or\_not | dict | 用于识别每个状态是否存在，元素格式：状态编号：项目集 |
| state\_need2generate\_relation | list | 存储“已经求完闭包但是没有求后续状态的状态”的堆栈 |
| characters\_upper | list | 非终结符集 |
| characters\_lower | list | 终结符集 |
| action\_goto\_table | list | action-goto表 |

## 系统界面设计

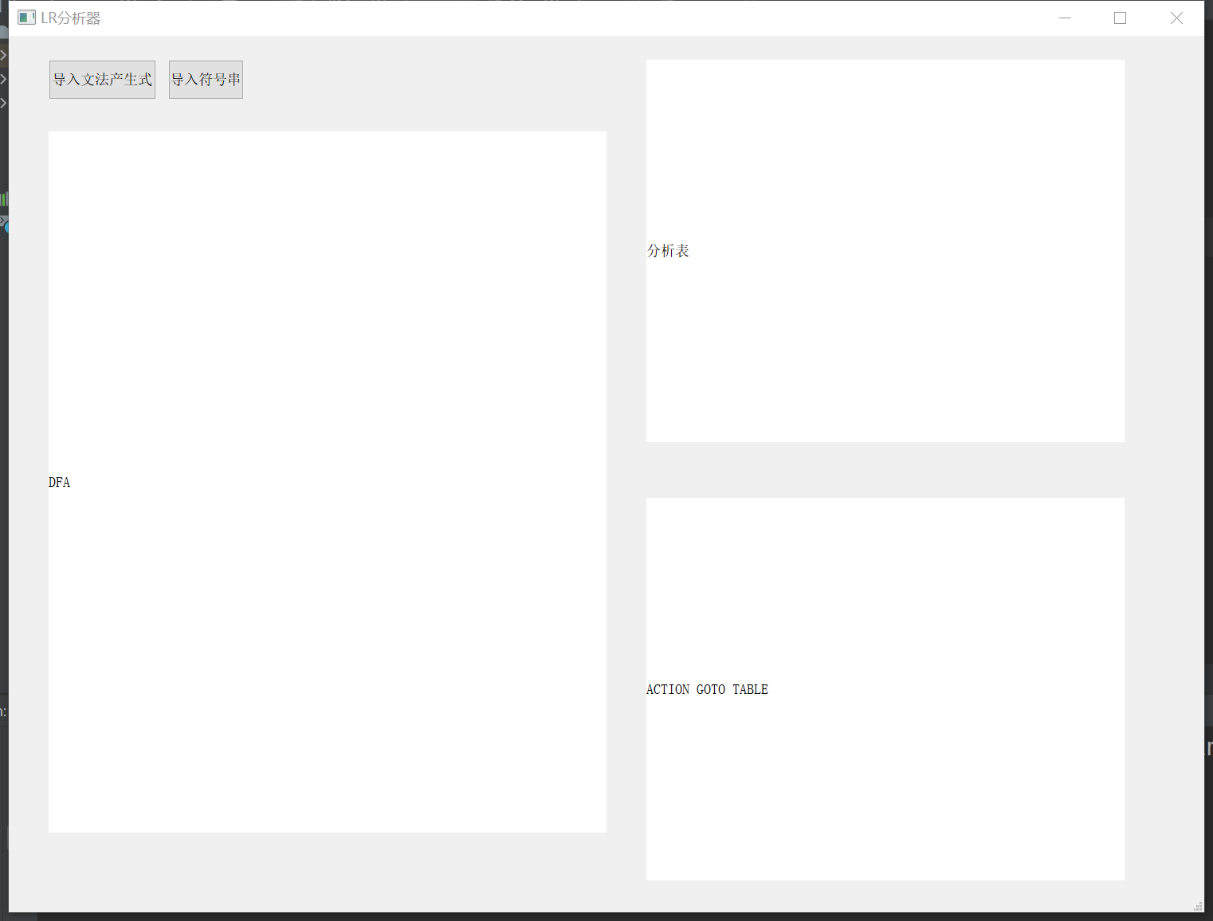


图4-4 系统界面设计

# 测试方法与测试结果

所有的测试用例输入均放在test\_input文件夹中，输出均放在test\_output文件夹中。

## 测试用例1

测试目的：测试是否能生成正确的DFA图、ACTION-GOTO表以及符号串分析表。

通过test\_data0.txt输入文法产生式：

E->aA

E->bB

A->cA

A->d

B->cB

B->d

输出结果：

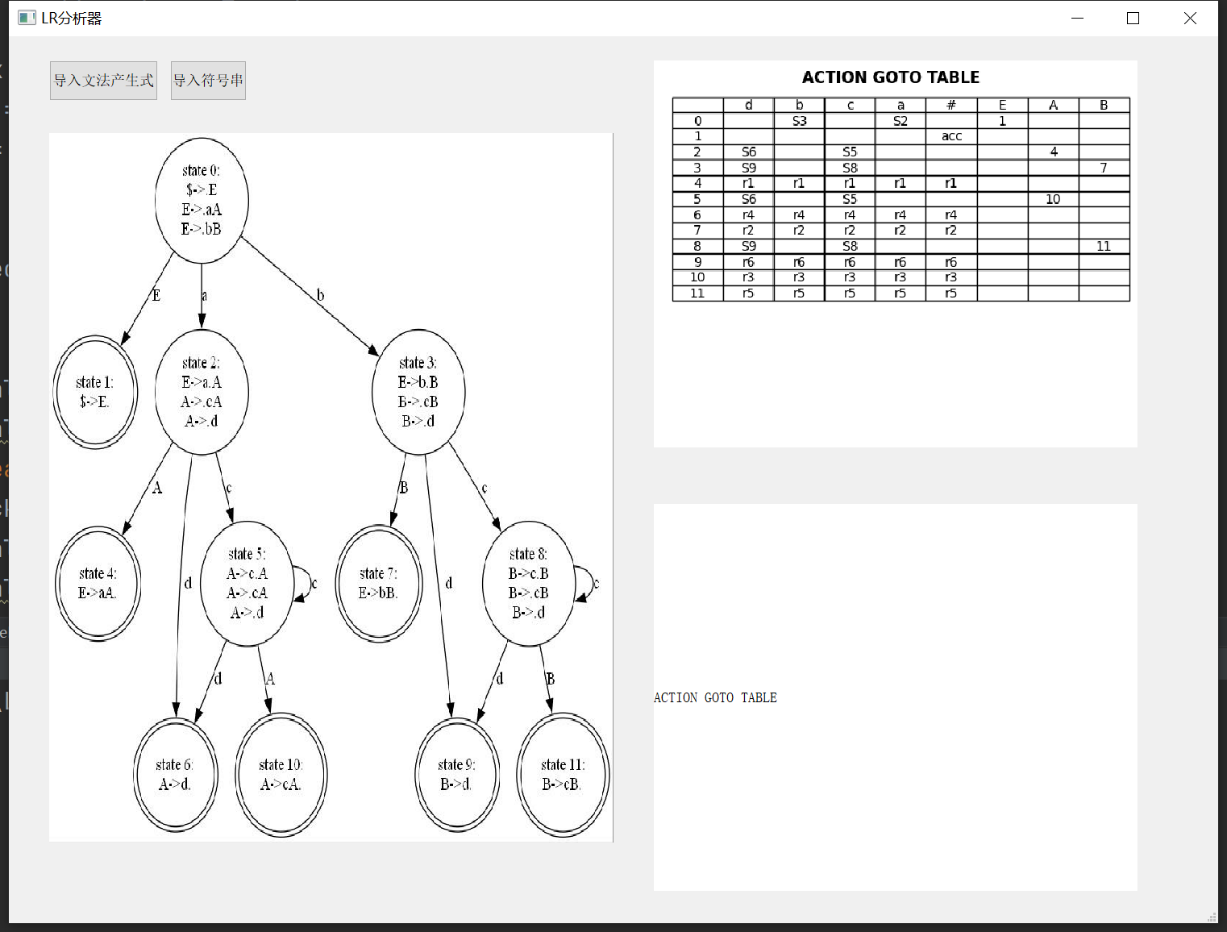


图5-1-1 测试结果1-1

通过test\_0\_0.txt输入字符串：bccd

输出结果：

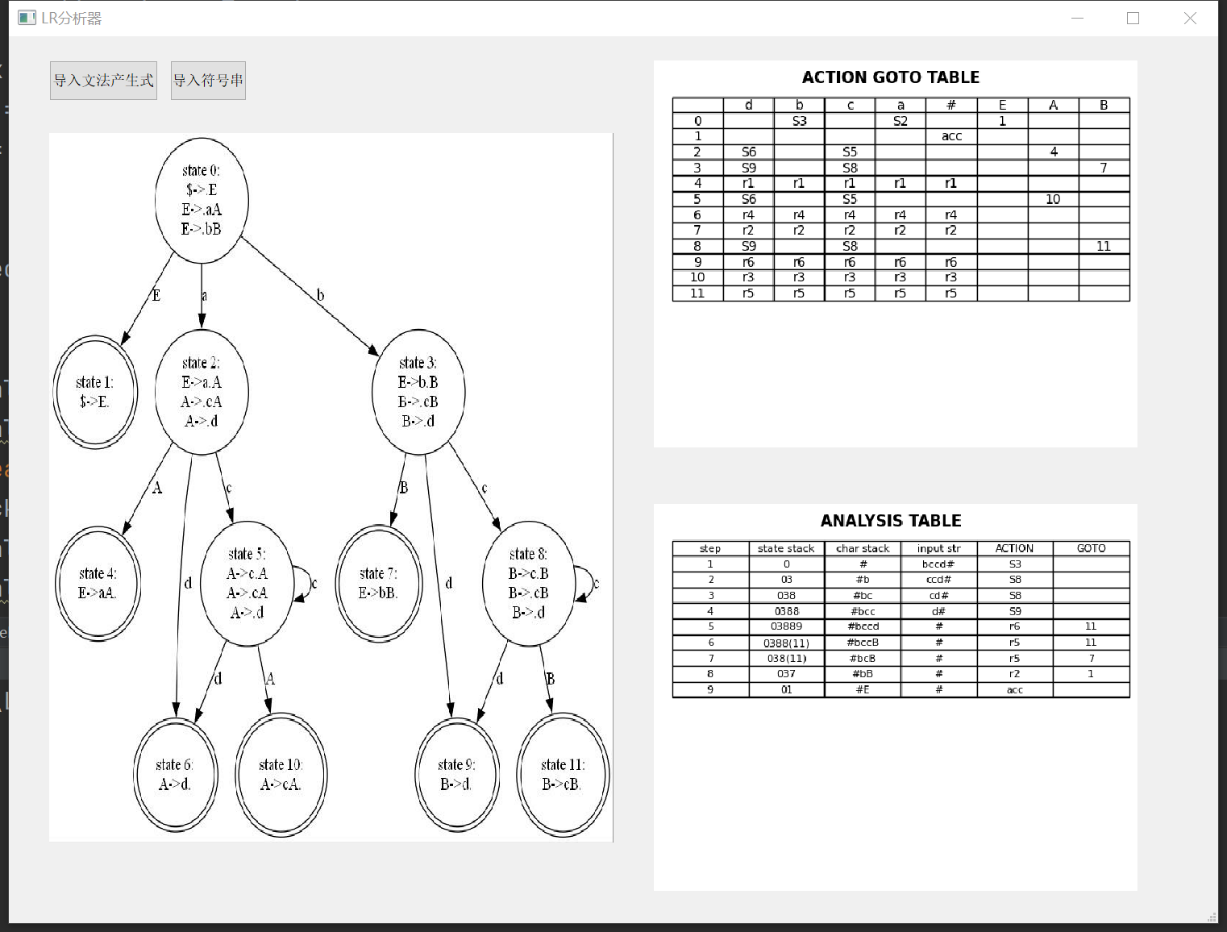


图5-1-2 测试结果1-2

## 测试用例2

测试目的：测试是否能生成正确的DFA图、ACTION-GOTO表以及符号串分析表。

通过test\_data1.txt输入文法产生式：

S->aAcBe

A->b

A->Ab

B->d

输出结果：

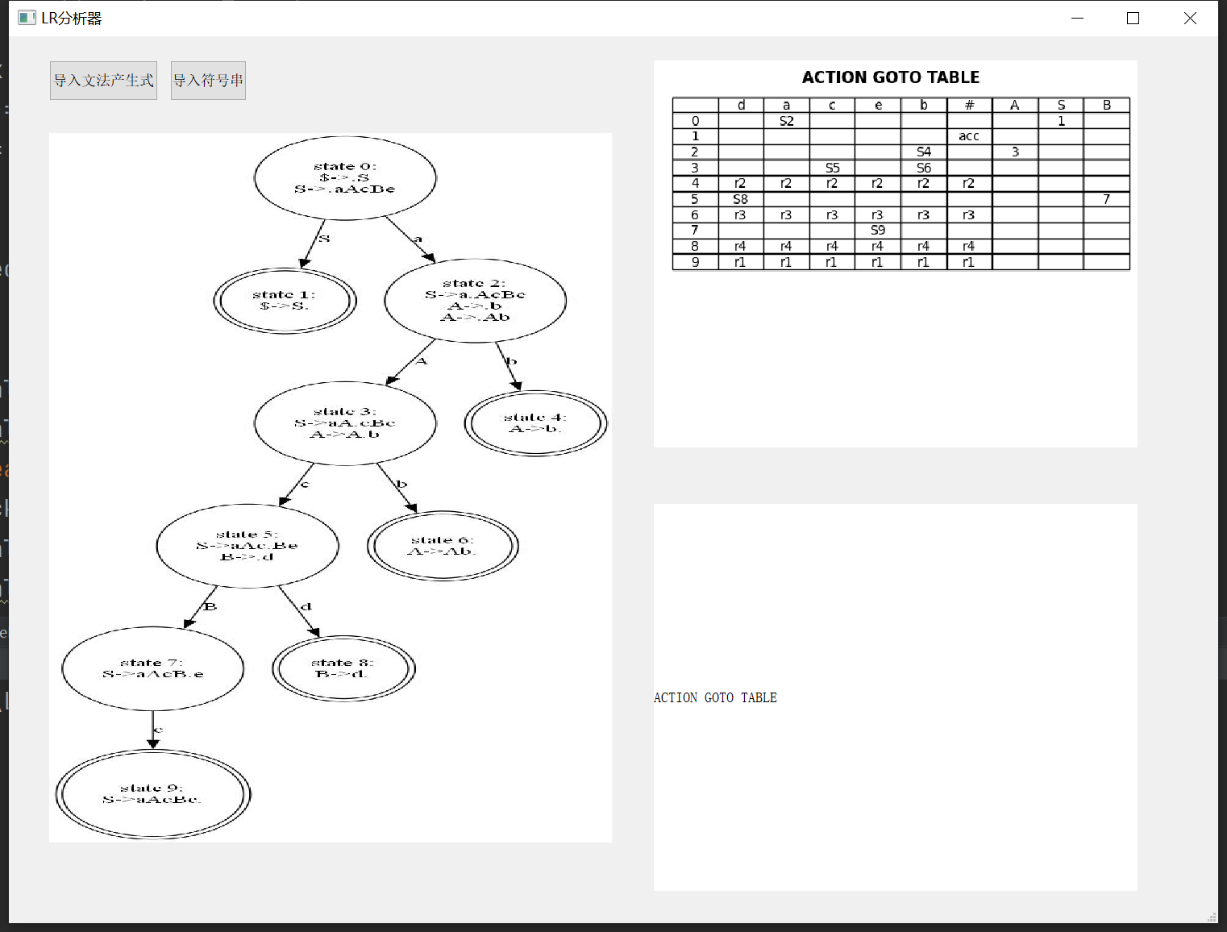


图5-2-1 测试结果2-1

通过test\_1\_0.txt输入字符串：abbcde

输出结果：

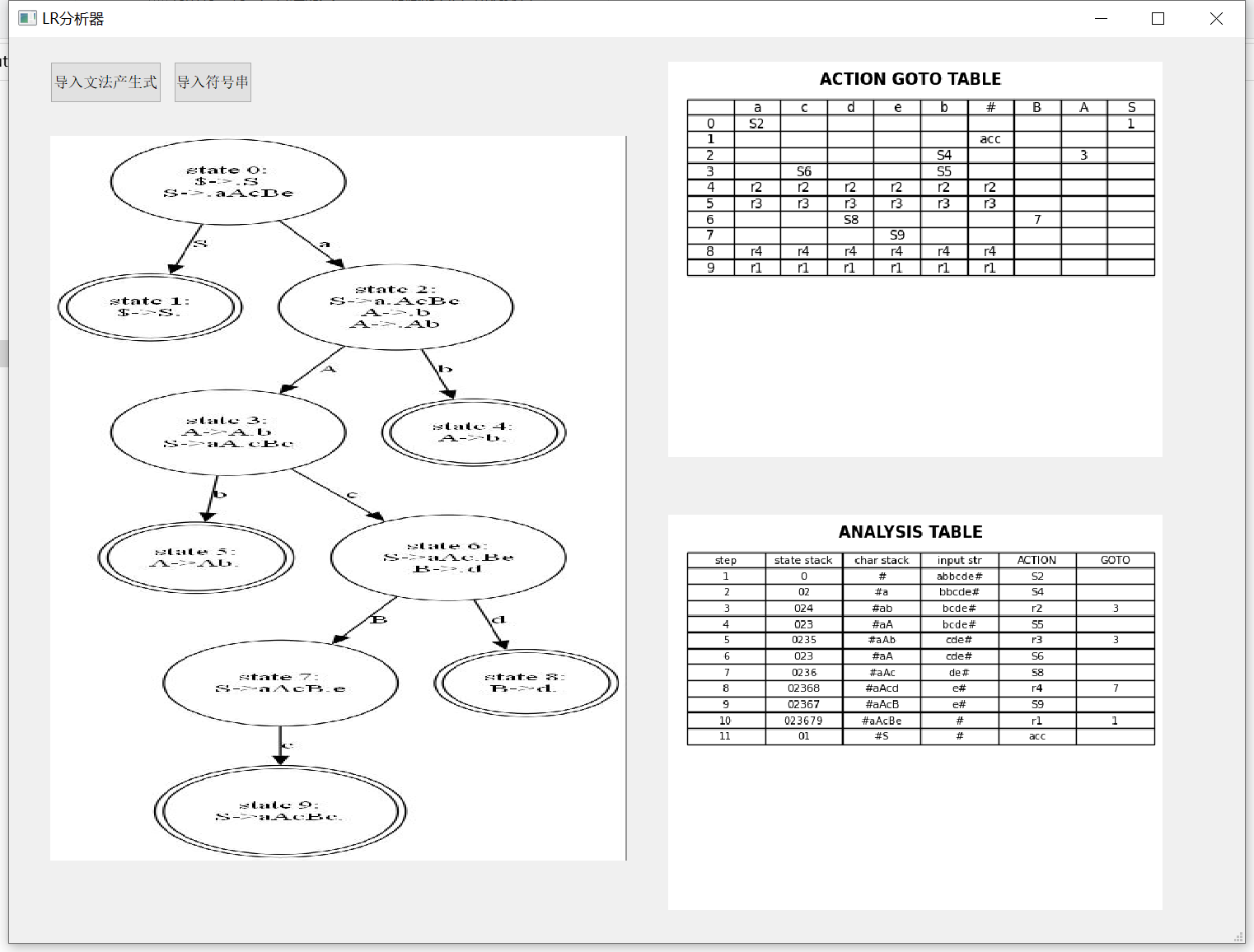


图5-2-2 测试结果2-2

## 测试用例3

测试目的：测试是否能生成正确的DFA图、ACTION-GOTO表，以及分析错误的符号串。

文法产生式输入输出同测试用例1。

通过test\_0\_1.txt输入字符串：b

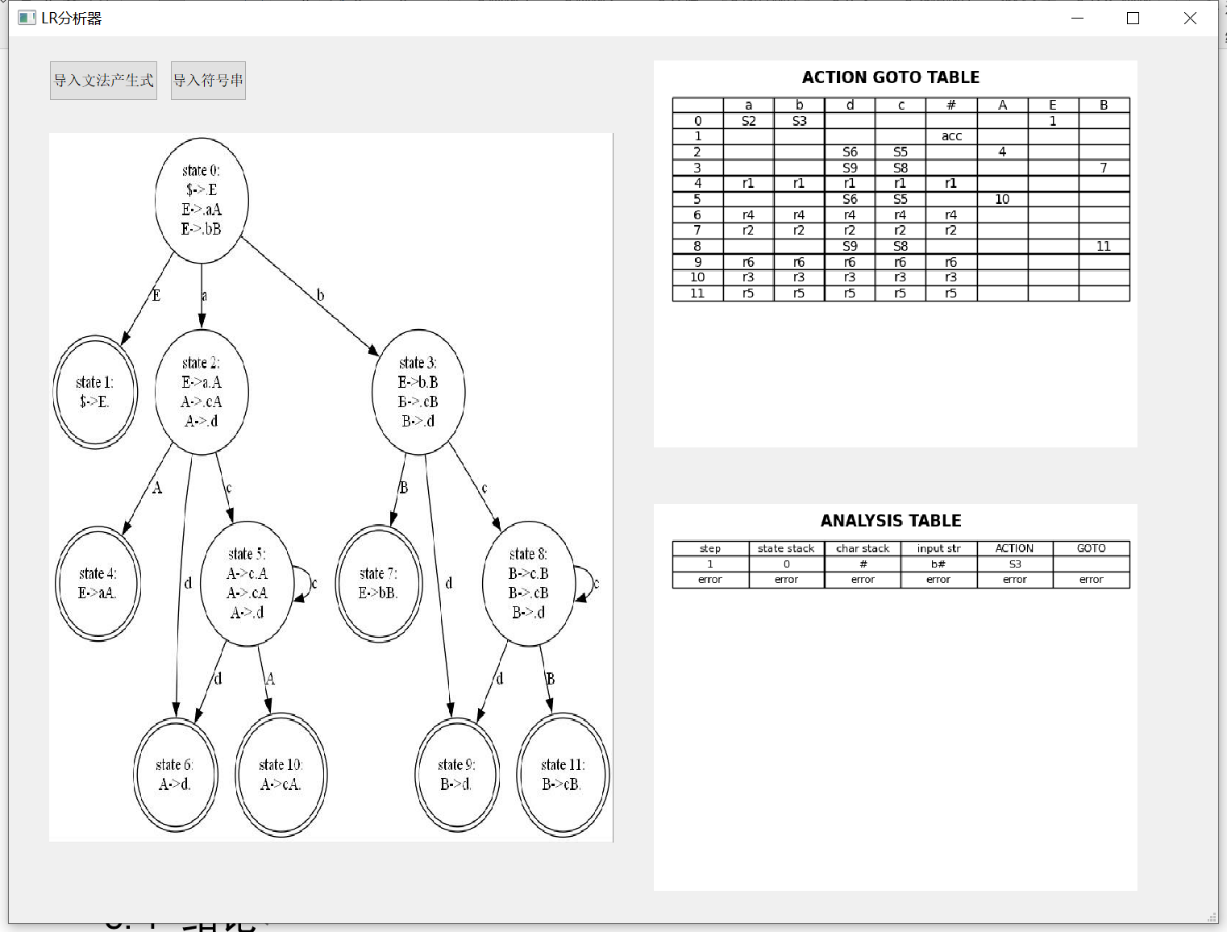


图5-3-1 测试结果3-1

# 结论和展望

## 结论

本次实验通过LR(0)分析法对LR(0)文法进行了分析和处理，得到了文法的项目集规范族以及状态间的关系，并且通过二者构建了LR(0)分析表。通过LR(0)分析表，我们可以判断符号串是否属于文法中的句子。

## 不足之处

本次课程设计的不足之处还有很多。

第一，程序只能识别形如“”的单个字符，而不能识别形如“”的字符。因此，输入的txt文件中只能包含单个字符，而拓广文法中的形如“”的字符均用$来代替。

第二，LR(0)分析法中构建项目规范族的方式很多，本次课设由于能力水平有限以及时间因素，仅展示了其中的一种方式。

第三，界面较为简陋，结果的展示形式有待丰富。

## 收获

通过本次课程设计，我收获到了很多知识。

首先，通过利用高级语言Python实现LR(0)分析法，我对编译技术和理论的理解进一步加深，同时对语法分析的原理和过程都有了进一步的巩固。

其次，由于本次课程设计基于Python语言实现，我对Python语言的应用更加熟练，对我的编程水平起到了锻炼作用，将平时学习的理论知识应用于实际。

最后，通过Graphviz和Matplotlib将LR(0)分析法的分析过程进行可视化，使得我对LR(0)分析法有了更加清晰和直观的认识，并激发了我进一步对编译原理学习的热情。

## 学习体会和教学评价

通过进行本学期的《编译原理》课程的学习，我有许多感悟和体会。

刚接触到这门课的时候，学的一头雾水，感觉像是在听天书，看书也看不明白在说什么，即使勉强学会了，学习过的知识也互相之间割裂开来，很难联系在一起。慢慢的，通过老师耐心细致的讲解，我开始逐渐体会到了编译原理的魅力，曾经课本上零散的知识逐渐在脑海中形成体系，并与其他学过和正在学习的知识和科目融会贯通。

通过《编译原理》这门课的学习，我了解到了什么是编译程序，掌握了编译程序工作的基本过程，从微观的角度探究了编译程序各个阶段所需完成的任务，从而使我对程序有了更加深刻的理解，为我后续的计算机学习打下了坚实的基础。

在此，十分感谢老师这学期以来的辛苦付出，没有您专业和耐心的讲解，就没有我今天对于编译知识的了解和掌握。

# 参考文献

[1]王生原,董渊,张素琴,吕映芝,蒋维杜.编译原理.第三版.北京:清华大学出版社,2015. 123-137.