中山大学计算机学院

人工智能

本科生实验报告

课程名称：Artificial Intelligence

学号	23336266	姓名	熊彦钧

一、 实验题目

实现一阶逻辑归结算法

二、 实验内容

1. 算法原理

一阶逻辑归结算法是命题逻辑归结原理的扩展，用于处理包含量词和变量的逻辑公式，相当于把实验三的归结原理和最一般合一算法结合起来，从而处理归结原理中存在的变量替换问题。

2. 关键代码展示

由于篇幅过长，这里仅展示代码的整体思路

在代码中，我们创建了两个类来方便存储和格式化输出

```python
class FirstOrderLogic:      #初始化一阶逻辑原子公式
    def __init__(self, is_negated, predicate, arguments):
        self.is_negated = is_negated #是否被否定
        self.predicate = predicate #谓词名称
        self.arguments = arguments #谓词参数列表
    def __str__(self):      #将一阶逻辑原子公式转换为字符串表示
        return ("" if self.is_negated else "~") + self.predicate + '(' + list_to_str(self.arguments) + ')'
```

```python
class ResolutionStep: #初始化归结步骤
    def __init__(self, is_premise: bool, mgu_dict: dict, parent1_index: int, parent1_atom_index: int, parent2_index: int, parent2_atom_index: int, result_clause=list[FirstOrderLogic]):
        self.mgu_dict = mgu_dict    #最一般合一（MGU）字典
        self.parent1_index = parent1_index + 1  #父句的索引
        self.parent2_index = parent2_index + 1
        self.is_premise = is_premise    #是否是前提（初始子句）
        self.parent1_atom_index = parent1_atom_index    #父句中归结原子的索引
        self.parent2_atom_index = parent2_atom_index
        self.result_clause = result_clause  #归结后的新子句

    def __str__(self):  #将归结步骤转换为字符串表示
        if self.is_premise:
            return f"({FOL_list_to_str(self.result_clause)[:-1]})"
        else:
            mgu_str = "("
            for key, value in self.mgu_dict.items():
                mgu_str += key + "=" + value + " "
            mgu_str = mgu_str[:-1] + ")" if len(self.mgu_dict) > 0 else ""
            result_str = FOL_list_to_str(self.result_clause)[:-1]
            return f"R[{self.parent1_index}{num_to_letter(self.parent1_atom_index) if self.parent1_atom_index >= 0 else ''},{self.parent2_index}{num_to_letter(self.parent2_atom_index) if self.par
```

在进行归结到过程中，我们同样需要对 kb 字符串转化成的列表进行遍历，但在遍历的过程中，由于量词泛化和例化的存在，我们需要判断能不能量词之间能不能统一，即能不能泛化/例化。

三、 实验结果及分析

1. 实验结果展示示例

为了方便，这里把例题和作业 1、2 都放在了同一个程序里面，分别作为输入 kb1,2,3，

并在同一个程序中产生了三个输出。

输入分别如下：

```
kb1= [['GradStudent(sue)',], ['~GradStudent(x)', 'Student(x)'], ['~Student(x)', 'HardWorker(x)'],
   ['~HardWorker(sue)',]]
kb1=convert_list_to_FOL(kb1)

kb2 = [
['A(tony)'],
['A(mike)'],
['A(john)'],
['L(tony,rain)'],
['L(tony,snow)'],
['~A(x)', 'S(x)', 'C(x)'],
['~C(y)', '~L(y,rain)'],
['L(z,snow)', '~S(z)'],
['~L(tony,u)', '~L(mike,u)'],
['L(tony,v)', 'L(mike,v)'],
['~A(w)', '~C(w)', 'S(w)']
]
kb2 = convert_list_to_FOL(kb2)

kb3 =[['On(tony,mike)'],['On(mike,john)'],['Green(tony)'],['~Green(john)'],['~On(xx,yy)','~Green(xx)','Green(yy)']]
kb3 = convert_list_to_FOL(kb3)
```

程序的输出如下：

```
PS C:\Users\jhinx\Desktop\大学课件\人工智能作业> & C:/Python313/python.exe "c:/Users/jhinx/Desktop/大学课件/人工智能作业/3-4 归结原理/第四周作业（归结原理plus）.py"
----------------------------例题: ----------------------------
1:(GradStudent(sue))
2:(~GradStudent(x),Student(x))
3:(~Student(x),HardWorker(x))
4:(~HardWorker(sue))
5:R[1,2a](x=sue) = (Student(sue))
6:R[3a,5](x=sue) = (HardWorker(sue))
7:R[4,6] = ()
----------------------------作业1: ----------------------------
1:(A(tony))
2:(A(mike))
3:(A(john))
4:(L(tony,rain))
5:(L(tony,snow))
6:(~A(x),S(x),C(x))
7:(~C(y),~L(y,rain))
8:(L(z,snow),~S(z))
9:(~L(tony,u),~L(mike,u))
10:(L(tony,v),L(mike,v))
11:(~A(w),~C(w),S(w))
12:R[2,11a](w=mike) = (~C(mike),S(mike))
13:R[5,9a](u=snow) = (~L(mike,snow))
14:R[6c,12a](x=mike) = (~A(mike),S(mike),S(mike))
15:R[8b,14b](z=mike) = (L(mike,snow),~A(mike),S(mike))
16:R[9b,15a](u=snow) = (~L(tony,snow),~A(mike),S(mike))
17:R[2,16b] = (~L(tony,snow),S(mike))
18:R[5,17a] = (S(mike))
19:R[8b,18](z=mike) = (L(mike,snow))
20:R[13,19] = ()
----------------------------作业2: ----------------------------
1:(On(tony,mike))
2:(On(mike,john))
3:(Green(tony))
4:(~Green(john))
5:(~On(xx,yy),~Green(xx),Green(yy))
6:R[2,5a](xx=mike yy=john) = (~Green(mike),Green(john))
7:R[4,6b] = (~Green(mike))
8:R[5c,7](yy=mike) = (~On(xx,mike),~Green(xx))
9:R[1,8a](xx=tony) = (~Green(tony))
10:R[3,9] = ()
PS C:\Users\jhinx\Desktop\大学课件\人工智能作业>
```

经过检验，虽然归结的过程没有和实验的示例输出完全一致，但是结果都得到了空子句，并且过程的逻辑是正确的。猜测的原因可能是加入新子句的位置有区别，导致遍历过程中某两个不同的子句先进行了归结。

三个示例输入的输出均正确，实验任务完成。

四、 参考资料

https://github.com/Adam-226/Resolution-Algorithm/blob/main/AI_lab3.py

本人参考了这个同学的思路，构建了类来处理输入和输出杂乱这一问题。