中山大学计算机学院

人工智能

本科生实验报告

（2021 学年春季学期）

课程名称：Artificial Intelligence

教学班级	网安软工合班	专业（方向）	网络空间安全
学号	20337251	姓名	伍建霖

一、 实验题目

编写程序，实现一阶逻辑归结算法，并用于求解给出的三个逻辑推理问题，要求输出按照如下格式：

1. (P(x),Q(g(x)))
2. (R(a),Q(z),¬P(a))
3. R[1a,2c]{X=a} (Q(g(a)),R(a),Q(z))

… …

“R” 表示归结步骤.

“1a” 表示第一个子句(1-th)中的第一个 (a-th)个原子公式，即P(x).

“2c”表示第二个子句(1-th)中的第三个 (c-th)个原子公式，即¬P(a).

“1a”和“2c”是冲突的，所以应用最小合一{X = a}.

二、 实验内容

1. 算法原理

一阶逻辑归结原理：

每次归结都从两个子句中寻找一对互为否定的原子，判断是否需要合一，若不需要合一则去掉该对原子并将两个子句合为一个子句，若需要合一则调用合一算法判断能否合一，若需要但不能合一则 pass，若能合一则得出变量的替换，将两个子句对应的变量替换后合并成一个子句

合一算法：

依次取出两个原子公式的项，若存在 a 项是 b 项的一部分的情况则合一失败，若不存在该情况则用不是变量的项替换是变量的项并将此次替换记录进替换集，若两项都不是变量也合一失败

2. 伪代码

```
读入数据,存在 S 中
开始归结：
    将 S 以 len 为 key 进行升序排序
```

```
输出现有 S
循环 10 次:
    将 S 以 len 为 key 进行升序排序
    for first in S:
        若 first 的下标大于两倍的 len(S):
            break # sort 一下再进行归并
        for second in S:
            若 S 中存在空列表:
                归结成功
            若 first 和 second 都是原子公式:
                若互为否定:
                    若需要合一:
                        若不能合一:
                            pass
                        若能合一:
                            S 中添加一个空列表 # 此时已归结成功
                    否则:
                        S 中添加一个空列表 # 此时已归结成功
                否则 pass
            若 first 和 second 中一个是原子公式一个是子句:
                遍历子句中的原子公式
                    若和原子公式互为否定:
                        若需要合一:
                            若不能合一:
                                pass
                            若能合一:
                                将(变量替换且
                                    删除原子公式的否定)的子句加入 S
                        否则:
                            将删除原子公式的否定后的子句加入 S
                    否则 pass
            若 first 和 second 都是子句:
                遍历两个子句:
                    若存在一对互为否定的原子公式:
                        若需要合一:
                            若不能合一:
                                pass
                            若能合一:
                                将(变量替换且删除原子公式的否定
                                    且合并了的)的子句加入 S
                        否则:
                            将删除原子公式的否定且合并后的子句加入 S
                    否则 pass
```

```
    输出 fail,归结失败
```

3. 关键代码展示（带注释）

合一算法:解释见于伪代码中的相应部分

```
def MGU(a: 'dict', b: 'dict') -> "bool or dict":
    valueset = {}
    ainfunc = copy.deepcopy(a)
    binfunc = copy.deepcopy(b)
    alist = [x for x in ainfunc.values()] # [['x','y']]
    blist = [x for x in binfunc.values()] # [['tony','mike']]
    if alist == blist:
        # ab same
        # 即不需要合一
        return true
    else:
        for x in range(len(alist[0])):
            if (alist[0][x] in blist[0][x]) and\
                (alist[0][x] != blist[0][x]):
                # 若其中一项被包含在另一项中,则无法合一
                break
            else:
                if alist[0][x] == blist[0][x]:
                    # 该项不需要替换
                    pass
                elif (len(alist[0][x]) > 1 and len(blist[0][x]) > 1) or\
                     (len(alist[0][x]) == 1 and len(blist[0][x]) == 1):
                    # haven't solve f(g(y)) - f(u)
                    # 两项都不是变量 or 两项都是变量
                    return false
                elif len(alist[0][x]) > len(blist[0][x]):
                    # 用不是变量的项替换是变量的项
                    valueset[blist[0][x]] = alist[0][x]
                    blist[0][x] = alist[0][x]
                else:
                    # 用不是变量的项替换是变量的项
                    valueset[alist[0][x]] = blist[0][x]
                    alist[0][x] = blist[0][x]
    if alist != blist:  # MGU Error
        return false
    return valueset
```

归结算法:解释见于伪代码中的相应部分

```
def resolution():
    global S
    S.sort(key=lambda i: len(i))
```

```
print("---------------------")
for x in S:
    print(x, S.index(x)+1)
print("-------开始归结-------")
for ccc in range(10):
    S.sort(key=lambda i: len(i))
    print("-------sorted--------")
    longest = len(S[-1])-1
    length = len(S)
    havenewclause = 0
    for first in S:
        if S.index(first) > length*2:
            break
        for second in S:
            if [] in S:
                return
            if (len(first) == 1 and len(second) == 1):  # set - set
                skey = list(second[0].keys())
                if (fanyi(skey[0]) == list(first[0].keys())):
                    newclause = MGU(first[0], second[0])
                    if newclause == true:
                        havenewclause = 1
                        S.append([])
                        # 输出新生成的子句
                        print('R[', S.index(first)+1, ',',
                              S.index(second)+1, ']',
                              end='', sep='')
                        judge(newclause, havenewclause)
                        # 输出完毕
                        havenewclause = 0
                        pass
                    elif newclause == false:
                        pass
                    else:
                        havenewclause = 1
                        S.append([])
                        # 输出新生成的子句
                        print('R[', S.index(first)+1, ',',
                              S.index(second)+1, ']',
                              end='', sep='')
                        judge(newclause, havenewclause)
                        # 输出完毕
                        pass
                pass
```

```
            elif (len(first) == 1 and len(second) > 1):  # sample - set
                for x in range(len(second)):
                    skey = list(second[x].keys())
                    if (fanyi(skey[0]) == list(first[0].keys())):
                        newclause = MGU(first[0], second[x])
                        if newclause == true:
                            havenewclause = 1
                            temp = copy.deepcopy(second)
                            del temp[x]
                            if temp in S:
                                pass
                            elif len(temp) > longest:
                                pass
                            else:
                                S.append(temp)
                                # 输出新生成的子句
                                print('R[', S.index(first)+1, ',',
                                      S.index(second)+1, ']',
                                      end='', sep='')
                                judge(newclause, havenewclause)
                                # 输出完毕
                            havenewclause = 0
                        elif newclause == false:
                            pass
                        else:
                            havenewclause = 1
                            temp = copy.deepcopy(second)
                            del temp[x]
                            # 替换变量
                            for y in range(len(temp)):
                                for z in temp[y].values():
                                    for w in newclause.keys():
                                        if w in z:
                                            l = list(temp[y].values())
                                            for v in range(len(l[0])):
                                                if w == z[v]:
                                                    t = list(
                                                        temp[y].keys())
                                                    temp[y][t[0]
                                                            ][v] = newclause[w]
                                        else:
                                            pass
                            if temp in S:
                                pass
```

```
                        elif len(temp) > longest:
                            pass
                        else:
                            S.append(temp)
                            # 输出新生成的子句
                            print('R[', S.index(first)+1, ',',
                                  S.index(second)+1, chr(97+x), ']',
                                  end='', sep='')
                            judge(newclause, havenewclause)
                            # 输出完毕
                        havenewclause = 0
                else:
                    pass
            pass
        elif (len(first) > 1 and len(second) == 1):  # set - sample
            # this situation will be reached in elif2
            firstcopy = copy.deepcopy(first)
            secondcopy = copy.deepcopy(second)
            for x in range(len(secondcopy)):
                skey = list(secondcopy[x].keys())
                if (fanyi(skey[0]) == list(firstcopy[0].keys())):
                    newclause = MGU(firstcopy[0], secondcopy[x])
                    if newclause == true:
                        havenewclause = 1
                        temp = copy.deepcopy(firstcopy)
                        del temp[x]
                        if temp in S:
                            pass
                        elif len(temp) > longest:
                            pass
                        else:
                            S.append(temp)
                            # 输出新生成的子句
                            print('R[', S.index(first)+1, ',',
                                  S.index(second)+1, ']',
                                  end='', sep='')
                            judge(newclause, havenewclause)
                            # 输出完毕
                        havenewclause = 0
                    elif newclause == false:
                        pass
                    else:
                        havenewclause = 1
                        temp = copy.deepcopy(firstcopy)
```

```
                        del temp[x]
                        # 替换变量
                        for y in range(len(temp)):
                            for z in temp[y].values():
                                for w in newclause.keys():
                                    if w in z:
                                        l = list(temp[y].values())
                                        for v in range(len(l[0])):
                                            if w == z[v]:
                                                t = list(
                                                    temp[y].keys())
                                                temp[y][t[0]
                                                        ][v] = newclause[w]
                                    else:
                                        pass
                        if temp in S:
                            pass
                        elif len(temp) > longest:
                            pass
                        else:
                            S.append(temp)
                            # 输出新生成的子句
                            print('R[', S.index(first)+1, ',',
                                  S.index(second)+1, chr(97+x), ']',
                                  end='', sep='')
                            judge(newclause, havenewclause)
                            # 输出完毕
                        havenewclause = 0
                else:
                    pass
            pass
        else:  # set - set
            for x in range(len(first)):
                for y in range(len(second)):
                    fkey = list(first[x].keys())
                    if (fanyi(fkey[0]) == list(second[y].keys())):
                        newclause = MGU(first[x], second[y])
                        if newclause == false:
                            pass
                        elif newclause == true:
                            # 合并并删除对应原子
                            firstcopy = copy.deepcopy(first)
                            secondcopy = copy.deepcopy(second)
                            del firstcopy[x]
```

```
                                del secondcopy[y]
                                temp = firstcopy + secondcopy
                                if temp in S:
                                    pass
                                elif len(temp) > longest:
                                    pass
                                else:
                                    havenewclause = 1
                                    S.append(temp)
                                    # 输出新生成的子句
                                    print('R[', S.index(first)+1, chr(97+x), ',',
                                          S.index(second) +
                                          1, chr(97+y), ']',
                                          end='', sep='')
                                    judge(newclause, havenewclause)
                                    # 输出完毕
                                havenewclause = 0
                                pass
                            else:
                                # 合并并删除对应原子
                                firstcopy = copy.deepcopy(first)
                                secondcopy = copy.deepcopy(second)
                                del firstcopy[x]
                                del secondcopy[y]
                                temp = firstcopy + secondcopy
                                # 替换变量
                                for xx in range(len(temp)):
                                    for z in temp[xx].values():
                                        for w in newclause.keys():
                                            if w in z:
                                                l = list(temp[xx].values())
                                                for v in range(len(l[0])):
                                                    if w == z[v]:
                                                        t = list(
                                                            temp[xx].keys())
                                                        temp[xx][t[0]
                                                                 ][v] =
newclause[w]
                                            else:
                                                pass
                                if temp in S:
                                    pass
                                elif len(temp) > longest:
                                    pass
```

```
                                else:
                                    havenewclause = 1
                                    S.append(temp)
                                    # 输出新生成的子句
                                    print('R[', S.index(first)+1, chr(97+x), ',',
                                          S.index(second) +
                                          1, chr(97+y), ']',
                                          end='', sep='')
                                    judge(newclause, havenewclause)
                                    # 输出完毕
                                havenewclause = 0
                pass
    print("--------fail---------")
    print("---this is the end---")
```

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

```
[{'On': ['aa', 'bb']}] 1
[{'On': ['bb', 'cc']}] 2
[{'Green': ['aa']}] 3
[{'!Green': ['cc']}] 4
[{'!On': ['x', 'y']}, {'!Green': ['x']}, {'Green': ['y']}] 5
-------开始归结-------
-------sorted--------
R[1,5a](x=aa,y=bb) = !Green(aa) Green(bb)
R[2,5a](x=bb,y=cc) = !Green(bb) Green(cc)
R[3,5b](x=aa) = !On(aa,y) Green(y)
R[3,6] = Green(bb)
R[4,5c](y=cc) = !On(x,cc) !Green(x)
R[4,7] = !Green(bb)
R[4,8b](y=cc) = !On(aa,cc)
R[6b,7a] = !Green(aa) Green(cc)
R[6a,8b](y=aa) = Green(bb) !On(aa,aa)
R[6b,10b](x=bb) = !Green(aa) !On(bb,cc)
R[7a,6b] = Green(cc) !Green(aa)
R[7a,8b](y=bb) = Green(cc) !On(aa,bb)
R[7,9] = Green(cc)
R[7b,10b](x=cc) = !Green(bb) !On(cc,cc)
R[7a,14a] = Green(cc) !On(aa,aa)
R[8b,6a](y=aa) = !On(aa,aa) Green(bb)
R[8b,7a](y=bb) = !On(aa,bb) Green(cc)
R[8b,13a](y=aa) = !On(aa,aa) Green(cc)
R[8b,15a](y=aa) = !On(aa,aa) !On(bb,cc)
R[8b,19a](y=bb) = !On(aa,bb) !On(cc,cc)
R[9,5b](x=bb) = !On(bb,y) Green(y)
R[9,10b](x=bb) = !On(bb,cc)
R[9,11] = []
PS D:\CodeField\AI\lab>
```

```
----------------------
[{'GradStudent': ['sue']}] 1
[{'!HardWorker': ['sue']}] 2
[{'!GradStudent': ['x']}, {'Student': ['x']}] 3
[{'!Student': ['x']}, {'HardWorker': ['x']}] 4
-------开始归结-------
-------sorted--------
R[1,3a](x=sue) = Student(sue)
R[2,4b](x=sue) = !Student(sue)
R[4,5a](x=sue) = HardWorker(sue)
R[5,6] = []
PS D:\CodeField\AI\lab>
```

```
R[15,71] = S(mike)
R[15a,72a] = S(mike) !S(tony)
R[15a,74a] = S(mike) C(tony)
R[15a,75a] = S(mike) !C(tony)
R[15a,76a] = S(mike) S(tony)
R[16b,6a](y=john) = S(john) !L(john,rain)
R[16a,7b](z=john) = C(john) L(john,snow)
R[16b,17a] = S(john) S(john)
R[16b,31b] = S(john) L(john,snow)
R[17b,7b](z=john) = !C(john) L(john,snow)
R[18,10c](x=tony) = !A(tony) S(tony)
R[18,12] = S(tony)
R[18,80] = !C(mike)
R[21b,8a](u=rain) = !C(mike) !L(mike,rain)
R[21a,72a] = L(tony,rain) !S(tony)
R[21a,74a] = L(tony,rain) C(tony)
R[21a,75a] = L(tony,rain) !C(tony)
R[21a,76a] = L(tony,rain) S(tony)
R[22b,7b](z=mike) = !L(mike,rain) L(mike,snow)
R[22b,25a] = !L(mike,rain) !L(tony,snow)
R[22b,35a] = !L(mike,rain) !S(tony)
R[22b,58b] = !L(mike,rain) C(tony)
R[22b,63b] = !L(mike,rain) !C(tony)
R[22b,64b] = !L(mike,rain) S(tony)
R[23b,7b](z=john) = !L(john,rain) L(john,snow)
-------sorted--------
R[9,11] = []
PS D:\CodeField\AI\lab>
```

由于我采用的是遍历归结所以得出的项较多，尽管使用了一些策略，如每隔一段时间 sort 一下以便能优先归并原子公式，但性能依旧不是很理想；同时还存在一个问题：由于我是用列表中存字典的方式来存 S 的，所以存在重复的问题(有解决的思路：利用集合的无序无重的特点)

四、 参考资料

用 Python 实现命题逻辑归结推理系统--人工智能_索儿呀的博客-CSDN 博客_python 逻辑推理