**CƠ SỞ TRÍ TUỆ NHÂN TẠO**

**BÁO CÁO ĐỒ ÁN – LAB 2**

**Họ tên: Nguyễn Đại Nghĩa**

**MSSV: 19120735**

**1/ Mã nguồn**

Ở phần này, em xin trình bày cấu trúc, xử lý của các hàm đã cài đặt. Với mỗi hàm, phần mã nguồn được thể hiện ngay sau phần giải thích.

Thư viện sử dụng: os – dùng để lấy danh sách tập tin của thư mục, lấy đường dẫn tuyệt đối, kiểm tra tồn tại của tập tin và thư mục, tạo mới thư mục khi cần thiết.

from os import walk, mkdir

from os.path import dirname, abspath, exists

Các hàm được cài đặt trong chương trình:

* **main**: thực thi chương trình, đưa ra các lời gọi hàm để đọc dữ liệu đầu vào, chạy giải thuật, ghi dữ liệu và kết quả đầu ra
* **list\_all\_files()**: đọc danh sách tập tin của một thư mục cho trước
* **parse\_line()** và **normalize\_list()**: chuyển đổi và chuẩn hóa một mệnh đề logic từ dạng string sang dạng một list của các literals nối rời và sắp xếp theo thứ tự từ điển (tính theo positive literal)
* **PL\_resolution()**: hàm chứa phần triển khai chính của giải thuật hợp giải trong chương trình. Hàm được triển khai theo mã giả tham khảo từ Hình 7.12, trang 255, sách Artificial Intelligence - A Modern Approach, 3e.
* **PL\_resolve()**: là hàm phụ trợ cho PL\_resolution(), có nhiệm vụ kiểm tra số cặp literal đối ngẫu của hai mệnh đề và phát sinh mệnh đề hợp giải có ý nghĩa (nếu có).

a/ Hàm **main**:

* Khai báo các hằng chứa tiền tố của tên tập tin đầu vào và kết quả
* Sử dụng hàm list\_all\_files() để đọc lên danh sách các tập tin đầu vào
* Kiểm ra và tạo mới thư mục kết quả nếu thư mục chưa tồn tại
* Với mỗi tập tin đầu vào sẽ tạo một tập tin đầu ra tương ứng
* Sử dụng biến toàn cục là output (list) để lưu các mệnh đề phát sinh trong quá trình thực thi giải thuật hợp giải
* Thực hiện 3 bước chính: (1) Đọc dữ liệu đầu vào, (2) gọi hàm PL\_resolution() để thực thi giải thuật hợp giải, (3) ghi các mệnh đề phát sinh trong quá trình hợp giải và kết quả hợp giải vào tập tin đầu ra.

if \_\_name\_\_ == "\_\_main\_\_":

    global output

    INPUT = 'input'

    OUTPUT = 'output'

    # get file list for queries

    if exists(INPUT):

        files = list\_all\_files(dirname(abspath(\_\_file\_\_)) + '/' + INPUT)

        if not exists(OUTPUT):

            mkdir(OUTPUT)

        for i in range(len(files)):

            output = []

            input\_file = INPUT + '/' + files[i]

            output\_file = input\_file.replace(INPUT, OUTPUT)

            # reading input file

            alpha = ''

            n = 0

            KB = []

            with open(input\_file) as file:

                alpha = file.readline()

                n = int(file.readline())

                for i in range(n):

                    KB.append(file.readline())

            # main algorithm

            res = PL\_resolution(KB, alpha)

            # writing result

            with open(output\_file, 'w') as file:

                for clauses in output:

                    file.write(str(len(clauses)) + '\n')

                    for clause in clauses:

                        s = ' OR '.join(clause)

                        file.write(s + '\n')

                if res:

                    file.write('YES')

                else:

                    file.write('NO')

b/ Hàm lấy danh sách tập tin đầu vào: **list\_all\_files()**

* Tham số truyền vào: đường dẫn thư mục (string)
* Dử dụng hàm os.walk()
* Trả về list các tập tin đầu vào trong thư mục đầu vào

def list\_all\_files(path):

    lst\_file = []

    for (dirpath, dirname, filename) in walk(path):

        lst\_file.extend(filename)

    return lst\_file

c/ Hàm chuyển dòng dữ liệu string đầu vào thành một list các phần tử **parse\_line()** và hàm chuẩn hóa mệnh đề **normalize\_list()**

* Tham số của parse\_line() là string biểu diễn một mệnh đề đọc được từ tập tin văn bản, tham số của normalize\_list() là một list các literals, có thể chưa loại phần tử trùng và chưa sắp xếp
* parse\_line() loại bỏ ký tự trắng và xuống dòng trong một string, sau đó tách các phần tử theo delimiter là ‘OR’, trả về một mảng các literals
* normalize\_list() chuyển một list thành một set để loại bỏ các phần tử trùng nhau và sắp xếp các literals theo thứ tự bảng chữ cái (dùng index -1 để chạy được cho cả literal dương và âm). Có kiểm tra các mệnh đề không ý nghĩa cho suy luận (ví dụ A OR -A) để loại mệnh đề này bằng cách trả về [True] để thông báo cho hàm nào có sử dụng lời gọi hàm này

def parse\_line(line):

    return line.replace('\n', '').replace(' ', '').split('OR')

def normalize\_list(s):

    # remove duplicate literals and sort alphabetically

    s = sorted(list(set(s)), key = lambda k: k[-1])

    # check if exist duality literal

    for i in range(len(s) - 1):

        if s[i][-1] == s[i + 1][-1]:

            return [True]

    return s

d/ Hàm **PL\_resolution():** giải thuật chính của chương trình

* Tham số truyền vào: mệnh đề alpha (string) và tập KB (list các tuple)
* Các biến sử dụng trong hàm:
  + clauses (kiểu set): lưu tất cả mệnh đề bao gồm giá trị ban đầu là KB giao -alpha, và sau đó bổ sung thêm các mệnh đề mới qua từng vòng lặp
  + new\_clauses (kiểu set): lưu tất cả mệnh đề phát sinh ở mỗi vòng lặp, bao gồm cả mệnh đề cũ đã tồn tại trong clauses
  + output(kiểu list): là biến toàn cục, để ghi lại một tập các mệnh đề hợp giải phát sinh mới trong mỗi vòng lặp, tập này là new\_clauses.difference(clauses)
* Gồm 2 giai đoạn chính:
  + Giai đoạn 1 - Khởi tạo tập clauses: Đọc vào các giá trị của alpha, n, KB, chuẩn hóa (sẽ nói cụ thể ở phần tiếp theo) và đưa vào clauses KB và các literals của mệnh đề phủ định alpha

def PL\_resolution(KB, alpha):

    global output

    # 1 - add KB and -alpha to clauses

    clauses = set()

    # add all clauses of KB to clauses

    for line in KB:

        c = normalize\_list(parse\_line(line))

        # transform clause from list to tuple -> add into a set

        clauses.add(tuple(c))

    # add negative of alpha to clauses

    alpha = normalize\_list(parse\_line(alpha))

    if alpha == [True]:

        output = []

        return True

    for literal in alpha:

        if len(literal) == 1:

            clauses.add(tuple(['-' + literal]))

        else:

            clauses.add(tuple([literal[-1]]))

* + Giai đoạn 2 - Các vòng lặp: Thực hiện các vòng lặp của quá trình hợp giải
    - Khởi tạo biến new\_clauses (set) chứa tất cả kết quả của mỗi lần lặp trên toàn bộ các mệnh đề của clauses. Khởi tạo biến empty = False, biến này đánh dấu đã xuất hiện mệnh đề rỗng hay chưa.
    - Sử dụng nested loop để duyệt qua từng cặp mệnh đề trong clauses, sử dụng hàm PL\_resolve() để thử hợp giải hai mệnh đề. Tùy vào kết quả của việc hợp giải này mà có các hướng xử lý: (1) Nếu xuất hiện mệnh đề rỗng thì thêm {} vào tập kết quả và đánh dấu biến empty = True; (2) nếu hợp giải tạo ra mệnh đề không có ý nghĩa cho quá trình suy luận (mệnh đề True hoặc hợp giải hai mệnh đề không có literal đối ngẫu) thì bỏ qua; (3) nếu hợp giải từ 2 mệnh đề có đúng 1 cặp literal đối ngẫu thì sẽ tạo ra mệnh đề có ý nghĩa, khi đó đưa mệnh đề này vào new\_clauses (tự động kiểm tra chưa tồn tại mới đưa vào, vì new\_clauses kiểu set)
    - Cuối mỗi vòng lặp thực hiện đưa các mệnh đề phát sinh mới vào biến toàn cục output để phục vụ cho việc ghi kết quả. Tùy vào các kết quả trước đó mà quyết định kết thúc lặp hay chưa: (1) nếu xuất hiện mệnh đề rỗng thì quá trình hợp giải kết thúc, trả về True; (2) nếu new\_clauses là tập con của clauses tức không phát sinh mệnh đề mới trong vòng lặp thì quá trình hợp giải cũng kết thúc, trả về False; (3) tiếp tục lặp, cập nhật tập clauses trước khi lặp vòng lặp mới.

    # 2 - loop

    while True:

        empty = False

        new\_clauses = set()

        for c1 in clauses:

            for c2 in clauses:

                r = PL\_resolve(c1, c2)

                if len(r) == 0:

                    empty = True

                    new\_clauses.add(tuple(['{}']))

                elif r == [True]:

                    continue

                else:

                    new\_clauses.add(tuple(r))

        # put new clauses generated in this loop to output

        # for file writing

        output.append(new\_clauses.difference(clauses))

        # check loop stop

        if empty:

            return True

        if new\_clauses.issubset(clauses):

            return False

        # continue to next loop

        clauses = clauses | new\_clauses

e/ Hàm **PL\_resolve()**

* Tham số truyền vào: hai mệnh đề cần hợp giải (list), mỗi mệnh đề là một list của các literals
* Duyệt qua từng cặp literal của hai mệnh đề và lưu lại literal dương của các cặp literal đối ngẫu xuất hiện trong hai mệnh đề vào biến dual (list)
* Kiểm tra số cặp literal đối ngẫu của hai mệnh đề (tức len(dual)):
  + Nếu hai mệnh đề có chứa đúng một cặp literal đối ngẫu thì tiến hành nối liền hai mệnh đề này rồi xóa đi cặp literal đối ngẫu để phát sinh mệnh đề mới có ý nghĩa. Kết quả trả về là mệnh đề mới (list) đã sắp xếp các literals theo thứ tự bảng chữ cái.
  + Ngược lại nếu có từ hai cặp literal đối ngẫu trở lên hoặc không có cặp literal đối ngẫu nào giữa hai mệnh đề thì việc hợp giải không tạo ra thêm mệnh đề có nghĩa cho quá trình suy luận. Kết quả trả về là [True], là list chứa 1 phần tử True, để đánh dấu hợp giải hai mệnh đề này không có ý nghĩa.

def PL\_resolve(clause1, clause2):

    # checking the duality literals

    dual = list()

    for lit1 in clause1:

        for lit2 in clause2:

            if lit1[-1] == lit2[-1] and len(lit1) != len(lit2):

                dual.append(lit1[-1])

    if len(dual) == 1:

        li = list(set(clause1) | set(clause2))

        li.remove(dual[0])

        li.remove('-' + dual[0])

        return sorted(li, key = lambda k: k[-1])

    # len(dual) == 0 -> no useful when merging (AND) 2 clauses

    # len(dual) >= 2 -> no useful, too

    return [True]

**2/ Kết quả thử nghiệm**

Phần bài làm này bao gồm thử nghiệm trên 5 testcases tự tạo và được lưu trong các tập tin có dạng input<x>.txt trong thư mục input, với x nhận giá trị từ 1 tới 5. Sau khi chạy chương trình được mô tả ở phần 1, các kết quả được lưu trong các tập tin output<x>.txt ở thư mục output (thư mục này được tạo tự động nếu chưa có sẵn).

Dưới đây là nội dung chi tiết của các tập tin input<x>.txt và outout<x>.txt cùng với phần giải thích kết quả

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **Input** | **Output** | **Giải thích** |
| 1 | P  7  Q  -Q OR R  -R OR S  -S OR T  -T OR U  -U OR V  P OR -V | 7  -Q OR S  -V  -R OR T  -S OR U  P OR -U  R  -T OR V  11  S  -R OR V  -R OR U  -U  -Q OR U  -S OR V  T  -Q OR T  -T  P OR -T  P OR -S  10  P  {}  -R  P OR -R  -Q OR V  -Q  P OR -Q  U  V  -S  YES | (-Q OR R) hợp giải với (-R OR S)  (P) hợp giải với (P OR -V)  (-R OR S)  (-T OR U) hợp giải với (-S OR T)  (-U OR V) hợp giải với (P OR -V)  (Q) hợp giải với (-Q OR R)  (-U OR V) hợp giải với (-T OR U)  Q hợp giải với (-Q OR S)  (-R OR T) hợp giải với (-T OR V)  (-R OR T) hợp giải với (-T OR U)  (-U OR V) hợp giải với (-V)  (-T OR U) hợp giải với (-Q OR T)  (-S OR U) hợp giải với (-U OR V)  (R) hợp giải với (-R OR T)  (-Q OR R) hợp giải với (-R OR T)  (-V) hợp giải với (-T OR V)  (-T OR U) hợp giải với (P OR -U)  (P OR -U) hợp giải với (-S OR U)  (P OR -S) hợp giải với (S)  (T) hợp giải với (-T)  (-U) hợp giải với (-R OR U)  (-R OR S) hợp giải với (P OR -S)  (-R OR V) hợp giải với (-Q OR R)  (-Q OR U) hợp giải với (-U)  (-Q OR U) hợp giải với (P OR -U)  (T) hợp giải với (-T OR U)  T hợp giải với (-T OR V)  (-U) hợp giải với (-S OR U)  Xuất hiện {} |
| 2 | S OR T  6  -P OR Q OR U  R OR -S  P OR T  -Q OR -R  -T OR U  R OR T | 8  P  Q OR T OR U  R OR U  R  -Q OR T  P OR U  -Q OR -S  -P OR -R OR U  10  -P OR T OR U  T OR U  Q OR U  -R OR T OR U  -P OR U  -R OR U  -Q  -Q OR U  -P OR -S OR U  -S OR T OR U  2  -S OR U  U  0  NO | (P OR T) hợp giải với (-T)  (P OR T) hợp giải với (-P OR Q OR U)  (-T OR U) hợp giải với (R OR T)  (R OR T) hợp giải với (-T)  (R OR T) hợp giải với (-Q OR -R)  (P OR T) hợp giải với (-T OR U)  (R OR -S) hợp giải với (-Q OR -R)  (-Q OR -R) hợp giải với (-P OR Q OR U)  (-P OR Q OR U) hợp giải với (-Q OR T)  (-Q OR T) hợp giải với (Q OR T OR U)  (-P OR Q OR U) hợp giải với (P OR U)  (P OR T) hợp giải với (-P OR -R OR U)  (-P OR -R OR U) hợp giải với (R OR)  (P OR U) hợp giải với (-P OR -R OR U)  (-T OR) hợp giải với (-Q OR T)  (-Q OR T) hợp giải với (-T OR U)  (R OR -S) hợp giải với (-P OR -R OR U)  (R OR -S) hợp giải với (-P OR -R OR U)  (R OR -S) hợp giải với (-R OR U)  (T OR U) hợp giải với (-T OR U)  Không {} và không phát sinh mệnh đề mới |
| 3 | R OR -T  6  Q OR -T  -S  P OR S  -P OR -Q OR R  A OR R OR S  B OR -S | 9  A OR B OR R  P  A OR S  A OR R  Q  B OR P  -Q OR R OR S  -P OR R OR -T  -P OR -Q  14  R OR S  -P OR R  -P OR -T  R OR S OR -T  A OR B  -Q OR S  B OR -Q OR R  B OR -Q  R OR -T  -P  A  -Q  B OR R OR -T  -Q OR R  8  S OR -T  -T  R  B  S  B OR -T  {}  B OR R  YES | (A OR R OR S) hợp giải với (B OR -S)  (-S) hợp giải với (P OR S)  (-R) hợp giải với (A OR R OR S)  (A OR R OR S) hợp giải với (-S OR)  (T) hợp giải với (Q OR -T)  (B OR -S) hợp giải với (P OR S)  (P OR S) hợp giải với (-P OR -Q OR R)  (-P OR -Q OR R) hợp giải với (Q OR -T)  (-R) hợp giải với (-P OR -Q OR R)  (Q) hợp giải với (-Q OR R OR S)  (T) hợp giải với (-P OR R OR -T)  (-R) hợp giải với (-P OR R OR -T)  (-P OR R OR -T) hợp giải với (P OR S)  (A OR B OR R) hợp giải với (-R)  (-R) hợp giải với (-Q OR R OR S)  (-P OR -Q OR R) hợp giải với (B OR P)  (B OR P) hợp giải với (-P OR -Q)  (P) hợp giải với (-P OR R OR -T)  (Q) hợp giải với (-P OR -Q)  (-S) hợp giải với (A OR S)  (P) hợp giải với (-P OR -Q)  (-P OR R OR -T) hợp giải với (B OR P)  (P) hợp giải với (-P OR -Q OR R)  (-P OR -T) hợp giải với (P OR S)  (R OR -T) hợp giải với (-R)  (-S) hợp giải với (R OR S)  (B OR P) hợp giải với (-P)  (R OR S) hợp giải với (-R)  (Q OR -T) hợp giải với (B OR -Q)  (Q) hợp giải với (-Q)  (B OR P) hợp giải với (-P OR R)  Xuất hiện {} |
| 4 | -S OR U  6  P OR Q  Q OR S OR U  -Q OR R  -R OR T  -P OR U  R OR -S | 9  -P  R  R OR S OR U  Q OR S  -Q OR T  Q OR R OR U  P OR R  -S OR T  Q OR U  12  R OR U  S OR T OR U  Q OR T OR U  S OR T  Q OR R  P OR T  T  R OR S  T OR U  R OR T OR U  Q OR T  Q  1  R OR T  0  NO | (-U) hợp giải với (-P OR U)  (S) hợp giải với (R OR -S)  (Q OR S OR U) hợp giải với (-Q OR R)  (Q OR S OR U) hợp giải với (-U OR)  (-R OR T) hợp giải với (-Q OR R)  (Q OR S OR U) hợp giải với (R OR -S)  (P OR Q) hợp giải với (-Q OR R)  (R OR -S) hợp giải với (-R OR T)  (P OR Q) hợp giải với (-P OR U)  (R OR S OR U) hợp giải với (R OR -S)  (R OR S OR U) hợp giải với (-R OR T)  (-R OR T) hợp giải với (Q OR R OR U)  (Q OR S) hợp giải với (-Q OR T)  (-U OR) hợp giải với (Q OR R OR U)  (P OR Q) hợp giải với (-Q OR T)  (R) hợp giải với (-R OR T)  (R OR S OR U) hợp giải với (-U OR)  (-Q OR T) hợp giải với (Q OR U)  (R OR S OR U) hợp giải với (-S OR T)  (Q OR S) hợp giải với (-S OR T)  (P OR Q) hợp giải với (-P)  (S OR T) hợp giải với (R OR -S)  Không {} và không phát sinh mệnh đề mới |
| 5 | -P OR Q  7  S OR T OR U  Q OR -R  -P OR T  -B OR U  R OR -T  -S  B OR -Q OR U | 8  T  -Q OR U  B OR -R OR U  -P OR R  -R  Q OR -T  R OR S OR U  T OR U  14  R OR U  B OR -T OR U  Q  B OR -P OR U  -R OR U  S OR U  B OR S OR U  Q OR S OR U  R  Q OR U  -T  -P OR Q  -P  -T OR U  4  -P OR U  B OR U  {}  U  YES | (P) hợp giải với (-P OR T)  (B OR -Q OR U) hợp giải với (-B OR U)  (Q OR -R) hợp giải với (B OR -Q OR U)  (-P OR T) hợp giải với (R OR -T)  (Q OR -R) hợp giải với (-Q)  (Q OR -R) hợp giải với (R OR -T)  (R OR -T) hợp giải với (S OR T OR U)  (-S) hợp giải với (S OR T OR U)  (T OR U) hợp giải với (R OR -T)  (B OR -R OR U) hợp giải với (R OR -T)  (Q OR -T) hợp giải với (T OR)  (B OR -R OR U) hợp giải với (-P OR R)  (Q OR -R) hợp giải với (-Q OR U)  (-R) hợp giải với (R OR S OR U)  (R OR S OR U) hợp giải với (B OR -R OR U)  (Q OR -R) hợp giải với (R OR S OR U)  (T) hợp giải với (R OR -T)  (T OR U) hợp giải với (Q OR -T)  (Q OR -T) hợp giải với (-Q)  (Q OR -T) hợp giải với (-P OR T)  (-P OR R) hợp giải với (-R)  (Q OR -T) hợp giải với (-Q OR U)  (-P OR Q) hợp giải với (-Q OR U)  (B OR -Q OR U) hợp giải với (Q OR U)  (-R) hợp giải với (R)  (-R) hợp giải với (R OR U)  Xuất hiện {} |

**3/ Nhận xét**

* Giải thuật chạy hiệu quả trên dữ liệu đã chuẩn hóa theo quy ước
* Số lượng mệnh đề phát sinh tương đối lớn, tăng nhanh khi số lượng các literal khác nhau trong các mệnh đề ban đầu của KB nhiều

**4/ Tài liệu tham khảo**

Stuart J. Russell, Peter Norvig. Artificial Intelligence - A Modern Approach, 3rd, 2016

GS.TS Lê Hoài Bắc. Tài liệu bài giảng môn Cơ sở Trí tuệ nhân tạo, Trường ĐH KHTN, ĐHQG-HCM, 2021