fit@hcmus

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

CƠ SỞ TRÍ TUỆ NHÂN TẠO

LAB 02 - LOGIC

Giảng viên: Lê Hoài Bắc

Lớp: 19TN

Người thực hiện:

Họ và tên	MSSV
Đặng Thái Duy	19120491

MỤC LỤC

I. Cấu trúc thư mục SRC

1. **Thư mục input**
   Chứa các file inputk.txt chứa dữ liệu cần kiểm nghiệm, các file input này có định dạng như sau:
   - Dòng đầu tiên chưa 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
2. **Thư mục output**
   Chứa các file outputk.txt tương ứng chứa kết quả chạy chương trình, các file output này có định dạng 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 $M_1, M_2, \dots, M_n$ 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).
3. **File main.py**
   Chứa toàn bộ mã nguồn của chương trình, khi run file main.py thì với từng file inputk.txt tương ứng trong thư mục input sẽ in ra kết quả trong file outputk.txt tương ứng trong thư mục output.
   Hàm main thực hiện những thao tác sau đây:
   - Đọc dữ liệu đầu vào và lưu trong cấu trúc dữ liệu phù hợp
   - Gọi hàm pl_resolution để thực thi giải thuật hợp giải
   - Ghi dữ liệu đầu ra vào tập tin đầu ra tương ứng theo định dạng hợp lệ

II. Cài đặt mã nguồn

Các hàm được cài đặt bên trong file main.py như sau

1. **negative_literal(x)**
   Trả về phủ định của 1 literal x (ví dụ $A \rightarrow -A$ và ngược lại)
2. **negative_alpha(alpha)**
   Trả về 1 mệnh đề dạng list 2 chiều tạo bởi phủ định các literal có trong α, trước đó α đã được chuẩn hóa (xóa các literal trùng và sắp xếp lại theo alphabet)
   Ví dụ: $\alpha =$ ["B", "A", "-C"] → trả ra kết quả [["-A"], ["-B"], ["C"]]
3. **pl_resolve(clause_x, clause_y)**
   Trả về resolvent hợp giải 2 mệnh đề là đối số được truyền vào, có 2 trường hợp:

- Nếu chứa cặp đối ngẫu, hợp giải bằng cách xóa cặp đó đi và kiểm tra trong kết quả resolvent còn tồn tại cặp đối ngẫu nào không:
    - nếu không thì trả ra kết quả là resolvent sau khi được chuẩn hóa
    - nếu có tức là khi đó resolvent là 1 mệnh đề luôn đúng, khi đó trả ra kết quả là None. Ví dụ: B OR -B OR C = TRUE OR C = TRUE
- Nếu không tìm thấy bất kì cặp đối ngẫu nào để hợp giải thì trả ra None

4. **resolve_clause(_kb)**

   Trả về new là 1 list chứa các mệnh đề hợp giải mới không nằm trong KB
   - Ở đây hàm sẽ duyệt các cặp mệnh đề và tiến hành hợp giải bằng hàm pl_resolve() nêu trên
   - Kiểm tra kết quả hợp giải là resolvent có thỏa mãn các điều kiện cho phép không (không None, không nằm trong KB và new) → điều này đảm bảo các mệnh đề mới được sinh ra không bao giờ bị trùng với các mệnh đề đã có sẵn hoặc tìm được trước đó.
   - Nếu thỏa mãn thì thêm resolvent vào new
   - Tiếp tục đến khi duyệt hết tất cả các cặp mệnh đề và trả về new.
5. **pl_resolution(_kb, _alpha, result)**

   Hàm xử lý chính việc hợp giải $KB \wedge \neg\alpha$
   - Dùng hàm resolve_clause() để hỗ trợ việc tạo ra 1 list mệnh đề hợp giải mới trong mỗi vòng lặp
   - Lưu các kết quả tìm được vào biến result
   - Kiểm tra điều kiện kết thúc vòng lặp và trả về True/False tương ứng
6. **read_input(filename)**

   Trả về α và KB được đọc từ input là filename, trong đó:
   - α có dạng list 1 chiều gồm các phần tử là literal
   - KB có dạng list 2 chiều gồm các phần tử là các mệnh đề, các mệnh đề này cũng có dạng là list 1 chiều gồm các phần tử là literal

   Ví dụ: α = [“A”, “B”, “-C”] ; KB = [["-A", "B"], ["-B", "D"], ["-A", "C"]]
7. **write_output(filename, result)**

   Ghi kết quả trong result vào output là filename

   Ví dụ: result = ['3', '-A', 'B', '-C', '4', '-B OR C', 'A OR C', 'A OR -B', '{}', 'YES']
8. **main()**

   Hàm chính để chạy chương trình, sẽ làm những công việc sau:
   - Duyệt từng file inputk.txt trong thư mục input để lấy dữ liệu α và KB
   - Thực hiện hợp giải $KB \wedge \neg\alpha$ để lấy ra kết quả
   - In kết quả vừa tìm được vào file outputk.txt tương ứng trong thư mục output
   - Đồng thời in ra màn hình console kết quả KB entails α hay không? (YES/NO)

III. Test case mẫu

1. Test case 1

input1.txt	output1.txt	Ghi chú
C OR D	6	(negative of α gồm các literals: -C, -D)
5	A OR C OR -D	(B OR C) hợp giải với (A OR -B OR -D)
B OR C	C	(B OR C) hợp giải với (-B)
-A OR D	B	(B OR C) hợp giải với (-C)
B OR -D	-A OR B	(-A OR D) hợp giải với (B OR -D)
A OR -B OR -D	-A	(-A OR D) hợp giải với (-D)
-B	A OR -D	(B OR -D) hợp giải với (A OR -B OR -D)
	4	
	-B OR -D	(A OR -B OR -D) hợp giải với (-A)
	{}	(-B) hợp giải với (B)
	B OR C OR -D	(A OR C OR -D) hợp giải với (-A OR B)
	C OR -D	(A OR C OR -D) hợp giải với (-A)
	YES	KB entail alpha vì tồn tại mệnh đề rỗng trong KB

2. Test case 2

input2.txt	output2.txt	Ghi chú
-U	4	(negative of α gồm các literals: U)
5	P OR R	(P OR Q) hợp giải với (-Q OR R)
P OR Q	P OR S	(P OR Q) hợp giải với (-Q OR S)
-Q OR R	Q OR U	(P OR Q) hợp giải với (-P OR U)
-Q OR S	-Q OR U	(-Q OR R) hợp giải với (-R OR U)
-P OR U	3	
-R OR U	P OR U	(P OR Q) hợp giải với (-Q OR U)
	R OR U	(-Q OR R) hợp giải với (Q OR U)
	S OR U	(-Q OR S) hợp giải với (Q OR U)
	0	
	NO	KB không entail alpha vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng|

3. Test case 3

input3.txt	output3.txt	Ghi chú
E OR F OR H	4	(negative of α gồm các literals: -E, -F, -H)

4	E OR F	(F OR -G) hợp giải với (E OR G)
F OR -G	-G OR H	(F OR -G) hợp giải với (-F OR H)
E OR G	-G	(F OR -G) hợp giải với (-F)
-E	G	(E OR G) hợp giải với (-E)
-F OR H	5	
	F	(F OR -G) hợp giải với (G)
	E OR H	(E OR G) hợp giải với (-G OR H)
	E	(E OR G) hợp giải với (-G)
	H	(-G OR H) hợp giải với (G)
	{}	(-G) hợp giải với (G)
	YES	KB entail alpha vì tồn tại mệnh đề rỗng trong KB

4. Test case 4

input4.txt	output4.txt	Ghi chú
-X	5	(negative of α gồm các literals: X)
3	-X OR Y	(Y OR Z) hợp giải với (-X OR Y OR -Z)
Y OR Z	-X OR Z	(Y OR Z) hợp giải với (-X OR -Y)
-X OR Y OR -Z	-X OR -Z	(-X OR Y OR -Z) hợp giải với (-X OR -Y)
-X OR -Y	Y OR -Z	(-X OR Y OR -Z) hợp giải với (X)
	-Y	(-X OR -Y) hợp giải với (X)
	4	
	Y	(Y OR Z) hợp giải với (Y OR -Z)
	Z	(Y OR Z) hợp giải với (-Y)
	-X	(-X OR -Y) hợp giải với (-X OR Y)
	-Z	(X) hợp giải với (-X OR -Z)
	1	
	{}	(X) hợp giải với (-X)
	YES	KB entail alpha vì tồn tại mệnh đề rỗng trong KB

5. Test case 5

input5.txt	output5.txt	Ghi chú
-C OR D OR -E	4	(negative of α gồm các literals: C, -D, E)
4	A OR C OR D	(A OR B OR C) hợp giải với (-B OR D)
A OR B OR C	A OR E	(A OR -D) hợp giải với (D OR E)
A OR -D	A OR -B	(A OR -D) hợp giải với (-B OR D)
D OR E	-B	(-B OR D) hợp giải với (-D)

-B OR D	1	
	A OR C	(A OR B OR C) hợp giải với (A OR -B)
	0	
	NO	KB không entail alpha vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng|

IV. Đánh giá

Dưới đây là mã giả của giải thuật PL-RESOLUTION trong tài liệu tham khảo của bộ môn Cơ sở Trí tuệ nhân tạo

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*

1. **Ưu điểm**
   Giải thuật luôn tìm ra kết quả đầy đủ và chính xác
2. **Khuyết điểm**
   - Trong mỗi vòng lặp, phải duyệt hết tất cả các cặp mệnh đề C_i, C_j dù rằng những lần duyệt trước đó có thể đã được xét đến rồi → làm lãng phí tài nguyên hệ thống, tốn nhiều thời gian hơn
   - Giải thuật trên chỉ giải được trong phạm vi các mệnh đề KB và α được đưa vào đã được chuẩn hóa dưới dạng CNF
3. **Giải pháp**
   - Thay vì lấy C_i hợp giải với tất cả mọi mệnh đề C_j $(j \neq i)$ trong *clauses* thì với lần lặp thứ n, ta hợp giải C_i với C_j' trong đó C_j' ở đây là các mệnh đề mới được tạo ra ở lần lặp trước đó (n-1) → với cách này sẽ tránh được việc các lần lặp sau xét trùng ở các lần lặp trước đó

- Để mở rộng việc hợp giải trên logic mệnh đề cho mọi đầu vào KB và α bất kỳ, ta cần xây dựng thêm hàm *standard_cnf(clause)* để chuẩn hóa các mệnh đề về đúng dạng chuẩn CNF, rồi sau đó mới tiến hành hợp giải
  Ví dụ: $A \wedge B \rightarrow C$ chuẩn hóa về dạng CNF là $\neg A \vee \neg B \vee C$