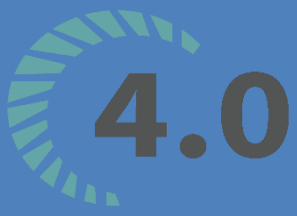
BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO



Sinh viên thực hiện: 20120575 – Nguyễn Khắc Tấn

GV phụ trách: GS.TS. Lê Hoài Bắc

BÀI TẬP MÔN HỌC - MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO

HỌC KỲ II – NĂM HỌC 2022 – 2023

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

BÁO CÁO LAB02

MUC LUC

A. Yêu cầu

STT	Description	Complete
1	Read input data and store in suitable data structure	100%
2	Implementation of resolution method	100%
3	Inference process and results	100%
4	Test cases, report, evaluations	100%

B. Kết quả

I. Chương trình

1. Giải thuật sử dụng:

Giải thuật Robinson

1. Biến đổi tất cả các câu thành dạng CNF
2. Lấy phủ định kết luận, đưa vào KB
3. Lặp
   a. Nếu trong KB có chứa hai mệnh đề phủ định nhau thì trả về ***false //trả về false => vô lý***
   b. Nếu có hai mệnh đề chứa các literal phủ định nhau thì áp dụng hợp giải.
   c. Lặp cho đến khi không thể áp dụng tiếp luật hợp giải.
4. Trả về ***true***

2. Chi tiết chương trình

2.1. *def extract_literals(clause)*
Từ các mệnh đề tách lấy các literal, dựa vào toán tử " OR "

2.2. *def is_complementary(literal_1: str, literal_2: str) -> bool*
Kiểm tra xem hai literal có bổ sung cho nhau không, nếu có trả về True

2.3. *def literals_sorter(literals: List[str]) -> List[str]*
Xắp xếp danh sách các literal theo bảng chữ cái, literal có dấu "-" đứng trước

2.4. *def remove_duplicates(literals: List[str]) -> List[str]*
Xóa các literal trùng nhau trong mệnh đề

2.5. *def create_clause(literals_1: List[str], literals_2: List[str]) -> List[str]*
Tạo một mệnh đề mới từ 2 mệnh đề

2.6. *def is_always_true(clause: List[str]) -> bool*
Kiểm tra một mệnh đề đúng hay không (chứa một cặp bổ sung), nếu đúng trả về True

2.7. *def is_subset(clauses1: List[List[str]], clauses2: List[List[str]]) -> bool*
Lấy hai danh sách mệnh đề, trả về đúng nếu tất cả mệnh đề trong danh sách 1 xuất hiện trong danh sách 2

2.8. *def negate_literal(literal: str) -> str*
Trả về phủ định của một literal

2.9. *def read_input(input_file: str) -> Tuple[str, List[str]]*
Đọc alpha, n, knowledge base từ file input

2.10. *def write_output(output_file: str, generated_clauses: List[List[str]], entails_alpha: bool)*
Ghi ra file output kết quả

2.11. *def pl_resolve(ci: List[str], cj: List[str]) -> List[str]*
Từ hai mệnh đề, tạo ra kết quả của hai mệnh đề đó

2.12. *def pl_resolution(kb: List[str], alpha: str) -> Tuple[List[str], bool]*
Thực hiện thuật toán:

B1. Thêm phủ định alpha vào tập mệnh đề (clauses)

B2. Lặp

- Gán record = [] (lưu các tập mệnh đề)
- Tạo ra danh sách các cặp mệnh đề có thể kết hợp với nhau (pairs) từ tập mệnh đề (clauses)
- Từ các cặp mệnh đề:
  - Kiểm tra pl_resolve của cặp mệnh đề không đúng và chưa xuất hiện trong tập danh sách các cặp mệnh đề mới (new_clauses), nếu đúng thì thêm cặp danh sách mệnh đề (ci, cj) vào new_clauses
- Kiểm tra xem new_clauses có nằm trong clauses không (không tạo ra mệnh đề mới):
  - Nếu đúng, trả về generated_clauses. False, kết thúc chương trình
- Kiểm tra xem các mệnh đề (clause) trong new_clauses không nằm trong tập mệnh đề clauses và không đúng
  - Thêm mệnh đề đó (clause) vào tập mệnh đề clauses và record
- Thêm record vào generated_clauses
- Kiểm tra xem "{}" có nằm trong trong record không
  - Nếu có, trả về tập generated_clauses, True, kết thúc chương trình

2.13. *def main()*
Hàm chính

II. 5 test cases

1. Test case 1:

1.1. Input

```
-A
4
-A OR B
-C OR B
A OR C OR -B
-B
```

1.2. Output

```
3
-A
B
-C

4
-B OR C
C OR A
-B OR A
{}

YES
```

2. Test case 2:

2.1. Input

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

2.2. Output

```
2
E OR A OR D
-C

1
A OR D

0

NO
```

3. Test case3:

3.1. Input

```
D
5
A OR B
-A OR C
-B OR C
-C OR D
-D
```

3.2. Output

```
5
C OR B
C OR A
D OR -A
-B OR D
-C

8
D OR B
A OR D
C
-A
-B
C OR D
B
A

2
D
{}

YES
```

4. Test case4:

4.1. Input

```
C
4
A OR B
B OR -C
A OR -B OR C
-B
```

4.2. Output

```
3
C OR A
A
-B OR A

1
-C OR A

0

NO
```

5. Test case5:

5.1. Input

```
-P OR -Q OR S
6
P OR Q OR S
P OR Q OR T
S OR T OR U
U OR V
-P OR -T
U
```

5.2. Output

```
2
Q OR -T OR S
U OR S OR -P

3
Q OR U OR S
Q OR S OR P
T OR Q OR U OR S

1
Q OR U OR S OR -P

0

NO
```

III. Ưu nhược điểm, cách giải quyết vấn đề

1. Ưu nhược điểm

1.1. Ưu điểm

Có thể tìm thấy một bằng chứng cho bất kỳ suy luận hợp lệ trong logic mệnh đề

Phương pháp giải hợp lý, có thể đưa ra những kết luận có giá trị logic

Tương đối hiệu quả, thiết thực cho các ứng dụng trong thế giới thực

1.2. Nhược điểm

Phương pháp giải có thể phức tạp và khó hiểu, tốn nhiều chi phí

Không thể sử dụng cho các logic phức tạp

Có thể sai trong một số trường hợp

2. Cách giải quyết vấn đề

Có thể sai trong một số trường hợp:

Ví dụ: (P and -P) entails Z

Từ danh sách mệnh đề ban đầu, chọn ra hai mệnh đề mà trong các mệnh đề ấy có các literal phủ định nhau. Ta gom hai mệnh đề con đó lại thành một mệnh đề. Khi nào trong danh sách mệnh đề ban đầu có hai mệnh đề phủ định nhau, hay danh sách mệnh đề không thể gom được nữa thì kết thúc bài toán.