
集美大学计算机工程学院

编译原理课程设计报告

选题名称: LL(1) 语法分析器的设计与开发

院(系): 计 算 机 工 程 学 院

专 业: 计算机科学与技术

班 级: 计算 1613 计算 1614

姓 名: 金辰宇 学 号: 201621121118

姓 名: 朱建玮 学 号: 201621121086

指导教师: 付永钢

学年学期: 2018 ~ 2019 学年 第 1 学期

2019 年 1 月 15 日

摘要:

LL 分析器是一种处理某些上下文无关文法的自顶向下分析器。因为它从左（Left）到右处理输入，再对句型执行最左推导出语法树（Left derivation，相对于 LR 分析器）。能以此方法分析的文法称为 LL 文法。本次课程设计中设计了基于 Python 3 实现的表格驱动的 LL(1)语法分析器，实现了从输入文法，对所输入文法进行分析，消除左递归，计算 FIRST、FOLLOW、SELECT 集，判断是否为 LL(1)文法，构造预测分析表，对输入的句子进行语法分析，预测分析中的错误分析等一系列操作。遵守 Python 3 面向对象的开发原则进行开发，具有良好的可移植性和安全性，并利用 Git 作为版本控制工具，代码托管在 GitHub 方便协作开发。项目地址：<https://github.com/AndyZhuAZ/Compilation-Course-Design>

关键词：LL 分析器；Python；表格驱动的分析器；

目 录

目录

1 选题内容.....	4
1.1 选题要求.....	4
1.2 需求分析.....	4
2 实现原理.....	4
2.1 LL(1)文法	4
2.1.1 文法的左递归.....	4
2.2 计算 FRIST 集.....	5
2.3 计算 FOLLOW 集	5
2.4 预测分析方法.....	5
2.5 计算 SELECT 集与 LL(1)文法判断.....	5
3 系统设计.....	6
3.1 数据结构.....	6
3.2 算法设计.....	6
3.3 流程图.....	7
4 主要代码.....	7
5 程序调试.....	8
实验环境.....	8
调试过程.....	9
6 运行样例.....	9

1 选题内容

1.1 选题要求

- (1) 基于 PL/0 语言，通过编程判断该文法是否为 LL(1)文法；
- (2) 计算出文法的 First() 、 Follow()
- (3) 构造相应文法的预测分析表
- (4) 对某个输入句子进行语法分析

1.2 需求分析

需要程序读入文本文件中的文法，首先需要对文法进行分析，当文法存在间接左递归时对文法进行间接左递归转换直接左递归，存在直接左递归时直接消除直接左递归，再对文法中出现的符号进行分类：非终结符集与终结符集，并分别计算 FIRST、FOLLOW、SELECT 集，构建预测分析表，完成对输入句子的语法分析。

2 实现原理

2.1 LL(1)文法

LL(1)文法是一类可以进行确定的自顶向下语法分析的文法。就是要求描述语言的文法是无左递归的和无回溯的。根据 LL(1)文法的定义,对于同一非终结符 A 的任意两个产生式 $A \rightarrow a$ 和 $A \rightarrow b$,都要满足: $SELECT(A \rightarrow a) \cap SELECT(A \rightarrow b) = \emptyset$ 。

2.1.1 文法的左递归

当一个文法是左递归文法时，采用自顶向下分析法会使分析过程进入无穷循环之中。所以采用自顶向下语法分析需要消除文法的左递归性。文法的左递归是指若文法中对任一非终结符 A 有推导 $A \Rightarrow A \dots$ ，则称该文法是左递归的。左递归又可以分为直接左递归和间接左递归。

直接左递归

若文法中的某一产生式形如 $A \rightarrow A\alpha$, $\alpha \in V^*$

则称该文法是直接左递归的。消除直接左递归的方法：设有产生式是关于非终结符 A 的直接左递归: $A \rightarrow A\alpha|\beta$ ($\alpha, \beta \in V^*$, 且 β 不以 A 开头) 对 A 引入一个新的非终结符 A', 把上式改写为:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' | \epsilon$

间接左递归

若文法中存在某一非终结符 A，使得 $A \Rightarrow A \dots$ 至少需要两步推导，则称该文法是间接左递归的。消除间接左递归的方法：

【方法一】采用代入法把间接左递归变成直接左递归。

【方法二】直接改写文法：设有文法 $G[0][S]$: $S \rightarrow A\alpha|\beta$ (1) $A \rightarrow S\gamma$ (2) 因为 $S \Rightarrow A\alpha \Rightarrow S\gamma\alpha$ ，所以 S 是一个间接递归的非终结符。为了消除这种间接左递归，将(2)式代入(1)式，即可得到与原文法等价的文法（可以证明）: $S \rightarrow S\gamma\alpha|\beta$ (3) (3)式是直接左递归的，可以采用前面介绍的消除直接左递归

的方法，对文法进行改写后可得文法： $S \rightarrow \beta S' \quad S' \rightarrow \gamma \alpha S' | \epsilon$

2.2 计算 FIRST 集

(1) 若 $X \in VT$ ，则 $First(X) = \{X\}$ (2) 若 $X \in VN$ ，且有产生式 $X \rightarrow a \dots$, $a \in VT$ 则 $First(X) = \{X\}$
(3) 若 $X \in VN$ ，且有产生式 $X \rightarrow \epsilon$, 则 $First(X) = \{X\}$ (4) 若 X, Y_1, Y_2, \dots, Y_n 都 $\in VN$ ，而由产生式 $X \rightarrow Y_1 Y_2 \dots Y_n$ 。当 Y_1, Y_2, \dots, Y_{i-1} 都能推导出 ϵ 时，(其中 $1 \leq i \leq n$)，则 $First(Y_1) - \{\epsilon\}, First(Y_2) - \{\epsilon\}, \dots, First(Y_i)$ 都包含在 $First(X)$ 中 (5) 当 (4) 中所有 Y_i 都能推导出 ϵ , ($i=1, 2, \dots, n$)，则 $First(X) = First(Y_1) \cup First(Y_2) \cup \dots \cup First(Y_n) \cup \{\epsilon\}$ 反复使用上述步骤直到每个符合的 $First$ 集合不再增大为止。

2.3 计算 FOLLOW 集

对文法中的每个 $A \in VN$ ，计算 $Follow(A)$ ：(1) 设 S 为文法的开始符合，把 $\{\#\}$ 加入 $Follow(S)$ 中；(2) 若 $A \rightarrow \alpha B \beta$ 是一个产生式，则把 $First(\beta)$ 的非空元素加入 $Follow(B)$ 中，如果 β 能推导出 ϵ ，则把 $Follow(A)$ 也加入 (B) 中；(3) 反复使用以上步骤直到每个非终结符号的 $Follow$ 集不再增大为止。

2.4 预测分析方法

预测分析方法是自顶向下分析的另一种方法，一个预测分析器是由三个部分组成：预测分析程序；先进后出栈；预测分析表。

2.5 计算 SELECT 集与 LL(1) 文法判断

SELECT 集合定义如下：

一个产生式的选择符号集 SELECT。给定上下文无关文法的产生式 $A \rightarrow \alpha, A \in VN, \alpha \in V^* A \rightarrow \alpha, A \in VN, \alpha \in V^*$ ，若 $\alpha \neq * \epsilon \alpha \neq * \epsilon$ ，则 $SELECT(A \rightarrow \alpha) = FIRST(\alpha) SELECT(A \rightarrow \alpha) = FIRST(\alpha)$ 。

如果 $\alpha \Rightarrow * \epsilon \alpha \Rightarrow * \epsilon$ ，则 $SELECT(A \rightarrow \alpha) = (FIRST(\alpha) - \{\epsilon\}) \cup FOLLOW(A) SELECT(A \rightarrow \alpha) = (FIRST(\alpha) - \{\epsilon\}) \cup FOLLOW(A)$ 。

根据 SELECT 集的定义，我们可以知道，实际上相同左部的产生式 SELECT 不相交与如下 LL(1) 文法判例等价：

文法 G 是 LL(1) 的，当且仅当 G 的任意两个具有相同左部的产生式 $A \rightarrow \alpha | \beta$ 满足下面的条件：

如果 α 和 β 均不能推导出 ϵ ，则 $FIRST(\alpha) \cap FIRST(\beta) = \Phi$

α 和 β 至多有一个能推导出 ϵ

如果 $\beta \Rightarrow * \epsilon$ ，则 $FIRST(\alpha) \cap FOLLOW(A) = \Phi$ ；如果 $\alpha \Rightarrow * \epsilon$ ，则 $FIRST(\beta) \cap FOLLOW(A) = \Phi$ ；

因此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， $A \rightarrow \alpha, A \rightarrow \beta \quad A \rightarrow \alpha, A \rightarrow \beta$ ，满足

$SELECT(A \rightarrow \alpha) \cap SELECT(A \rightarrow \beta) = \emptyset$

$SELECT(A \rightarrow \alpha) \cap SELECT(A \rightarrow \beta) = \emptyset$

其中 $\alpha, \beta \quad \alpha, \beta$ 不同时能 $\Rightarrow * \epsilon \Rightarrow * \epsilon$ 。

3 系统设计

3.1 数据结构

```
def __init__(self):
    self.representation = []    # 表达式
    self.first_state = ''      # 初始状态

    self.VT = set()            # 终结符
    self.VN = set()            # 非终结符

    self.first = dict()        # first集
    self.follow = dict()       # follow集
    self.select = dict()       # select集
    self.table = dict()        # 预测分析表
```

主要数据结构

表达式使用 list 数据结构，list 中每个字符串，代表一个表达式，初始状态使用字符串表示，终结符，非终结符，选用 set 集合形式，防治重复加入，而 first, follow, select 集，以及预测分析表，使用字典 dict 形式，便于查表，其中预测分析表为二级字典。

```
select = {dict}: {'E -> T E': {'(', 'id', 'E' -> T E': {'+', 'E' -> e': {'}', '$'}, 'T -> F T': {'(', 'id', 'T' -> * F T': {'*', 'T' -> e': {'+', '}', '$'}, 'F -> ( E ): {'(', '}', '$'}, 'E' -> T E' (4771777072) = {set} {'(', 'id'}
'E' -> T E' (4771777136) = {set} {'+'}
'E' -> e' (4771503952) = {set} {'}', '$'}
'T -> F T' (4771776944) = {set} {'(', 'id'}
'T' -> * F T' (4771777264) = {set} {'*'}
'T' -> e' (4771504040) = {set} {'+', '}', '$'}
'F -> ( E )' (4771777200) = {set} {'('}
'F -> id' (4771582280) = {set} {'id'}
__len__ = {int} 8
table = {dict}: {'E': {'E -> T E', 'id': 'E -> T E', '': 'synch', '$': 'synch', 'E': {'+', 'E' -> T E', '': 'E' -> e', '$': 'E' -> e'}, 'T': {'T -> F T', '': 'T -> * F T', '': 'T -> e', '$': 'T -> e'}}, 'E' (4544733848) = {dict}: {'E -> T E', 'id': 'E -> T E', '': 'synch', '$': 'synch'}
'E' (4544733848) = {dict}: {'E -> T E', 'id': 'E -> T E', '': 'synch', '$': 'synch'}
'E' (4771918040) = {dict}: {'+', 'E' -> T E', '': 'E' -> e', '$': 'E' -> e'}
'T' (4544712752) = {dict}: {'T -> F T', 'id': 'T -> F T', '': 'synch', '$': 'synch'}
'T' (4771918992) = {dict}: {'*', 'T' -> * F T', '': 'T' -> e', '$': 'T' -> e'}
'F' (4544733904) = {dict}: {'F -> ( E )', 'id': 'F -> id', '': 'synch', '': 'synch', '': 'synch', '$': 'synch'}
__len__ = {int} 5
```

字典数据结构

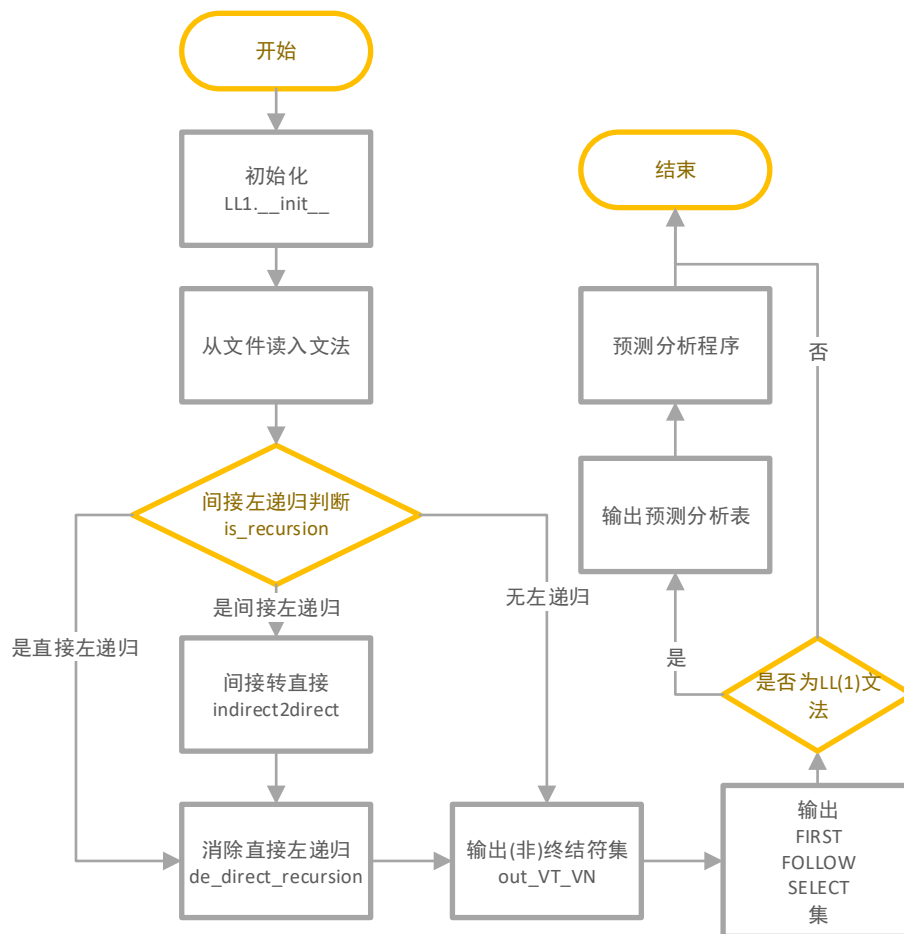
```
left_representation = {str} 'E'
representation = {str} 'E -> E + T | T'
right = {list} <class 'list': ['E', '+', 'T']
0 = {str} 'E'
1 = {str} '+'
2 = {str} 'T'
__len__ = {int} 3
right_representation = {str} 'E + T | T'
right_representation_list = {list} <class 'list': ['E + T', 'T']
0 = {str} 'E + T'
1 = {str} 'T'
__len__ = {int} 2
```

列表数据结构

3.2 算法设计

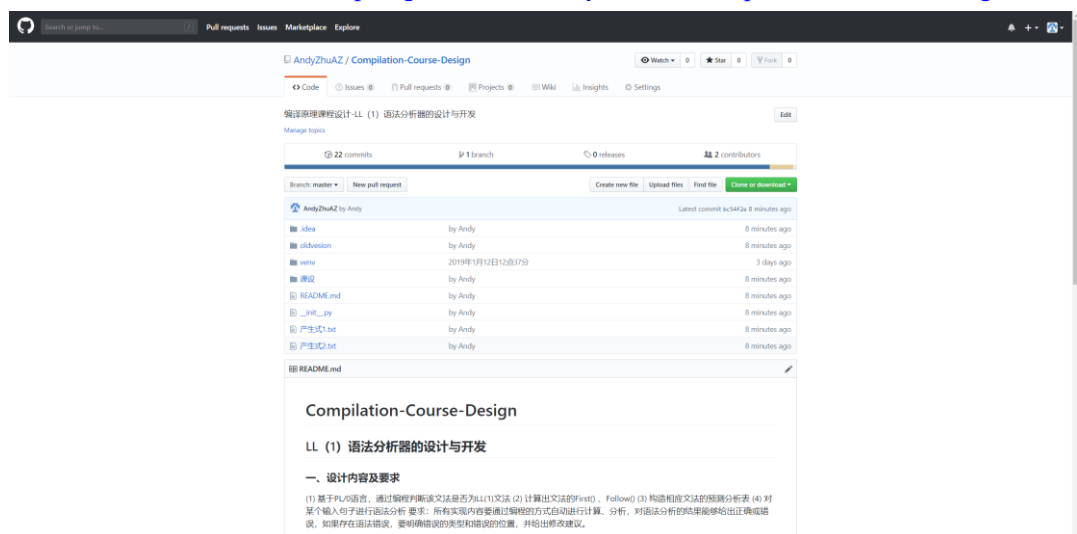
该算法从本地读取用户输入文法，检查文法是否存在直接或间接左递归，若存在间接左递归，则将间接左递归转化为直接左递归，并将消除直接左递归。将用户输入转化为不含左递归表达式后，根据推导关系，计算用户输入文法的非终结符，终结符集合。再根据表达式计算非终结符的 FIRST, FOLLOW, SELECT 集合，并根据 SELECT 集合，来判断该文法是否符合 LL(1)文法规则，若不符合文法规则，推出程序，若符合文法规则，根据 FIRST, FOLLOW 集合，构建预测分析表，并对用户输入表达式进行语法分析。

3.3 流程图



4 主要代码

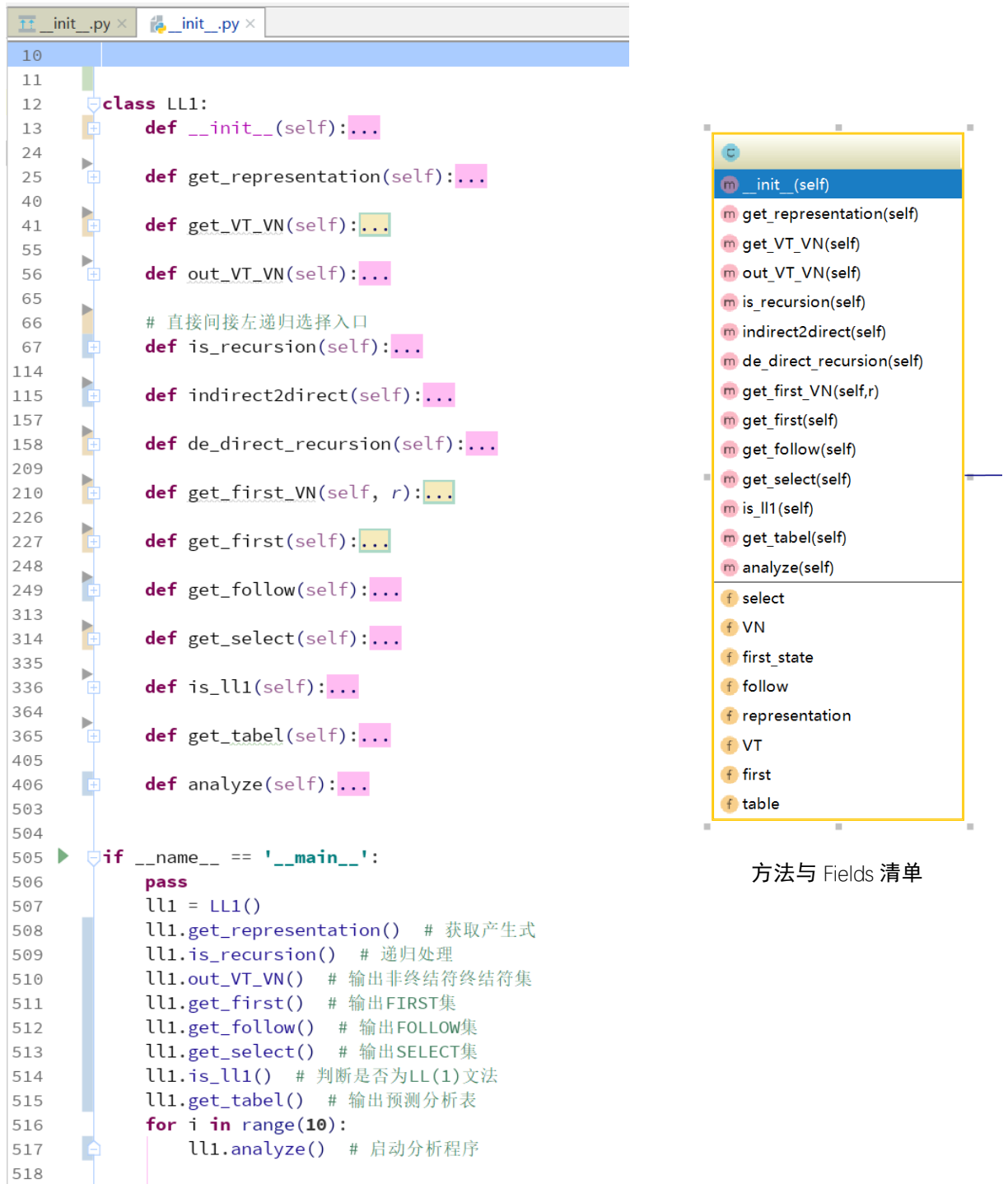
完整项目托管在 GitHub: <https://github.com/AndyZhuAZ/Compilation-Course-Design>



代码清单 https://github.com/AndyZhuAZ/Compilation-Course-Design/blob/master/LL1.__init__.py

此处仅列举主要代码结构

右图为 Python 类 方法与 Fields 清单：



The image shows a PyCharm IDE window with a Python file named `__init__.py`. The code defines a class `LL1` with several methods and a main execution block. To the right, a pop-up window displays the class's methods and fields.

```
10
11
12 class LL1:
13     def __init__(self):...
24
25     def get_representation(self):...
40
41     def get_VT_VN(self):...
55
56     def out_VT_VN(self):...
65
66     # 直接间接左递归选择入口
67     def is_recursion(self):...
114
115     def indirect2direct(self):...
157
158     def de_direct_recursion(self):...
209
210     def get_first_VN(self, r):...
226
227     def get_first(self):...
248
249     def get_follow(self):...
313
314     def get_select(self):...
335
336     def is_ll1(self):...
364
365     def get_tabel(self):...
405
406     def analyze(self):...
503
504
505 if __name__ == '__main__':
506     pass
507     ll1 = LL1()
508     ll1.get_representation() # 获取产生式
509     ll1.is_recursion() # 递归处理
510     ll1.out_VT_VN() # 输出非终结符终结符集
511     ll1.get_first() # 输出FIRST集
512     ll1.get_follow() # 输出FOLLOW集
513     ll1.get_select() # 输出SELECT集
514     ll1.is_ll1() # 判断是否为LL(1)文法
515     ll1.get_tabel() # 输出预测分析表
516     for i in range(10):
517         ll1.analyze() # 启动分析程序
518
```

The right-hand window lists the following methods (m) and fields (f):

- m `__init__(self)`
- m `get_representation(self)`
- m `get_VT_VN(self)`
- m `out_VT_VN(self)`
- m `is_recursion(self)`
- m `indirect2direct(self)`
- m `de_direct_recursion(self)`
- m `get_first_VN(self, r)`
- m `get_first(self)`
- m `get_follow(self)`
- m `get_select(self)`
- m `is_ll1(self)`
- m `get_tabel(self)`
- m `analyze(self)`
- f `select`
- f `VN`
- f `first_state`
- f `follow`
- f `representation`
- f `VT`
- f `first`
- f `table`

方法与 Fields 清单

5 程序调试

实验环境

Python 3.7.0

PyCharm 2018.3.3 (Professional Edition)

Windows 10 10.0

pandas 0.23.4

prettytable 0.7.2

调试过程

1 左递归调试

2 FOLLOW 集调试

计算非终结符的 FOLLOW 集时，由于非终结符使用 `set()` 数据类型，`set()` 是一个无序集，根据 FOLLOW 计算规则，“如果存在一个产生式 $A \rightarrow \alpha B$ ，或存在产生式 $A \rightarrow \alpha B \beta$ 且 $\text{first}(\beta)$ 包含 ϵ ，那么 $\text{follow}(A)$ 中的所有符号都在 $\text{follow}(B)$ 中”，当出现这种情况是，计算 $\text{FOLLOW}(B)$ 需要依赖 $\text{FOLLOW}(A)$ ，而因为非终结符集无序，所以存在计算顺序问题，此处使用暴力方法，计算 3 次 FOLLOW 集，第一次不考虑依赖情况只计算请他情况下的 FOLLOW 集，第二次考虑依赖，再次重新计算，第三次在第二次基础上，再次计算，以保证 FOLLOW 集计算正确。

3 分析程序调试

当输入表达式 `id + id` 时，无法报错，根据文法 $F \rightarrow (E) | id$ ，只有 $F \rightarrow ()$ ，而其他情况下，`(,)` 一律不会进入堆栈，当出现单独`(`时，`()`都进栈，可以识别缺少`)`，但是当出现单独`)`时，`(,)`均没有进栈，无法识别缺失`(`问题。经过查找发现，此处存在一个堆栈未排空的问题，通过在结束位置增加额外判断解决。

6 运行样例

输入含有直接左递归文法 $E \rightarrow E + T | T, T \rightarrow T * F | F, F \rightarrow (E) | id$

```
LL1_final x
/Users/jinch/anaconda3/bin/python /Users/jinch/study/houmework/编译原理/课设/LL1_final.py
/Users/jinch/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed, may indic
return f(*args, **kwargs)
产生式: ['E -> E + T | T', 'T -> T * F | F', 'F -> ( E ) | id']
直接左递归
消除左递归 ['E -> T E', 'E' -> + T E', 'E' -> ε', 'T -> F T', 'T' -> * F T', 'T' -> ε', 'F -> ( E )', 'F -> id']
非终结符: {'F', 'T', 'E', 'id'}
终结符: {'(', ')', '*', 'id', '+'}
FIRST
E {'id', '('}
E' {'ε', '+'}
T {'id', '('}
T' {'ε', '*'}
F {'id', '('}
FOLLOW
F {'$', '*', ')', '+'}
T {'$', ')', '+'}
E' {'$', ')', '+'}
E' {'$', '$'}
E' {'$', '$'}
select
E -> T E' {'id', '('}
E' -> + T E' {'+'}
E' -> ε {'ε'}, '$'
T -> F T' {'id', '('}
T' -> * F T' {'*'}
T' -> ε {'ε'}, '$', '+'
F -> ( E ) {'('}
F -> id {'id'}
符合LL1文法
TABLE
E {'id': 'E -> T E', '(': 'E -> T E', ')': 'synch', '$': 'synch'}
E' {'+': 'E' -> + T E', ')': 'E' -> ε', '$': 'E' -> ε'}
T {'id': 'T -> F T', '(': 'T -> F T', '$': 'synch', ')': 'synch', '+': 'synch'}
T' {'*': 'T' -> * F T', '$': 'T' -> ε', ')': 'T' -> ε', '+': 'T' -> ε'}
F {'(': 'F -> ( E )', 'id': 'F -> id', '$': 'synch', '*': 'synch', ')': 'synch', '+': 'synch'}
df_TABLE
      $      (      )      *      +      id
E      synch  E -> T E'      synch      E -> T E'
E'  E' -> ε      E' -> ε      E' -> + T E'
F      synch  F -> ( E )      synch      synch      synch      F -> id
T      synch  T -> F T'      synch      synch      synch      T -> F T'
T'  T' -> ε      T' -> ε      T' -> * F T'      T' -> ε
请输入:
```

消除左递归, FIRST, FOLLOW, SELECT, 及预测分析表结果

请输入: (id + id (

栈	输入	动作
['\$', 'E']	['(', 'id', '+', 'id', '(', '\$']	
['\$', "E'", 'T']	['(', 'id', '+', 'id', '(', '\$']	E → T E'
['\$', "E'", "T'", 'F']	['(', 'id', '+', 'id', '(', '\$']	T → F T'
['\$', "E'", "T'", ')', 'E', '(']	['(', 'id', '+', 'id', '(', '\$']	F → (E)
['\$', "E'", "T'", ')', 'E']	['id', '+', 'id', '(', '\$']	匹配(
['\$', "E'", "T'", ')', "E'", 'T']	['id', '+', 'id', '(', '\$']	E → T E'
['\$', "E'", "T'", ')', "E'", "T'", 'F']	['id', '+', 'id', '(', '\$']	T → F T'
['\$', "E'", "T'", ')', "E'", "T'", 'id']	['id', '+', 'id', '(', '\$']	F → id
['\$', "E'", "T'", ')', "E'", "T'"]	['+', 'id', '(', '\$']	匹配id
['\$', "E'", "T'", ')', "E'"]	['+', 'id', '(', '\$']	T' → ε
['\$', "E'", "T'", ')', "E'", 'T', '+']	['+', 'id', '(', '\$']	E' → + T E'
['\$', "E'", "T'", ')', "E'"]	['id', '(', '\$']	匹配+
['\$', "E'", "T'", ')', "E'", "T'", 'F']	['id', '(', '\$']	T → F T'
['\$', "E'", "T'", ')', "E'", "T'", 'id']	['id', '(', '\$']	F → id
['\$', "E'", "T'", ')', "E'", "T'"]	['(', '\$']	匹配id
['\$', "E'", "T'", ')', "E'"]	['(', '\$']	error, 位置: 4, 字符: (, 缺少: ['*']
['\$', "E'", "T'", ')', "E'"]	['\$']	T' → ε
['\$', "E'", "T'", ')']	['\$']	E' → ε
['\$', "E'", "T'"]	['\$']	error, 位置: 5, 字符: \$, 缺少:)
['\$', "E'"]	['\$']	T' → ε
['\$']	['\$']	E' → ε

输入错误表达式文法分析

请输入: id + (id * id)

栈	输入	动作
['\$', 'E']	['id', '+', '(', 'id', '*', 'id', ')', '\$']	
['\$', "E'", 'T']	['id', '+', '(', 'id', '*', 'id', ')', '\$']	E → T E'
['\$', "E'", "T'", 'F']	['id', '+', '(', 'id', '*', 'id', ')', '\$']	T → F T'
['\$', "E'", "T'", 'id']	['id', '+', '(', 'id', '*', 'id', ')', '\$']	F → id
['\$', "E'", "T'"]	['+', '(', 'id', '*', 'id', ')', '\$']	匹配id
['\$', "E'"]	['+', '(', 'id', '*', 'id', ')', '\$']	T' → ε
['\$', "E'", 'T', '+']	['+', '(', 'id', '*', 'id', ')', '\$']	E' → + T E'
['\$', "E'", 'T']	['(', 'id', '*', 'id', ')', '\$']	匹配+
['\$', "E'", "T'", 'F']	['(', 'id', '*', 'id', ')', '\$']	T → F T'
['\$', "E'", "T'", ')', 'E', '(']	['(', 'id', '*', 'id', ')', '\$']	F → (E)
['\$', "E'", "T'", ')', 'E']	['id', '*', 'id', ')', '\$']	匹配(
['\$', "E'", "T'", ')', "E'", 'T']	['id', '*', 'id', ')', '\$']	E → T E'
['\$', "E'", "T'", ')', "E'", "T'", 'F']	['id', '*', 'id', ')', '\$']	T → F T'
['\$', "E'", "T'", ')', "E'", "T'", 'id']	['id', '*', 'id', ')', '\$']	F → id
['\$', "E'", "T'", ')', "E'", "T'"]	['*', 'id', ')', '\$']	匹配id
['\$', "E'", "T'", ')', "E'", "T'", 'F', '*']	['*', 'id', ')', '\$']	T' → * F T'
['\$', "E'", "T'", ')', "E'", "T'", 'F']	['id', ')', '\$']	匹配*
['\$', "E'", "T'", ')', "E'", "T'", 'id']	['id', ')', '\$']	F → id
['\$', "E'", "T'", ')', "E'", "T'"]	[')', '\$']	匹配id
['\$', "E'", "T'", ')', "E'"]	[')', '\$']	T' → ε
['\$', "E'", "T'", ')']	[')', '\$']	E' → ε
['\$', "E'", "T'"]	['\$']	匹配)
['\$', "E'"]	['\$']	T' → ε
['\$']	['\$']	E' → ε

对输入正确表达式文法分析

输入含有间接左递归文法 $S \rightarrow Aa|b$, $A \rightarrow Ac|Sd|\epsilon$

产生式: ['S → A a | b', 'A → A c | S d | ε']

间接左递归

间接左递归变直接左递归 ['A → A c | ε | A a d | b d']

消除左递归 ["A → b d A'", "A → A'", "A' → c A'", "A' → ε", "A' → a d A'", "A' → ε"]

间接左递归转化直接左递归, 再消除左递归

```

非终结符: {'A', 'A'}
终结符: {'a', 'b', 'd', 'c'}
FIRST
A {'a', 'b', 'ε', 'c'}
A' {'a', 'ε', 'c'}
FOLLOW
A' {'$'}
A {'$'}
select
A → b d A' {'b'}
A → A' {'a', 'b', 'ε', 'c'}
A' → c A' {'c'}
A' → ε {'$'}
A' → a d A' {'a'}
符合LL1文法
TABLE
A {'b': "A → b d A'", 'a': "A → A'", 'c': "A → A'", '$': "A → A'"}
A' {'c': "A' → c A'", '$': "A' → ε", 'a': "A' → a d A'"}
df_TABLE
      $          a          b          c
A   A → A'      A → A'  A → b d A'  A → A'
A'  A' → ε  A' → a d A'          A' → c A'

```

计算 FIRST, FOLLOW, SELECT 集, 以及预测分析表

总 结

本次课程设计耗时 3 天, 利用 GitHub 协作开发, 使用了 Git 版本控制, 基于 Python 3 完成了 LL(1)分析器, 朱建玮负责了左递归处理的两个方法及 SELECT 集计算与 LL(1) 文法判断和代码维护, 金辰宇负责了 FIRST、FOLLOW 集计算和分析表构建与分析程序, 在整体开发中, 我们灵活的使用了 Python 特有的字典数据结构和对列表字符串处理的官方库, 使得各部分功能在较短的代码量下能完成地更多更安全。在 LL(1)判断中, 选用了同一非终结符的产生式的可选集不相交作为判断条件, 引入可选集概念, 使得算法思想相比课本更加简化。Python set()集合运算方法应用在 FIRST 集 FOLLOW 集 SELECT 集的计算和应用中, Python string.split()方法可以轻松的对输入字符串进行分割, 通过指定分隔符, 我们利用 split 方法实现了对多字符的符号进行处理例如 E'、id。Python dict 字典数据结构应用于 FIRST, FOLLOW, SELECT 集, 以及预测分析表, 便于查表, 其中预测分析表为二级字典。通过本次课程设计, 我们加深了对编译原理这门课程的理解, 熟练掌握了文法分析和表驱动分析程序的构建。

致 谢

[illegible]

注：

1. 致谢内容要求 600 字左右，主要感谢计算机工程学院提供的实践机会，实验室人员提供的实验环境，指导教师的辛勤指导，同组同学的互帮互助，参考文献的原作者，以及其他给你提供过帮助的所有人员和机构等；
2. 首行空二个汉字；
3. 字体为“宋体，小四号，1.5 倍行距”，数字、西文字母等用 Times New Roman 字体；
4. 其它要求同正文格式规范。

参 考 文 献

- 1 [美]Andrew W.Appel/[美]MaiaGinsburg. 现代编译原理.北京：人民邮电出版社，2018
- 2 LL 剖析器. Zh.wikipedia.org. [https://zh.wikipedia.org/wiki/ LL 剖析器](https://zh.wikipedia.org/wiki/LL_剖析器). Published 2019. Accessed January 15, 2019
- 3 编译原理 FIRST 集、FOLLOW 集、SELECT 集求法通俗解释 & LL(1)文法判定 - 多反思，多回顾，要坚持。 - CSDN 博客. Blog.csdn.net. <https://blog.csdn.net/liujian20150808/article/details/72998039>. Published 2019. Accessed January 15, 2019.

指导教师评语

学号		姓名		班级	
选题名称					
序号	评价内容			权重 (%)	得分
1	考勤记录、学习态度、工作作风与表现。			5	
2	自学情况： 上网检索机时数、文献阅读情况（笔记）。			10	
3	论文选题是否先进，是否具有前沿性或前瞻性。			5	
4	成果验收： 是否完成设计任务；能否运行、可操作性如何等。			20	
5	报告的格式规范程度、是否图文并茂、语言规范及流畅程度；主题是否鲜明、重心是否突出、论述是否充分、结论是否正确；是否提出了自己的独到见解。			30	
6	文献引用是否合理、充分、真实。			5	
7	答辩情况： 自我陈述、回答问题的正确性、用语准确性、逻辑思维、是否具有独到见解等。			25	
合计					
指导教师（签章）： _____					
_____年____月____日					