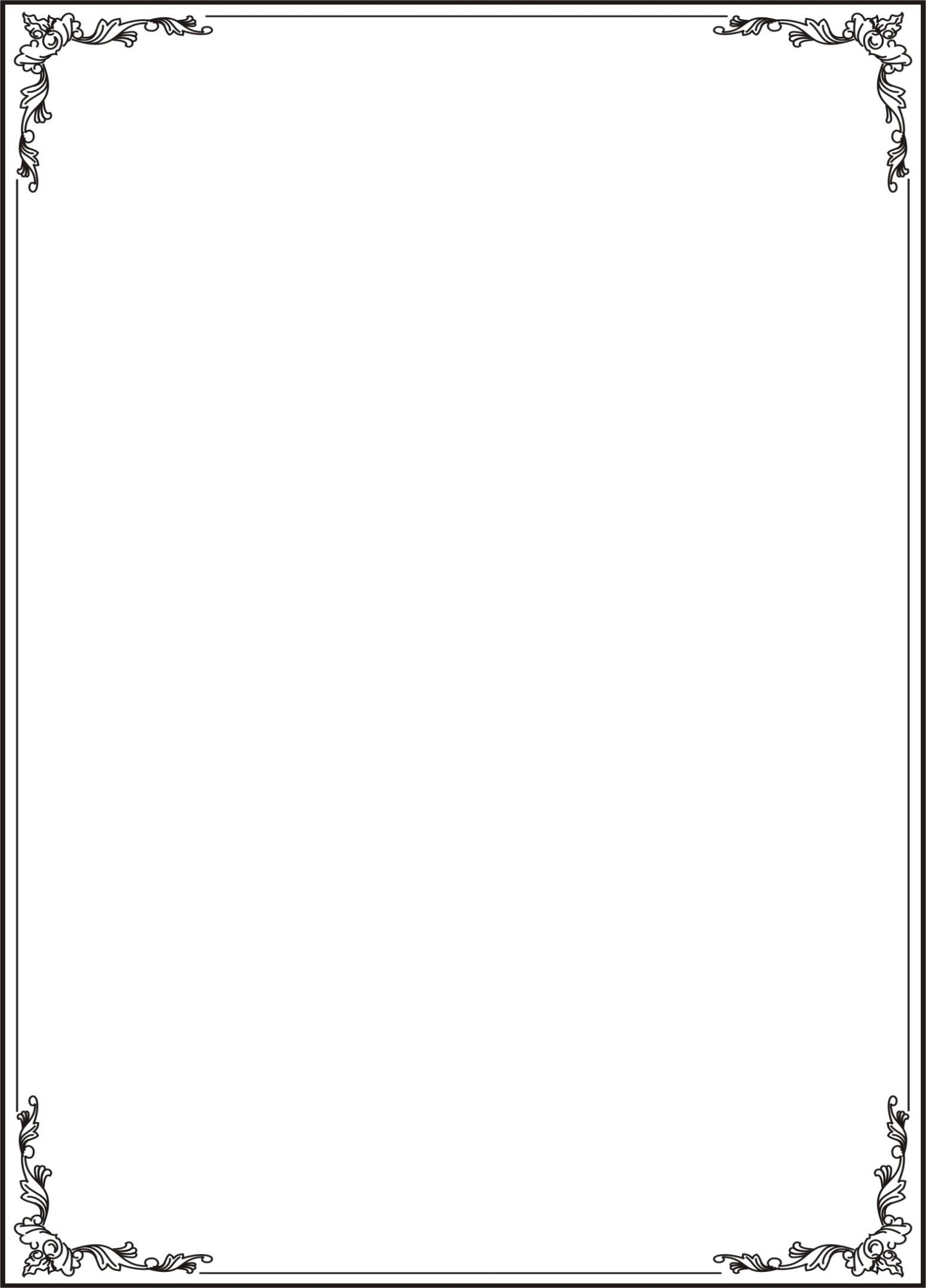
**TRƯỜNG ĐẠI HỌC LAO ĐỘNG – XÃ HỘI**

**KHÓA GIÁO DỤC ĐẠI CƯƠNG**

**--------\*\*\*--------**



**BÁO CÁO BÀI TẬP LỚN**

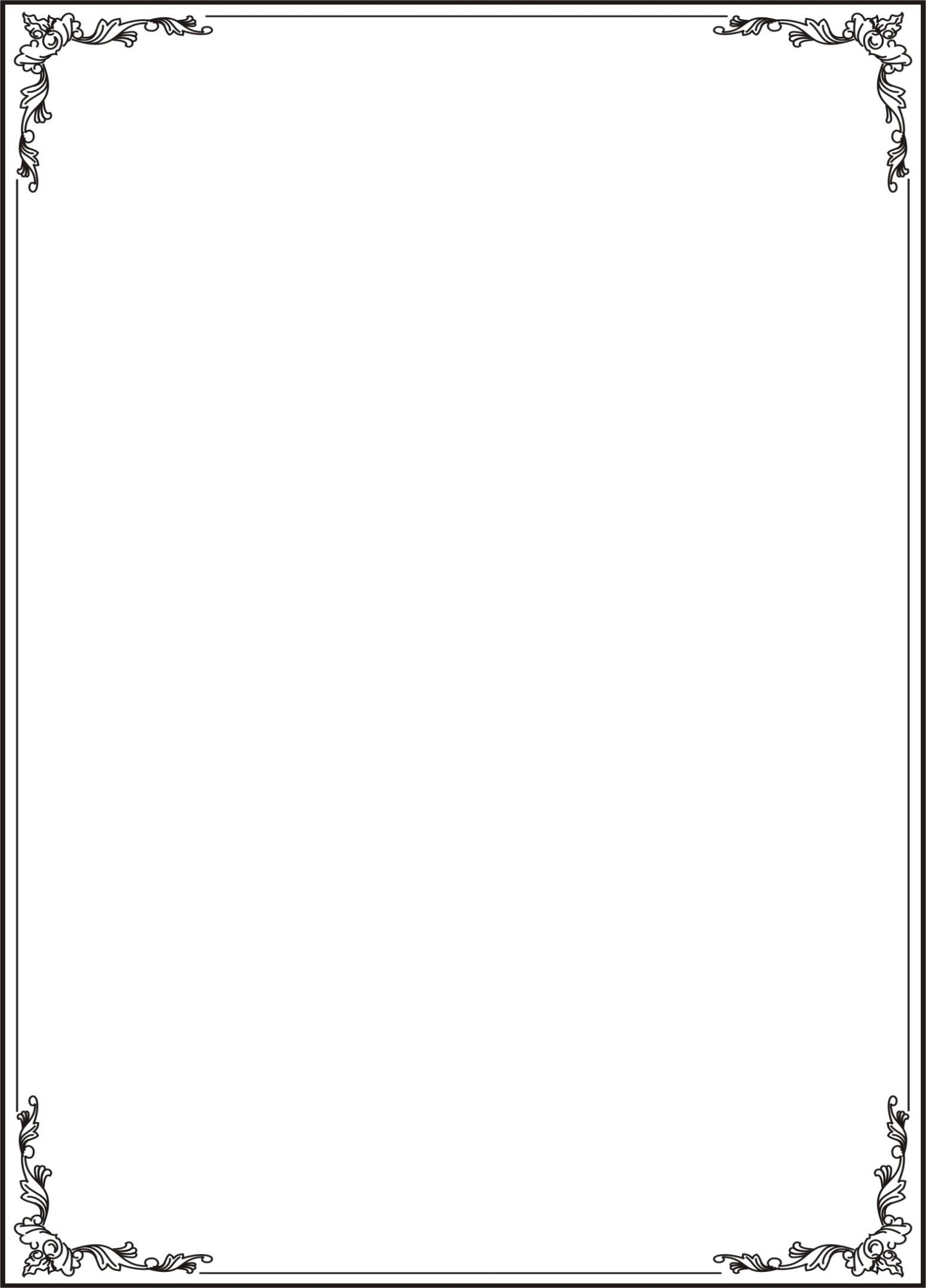
**TRÍ TUỆ NHÂN TẠO**

Hà Nội, năm 2024

**TRƯỜNG ĐẠI HỌC LAO ĐỘNG – XÃ HỘI**

**KHÓA GIÁO DỤC ĐẠI CƯƠNG**

**--------\*\*\*--------**



**BÁO CÁO BÀI TẬP LỚN**

**TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: SỐ 2**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | Nguyễn Anh Thơ |
| Thành viên nhóm: | 1117091629-Đinh Thị Hải Yến-134\_Phần 1\_ Câu 1  1117091626-Lê Thị Huyền Trang-122\_Phần 2\_Câu 2 |
|  | 111709411-Nguyễn Long Khánh- 57\_Phần 3\_Câu 3 |
| Lớp tín chỉ: | D17CN04\_LT |

Hà Nội, năm 2024

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 2](#_Toc166965051)

[**PHẦN I: BÀI TOÁN TÌM KIẾM LỜI GIẢI TRÊN KHÔNG GIAN TRẠNG THÁI** 3](#_Toc166965052)

[Khái niệm cơ bản: 3](#_Toc166965053)

[Câu 1.Bài toán tìm kiếm lời giải trên không gian trạng thái( đề 2) 10](#_Toc166965054)

[**PHẦN II: BÀI TOÁN LOGIC** 14](#_Toc166965055)

[Câu 2. Logic(đề 2) 14](#_Toc166965056)

[Bài tập 7 (bài tập nqueens-size) 14](#_Toc166965057)

[Bài tập 10 17](#_Toc166965058)

[Bài tập 11 23](#_Toc166965059)

[Bài tập 12 28](#_Toc166965060)

[Bài tập 27 33](#_Toc166965061)

[Bài tập 29 36](#_Toc166965062)

[Bài tập 30 38](#_Toc166965063)

[**PHẦN III: HỌC MÁY** 42](#_Toc166965064)

[Định nghĩa 43](#_Toc166965065)

[Biểu diễn 43](#_Toc166965066)

[Tính phổ quát 44](#_Toc166965067)

[Tương tác với con người 45](#_Toc166965068)

[Tương quan với Khai phá dữ liệu 46](#_Toc166965069)

[Các loại giải thuật 46](#_Toc166965070)

[Các chủ đề về máy học 48](#_Toc166965071)

[Câu 3: Học máy (đề 2) 49](#_Toc166965072)

[**KẾT LUẬN** 52](#_Toc166965073)

[**TÀI LIỆU THAM KHẢO** 53](#_Toc166965074)

# LỜI MỞ ĐẦU

Khi giải quyết các bài toán logic trong trí tuệ nhân tạo (AI), việc sử dụng logic bậc nhất (first-order logic) là một phương pháp phổ biến và mạnh mẽ. Logic bậc nhất cho phép chúng ta mô hình hóa các vấn đề theo cách chính xác và rõ ràng bằng cách sử dụng các quan hệ, hàm, và quantifier.

Trong các bài tập logic AI, việc đặc tả một bài toán thành các mệnh đề trong logic bậc nhất là một phần quan trọng. Điều này giúp chúng ta phân tích và hiểu rõ hơn về vấn đề cần giải quyết, cũng như tạo ra các phương pháp giải quyết hiệu quả sử dụng logic.

Trong quá trình làm bài tập, việc xác định một bộ từ vựng cụ thể là quan trọng. Bộ từ vựng này gồm các hàm, quan hệ và quantifier phù hợp với bài toán cụ thể. Khi đã xác định được bộ từ vựng, ta có thể dễ dàng biểu diễn các mệnh đề của bài toán thành các biểu thức logic bậc nhất.

Tiếp theo, chúng ta có thể sử dụng các công cụ và kỹ thuật logic để phân tích và giải quyết các bài toán. Điều này bao gồm việc sử dụng luật suy luận, tìm kiếm, và các phương pháp khác để rút ra những kết luận hoặc giải pháp cho các vấn đề logic.

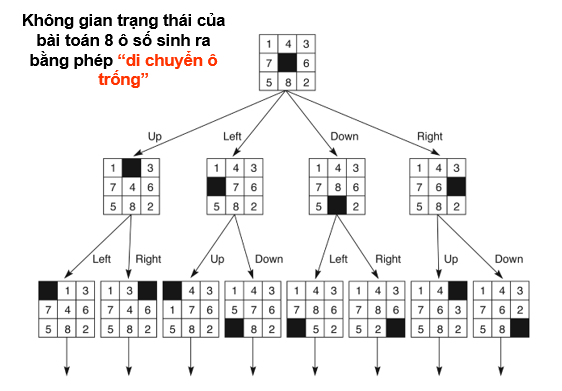
Trong quá trình giải quyết, cần chú ý đến tính chính xác và tính khả thi của các biểu diễn logic, đồng thời luôn kiểm tra và đảm bảo rằng các biểu thức logic đang được sử dụng phản ánh đúng vấn đề cần giải quyết.

Trong bài tập này, chúng ta sẽ thực hành đặc tả các vấn đề sử dụng logic bậc nhất và áp dụng các kỹ thuật logic để giải quyết chúng. Điều này giúp chúng ta phát triển khả năng suy luận logic và áp dụng chúng vào các vấn đề thực tế trong trí tuệ nhân tạo.

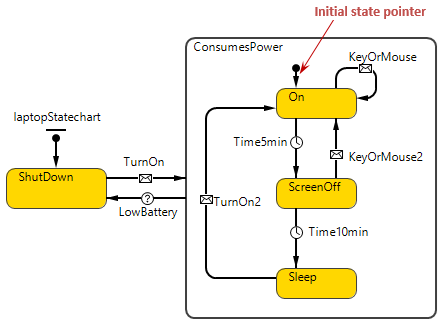
# PHẦN I: BÀI TOÁN TÌM KIẾM LỜI GIẢI TRÊN KHÔNG GIAN TRẠNG THÁI

## Khái niệm cơ bản:

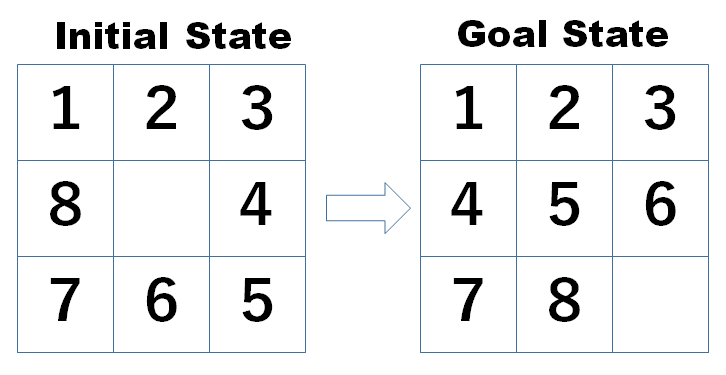
Không gian trạng thái (State Space) là một khái niệm quan trọng trong lĩnh vực Trí Tuệ Nhân Tạo. Nó đề cập đến tập hợp các trạng thái có thể của một hệ thống hoặc một bài toán cụ thể. Mỗi trạng thái trong không gian trạng thái có thể được mô tả bằng một tập hợp các thuộc tính, thông tin hoặc điều kiện cụ thể. Việc hiểu và mô hình hóa đúng không gian trạng thái là quan trọng để giải quyết các bài toán logic và tìm kiếm lời giải trong trí tuệ nhân tạo.



Trạng thái ban đầu (Initial State) là trạng thái xuất phát từ đó quá trình tìm kiếm bắt đầu. Trong một bài toán tìm kiếm trên không gian trạng thái, trạng thái ban đầu là trạng thái mà quá trình tìm kiếm bắt đầu. Từ trạng thái ban đầu, các hành động sẽ được thực hiện để di chuyển qua các trạng thái khác trong không gian trạng thái, với hy vọng tìm ra trạng thái kết thúc hoặc mục tiêu. Trạng thái ban đầu thường được xác định trước khi bắt đầu quá trình tìm kiếm và là một phần quan trọng của việc xác định bài toán.



Trạng thái kết thúc (Goal State) là mục tiêu mà chúng ta muốn đạt được trong quá trình tìm kiếm. Đối với mỗi bài toán, trạng thái kết thúc thường được xác định trước để làm rõ điều kiện hoặc mục tiêu cuối cùng mà hệ thống hoặc tác nhân cần đạt được. Mục tiêu của quá trình tìm kiếm là di chuyển từ trạng thái ban đầu (Initial State) thông qua các hành động và trạng thái trung gian để đạt đến trạng thái kết thúc. Đây là một phần quan trọng của quá trình tìm kiếm và đóng vai trò quyết định trong việc xác định chiến lược tìm kiếm và đánh giá hiệu suất của thuật toán tìm kiếm.

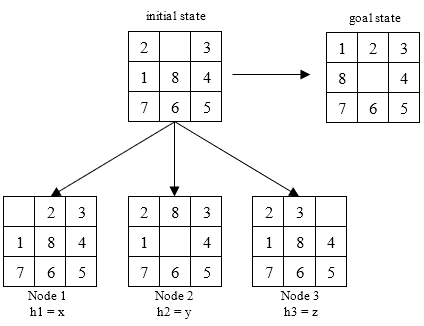


Hành động (Action) là các bước mà hệ thống hoặc tác nhân có thể thực hiện để chuyển từ một trạng thái sang một trạng thái khác trong không gian trạng thái. Trong một bài toán tìm kiếm, hành động định nghĩa cách mà hệ thống có thể thay đổi trạng thái hiện tại của mình. Mỗi hành động thường được kết hợp với một trạng thái ban đầu và tạo ra một trạng thái mới sau khi được thực hiện. Quá trình lựa chọn hành động phù hợp để tiến gần đến trạng thái kết thúc là một phần quan trọng của việc giải quyết bài toán tìm kiếm. Đôi khi, hành động cũng đi kèm với một chi phí hoặc một số hạn chế nào đó, phụ thuộc vào bài toán cụ thể.

Hàm trạng thái kế tiếp nhận vào một trạng thái hiện tại và trả về tập hợp các trạng thái mà có thể đạt được từ trạng thái hiện tại bằng cách thực hiện các hành động khả thi. Mỗi trạng thái trong tập hợp này được gọi là trạng thái kế tiếp của trạng thái hiện tại.

Hàm trạng thái kế tiếp cung cấp thông tin quan trọng để điều chỉnh quá trình tìm kiếm và giúp thuật toán quyết định hướng di chuyển tiếp theo trong không gian trạng thái. Điều này giúp tối ưu hóa quá trình tìm kiếm và tăng hiệu suất của thuật toán.

Hàm đánh giá (Heuristic Function) là một khái niệm quan trọng trong tìm kiếm trên không gian trạng thái. Hàm này được sử dụng để ước lượng chi phí từ một trạng thái hiện tại đến trạng thái mục tiêu, giúp định hướng quá trình tìm kiếm.



Hàm đánh giá thường được thiết kế để cung cấp một ước lượng khái quát về khoảng cách hoặc chi phí từ trạng thái hiện tại đến trạng thái mục tiêu. Điều này giúp thuật toán quyết định hướng di chuyển tiếp theo trong không gian trạng thái một cách thông minh và hiệu quả.

Một hàm đánh giá tốt là hàm mà ước lượng của nó gần với chi phí thực tế nhưng vẫn nhanh chóng tính toán được. Sử dụng một hàm đánh giá hiệu quả có thể giảm thiểu thời gian tìm kiếm và tăng hiệu suất của thuật toán tìm kiếm.

Phương pháp tìm kiếm: Có nhiều phương pháp khác nhau để tìm kiếm lời giải trên không gian trạng thái, bao gồm:

Tìm kiếm theo chiều sâu (Depth-First Search - DFS) là một thuật toán tìm kiếm trong không gian trạng thái, trong đó hệ thống hoặc tác nhân tìm kiếm từ trạng thái hiện tại theo một hành động cho đến khi đạt được trạng thái kết thúc hoặc không thể thực hiện thêm hành động nào nữa.

Thuật toán DFS hoạt động theo cơ chế "sâu trước rồi mới đi xa", nghĩa là nó sẽ lựa chọn một hành động và tiếp tục thực hiện các hành động tiếp theo từ trạng thái mới đó trước khi quay lại và kiểm tra các hành động khác từ các trạng thái trước đó. Khi không thể thực hiện thêm hành động nào nữa, thuật toán sẽ quay lại các trạng thái trước đó và thử các hành động khác.

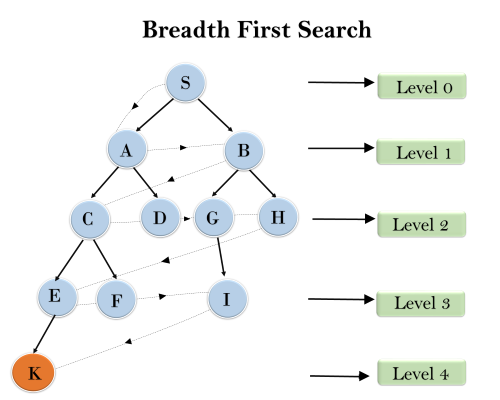
Ưu điểm của DFS là nó có thể phát hiện được các giải pháp sớm nếu một giải pháp tồn tại gần trạng thái ban đầu. Tuy nhiên, DFS có thể dẫn đến việc lạc vào vòng lặp trong không gian trạng thái lớn hoặc không có giải pháp. Đồng thời, việc sử dụng DFS có thể không đảm bảo tìm ra giải pháp tối ưu nếu có nhiều giải pháp có thể.



Tìm kiếm theo chiều rộng (Breadth-First Search - BFS) là một thuật toán tìm kiếm trong không gian trạng thái, trong đó hệ thống hoặc tác nhân tìm kiếm bằng cách thử tất cả các hành động có thể thực hiện từ trạng thái hiện tại trước khi di chuyển đến trạng thái kế tiếp.

Thuật toán BFS hoạt động theo cơ chế "rộng trước rồi mới đi sâu", nghĩa là nó sẽ tìm kiếm qua tất cả các trạng thái cùng cấp trước khi di chuyển xuống các trạng thái kế tiếp. Cụ thể, BFS duyệt qua tất cả các trạng thái con của trạng thái hiện tại trước khi di chuyển đến các trạng thái con của các trạng thái kế tiếp.

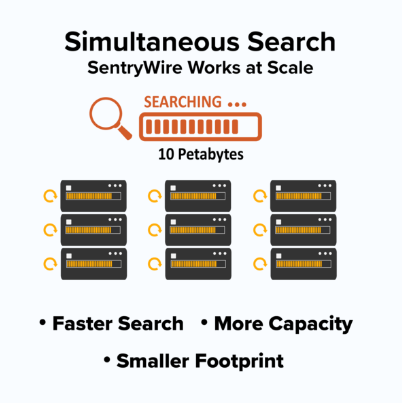
Ưu điểm của BFS là nó đảm bảo tìm ra giải pháp tối ưu đầu tiên mà nó tìm thấy. Nó cũng có thể giúp loại bỏ việc đi vào vòng lặp trong không gian trạng thái. Tuy nhiên, BFS yêu cầu một lượng lớn bộ nhớ để lưu trữ các trạng thái trung gian và có thể không hiệu quả khi sử dụng trên các không gian trạng thái lớn.



Tìm kiếm đồng thời (Simultaneous Search) là một phương pháp kết hợp cả hai phương pháp tìm kiếm theo chiều sâu và theo chiều rộng để tìm kiếm lời giải một cách hiệu quả hơn.

Trong tìm kiếm đồng thời, thuật toán sử dụng cả BFS và DFS để thực hiện tìm kiếm đồng thời trên không gian trạng thái. Cụ thể, thuật toán sẽ sử dụng BFS để duyệt qua các trạng thái ở mức độ gần trước, đồng thời sử dụng DFS để duyệt qua các trạng thái ở mức độ sâu trước. Bằng cách kết hợp cả hai phương pháp, tìm kiếm đồng thời có thể tận dụng được ưu điểm của cả hai phương pháp, đồng thời giảm thiểu được nhược điểm của từng phương pháp.

Tìm kiếm đồng thời có thể giúp tăng tốc độ tìm kiếm và giảm thiểu thời gian tìm kiếm so với việc sử dụng một phương pháp tìm kiếm duy nhất. Tuy nhiên, việc triển khai và quản lý tìm kiếm đồng thời có thể phức tạp hơn so với việc sử dụng một phương pháp tìm kiếm đơn lẻ.

****

## Câu 1.Bài toán tìm kiếm lời giải trên không gian trạng thái( đề 2)

1. Trình bày thuật toán Uniform Cost Search (UCS)
2. Ứng dụng Uniform Cost Search (UCS) giải bài toán người bán hàng
3. Cài đặt thuật toán UCS cho bài toán người bán hàng

Bài làm:

1 Trình bày thuật toán Uniform Cost Search (UCS)

- Thuật toán UCS là:một thuật toán tìm kiếm mù (uninformed search algorithm) thuật toán duyệt, tìm kiếm trên một cấu trúc cây, hoặc đồ thị có trọng số (chi phí). Việc tìm kiếm bắt đầu tại nút gốc và tiếp tục bằng cách duyệt các nút tiếp theo với trọng số hay chi phí thấp nhất tính từ nút gốc. Các nút được duyệt tiếp tục cho đến khi đến được nút đích cần đến. Hàng đợi sử dụng hàng đợi ưu tiên. UCS tìm kiếm từ trạng thái xuất phát và duyệt qua các trạng thái kế tiếp theo chi phí tăng dần, đảm bảo đó luôn luôn là đường đi ngắn nhất tới thời điểm đó.

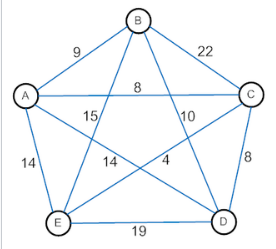
2 .Ứng dụng Uniform Cost Search (UCS) giải bài toán người bán hàng

- Ứng dụng Uniform Cost Search (UCS): dùng trong việc tìm đường đi từ các thành phố các địa điểm, tìm kiếm đường đi ngắn nhất,...

- Uniform Cost Search (UCS) giải bài toán người bán hàng :

- Bài toán người bán hàng:

+Là bài toán mà đặt ra yêu cầu mà sẽ tìm đường đi qua các đỉnh. Cho trước một danh sách các thành phố và khoảng cách giữa chúng, tìm chu trình ngắn nhất thăm mỗi thành phố đúng một lần.  
+ UCS có thể được áp dụng để giải bài toán người bán hàng(TSP) bằng cách mô hình hóa bài toán thành một đồ thị có trọng số, với các thành phố là các nút và khoảng cách giữa các thành phố là trọng số trên các cạnh. Sau đó, UCS có thể được sử dụng để tìm đường đi có chi phí tổng thể thấp nhất (tương đương với khoảng cách ngắn nhất) từ thành phố xuất phát đến tất cả các thành phố khác và quay trở lại.



+Mã giả thuật toán UCS:

***begin***

***procedure UniformCostSearch(Graph, root, goal)***

***node:= root, cost = 0***

***frontier:= priority queue containing node only***

***explored:= empty set***

***do***

***if frontier is empty***

***return failure***

***node:= frontier.pop()***

***if node is goal***

***return solution***

***explored.add(node)***

***for each of node's neighbors n***

***if n is not in explored***

***if n is not in frontier***

***frontier.add(n)***

***else if n is in frontier with higher cost***

***replace existing node with n***

Nêu ý tưởng để giải bài toán người bán hàng:

**Quy trình giải:**

1. **Mô hình hóa bài toán thành đồ thị:**
   * Tạo một nút cho mỗi thành phố.
   * Thêm các cạnh giữa các cặp thành phố với trọng số là khoảng cách giữa chúng.
2. **Khởi tạo thuật toán UCS:**
   * Đặt nút xuất phát là trạng thái hiện tại.
   * Khởi tạo một danh sách , mảng mở để lưu trữ các trạng thái tiềm năng.
   * Khởi tạo một danh sách, mảng đóng để lưu trữ các trạng thái đã được khám phá.
3. **Lặp lại cho đến khi tìm thấy trạng thái đích hoặc danh sách mở rỗng:**
   * Lấy trạng thái đầu tiên từ danh sách mở.
   * Kiểm tra xem trạng thái đó có phải là trạng thái đích hay không.
     + Nếu là trạng thái đích, thuật toán kết thúc và trả về đường đi.
     + Nếu không phải trạng thái đích, thêm trạng thái đó vào danh sách đóng.
   * Mở rộng trạng thái hiện tại:
     + Xem xét tất cả các trạng thái lân cận (các thành phố chưa được viếng thăm) của trạng thái hiện tại.
     + Tính toán chi phí tổng thể cho mỗi trạng thái lân cận bằng cách cộng chi phí của cạnh từ trạng thái hiện tại đến trạng thái lân cận với chi phí tổng thể hiện tại.
   * Thêm các trạng thái lân cận chưa được khám phá vào danh sách mở:
     + Đối với mỗi trạng thái lân cận, kiểm tra xem trạng thái đó có trong danh sách đóng hay không.
     + Nếu không có trong danh sách đóng, thêm trạng thái đó vào danh sách mở với chi phí tổng thể đã tính toán.

+ Mô tả code:

-Đầu tiên chúng ta sử dụng thuật toán UCS

-Khởi tạo các biến cần thiết (grap: đồ thị) trong nó sẽ khởi tạo kích thước của mảng và danh sách các cạnh(khoảng cách giữa các thành phố)

-Sau đó chúng ta sẽ thêm các đỉnh (các thành phố) và cạnh nối của chúng(khoảng cách giữa các thành phố). Sau khi đã thêm các đỉnh và cạnh là xử dụng thuật toán UCS. Đầu tiên chúng ta xuất phát tại đỉnh (thành phố) start là đỉnh(thành phố) mà chúng ta đưa vào khi gọi hàm (trong đây chúng ta sẽ tạo ra các biến để lưu các giá trị cần thiết một biến lưu tổng giá trị và một hàng đợi có sắp xếp một mảng các đỉnh(thành phố) đã xét và gán một con trỏ chạy là vị trí bắt đầu. Khởi chạy một vòng chạy chính để tính toán

B1: xóa hàng đợi và thêm các cạnh của vị trí hiện tại vào hàng đợi đó. Các cạnh(khoảng cách giữa các thành phố) được sắp xếp theo khoảng cách tăng dần.

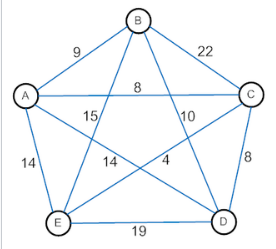
B2: Duyệt qua các phần tử hàng đợi đó nếu:

+Phần tử đang xét chưa đi qua hoặc không nằm trong các đỉnh(thành phố) đã đi thì gán vị trí hiện tại là đỉnh(thành phố) đó và cập nhật giá trị cho chi phí bằng với khoảng cách đỉnh(khoảng cách các thành phố) hiện tại đến đỉnh(thành phố) đó nếu không thì bỏ qua.

B3: Nếu tất cả các đỉnh(thành phố) mà đỉnh (thành phố) hiện tại có thể đi đến mà nằm trong danh sách các đỉnh(thành phố) đã đến thì sẽ thoát khỏi vòng lặp chính sau khi thoát khỏi vòng lặp thì sẽ cập nhật thêm chi phí đi từ đỉnh(thành phố) hiện tại đến đỉnh(thành phố) bắt đầu.

-In ra kết quả và kết thúc chương trình.

**3.** Cài đặt thuật toán UCS cho bài toán người bán hàng



Thực hiện trong code

# PHẦN II: BÀI TOÁN LOGIC

[Exercise 7 (nqueens-size-exercise)](https://aimacode.github.io/aima-exercises/fol-exercises/ex_7/)

## Câu 2. Logic(đề 2)

Thực hiện các bài tập 7, 10,11,12,27,29 và 30 trong link:

**Which of the following are valid (necessarily true) sentences?**

1. (∃x xx)⇒(∀y∃z yz)(∃𝑥 𝑥𝑥)⇒(∀𝑦∃𝑧 𝑦𝑧).  
2. ∀xP(x)∨¬P(x)∀𝑥𝑃(𝑥)∨¬𝑃(𝑥).  
3. ∀xSmart(x)∨(xx)∀𝑥𝑆𝑚𝑎𝑟𝑡(𝑥)∨(𝑥𝑥).

Dịch

### **Bài tập 7** (bài tập nqueens-size)

**Câu nào sau đây là câu hợp lệ (nhất thiết phải đúng)?**

1. (∃x xx)⇒(∀y∃z yz)(∃𝑥 𝑥𝑥)⇒(∀𝑦∃𝑧 𝑦𝑧).

2. ∀xP(x)∨иP(x)∀𝑥𝑃(𝑥)∨ и𝑃(𝑥).

3. ∀xSmart(x)∨(xx)∀𝑥𝑆𝑚𝑎𝑟𝑡(𝑥)∨(𝑥𝑥).

Giải

Tất cả ba câu đều đề cập đến logic trong toán học và logic propositional.

**a. (∃𝑥𝑥=𝑥)⇒(∀𝑦∃𝑧𝑦=𝑧):**

Để xác định tính chất của (∃𝑥𝑥=𝑥), ta phải hiểu rằng nó nói rằng "tồn tại một phần tử 𝑥 sao cho 𝑥 bằng chính nó". Trong logic tập hợp, điều này luôn đúng vì mọi phần tử trong một tập hợp đều bằng chính nó.

Bây giờ, chúng ta xem xét phần (∀𝑦∃𝑧𝑦=𝑧). Điều này nghĩa là "đối với mọi phần tử 𝑦, tồn tại một phần tử 𝑧 sao cho 𝑦 bằng 𝑧". Vì mọi phần tử 𝑦 đều có thể là chính nó (𝑧 = 𝑦), nên phần này cũng là đúng.

Khi (∃𝑥𝑥=𝑥) là đúng, thì (∀𝑦∃𝑧𝑦=𝑧) cũng phải là đúng, vì mọi giá trị 𝑦 đều có thể là giá trị 𝑧 (giá trị bằng chính nó).

Vậy, toàn bộ biểu thức là đúng.

**b. ∀𝑥𝑃(𝑥)∨¬𝑃(𝑥):**

Biểu thức này nói rằng "đối với mọi phần tử 𝑥, 𝑃(𝑥) đúng hoặc phủ định của 𝑃(𝑥) đúng".

Trong logic propositional, bất kỳ mệnh đề nào dạng này cũng là đúng. Nếu 𝑃(𝑥) đúng, thì 𝑃(𝑥)∨¬𝑃(𝑥) cũng đúng. Nếu 𝑃(𝑥) sai, thì ¬𝑃(𝑥) đúng và 𝑃(𝑥)∨¬𝑃(𝑥) cũng đúng.

Vậy, biểu thức này cũng là đúng.

**c. ∀𝑥Smart(𝑥)∨(𝑥=𝑥):**

∀𝑥: Đây là một mệnh đề về tất cả các phần tử 𝑥 trong miền được xét.

Smart(𝑥)∨(𝑥=𝑥): Đây là một phát biểu logic propositional. Nó nói rằng "hoặc 𝑥 thông minh hoặc 𝑥 bằng chính nó". Vì mọi thứ đều bằng chính nó, vì vậy phần (𝑥=𝑥) luôn đúng. Và do đó, dù 𝑥 có thông minh hay không, phát biểu này vẫn đúng.

[Exercise 10](https://aimacode.github.io/aima-exercises/fol-exercises/ex_10/)

This exercise uses the function MapColor𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟 and predicates In(x,y)𝐼𝑛(𝑥,𝑦), Borders(x,y)𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦), and Country(x)𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥), whose arguments are geographical regions, along with constant symbols for various regions. In each of the following we give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

**1. Paris and Marseilles are both in France.**

1. In(Paris∧Marseilles,France)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠∧𝑀𝑎𝑟𝑠𝑒𝑖𝑙𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).  
2. In(Paris,France)∧In(Marseilles,France)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠,𝐹𝑟𝑎𝑛𝑐𝑒)∧𝐼𝑛(𝑀𝑎𝑟𝑠𝑒𝑖𝑙𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).  
3. In(Paris,France)∨In(Marseilles,France)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠,𝐹𝑟𝑎𝑛𝑐𝑒)∨𝐼𝑛(𝑀𝑎𝑟𝑠𝑒𝑖𝑙𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).

**2. There is a country that borders both Iraq and Pakistan.**

1. ∃c∃𝑐 Country(c)∧Border(c,Iraq)∧Border(c,Pakistan)𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼𝑟𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛).  
2. ∃c∃𝑐 Country(c)⇒[Border(c,Iraq)∧Border(c,Pakistan)]𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼𝑟𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛)].  
3. [∃c[∃𝑐 Country(c)]⇒[Border(c,Iraq)∧Border(c,Pakistan)]𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)]⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼𝑟𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛)].  
4. ∃c∃𝑐 Border(Country(c),Iraq∧Pakistan)𝐵𝑜𝑟𝑑𝑒𝑟(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐),𝐼𝑟𝑎𝑞∧𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛).  
**3. All countries that border Ecuador are in South America.**

1. ∀cCountry(c)∧Border(c,Ecuador)⇒In(c,SouthAmerica)∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐𝑢𝑎𝑑𝑜𝑟)⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).  
2. ∀cCountry(c)⇒[Border(c,Ecuador)⇒In(c,SouthAmerica)]∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐𝑢𝑎𝑑𝑜𝑟)⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)].  
3. ∀c[Country(c)⇒Border(c,Ecuador)]⇒In(c,SouthAmerica)∀𝑐[𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐𝑢𝑎𝑑𝑜𝑟)]⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).  
4. ∀cCountry(c)∧Border(c,Ecuador)∧In(c,SouthAmerica)∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐𝑢𝑎𝑑𝑜𝑟)∧𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).  
**4. No region in South America borders any region in Europe.**

1. ¬[∃c,dIn(c,SouthAmerica)∧In(d,Europe)∧Borders(c,d)]¬[∃𝑐,𝑑𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)∧𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑)].  
2. ∀c,d[In(c,SouthAmerica)∧In(d,Europe)]⇒¬Borders(c,d)]∀𝑐,𝑑[𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)∧𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)]⇒¬𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑)].  
3. ¬∀cIn(c,SouthAmerica)⇒∃dIn(d,Europe)∧¬Borders(c,d)¬∀𝑐𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)⇒∃𝑑𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)∧¬𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑). 4. ∀cIn(c,SouthAmerica)⇒∀dIn(d,Europe)⇒¬Borders(c,d)∀𝑐𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)⇒∀𝑑𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)⇒¬𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑).  
**5. No two adjacent countries have the same map color.**

1. ∀x,y¬Country(x)∨¬Country(y)∨¬Borders(x,y)∨∀𝑥,𝑦¬𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∨¬𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦)∨¬𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∨\ ¬(MapColor(x)=MapColor(y))¬(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).  
2. ∀x,y(Country(x)∧Country(y)∧Borders(x,y)∧¬(x=y))⇒∀𝑥,𝑦(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∧¬(𝑥=𝑦))⇒\ ¬(MapColor(x)=MapColor(y))¬(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).  
3. ∀x,yCountry(x)∧Country(y)∧Borders(x,y)∧∀𝑥,𝑦𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∧\ ¬(MapColor(x)=MapColor(y))¬(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).  
4. ∀x,y(Country(x)∧Country(y)∧Borders(x,y))⇒MapColor(x≠y)∀𝑥,𝑦(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦))⇒𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥≠𝑦).

Dịch

### Bài tập 10

Bài tập này sử dụng hàm MapColor𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟 và các vị từ In(x,y)𝐼𝑛(𝑥,𝑦), Borders(x,y)𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦), và Country(x)𝐶 𝑜𝑢𝑛𝑡𝑟𝑦(𝑥), có đối số là các khu vực địa lý, cùng với các ký hiệu không đổi cho các vùng khác nhau. Trong mỗi phần sau đây, chúng tôi đưa ra một câu tiếng Anh và một số biểu thức logic ứng cử viên. Đối với mỗi biểu thức logic, hãy cho biết liệu nó (1) có diễn đạt chính xác câu tiếng Anh hay không; (2) không hợp lệ về mặt cú pháp và do đó vô nghĩa; hoặc (3) có giá trị về mặt cú pháp nhưng không thể hiện được nghĩa của câu tiếng Anh.

**1. Paris và Marseilles đều ở Pháp.**

1. Ở (Paris∧Marseilles,Pháp)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠∧𝑀𝑎𝑟𝑠𝑒𝑖𝑙𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).

2.In(Paris,France)∧In(Marseilles,France)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠,𝐹𝑟𝑎𝑛𝑐𝑒)∧𝐼𝑛(𝑀𝑎𝑟𝑠𝑒𝑖𝑙 𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).

3.In(Paris,France)∨In(Marseilles,France)𝐼𝑛(𝑃𝑎𝑟𝑖𝑠,𝐹𝑟𝑎𝑛𝑐𝑒)∨𝐼𝑛(𝑀𝑎𝑟𝑠𝑒𝑖𝑙 𝑙𝑒𝑠,𝐹𝑟𝑎𝑛𝑐𝑒).

**2. Có một quốc gia giáp cả Iraq và Pakistan.**

1. ∃c∃𝑐 Quốc gia(c)∧Border(c,Iraq)∧Border(c,Pakistan)𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼𝑟 𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛).

2.∃c∃𝑐Quốcgia(c)⇒[Border(c,Iraq)∧Border(c,Pakistan)]𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼𝑟 𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛)].

3.[∃c[∃𝑐Quốcgia(c)]⇒[Border(c,Iraq)∧Border(c,Pakistan)]𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)]⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐼 𝑟𝑎𝑞)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝑃𝑎𝑘𝑖𝑠𝑡𝑎𝑛)] .

4.∃c∃𝑐Biêngiới(Quốcgia(c),Iraq∧Pakistan)𝐵𝑜𝑟𝑑𝑒𝑟(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐),𝐼𝑟𝑎𝑞∧𝑃𝑎 𝑘𝑖𝑠𝑡𝑎𝑛).

**3. Tất cả các quốc gia giáp Ecuador đều ở Nam Mỹ.**

1.∀cQuốcgia(c)∧Border(c,Ecuador)⇒In(c,SouthAmerica)∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐 𝑢𝑎𝑑𝑜𝑟)⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).

2.∀cQuốcgia(c)⇒[Border(c,Ecuador)⇒In(c,SouthAmerica)]∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒[𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸 𝑐𝑢𝑎𝑑𝑜𝑟)⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)].

3.∀c[Quốcgia(c)⇒Border(c,Ecuador)]⇒In(c,SouthAmerica)∀𝑐[𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)⇒𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸 𝑐𝑢𝑎𝑑𝑜𝑟)]⇒𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).

4.∀cCountry(c)∧Border(c,Ecuador)∧In(c,SouthAmerica)∀𝑐𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑐)∧𝐵𝑜𝑟𝑑𝑒𝑟(𝑐,𝐸𝑐 𝑢𝑎𝑑𝑜𝑟)∧𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎).

**4. Không có khu vực nào ở Nam Mỹ giáp với bất kỳ khu vực nào ở Châu Âu.**

1.и[∃c,dIn(c,SouthAmerica)∧In(d,Europe)∧Borders(c,d)]и[∃𝑐,𝑑𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎) ∧𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐 ,𝑑)].

2.∀c,d[In(c,SouthAmerica)∧In(d,Europe)]⇒€Borders(c,d)]∀𝑐,𝑑[𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎) ∧𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)]⇒ и𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑)].

3.и∀cIn(c,SouthAmerica)⇒∃dIn(d,Europe)∧€Borders(c,d)€∀𝑐𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)⇒∃𝑑𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)∧и𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑) .

4.∀cIn(c,SouthAmerica)⇒∀dIn(d,Europe)⇒€Borders(c,d)∀𝑐𝐼𝑛(𝑐,𝑆𝑜𝑢𝑡ℎ𝐴𝑚𝑒𝑟𝑖𝑐𝑎)⇒∀𝑑 𝐼𝑛(𝑑,𝐸𝑢𝑟𝑜𝑝𝑒)⇒и𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑐,𝑑).

**5. Không có hai quốc gia liền kề nào có màu bản đồ giống nhau.**

1.∀x,y-Quốcgia(x)∨-Quốcgia(y)∨-Borders(x,y)∨∀𝑥,𝑦ctured (𝑦)∨и𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∨и(MapColor(x)=MapColor(y))€(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).

2.∀x,y(Quốcgia(x)∧Quốcgia(y)∧Borders(x,y)∧и(x=y))⇒∀𝑥,𝑦(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∧и(𝑥=𝑦))⇒\и(MapColor(x)=MapColor(y))€(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).

3.∀x,yCountry(x)∧Country(y)∧Borders(x,y)∧∀𝑥,𝑦𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑦(𝑦) )∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥,𝑦)∧\ и(MapColor(x)= MapColor(y))€(𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥)=𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑦)).

4.∀x,y(Quốcgia(x)∧Quốc gia(y)∧Borders(x,y))⇒MapColor(x≠y)∀𝑥,𝑦(𝐶𝑜𝑢𝑛𝑡𝑟𝑦(𝑥)∧𝐶𝑜𝑢𝑛𝑡 𝑟𝑦(𝑦)∧𝐵𝑜𝑟𝑑𝑒𝑟𝑠(𝑥, 𝑦))⇒𝑀𝑎𝑝𝐶𝑜𝑙𝑜𝑟(𝑥≠𝑦).

Giải

**1 Paris và Marseille đều ở Pháp.**

a. In(Paris∧Marseilles,France): (2) Cú pháp không hợp lệ và vô nghĩa về mặt cú pháp vì nó cố gắng chỉ ra cả Paris và Marseille đều ở Pháp trong cùng một đối số, điều này không hợp lệ.

b. In(Paris,France)∧In(Marseilles,France): (1) Đúng. Biểu thức này nói rằng Paris ở Pháp VÀ Marseille ở Pháp, phù hợp với câu tiếng Anh.

c. In(Paris,France)∨In(Marseilles,France): (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này nói rằng hoặc Paris ở Pháp HOẶC Marseille ở Pháp, điều này không thể hiện được cả hai thành phố đều ở Pháp.

**2 Có một quốc gia giáp cả Iraq và Pakistan.**

a. ∃c Country(c)∧Border(c,Iraq)∧Border(c,Pakistan): (1) Đúng. Biểu thức này nói rằng có một quốc gia 𝑐, là một quốc gia, giáp với Iraq và giáp với Pakistan.

b. ∃c Country(c)⇒[Border(c,Iraq)∧Border(c,Pakistan)]: (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này gợi ý rằng nếu có một quốc gia, thì nó giáp cả Iraq và Pakistan, điều này không đúng với câu tiếng Anh.

c. [∃c Country(c)]⇒[Border(c,Iraq)∧Border(c,Pakistan)]: (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này tương tự như biểu thức trước và không diễn đạt đúng mối quan hệ.

d. ∃c Border(Country(c),Iraq∧Pakistan): (2) Cú pháp không hợp lệ và vô nghĩa về mặt cú pháp vì nó cố gắng kết hợp giữa biểu thức Border với sự giao nhau giữa Iraq và Pakistan, điều này không phải là cú pháp hợp lệ.

**3 Tất cả các quốc gia giáp với Ecuador đều ở Nam Mỹ.**

a. ∀cCountry(c)∧Border(c,Ecuador)⇒In(c,SouthAmerica): (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này nói rằng nếu một quốc gia giáp với Ecuador, thì nó ở Nam Mỹ, nhưng nó không giới hạn việc đếm các quốc gia đó.

b. ∀cCountry(c)⇒[Border(c,Ecuador)⇒In(c,SouthAmerica)]: (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này có cùng vấn đề như biểu thức trước, nó không đếm chính xác các quốc gia giáp với Ecuador

c. ∀c[Country(c)⇒Border(c,Ecuador)]⇒In(c,SouthAmerica): (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này gợi ý rằng mỗi quốc gia hoặc giáp Ecuador hoặc ở Nam Mỹ, điều này không đúng với câu tiếng Anh.

d.∀cCountry(c)∧Border(c,Ecuador)∧In(c,SouthAmerica): (1) Đúng. Biểu thức này nói rằng đối với mọi quốc gia 𝑐, nếu nó giáp với Ecuador, thì nó ở Nam Mỹ.

**4 Không có khu vực nào ở Nam Mỹ giáp với bất kỳ khu vực nào ở Châu Âu.**

a. ¬[∃c,dIn(c,SouthAmerica)∧In(d,Europe)∧Borders(c,d)]: (1) Đúng. Biểu thức này phủ nhận sự tồn tại của bất kỳ cặp khu vực nào, một ở Nam Mỹ và một ở Châu Âu, mà có chung biên giới.

b. ∀c,d[In(c,SouthAmerica)∧In(d,Europe)]⇒¬Borders(c,d)]: (2) Cú pháp không hợp lệ và vô nghĩa về mặt cú pháp vì nó cố gắng sử dụng phép gợi ý một cách không hợp lý trong ngữ cảnh của câu tiếng Anh.

c. ¬∀cIn(c,SouthAmerica)⇒∃dIn(d,Europe)∧¬Borders(c,d): (3) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này không phủ nhận một cách chính xác tất cả các khu vực ở Nam Mỹ giáp với các khu vực ở Châu Âu.

d. ∀cIn(c,SouthAmerica)⇒∀dIn(d,Europe)⇒¬Borders(c,d): (4) Cú pháp hợp lệ nhưng không diễn đạt đúng ý nghĩa của câu tiếng Anh. Biểu thức này không phủ nhận một cách chính xác tất cả các khu vực ở Nam Mỹ giáp với các khu vực ở Châu Âu.

**5 Không có hai quốc gia giáp nhau có màu bản đồ giống nhau.**

a. ∀x,y¬Country(x)∨¬Country(y)∨¬Borders(x,y)∨¬(MapColor(x)=MapColor(y)): (1) Đúng. Biểu thức này nói rằng cho mọi cặp quốc gia, nếu chúng giáp nhau thì chúng có màu bản đồ khác nhau.

b.∀x,y(Country(x)∧Country(y)∧Borders(x,y)∧¬(x=y))⇒¬(MapColor(x)=MapColor(y)): (2) Không diễn đạt đúng ý nghĩa. Biểu thức này không xác định rõ ràng rằng chỉ có các quốc gia kề nhau mới được xem xét.

c. ∀x,yCountry(x)∧Country(y)∧Borders(x,y)∧¬(MapColor(x)=MapColor(y)): (3) Không diễn đạt đúng ý nghĩa. Biểu thức này không xác định rõ ràng rằng chỉ có các quốc gia kề nhau mới được xem xét.

d. ∀x,y(Country(x)∧Country(y)∧Borders(x,y))⇒MapColor(x≠y): (4) Không diễn đạt đúng ý nghĩa. Biểu thức này không xác định rõ ràng rằng chỉ có các quốc gia kề nhau mới được xem xét và không diễn đạt đúng về việc màu bản đồ của các quốc gia.

// TÓM TẮT

**1 Paris và Marseille đều ở Pháp.**

a. Không hợp lệ.

b. Đúng.

c. Không diễn đạt đúng ý nghĩa.

**2 Có một quốc gia giáp cả Iraq và Pakistan.**

a. Đúng.

b. Không diễn đạt đúng ý nghĩa.

c. Không diễn đạt đúng ý nghĩa.

d. Không hợp lệ.

**3 Tất cả các quốc gia giáp với Ecuador đều ở Nam Mỹ.**

a. Không diễn đạt đúng ý nghĩa.

b. Không diễn đạt đúng ý nghĩa.

c. Không diễn đạt đúng ý nghĩa.

d. Đúng.

**4 Không có khu vực nào ở Nam Mỹ giáp với bất kỳ khu vực nào ở Châu Âu.**

a. Đúng.

b. Không hợp lệ.

c. Không diễn đạt đúng ý nghĩa.

**5 Không có hai quốc gia giáp nhau có màu bản đồ giống nhau.**

a. Đúng.

b. Không diễn đạt đúng ý nghĩa.

c. Không diễn đạt đúng ý nghĩa.

d. Không diễn đạt đúng ý nghĩa.

Exercise 11

Consider a vocabulary with the following symbols:

> Occupation(p,o)𝑂𝑐𝑐𝑢𝑝𝑎𝑡𝑖𝑜𝑛(𝑝,𝑜): Predicate. Person p𝑝 has occupation o𝑜. > Customer(p1,p2)𝐶𝑢𝑠𝑡𝑜𝑚𝑒𝑟(𝑝1,𝑝2): Predicate. Person p1𝑝1 is a customer of person p2𝑝2. > Boss(p1,p2)𝐵𝑜𝑠𝑠(𝑝1,𝑝2): Predicate. Person p1𝑝1 is a boss of person p2𝑝2. > Doctor𝐷𝑜𝑐𝑡𝑜𝑟, Surgeon𝑆𝑢𝑟𝑔𝑒𝑜𝑛, Lawyer𝐿𝑎𝑤𝑦𝑒𝑟, Actor𝐴𝑐𝑡𝑜𝑟: Constants denoting occupations. > Emily𝐸𝑚𝑖𝑙𝑦, Joe𝐽𝑜𝑒: Constants denoting people. Use these symbols to write the following assertions in first-order logic:

1. Emily is either a surgeon or a lawyer.

2. Joe is an actor, but he also holds another job.

3. All surgeons are doctors.

4. Joe does not have a lawyer (i.e., is not a customer of any lawyer).

5. Emily has a boss who is a lawyer.

6. There exists a lawyer all of whose customers are doctors.

7. Every surgeon has a lawyer.

Dịch

### Bài tập 11

Hãy xem xét một từ vựng có các ký hiệu sau:

> Nghề nghiệp(p,o)𝑂𝑐𝑐𝑢𝑝𝑎𝑡𝑖𝑜𝑛(𝑝,𝑜): Vị ngữ. Người p𝑝 có nghề nghiệp o𝑜. > Khách hàng(p1,p2)𝐶𝑢𝑠𝑡𝑜𝑚𝑒𝑟(𝑝1,𝑝2): Vị ngữ. Người p1𝑝1 là khách hàng của người p2𝑝2. > Boss(p1,p2)𝐵𝑜𝑠𝑠(𝑝1,𝑝2): Vị ngữ. Người p1𝑝1 là ông chủ của người p2𝑝2. > Bác sĩ𝐷𝑜𝑐𝑡𝑜𝑟, Bác sĩ phẫu thuật𝑆𝑢𝑟𝑔𝑒𝑜𝑛, Luật sư𝐿𝑎𝑤𝑦𝑒𝑟, Diễn viên𝐴𝑐𝑡𝑜𝑟: Các hằng số biểu thị nghề nghiệp. > Emily𝐸𝑚𝑖𝑙𝑦, Joe𝐽𝑜𝑒: Các hằng số biểu thị con người. Sử dụng các ký hiệu này để viết các xác nhận sau theo logic bậc nhất:

1. Emily là bác sĩ phẫu thuật hoặc luật sư.

2. Joe là một diễn viên nhưng anh ấy còn làm một công việc khác.

3. Tất cả bác sĩ phẫu thuật đều là bác sĩ.

4. Joe không có luật sư (tức là không phải là khách hàng của bất kỳ luật sư nào).

5. Emily có ông chủ là luật sư.

6. Tồn tại một luật sư mà khách hàng của họ đều là bác sĩ.

7. Mỗi bác sĩ phẫu thuật đều có luật sư.

Giải

**1 Emily là bác sĩ phẫu thuật hoặc luật sư.**

Định danh: Emily

Mệnh đề logic: Emily có chức năng là bác sĩ phẫu thuật hoặc luật sư.

Biểu thức logic:

Occupation(Emily,Surgeon)∨Occupation(Emily,Lawyer)Occupation(Emily,Surgeon)∨Occupation(Emily,Lawyer)

Biểu thức này nói rằng Emily có thể là bác sĩ phẫu thuật hoặc luật sư. Nếu Emily là bác sĩ phẫu thuật, mệnh đề đầu tiên sẽ đúng; nếu Emily là luật sư, mệnh đề thứ hai sẽ đúng.

**2 Joe là diễn viên, nhưng anh ấy cũng đang làm một công việc khác.**

Định danh: Joe

Mệnh đề logic: Joe là diễn viên và Joe còn làm một công việc khác.

Biểu thức logic:

Occupation(Joe,Actor)∧∃𝑜≠Actor Occupation(Joe,𝑜)Occupation(Joe,Actor)∧∃o=ActorOccupation(Joe,o)

Biểu thức này nói rằng Joe là diễn viên và còn làm một công việc khác ngoài diễn viên. Mệnh đề thứ hai cho biết có một công việc o khác với việc làm diễn viên mà Joe đang làm.

**3 Tất cả các bác sĩ phẫu thuật đều là bác sĩ.**

Mệnh đề logic: Mọi người là bác sĩ phẫu thuật đều cũng là bác sĩ.

Biểu thức logic:

∀𝑝 (Occupation(𝑝,Surgeon)⇒Occupation(𝑝,Doctor))∀p(Occupation(p,Surgeon)⇒Occupation(p,Doctor))

Biểu thức này nói rằng nếu ai đó là bác sĩ phẫu thuật, họ cũng là bác sĩ. Mọi cá nhân p đều phải thỏa mãn điều kiện này.

**4 Joe không có luật sư (tức là không phải là khách hàng của bất kỳ luật sư nào).**

Định danh: Joe

Mệnh đề logic: Joe không là khách hàng của bất kỳ luật sư nào.

Biểu thức logic:

¬∃𝑥 (Occupation(𝑥,Lawyer)∧Customer(Joe,𝑥))¬∃x(Occupation(x,Lawyer)∧Customer(Joe,x))

Biểu thức này nói rằng không có luật sư nào là khách hàng của Joe. Nếu tồn tại một luật sư là khách hàng của Joe, mệnh đề sẽ sai.

**5 Emily có một sếp là luật sư.**

Định danh: Emily

Mệnh đề logic: Emily có một sếp và sếp đó là luật sư.

Biểu thức logic:

∃𝑥 (Boss(𝑥,Emily)∧Occupation(𝑥,Lawyer))∃x(Boss(x,Emily)∧Occupation(x,Lawyer))

Biểu thức này nói rằng có một người làm sếp của Emily và người đó là luật sư. Mệnh đề này nói rằng Emily có một sếp là luật sư.

**6 Tồn tại một luật sư mà tất cả khách hàng của anh ta đều là bác sĩ.**

Mệnh đề logic: Tồn tại một luật sư mà tất cả khách hàng của anh ta đều là bác sĩ.

Biểu thức logic:

∃𝑥 (Occupation(𝑥,Lawyer)∧∀𝑦 (Customer(𝑦,𝑥)⇒Occupation(𝑦,Doctor)))∃x(Occupation(x,Lawyer)∧∀y(Customer(y,x)⇒Occupation(y,Doctor)))

Biểu thức này nói rằng có một luật sư mà tất cả khách hàng của anh ta đều là bác sĩ. Một luật sư x nào đó cần phải làm nghề luật và mọi người y là khách hàng của anh ta thì cũng phải là bác sĩ.

**7 Mỗi bác sĩ phẫu thuật đều có một luật sư.**

Mệnh đề logic: Mỗi bác sĩ phẫu thuật đều có một luật sư.

Biểu thức logic:

∀𝑝 (Occupation(𝑝,Surgeon)⇒∃𝑥 (Boss(𝑥,𝑝)∧Occupation(𝑥,Lawyer)))∀p(Occupation(p,Surgeon)⇒∃x(Boss(x,p)∧Occupation(x,Lawyer)))

Biểu thức này nói rằng mỗi bác sĩ phẫu thuật đều có một luật sư làm sếp của họ. Với mọi bác sĩ phẫu thuật p, tồn tại một người làm sếp x của họ và người đó cũng phải là luật sư.

[**Exercise 12**](https://aimacode.github.io/aima-exercises/fol-exercises/ex_12/)

In each of the following we give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

**1. Every cat loves its mother or father.**

1. ∀xCat(x)⇒Loves(x,Mother(x)∨Father(x))∀𝑥𝐶𝑎𝑡(𝑥)⇒𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟(𝑥)∨𝐹𝑎𝑡ℎ𝑒𝑟(𝑥)).  
2. ∀x¬Cat(x)∨Loves(x,Mother(x))∨Loves(x,Father(x))∀𝑥¬𝐶𝑎𝑡(𝑥)∨𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟(𝑥))∨𝐿𝑜𝑣𝑒𝑠(𝑥,𝐹𝑎𝑡ℎ𝑒𝑟(𝑥)).  
3. ∀xCat(x)∧(Loves(x,Mother(x))∨Loves(x,Father(x)))∀𝑥𝐶𝑎𝑡(𝑥)∧(𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟(𝑥))∨𝐿𝑜𝑣𝑒𝑠(𝑥,𝐹𝑎𝑡ℎ𝑒𝑟(𝑥))).  
**2. Every dog who loves one of its brothers is happy.**

1. ∀xDog(x)∧(∃y Brother(y,x)∧Loves(x,y))⇒Happy(x)∀𝑥𝐷𝑜𝑔(𝑥)∧(∃𝑦 𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥)∧𝐿𝑜𝑣𝑒𝑠(𝑥,𝑦))⇒𝐻𝑎𝑝𝑝𝑦(𝑥).  
2. ∀x,yDog(x)∧Brother(y,x)∧Loves(x,y)⇒Happy(x)∀𝑥,𝑦𝐷𝑜𝑔(𝑥)∧𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥)∧𝐿𝑜𝑣𝑒𝑠(𝑥,𝑦)⇒𝐻𝑎𝑝𝑝𝑦(𝑥).  
3. ∀xDog(x)∧[∀yBrother(y,x)⇔Loves(x,y)]⇒Happy(x)∀𝑥𝐷𝑜𝑔(𝑥)∧[∀𝑦𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥)⇔𝐿𝑜𝑣𝑒𝑠(𝑥,𝑦)]⇒𝐻𝑎𝑝𝑝𝑦(𝑥).

**3. No dog bites a child of its owner.**

1. ∀xDog(x)⇒¬Bites(x,Child(Owner(x)))∀𝑥𝐷𝑜𝑔(𝑥)⇒¬𝐵𝑖𝑡𝑒𝑠(𝑥,𝐶ℎ𝑖𝑙𝑑(𝑂𝑤𝑛𝑒𝑟(𝑥)).  
2. ¬∃x,yDog(x)∧Child(y,Owner(x))∧Bites(x,y)¬∃𝑥,𝑦𝐷𝑜𝑔(𝑥)∧𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛𝑒𝑟(𝑥))∧𝐵𝑖𝑡𝑒𝑠(𝑥,𝑦).  
3. ∀xDog(x)⇒(∀yChild(y,Owner(x))⇒¬Bites(x,y))∀𝑥𝐷𝑜𝑔(𝑥)⇒(∀𝑦𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛𝑒𝑟(𝑥))⇒¬𝐵𝑖𝑡𝑒𝑠(𝑥,𝑦)).  
4. ¬∃xDog(x)⇒(∃yChild(y,Owner(x))∧Bites(x,y))¬∃𝑥𝐷𝑜𝑔(𝑥)⇒(∃𝑦𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛𝑒𝑟(𝑥))∧𝐵𝑖𝑡𝑒𝑠(𝑥,𝑦)).

**4. Everyone’s zip code within a state has the same first digit.**  
1. ∀x,s,z1[State(s)∧LivesIn(x,s)∧Zip(x)z1]⇒∀𝑥,𝑠,𝑧1[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧𝑍𝑖𝑝(𝑥)𝑧1]⇒\ [∀y,z2LivesIn(y,s)∧Zip(y)z2⇒Digit(1,z1)Digit(1,z2)][∀𝑦,𝑧2𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖𝑝(𝑦)𝑧2⇒𝐷𝑖𝑔𝑖𝑡(1,𝑧1)𝐷𝑖𝑔𝑖𝑡(1,𝑧2)].  
2. ∀x,s[State(s)∧LivesIn(x,s)∧∃z1Zip(x)z1]⇒∀𝑥,𝑠[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧∃𝑧1𝑍𝑖𝑝(𝑥)𝑧1]⇒\ [∀y,z2LivesIn(y,s)∧Zip(y)z2∧Digit(1,z1)Digit(1,z2)][∀𝑦,𝑧2𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖𝑝(𝑦)𝑧2∧𝐷𝑖𝑔𝑖𝑡(1,𝑧1)𝐷𝑖𝑔𝑖𝑡(1,𝑧2)].  
3. ∀x,y,sState(s)∧LivesIn(x,s)∧LivesIn(y,s)⇒Digit(1,Zip(x)Zip(y))∀𝑥,𝑦,𝑠𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)⇒𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑥)𝑍𝑖𝑝(𝑦)).  
4. ∀x,y,sState(s)∧LivesIn(x,s)∧LivesIn(y,s)⇒∀𝑥,𝑦,𝑠𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)⇒\ Digit(1,Zip(x))Digit(1,Zip(y))𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑥))𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑦)).

Dịch

### Bài tập 12

Trong mỗi phần sau đây, chúng tôi đưa ra một câu tiếng Anh và một số biểu thức logic ứng cử viên. Đối với mỗi biểu thức logic, hãy cho biết liệu nó (1) có diễn đạt chính xác câu tiếng Anh hay không; (2) không hợp lệ về mặt cú pháp và do đó vô nghĩa; hoặc (3) có giá trị về mặt cú pháp nhưng không thể hiện được nghĩa của câu tiếng Anh.

**1. Con mèo nào cũng yêu bố hoặc mẹ của mình.**

1. ∀xCat(x)⇒Loves(x,Mẹ(x)∨Cha(x))∀𝑥𝐶𝑎𝑡(𝑥)⇒𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟(𝑥)∨ 𝐹𝑎𝑡ℎ𝑒𝑟(𝑥)).

2. ∀x€Cat(x)∨Loves(x,Mẹ(x))∨Loves(x,Cha(x))∀𝑥и𝐶𝑎𝑡(𝑥)∨𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟( 𝑥))∨𝐿𝑜𝑣𝑒𝑠(𝑥 ,𝐹𝑎𝑡ℎ𝑒𝑟(𝑥)).

3. ∀xCat(x)∧(Loves(x,Mẹ(x))∨Loves(x,Cha(x)))∀𝑥𝐶𝑎𝑡(𝑥)∧(𝐿𝑜𝑣𝑒𝑠(𝑥,𝑀𝑜𝑡ℎ𝑒𝑟( 𝑥))∨𝐿𝑜𝑣𝑒𝑠(𝑥, 𝐹𝑎𝑡ℎ𝑒𝑟(𝑥))).

2. Con chó nào yêu được anh em của mình thì hạnh phúc.

1. ∀xDog(x)∧(∃y Brother(y,x)∧Loves(x,y))⇒Happy(x)∀𝑥𝐷𝑜𝑔(𝑥)∧(∃𝑦 𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥)∧ 𝐿𝑜𝑣𝑒𝑠(𝑥, 𝑦))⇒𝐻𝑎𝑝𝑝𝑦(𝑥).

2. ∀x,yDog(x)∧Brother(y,x)∧Loves(x,y)⇒Happy(x)∀𝑥,𝑦𝐷𝑜𝑔(𝑥)∧𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥)∧𝐿 𝑜𝑣𝑒𝑠(𝑥,𝑦)⇒ 𝐻𝑎𝑝𝑝𝑦(𝑥).

3.∀xDog(x)∧[∀yBrother(y,x)⇔Loves(x,y)]⇒Happy(x)∀𝑥𝐷𝑜𝑔(𝑥)∧[∀𝑦𝐵𝑟𝑜𝑡ℎ𝑒𝑟(𝑦,𝑥) ⇔𝐿𝑜𝑣𝑒𝑠(𝑥,𝑦) ]⇒𝐻𝑎𝑝𝑝𝑦(𝑥).

**3. Không có con chó nào cắn con của chủ nó.**

1. ∀xDog(x)⇒€Bites(x,Child(Chủ sở hữu(x)))∀𝑥𝐷𝑜𝑔(𝑥)⇒и𝐵𝑖𝑡𝑒𝑠(𝑥,𝐶ℎ𝑖𝑙𝑑(𝑂𝑤𝑛 𝑒𝑟(𝑥)).

2. и∃x,yDog(x)∧Child(y,Owner(x))∧Bites(x,y)и∃𝑥,𝑦𝐷𝑜𝑔(𝑥)∧𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛𝑒𝑟( 𝑥))∧𝐵𝑖𝑡𝑒𝑠(𝑥, 𝑦).

3. ∀xDog(x)⇒(∀yChild(y,Owner(x))⇒€Bites(x,y))∀𝑥𝐷𝑜𝑔(𝑥)⇒(∀𝑦𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛 𝑒𝑟(𝑥))⇒и𝐵𝑖𝑡𝑒𝑠(𝑥 ,𝑦)).

4. и∃xDog(x)⇒(∃yChild(y,Owner(x))∧Bites(x,y))€∃𝑥𝐷𝑜𝑔(𝑥)⇒(∃𝑦𝐶ℎ𝑖𝑙𝑑(𝑦,𝑂𝑤𝑛 𝑒𝑟(𝑥))∧𝐵𝑖𝑡𝑒𝑠(𝑥 ,𝑦)).

**4. Mã zip của mọi người trong một tiểu bang đều có chữ số đầu tiên giống nhau.**

1. ∀x,s,z1[(các) Bang∧LivesIn(x,s)∧Zip(x)z1]⇒∀𝑥,𝑠,𝑧1[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛( 𝑥,𝑠)∧𝑍𝑖𝑝(𝑥 )𝑧1]⇒\ [∀y,z2LivesIn(y,s)∧Zip(y)z2⇒Digit(1,z1)Digit(1,z2)][∀𝑦,𝑧2𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖 𝑝(𝑦) 𝑧2⇒𝐷𝑖𝑔𝑖𝑡(1,𝑧1)𝐷𝑖𝑔𝑖𝑡(1,𝑧2)].

2. ∀x,s[State(s)∧LivesIn(x,s)∧∃z1Zip(x)z1]⇒∀𝑥,𝑠[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥, 𝑠)∧∃𝑧1𝑍𝑖𝑝(𝑥)𝑧1 ]⇒\ [∀y,z2LivesIn(y,s)∧Zip(y)z2∧Digit(1,z1)Digit(1,z2)][∀𝑦,𝑧2𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖𝑝( 𝑦)𝑧2∧ 𝐷𝑖𝑔𝑖𝑡(1,𝑧1)𝐷𝑖𝑔𝑖𝑡(1,𝑧2)].

3.∀x,y,sState(s)∧LivesIn(x,s)∧LivesIn(y,s)⇒Digit(1,Zip(x)Zip(y))∀𝑥,𝑦,𝑠𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒 𝑠𝐼𝑛 (𝑥,𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)⇒𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑥)𝑍𝑖𝑝(𝑦)).

4. ∀x,y,sState(s)∧LivesIn(x,s)∧LivesIn(y,s)⇒∀𝑥,𝑦,𝑠𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛( 𝑥,𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)⇒ \ Digit(1,Zip(x))Digit(1,Zip(y))𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑥))𝐷𝑖𝑔𝑖𝑡(1,𝑍𝑖𝑝(𝑦)).

Giải

**1. Con mèo nào cũng yêu bố hoặc mẹ của mình.**

Biểu thức 1:

∀𝑥 Cat(𝑥)⇒Loves(𝑥,Mother(𝑥)∨Father(𝑥))∀xCat(x)⇒Loves(x,Mother(x)∨Father(x))

Sai ý nghĩa: Biểu thức này ngụ ý rằng mỗi con mèo yêu cả mẹ và bố của nó, không phản ánh ý nghĩa của câu.

Biểu thức 2:

∀𝑥 ¬Cat(𝑥)∨Loves(𝑥,Mother(𝑥))∨Loves(𝑥,Father(𝑥))∀x¬Cat(x)∨Loves(x,Mother(x))∨Loves(x,Father(x))

Sai ý nghĩa: Biểu thức này không diễn đạt đúng ý "mỗi con mèo".

Biểu thức 3:

∀𝑥 Cat(𝑥)∧(Loves(𝑥,Mother(𝑥))∨Loves(𝑥,Father(𝑥)))∀xCat(x)∧(Loves(x,Mother(x))∨Loves(x,Father(x)))

Sai ý nghĩa: Biểu thức này ngụ ý rằng mỗi con mèo đều yêu cả mẹ và bố của nó.

**2. Con chó nào yêu được anh em của mình thì hạnh phúc.**

Biểu thức 1:

∀𝑥 Dog(𝑥)∧(∃𝑦 Brother(𝑦,𝑥)∧Loves(𝑥,𝑦))⇒Happy(𝑥)∀xDog(x)∧(∃yBrother(y,x)∧Loves(x,y))⇒Happy(x)

Sai ý nghĩa: Biểu thức này không chính xác về ý nghĩa của câu vì nó không bao gồm điều kiện "who loves one of its brothers".

Biểu thức 2:

∀𝑥,𝑦 Dog(𝑥)∧Brother(𝑦,𝑥)∧Loves(𝑥,𝑦)⇒Happy(𝑥)∀x,yDog(x)∧Brother(y,x)∧Loves(x,y)⇒Happy(x)

Sai về cú pháp: Cú pháp của biểu thức này không chính xác vì nó thiếu các toán tử logic và mối quan hệ giữa các thành phần.

Biểu thức 3:

∀𝑥 Dog(𝑥)∧[∀𝑦 Brother(𝑦,𝑥)↔Loves(𝑥,𝑦)]⇒Happy(𝑥)∀xDog(x)∧[∀yBrother(y,x)↔Loves(x,y)]⇒Happy(x)

Sai ý nghĩa: Biểu thức này không diễn đạt đúng ý "who loves one of its brothers".

**3. Không có con chó nào cắn con của chủ nó.**

Biểu thức 1:

∀𝑥 Dog(𝑥)⇒¬Bites(𝑥,Child(Owner(𝑥)))∀xDog(x)⇒¬Bites(x,Child(Owner(x)))

Đúng ý nghĩa: Không có con chó nào cắn con của chủ nhân của nó.

Phân tích: Biểu thức này diễn đạt chính xác ý nghĩa của câu.

Biểu thức 2:

¬∃𝑥,𝑦 Dog(𝑥)∧Child(𝑦,Owner(𝑥))∧Bites(𝑥,𝑦)¬∃x,yDog(x)∧Child(y,Owner(x))∧Bites(x,y)

Sai về cú pháp: Biểu thức này có cú pháp không chính xác vì thiếu quan hệ giữa các thành phần.

Biểu thức 3:

∀𝑥 Dog(𝑥)⇒(∀𝑦 Child(𝑦,Owner(𝑥))⇒¬Bites(𝑥,𝑦))∀xDog(x)⇒(∀yChild(y,Owner(x))⇒¬Bites(x,y))

Sai ý nghĩa: Biểu thức này ngụ ý rằng mọi con chó không cắn bất kỳ ai, không chỉ là con của chủ nhân của chúng.

Biểu thức 4:

¬∃𝑥 Dog(𝑥)⇒(∃𝑦 Child(𝑦,Owner(𝑥))∧Bites(𝑥,𝑦))¬∃xDog(x)⇒(∃yChild(y,Owner(x))∧Bites(x,y))

Sai ý nghĩa: Biểu thức này ngụ ý rằng không có con chó nào tồn tại, do đó không thể so sánh các thông tin khác nhau về con chó.

**4. Mã zip của mọi người trong một tiểu bang đều có cùng chữ số đầu tiên.**

Biểu thức 1:

∀𝑥,𝑠,𝑧1[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧𝑍𝑖𝑝(𝑥)=𝑧1]⇒∀𝑦,𝑧2[𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖𝑝(𝑦)=𝑧2⇒𝐷𝑖𝑔𝑖𝑡(1,𝑧1)=𝐷𝑖𝑔𝑖𝑡(1,𝑧2)]∀x,s,z1[State(s)∧LivesIn(x,s)∧Zip(x)=z1]⇒∀y,z2[LivesIn(y,s)∧Zip(y)=z2⇒Digit(1,z1)=Digit(1,z2)]

Sai về cú pháp: Biểu thức này không chính xác về cú pháp vì nó sử dụng dấu "=" để so sánh, trong khi đó cần sử dụng toán tử logic phù hợp như "≡≡" hoặc "→→".

Biểu thức 2:

∀𝑥,𝑠[𝑆𝑡𝑎𝑡𝑒(𝑠)∧𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑥,𝑠)∧∃𝑧1𝑍𝑖𝑝(𝑥)=𝑧1]⇒∀𝑦,𝑧2[𝐿𝑖𝑣𝑒𝑠𝐼𝑛(𝑦,𝑠)∧𝑍𝑖𝑝(𝑦)=𝑧2∧𝐷𝑖𝑔𝑖𝑡(1,𝑧1)=𝐷𝑖𝑔𝑖𝑡(1,𝑧2)]∀x,s[State(s)∧LivesIn(x,s)∧∃z1Zip(x)=z1]⇒∀y,z2[LivesIn(y,s)∧Zip(y)=z2∧Digit(1,z1)=Digit(1,z2)]

Sai về ý nghĩa: Biểu thức này không diễn đạt đúng ý nghĩa của câu. Nó ngụ ý rằng mỗi người trong một tiểu bang có một mã zip có chứa chữ số đầu tiên giống nhau, không phản ánh rằng tất cả mọi người trong tiểu bang đều có mã zip chứa chữ số đầu tiên giống nhau.

Biểu thức 3:

∀𝑥,𝑦,𝑠[State(𝑠)∧LivesIn(𝑥,𝑠)∧LivesIn(𝑦,𝑠)]⇒Digit(1,Zip(𝑥))=Digit(1,Zip(𝑦))∀x,y,s[State(s)∧LivesIn(x,s)∧LivesIn(y,s)]⇒Digit(1,Zip(x))=Digit(1,Zip(y))

Sai ý nghĩa: Biểu thức này không diễn đạt đúng ý "của mọi người".

Biểu thức 4:

∀𝑥,𝑦,𝑠[State(𝑠)∧LivesIn(𝑥,𝑠)∧LivesIn(𝑦,𝑠)]⇒[Digit(1,Zip(𝑥))=Digit(1,Zip(𝑦))]∀x,y,s[State(s)∧LivesIn(x,s)∧LivesIn(y,s)]⇒[Digit(1,Zip(x))=Digit(1,Zip(y))]

Sai ý nghĩa: Biểu thức này không diễn đạt đúng ý "của mọi người".

Tóm lại, không có biểu thức nào trong số này đạt được cả hai yêu cầu về cú pháp và ý nghĩa. Đề bài cần một biểu thức mới có cú pháp chính xác và diễn đạt ý nghĩa "của mọi người".

[Exercise 27](https://aimacode.github.io/aima-exercises/fol-exercises/ex_27/)

For each of the following sentences in English, decide if the accompanying first-order logic sentence is a good translation. If not, explain why not and correct it. (Some sentences may have more than one error!)

**1. No two people have the same social security number.**

¬∃x,y,nPerson(x)∧Person(y)⇒[HasSS#(x,n)∧HasSS#(y,n)].¬∃𝑥,𝑦,𝑛𝑃𝑒𝑟𝑠𝑜𝑛(𝑥)∧𝑃𝑒𝑟𝑠𝑜𝑛(𝑦)⇒[𝐻𝑎𝑠𝑆𝑆#(𝑥,𝑛)∧𝐻𝑎𝑠𝑆𝑆#(𝑦,𝑛)].  
**2. John’s social security number is the same as Mary’s.**

∃nHasSS#(John,n)∧HasSS#(Mary,n).∃𝑛𝐻𝑎𝑠𝑆𝑆#(𝐽𝑜ℎ𝑛,𝑛)∧𝐻𝑎𝑠𝑆𝑆#(𝑀𝑎𝑟𝑦,𝑛).  
**3. Everyone’s social security number has nine digits.**

∀x,nPerson(x)⇒[HasSS#(x,n)∧Digits(n,9)].∀𝑥,𝑛𝑃𝑒𝑟𝑠𝑜𝑛(𝑥)⇒[𝐻𝑎𝑠𝑆𝑆#(𝑥,𝑛)∧𝐷𝑖𝑔𝑖𝑡𝑠(𝑛,9)].

**4. Rewrite each of the above (uncorrected) sentences using a function symbol SS#𝑆𝑆# instead of the predicate HasSS#𝐻𝑎𝑠𝑆𝑆#.**

Dịch

### Bài tập 27

Đối với mỗi câu bằng tiếng Anh sau đây, hãy quyết định xem câu logic bậc nhất kèm theo có phải là bản dịch tốt hay không. Nếu không, hãy giải thích tại sao không và sửa lại. (Một số câu có thể có nhiều lỗi!)

**1. Không có hai người nào có số an sinh xã hội giống nhau.**

и∃x,y,nPerson(x)∧Person(y)⇒[HasSS#(x,n )∧HasSS#(y,n)].€∃𝑥,𝑦,𝑛𝑃𝑒𝑟𝑠𝑜𝑛(𝑥)∧𝑃𝑒𝑟𝑠𝑜𝑑 (𝑦)⇒ [𝐻𝑎𝑠𝑆𝑆#(𝑥,𝑛)∧𝐻𝑎𝑠𝑆𝑆#(𝑦,𝑛)].

**2. Số an sinh xã hội của John giống với số an sinh xã hội của Mary.**

∃nHasSS#(John,n)∧HasSS#(Mary,n).∃𝑛𝐻𝑎𝑠𝑆𝑆#(𝐽𝑜ℎ𝑛,𝑛)∧𝐻𝑎𝑠𝑆𝑆#(𝑀𝑎𝑟𝑦, 𝑛).

**3. Số an sinh xã hội của mọi người đều có chín chữ số.**

∀x,nPerson(x)⇒[HasSS#(x,n)∧Digits(n,9)].∀𝑥,𝑛𝑃𝑒𝑟𝑠𝑜𝑛(𝑥)⇒[𝐻𝑎𝑠𝑆𝑆#(𝑥,𝑛)∧𝐷 𝑖𝑔𝑖𝑡𝑠(𝑛,9)].

**4. Viết lại từng câu trên (không sửa) bằng ký hiệu hàm SS#𝑆𝑆# thay cho vị từ HasSS#𝐻𝑎𝑠𝑆𝑆#.**

Giải

**1. Không có hai người nào có cùng một mã an sinh xã hội.**

**Bản dịch logic:** ¬∃x,y,nPerson(x)∧Person(y)⇒[HasSS#(x,n)∧HasSS#(y,n)].

**Lý do sai:**

Bản dịch logic hiện tại chỉ ghi nhận việc có cùng một số mà không loại trừ trường hợp có các số khác nhau.

Câu gốc khẳng định rằng không có hai người nào có cùng mã an sinh xã hội, nghĩa là mọi cặp người khác nhau đều không thể có cùng mã số.

Bản dịch logic hiện tại sử dụng phủ định logic (¬) cho toàn bộ mệnh đề, dẫn đến việc nó chỉ phủ định sự tồn tại của các trường hợp thỏa mãn cả hai điều kiện: là người (Person(x) ∧ Person(y)) và có cùng mã số (HasSS#(x,n) ∧ HasSS#(y,n)).

**Sửa chữa:**

¬∃x, y, n (Person(x) ∧ Person(y) ∧ x ≠ y) ⇒ (SS(x) = SS(y))

Sử dụng phủ định logic (¬) cho phần điều kiện (Person(x) ∧ Person(y) ∧ x ≠ y) để đảm bảo chỉ phủ định trường hợp hai người khác nhau có cùng mã số.

Sử dụng ký hiệu so sánh bằng (=) để thể hiện chính xác ý nghĩa "cùng mã số".

**2. Mã an sinh xã hội của John giống với mã an sinh xã hội của Mary.**

**Bản dịch logic:** ∃nHasSS#(John,n)∧HasSS#(Mary,n).

**Lý do đúng:**

Bản dịch logic hiện tại chính xác thể hiện việc John và Mary có cùng một mã an sinh xã hội (được xác định bởi n).

Sử dụng ký hiệu tồn tại logic (∃) để thể hiện rằng có ít nhất một giá trị n thỏa mãn cả hai điều kiện: John có mã số n (HasSS#(John,n)) và Mary có mã số n (HasSS#(Mary,n)).

**3. Mã an sinh xã hội của mọi người đều có chín chữ số.**

**Bản dịch logic:** ∀x,nPerson(x)⇒[HasSS#(x,n)∧Digits(n,9)].

**Lý do sai:**

Bản dịch logic hiện tại chỉ gán mã an sinh xã hội (n) cho một người (x) nhưng không quy định số chữ số.

Câu gốc khẳng định rằng mọi người đều có mã an sinh xã hội chín chữ số.

Bản dịch logic hiện tại sử dụng suy luận toàn thể (∀) cho phần điều kiện (Person(x)), nghĩa là nó áp dụng quy tắc chín chữ số cho mọi người, ngay cả khi họ không có mã an sinh xã hội.

**Sửa chữa:**

∀x Person(x) ⇒ Digits(SS(x), 9)

Thay thế HasSS#(x,n) bằng SS (x) để sử dụng ký hiệu hàm SS cho mã an sinh xã hội.

Giữ nguyên sử dụng suy luận toàn thể (∀) cho phần điều kiện (Person(x)) để áp dụng quy tắc chín chữ số cho mọi người.

Thêm điều kiện Digits(SS (x), 9) để đảm bảo mã an sinh xã hội SS (x) có chính xác chín chữ số.

**4. Viết lại bằng ký hiệu hàm SS#:**

Câu 1: ¬∃x, y, n (Person(x) ∧ Person(y) ∧ x ≠ y) ⇒ (SS(x) = SS(y))

Câu 2: ∃n (SS(John) = n ∧ SS(Mary) = n)

Câu 3: ∀x Person(x) ⇒ Digits(SS(x), 9)

[Exercise 29](https://aimacode.github.io/aima-exercises/fol-exercises/ex_29/)

For each of the following sentences in English, decide if the accompanying first-order logic sentence is a good translation. If not, explain why not and correct it.  
**1. Any apartment in London has lower rent than some apartments in Paris.**

∀x[Apt(x)∧In(x,London)]⟹∃y([Apt(y)∧In(y,Paris)]⟹(Rent(x)<Rent(y)))∀𝑥[𝐴𝑝𝑡(𝑥)∧𝐼𝑛(𝑥,𝐿𝑜𝑛𝑑𝑜𝑛)]⟹∃𝑦([𝐴𝑝𝑡(𝑦)∧𝐼𝑛(𝑦,𝑃𝑎𝑟𝑖𝑠)]⟹(𝑅𝑒𝑛𝑡(𝑥)<𝑅𝑒𝑛𝑡(𝑦)))

**2. There is exactly one apartment in Paris with rent below $1000.**

∃xApt(x)∧In(x,Paris)∧∀y[Apt(y)∧In(y,Paris)∧(Rent(y)<Dollars(1000))]⟹(y=x)∃𝑥𝐴𝑝𝑡(𝑥)∧𝐼𝑛(𝑥,𝑃𝑎𝑟𝑖𝑠)∧∀𝑦[𝐴𝑝𝑡(𝑦)∧𝐼𝑛(𝑦,𝑃𝑎𝑟𝑖𝑠)∧(𝑅𝑒𝑛𝑡(𝑦)<𝐷𝑜𝑙𝑙𝑎𝑟𝑠(1000))]⟹(𝑦=𝑥)

**3. If an apartment is more expensive than all apartments in London, it must be in Moscow.**

∀xApt(x)∧[∀yApt(y)∧In(y,London)∧(Rent(x)>Rent(y))]⟹In(x,Moscow).

Dịch

### Bài tập 29

Đối với mỗi câu bằng tiếng Anh sau đây, hãy quyết định xem câu logic bậc nhất kèm theo có phải là bản dịch tốt hay không. Nếu không, hãy giải thích tại sao không và sửa lại.

**1. Bất kỳ căn hộ nào ở London đều có giá thuê thấp hơn một số căn hộ ở Paris.**

∀x[Apt(x)∧In(x,London)]⟹∃y([Apt(y)∧In(y,Paris)]⟹(Rent(x)<Rent(y)))∀𝑥[𝐴𝑝𝑡( 𝑥)∧𝐼𝑛(𝑥,𝐿𝑜𝑛𝑑𝑜𝑛)]⟹∃𝑦([𝐴𝑝𝑡(𝑦)∧𝐼𝑛(𝑦,𝑃𝑎𝑟𝑖𝑠)]⟹( 𝑅𝑒𝑛𝑡(𝑥)<𝑅𝑒𝑛𝑡(𝑦)))

**2. Có đúng một căn hộ ở Paris có giá thuê dưới 1000 USD.**

∃xApt(x)∧In(x,Paris)∧∀y[Apt(y)∧In(y,Paris)∧(Rent(y)<Dollars(1000))]⟹(y=x)∃𝑥𝐴𝑝𝑡(𝑥)∧𝐼𝑛(𝑥,𝑃𝑎𝑟𝑖𝑠)∧∀𝑦[𝐴𝑝𝑡(𝑦)∧𝐼𝑛(𝑦,𝑃𝑎𝑟𝑖𝑠)∧(𝑅𝑒 𝑛𝑡(𝑦)<𝐷𝑜𝑙𝑙𝑎𝑟𝑠(1000))]⟹(𝑦=𝑥)

**3. Nếu một căn hộ đắt hơn tất cả các căn hộ ở London thì nó phải ở Moscow.**

∀xApt(x)∧[∀yApt(y)∧In(y,London)∧(Rent(x)>Rent(y))]⟹In(x,Moscow).

Giải

**1 sai:**

Câu gốc nói rằng bất kỳ căn hộ nào ở London đều rẻ hơn một số căn hộ ở Paris.

Bản dịch được cung cấp sử dụng định lượng phổ quát (∀) cho cả "bất kỳ" và "một số" không chính xác.

**Điều chỉnh:**

∀x[Apt(x) ∧ In(x, London)] ⟹ ∃y[Apt(y) ∧ In(y, Paris) ∧ Rent(x) < Rent(y)]

**2 Đúng:**

Câu gốc và bản dịch được cung cấp thể hiện chính xác rằng tồn tại chính xác một căn hộ ở Paris với giá thuê dưới 1000 USD (được xác định bằng x).

**3 Không đúng:**

Câu gốc nói rằng nếu một căn hộ đắt hơn tất cả các căn hộ ở London thì nó phải ở Moscow.

Bản dịch được cung cấp sử dụng định lượng phổ quát (∀) cho "tất cả", điều này đúng nhưng nó chỉ kiểm tra giá thuê so với các căn hộ hiện có chứ không phải tất cả các căn hộ có thể có.Điều chỉnh:

Có hai cách để tiếp cận sự điều chỉnh này, tùy thuộc vào ý nghĩa dự kiến:

Trường hợp 1 (Mạnh hơn): Đảm bảo căn hộ đắt hơn bất kỳ căn hộ tiềm năng nào ở London, kể cả những căn hộ chưa tồn tại. Điều này đòi hỏi phải giới thiệu các vị từ về giá thuê tối đa ở Luân Đôn (MaxRentLondon) hoặc tất cả các căn hộ có thể có (AllApartments).

∀x Apt(x) ∧ [∀y (PossibleApartment(y) ∧ In(y, London)) ⟹ Rent(x) > Rent(y)] ⟹ In(x, Moscow)

Trường hợp (Yếu hơn): Đảm bảo căn hộ đắt hơn tất cả các căn hộ hiện có ở London. Điều này giữ cấu trúc ban đầu.

∀x Apt(x) ∧ [∀y Apt(y) ∧ In(y, London) ⟹ Rent(x) > Rent(y)] ⟹ In(x, Moscow)

Exercise 30

Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define):

1. Some students took French in spring 2001.

2. Every student who takes French passes it.

3. Only one student took Greek in spring 2001.

4. The best score in Greek is always higher than the best score in French.

5. Every person who buys a policy is smart.

6. No person buys an expensive policy.

7. There is an agent who sells policies only to people who are not insured.

8. There is a barber who shaves all men in town who do not shave themselves.

9. A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.

10. A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

11. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can’t fool all of the people all of the time.

12. All Greeks speak the same language. (Use Speaks(x,l)𝑆𝑝𝑒𝑎𝑘𝑠(𝑥,𝑙) to mean that person x𝑥 speaks language l𝑙.)

Dịch

### Bài tập 30

Trình bày các câu sau theo logic bậc nhất, sử dụng từ vựng nhất quán (bạn phải xác định):

1. Một số học sinh học tiếng Pháp vào mùa xuân năm 2001.

2. Mọi học sinh học tiếng Pháp đều đậu.

3. Chỉ có một học sinh học tiếng Hy Lạp vào mùa xuân năm 2001.

4. Điểm tốt nhất ở tiếng Hy Lạp luôn cao hơn điểm cao nhất ở tiếng Pháp.

5. Người nào mua hợp đồng đều thông minh.

6. Không ai mua hợp đồng bảo hiểm đắt tiền.

7. Có một đại lý chỉ bán hợp đồng bảo hiểm cho những người không có bảo hiểm.

8. Có một người thợ cắt tóc chuyên cạo râu cho tất cả đàn ông trong thị trấn không tự cạo râu.

9. Một người sinh ra ở Vương quốc Anh, mỗi người có cha mẹ là công dân Vương quốc Anh hoặc cư dân Vương quốc Anh, thì khi sinh ra đã là công dân Vương quốc Anh.

10. Một người sinh ra bên ngoài Vương quốc Anh, có cha mẹ khi sinh ra là công dân Vương quốc Anh, thì có nguồn gốc là công dân Vương quốc Anh.

11. Các chính trị gia có thể lừa gạt một số người mọi lúc, và họ có thể lừa gạt tất cả mọi người trong một lúc nào đó, nhưng họ không thể lừa dối tất cả mọi người mọi lúc.

12.Tất cả người Hy Lạp đều nói cùng một ngôn ngữ. (Sử dụng Speaks(x,l)𝑆𝑝𝑒𝑎𝑘𝑠(𝑥,𝑙) để chỉ người đó x𝑥 nói ngôn ngữ l𝑙.)

Giải

**Từ vựng:**

Person(x): x is a person (x là một người)

Student(x): x is a student (x là một sinh viên

Took(x, course, semester): x took course in semester (x đã học trong học kỳ)

Passed(x, course): x passed course (x đã vượt qua khóa học)

Greek, French: Names of languages (can be replaced with constants) Tên các ngôn ngữ (có thể thay thế bằng hằng số)

Spring2001: Spring semester of 2001 (constant) Học kỳ mùa xuân năm 2001 (liên tục)

BestScore(course, semester): best score achieved in course during semester (điểm số cao nhất đạt được trong khóa học trong học kỳ)

Sells(agent, policy): agent sells policy (chính sách bán đại lý)

Buys(person, policy): person buys policy (người mua chính sách)

Expensive(policy): policy is expensive (chính sách đắt tiền)

Insured(person): person is insured (người được bảo hiểm)

Agent(x): x is an agent (x là một đại lý)

Barber(x): x is a barber (x là thợ cắt tóc)

Shaves(x, y): x shaves y (y being a person) (x cạo y (y là một người))

Town(x): x is a town (x là một thị trấn)

UKCitizen(x): x is a UK citizen (by birth) (x là công dân Vương quốc Anh (khi sinh ra))

UKResident(x): x is a UK resident (x là cư dân Vương quốc Anh)

Born(x, location): x was born in location (x được sinh ra ở vị trí)

Politician(x): x is a politician (x là một chính trị gia)

CanFool(x, y): x can fool y (y being a person) (x có thể đánh lừa y (y là một người))

AllPeople: refers to the set of all people (đề cập đến tập hợp của tất cả mọi người)

**Các câu trong logic bậc nhất:**

1. ∃x Student(x) ∧ Took(x, French, Spring2001) (There exists a student who took French in Spring 2001) (Có một sinh viên học tiếng Pháp vào mùa xuân năm 2001)

2. ∀x Student(x) ∧ Took(x, French) ⟹ Passed(x, French) (Every student who takes French passes it) (Mỗi học sinh học tiếng Pháp đều đậu)

3. ∃x! Took(x, Greek, Spring2001) (There exists exactly one student who took Greek in Spring 2001) (❗ symbol denotes unique existence) (Tồn tại chính xác một học sinh đã học tiếng Hy Lạp vào mùa xuân năm 2001) (❗ ký hiệu biểu thị sự tồn tại duy nhất)

4. ∀semester, course (BestScore(Greek, semester) > BestScore(French, semester)) ∨ (course ≠ Greek ∧ course ≠ French) (For any semester and course, either the best score in Greek is higher than the best score in French in that semester, or the course is not Greek or French) (Đối với bất kỳ học kỳ và khóa học nào, điểm tốt nhất môn tiếng Hy Lạp phải cao hơn điểm cao nhất môn tiếng Pháp trong học kỳ đó hoặc khóa học không phải là tiếng Hy Lạp hoặc tiếng Pháp)

5. ∀x Buys(x, policy) ⟹ Smart(x) (Every person who buys a policy is smart) (Người mua bảo hiểm nào cũng thông minh)

6. ¬∃x Buys(x, policy) ∧ Expensive(policy) (There does not exist a person who buys an expensive policy) (This can also be written as: ∀x Buys(x, policy) ⟹ ¬Expensive(policy))

7. ∃x Agent(x) ∧ ∀person (Sells(x, policy) ⟹ ¬Insured(person)) (There exists an agent who sells policies only to people who are not insured) (Có đại lý chỉ bán hợp đồng cho người không có bảo hiểm)

8. ∃x Barber(x) ∧ ∀man Town(man) ⟹ (Shaves(x, man) ↔ ¬Shaves(man, man)) (There exists a barber who shaves all men in town who do not shave themselves) (Có một thợ cắt tóc cạo râu cho tất cả đàn ông trong thị trấn không tự cạo râu)

9. ∀xPerson(x) ∧ Born(x, UK) ∧ (UKCitizen(parent1(x)) ∨UKResident(parent1(x))) ∧ (UKCitizen(parent2(x)) ∨ UKResident(parent2(x))) ⟹ UKCitizen(x) (Any person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth) (parent1(x) and parent2(x) represent the first and second parent)

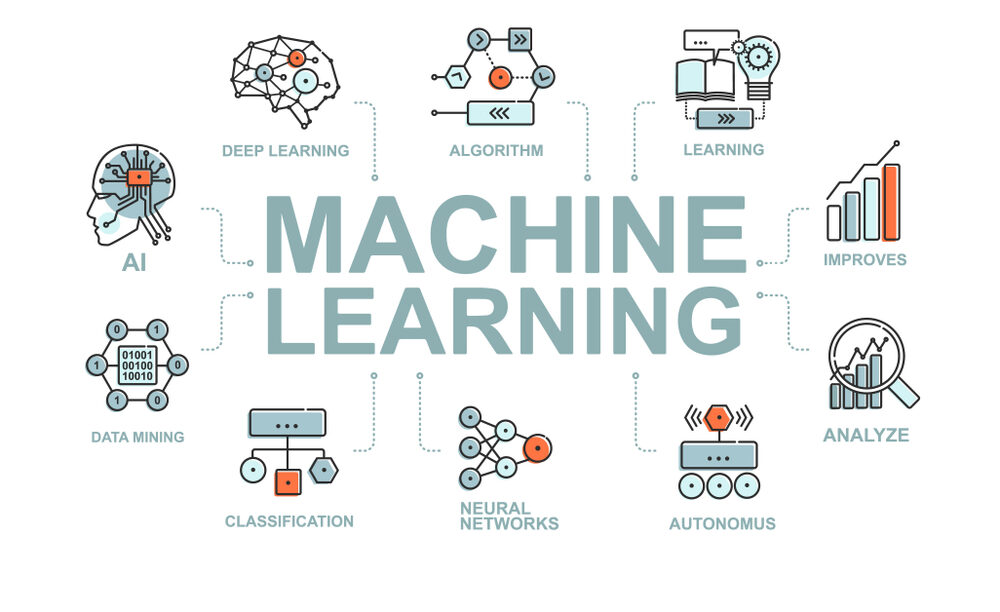
10. ∀x Person(x) ∧ ¬Born(x, UK) ∧ ∃parent(y, x) ∧ UKCitizen(y) ⟹ UKCitizen(x) (A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent) (Một người sinh ra bên ngoài Vương quốc Anh, một trong hai người có cha mẹ là công dân Vương quốc Anh khi sinh ra, là công dân Vương quốc Anh theo dòng dõi)

11. ∃politicians AllPeople ∀time CanFool(politicians, somePeople(time)) ∧ ∃politicians ∃time CanFool(politicians, AllPeople) ∧ ¬∃politicians ∀time CanFool(politicians, AllPeople) (There are politicians who can fool some people at all times, and some politicians who can fool all the people at some time, but no politician can fool all the people all the time) (somePeople(time) refers to a function that returns a subset of people for a given time) (Có những chính trị gia có thể đánh lừa một số người mọi lúc, và một số chính trị gia có thể đánh lừa tất cả mọi người vào một lúc nào đó, nhưng không một chính trị gia nào có thể đánh lừa tất cả mọi người mọi lúc) (somePeople(time) đề cập đến một hàm trả về một tập hợp con người trong một thời gian nhất định)

12. ∀greek Greek(x) ⟹ Speaks(x, sameLanguage) (All Greeks speak the same language) (sameLanguage is a constant representing the language all Greeks speak) (Tất cả người Hy Lạp đều nói cùng một ngôn ngữ) (sameLanguage là hằng số đại diện cho ngôn ngữ mà tất cả người Hy Lạp nói)

# PHẦN III: HỌC MÁY

Học máy hay máy học (tiếng Anh: machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Các thuật toán học máy xây dựng một mô hình dựa trên dữ liệu mẫu, được gọi là dữ liệu huấn luyện, để đưa ra dự đoán hoặc quyết định mà không cần được lập trình chi tiết về việc đưa ra dự đoán hoặc quyết định này.

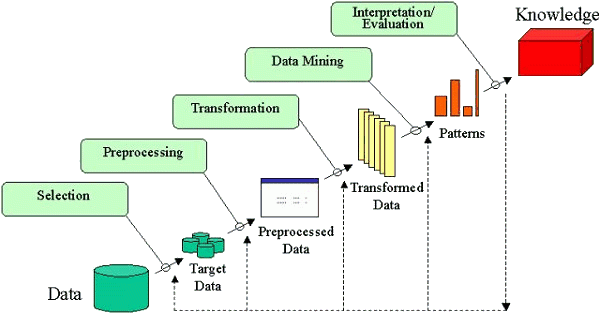


Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào thư mục tương ứng. Học máy rất gần với suy diễn thống kê (statistical inference) tuy có khác nhau về thuật ngữ. Một nhánh của học máy là học sâu phát triển rất mạnh mẽ gần đây và có những kết quả vượt trội so với các phương pháp học máy khác Học máy có liên quan lớn đến thống kê, vì cả hai lĩnh vực đều nghiên cứu việc phân tích dữ liệu, nhưng khác với thống kê, học máy tập trung vào sự phức tạp của các giải thuật trong việc thực thi tính toán. Nhiều bài toán suy luận được xếp vào loại bài toán NP-khó, vì thế một phần của học máy là nghiên cứu sự phát triển các giải thuật suy luận xấp xỉ mà có thể xử lý được.

Học máy có hiện nay được áp dụng rộng rãi bao gồm máy truy tìm dữ liệu, chẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại các chuỗi DNA, nhận dạng tiếng nói và chữ viết, dịch tự động, chơi trò chơi và cử động rô-bốt (robot locomotion).

## Định nghĩa

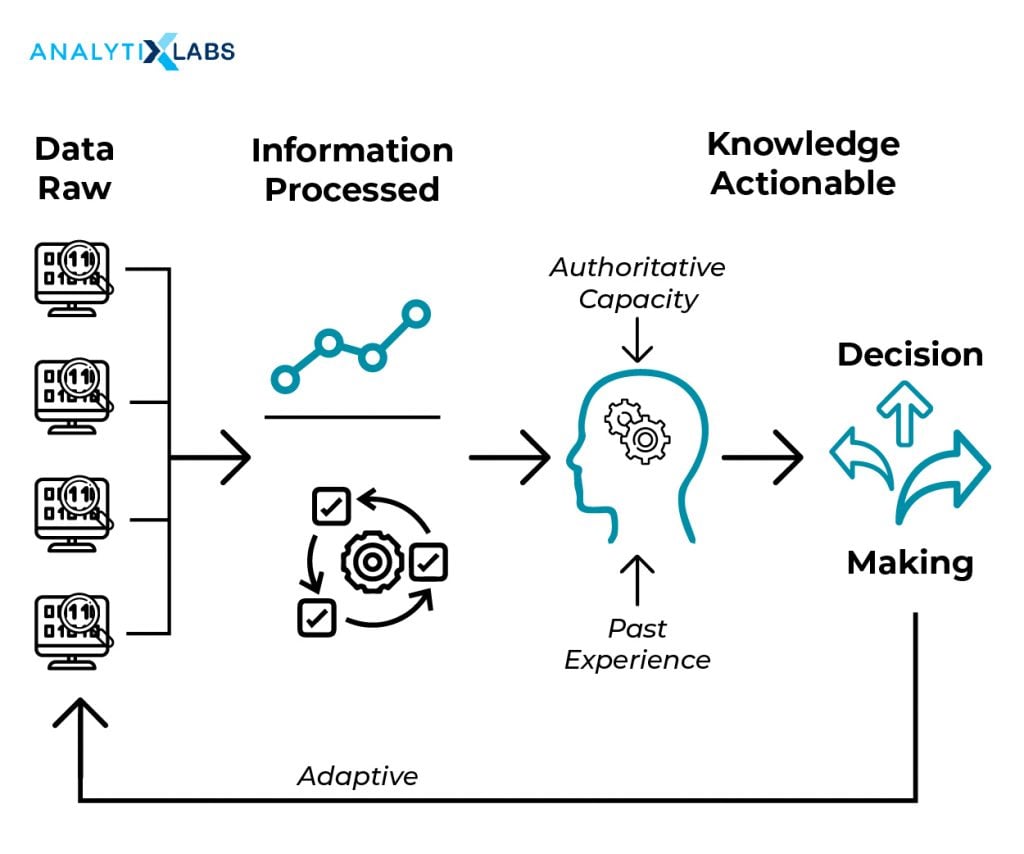
Dưới góc nhìn của trí tuệ nhân tạo, động lực chính học máy bởi là nhu cầu thu nhận tri thức (knowledge acquisition). Thật vậy, trong nhiều trường hợp ta cần kiến thức chuyên gia là khan hiếm (không đủ chuyên gia ngồi phân loại lừa đảo thẻ tín dụng của tất cả giao dịch hàng ngày) hoặc chậm vì một số nhiệm vụ cần đưa ra quyết định nhanh chóng dựa trên xử lý dữ liệu khổng lồ (trong mua bán chứng khoán phải quyết định trong vài khoảng khắc của giây chẳng hạn) và thiếu ổn định thì buộc phải cần đến máy tính. Ngoài ra, đại đa số dữ liệu sinh ra ngày nay chỉ phù hợp cho máy đọc (computer readable) tiềm tàng ngưồn kiến thức quan trọng. Máy học nghiên cứu cách thức để mô hình hóa bài toán cho phép máy tính tự động hiểu, xử lý và học từ dữ liệu để thực thi nhiệm vụ được giao cũng như cách đánh giá giúp tăng tính hiệu quả.



Tom Mitchell, giáo sư nổi tiếng của Đại học Carnegie Mellon University - CMU định nghĩa cụ thể và chuẩn mực hơn như sau: "Một chương trình máy tính CT được xem là học cách thực thi một lớp nhiệm vụ NV thông qua trải nghiệm KN, đối với thang đo năng lực NL nếu như dùng NL ta đo thấy năng lực thực thi của chương trình có tiến bộ sau khi trải qua KN" (máy đã học).[1]

## Biểu diễn

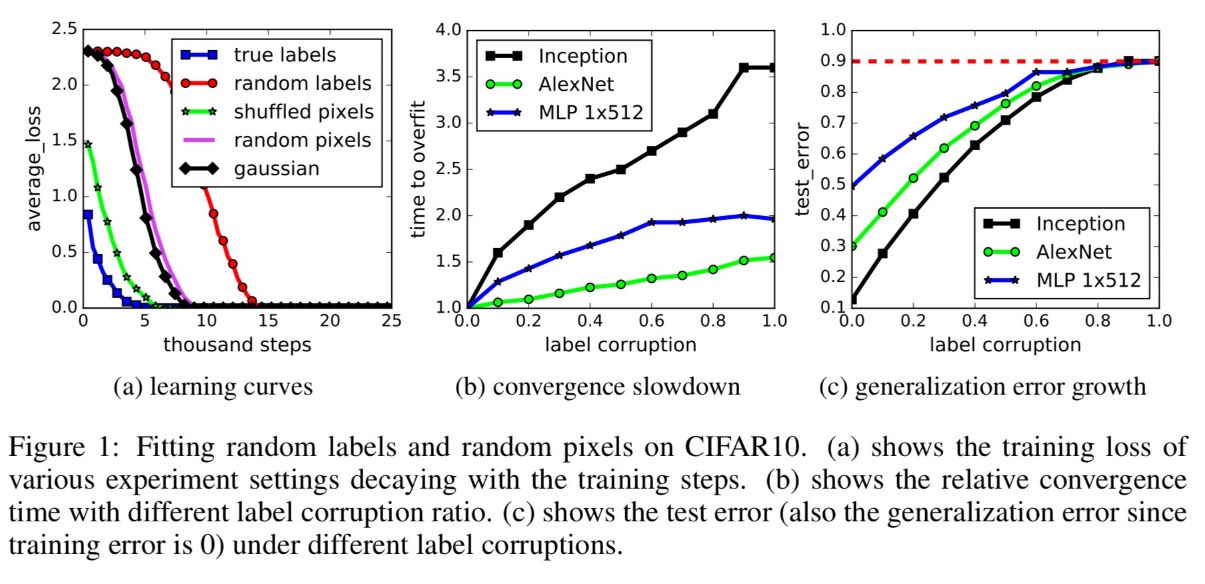
Biểu diễn (tiếng Anh: representation) là một trong những vấn đề quan trọng của học máy. Biểu diễn ở đây có thể hiểu làm sao mã hóa (encode) những thông tin của thế giới thật giúp hoàn thành nhiệm vụ một cách hiệu quả và đầy đủ nhất có thể. Thông tin ở đây bao hàm cả thông tin về dữ liệu đầu vào, đầu ra hay các trạng thái của hệ thống; cũng như cách đánh giá hiệu quả của chương trình.



Thông thường, trong học máy người ta hay xây dựng các mô hình sử dụng những biến ngẫu nhiên cho việc biểu diễn dữ liệu và nội trạng thái của hệ thống. Ví dụ: dùng biến ngẫu nhiên để biểu thị cho tính chất của email là spam (tương ứng giá trị 0) hay là bình thường (tương ứng 1). Mối tương quan giữa các biến ngẫu nhiên này có thể sử dụng ví dụ như mô hình xác suất dạng đồ thị để miêu tả. Mặt khác, để đo hiệu quả có thể dùng các hàm thiệt hại (hay hàm tiện ích, trong tiếng Anh là loss function và utility function tương ứng).

## Tính phổ quát

Một trong những trọng tâm khác của học máy là đạt được tính phổ quát (tiếng Anh: generalization), nói cách khác là tính chất của chương trình có thể làm việc tốt với dữ liệu mà nó chưa gặp bao giờ (tiếng Anh: unseen data). Một chương trình chỉ hiệu quả với dữ liệu đã gặp nhìn chung không có nhiều tính hữu dụng.



Lấy ví dụ về xếp thư điện tử tự động như trên, một hệ thống tự động sau khi trải qua quá trình học từ dữ liệu ("training") có thể suy diễn một số nguyên tắc riêng (chẳng hạn như xem xét nội dung: nếu thư được viết bằng tiếng Anh mà chứa một số từ như "porn", "sell", "good product" hoặc người gửi đến từ Somalia trong khi người nhận ở Hà Nội không thân quen nhau) để quyết định xem có phải là thư rác hay không. Tuy nhiên, nếu như trong dữ liệu bài giảng (training data) có ngôn ngữ khác trong thực tế (tiếng Việt thay vì tiếng Anh) hoặc thậm chí không phải dạng thuần văn bản (dạng ảnh khiến cho bóc tách nội dung khó hơn hoặc không thể) thì rất có thể máy sẽ dự báo không chính xác nữa.

Một số chương trình có thể tự động cập nhật trong thời gian thực (ví dụ như người sử dụng có chỉ ra rằng thư bị sắp xếp sai danh mục).

## Tương tác với con người

Một số hệ thống học máy nỗ lực loại bỏ nhu cầu trực giác của con người trong việc phân tích dữ liệu, trong khi các hệ thống khác hướng đến việc tăng sự cộng tác giữa người và máy. Không thể loại bỏ hoàn toàn tác động của con người vì các nhà



## Tương quan với Khai phá dữ liệu

Khai phá dữ liệu và học máy là hai khái niệm hay bị nhầm lẫn. Hai lĩnh vực này nhìn chung gần với nhau và đôi khi dùng chung nhiều phương pháp, công cụ nhưng khác biệt chính là ở mục tiêu:



Khai phá dữ liệu: thường mục tiêu là tìm kiếm những thông tin, tri thức hoàn toàn mới tiềm năng có ích trong nguồn dữ liệu.

Học máy: dự đoán một số thông tin của dữ liệu dựa trên những đặc tính đã biết.

## Các loại giải thuật

Các thuật toán học máy được phân loại theo kết quả mong muốn của thuật toán. Các loại thuật toán thường dùng bao gồm:



Học có giám sát—trong đó, thuật toán tạo ra một hàm ánh xạ dữ liệu vào tới kết quả mong muốn. Một phát biểu chuẩn về một việc học có giám sát là bài toán phân loại: chương trình cần học (cách xấp xỉ biểu hiện của) một hàm ánh xạ một vector

[𝑋1,𝑋2,…𝑋𝑁] tới một vài lớp bằng cách xem xét một số mẫu dữ liệu - kết quả của hàm đó.

Học không giám sát—mô hình hóa một tập dữ liệu, không có sẵn các ví dụ đã được gắn nhãn.

Học nửa giám sát—kết hợp các ví dụ có gắn nhãn và không gắn nhãn để sinh một hàm hoặc một bộ phân loại thích hợp.

Học tăng cường—trong đó, thuật toán học một chính sách hành động tùy theo các quan sát về thế giới. Mỗi hành động đều có tác động tới môi trường, và môi trường cung cấp thông tin phản hồi để hướng dẫn cho thuật toán của quá trình học.

Chuyển đổi—tương tự học có giám sát nhưng không xây dựng hàm một cách rõ ràng. Thay vì thế, cố gắng đoán kết quả mới dựa vào các dữ liệu huấn luyện, kết quả huấn luyện, và dữ liệu thử nghiệm có sẵn trong quá trình huấn luyện.

Học cách học—trong đó thuật toán học thiên kiến quy nạp của chính mình, dựa theo các kinh nghiệm đã gặp.

Phân tích hiệu quả các thuật toán học máy là một nhánh của ngành thống kê, được biết với tên lý thuyết học điện toán.

## Các chủ đề về máy học

Mô hình hóa các hàm mật độ xác suất điều kiện: hồi quy và phân loại

Mạng nơ-ron

Máy học cực độ (Extreme learning machine)

Cây quyết định

Lập trình biểu thức gen

Lập trình di truyền

Hồi quy quá trình Gauss

Phân tích biệt thức tuyến tính

k láng giềng gần nhất

Độ dài thông điệp tối thiểu

Cảm tri nguyên

Hàm cơ sở xuyên tâm

Máy vector hỗ trợ (Support Vector Machine)

Mô hình hóa các hàm mật độ xác suất qua các mô hình phát sinh:

Thuật toán cực đại kì vọng

Các mô hình đồ họa gồm mạng Bayes và mạng Markov

Ánh xạ topo phát sinh

Các kỹ thuật suy luận xấp xỉ đúng:

Chuỗi Markov phương pháp Monte Carlo

Phương pháp biến thiên

Tối ưu hóa: hầu hết các phương pháp trên đều sử dụng tối ưu hóa hoặc là các thể hiện của các thuật toán tối ưu hóa.

## Câu 3: Học máy (đề 2)

1. ***Trình bày thuật toán Cây quyết định (C4.5)***

Xây dựng cây quyết định gồm 2 bước:

* Phát triển cây quyết định: đi từ gốc, đến các nhánh, phát triển quy nạp theo hình thức chia để trị.
* Chọn thuộc tính “tốt” nhất bằng một độ đo đã định trước
* Phát triển cây bằng việc thêm các nhánh tương ứng với từng giá trị của thuộc tính đã chọn
* Sắp xếp, phân chia tập dữ liệu đào tạo tới node con
* Nếu các ví dụ được phân lớp rõ ràng thì dừng.
* Ngược lại: lặp lại bước 1 tới bước 4 cho từng node con
* Cắt tỉa cây: nhằm đơn giản hóa, khái quát hóa cây, tăng độ chính xác

Điểm mạnh của cây quyết định:

* Sinh ra các quy tắc hiểu được: chuyển đổi được sang tiếng Anh hoặc SQL.
* Thực thi trong lĩnh vực hướng quy tắc.
* Dễ dàng tính toán trong khi phân lớp.
* Xử lý với thuộc tính liên tục và rời rạc.
* Thể hiện rõ ràng những thuộc tính tốt nhất: phân chia dữ liệu từ gốc.

Điểm yếu của cây quyết định:

* Dễ xảy ra lỗi khi có nhiều lớp: do chỉ thao tác với các lớp có giá trị dạng nhị phân.
* Chi phí tính toán đắt để học: do phải đi qua nhiều node để đến node lá cuối cùng

Thuật toán C4.5

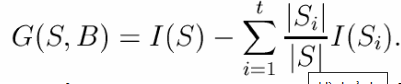
* Là sự phát triển từ CLS và ID3.
* ID3 (Quinlan, 1979)- 1 hệ thống đơn giản ban đầu chứa khoảng 600 dòng lệnh Pascal
* Năm 1993, J. Ross Quinlan phát triển thành C4.5 với 9000 dòng lệnh C.
* Hiện tại: phiên bản See5/C5.0.
* Tư tưởng thuật toán: Hunt, chiến lược phát triển theo độ sâu.

Pseudocode:

* Kiểm tra case cơ bản
* Với mỗi thuộc tính A tìm thông tin nhờ việc tách thuộc tính A
* Chọn a\_best là thuộc tính mà độ đo lựa chọn thuộc tính “tốt nhất”
* Dùng a\_best làm thuộc tính cho node chia cắt cây.
* Đệ quy trên các danh sách phụ được tạo ra bởi việc phân chia theo a\_best, và thêm các node này như là con của node

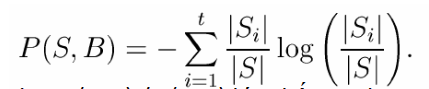
Độ đo lựa chọn thuộc tính “tốt nhất”:

* Information gain: Test B chia S={S1,S2,…,St)



Test B sẽ được chọn nếu có G(S, B) đạt giá trị lớn nhất.

* Thông tin tiềm năng (potential information) của bản thân mỗi phân hoạch:



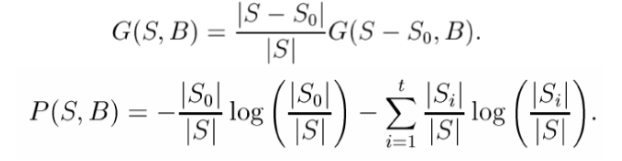
* ​Gain ratio = G(S, B) / P(S, B) lớnnhất => chọn test B

Xử lý “quá vừa” dữ liệu:

* Cho phép cây quá vừa dữ liệu, sau đó cắt tỉa cây.

Xử lý những giá trị thiếu:

* Là case có thuộc tính không có giá trị.
* information gain và potential information: S0 = case thuộc S có giá trị thuộc tính = null.



* Nếu test B được chọn, C4.5 phân chia các case trong S0 về các tập con Si

Đặc điểm C4.5:

* Chiếm thời gian sử dụng CPU và bộ nhớ lớn:
* vd với 10k tới 100k case, tạo cây quyết định tăng từ 1,4s lên 61s, tạo luật tăng từ 32s lên 9,715s.
* Sử dụng cơ chế lưu dữ liệu thường trú trong bộ nhớ => ứng dụng với database nhỏ ( tần số lỗi lặp lại 4% với database 20000 cases).
* Có cơ chế xử lý thiếu, lỗi hoặc quá vừa dữ liệu.
* Luật tạo ra đơn giản.

1. ***Thực hiện phân lớp dữ liệu adult bằng thuật toán C4.5***

# KẾT LUẬN

Trong lĩnh vực Trí Tuệ Nhân Tạo (AI), việc giải quyết các bài toán logic đóng vai trò quan trọng trong việc xây dựng các hệ thống thông minh có khả năng suy luận và ra quyết định. Logic bậc nhất (first-order logic) là một trong những phương pháp phổ biến được sử dụng để biểu diễn và giải quyết các vấn đề logic trong AI. Tiểu luận này sẽ tập trung vào quá trình giải quyết các bài toán logic trong AI và vai trò của logic bậc nhất trong quá trình này.

Để giải quyết một bài toán logic trong AI, việc đầu tiên là xác định một bộ từ vựng cụ thể phù hợp với bài toán. Bộ từ vựng này bao gồm các hàm, quan hệ và quantifier mà chúng ta sẽ sử dụng để biểu diễn các yêu cầu, điều kiện và quy tắc của bài toán. Việc xác định bộ từ vựng chính xác là quan trọng để đảm bảo rằng các biểu diễn logic sau đó sẽ phản ánh đúng vấn đề cần giải quyết.

Sau khi xác định bộ từ vựng, chúng ta sử dụng logic bậc nhất để biểu diễn các mệnh đề của bài toán. Logic bậc nhất cho phép chúng ta mô hình hóa các yêu cầu, điều kiện và quy tắc của bài toán bằng cách sử dụng quantifier để chỉ định phạm vi của các biến và các biểu thức logic.

Tiếp theo, chúng ta sử dụng các kỹ thuật logic để giải quyết bài toán. Các kỹ thuật này có thể bao gồm sử dụng luật suy luận, tìm kiếm, ràng buộc và các phương pháp khác để phân tích và đưa ra các giải pháp hoặc kết luận logic cho bài toán.

Cuối cùng, chúng ta cần kiểm tra và đảm bảo tính đúng đắn của các biểu diễn logic và giải pháp của chúng. Việc này đảm bảo rằng các giải pháp được đưa ra là logic và chính xác, phản ánh đúng vấn đề cần giải quyết.

Trong tiểu luận này, chúng ta đã thảo luận về quá trình giải quyết các bài toán logic trong AI và vai trò của logic bậc nhất trong quá trình này. Bằng cách sử dụng logic bậc nhất và các kỹ thuật logic phù hợp, chúng ta có thể xây dựng các hệ thống thông minh có khả năng suy luận và ra quyết định, đóng góp vào sự phát triển của trí tuệ nhân tạo.

# TÀI LIỆU THAM KHẢO

<https://brainly.com/>

<https://www.chegg.com/homework-help/questions-and-answers/following-valid-necessarily-true-sentences-1-x-xx-y-z-yz-2-xp-x-p-x-3-xsmart-x-xx--q146491116>

<https://quizlet.com/>

<https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/>

<https://zoo.cs.yale.edu/classes/cs470/materials/hws/hw4/hw4ans.html>

<https://www.chegg.com/>

<https://brainly.com/>

<https://translate.google.com/>

<https://www.researchgate.net/figure/Initial-state-and-goal-state-of-8-puzzle_fig1_357733694>

<https://www.sentrywire.com/simultaneous-search>

<https://ai.stackexchange.com/questions/41990/what-does-the-branching-factor-mean-in-the-time-complexity-of-breadth-first-sear>

<https://www.elcom.com.vn/may-hoc-machine-learning-la-gi-ung-dung-cong-nghe-may-hoc-trong-thuc-tien-1666003970>

<https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_m%C3%A1y>

<https://lotusacademy.edu.vn/blog/bai-2-khong-gian-trang-thai-va-cac-phuong-phap-tim-kiem-mu-187>

<https://nghienmaytinh.wordpress.com/2012/04/02/khong-gian-tr%E1%BA%A1ng-thai-va-cac-gi%E1%BA%A3i-thu%E1%BA%ADt-tim-ki%E1%BA%BFm/>

<https://users.soict.hust.edu.vn/huonglt/AI/Chuong%203.%20Tim%20kiem%20co%20ban.pdf>

<https://viblo.asia/p/cac-thuat-toan-co-ban-trong-ai-phan-biet-best-first-search-va-uniform-cost-search-ucs-Eb85omLWZ2G>