# ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated---------------o0o---------------

**BÁO CÁO: HỢP GIẢI TRÊN LOGIC MỆNH ĐỀ**

***Môn học:* CƠ SỞ TRÍ TUỆ NHÂN TẠO**

***Chủ đề:* HỢP GIẢI TRÊN LOGIC MỆNH ĐỀ**

***Giảng viên lý thuyết:* thầy BÙI TIẾN LÊN**

***Giảng viên thực hành:* thầy NGUYỄN THÁI VŨ**

***Họ tên:* LÊ NGỌC TƯỜNG**

***Mã số sinh viên:* 20127383**

***Lớp học phần:* 20CLC10**

***THÀNH PHỐ HỒ CHÍ MINH – 2022***

1. **THÔNG TIN SINH VIÊN:**

*Họ tên: Lê Ngọc Tường.*

*Mã số sinh viên: 20127383.*

*Lớp học phần: 20CLC10.*

1. **MÔI TRƯỜNG LÀM VIỆC:**

*Python 3.10.0.*

*Visual Studio Code & Github:* [*https://github.com/lengoctuong/Search\_Introduce-to-AI*](https://github.com/lengoctuong/Search_Introduce-to-AI)

1. **GIỚI THIỆU VỀ CHƯƠNG TRÌNH:**
2. ***Chạy chương trình:***

Chương trình đọc input.txt, thực hiện hợp giải và ghi output.txt.

Sau đó lần lượt làm tương tự cho 4 test case còn lại (đọc input2.txt, input3.txt, input4.txt, input5.txt và ghi output2.txt, output3.txt, output4.txt, output5.txt).

1. ***Các biến toàn cục và hàm:***
2. *Các biến toàn cục:*

KB = []

alpha = []

outputLstMi = []

# Danh sách các resolvents mới để thực hiện ghi vào file output.txt

# outputLstMi = [ LstM1, LstM2, ..., LstMn ]

# Voi LstMi la tap hop cac resolvents duoc sinh ra trong vong lap thu i, len(LstMi) la tong so resolvents do

1. *Các hàm:*

# Doc file input

def Read\_CNF(fileName):

# Ghi file output

def Write(fileName, outputLstMi, result):

# Sap xep lai cac literals trong clause theo thu tu bang chu cai

def Sort\_Clause(clause):

# Rut gon literals

def Minimize\_Clause(clause):

# Phu dinh clause

def Not\_CNFClause(clause):

# Kiem tra hai tap hop chua clause set va subset, lieu subset co la con cua set khong

def Is\_Subset(set, subset):

# Kiem tra 2 clause theo CNF co tuong duong hay khong

def CNF\_Equiv(clause1, clause2):

# Them 1 clause 'newClause' vao list cac clauses (phep hop cua clauses va newClause)

def Add\_Clause(clauses, newClause):

# Them list cac clauses moi 'newClauses' vao list cac clauses (phep hop cua clauses va newClauses)

def Add\_Clauses(clauses, newClauses):

# Ham hop giai hai clause Ci va Cj

def PL\_Resolve(Ci, Cj):

# Ham tra ve KB suy ra alpha bang Propositional Logic - Resolution

# Ham luu thong tin vao cau truc outputLstMi de ghi ra file output.txt

def PL\_Resolution(KB, alpha, outputLstMi):

1. ***Thuật toán:***

Thuật toán tham khảo từ sách Artificial Intelligence: A Modern Approach, Third Edition, Chương 7, Hình 7.12, hàm PL-RESOLUTION và viết thêm các hàm cần thiết hỗ trợ.

1. **KỊCH BẢN KIỂM THỬ:**
2. ***Kịch bản 1: False***

*Input:*

A

4

-A OR B

-C OR B

A OR C OR -B

-B

*Output:*

2

-C

-B OR C

2

-A OR C

A OR -B

1

A OR -C

0

NO

1. ***Kịch bản 2: True***

*Input:*

-A

4

-B OR C

-C OR A OR B

-A OR C

-C

*Output:*

3

-B

-A

C

4

A OR -C

B OR -C

A OR B

{}

YES

1. ***Kịch bản 3: False***

*Input:*

-C

6

-A OR -B OR C

-D OR -E OR B

-F OR A

-E OR A

E

D

*Output:*

6

-A OR C OR -D OR -E

-B OR C OR -F

-B OR C OR -E

B OR -D

B OR -E

A

9

-A OR C OR -D

-A OR C OR -E

-B OR C

C OR -D OR -E OR -F

C OR -D OR -E

B

C OR -D OR -F

C OR -E OR -F

C OR -E

3

-A OR C

C OR -D

C OR -F

0

NO

1. ***Kịch bản 4: False***

*Input:*

-C

4

-A OR C

-A OR K

A OR -K

K

*Output:*

2

C OR -K

A

0

NO

1. ***Kịch bản 5: True***

*Input:*

Q

7

-P OR Q

-L OR -M OR P

-B OR -L OR M

-A OR -P OR L

-A OR -B OR L

A

B

*Output:*

10

-L OR -M OR Q

-P

-B OR -L OR P

-A OR -B OR -M OR P

-A OR -B OR M OR -P

-A OR -B OR M

-L OR M

L OR -P

-B OR L

-A OR L

26

-B OR -L OR Q

-A OR -B OR -M OR Q

-L OR -M

-A OR -B OR -L OR P

-L OR P

-B OR -M OR P

-A OR -M OR P

-B OR M OR -P

-B OR M

-A OR -M OR -P OR Q

-A OR -B OR L OR -M

-A OR M OR -P

-A OR -B OR P

L

-A OR M

-A OR -B OR -L OR -P OR Q

-A OR -B OR -L OR Q

-L OR Q

-M OR -P OR Q

-B OR -M OR Q

-A OR -M OR Q

-B OR -L

-A OR -B OR -M

-A OR -B OR -L OR M

-B OR P

M OR -P

41

-A OR -B OR Q

-B OR Q

-A OR -L OR -M OR Q

-M OR P

-A OR -L OR P

-A OR -B OR -L OR -M OR Q

-B OR -L OR -P OR Q

-A OR -B OR -L

-A OR -B OR -P OR Q

-A OR -M OR -P

-A OR L OR -M

-A OR -P OR Q

-A OR -B OR -P

-A OR -B

-B OR L OR -M

M

-M OR Q

-B OR -M

-A OR P

-A OR -L OR -P OR Q

-A OR -L OR Q

-L

-A OR -M

-A OR -L OR M

P

-A OR -B OR -L OR -P

-M OR -P

-L OR -P OR Q

-B

-B OR -L OR -M OR Q

-A OR -B OR L OR -P

-B OR -P OR Q

-B OR -P

-A OR Q

-B OR -L OR -P

-A OR -L OR -P

-M

-A OR -L

-L OR -P

-A OR -B OR -M OR -P OR Q

Q

9

-A OR -L OR -M

-A OR -B OR -L OR -M

-B OR -L OR -M

-A OR -P

L OR -M

-B OR L OR -P

-B OR -M OR -P OR Q

-A

{}

YES

1. **ĐÁNH GIÁ:**
2. ***Ưu điểm:***

Dễ dàng để mô tả thế giới bằng các mệnh đề.

Bểu đạt bằng CNF là đơn giản.

Chứng minh dễ dàng chỉ bằng 1 luật logic là resolution.

1. ***Khuyết điểm:***

Sẽ véc cạn tất cả các cặp clause khi không có clause nào là {}.

Khi thế giới là phức tạp thì cần rất nhiều sentences và symbols để có thể mô tả dẫn đến tốn nhiều thời gian và không gian.

Chỉ có thể trả lời True/False/Unknow, không thể xử lí được trường hợp không chắc chắn.

Chứng minh chỉ được dựa trên các luật resolution, không thể tinh chỉnh dữ liệu.

1. ***Giải pháp khắc phục:***

Khắc phục khuyết điểm 2: Có thể dùng logic vị từ.

Khắc phục khuyết điểm 3: Muốn xử lý được uncertain cần phải áp dụng xác suất.

Khắc phục khuyết điểm 4: Có thể sử dụng machine learning.

1. **TÀI LIỆU THAM KHẢO:**

[1] Slides của thầy Bùi Tiến Lên:

<https://drive.google.com/drive/folders/1nLDrkQHmcBk008KTpR3WS_YQhtm15eGW>

[2] Đề: Project02\_logic.ipynb