Discrete Mathematics(1)
王浩算法 Lab

王子轩,
wang-zx23@mails.tsinghua.edu.cn

November 2, 2024

1 运行结果

项目组织如下，运行结果如图。

```
- proj1
    - main.py          # 程序入口
    - input_proc.py    # 输入处理函数
    - binary_tree.py   # 构建表达式二叉树
    - state.py         # 存储逻辑连接词
    - infer.py         # 王浩算法的实现（10条推理规则+1条公理）
    - test.py
    - test.txt
    - result.txt
    - result.pdf       # 将txt文件前30条推理结果复制到typora后导出的pdf
```

实现了以下功能点：
1. 将以 markdown 格式输入的中缀表达式转换为后缀表达式。
2. 从得到的后缀表达式构建表达式二叉树。
3. 初始化存储表达式树的前后件列表，根据推理规则进行推理。
4. 输出推理步骤和结果。

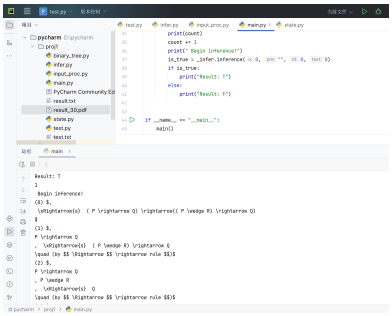

图 1: Enter Caption

2 实现思路

2.1 Input Processing

首先接受合式中缀表达式的输入, 转换为逆波兰表达式（后缀表达式），使用栈结构的优先级处理确保运算符顺序。然后将得到的后缀表达式转换为二叉树存储，实现树建立、插入、删除、遍历等操作。

Algorithm 1 Convert Infix to Reverse Polish Notation

```
Initialize res and stk as empty
for each char t in input do
    if t is an operand then
        Add t to res
    else if t is an operator then
        while stk not empty and priority(top) ≥ priority(t) do
            Pop stk to res
        end while
        Push t onto stk
    else if t is '(' then
        Push t onto stk
    else if t is ')' then
        while top of stk is not '(' do
            Pop stk to res
        end while
        Pop '(' from stk
    end if
end for
while stk not empty do
    Pop stk to res
end while
```

Algorithm 2 BinaryTree Initialization and Traversal

```
Initialize an empty stack stk
for each value in input do
    Create a new node node with value
    if value starts with '\' then
        node.right ← Pop from stk
        if stk is not empty then
            node.left ← Pop from stk
        end if
    end if
    Push node onto stk
end for
if length of stk is 1 then
    Return True
end if
```

2.2 Inference by Wanghao Algorithm

实例化“推理”类，初始化空的前件列表，后件列表初始化为只包含待推理的表达式二叉树，根据王浩算法进行合式推理。

Algorithm 3 Inference Algorithm

```
cnt ← 0
while True do
  print_prove(o, pre, id, test)
  for each v ∈ LHS do
    if v.root.value[0] then
      o ← int(v.root.value[1])
      if o = 1 then
        RHS.append(subtree from v.root.right)
      else if o = 2 then
        LHS.append(subtree from v.root.right)
        LHS.append(subtree from v.root.left)
      end if
      LHS.remove(v)
    end if
  end for
  for each v ∈ RHS do
    if v.root.value[0] = '//' then
      o ← int(v.root.value[1])
      if o = 1 then
        LHS.append(subtree from v.root.right)
      else if o ∈ {3, 4} then
        RHS.append(subtree from v.root.right)
        if o = 4 then
          LHS.append(subtree from v.root.left)
        end if
      end if
      RHS.remove(v)
    end if
  end for
  if common trees found in LHS and RHS then
    RETURN True
  end if
end while
```

2.3 Data Structure

```
class BinaryTree:
    def __init__(self, _input):
        self.root = None
        self.nodes = []
```

```
        self.L = []
        self.R = []
        self._input = _input
        self.is_valid = self.init_tree()

class Inference:
    def __init__(self, l, r, class_tree_instance, rules=S,):
        """
        :param:LHS存储前件所有命题公式对应的表达式二叉树
        :param:RHS存储后件所有命题公式对应的表达式二叉树
        """
        self.rules = rules
        self.cnt = 0
        self.LHS = l
        self.RHS = r
        self.nodes = []
        self.class_tree = class_tree_instance
```

acknowledgement

《数理逻辑与集合论》（第二版）