LAB 2

LOGIC

Sinh Viên Thực Hiện:

MSSV: 19120047

MSSV: Trần Xuân An

Giáo Viên Hướng Dẫn:

1. Lê Hoài Bắc

2. Nguyễn Ngọc Băng Tâm

3. Nguyễn Duy Khánh

Mục lục

1. Tổng quan

1.1 Mô tả bài toán

- Cho cơ sở tri thức (KB) và một câu α, cả hai đều được biểu diễn bằng logic mệnh đề và được chuẩn hóa về dạng hội chuẩn CNF. Xác định KB entails α ($KB \models \alpha$) bằng hợp giải.
- Đặc tả dữ liệu đầu vào và đầu ra

❖ **Dữ liệu đầu vào**: KB và α theo dạng chuẩn CNF được lưu trong tập tin input.txt. Tập tin có định dạng quy ước như sau:

- Dòng đầu tiên chứa câu α
- Dòng thứ hai chứa số nguyên N – số mệnh đề có trong KB
- N dòng tiếp theo biểu diễn các mệnh đề trong KB, mỗi mệnh đề trên một dòng
- Literal dương được biểu diễn bằng ký tự đơn viết hoa (A-Z). Literal âm là literal dương có dấu trừ ('-') ngay trước ký tự
- Từ khóa OR nối các literal nối với nhau. Có thể có một hay nhiều khoảng trắng giữa các literal và từ khóa

❖ **Dữ liệu đầu ra:** Tập hợp mệnh đề được phát sinh trong quá trình hợp giải và câu kết luận được lưu trong tập tin output.txt. Tập tin có định dạng quy ước như sau:

- Dòng đầu tiên chứa số nguyên M1 – số mệnh đề được phát sinh trong vòng lặp đầu tiên. M1 dòng tiếp theo biểu diễn các mệnh đề được phát sinh trong vòng lặp đầu tiên (kể cả mệnh đề rỗng), mỗi mệnh đề trên một dòng. Mệnh đề rỗng được biểu diễn bằng chuỗi "{}"
- Các vòng lặp tiếp theo (lần lượt có M2 , M3 ,…, Mn mệnh đề) được biểu diễn tương tự như trên
- Dòng cuối cùng trình bày câu kết luận, tức là trả lời câu hỏi "KB entails α ?". In YES nếu KB entails α. Ngược lại, in NO.
- Bỏ qua các mệnh đề trùng (xuất hiện trong cùng vòng lặp hay, KB ban đầu hay những vòng lặp trước đó).

1.2 Đánh giá mức độ hoàn thành

Yêu cầu	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%

2. Giải Quyết Bài Toán

2.1 Ý tưởng giải quyết bài toán

- Logic mệnh đề (Propositional logic) được xem là một trong những cách biểu diễn tri thức đơn giản nhất. Trong đó, đơn vị cơ bản nhất để biểu diễn tri thức là mệnh đề. Mỗi mệnh đề chỉ có thể nhận một trong hai chân trị: đúng/sai.

- Hợp giải (Resolution) trên logic mệnh đề là phương pháp suy diễn ứng dụng chứng minh phản chứng: để chứng minh KB $\models \alpha$, ta chứng minh không tồn tại (KB $\wedge \neg\alpha$).

❖ **Thuật toán sử dụng:**

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
inputs: KB, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
new ← { }
loop do
for each pair of clauses C_i, C_j **in** *clauses* **do**
resolvents ← PL-RESOLVE(C_i, C_j)
if *resolvents* contains the empty clause **then return** *true*
new ← *new* ∪ *resolvents*
if *new* ⊆ *clauses* **then return** *false*
clauses ← *clauses* ∪ *new*

Figure 7.12 A simple resolution algorithm for propositional logic. The function PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

2.2 Thuật Toán

❖ Class:

	Giải Thích

class **Resolution**:
def **__init__**(self):
self.alpha=[]
self.KB=[]
	- Class chính giải bài toán
	- alpha là α của bài toán
	- KB các mệnh đề để giải quyết bài toán

def **readFile**(self, filename):
f=open(filename,'r')
str=f.readline()
str=str[:len(str)-1].split(' OR ')
self.alpha.append(str)
num= int(f.readline())
for i in range(num):
str=f.readline()
str=str[:len(str)-1].split(' OR ')
self.KB.append(str)
self.alpha = self.**cnfSentence**(self.alpha)
self.KB = self.**cnfSentence**(self.KB)
	- Hàm đọc thông tin từ file lên và lần lượt chuyển các mệnh đề đầu vào về dạng cnf
	- Đầu vào là tên file input
	- Đầu ra là lưu dữ liệu đọc từ file vào alpha và KB

def **cnfSentence**(self, sentences):
listSentence=[]
for s in sentences:
newS = **sorted**(list(set(deepcopy(s))), key=lambda x: x[-1])
if self.**checkSentence**(newS) and newS not in listSentence:
listSentence.append(newS)
return listSentence
	- sắp xếp mệnh đề theo thứ tự chữ cái
	- Loại bỏ các mệnh đề vô ích và mệnh đề bị trùng
	- Đầu vào là danh sách các mệnh đề cần chuyển về dạng cnf
	- Đầu ra là mệnh đề đã được chuyển về dạng cnf

def **checkLiterals**(self,l1, l2):
if(l1[0] != l2[0] and l1[-1] == l2[-1]):
return True
return False
	- Kiểm tra 2 literals có phải là α và -α hay không
	- Đầu vào là: 2 literals cần kiểm tra
	- Đầu ra là true nếu 2 literals có dạng: α và -α
	- Đầu ra là false nếu 2 literal không có dạng: α và -α

def **checkSentence**(self, sentence):
for i in range(len(sentence) - 1):
if self.**checkLiterals**(sentence[i], sentence[i+1]):
return False
return True
	- Kiểm tra mệnh đề có hợp lệ hay không.
	- Các mệnh đề hợp lệ là mệnh đề không có dạng $A \vee B \vee -B$
	- Đầu vào là mệnh đề cần kiểm tra
	- Đầu ra là true nếu mệnh đề không có dạng: $A \vee B \vee -B$
	- Đầu ra là false nếu mệnh đề có dạng: $A \vee B \vee -B$

def **negationSentence**(self, sentence):
negationS=[]
for s in sentence:
for literal in s:
newLiteral=[self.**negationLiteral**(literal)]
negationS.append(newLiteral)
return negationS
	- Phủ định mệnh đề alpha. Ta sẽ phủ định từng literal và thêm từng literal phủ định vào mệnh đề phủ định. Trả về mệnh đề phủ định đó
	- Đầu vào là mệnh đề cần phủ định
	- Đầu ra là mệnh đề đã được phủ định

def **negationLiteral**(self, literal):
if literal[0] == '-':
return literal[1]
return '-' + literal
	- Phủ định từng literal .
	- Đầu vào là literal phủ định
	- Đầu ra là literal đã được phủ định
	VD:
	+ Đầu vào là A--> Đầu ra là -A
	+ Đầu vào là -A--> Đầu ra là A

def **solve2Sentence**(self, s1, s2):
newSentences= []
for i in range(len(s1)):
for j in range(len(s2)):
if self.**checkLiterals**(s1[i], s2[j]):
newS = s1[:i] + s1[i + 1:] + s2[:j] + s2[j + 1:]
k = **sorted**(list(set(deepcopy(newS))), key=lambda x: x[-1])
newSentences.append(k)
return newSentences
	- Giải quyết 2 mệnh đề s1 và s2
	- Nếu trong s1 chứa literal α và s2 chứa literal -α thì ta cần bỏ qua 2 literal này và nối s1 và s2
	- Đầu vào là 2 câu cần hợp giải
	- Đầu ra là danh sách các câu đã được hợp giải từ s1 và s2
	VD:
	+ Đầu vào s1: $A \vee B$, s2: $-B$
	+ Đầu ra s2: A

def **resolution**(self, filename):
f = open(filename, 'w')
listSentence = deepcopy(self.KB)
negOfAlpha = self.**cnfSentence**(self.**negationSentence**(self.alpha))
for sentence in negOfAlpha:
if sentence not in listSentence:
listSentence.append(sentence)
while True:
newSentence=[]
for i in range(len(listSentence)):
for j in range(i + 1, len(listSentence)):
newS = self.**solve2Sentence**(listSentence[i], listSentence[j])
	- Hàm hợp giải bài toán.
	- Hàm hợp giải này dược xây dựng dựa trên thuật toán trong Giáo trình Artificial Intelligence - A Modern Approach 3rd edition
	- Đầu vào là KB và alpha
	- Đầu ra: tập hợp mệnh đề được phát sinh trong quá trình hợp giải và câu kết luận được lưu trong tập tin output.txt

if [] in newS:
newSentence.append([])
print('len: ', str(len(newSentence)))
f.write(str(len(newSentence)) + '\n')
for i in range(len(newSentence)):
listSentence.append(newSentence[i])
print(self.sentenceToString(newSentence[i]))
f.write(self.**sentenceToString**(newSentence[i]) + '\n')
f.write('YES\n')
f.close()
return True
 	
for s in newS:
if not self.checkSentence(s):
break
if s not in listSentence and s not in newSentence:
newSentence.append(s)
print('len: ', str(len(newSentence)))
f.write(str(len(newSentence)) + '\n')
if len(newSentence) == 0:
f.write('NO\n')
f.close()
return False
for i in range(len(newSentence)):
listSentence.append(newSentence[i])
print(self.sentenceToString(newSentence[i]))
f.write(self.**sentenceToString**(newSentence[i]) + '\n')	

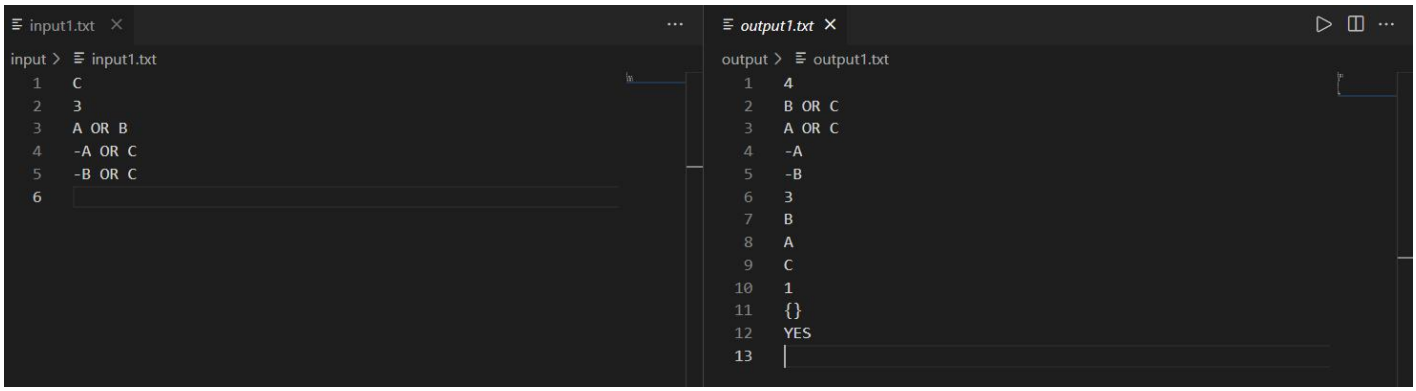
def **sentenceToString**(self, sentence):
if len(sentence) == 0:
return '{}'
string = ''
for i in range(len(sentence) - 1):
string =string+ str(sentence[i]) + ' OR '
string =string+ str(sentence[len(sentence) - 1])
 	
return string
	- Chuyển từ dạng các mệnh đề đang được lưu theo mảng kí tự thành chuỗi để in ra file output
	- Đầu vào: mệnh đề lưu ở dạng danh sách kí tự
	- Đầu ra: mệnh đề ở dạng chuỗi

input=["input/input1.txt", "input/input2.txt", "input/input3.txt", "input/input4.txt", "input/input5.txt"] output=["output/output1.txt", "output/output2.txt", "output/output3.txt", "output/output4.txt", "output/output5.txt"]	- Danh sách các file input và output

def main():
for i in range(0, len(input)):
current_dir = dirname(__file__)
print('File input '+str(i))
fileIn = join(current_dir, input[i])
fileOut = join(current_dir, output[i])
resolution = **Resolution**()
resolution.**readFile**(fileIn)
resolution.**resolution**(fileOut)
	- Hàm main:
	+ Đọc dữ liệu đầu vào và lưu trong cấu trúc dữ liệu thích hợp
	+ Gọi hàm PL-Resolution để thực thi giải thuật hợp giải
	+ Ghi dữ liệu đầu ra vào tập tin đầu ra theo định dạng hợp lệ

3. Test Case

❖ **Test case 1:**



Input	Output	Ghi chú
C 3 A OR B -A OR C -B OR C	4 B OR C A OR C -A -B 3 B A C 1 {} YES	 (A OR B) hợp giải với (-A OR C) (A OR B) hợp giải với (-B OR C) (-A OR C) hợp giải với (C) (-B OR C) hợp giải với (C) (B OR C) hợp giải với (-C) (A OR B) hợp giải với (-B) (B OR C) hợp giải với (-B) (C) hợp giải với (-C)

❖ Test case 2

input2.txt

```
-A OR F
4
-A OR B OR C
-B OR D OR F
-A OR -D OR F
-C OR F

```

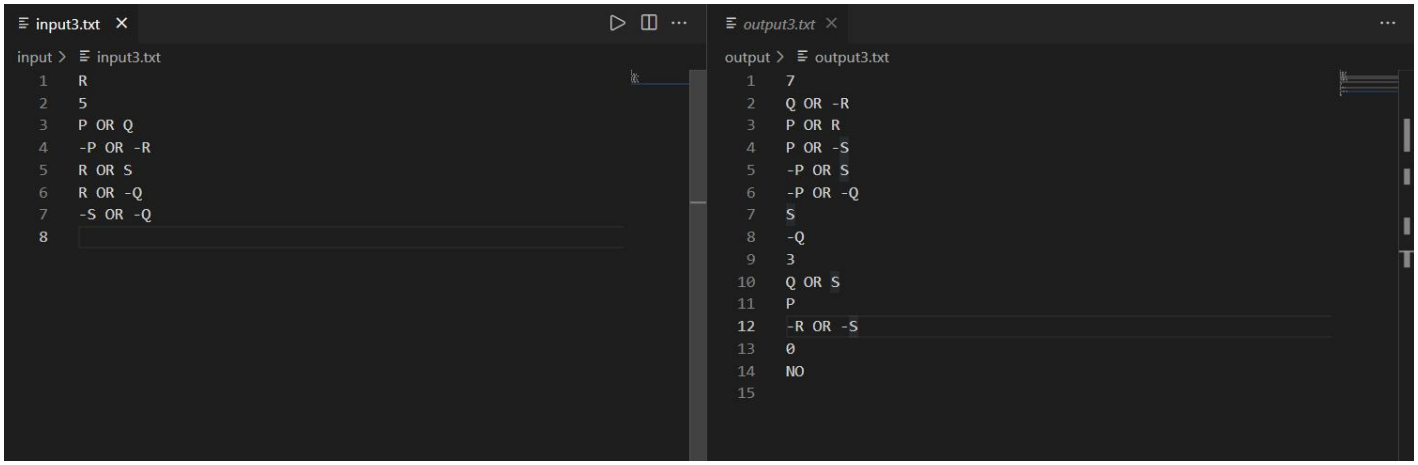
output2.txt

```
8
-A OR C OR D OR F
-A OR B OR F
B OR C
-A OR -B OR F
-B OR D
-D OR F
-A OR -D
-C
12
-A OR C OR F
-A OR C OR D
-A OR B
-A OR D OR F
C OR D OR F
-B OR F
B OR F
-D
-A OR -B
-A OR F
C OR D
B
9
-A OR C
D OR F
C OR F
-B
F
-A OR D
-A
D
C
1
{}
YES

```

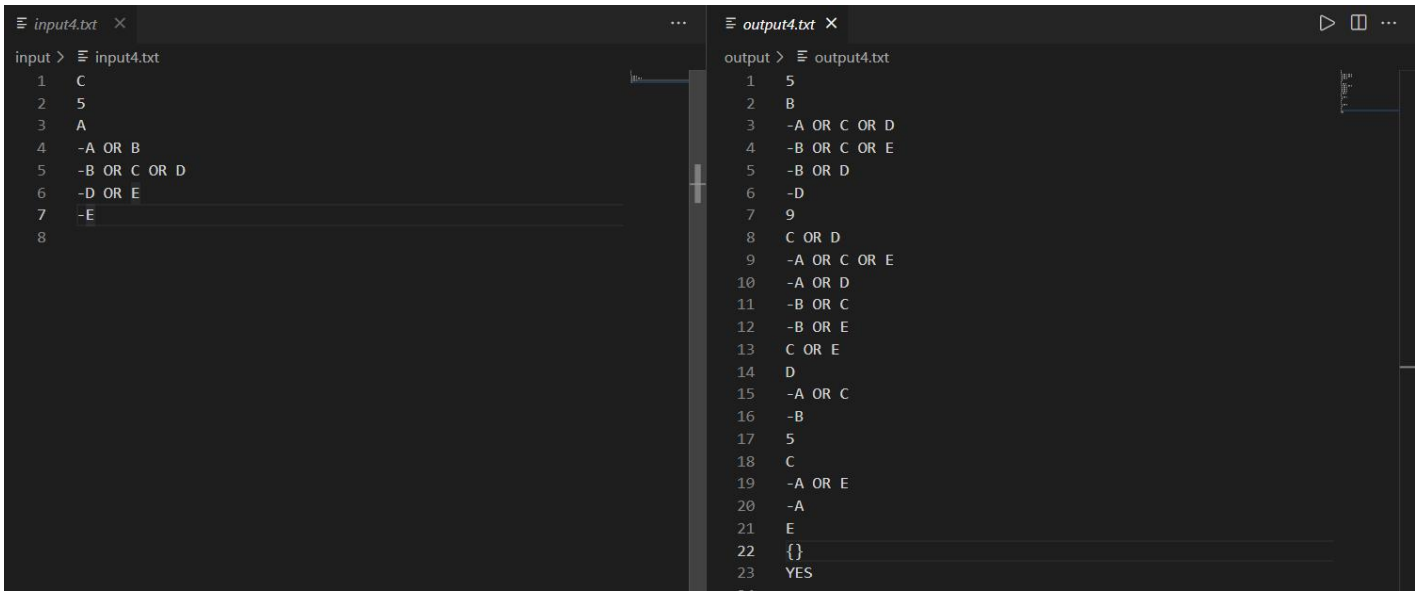
Input	Output	Ghi chú
-A OR F 4 -A OR B OR C -B OR D OR F -A OR -D OR F -C OR F	8 -A OR C OR D OR F -A OR B OR F B OR C -A OR -B OR F -B OR D -D OR F -A OR -D -C 12 -A OR C OR F -A OR C OR D -A OR B -A OR D OR F C OR D OR F -B OR F B OR F -D -A OR -B -A OR F C OR D B 9 -A OR C D OR F C OR F -B F -A OR D -A D C 1 {} YES	 (-A OR B OR C) hợp giải với (-B OR D OR F) (-A OR B OR C) hợp giải với (-C OR F) (-A OR B OR C) hợp giải với (A) (-B OR D OR F) hợp giải với (-A OR -D OR F) (-B OR D OR F) hợp giải với (-F) (-A OR -D OR F) hợp giải với (A) (-A OR -D OR F) hợp giải với (-F) (-C OR F) hợp giải với (-F) (-A OR B OR C) hợp giải với (-A OR -B OR F) (-A OR B OR C) hợp giải với (-B OR D) (-A OR B OR C) hợp giải với (-C) (-B OR D OR F) hợp giải với (-A OR B OR F) (-B OR D OR F) hợp giải với (B OR C) (-B OR D OR F) hợp giải với (-D OR F) (-C OR F) hợp giải với (B OR C) (A) hợp giải với (-A OR -D) (-F) hợp giải với (-A OR -B OR F) (-A OR B OR F) hợp giải với (-A OR -B OR F) (B OR C) hợp giải với (-B OR D) (B OR C) hợp giải với (-C) (-A OR C OR F) hợp giải với (-F) (-C OR D OR F) hợp giải với (C) (C OR D OR F) hợp giải với (-D) (-B OR F) hợp giải với (-F) (-B OR F) hợp giải với (B OR F) (-A OR D OR F) hợp giải với (-F) (-A OR F) hợp giải với (-F) (C OR D) hợp giải với (-C) (C OR D) hợp giải với (-D) (F) hợp giải với (-F)

❖ Test case 3



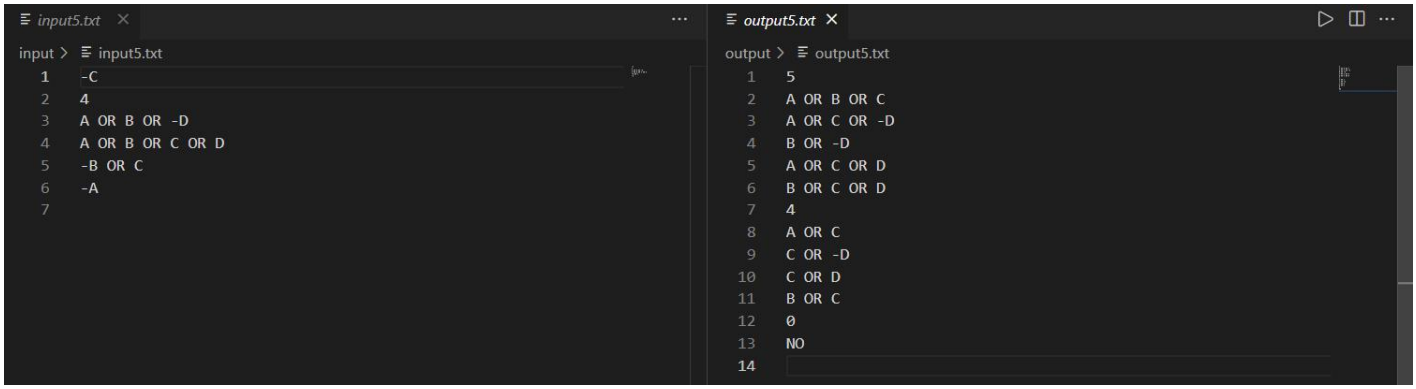
Input	Output	Ghi chú
R 5 P OR Q -P OR -R R OR S R OR -Q -S OR -Q	7 Q OR -R P OR R P OR -S -P OR S -P OR -Q S -Q 3 Q OR S P -R OR -S 0 NO	 (P OR Q) hợp giải với (-P OR -R) (P OR Q) hợp giải với (R OR -Q) (P OR Q) hợp giải với (-S OR -Q) (-P OR -R) hợp giải với (R OR S) (-P OR -R) hợp giải với (R OR -Q) (R OR S) hợp giải với (-R) (R OR -Q) hợp giải với (-R) (P OR Q) hợp giải với (-P OR S) (P OR Q) hợp giải với (-Q) (-S OR -Q) hợp giải với (Q OR -R)

❖ Test case 4



Input	Output	Ghi chú
C 5 A -A OR B -B OR C OR D -D OR E -E	5 B -A OR C OR D -B OR C OR E -B OR D -D 9 C OR D -A OR C OR E -A OR D -B OR C -B OR E C OR E D -A OR C -B 5 C -A OR E -A E {} YES	 (A) hợp giải với (-A OR B) (-A OR B) hợp giải với (-B OR C OR D) (-B OR C OR D) hợp giải với (-D OR E) (-B OR C OR D) hợp giải với (-C) (-D OR E) hợp giải với (-C) (-B OR C OR D) hợp giải với (B) (-A OR C OR D) hợp giải với (-D OR E) (-A OR B) hợp giải với (-B OR D) (-B OR C OR E) hợp giải với (-E) (-B OR C OR E) hợp giải với (-C) (-B OR C OR E) hợp giải với (B) (-B OR D) hợp giải với (B) (-A OR C OR D) hợp giải với (-D) (-B OR D) hợp giải với (-D) (C OR D) hợp giải với (-D) (-A OR C OR E) hợp giải với (A) (-A OR D) hợp giải với (-D) (-B OR E) hợp giải với (-B) (D) hợp giải với (-D)

❖ Test case 5



Input	Output	Ghi chú
-C 4 A OR B OR -D A OR B OR C OR D -B OR C -A	5 A OR B OR C A OR C OR -D B OR -D A OR C OR D B OR C OR D 4 A OR C C OR -D C OR D B OR C 0 NO	 (A OR B OR -D) hợp giải với (A OR B OR C OR D) (A OR B OR -D) hợp giải với (-B OR C) (A OR B OR -D) hợp giải với (-A) (A OR B OR C OR D) hợp giải với (-B OR C) (A OR B OR C OR D) hợp giải với (-A) (A OR B OR C) hợp giải với (-B OR C) (A OR C OR -D) hợp giải với (-A) (A OR C OR D) hợp giải với (-A) (B OR C OR D) hợp giải với (B OR -D)

4. Đánh Giá

❖ **Ưu điểm**

- Thuật toán đơn giản dễ cài dặt và sử dụng
- Tính đúng đắn: thuật toán này giải được các logic mệnh đề ra và cho ra kết quả đúng
- Tính dừng: thuật toán sẽ luôn dừng

❖ **Nhược điểm**

- Thuật toán sử dụng nhiều vòng lặp
- Phát sinh tất cả các mệnh đề liên quan tới các mệnh đề ban đầu. Dẫn đến khối lượng mệnh đề cần hợp giải là rất lớn
- Có phát sinh các mệnh đề vô ích
- Thiếu cơ đế định hướng: các mệnh đề phát sinh có thể không liên quan và cũng không hướng đến kết quả cần tìm
- Chưa ưu tiên hợp giải các mệnh đề ngắn giúp. Các mệnh đề sinh ra có thể chứa rất nhiều literal

❖ **Cải tiến**

- Ưu tiên hợp giải các mệnh đề có ít literal giúp sau khi hợp giải các mệnh đề mới sinh ra ngắn hơn
- Xây dựng cơ chế điều hướng. Ưu tiên phát sinh liên quan và cũng không hướng đến kết quả cần tìm

5. Tài liệu tham khảo

- Giáo trình Artificial Intelligence - A Modern Approach 3rd edition