ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LỚP 21CNTN

BÁO CÁO ĐỒ ÁN 2
LOGIC

Môn: Cơ sở trí tuệ nhân tạo

Người thực hiện:

21120139 - Lê Long Trường Thịnh

Giảng viên hướng dẫn:
GS.TS Lê Hoài Bắc
GVPT. Nguyễn Duy Khánh

Thành phố Hồ Chí Minh – tháng 11 năm 2023

MỤC LỤC

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH ĐỒ ÁN

STT	Đặc tả tiêu chí	Điểm
1	Đọc dữ liệu đầu vào và lưu trong cấu trúc dữ liệu phù hợp	0.5
2	Cài đặt giải thuật hợp giải trên logic mệnh đề	1
3	Các bước suy diễn phát sinh đủ mệnh đề và kết luận đúng	2.5
4	Tuân thủ mô tả định dạng của đề bài	0.5
5	Báo cáo test case và đánh giá	0.5
	Tổng điểm tự đánh giá	5.0

I. Tổng quan về lập trình bài Logic:

1. Đặc tả dữ liệu đầu vào và đầu ra:

A, Dữ liệu đầu vào: KB và $\propto$ theo dạng chuẩn CNF được lưu trong các tập tin **input.txt**. Tập tin có định dạng quy ước như sau:

- Dòng đầu tiên chứa câu $\propto$
- 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

B, Dữ liệu đầu ra: Tập hợp mệnh đề được phát sinh trong quá trình hợp giải và câu kết luận được lưu trong các 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 "{}"
- Dòng cuối cùng trình bày câu kết luận, tức là trả lời câu hỏi "KB entails $\propto$?". In YES nếu KB entails $\propto$. 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).

C, Một số lưu ý:

- Các literal trong cùng mệnh đề (đối với cả dữ liệu đầu vào và đầu ra) được xếp theo thứ tự chữ cái
- Kiểm tra điều kiện suy dẫn ở cuối mỗi vòng lặp, tức là khi đã phát sinh hết câu mới từ KB hiện hành, chứ không kiểm tra sau mỗi lần phát sinh một câu
- Các mệnh đề dạng *A* v *B* v *-B* có chân trị *True* vì tương đương với *A* $\vee$ *True*. Những mệnh đề như thế này vô ích cho việc suy dẫn và do đó có thể bỏ đi

2. Sơ lược về code:

Chương trình thực thi gồm 3 tập tin:

- main.py: là script Python chính sẽ chạy để thực hiện quá trình hợp giải logic mệnh đề. Nó có các chức năng chính sau:

- Đọc dữ liệu đầu vào: Hàm read_input_file đọc từng file trong thư mục INPUT. Nó phân tách file thành các mệnh đề và lưu trữ chúng cùng với câu α (alpha) để xử lý tiếp theo.
- Xử lý hợp giải: Hàm main gọi hàm pl_resolution từ file resolution.py để thực hiện quá trình hợp giải. Nó truyền cơ sở tri thức (KB) và α vào hàm này và nhận về kết quả hợp giải.

- Ghi kết quả ra file: Sau khi quá trình hợp giải hoàn tất, hàm write_output_file sẽ ghi kết quả vào các file trong thư mục OUTPUT. Điều này bao gồm kết quả của mỗi bước hợp giải và kết luận cuối cùng (YES/NO).

- resolution.py: chứa hàm pl_resolution, là cả quá trình hợp giải logic mệnh đề. Chức năng chính của nó bao gồm:

- Thuật toán hợp giải: Hàm pl_resolution thực hiện thuật toán hợp giải logic mệnh đề, một phần quan trọng trong logic và trí tuệ nhân tạo. Nó xử lý các mệnh đề từ KB cùng với phủ định của α, và cố gắng tìm ra mệnh đề rỗng, từ đó suy ra liệu KB có suy ra được α hay không.
- Xử lý logic mệnh đề: File này chứa các hàm phụ trợ như split_clause và negate_literal để xử lý các mệnh đề trong KB và chuyển đổi chúng thành dạng phù hợp để thực hiện hợp giải.

- run.sh: dùng để tự động hóa quá trình chạy main.py cho nhiều file đầu vào. Đây là chức năng chính của nó:

- Tự động hóa quá trình chạy: Script này tự động xử lý tất cả các file trong thư mục INPUT, chạy main.py cho mỗi file, và đảm bảo rằng kết quả được lưu vào thư mục OUTPUT.
- Kiểm tra và chuẩn bị môi trường: Script kiểm tra sự tồn tại của thư mục INPUT và OUTPUT, tạo thư mục OUTPUT nếu nó chưa tồn tại.
- Tính linh hoạt: Bạn có thể dễ dàng chạy toàn bộ bộ test mà không cần phải thực hiện từng bước một cách thủ công.

II. Giải thích cách tư duy lập trình:

1. Xử lý input:

A, Đọc File Input: Khi main.py được chạy, nó sẽ đọc từng file trong thư mục INPUT thông qua hàm read_input_file.

- Mỗi file input.txt sẽ chứa một dòng đầu tiên biểu diễn câu α (alpha).
- Dòng thứ hai chứa số nguyên n, biểu diễn số lượng mệnh đề trong Cơ sở Tri thức (KB).
- Các dòng tiếp theo chứa n mệnh đề của KB.

B,Chuyển Dữ Liệu Đến Thuật Toán Hợp Giải: Sau khi đọc, KB và α sẽ được chuyển đến hàm pl_resolution trong resolution.py.
Hàm này xử lý KB và α để xác định liệu KB có suy ra được α hay không.
C,Giải thích sơ về INPUT:

1) input_ 1.txt:
   a) Cơ sở tri thức (KB):

i) A OR B: Mệnh đề này nói rằng ít nhất một trong hai literal A hoặc B phải đúng.
ii) B OR -C: Mệnh đề này nói rằng B đúng hoặc C phải sai.
iii) -A OR C: Mệnh đề này nói rằng nếu A sai thì C phải đúng.

b) Alpha (α): B
c) Mục đích: Kiểm tra xem từ KB có suy ra được B là đúng hay không.

2) input_ 2.txt:
a) Cơ sở tri thức (KB):
i) A OR -B: A đúng hoặc B phải sai.
ii) -A OR B: Nếu A sai thì B phải đúng.
iii) B OR -C: B đúng hoặc C phải sai.
iv) -B OR C: Nếu B sai thì C phải đúng.
b) Alpha (α): C
c) Mục đích: Xác định liệu C có là kết quả suy diễn từ KB hay không.

3) input_ 3.txt:
a) Cơ sở tri thức (KB):
i) A: A phải đúng.
ii) -A OR B: Nếu A sai thì B phải đúng.
iii) -B OR -C: Nếu B đúng thì C phải sai và ngược lại.
iv) C OR D: C đúng hoặc D phải đúng.
v) -D OR E: Nếu D sai thì E phải đúng.
b) Alpha (α): E
c) Mục đích: Kiểm tra xem E có được suy ra từ KB không.

4) input_ 4.txt:
a) Cơ sở tri thức (KB):
i) -A OR -B: A phải sai hoặc B phải sai.
ii) A OR B: A phải đúng hoặc B phải đúng.
iii) -B OR C: Nếu B sai thì C phải đúng.
iv) A OR -C: A đúng hoặc C phải sai.
v) -A OR D: Nếu A sai thì D phải đúng.
b) Alpha (α): -D
c) Mục đích: Kiểm tra xem D có phải là sai từ KB không.

5) input_ 5.txt:
a) Cơ sở tri thức (KB):
i) A OR B OR C: Ít nhất một trong ba literal A, B, hoặc C phải đúng.
ii) -A OR -B: A phải sai hoặc B phải sai, và ngược lại.
iii) -B OR -C OR D: Nếu B và C cùng đúng thì D phải đúng, hoặc ít nhất một trong B hoặc C phải sai.
iv) -D OR E: Nếu D sai thì E phải đúng.
v) -E OR F: Nếu E sai thì F phải đúng.

b) Alpha (α): A
c) Mục đích: Xác định liệu A có là kết quả suy diễn từ KB hay không.

2. Xử lý output:

A,Quá Trình Hợp Giải: Hàm pl_resolution thực hiện quá trình hợp giải và trả về hai thứ:

- Một danh sách các bước trong quá trình hợp giải. Mỗi bước là một tập hợp các mệnh đề mới được tạo ra hoặc {} nếu một mệnh đề rỗng được tạo ra.
- Một chuỗi "YES" hoặc "NO" tùy thuộc vào việc liệu KB có suy ra được α hay không.

B, Ghi File Output: Hàm write_output_file trong main.py sau đó sẽ ghi các kết quả này vào file đầu ra trong thư mục OUTPUT.

- Mỗi file output.txt sẽ chứa kết quả của mỗi bước trong quá trình hợp giải, bao gồm cả số lượng mệnh đề mới tạo ra và các mệnh đề đó.
- Cuối cùng, file output.txt sẽ chứa kết luận "YES" hoặc "NO".

3. Xử lý resolution.py:

File resolution.py chứa hàm pl_resolution, đây là hàm trung tâm thực hiện thuật toán hợp giải logic mệnh đề. Dưới đây là giải thích chi tiết về từng phần của hàm này:

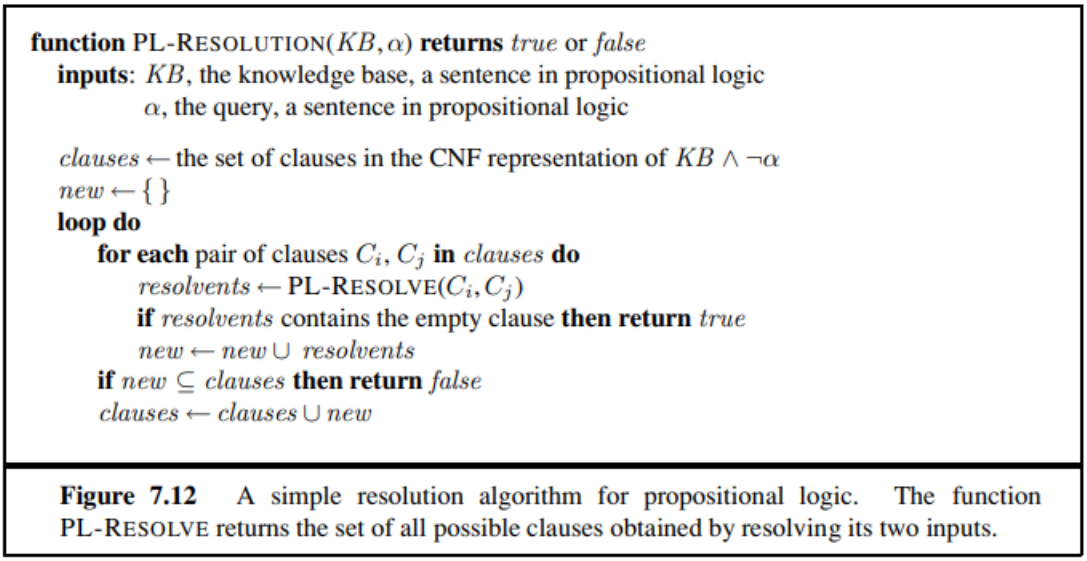
A, Hàm pl_resolution(kb, α):

Hàm này nhận vào một Cơ sở Tri thức (KB) và một mệnh đề α (alpha). Mục tiêu của hàm là xác định liệu từ KB có thể suy ra α hay không thông qua quá trình hợp giải.

- split_clause(clause): Hàm này chia một mệnh đề thành các literal riêng biệt. Mỗi mệnh đề được tách ra thành một frozenset của các literal, cho phép xử lý dễ dàng hơn trong quá trình hợp giải.
- negate_literal(literal): Hàm này phủ định một literal. Nếu literal bắt đầu bằng '-', nó loại bỏ dấu '-' để tạo ra phủ định của nó; nếu không, nó thêm dấu '-' vào trước literal.
- resolve(clause1, clause2): Hàm này thực hiện hợp giải giữa hai mệnh đề. Nếu tìm thấy một cặp literal phủ định lẫn nhau trong hai mệnh đề, nó loại bỏ cặp literal này và tạo ra một mệnh đề mới từ phần còn lại của cả hai mệnh đề.

B,Quá Trình Hợp Giải:

Mã giả của hợp giả Robinson:



```
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 ∧ ¬α
  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.

Trong vòng lặp while True, thuật toán xử lý các mệnh đề trong KB: (có thể xem rõ hơn ở trong file resolution.py)

- Tạo Các Cặp Mệnh Đề: Đầu tiên, tạo ra tất cả các cặp mệnh đề có thể từ KB để hợp giải chúng.
- Hợp Giải Các Cặp Mệnh Đề: Với mỗi cặp, sử dụng hàm resolve để thử hợp giải. Nếu tạo ra mệnh đề mới, kiểm tra xem nó có phải là mệnh đề rỗng không.
- Kết Quả Hợp Giải:
  - Nếu một mệnh đề rỗng được tạo ra, nghĩa là KB suy ra được α, hàm trả về kết quả 'YES' cùng với quá trình hợp giải dẫn đến mệnh đề rỗng.
  - Nếu không tìm thấy mệnh đề mới hoặc tất cả các mệnh đề mới đều đã tồn tại trong KB, hàm trả về 'NO', cho thấy không thể suy ra α từ KB.
- Lưu Trữ Quá Trình Hợp Giải: Hàm duy trì một danh sách resolution_process để theo dõi các bước của quá trình hợp giải, bao gồm các mệnh đề mới được tạo ra ở mỗi bước.

C,Kết Thúc Quá Trình:

Cuối cùng, hàm pl_resolution trả về quá trình hợp giải (được lưu trong resolution_process) và kết luận cuối cùng ('YES' hoặc 'NO'). Điều này cho phép main.py ghi thông tin này vào các file output tương ứng.

4. Xử lý main.py:

File main.py là script Python chính để chạy thuật toán hợp giải. Nó chứa các hàm chính để đọc dữ liệu đầu vào, gọi hàm hợp giải và ghi kết quả vào file đầu ra. Dưới đây là giải thích chi tiết cho từng phần của file:

A,Hàm read_input_file(file_path):

- Chức năng: Đọc file đầu vào và phân tích cú pháp dữ liệu.
- Cách thức hoạt động:

- Đọc từng dòng trong file.
- Dòng đầu tiên chứa câu α (alpha).
- Dòng thứ hai chứa số nguyên n, là số lượng mệnh đề trong Cơ sở Tri thức (KB).
- Các dòng tiếp theo chứa n mệnh đề của KB.
- Trả về KB và α để sử dụng sau này trong quá trình hợp giải.

B,Hàm write_output_file(file_path, resolution_process, conclusion):

- Chức năng: Ghi kết quả của quá trình hợp giải vào file đầu ra.
- Cách thức hoạt động:
  - Mở file đầu ra để ghi.
  - Ghi từng bước của quá trình hợp giải.
  - Nếu bước là mệnh đề rỗng "{ }", chỉ ghi nó ra.
  - Nếu không, ghi số lượng mệnh đề trong bước này, sau đó ghi từng mệnh đề.
  - Cuối cùng, ghi kết luận của quá trình hợp giải ('YES' hoặc 'NO').

C,Hàm main(input_folder='INPUT', output_folder='OUTPUT'):

- Chức năng: Là điểm khởi đầu của chương trình, xử lý từng file đầu vào và tạo ra file đầu ra tương ứng.
- Cách thức hoạt động:
  - Đảm bảo rằng thư mục đầu ra tồn tại.
  - Xử lý từng file trong thư mục đầu vào.
  - Đối với mỗi file:
    - Đọc dữ liệu từ file đầu vào.
    - Gọi hàm pl_resolution để thực hiện quá trình hợp giải với dữ liệu này.
    - Ghi kết quả của quá trình hợp giải vào file đầu ra trong thư mục OUTPUT, với tên file tương ứng với file đầu vào.

TÀI LIỆU THAM KHẢO

Zarembo, Imants & Kodors, Sergejs. (2015). Pathfinding Algorithm Efficiency Analysis in 2D Grid. Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference. 2. 46. 10.17770/etr2013vol2.868.

CS 294-5: Statistical Natural Language Processing course, University of California, Berkeley.

Tham khảo mã giả hợp giải Robinson từ Sách Artificial Intelligence: A Modern Approach, Third Edition, Chương 7, Hình 7.12, hàm PL-RESOLUTION.

Resolution Completeness and clauses in Artificial Intelligence
https://www.geeksforgeeks.org/resolution-completeness-and-clauses-in-artificial-intelligence/

Artificial Intelligence Propositional Logic
https://www.user.tu-berlin.de/mtoussai/teaching/15-ArtificialIntelligence/09-propositionalLogic.pdf