中山大学计算机学院

人工智能

本科生实验报告

（2023 学年春季学期）

课程名称：Artificial Intelligence

教学班级	**刘咏梅**	专业（方向）	**信息与计算科学**
学号	**22336313**	姓名	**郑鸿鑫**

一、 实验题目

编写程序，实现一阶逻辑归结算法，并用于求解以下两个给出的逻辑推理问题：

二、 实验内容

1. 算法原理

一阶逻辑归结算法的原理是基于逻辑推理的一种证明方法，主要包括以下几个步骤：

1. **合一操作**：找到两个子句中的一个谓词，使得它们的参数列表可以通过一个合一替换操作来统一，即使它们相等。合一是指找到两个谓词之间的替换关系，使得它们可以变得相同。
2. **归结操作**：在找到合一替换关系后，将两个子句进行归结操作，即消解操作，得到一个新的子句。这个新的子句可以作为下一步归结的依据。
3. **重复合一和归结操作**：不断重复进行合一和归结操作，直到得到一个空子句，或者无法再进行合一和归结操作为止。
4. **判断是否推导出目标公式**：如果最终得到一个空子句，则说明原始子句集合是不一致的，即目标公式成立；如果无法得到空子句，则说明无法推导出目标公式。

2. 伪代码

打开文件读取输入

通过正则表达式识别文件中的子句并存储成字符串

将每个字符串形式转换为类似于[On,aa,bb]的格式，存储在列表 s 中

将变量存储在一个列表 Vairable

新建两个空列表 parent 用于存储得到某语句的两个语句和对应的原子公式，assignment

用于存储得到某语句需要的置换操作

遍历第 i 个子句

    遍历第 j 个子句

        I == J ,continue

            遍历第 i 个子句的第 p 个原子公式

遍历第 j 个子句的第 q 个原子公式

If s[i][p]与 s[j][q]的谓词相反且项数相同

If s[i][p]和 s[j][q]后面所有的项都相同

将第 i 和第 j 个子句深拷贝一份

去除谓词

归结后插入 s

更新列表 parent 和 assignment

如果新得到的子句为空 则跳出所有循环

Else 判断能否合一

If 不能合一 continue

Else

将第 i 和第 j 个子句深拷贝一份

去除谓词

归结后插入 s

更新列表 parent 和 assignment

如果新得到的子句为空 则跳出所有循环

回溯归结过程利用 parent 列表和队列寻找所有真正有用的子句存储在列表 useful 中

对这些子句重新编号后按序打印并按照规定格式输出

3. 关键代码展示（带注释）

先对文件读取并进行信息处理，用列表结构存储初始子句：

```
with open('D:\Code\Python\Lab2\input1.txt', 'r', encoding='utf-8') as in_file:
    #打开文件读取输入，并指定编码格式便于打印'¬'
    n = int(in_file.readline())
    s = []
    import re
    for i in range(n):
        s.append(re.findall(r'¬?\w+\(\w+\,*\w*\)',in_file.readline()))
        #通过正则表达式识别文件中的子句并存储成字符串
        for j  in range (len(s[i])):
            s[i][j] = s[i][j].replace('(',',')
            s[i][j] = s[i][j].replace(')','')
            s[i][j] = s[i][j].split(',')
    #利用 replace 函数将每个字符串形式转换为类似于[On,aa,bb]的格式
```

取反函数，实现对任意谓词都取反并返回：

```
def opposite(str):
    if str[0] == '¬':
        return str[1:]
    else:
        return '¬'+str
    #取反函数，以得到一个谓词的取反
```

去重函数：

```
def unique(s_list):
    u_list = list(set(map(tuple, s_list)))
    return u_list
    #去重函数，用于去掉集合中的重复元素
```

求得变量集：

```
variable = []#存储子句中所有出现的变量，用于后续辨别项是变量或常量
for i in range(n):
    for j in range(len(s[i])):
        for k in range(1,len(s[i][j])):
            if len(s[i][j][k]) == 1:
                variable.append(s[i][j][k])
```

solution 函数，用于进行归结过程：

```
parent = []#parent 列表用于存储得到某个子句的两个前驱的子句及其对应的原子公式编号
assignment = []#assignment 用于存储得到某个子句需要进行的替换操作
import copy
def solution(s:list,assignment:list,parent:list):
    for i,si in enumerate(s):#遍历第 i 个子句
        for j,ji in enumerate(s):#遍历第 j 个子句
            if i == j:
                continue
            for p in range(len(s[i])):#遍历第 i 个子句的第 p 个原子公式
                for q in range(len(s[j])):#遍历第 j 个子句的第 q 个原子公式
                    if s[i][p][0] == opposite(s[j][q][0]) and len(s[i][p]) ==
len(s[j][q]):
                        #if 谓词相反且项数相等
                        if s[i][p][1:] == s[j][q][1:]:#if 除了谓词外的其他所有项都相等
                            temp1 = copy.deepcopy(s[i])
                            temp2 = copy.deepcopy(s[j])#深拷贝
                            del temp1[p]
                            del temp2[q]#去除谓词
                            temp = unique(temp1 + temp2)#合并得到归结后的子句
                            parent.append([i,p,j,q])
                            assignment.append([])#更新 parent 和 assignment
                            s.append(temp)#更新 s 列表
                            if temp == []:
                                return #得到空子句则 return
                        else:#仅谓词相反，需进行进一步判断
                            judge = [] #二元组前一个为常量，后一个为变量
                            for m in range(1,len(s[i][p])):
                                if s[i][p][m] not in variable and s[j][q][m] not in
variable and s[i][p][m] == s[j][q][m]:
                                    continue #都为常量且相同
                                elif s[i][p][m] not in variable and s[j][q][m] in
variable:
                                    judge.append((s[i][p][m],s[j][q][m]))#前一个为常
量，后一个为变量，则可合一，记录置换操作
                                else:
                                    judge = False#无法合一
                                    break
                            if judge == False:
                                continue
                            else:
```

```
                                temp1 = copy.deepcopy(s[i])
                                temp2 = copy.deepcopy(s[j])
                                del temp1[p]
                                del temp2[q]
                                for m in range(len(judge)):#对每一次置换操作
                                    for k in range(len(temp2)):#遍历整个子句
                                        while judge[m][1] in temp2[k]:
                                            index = temp2[k].index(judge[m][1])
                                            temp2[k] = list(temp2[k])
                                            temp2[k].remove(judge[m][1])
                                            temp2[k].insert(index,judge[m][0])
                                            temp2[k] = tuple(temp2[k])
                                temp = unique(temp1 + temp2)
                                parent.append([i,p,j,q])
                                assignment.append(judge)
                                s.append(temp)
                                if temp == []:
                                    return
```

useful_sentence 函数，用于回溯有用的子句：

```
def useful_senten(assignment:list,parent:list):#回溯挑选有用的子句，并返回相关的所有信息
    useful = []
    q = queue.Queue()
    q.put(parent[-1])
    useful.append([s[-1],parent[-1],assignment[-1]])#将得到空子句的前驱和置换操作存储
    while not q.empty():
        pre = q.get()
        if pre[0] >= n:#若得到此子句的第一个子句不是原来的子句，则需要存储
            useful.append([s[pre[0]],parent[pre[0]-n],assignment[pre[0]-n]])
            q.put(parent[pre[0]-n])
        if pre[2] >= n:#若得到此子句的第二个子句不是原来的子句，则需要存储
            useful.append([s[pre[2]],parent[pre[2]-n],assignment[pre[2]-n]])
            q.put(parent[pre[2]-n])
    return useful
```

实例化上述两个函数，并对 uu 进行排序和重新编号处理：

```
solution(s,assignment,parent)#通过 solution 求得归结过程中的所有信息
uu = useful_senten(assignment,parent)#简化 solution 中得到的信息，挑选有用的
reindex = dict()#对这些有用的子句进行重新编号
for i in range(n):
    reindex[i] = None#文件中提供的原先有的子句
for i,ui in enumerate(uu):
    if ui[1][0] not in reindex:#第一个前驱子句还没进入字典
        reindex[ui[1][0]] = None
    if ui[1][2] not in reindex:#第二个前驱子句还没进入字典
        reindex[ui[1][2]] = None
reindex = sorted(reindex.keys())
reindex = {x:reindex.index(x)+1 for x in reindex}
def takeindex(u:list, s:list):
    return s.index(u[0])  # 返回 u 中第一个元素在 s 中的索引位置
uu.sort(key=lambda x: takeindex(x, s))#利用索引排序，先归结得到的子句放置在前
```

打印函数，按照格式进行打印输出：

```
for i in uu:#按照输出限制打印输出
    par = i[1]
    ass = i[2]
    ss = i[0]
    index1 = reindex[par[0]]
    index2 = reindex[par[2]]
    ind1 = chr(par[1] + 97) if len(s[par[0]]) > 1 else ''
    ind2 = chr(par[3] + 97) if len(s[par[2]]) > 1 else ''
    print('R[',index1,ind1,',',index2,ind2,']',sep = '',end = '')
    for j in ass:
```

```
        if len(j) != 0:
            print('(',j[1],'=',j[0],')',sep = '',end = '')
    print(' =',end = ' ')
    if len(ss) == 0:
        print("[]")#归结结束打印空
    for j in ss:
        for k in range(len(j)):
            if k == 0:
                print(j[0],'(',sep='',end = '')
            elif k == len(j)-1:
                print(j[k],')',sep='',end = ',')
            else:
                print(j[k],',',sep='',end = '')
    print()
input("请按下任意键继续...")
```

三、 实验结果及分析

1. 实验结果展示示例

求解逻辑推理问题 1（Alpine Club）：
文件输入如下：

Code > Python > Lab2 > input1.txt

```
11
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))
```

程序运行得到归结过程如下

```
D:\Code\Python\Lab2\project.py:1: SyntaxWarning: invalid escape sequence '\C'
  with open('D:\Code\Python\Lab2\input1.txt', 'r', encoding='utf-8') as in_file:
R[2,6a](x=mike) = S(mike),C(mike),
R[2,11a](w=mike) = S(mike),¬C(mike),
R[5,9a](u=snow) = ¬L(mike,snow),
R[12b,13b] = S(mike),
R[14,8a](z=mike) = ¬S(mike),
R[15,16] = []

请按下任意键继续...
```

求解逻辑推理问题 2（Block World）：
文件输入如下：

Code > Python > Lab2 > input2.txt

```
5
On(aa,bb)
On(bb,cc)
Green(aa)
¬Green(cc)
(¬On(x,y),¬Green(x),Green(y))
```

程序运行得到归结过程如下

```
D:\Code\Python\Lab2\project.py:1: SyntaxWarning: invalid escape sequence '\C'
  with open('D:\Code\Python\Lab2\input2.txt', 'r', encoding='utf-8') as in_file:
R[1,5a](x=aa)(y=bb) = ¬Green(aa),Green(bb),
R[2,5a](x=bb)(y=cc) = Green(cc),¬Green(bb),
R[3,6a] = Green(bb),
R[4,7a] = ¬Green(bb),
R[8,9] = []

请按下任意键继续...
```

四、 参考资料

1. 【人工智能】归结演绎推理-CSDN 博客
2. 【人工智能导论】实验二 一阶逻辑归结算法 - 哔哩哔哩 (bilibili.com)