**中山大学计算机学院**

**人工智能**

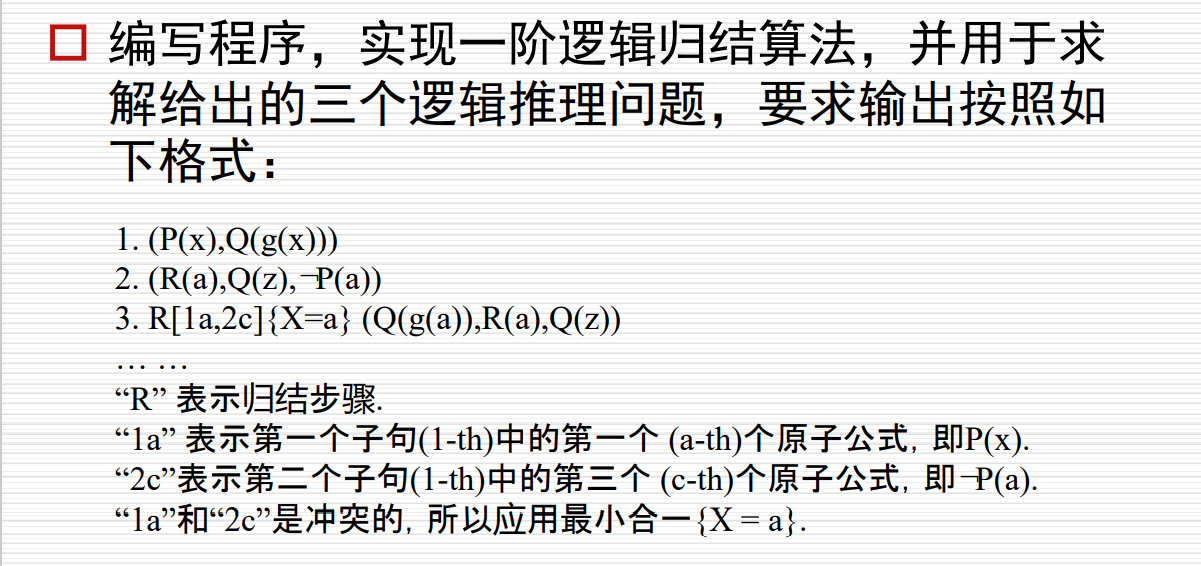
**本科生实验报告**

**（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** |

# 实验题目



# 实验内容

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

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

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

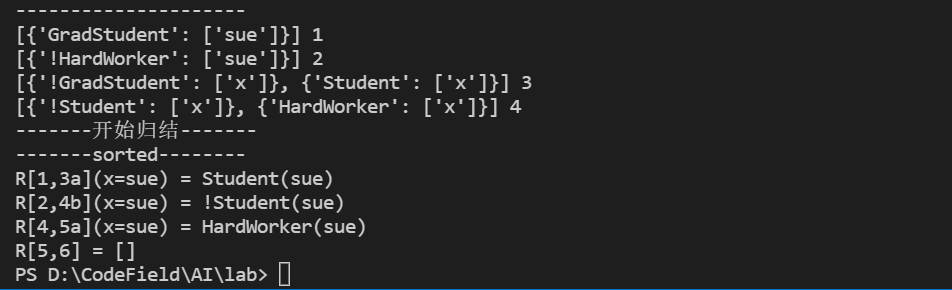
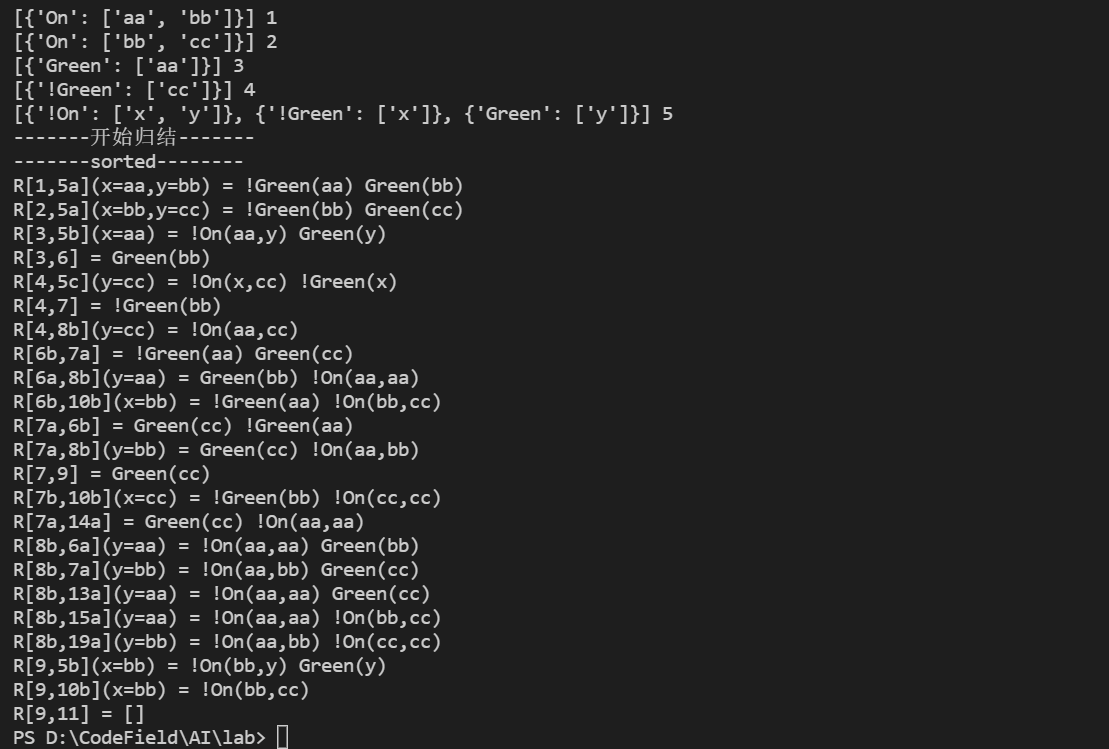
                    pass

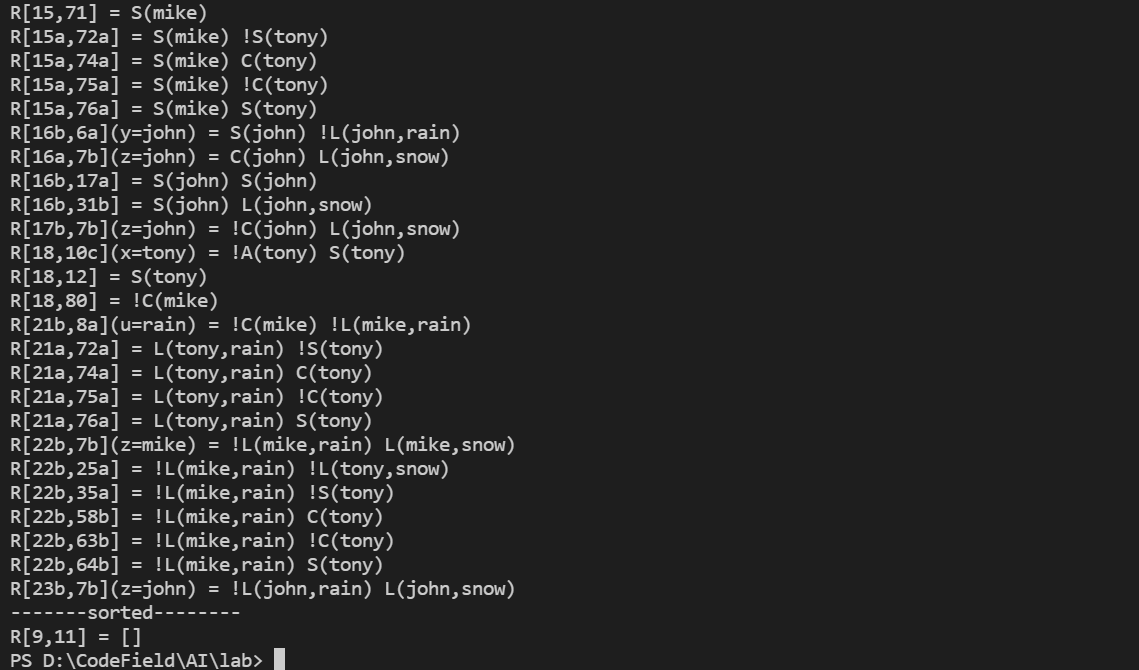
    print("--------fail---------")

    print("---this is the end---")

# 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）





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)

# 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](https://blog.csdn.net/Zhangguohao666/article/details/105471307)