ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

LAB 2 : LOGIC

MÔN HỌC: CƠ SỞ TRÍ TUỆ NHÂN TẠO

20120541 – Phan Thị Yến Nhi

Thành phố Hồ Chí Minh - 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

LAB 2 : LOGIC
MÔN HỌC: CƠ SỞ TRÍ TUỆ NHÂN TẠO

| Giảng viên |

Thầy Nguyễn Ngọc Đức

Thầy Nguyễn Duy Khánh

Thành phố Hồ Chí Minh - 2022

Mục lục

NỘI DUNG BÁO CÁO:

1. Thông tin:

a. Cấu trúc file:

- File input:
  - Dòng đầu tiên chứa mệnh đề α.
  - Dòng thứ hai chứa số lượng mệnh đề trong KB.
  - Các dòng chứa từng mệnh đề trong KB.
- File output:
  - Dòng đầu chứa số lượng mệnh đề được phát sinh trong vòng lặp(N).
  - N dòng tiếp theo biểu diễn các mệnh đề được phát sinh trong vòng lặp.
  - Dòng cuối chứa kết quả YES hoặc NO.

2. Đánh giá:

Tiêu chí đánh giá	Mức độ hoàn thành
Đọc dữ liệu đầu vào và lưu trong cấu trúc dữ liệu phù hợp	100%
Cài đặt giải thuật hợp giải trên logic mệnh đề	100%
Các bước suy diễn phát sinh đủ mệnh đề và kết luận đúng	100%
Tuân thủ mô tả định dạng của đề bài	100%
Báo cáo test case và đánh giá	100%

3. Giải thích code:

a. File main.py:

```
import os
from PL_Resolution import resolution
ROOT_DIR = os.path.dirname(os.path.abspath(__file__))
ROOT_INPUT=ROOT_DIR+'\INPUT'
ROOT_OUTPUT=ROOT_DIR+'\OUTPUT'

def main():
    for file in os.listdir(ROOT_INPUT):
        solution = resolution()
        solution.initSolution(file)
        file_output='output'+ file[file.find('input') + 5:]
        print(file[file.find('input') + 5:])
        solution.PL_RESOLUTION()

if __name__ == '__main__':

   main()
```

- **File main.py chứa hàm main dùng để thực thi chương trình.**
- **File main.py import:**
  - **os:** cho phép chúng ta làm việc với các tập tin và thư mục.
  - **PL_Resolution:** file chứa code giải quyết bài toán.
- **Hàm main:** đọc tất cả file input có trong thư mục INPUT, thực hiện hợp giải với từng input và ghi kết quả vào file output tương ứng trong thư mục OUTPUT.

b. File PL_ Resolution.py:

- Import os.
- Tạo class resolution chứa thông tin: KB, α, và kết quả sau khi thực thi chương trình.
- Các phương thức của class resolution:
  - Hàm __init__: hàm khởi tạo.

```
def __init__(self) -> None:

        self.alpha = []

        self.clauseKB=[]

        self.result = []
```

  - Hàm compactLiteral:
    - Input: Nhận vào một mệnh đề gồm các literal được nối với nhau bằng OR.
    - Output: trả về danh sách các literal của mệnh đề đã được rút gọn các literal trùng nhau.

```
def compactLiteral(self, clause)-> list:
        res = []
        for literal in clause:
            if literal.strip() not in res :
                res.append(literal.strip())

        return res
```

  - Hàm invertLiteral: dùng để phủ định một literal.
    - Input: một literal.
    - Output: phủ định của literal.

```
def invertLiteral(self, literal)-> list:
        if '-' in literal:
            return literal[1]
        return '-' + literal
```

  - Hàm resultLiteral:
    - Input: mệnh đề.
    - Output: phủ định của mệnh đề input.

```
def resultLiteral(self, clause)-> list:
        clause = self.compactLiteral(clause)
```

```
        res = []
        for literal in clause:
            if self.invertLiteral(literal.strip()) not in res :
                res.append(literal.strip())
        return res
```

- Hàm negateClause: phủ định một mệnh đề.
  - Input: một mệnh đề cần phủ định.
  - Output: phủ định của mệnh đề.

```
def negateClause(self, clause)-> list:
        listLiteral = clause.split('OR')
        listLiteral = self.compactLiteral(listLiteral)

        res = []
        for literal in listLiteral:
            literal = literal.strip()
            if(len(literal) == 2):
                res.append(literal[1])
            elif len(literal) == 1:
                res.append('-'+ literal)
        return res
```

- Hàm initSolution: tạo Solution chứa thông tin KB và mệnh đề α:
  - Input: tên file để mở và đọc dữ liệu về KB và α.
  - Output: kết quả đọc dữ liệu.

```
def initSolution(self, fileName) ->bool:
        path =
os.path.join(os.path.dirname(__file__),r'INPUT/'+fileName)
        with open(path, 'r') as f:
            try:
                self.alpha = f.readline().strip()
                numberOfKB = int(f.readline())
                self.clauseKB = []

                for i in range(numberOfKB):
                    self.clauseKB.append(f.readline().strip())
                listLiteral = self.negateClause(self.alpha)
                for literal in listLiteral:
                    self.clauseKB.append(literal)
                return True
            except:
                return False
```

- Hàm outputSolution: ghi dữ liệu ra file.
  - Input: tên file để mở và ghi dữ liệu.

- Từ kết quả result, ghi kết quả xuống file output tương ứng.

```
def outputSolution(self, fileName)->None:
        path =
os.path.join(os.path.dirname(__file__),r'OUTPUT/'+fileName)
        pl_resolution = self.PL_RESOLUTION()
        with open(path, 'w') as f:
            for step in self.result:
                f.write(f"{len(step)}\n")
                for clause in step:
                    f.write(f"{clause}\n")
            f.write("YES" if pl_resolution else "NO")
```

- Hàm sortClause: sắp xếp mệnh đề theo literal.
  - Input: mệnh đề cần sắp xếp.
  - Output: mệnh đề sau khi sắp xếp.

```
def sortClause(self, clause) -> list:
        literals = self.splitClause(clause)
        literals = sorted(literals, key = lambda x: x[-1])
        res= ""
        for i in range(len(literals)):
            if i== len(literals)-1:
                res +=  literals[i]
            else:
                res += literals[i] + ' OR '
        return res
```

- Hàm checkClause: kiểm tra mệnh đề có đối ngẫu hay không.
  - Input: hai mệnh đề cần kiểm tra.
  - Output: kết quả kiểm tra.

```
    def checkClause(self, clause1, clause2) -> bool:
        literals1 = self.splitClause(clause1)
        literals2 = self.splitClause(clause2)
        for i in literals1:
            for j in literals2:
                if self.invertLiteral(i) == j:
                    return True
        return False
```

- Hàm solveClause: giải quyết hai mệnh đề đối ngẫu:
  - Input: hai mệnh đề cần hợp giải.
  - Output: kết quả hợp giải.
  - Các bước:

- Bước 1: tách mỗi mệnh đề thành danh sách các literal tương ứng.
- Bước 2: khai báo mảng res chứa kết quả trả về.
- Bước 3: duyệt qua từng phần tử của từng mệnh đề. Nếu hai literal là phủ định của nhau => bước 4, ngược lại => bước 5:
- Bước 4: tiến hànhh hợp giải:
    - Tạo biến resolve chứa các literal của hai mệnh đề, trừ hai literal phủ định nhau.
    - Kết hợp các literal thành một mệnh đề được nối bằng OR.
    - Thêm mệnh đề mới vào res.
- Duyệt qua tất cả phần tử của hai mệnh đề và tiến hành hợp giải từng cặp literal phủ định nhau.
- Bước 5: Nếu là phần tử cuối của hai mệnh đề => trả về kết quả và kết thúc thuật toán. Ngược lại quay lại bước 3.

```
    def solveClause(self, clause1, clause2)-> list:
        clause1 = self.splitClause(clause1)
        clause2 = self.splitClause(clause2)
        res = []

        for i in range(len(clause1)):
            for j in range(len(clause2)):
                if self.invertLiteral(clause1[i]) == clause2[j]:
                    resolve = clause1[:i]  + clause1[i + 1:] +
clause2[:j] + clause2[j + 1:]
                    resolve = self.compactLiteral(resolve)
                    resClauses = ''
                    for k in range(len(resolve)):
                        if k== len(resolve)-1:
                            resClauses +=  resolve[k]
                        else:
                            resClauses += resolve[k] + ' OR '
                    if(resClauses == ''):
                        res.append('')
                    else :
                        res.append(self.sortClause(resClauses))

        return res
```

- Hàm checkvalid: kiểm tra mệnh đề có luôn đúng hay không.
    - Input: mệnh đề cần kiểm tra.
    - Output: kết quả kiểm tra.

- Duyệt qua từng cặp phần tử của mệnh đề. Nếu tối tại hai literal phủ định nhau => kết thúc thuật toán và trả về kết quả.

```
def checkValid(self, clause) -> bool:
        clause = self.splitClause(clause)
        for i in range(len(clause) - 1):
            for j in range(i + 1, len(clause)):
                if (self.invertLiteral(clause[i]) == clause[j]):
                    return True
        return False
```

- Hàm PL_RESOLUTION: thực hiện hợp giải trên KB và α:
  - Output: Kết quả bài toán.
  - Các bước thực hiện:
    - Bước 1: Tạo danh sách detailStep chứa các mệnh đề hợp giải ở từng bước, biến check kiểm tra xem có thể tiếp tục thực thi hay không.
    - Bước 2: thêm vào result, detailStep, new_clauses, new_steps một danh sách rỗng.
    - Bước 3: Với mỗi cặp mệnh đề trong KB, ta tiến hành hợp giải và ghi kết quả vào biến resultClauses.
    - Bước 4: Kiểm tra có tồn tại mệnh đề roonxng không. Nếu tồn tại mệnh đề rỗng, gán check = True. Duyệt qua từng phần tử của resultClauses, nếu mệnh đề trong resultClauses luôn đúng => Bước 5, nếu mệnh đề không có trong KB và không ở trong result => thêm mệnh đề vào new_clause và thêm hai mệnh đề hợp giải vào new_steps.
    - Bước 5: nếu đã duyệt qua tất cả phần tử của KB => bước 6, ngược lại => Bước 3.
    - Bước 6: nếu new_clause rỗng => kết thúc thuật toán và trả về false. Ngược lại => Bước 7.
    - Bước 7: duyệt qua tất cả phần tử của new_clause. Nếu mệnh đề chưa tồn tại trong KB => thêm vào KB, nếu mệnh đề chưa có trong phần tử cuối của result => thêm vào phần tử cuối của result.
    - Bước 8: kiểm tra biến check. Nếu kết quả bằng True => trả về True và kết thúc thuật toán. Ngược lại => Bước 2.

```
def PL_RESOLUTION(self)-> bool:
        detailStep = []
        check = False
        while True:
```

```
            self.result.append([])
            detailStep.append([])
            new_clauses = []
            new_steps = []
            n = len(self.clauseKB)
            for i in range(n):
                for j in range(i+1, n):
                    resultClauses = self.solveClause(self.clauseKB[i],
self.clauseKB[j])

                    if '' in resultClauses:
                        new_clauses.append({})
                        new_steps.append((self.clauseKB[i],
self.clauseKB[j]))
                        check = True
                        continue
                    for r in resultClauses:
                        if self.checkValid(r):
                            continue
                        if r not in self.clauseKB and r not in
self.result:
                            new_clauses.append(r)
                            new_steps.append((self.clauseKB[i],
self.clauseKB[j]))

            if len(new_clauses) == 0:
                return False

            for i in range(len(new_clauses)):
                if new_clauses[i] not in self.clauseKB:
                    self.clauseKB.append(new_clauses[i])

                if new_clauses[i] not in self.result[-1]:
                    self.result[-1].append(new_clauses[i])
                    detailStep[-1].append(new_steps[i])
            if check:
                return True
```

4. Test case:

a. Test case 1:

Input	Output	Ghi chú
-A OR B OR -C	2	
4	C OR -D	(-B OR C OR -D) hợp giải với (B OR C OR -D)
A	A OR D	(A OR -C OR D) hợp giải với C

-B OR C OR -D	3	
B OR C OR -D	A OR -B OR C	(-B OR C OR -D) hợp giải với (A OR D)
A OR -C OR D	A OR B OR C	(B OR C OR -D) hợp giải với (A OR D
	A OR C	(C OR -D) hợp giải với (A OR D
	3	
	A OR C OR -D	(-B OR C OR -D) hợp giải với (A OR B OR C
	A OR -B OR D	(A OR -C OR D) hợp giải với (A OR -B OR C
	A OR B OR D	
	1	
	A OR C OR D	(A OR -B OR C) hợp giải với (A OR B OR D
	0	
	NO	KB không entails α vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng.

b. Test case 2:

Input	Output	Ghi chú
A OR -B	7	
5	-A OR D OR -E	(-A OR -C) hợp giải với (C OR D OR -E
-A OR -C	-C OR -D	(-A OR -C) hợp giải với (A OR -D
C OR D OR -E	A OR C OR -E	(C OR D OR -E) hợp giải với (A OR -D
A OR -D	C OR D	(C OR D OR -E) hợp giải với E
-B OR -E	-D	(A OR -D) hợp giải với (-A)
E	-B	(-B OR -E) hợp giải với E
	-E	(-B OR -E) hợp giải với B)
	8	
	-A OR D	(-A OR -C) hợp giải với (C OR D)
	C OR -E	(C OR D OR -E) hợp giải với -D
	A OR C	(A OR -D) hợp giải với (C OR D)
	{}	E hợp giải với -E
	-A OR -C OR -E	(-A OR D OR -E) hợp giải với (-C OR -D)
	-A OR -E	(-A OR D OR -E) hợp giải với -D
	A OR -D OR -E	(-C OR -D) hợp giải với (A OR C OR -E)
	C	(C OR D) hợp giải với -D

	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

c. Test case 3:

Input	Output	Ghi chú
A OR B OR -C	3	
4	-B OR C OR -D	(A OR -B OR C OR -D) hợp giải với -A
A OR -B OR C OR -D	A OR -B OR C	(A OR -B OR C OR -D) hợp giải với (C OR D)
B OR D	D	(B OR D) hợp giải với -B
-A	2	
C OR D	A OR C OR D	(B OR D) hợp giải với (A OR -B OR C)
	-B OR C	-A hợp giải với (A OR -B OR C)
	0	
	NO	KB không entails α vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng.

d. Test case 4:

Input	Output	Ghi chú
H OR I OR N	13	
5	A OR -I OR N OR -P	(A OR H OR N OR -P) hợp giải với (-H OR -I OR N)
A OR H OR OR N OR -P	A OR H OR I OR N	(A OR H OR N OR -P) hợp giải với (H OR I OR N OR P)
-H OR -I OR N	A OR N OR -P	(A OR H OR N OR -P) hợp giải với -H
H OR N	A OR H OR -P	(A OR H OR N OR -P) hợp giải với -N
A OR I	-I OR N	(-H OR -I OR N) hợp giải với (H OR N)
H OR I OR N OR P	A OR -H OR N	(-H OR -I OR N) hợp giải với (A OR I)
	-H OR -I	(-H OR -I OR N) hợp giải với -N
	N	(H OR N) hợp giải với -H
	H	(H OR N) hợp giải với -N
	A	(A OR I) hợp giải với -I
	I OR N OR P	(H OR I OR N OR P) hợp giải với -H
	H OR N OR P	(H OR I OR N OR P) hợp giải với -I
	H OR I OR P	(H OR I OR N OR P) hợp giải với -N

	16	
	A OR H OR N	(A OR H OR N OR -P) hợp giải với (H OR N OR P)
	-H OR N OR P	(-H OR -I OR N) hợp giải với (I OR N OR P)
	-I OR N OR P	-H OR -I OR N) hợp giải với (H OR N OR P
	A OR N	(H OR N) hợp giải với (A OR -H OR N)
	A OR -H	(A OR I) hợp giải với (-H OR -I)
	A OR I OR N OR P	(H OR I OR N OR P) hợp giải với (A OR -H OR N)
	A OR I OR N	-H hợp giải với (A OR H OR I OR N)
	A OR -P	-H hợp giải với (A OR H OR -P)
	{}	-H hợp giải với H
	N OR P	-H hợp giải với (H OR N OR P)
	I OR P	-H hợp giải với (H OR I OR P)
	H OR P	-I hợp giải với (H OR I OR P)
	A OR -I OR -P	-N hợp giải với (A OR -I OR N OR -P)
	A OR H OR I	-N hợp giải với (A OR H OR I OR N)
	A OR H OR -I OR N	(A OR -I OR N OR -P) hợp giải với (H OR N OR P)
	A OR N OR P	(A OR -H OR N) hợp giải với (H OR N OR P)
	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

e. Test case 5:

Input	Output	Ghi chú
A OR B	4	
3	A OR -B	(A OR -D) hợp giải với (A OR -B OR D)
A OR -D	-D	(A OR -D) hợp giải với -A
C OR -D	A OR -B OR C	(C OR -D) hợp giải với (A OR -B OR D)
A OR -B OR D	-B OR D	(A OR -B OR D) hợp giải với -A
	1	
	-B OR C	(C OR -D) hợp giải với (-B OR D)
	0	

	NO	KB không entails α vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng.

5. Đánh giá thuật toán:

a. Ưu điểm:

- Việc cài đặt thuật toán không quá phức tạp.
- Đơn giản, dễ hiểu.
- Trả về kết quả trong hầu hết các trường hợp.
- Kết quả chính xác, đúng và đủ.

b. Nhược điểm:

- Tùy theo thứ tự việc lấy các cặp mệnh đề hợp giải có thể xảy ra hiện tượng tràn bộ nhớ đối với các bài toán có kích thước lớn.
- Việc duyệt hết các mệnh đề trong *KB* mất nhiều thời gian và không khả thi, phát sinh các mệnh đề trùng lặp.
- KB và α phải được viết ở dạng hội chuẩn CNF.
- Chỉ có thể áp dụng cho các bài toán trong lĩnh vực logic mệnh đề.

c. Giải pháp:

- Để tránh sự bùng nổ tổ hợp. Có thể áp dụng các heuristics: Chiến lược ưu tiên các biểu thức đơn, Chiến lược đơn giản hóa các biểu thức, Chiến lược giảm số lần hợp giải, Chiến lược sắp thứ tự các hợp giải, …
- Hợp giải tất cả cặp các mệnh đề phức hợp có thể có, dừng khi gặp mâu thuẫn. Thêm các mệnh đề phức hợp kết quả, bỏ các mệnh đề phức hợp chứa biến mệnh đề này.
- Thiết lập độ ưu tiên cho các mệnh đề trong KB, tìm duyệt các mệnh đề có số lượng các literal ít.
- Bỏ qua không hợp giải các mệnh đề cũ đã hợp giải trong KB.