VNUHCM – University Of Science
Information Technology (High – Quality Program)

Project 02 – Logic

ROBINSON'S RESOLUTION ALGORITHM

Introduction to Artificial Intelligent

Lecturers:

Mr. Chau Thanh Duc

Mr. Ngo Dinh Hy

Ms. Phan Thi Phuong Uyen

Semester 3 – August 08, 2021

MỤC LỤC

I. THÔNG TIN CÁ NHÂN

- **Họ và tên :** Ngô Huy Anh
- **MSSV :** 19127095

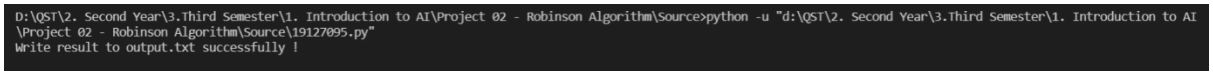
II. CÁCH BIÊN DỊCH VÀ CHẠY MÃ NGUỒN

Đặc điểm

1. Ngôn ngữ sử dụng: Python – Phiên bản: 3.9.6
2. Chương trình được viết trên **Visual Studio Code editor**

Cách biên dịch và chạy mã nguồn

1. Tạo folder ở vị trí muốn lưu trữ code.
2. Vào editor Visual Studio Code, nhấn tổ hợp phím Ctrl + Shift + N để tạo window mới,bấm "Open Folder" và dẫn đến vị trí folder bạn vừa tạo, tạo file với tên file kết thúc bằng **.py** (ngôn ngữ Python).
3. Đưa file input.txt **vào đúng địa chỉ folder đã tạo – cùng vị trí với file .py**.
4. Quay về màn hình Visual Studio Code, nhấn Ctrl + Alt + N để run code.
5. Khi chương trình báo **'Write result to output.txt successfully ! '** tức là đã hoàn thành.
6. Vào file output.txt để kiểm tra kết quả



HÌNH ẢNH VÍ DỤ KHI RUN CODE THÀNH CÔNG

III. GIẢI THÍCH CỤ THỂ THUẬT TOÁN

Thư viện bổ trợ và cấu trúc dữ liệu sử dụng

Thư viện bổ trợ:

copy – Bởi vì các phép gán hay tính toán trong Python đều thông qua địa chỉ của biến. Nên cần phải sử dụng lệnh **copy.deepcopy()** để dữ liệu gốc không bị thay đổi khi thao tác ở trên biến tạm.

Cấu trúc dữ liệu:

List, Set

Robinson's resolution algorithm

1. Pseudo-code

```
function PL_Resolution(KB, alpha) return True or False
    inputs:
        KB - Một cơ sở / tập hợp mệnh đề cho trước, đã được biến đổi dưới dạng hội chuẩn CNF
        alpha - Câu cần kiểm tra, đã được phủ định và biến đổi dưới dạng hội chuẩn CNF

    # CODE
    list_resolution: List ← Lưu trữ lại các biểu thức suy diễn, dành cho phần output
    clauses: List ← Tập hợp các mệnh đề biểu diễn theo CNF của KB ∧ ~alpha
    new: Set ← Tập hợp các mệnh đề mới được tạo ra từ clauses

    loop do:
        for each pair of clauses Ci,Cj in clauses do:
            resolvents ← PL_Resolve(Ci,Cj)

            if length of resolvents equal to 0 then the loop will continue

            if resolvents contains the empty clause then return True

            # new ← new ∪ resolvents
            for each new clause in resolvents:
                if new clause not in new and is tautology:# p|~p|q -> p|~p is tautology->True
                    new: Update
                    list_resolution: Update

        if new <= clauses then return False
        clauses ← clauses ∪ new

    return
```

Thuật toán được tham khảo từ sách: Artificial Intelligence A Modern Approach – Third Edition

Tác giả: Stuart Russel, Peter Norvig

Cấu trúc dữ liệu Set giúp ta chứa các mệnh đề mới mà không cần bận tâm đến vấn đề trùng lắp.

Thuật toán Robinson's resolution là một phương pháp chứng minh phản chứng. Muốn chứng minh KB ⇒ α là đúng, ta sẽ đi chứng mình điều ngược lại – tức KB ∧ ~ α là sai.

Thuật toán ở trong Pseudo-code sẽ được thực thi qua các bước sau:

Bước 1: Viết lại giả thiết và kết luận dưới dạng chuẩn CNF

Bước 2: Phủ định lại kết luận

Bước 3: Thêm giả thiết và phủ định của kết luận vào KB

Bước 4: Xây dựng các mệnh đề mới bằng cách kết hợp các cặp câu trong KB, nếu có cặp đối ngẫu thì cặp này sẽ được loại bỏ.

Bước 5: Nếu trong KB có cặp mệnh đề đối ngẫu thì thuật toán sẽ trả về True

Bước 6: Kiểm tra xem mệnh đề mới có bị trùng lắp với các mệnh đề trước đó và có khẳng định đúng (Tautology) hay không. Nếu có thì thêm vào **cấu trúc set** New

Bước 7: Nếu **cấu trúc set** New là tập con của KB thì thuật toán sẽ trả về False

Bước 8: Cập nhật lại KB

Bước 9: Lặp lại từ bước 4 cho đến khi tìm được kết quả.

2. Hàm NegateClause

```
def NegateClause(alpha):
    negated_clause = ""
    temp = alpha.split('&')

    # Negate the Clause
    for i in range(0,len(temp)):
        if (len(temp[i]) == 1):
            temp[i] = '~' + temp[i]
        else:
            temp[i] = temp[i][1]

        # Make Negated Clause
        if (i != len(temp) - 1):
            negated_clause += temp[i] + '|'
        else:
            negated_clause += temp[i]

    return negated_clause
```

Hàm này dùng để phủ định lại kết luận cần kiểm tra, kết quả trả về sẽ là chuỗi kết luận đã được phủ định.

Đầu tiên, dùng split để tách từng phần tử ở trong kết luận ra. Nếu độ dài của phần tử là 1 thì function sẽ thêm ~ vào, nếu độ dài của phần tử là 2 thì function sẽ lược bớt kí tự ở vị trí đầu tiên – tức là dấu ~.

Cuối cùng, hàm sẽ ghép chuỗi mới lại và return về.

3. Hàm IsHasTautology

```
def IsHasTautology(clause):
    list_clause = clause.split('|')

    for i in list_clause:
        if len(i) == 1:
            for j in list_clause:
                if len(j) == 2:
                    if j[1] == i[0]:
                        return True
    return False
```

Hàm này dùng để kiểm tra xem mệnh đề đang xét có chứa khẳng định luôn đúng hay không.

Ví dụ: p|~p|q thì p|~p là khẳng định luôn đúng

4. Hàm PL Resolve

```
def PL_Resolve(C1, C2):
    new_clause = []
    temp_clause = []

    # Convert string to list
    list_C1 = C1.split('|')
    list_C2 = C2.split('|')

    for i in list_C1:
        for j in list_C2:
            if (i == NegateClause(j)):
                # Find p & ~p then delete
                temp_C1 = copy.deepcopy(list_C1)
                temp_C2 = copy.deepcopy(list_C2)
                temp_C1.remove(i)
                temp_C2.remove(j)

                # Check if i is negative
                if (i[0] == '~'):
                    # Extend -> Add multiple element
                    temp_clause.extend(temp_C2)
                    temp_clause.extend(temp_C1)
                else:
                    temp_clause.extend(temp_C1)
                    temp_clause.extend(temp_C2)

                # Add new clause
                clause = '|'.join(temp_clause)
                new_clause.append(clause)  # Append -> Add one element

    return new_clause
```

Hàm này dùng để tạo ra một mệnh đề mới, với hai mệnh đề C_i C_j được xét từ KB

Đầu tiên, dùng split để tách từng phần tử ở trong hai mệnh đề ra. Sau đó kiểm tra xem có mệnh đề đối ngẫu nào xuất hiện trong chúng hay không. Nếu không, function sẽ trả về list rỗng. Nếu có, function sẽ tiếp tục kiểm tra vị trí của mệnh đề phủ định để tạo ra mệnh đề mới đúng với logic.

VD: ~p|r với p|s thì mệnh đề mới được tạo ra sẽ là s|r

IV. THAM KHẢO

1) Giáo trình Cơ sở trí tuệ nhân tạo – Tác giả: Thầy Lê Hoài Bắc, Thầy Tô Hoài Việt – Nhà xuất bản Khoa học và Kỹ thuật – Chỉnh sửa và bổ sung năm 2014.
2) Sách Artificial Intelligence A Modern Approach – Third Edition – Tác giả: Stuart Russell, Peter Norvig
3) Slide bài giảng môn Cơ sở trí tuệ nhân tạo – Lớp: 19CLC6