TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM

KHOA CÔNG NGHỆ THÔNG TIN

PROJECT 02

HỢP GIẢI LOGIC MỆNH ĐỀ

Học phần: Nhập môn cơ sở trí tuệ nhân tạo

Lớp: 19_22

Họ và tên: Lê Kiệt

MSSV: 19120554

MỤC LỤC

I. THIẾT KẾ 1

1. Cấu trúc dữ liệu mới phục vụ cho hợp giải (class) 1

a) Class Literal 1

b) Class Clause 1

2. Các hàm chính của chương trình hợp giải 2

a) Hàm đọc input 2

b) Hàm hợp giải 2 câu: **resolve_pair**(c1,c2) 2

c) Hàm pl_resolution: **pl_resolution**(alpha, KB) 3

II. KỊCH BẢN KIỂM THỬ 3

1. input9.txt 4

2. input8.txt 4

3. input4.txt 4

4. input5.txt 5

5. input3.txt 5

III. ĐÁNH GIÁ GIẢI THUẬT HỢP GIẢI 6

I. THIẾT KẾ

1. Cấu trúc dữ liệu mới phục vụ cho hợp giải (class)

a) Class Literal

- Mục đích: lưu trữ duy nhất 1 literal, ví dụ: -A hoặc A. Constructor nhận vào chuỗi string 1 phần tử (VD: 'A') hoặc 2 phần tử (VD: '-A') và chuỗi string này được lưu vào thuộc tính **value** của 1 thể hiện **Literal**
- Phương thức chính:
  - **not_literal(self)**: hàm trả về phủ định của literal hiện tại. VD: A → -A hoặc -A → A

- Phương thức overload nhằm phục vụ cho class Clause (đề cập ở mục b):
  - **__repr__(self)**: hàm định nghĩa cách xuất ra 1 literal cho file output
  - **__eq__(self, other)**: hàm định nghĩa thế nào là 2 literal bằng nhau. 2 literal bằng nhau khi thuộc tính **value** của 2 thể hiện bằng nhau
  - **__hash__(self)**: lưu trữ địa chỉ thể hiện ở dưới hệ thống. Phục vụ cho việc gọi set() của các literal trong Python

b) Class Clause

- Mục đích: lưu trữ 1 câu chứa 1 hoặc nhiều literal, câu luôn được đảm bảo theo thứ tự từ điển. Câu được thiết kế để chỉ chứa phép OR và không có dấu ngoặc
- Constructor: nhận vào 1 list các string, mỗi string thể hiện cho 1 literal, construct sẽ sắp xếp list string này theo thứ tự từ điển (bỏ qua dấu '-' khi sắp xếp) xong biến đổi list string mới sắp xếp này thành list các thể hiện **Literal** và gán vào thuộc tính **value** cho 1 thể hiện **Clause**. Điều này đảm bảo các câu được in ra theo thứ tự từ điển
- Phương thức chính:
  - Phương thức **not_clause(self)**: trả về phủ định của câu. Phủ định của 1 câu (như đã nói thì câu chỉ chứa OR) là việc lấy phủ định của từng literal trong câu (tức gọi đến **Literal.not_literal()**), và vì 1 câu chứa OR sau khi phủ định sẽ chứa toàn AND nên tách mỗi literal trong câu AND thành 1 clause riêng biệt. Phương thức trả về 1 list các câu **Clause** sau khi lấy phủ định câu gốc
    Minh họa:
    ```
    not_clause([-A,B])  # -A OR B
    =   [[A], [-B]]     # A AND -B
    ```

  - Phương thức **drop_literal(self, literal)**: câu mới trả về bỏ đi 1 **literal** trong câu cũ. Nếu có nhiều hơn 1 literal giống nhau thì bỏ đi literal ở vị trí trái nhất của clause

  - Phương thức **merge(self, other)**: kết quả trả về chính là việc nối 2 câu lại với nhau. Lưu ý phương thức này sẽ rút gọn câu kết quả nếu 2 câu có chung literal. Câu kết quả luôn được sắp xếp theo thứ tự từ điển và là câu ngắn nhất
    VD: clause_1 = [A,B] # A OR B
    clause_2 = [B,-C] # B OR -C
    → clause_1.merge(clause_2) = [A,B,-C]

- Phương thức **is_true(self)**: trả về True nếu câu tương đương với chân trị True, tức có 2 literal giống nhau bất kỳ trái dấu (VD: -A OR B OR A = True OR B = True); ngược lại trả về False

- Phương thức **is_empty(self)**: kiểm tra câu có là câu rỗng

- Các phương thức overloads:
  - Phương thức **__repr__**: định nghĩa lại output của thể hiện clause, theo đề là các literal ngăn cách bởi chữ 'OR'
  - Phương thức **__eq__**: định nghĩa thế nào là 2 câu bằng nhau. 2 câu bằng nhau khi mọi literal của 2 câu phải bằng nhau
  - Phương thức **__hash__**: hỗ trợ cho việc lấy set() các câu trong Python

2. Các hàm chính của chương trình hợp giải

a) Hàm đọc input

- Input: đường dẫn file tới file input*k*.txt
- Output: alpha và KB đọc được từ file
- Mô tả:
  - Đọc vào dòng đầu (cũng là câu alpha, alpha không chứa AND hay ngoặc) → xóa ký tự xuống dòng và chữ OR trong chuỗi string → split chuỗi dựa trên 1 hay nhiều khoảng trắng thành 1 list các chuỗi literal và tạo được 1 câu **alpha** thuộc kiểu **Clause** từ list này
  - Đọc vào dòng 2 tương ứng với số **n** - số mệnh đề có trong KB
  - n dòng tiếp theo, mỗi dòng cũng đọc vào và xử lý tương tự như alpha, kết quả mỗi dòng là 1 câu **clause** thuộc kiểu **Clause**, thêm câu vừa tạo vào **KB**
  - Sau cùng, trả về **alpha** và **KB**

b) Hàm hợp giải 2 câu: **resolve_pair**(c1,c2)

- Input: 2 câu **c1**,**c2** thuộc kiểu Clause
- Output: tất cả các câu có thể có từ việc hợp giải **c1** và **c2**. Nếu 2 câu không có gì để hợp giải thì trả về list gồm 1 phần tử None
  - VD 1: c1 = [A,B], c2 = [C,D]
    → resolve_pair(c1,c2) = [None]
  - VD 2: c1 = [A,B,C], c2 = [-B,-C,F] → triệt tiêu cặp B,-B hoặc C,-C
    → resolve_pair(c1,c2) = [[A,-C,C,F], [A,B,-B,F]]

- Mô tả: hàm được mô phỏng theo mã giả sau

```
res = { } # set trống

với mỗi literal_1 trong c1:
    với mỗi literal_2 trong c2:
        if(tồn tại 2 cặp literal trái dấu nhau trong c1 và c2):
            # tiến hành hợp giải bằng cách triệt tiêu literal_1 và literal_2
            tmp1 = c1\{literal_1} # bỏ đi literal_1 trong c1 và lưu vào tmp1
```

```
            tmp2 = c2\{literal_2} # bỏ đi literal_2 trong c2 và lưu vào tmp2
            kq = tmp1.merge(tmp2) # merge c1,c2 và lưu vào kq
            res.add(kq) # thêm kq này vào list trả về res
# nếu c1,c2 không có cặp literal nào trái dấu → không có gì để hợp giải…
if(res = rỗng):
    return [c1, c2] #...trả về list gồm chính c1, c2
# ngược lại, return res
return res
```

- Lưu ý rằng quá trình hợp giải 2 câu có thể phát sinh 2 mệnh đề giống nhau nên **res** phải là kiểu set() (**set of Clause objects**) để tránh 2 câu phát sinh trùng nhau
- Việc trả về list [c1, c2] sẽ không ảnh hưởng tới KB ở hàm **pl_resolution** ở dưới vì KB luôn là set (toán tập hợp nên không bị trùng)

c) Hàm pl_resolution: **pl_resolution**(alpha, KB)

- Input: alpha thuộc kiểu **Clause**, KB thuộc kiểu **list of Clause objects**
- Output: danh sách (list) các set với mỗi set là 1 bước của việc chứng minh KB có suy dẫn được alpha. Mỗi set gồm các câu mới được sinh ra từ việc hợp giải các câu trong KB và ¬alpha
- Mô tả: giải thuật được dựa trên mã giả đề cập trong đề và được điều chỉnh để phù hợp với yêu câu đề bài

```
kb = KB ∧ ¬alpha
new = { } # set trống
res = [ ] # danh sách mỗi gồm các câu mới sinh ra chưa có trong kb ở mỗi bước lặp

while(1):
    # sinh toàn bộ các câu mới từ việc hợp giải mọi cặp câu trong kb
    for pair in kb:
        resolvents = resolve_pair(pair[0], pair[1]) # hợp giải mỗi cặp câu trong kb
        new = new ∪ resolvents
    new = new\{TRUE clauses} # loại bỏ các câu có chân trị True
    res.append(new - kb) # lấy hiệu 2 tập để ra được các câu mới chưa có trong kb

    # kiểm tra điều kiện suy dẫn
    if(new contains an empty clause) # nếu new chứa chuỗi rỗng → KB entails alpha
        res.append('YES')
        return res
    if new ⊆ kb # nếu new là tập con kb
        res.append('NO')
        return res

    kb = kb ∪ new
```

II. KỊCH BẢN KIỂM THỬ

- Mỗi file input, những dòng có màu đỏ là phủ định kết luận, được thêm vào báo cáo để tăng ý nghĩa diễn giải, **không có các phủ định kết luận này** trong file input
- Các file input*k*.txt báo cáo là các file input nổi trội trong folder input nên do đó không theo thứ tự k= 1,2,3,…

1. input9.txt

Stt	input9.txt	output9.txt	Ghi chú (hợp giải cặp câu nào)
1	-P	7	
2	5	L OR P (9)	5,7
3	-L OR -M OR P	-A OR -M OR P (10)	3,5
4	-B OR -L OR M	-B OR -L OR P (11)	3,4
5	-A OR P OR L	-B OR L (12)	6,7
6	-A OR -B OR L	-A OR -B OR -M OR P (13)	3,6
7	A	-A OR -B OR M OR P (14)	4,5
8	P	-A OR -B OR M (15)	4,6
		7	
		-B OR -M OR P	7,13
		-A OR -B OR -L OR P	4,13
		-B OR P	11,12
		-B OR M	7,15
		-M OR P	7,10
		-A OR -B OR P	13,14
		-B OR M OR P	7,14
		0	Không có câu mới nào được tạo
		NO	

2. input8.txt

Stt	input8.txt	output8.txt	Ghi chú (hợp giải cặp câu nào)
1	R	7	
2	5	P OR -S (9)	3,7
3	P OR Q	-P OR -Q (10)	4,6
4	-R OR -P	S (11)	8,5
5	R OR S	-Q (12)	8,6
6	R OR -Q	P OR R (13)	3,6
7	-Q OR -S	-P OR S (14)	4,5
8	-R	Q OR -R (15)	3,4
		3	
		P	8,13
		-R OR -S	4,9
		Q OR S	3,14
		0	Không có câu mới nào được tạo
		NO	

3. input4.txt

Stt	input4.txt	output4.txt	Ghi chú (hợp giải cặp câu nào)

1	R	5	
2	5	Q (9)	6,7
3	P	-P OR R OR -S (10)	4,5
4	-P OR -Q OR R	-P OR -Q (11)	8,4
5	-S OR Q	-Q OR R (12)	3,4
6	-T OR Q	-P OR R OR -T (13)	4,6
7	T	8	
8	-R	-P (14)	9,11
		-Q (15)	3,11
		-P OR -T (16)	6,11
		R OR -T (17)	6,12
		-P OR R (18)	7,13
		R OR -S (19)	5,12
		-P OR -S (20)	8,10
		R (21)	9,12
		3	
		-T	3,16
		-S	3,20
		{}	8,21
		YES	

4. input5.txt

Stt	input5.txt	output5.txt	Ghi chú (hợp giải cặp câu nào)
1	A OR R	4	
2	4	R OR S (9)	3,4
3	-P OR R	-P OR Q (10)	3,6
4	P OR S	-P (11)	3,8
5	-S	P (12)	4,5
6	Q OR -R	5	
7	-A	Q OR S	4,10
8	-R	Q	10,12
		R	3,12
		{}	11,12
		S	8,9
		YES	

5. input3.txt

Stt	input3.txt	output3.txt	Ghi chú (hợp giải cặp câu nào)
1	-A OR F	8	
2	4	-C (9)	6,8
3	-A OR B OR C	-D OR F (10)	5,7
4	-B OR D OR F	-B OR D (11)	4,8
5	-A OR -D OR F	-A OR -B OR F (12)	4,5
6	-C OR F	B OR C (13)	3,7
7	A	-A OR -D (14)	5,8

8	-F	-A OR C OR D OR F (15)	3,4
		-A OR B OR F (16)	3,6
		12	
		-B OR F (17)	7,12
		C OR D OR F (18)	7,15
		-A OR C OR D (19)	8,15
		C OR D (20)	11,13
		-A OR F (21)	12,16
		-A OR -B (22)	8,12
		-A OR B (23)	8,16
		-A OR C OR F (24)	10,15
		B (25)	9,13
		B OR F (26)	7,16
		-D (27)	7,14
		-A OR D OR F (28)	9,15
		9	
		C OR F	7,24
		C	20,27
		-A (29)	8,21
		F	17,25
		-B	8,17
		D	9,20
		-A OR C	8,24
		-A OR D	8,28
		D OR F	7,28
		1	
		{}	7,29
		YES	

III. ĐÁNH GIÁ GIẢI THUẬT HỢP GIẢI

Ưu điểm	Khuyết điểm
- Đảm bảo tính đầy đủ, nghĩa là luôn trả về kết quả chứa YES hoặc NO nếu đầu vào input tuân theo chuẩn đề ra - Có thể gián tiếp dùng để kiểm tra tính nhất quán của KB. KB gọi là nhất quán khi không có chuỗi rỗng được tạo trong quá trình hợp giải tất cả các cặp câu trong KB - Sử dụng toán tập hợp (set) nên không lo về vấn đề phát sinh câu trùng	- Input đầu vào phải tuân theo chuẩn CNF. Vì vậy nên có thể dẫn tới mất mát nghĩa gốc của câu ➔ Giải pháp: dùng 1 file text lưu lại tên biến/câu ứng với ý nghĩa chuyển đổi - Không thể dùng chung với các luật nằm ngoài luật logic. VD: sử dụng luật dấu lớn, bé,…trong toán để suy diễn True/False thay vì dùng luật logic - Sau mỗi lần lặp KB, thuật toán sẽ hợp giải luôn các cặp câu đã từng hợp giải ở các lần lặp trước → Tốn chi phí duyệt ➔ Giải pháp: trong lớp **Clause** thêm thuộc tính cờ hiệu flag để ám thị rằng nếu flag = True thì câu

	Clause đã có ở vòng lặp trước đó; ngược lại thì False. Như vậy mỗi lần lặp, trước khi hợp giải 2 câu bất kỳ, chỉ cần 1 trong 2 cờ bằng False thì có thể hợp giải → giảm chi phí duyệt qua các cặp câu đã hợp giải trước đó