ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

---oOo---

BÁO CÁO BÀI THỰC HÀNH 2

LOGIC

Họ và tên sinh viên:	Huỳnh Tấn Thọ
Mã số sinh viên:	19120383
Lớp:	19_3
Môn học:	Cơ sở Trí tuệ Nhân tạo
Giảng viên:	Lê Hoài Bắc

Thành phố Hồ Chí Minh, tháng 5 năm 2022

MỤC LỤC

1. Giới thiệu chung

- Mục đích của báo cáo này nhằm định nghĩa, giải thích code của Câu 4 – Hợp giải trong logic mệnh đề, của đồ án thực hành 2, môn Cơ sở Trí tuệ Nhân tạo.
- Các phần chính của báo cáo:
  - Hàm hỗ trợ
  - Hàm hợp giải mệnh đề
  - Hàm đọc ghi file
  - Mô tả, giải thích kết quả các kịch bản test
  - Đánh giá thuật toán

2. Các hàm hỗ trợ trong hợp giải mệnh đề (solver.py)

2.1. Một số khái niệm

- Một literal là một ký tự in hoa được lưu trong một `string`, kèm theo dấu "-" mang ý nghĩa phủ định (nếu có). Ví dụ: "-A", "B", "C",…
- Một literals là một danh sách các literal, được lưu trong cấu trúc dữ liệu dạng `list`. Ví dụ: ["A", "-B", "-C", "D"], ["-P", "Q", "R"],…
- Một clause (mệnh đề) là một `string` gồm các literal nối với nhau bằng chuỗi "OR". Ví dụ: "A OR B OR -C", "-P OR Q",…
- Một clauses (tập các mệnh đề) là một danh sách các clause được lưu trong cấu trúc dữ liệu dạng `list`. Ví dụ: ["P OR -Q", "-P", "S OR T"], ["M", "M OR -N OR O"],…

2.2. Hàm extract_literals(clause)

- Nhận vào một mệnh đề, là một string gồm các literal nối với nhau bằng chuỗi "OR".
- Trả về một list các literal tương ứng với mệnh đề đó.

```
print(extract_literals("A OR -B OR C"))
print(extract_literals("-B OR C"))
print(extract_literals("C"))
```

Output

```
['A', '-B', 'C']
['-B', 'C']
['C']
```

2.3. Hàm negate(clause)

- Nhận vào một `string` gồm các literal nối với nhau bằng chuỗi "OR".
- Trả về một `list` các literal tương ứng với mệnh đề đó dưới dạng **phủ định**.

```
print(negate("A OR -C"))
print(negate("-A OR B OR -C OR D"))
print(negate("D"))
print(negate("-D"))
```

Output

```
['-A', 'C']

['A', '-B', 'C', '-D']

['-D']

['D']
```

2.4. Hàm is_complementary(literal_1, literal_2)

- Nhận vào hai literal.
- Trả về `True` nếu chúng bù nhau, ngược lại trả về `False`.

```
print(is_complementary("-A", "B"))
print(is_complementary("C", "-C"))
print(is_complementary("A", "-B"))
print(is_complementary("A", "A"))
print(is_complementary("-A", "-B"))
print(is_complementary("-D", "D"))
print(is_complementary("-C", "-C"))
```

Output

```
False

True

False

False

False

True

False
```

2.5. Hàm remove_duplicates(literals)

- Nhận vào một `list` các `string`, mỗi `string` là một literal.
- Trả về một `list` các literal tương ứng đã loại bỏ các literal trùng nhau.

```
print(remove_duplicates(["A", "A", "B", "C"]))
print(remove_duplicates(["A", "B", "B", "C", "C"]))
print(remove_duplicates(["-A", "-A", "C", "D", "D"]))
print(remove_duplicates(["A", "-A", "B", "C"]))
print(remove_duplicates(["A", "B", "C", "D"]))
```

Output

```
['A', 'B', 'C']
['A', 'B', 'C']
['-A', 'C', 'D']
['A', '-A', 'B', 'C']
['A', 'B', 'C', 'D']
```

2.6. Hàm literals_sorter(literals)

- Nhận vào một `list` các `string`, mỗi `string` là một literal.
- Trả về một `list` các literal tương ứng đã được sắp xếp theo thứ tự bảng chữ cái.

```
print(literals_sorter(["A", "-C", "B"]))
print(literals_sorter(["-B", "-C", "A"]))
print(literals_sorter(["-D", "C", "-A", "B", '-E']))
print(literals_sorter(["A", "C"]))
```

Output

```
['A', 'B', '-C']
['A', '-B', '-C']
['-A', 'B', 'C', '-D', '-E']
['A', 'C']
```

2.7. Hàm create_clause(literals_1, literals_2=None)

- Nhận vào ít nhất một `list` các literal, tối đa là hai.
- Trả về một mệnh đề tạo ra từ danh sách các literal đã cho, sắp xếp theo thứ tự tăng dần, và loại bỏ các literal trùng nhau.

```
print(create_clause(["A"]))
print(create_clause(["A", "-B", "C"]))
print(create_clause(["-A", "B"], ["C"]))
print(create_clause(['D', '-B', '-B', 'C'], ['E', 'C', '-B', 'A', 'F']))
```

Output

```
A

A OR -B OR C

-A OR B OR C

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

2.8. Hàm is_always_true(clause)

- Nhận vào một mệnh đề, là một `string` gồm các literal nối với nhau bằng chuỗi "OR".
- Trả về `True` nếu trong mệnh đề có hai literal bù nhau, ngược lại trả về `False`.

```
print(is_always_true("A"))
print(is_always_true("A OR B OR C"))
print(is_always_true("A OR B OR -B"))
print(is_always_true("A OR -A OR C"))
print(is_always_true("A OR B OR -B OR D OR -D"))
```

Output

```
False

False

True

True

True
```

2.9. Hàm is_subset(clauses, new_clauses)

- Nhận vào hai `list` mệnh đề, mỗi `list` là danh sách các `string` gồm các mệnh đề (literal nối với nhau bằng chuỗi "OR").
- Trả về `True` nếu mệnh đề thứ nhất có chứa mệnh đề thứ hai hoặc nếu mệnh đề thứ hai rỗng, ngược lại trả về `False`.

```
print(is_subset(["A OR B", "A OR C", "-A OR B", "D"], []))
print(is_subset(["A OR B", "A OR C", "-A OR B", "D"], ["A OR C"]))
print(is_subset(["A OR B", "A OR C", "-A OR B", "D"], ["D"]))
print(is_subset(["A OR B", "A OR C", "-A OR B", "D"], ["A OR -B", "A OR
C"]))
```

Output

```
True

True

True

False
```

3. Hàm hợp giải mệnh đề (solver.py)

3.1. Hàm pl_resolve(clause_1, clause_2)

- Nhận vào hai mệnh đề, là hai `string` gồm các literal nối với nhau bằng chuỗi "OR".
- Trả về một mệnh đề mới nếu có thể hợp giải từ hai mệnh đề đã cho, ngược lại trả về hai mệnh đề ban đầu.
- Mỗi lần chỉ hợp giải **một** cặp literal đối ngẫu.

```
print(pl_resolve("A OR B", "-A"))
print(pl_resolve("A OR B OR C", "-A OR D"))
print(pl_resolve("A OR B", "B OR -C OR D"))
print(pl_resolve("B", "-A OR -B OR C"))
print(pl_resolve("-A OR -B OR C", "A OR B OR D"))
```

Output

```
['B']

['B OR C OR D']

['A OR B', 'B OR -C OR D']

['-A OR C']

['-B OR C OR B OR D']
```

3.2. Hàm pl_resolution(knowledge_base, alpha)

- Mã giả dựa trên Sách Artificial Intelligence: A Modern Approach, Third Edition.

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

- Hàm nhận vào một cơ sở tri thức `knowledge_base` dưới dạng một list các mệnh đề (clause) và `alpha` là một mệnh đề cần kiểm tra.
- Trả về kết quả là `True` nếu thành công, ngược lại trả về `False`. Ngoài ra, kèm theo một `records` là một mảng hai chiều, trong đó chứa các mảng một chiều (`list`), mỗi mảng một chiều chứa các mệnh đề phát sinh ở mỗi bước lặp khi chạy vòng lặp, mỗi mệnh đề là một `string`.

4. Các hàm đọc ghi file (files.py) và main.py

4.1. Hàm files_count(path)

- Nhận vào một `string` là đường dẫn đến thư mục chứa các file input.
- Trả về một số nguyên cho biết số lượng các tập tin có trong thư mục đó.

```
def files_count(path):
    count = 0
    files = list(os.scandir(path))

    for file in files:
        if os.path.isfile(file):
            count += 1

    return count
```

4.2. Hàm read_input_file_content(input_file)

- Nhận vào một `string` là đường dẫn đến tập tin cần đọc thông tin.
- Trả về KB `knowledge_base` và α `alpha`, trong đó
  - `knowledge_base` là một `list` chứa các mệnh đề (clause) trong cơ sở tri thức, mỗi mệnh đề là một `string`.
  - `alpha` là một `string` chứa câu α.

```
def read_input_file_content(input_file):
    fi = open(input_file, 'r')
    alpha = fi.readline().strip()
    data = fi.readline()
    count = int(data)
    knowledge_base = []

    for i in range(0, count):
        data = fi.readline()
        knowledge_base.append(data.strip())

    fi.close()
    return knowledge_base, alpha
```

4.3. Hàm write_to_output_file(input_file, output_file)

- Nhận vào hai `string` lần lượt là đường dẫn đến tập tin input và output.
- Ghi kết quả của thuật toán hợp giải từ hai biến `result`, `records` trả về từ hàm `pl_resolution`, gồm số lượng mệnh đề và các mệnh đề phát sinh được ở mỗi lần lặp, kết quả của quá trình hợp giải (“YES”, “NO”)

```
def write_to_output_file(input_file, output_file):
    knowledge_base, alpha = read_input_file_content(input_file)
    result, records = pl_resolution(knowledge_base, alpha)

    fo = open(output_file, 'w')

    for record in records:
        fo.write(f"{len(record)}\n")
        for r in record:
            fo.write(f"{r}\n")

    fo.write("YES" if result else "NO")
    fo.close()
```

4.4. File main.py

- Import các hàm từ files.py để thực hiện việc đọc ghi file.
- Đọc tất cả các tập tin trong thư mục input, đánh số cho từng tập tin, rồi ghi kết quả vào các tập tin có tên tương ứng trong thư mục output.

```
from files import *

input_folder, output_folder = 'input', 'output'
count = files_count(input_folder)
input_files_names, output_files_names = [], []

for i in range(count):
    input_files_names.append(f"{input_folder}/input{i}.txt")
    output_files_names.append(f"{output_folder}/output{i}.txt")

for i in range(count):
    write_to_output_file(input_files_names[i], output_files_names[i])
```

5. Mô tả, giải thích các kịch bản test

5.1. Kịch bản 1

input0.txt	output0.txt	Ghi chú
B	4	
3	B OR -C	(A OR B) hợp giải với (-A OR -C)
A OR B	A	(A OR B) hợp giải với (negative of B)
-A OR -C	-A OR B	(-A OR -C) hợp giải với (B OR C)
B OR C	C	(B OR C) hợp giải với (negative of B)
	3	
	B	(A OR B) hợp giải với (-A OR B)
	-C	(-A OR -C) hợp giải với (A)
	-A	(-A OR -C) hợp giải với (C)
	1	
	{}	(negative of B) hợp giải với (B)
	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

5.2. Kịch bản 2

input1.txt	output1.txt	Ghi chú
C	5	
5	-B OR C	(A) hợp giải với (-A OR -B OR C)
A	-A OR C OR -D	(-A OR -B OR C) hợp giải với (B OR -D)
-A OR -B OR C	-A OR C OR -E	(-A OR -B OR C) hợp giải với (B OR -E)
B OR -D	-A OR -B	(-A OR -B OR C) hợp giải với (negative of C)
B OR -E	B	(B OR -D) hợp giải với (D)
D	8	
	C OR -D	(A) hợp giải với (-A OR C OR -D)
	C OR -E	(A) hợp giải với (-A OR C OR -E)
	-B	(A) hợp giải với (-A OR -B)
	-A OR C	(-A OR -B OR C) hợp giải với (B)
	-A OR -D	(B OR -D) hợp giải với (-A OR -B)
	-A OR -E	(B OR -E) hợp giải với (-A OR -B)
	C	(-B OR C) hợp giải với (B)
	-A	(-A OR -B) hợp giải với (B)
	3	
	-D	(B OR -D) hợp giải với (-B)
	-E	(B OR -E) hợp giải với (-B)
	{}	(A) hợp giải với (-A)
	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

5.3. Kịch bản 3

input2.txt	output2.txt	Ghi chú
-R	6	
5	P OR -R	(P OR Q) hợp giải với (-Q OR -R)
P OR Q	-R OR -S OR -T	(-Q OR -R) hợp giải với (Q OR -S OR -T)
-Q OR -R	-R OR S	(-Q OR -R) hợp giải với (Q OR S)
Q OR -S OR -T	-Q	(-Q OR -R) hợp giải với (negative of -R)
Q OR S	Q OR -T	(Q OR -S OR -T) hợp giải với (Q OR S)
T	Q OR -S	(Q OR -S OR -T) hợp giải với (T)
	10	
	P	(P OR Q) hợp giải với (-Q)
	-R OR -T	(-Q OR -R) hợp giải với (Q OR -T)
	-R OR -S	(-Q OR -R) hợp giải với (Q OR -S)
	Q OR -R OR -T	(Q OR -S OR -T) hợp giải với (-R OR S)
	-S OR -T	(Q OR -S OR -T) hợp giải với (-Q)
	S	(Q OR S) hợp giải với (-Q)
	Q	(Q OR S) hợp giải với (Q OR -S)
	Q OR -R	(-R OR S) hợp giải với (Q OR -S)
	-T	(-Q) hợp giải với (Q OR -T)
	-S	(-Q) hợp giải với (Q OR -S)
	2	
	-R	(-Q OR -R) hợp giải với (Q)
	{}	(T) hợp giải với (-T)
	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

5.4. Kịch bản 4

input3.txt	output3.txt	Ghi chú
P	3	
5	P OR R OR T	(P OR -Q OR R) hợp giải với (Q OR R OR T)
P OR -Q OR R	-Q OR R	(P OR -Q OR R) hợp giải với (negative of P)
Q OR R OR T	Q OR R	(Q OR R OR T) hợp giải với (-T)
P OR S	3	
S	P OR R	(P OR -Q OR R) hợp giải với (Q OR R)
-T	R OR T	(Q OR R OR T) hợp giải với (-Q OR R)
	R	(-Q OR R) hợp giải với (Q OR R)
	0	
	NO	KB không entails α do không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng

5.5. Kịch bản 5

input4.txt	output4.txt	Ghi chú
-C OR D	8	
5	-B OR C OR D	(A OR C OR D) hợp giải với (-A OR -B OR C)
A OR C OR D	A OR D OR E	(A OR C OR D) hợp giải với (-C OR E)
-A OR -B OR C	A OR C	(A OR C OR D) hợp giải với (-D) from negative of (-C OR D)
-C OR E	-A OR -B OR E	(-A OR -B OR C) hợp giải với (-C OR E)
B OR D	-A OR C OR D	(-A OR -B OR C) hợp giải với (B OR D)
-E	-C	(-C OR E) hợp giải với (-E)
	E	(-C OR E) hợp giải với (C) from negative of (-C OR D)
	B	(B OR D) hợp giải với (-D) from negative of (-C OR D)
	16	
	-B OR C OR D OR E	(A OR C OR D) hợp giải với (-A OR -B OR E)
	C OR D	(A OR C OR D) hợp giải với (-A OR C OR D)
	A OR D	(A OR C OR D) hợp giải với (-C)
	-B OR C	(-A OR -B OR C) hợp giải với (A OR C)
	-A OR -B	(-A OR -B OR C) hợp giải với (-C)
	-A OR C	(-A OR -B OR C) hợp giải với (B)
	-B OR D OR E	(-C OR E) hợp giải với (-B OR C OR D)
	A OR E	(-C OR E) hợp giải với (A OR C)

	-A OR D OR E	(-C OR E) hợp giải với (-A OR C OR D)
	{}	(-E) hợp giải với (E)
	-B OR D	(-B OR C OR D) hợp giải với (-C)
	C OR D OR E	(A OR D OR E) hợp giải với (-A OR C OR D)
	-B OR C OR E	(A OR C) hợp giải với (-A OR -B OR E)
	A	(A OR C) hợp giải với (-C)
	-A OR E	(-A OR -B OR E) hợp giải với (B)
	-A OR D	(-A OR C OR D) hợp giải với (-C)
	YES	KB entails α vì tồn tại mệnh đề rỗng trong KB

5.6. Kịch bản 6

input5.txt	**output5.txt**	**Ghi chú**
-P OR -Q OR S	5	
6	Q OR S OR -T	(P OR Q OR S) hợp giải với (-P OR -T)
P OR Q OR S	P OR Q	(P OR Q OR S) hợp giải với (-S) from negative of (-P OR -Q OR S)
P OR Q OR T	-P OR S OR U	(S OR T OR U) hợp giải với (-P OR -T)
S OR T OR U	T OR U	(S OR T OR U) hợp giải với (-S) from negative of (-P OR -Q OR S)
U OR V	-T	(-P OR -T) hợp giải với (P)
-P OR -T	5	
U	Q OR S OR U	(P OR Q OR S) hợp giải với (-P OR S OR U)
	Q OR S OR T OR U	(P OR Q OR T) hợp giải với (-P OR S OR U)
	S OR U	(S OR T OR U) hợp giải với (-T)
	Q OR -T	(-P OR -T) hợp giải với (P OR Q)
	-P OR U	(-P OR -T) hợp giải với (T OR U)
	3	
	Q OR T OR U	(P OR Q OR T) hợp giải với (-P OR U)
	-P OR Q OR S OR U	(-P OR -T) hợp giải với (Q OR S OR T OR U)
	Q OR U	(-S) from negative of (-P OR -Q OR S) hợp giải với (Q OR S OR U)
	1	
	-P OR Q OR U	(-P OR -T) hợp giải với (Q OR T OR U)
	0	
	NO	KB không entails α do không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng

6. Đánh giá thuật toán

- **Ưu điểm**
  - Việc cài đặt thuật toán không quá phức tạp.
  - Dễ dàng kết luận điều cần suy diễn từ tập KB đã có.
  - Có được lập luận từng bước khi giải các bài toán logic mệnh đề.
- **Nhược điểm**
  - KB phải được viết ở dạng hội chuẩn CNF.
  - Dễ phát sinh các mệnh đề trùng lặp, tốn chi phí cho việc kiểm tra trùng.
    Đề xuất giải pháp: Tại mỗi lần lặp, chỉ tạo và xét các cặp (clause_i, clause_j) từ các mệnh đề vừa mới được phát sinh ở vòng lặp trước đó (nếu có) thay vì tạo các cặp từ tất cả mệnh đề đang có.
  - Chỉ có thể lập luận các bài toán trong lĩnh vực logic mệnh đề, không thể mở rộng sang các lĩnh vực khác (như toán học…)

TÀI LIỆU THAM KHẢO

1. Artificial Intelligence: A Modern Approach, Third Edition