Họ và tên: Lê Phạm Lan Anh

MSSV: 19120447

BÁO CÁO ĐỒ ÁN LAB 2 – LOGIC

Môn: Cơ sở trí tuệ nhân tạo

1. Giải thích code

Sau đây là hàm main của chương trình với các nhiệm vụ: đọc và lưu dữ liệu đầu vào trong file inputk.txt, gọi hàm PL_Resolution để thực hiện thuật toán hợp giải và ghi dữ liệu vào file đầu outputk.txt.

```
if __name__=="__main__":
    # Khai báo biến output là biến toàn cục để chứa các lần hợp giải
    global output
    # Lấy tên các file inputk.txt trong thư mục input và lưu vào file_list
    file_list = GetFiles(os.path.dirname(os.path.abspath(__file__)) + "\input")
    # Duyệt qua từng file input trong file_list
    for file in range(1, len(file_list) + 1):
        # Khởi tạo output là list rỗng
        output = []
        ####### Đọc dữ liệu đầu vào ##############################################
        inFile = open("input\\input" + str(file) + ".txt", "r")
        a = inFile.readline() # câu alpha
        n = int(inFile.readline()) # số mệnh đề trong KB
        kb = [] # KB
        # đọc các mệnh đề trong KB
        for i in range(n):
            kb.append(inFile.readline())
        inFile.close()
        ##########################################################################
        ####### Gọi hàm PL_Resolution để thực hiện hợp giải ####################
        flag = PL_Resolution(kb, a)
        ##########################################################################
        ####### Ghi dữ liệu đầu ra ###############################################
        outFile = open("output\\output" + str(file) + ".txt", "w")
        for i in output: # i là tập các mệnh đề trong một vòng lặp
            outFile.write(str(len(i)) + "\n") # ghi số mệnh đề trong vòng lặp i
            # ghi ra các mệnh đề trong i
            for j in i:
                tmp = " OR ".join(j)
                outFile.write(tmp + "\n")
```

```
        if flag: outFile.write("YES") # ghi "YES" nếu KB entails a
        else: outFile.write("NO") # ghi "NO" nếu ngược lại
        outFile.close()
        ###########################################################################
```

Để lấy được tên các file trong thư mục input, em gọi hàm GetFiles(path) như sau:

```
import os

def GetFiles(path):
    file_list = []
    for dir, subdirs, files in os.walk(path):
        file_list.extend([f for f in files])
    return file_list
```

Hàm PL_Resolution(kb, a) dùng để thực hiện hợp giải và trả về True nếu KB entails α hoặc False nếu ngược lại. Trong hàm này, em dùng cấu trúc dữ liệu set để phân biệt giữa các mệnh đề giống nhau. Gồm các biến cần quan tâm như sau:

- clauses: là tập chứa các mệnh đề có sẵn trong KB, phủ định của α và các mệnh đề phát sinh trong quá trình hợp giải.
- flag: là biến boolean đánh dấu kết quả trả về tương ứng True – KB entails α, False – KB không entails α.
- output: là list các lần lặp, mỗi lần lặp chứa tập mệnh đề mới phát sinh, tập này phục vụ cho việc ghi ra file outputk.txt.
- new: là tập chứa các mệnh đề phát sinh trong một vòng lặp
- resolvents: là mệnh đề được hợp giải từ i và j

```
def PL_Resolution(kb, a):
    global output
    # n là số mệnh đề trong KB
    n = len(kb)
    # clauses là tập các mệnh đề có sẵn và phát sinh trong quá trình hợp giải
    clauses = set()
    # Chuẩn hóa a thành một list chứa các literal trong mệnh đề
    a = Standardize(a.replace(" ", "").replace("\n", "").split("OR"))
    # Nếu a là mệnh đề mang giá trị True thì trả về True
    if a[0] == 1: return True
    # Phủ định mệnh đề a và đưa vào clauses
    for i in a:
```

```
        if len(i) == 2: clauses.add(tuple([i[-1]]))
        else: clauses.add(tuple(["-" + i]))
    # Đưa các mệnh đề trong KB vào clauses
    for i in range(n):
        # Chuẩn hóa mệnh đề i thành một list chứa các literal
        kb[i] = Standardize(kb[i].replace(" ", "").replace("\n", "").split("OR"))
        # Đưa mệnh đề i vào clauses
        if kb[i][0] != 1: clauses.add(tuple(kb[i]))
    # n là số mệnh đề trong clauses và flag là biến đánh dấu kết quả trả về
    n, flag = len(clauses), False
    while True:
        # new là tập các mệnh đề phát sinh trong vòng lặp này
        new = set()
        # Duyệt qua các cặp mệnh đề i, j trong clauses
        for i in clauses:
            for j in clauses:
                resolvents = PL_Resolve(i, j)# resolvents là kết quả i hợp giải j
                if len(resolvents) == 0: # nếu resolvents là rỗng
                    flag = True # đánh dấu flag = True (tức KB entails a)
                    new.add(tuple(['{}'])) # Thêm {} vào new
                # Nếu resolvents là mệnh đề True hoặc không nằm trong clauses thì
                   thêm nó vào new
                elif(resolvents[0] != -1 and (tuple(resolvents) not in clauses)):
                    new.add(tuple(resolvents))
        # Thêm new vào output
        output.append(new)
        # Nếu không phát sinh thêm mệnh đề nào sau lần lặp này thì trả về False
        if not(flag) and len(new) == 0: return False
        # Thêm new vào tập clauses
        clauses = clauses | new
        # Nếu flag == True thì trả về True
        if flag: return True
```

Hàm Standardize(s) giúp chuẩn hóa mệnh đề s sao cho không có các literal trùng nhau, sắp xếp theo thứ tự chữ cái và trả về 1 nếu tồn tại 2 literal đối ngẫu hợp nhau (vì tồn tại 2 literal đối ngẫu sẽ là mệnh đề True, không giúp ích cho việc hợp giải nên sẽ bị bỏ qua)

```
def Standardize(s):
    # Loại bỏ các literal trùng nhau trong s
    s = list(set(s))
    # Sắp xếp các literal trong s theo bảng chữ cái
    s = sorted(s, key = lambda x: x[-1])
    # Duyệt qua các literal để tìm ra có 2 literal đối ngẫu thì trả về 1 (True)
```

```
    for i in range(1, len(s)):
        if s[i - 1][-1] == s[i][-1]: return [1]
    # Trả về s được chuẩn hóa
    return s
```

Để hợp giải c1 với c2, em gọi hàm PL_Resolve(c1, c2). Hàm này sẽ xem có tồn tại từ 2 cặp đối ngẫu trong c1 và c2 không. Nếu tồn tại thì khi hợp giải nó sẽ ra True nên cần loại bỏ vì không cần thiết. Nếu không tồn tại sẽ gọi hàm Union để trả về kết quả hợp giải.

```
def PL_Resolve(c1, c2):
    # Nếu c1 và c2 chứa từ 2 literal đối ngẫu nhau thì khi hợp giải sẽ ra True
    # cont là biến chứa các literal đối ngẫu trong c1 và c2 (lưu 1 biến dưới dạng dương)
    cont = []
    # Duyệt hết các cặp literal trong c1 và c2
    for i in c1:
        for j in c2:
            # Nếu i và j đối ngẫu thì thêm vào cont
            if len(i) != len(j) and i[-1] == j[-1]: cont.append(i[-1])
    # Nếu cont chứa từ 2 đối ngẫu thì trả về -1 đánh dấu là mệnh đề True không cần xét
    if len(cont) != 1: return [-1]
    # Ngược lại thì trả về hợp giải c1 và c2 bằng cách lấy c1 hợp c2 hiệu với các literal đối ngẫu
    else: return Union(c1, c2, cont[0])
```

Hàm Union(c1, c2, c3) dùng để hỗ trợ cho hàm PL_Resolve(c1, c2) trong việc hợp giải c1 và c2 biết c3 là literal dương đối ngẫu. Hàm trả về list các literal trong mệnh đề kết quả hợp giải:

```
def Union(c1, c2, c3):
    # Hợp c1 và c2
    res = list(set(c1) | set(c2))
    # Bỏ đi literal đối ngẫu
    res.remove(c3)
    res.remove("-"+c3)
    # Sắp xếp các literal theo bảng chữ cái
    res = sorted(res, key = lambda x: x[-1])
    # Trả kết quả hợp giải
    return res
```

2. Giải thích test case

STT	Input	Output	Giải thích
1	-A OR B 4 -A OR C OR E B OR -C -A OR D C OR -E OR -I	6 B OR -E OR -I -A OR B OR E -C -A OR C OR -I D C OR E 9 C OR -I B OR E -A OR E -A OR B OR C OR -I -A OR -I E -A OR B OR -I -E OR -I B OR C OR -I 2 B OR -I -I 0 NO	 (C OR -E OR -I) hợp giải với (B OR -C) (-A OR C OR E) hợp giải với (B OR -C) (-B) hợp giải với (B OR -C) (C OR -E OR -I) hợp giải với (-A OR C OR E) (A) hợp giải với (-A OR D) (A) hợp giải với (-A OR C OR E) (A) hợp giải với (-A OR C OR -I) (A) hợp giải với (-A OR B OR E) (-A OR B OR E) hợp giải với (-B) (B OR -E OR -I) hợp giải với (-A OR C OR E) (-C) hợp giải với (-A OR C OR -I) (-C) hợp giải với (C OR E) (B OR -E OR -I) hợp giải với (-A OR B OR E) (B OR -E OR -I) hợp giải với (-B) (B OR -E OR -I) hợp giải với (C OR E) (C OR -I) hợp giải với (B OR -C) (C OR -I) hợp giải với (-C)
2	-A OR B 5 -A OR D OR E -D J -J OR I -E OR -I	5 -E OR -J D OR E -A OR D OR -I -A OR E I 8 -A OR -J -E D OR -I D OR -J -A OR D OR -J -A OR -I E -A OR D 5 -J -A -I D {}	 (I OR -J) hợp giải với (-E OR -I) (A) hợp giải với (-A OR D OR E) (-A OR D OR E) hợp giải với (-E OR -I) (-D) hợp giải với (-A OR D OR E) (I OR -J) hợp giải với (J) (-E OR -J) hợp giải với (-A OR E) (-E OR -J) hợp giải với (J) (D OR E) hợp giải với (-E OR -I) (-E OR -J) hợp giải với (D OR E) (-E OR -J) hợp giải với (-A OR D OR E) (-D) hợp giải với (-A OR D OR -I) (-D) hợp giải với (D OR E) (-A OR D OR -I) hợp giải với (I) (-E OR -J) hợp giải với (E) (I) hợp giải với (-A OR -I) (-A OR -I) hợp giải với (A) (D OR E) hợp giải với (-E) (-E) hợp giải với (E)

		YES	
3	-A 5 -A OR B OR -C -D OR E C OR -E -B OR -D D	6 -A OR B OR -E -B C OR -D -A OR -C OR -D E B OR -C 11 -A OR B OR -D -C OR -D -A OR -D -A OR -E B OR -D C -C -A OR -C -A OR B -A OR -D OR -E B OR -E 6 -D -E -A -D OR -E {} B YES	 (C OR -E) hợp giải với (-A OR B OR -C) (D) hợp giải với (-B OR -D) (-D OR E) hợp giải với (C OR -E) (-B OR -D) hợp giải với (-A OR B OR -C) (-D OR E) hợp giải với (D) (A) hợp giải với (-A OR B OR -C) (-A OR B OR -E) hợp giải với (-D OR E) (A) hợp giải với (-A OR -C OR -D) (C OR -D) hợp giải với (-A OR -C OR -D) (-A OR B OR -E) hợp giải với (-B) (C OR -D) hợp giải với (B OR -C) (C OR -D) hợp giải với (D) (-B) hợp giải với (B OR -C) (-B) hợp giải với (-A OR B OR -C) (-A OR B OR -E) hợp giải với (E) (-A OR B OR -E) hợp giải với (-B OR -D) (-A OR B OR -E) hợp giải với (A) (-C OR -D) hợp giải với (C OR -D) (-B) hợp giải với (B OR -E) (-B) hợp giải với (-A OR B) (-C OR -D) hợp giải với (C OR -E) (-C) hợp giải với (C) (D) hợp giải với (B OR -D)
4	-A 6 -A OR D OR E B OR -C -D OR E OR -I -E OR F I D	6 D OR E E OR -I -A OR D OR F -D OR E -D OR F OR -I -A OR E OR -I 9 -A OR E -D OR F -A OR E OR F F OR -I -A OR E OR F OR -I E D OR F E OR F OR -I -A OR F OR -I 3	 (A) hợp giải với (-A OR D OR E) (-D OR E OR -I) hợp giải với (D) (-E OR F) hợp giải với (-A OR D OR E) (-D OR E OR -I) hợp giải với (I) (-E OR F) hợp giải với (-D OR E OR -I) (-D OR E OR -I) hợp giải với (-A OR D OR E) (I) hợp giải với (-A OR E OR -I) (-E OR F) hợp giải với (-D OR E) (-A OR D OR F) hợp giải với (-D OR E) (-E OR F) hợp giải với (E OR -I) (-D OR E OR -I) hợp giải với (-A OR D OR F) (D OR E) hợp giải với (-D OR E) (-E OR F) hợp giải với (D OR E) (D OR E) hợp giải với (-D OR F OR -I) (-E OR F) hợp giải với (-A OR E OR -I)

		F -A OR F E OR F 0 NO	(-E OR F) hợp giải với (E) (-E OR F) hợp giải với (-A OR E) (D OR E) hợp giải với (-D OR F)
5	-A OR B OR E 6 D OR E OR F B OR -C -A OR -E I C OR -E OR -I -D	7 B OR -E OR -I -A OR D OR F -C D OR F C OR D OR F OR -I E OR F C OR -E 10 D OR F OR -I F B OR D OR F OR -I C OR F OR -I -A OR F B OR F OR -I C OR F C OR D OR F -E OR -I B OR -E 3 F OR -I B OR D OR F B OR F 0 NO	 (C OR -E OR -I) hợp giải với (B OR -C) (-A OR -E) hợp giải với (D OR E OR F) (-B) hợp giải với (B OR -C) (-E) hợp giải với (D OR E OR F) (D OR E OR F) hợp giải với (C OR -E OR -I) (-D) hợp giải với (D OR E OR F) (I) hợp giải với (C OR -E OR -I) (-C) hợp giải với (C OR D OR F OR -I) (-D) hợp giải với (D OR F) (B OR -E OR -I) hợp giải với (D OR E OR F) (-D) hợp giải với (C OR D OR F OR -I) (-D) hợp giải với (-A OR D OR F) (B OR -E OR -I) hợp giải với (E OR F) (E OR F) hợp giải với (C OR -E) (I) hợp giải với (C OR D OR F OR -I) (B OR -E OR -I) hợp giải với (-B) (B OR -E OR -I) hợp giải với (I) (C OR F OR -I) hợp giải với (-C) (B OR D OR F OR -I) hợp giải với (I) (I) hợp giải với (B OR F OR -I)

3. Tài liệu tham khảo

Sách Artificial Intelligence: A Modern Approach, Third Edition