ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

ĐỀ THI DỰ PHÒNG
Học phần: NHẬP MÔN TRÍ TUỆ NHÂN TẠO
ĐỀ SỐ 2

Thời gian làm bài: 90 phút
Tổng điểm: 10

Phần 1: Đề bài

Bài 1 (1.5 điểm): Lịch sử phát triển của AI chứng kiến sự cạnh tranh và bổ sung giữa hai trường phái chính: **Chủ nghĩa Tượng trưng (Symbolicism)** và **Chủ nghĩa Kết nối (Connectionism)**.

a) Mô tả ngắn gọn cách tiếp cận cốt lõi của mỗi trường phái trong việc tạo ra trí tuệ.

b) Nêu một ví dụ về một hệ thống hoặc kỹ thuật AI tiêu biểu cho mỗi trường phái và giải thích ngắn gọn tại sao nó thuộc về trường phái đó.

Bài 2 (1.0 điểm): Cho một tập các mệnh đề Horn (tri thức - KB) và một câu truy vấn (query) Q:

- **KB:**
  1. P
  2. W
  3. $(P \land Q) \Rightarrow L$
  4. $(W \land L) \Rightarrow S$
- **Query:** S

Hãy sử dụng thuật toán **suy diễn tiến (Forward Chaining)** để xác định xem S có thể được suy ra từ KB hay không. Trình bày các bước của thuật toán.

Bài 3 (1.0 điểm): Trong phép hợp nhất (unification) của logic vị từ bậc nhất, "occurs check" (kiểm tra xuất hiện) là một bước quan trọng để đảm bảo thuật toán kết thúc.

a) Giải thích "occurs check" là gì?

b) Cho ví dụ về hai biểu thức (terms) mà việc hợp nhất chúng sẽ thất bại do vi phạm "occurs check" và giải thích tại sao.

Bài 4 (1.0 điểm): Giải thích vấn đề "bùng nổ tổ hợp" (combinatorial explosion) trong bối cảnh tìm kiếm một bằng chứng logic. Tại sao không gian tìm kiếm cho một bằng chứng lại có thể tăng lên một cách nhanh chóng? Nêu một chiến lược chung mà các hệ thống AI sử dụng để đối phó với vấn đề này.

Bài 5 (1.5 điểm): Toán tử `cut` (dấu `!`) là một cơ chế điều khiển thực thi quan trọng trong PROLOG, nhưng nó có thể thay đổi ngữ nghĩa của chương trình. Xét chương trình tìm giá trị nhỏ nhất của hai số:

```
% Phiên bản 1: Không có cut
min(X, Y, X) :- X =< Y.
min(X, Y, Y) :- X > Y.

% Phiên bản 2: Có cut
min_cut(X, Y, X) :- X =< Y, !.
min_cut(X, Y, Y).
```

a) Với truy vấn `min(2, 5, M)`, cả hai phiên bản có cho cùng một kết quả không? Tại sao?

b) Với phiên bản 2, nếu ta truy vấn `min_cut(5, 2, 2)`, chương trình sẽ trả về `true` hay `false`? Giải thích tại sao hành vi này có thể được coi là một sai sót về mặt logic so với phiên bản 1.

Bài 6 (2.0 điểm): Trong thuật toán tìm kiếm A*, chất lượng của hàm heuristic là yếu tố quyết định hiệu suất và tính đúng đắn.

a) Định nghĩa một hàm heuristic $h(n)$ là **"admissible" (chấp nhận được)** và **"consistent" (nhất quán)**.

b) Tại sao tính "admissible" lại là điều kiện cần thiết để A* đảm bảo tìm được lời giải tối ưu? Giải thích bằng cách chỉ ra điều gì có thể xảy ra nếu heuristic không "admissible" (tức là đánh giá quá cao chi phí thực tế).

Bài 7 (2.0 điểm): Giả sử bạn có một Mạng Bayesian mô tả mối quan hệ giữa việc trời mưa (`Rain`), sân bị trơn trượt (`Slippery`), và có tai nạn xảy ra (`Accident`).

$$Rain \rightarrow Slippery \rightarrow Accident$$

a) Viết công thức xác suất đồng thời $P(R, S, A)$ cho mạng này.

b) Dựa vào cấu trúc mạng, hãy cho biết câu nào sau đây là đúng và giải thích:

   (a) `Rain` và `Accident` là độc lập.

   (b) `Rain` và `Accident` là độc lập có điều kiện khi biết `Slippery`.

- HẾT -

Cán bộ coi thi không giải thích gì thêm. Sinh viên không được sử dụng tài liệu.

Phần 2: Đáp án và Thang điểm

Câu 1 (1.5 điểm)

a) Mô tả cách tiếp cận (1.0 điểm):

- **Chủ nghĩa Tượng trưng (Symbolicism) (0.5đ):** Cách tiếp cận này cho rằng trí tuệ có thể được tạo ra bằng cách thao tác trên một hệ thống các ký hiệu (symbol) theo các quy tắc hình thức (formal rules). Nó tập trung vào việc biểu diễn tri thức một cách tường minh thông qua logic, quy tắc sản xuất (production rules), và các cấu trúc dữ liệu phức tạp. Cốt lõi là suy luận logic.
- **Chủ nghĩa Kết nối (Connectionism) (0.5đ):** Cách tiếp cận này lấy cảm hứng từ bộ não sinh học, cho rằng trí tuệ xuất hiện từ sự tương tác của nhiều đơn vị xử lý đơn giản (nơ-ron nhân tạo) được kết nối với nhau trong một mạng lưới. Tri thức không được biểu diễn tường minh mà được "mã hóa" trong trọng số của các kết nối, và hệ thống học hỏi bằng cách điều chỉnh các trọng số này dựa trên dữ liệu.

b) Ví dụ (0.5 điểm):

- **Tượng trưng:** Hệ chuyên gia (Expert System) như MYCIN. Hệ thống này sử dụng hàng trăm quy tắc dạng `IF-THEN` để chẩn đoán bệnh nhiễm trùng máu, thể hiện rõ việc biểu diễn tri thức tường minh và suy luận dựa trên quy tắc.
- **Kết nối:** Mạng nơ-ron tích chập (CNN) dùng để nhận dạng hình ảnh. Mạng này học cách nhận biết các đối tượng (mèo, chó) bằng cách xử lý hàng triệu hình ảnh và tự động điều chỉnh trọng số kết nối để rút ra các đặc trưng từ cấp thấp (cạnh, góc) đến cấp cao (mắt, tai), mà không cần các quy tắc tường minh.

Câu 2 (1.0 điểm)

- **Khởi tạo (0.25đ):**
  - `agenda` = {P, W} (các sự kiện đã biết)
  - `count` = {`c1`: 2, `c2`: 2} (số lượng vế trái của mỗi luật, `c1` cho luật 3, `c2` cho luật 4)
  - `inferred` = {P: false, W: false, L: false, S: false}
- **Vòng lặp 1 (0.25đ):**
  - Lấy `P` từ `agenda`. `inferred[P]` = true.
  - `P` có trong vế trái của luật 3. Giảm `count[c1]` từ 2 xuống 1.
- **Vòng lặp 2 (0.25đ):**
  - Lấy `W` từ `agenda`. `inferred[W]` = true.
  - `W` có trong vế trái của luật 4. Giảm `count[c2]` từ 2 xuống 1.
- **Vòng lặp 3 (0.25đ):**

- Không có gì trong `agenda`. Thuật toán có thể dừng nếu không có gì được thêm vào. (Lưu ý: Thứ tự xử lý P, W không quan trọng. Hiện tại trong KB không có Q, vậy nên S không thể được chứng minh. Giả sử Q được thêm vào KB).
- Vậy giả sử KB có thêm {Q}. `agenda` = {P, W, Q}.
- Sau khi xử lý P, W, ta xử lý Q. `inferred[Q]` = true. `Q` có trong vế trái luật 3. Giảm `count[c1]` từ 1 xuống 0. Luật 3 được kích hoạt. Thêm `L` vào `agenda`.

- **Vòng lặp 4:**
  - Lấy `L` từ `agenda`. `inferred[L]` = true. `L` có trong vế trái luật 4. Giảm `count[c2]` từ 1 xuống 0. Luật 4 được kích hoạt. Thêm `S` vào `agenda`.

- **Kết luận:**
  - Nếu có thêm Q trong KB, thì: `S` được thêm vào `agenda`, do đó `S` có thể được suy ra từ KB.
  - Nếu không có Q, thì `S` không thể được suy ra từ KB.

Câu 3 (1.0 điểm)

a) Giải thích (0.5đ): "Occurs check" là một bước trong thuật toán hợp nhất (unification) để kiểm tra xem một biến có xuất hiện bên trong biểu thức mà nó đang được hợp nhất với hay không. Nếu biến đó xuất hiện, phép hợp nhất phải thất bại để tránh tạo ra một biểu thức đệ quy vô hạn (ví dụ: X = f(X)).

b) Ví dụ (0.5đ):

- **Biểu thức 1:** X
- **Biểu thức 2:** $father(X)$
- **Giải thích:** Khi hợp nhất X với $father(X)$, thuật toán sẽ cố gắng tạo ra một phép thế $\{X/father(X)\}$. Tuy nhiên, "occurs check" sẽ phát hiện ra rằng biến X xuất hiện bên trong biểu thức $father(X)$. Nếu cho phép phép thế này, X sẽ trở thành $father(father(father(...)))$, một cấu trúc vô hạn. Do đó, phép hợp nhất thất bại.

Câu 4 (1.0 điểm)

- **Giải thích vấn đề (0.5đ):** "Bùng nổ tổ hợp" trong suy luận logic là hiện tượng không gian tìm kiếm (search space) cho một bằng chứng tăng lên theo cấp số nhân (hoặc nhanh hơn) khi số lượng sự kiện và quy tắc trong cơ sở tri thức tăng lên. Tại mỗi bước suy luận, có thể có nhiều quy tắc được áp dụng, và mỗi quy tắc lại mở ra nhiều khả năng mới, tạo thành một cây tìm kiếm rất lớn và sâu.
- **Chiến lược đối phó (0.5đ):** Một chiến lược chung là sử dụng **heuristics** để hướng dẫn việc tìm kiếm. Thay vì thử tất cả các khả năng một cách mù quáng (như tìm kiếm theo chiều rộng), hệ thống sẽ ưu tiên áp dụng những quy tắc hoặc đi theo những hướng có vẻ "hứa hẹn" nhất để dẫn đến lời giải. Ví dụ: chiến lược "set-of-support" trong hợp giải, ưu tiên các hợp giải liên quan đến mệnh đề phủ định của câu truy vấn.

Câu 5 (1.5 điểm)

a) Truy vấn `min(2, 5, M)` (0.5đ):

- Cả hai phiên bản đều cho cùng một kết quả `M = 2`.
- **Giải thích:** Trong cả hai trường hợp, truy vấn `min(2, 5, M)` khớp với quy tắc đầu tiên (`X =< Y` tức là `2 =< 5` là đúng). PROLOG sẽ gán `M = X = 2`. Vì đã tìm thấy một lời giải, và trong phiên bản 2, `cut` (`!`) ngăn chặn việc tìm kiếm các lời giải khác, PROLOG sẽ dừng lại và báo cáo kết quả này.

b) Truy vấn `min_cut(5, 2, 2)` (1.0đ):

- Chương trình sẽ trả về **`true`**. (0.5đ)
- **Giải thích (0.5đ):**
  1. Truy vấn `min_cut(5, 2, 2)` được thử khớp với `min_cut(X, Y, X) :- X =< Y, !.`. Phép hợp nhất `X=5, Y=2` được thực hiện. Phần thân được thực thi: `5 =< 2` là **sai**. Quy tắc này thất bại.
  2. PROLOG chuyển sang quy tắc thứ hai `min_cut(X, Y, Y).`. Nó hợp nhất `min_cut(5, 2, 2)` với `min_cut(X, Y, Y)`. Phép hợp nhất này thành công với `X=5, Y=2`. Vì quy tắc này không có phần thân, nó trả về **`true`**.

  Sai sót logic: Vấn đề ở đây là `min_cut(5, 2, 2)` lại là `true`, trong khi rõ ràng min(5, 2) phải là 5. Quy tắc thứ hai `min_cut(X, Y, Y)` lẽ ra chỉ nên được thực thi khi quy tắc đầu tiên sai. Việc thêm `cut` vào quy tắc đầu tiên đã phá vỡ logic này, khiến quy tắc thứ hai trở thành một sự thật đúng một cách vô điều kiện (`min_cut` của X và Y luôn là Y) mỗi khi quy tắc đầu tiên thất bại.

Câu 6 (2.0 điểm)

a) Định nghĩa (1.0 điểm):

- **Admissible (chấp nhận được) (0.5đ):** Một hàm heuristic $h(n)$ được gọi là "admissible" nếu với mọi nút n, giá trị của nó không bao giờ lớn hơn chi phí thực tế để đi từ n đến nút đích gần nhất. Tức là, $0 \leq h(n) \leq h^*(n)$, trong đó $h^*(n)$ là chi phí tối ưu thực sự. Heuristic là một ước tính "lạc quan".
- **Consistent (nhất quán) (0.5đ):** Một hàm heuristic $h(n)$ được gọi là "consistent" (hoặc đơn điệu) nếu với mọi nút n và mọi nút n' là hàng xóm của n, chi phí ước tính từ n không lớn hơn chi phí đi từ n đến n' cộng với chi phí ước tính từ n'. Tức là, $h(n) \leq c(n, a, n') + h(n')$. Đây là một dạng của bất đẳng thức tam giác.

b) Tầm quan trọng của "Admissibility" (1.0 điểm):

- A* đảm bảo tìm được lời giải tối ưu nếu heuristic là "admissible". A* sử dụng hàm đánh giá $f(n) = g(n) + h(n)$, trong đó $g(n)$ là chi phí thực tế từ điểm bắt đầu đến n.
- Nếu heuristic **không admissible**, tức là $h(n) > h^*(n)$ tại một nút n nào đó, A* có thể bị "đánh lừa". Giả sử có một nút n nằm trên đường đi không tối

ưu, nhưng do $h(n)$ bị đánh giá quá cao, giá trị $f(n)$ của nó có thể trở nên rất lớn. Trong khi đó, một nút m khác nằm trên đường đi tối ưu thực sự, nhưng nếu $h(m)$ được ước tính một cách chính xác (hoặc thấp hơn), có thể xảy ra $f(n) < f(m)$. Khi đó, A* sẽ không bao giờ mở rộng nút m trên đường đi tối ưu vì nó ưu tiên mở rộng các nút có $f(n)$ nhỏ hơn. Thuật toán sẽ tìm thấy một lời giải, nhưng đó là lời giải đi qua n và không phải là lời giải tối ưu.

Câu 7 (2.0 điểm)

a) Công thức xác suất đồng thời (1.0 điểm): Dựa trên quy tắc chuỗi và cấu trúc mạng (một chuỗi Markov), ta có:

$$P(R, S, A) = P(R) \cdot P(S|R) \cdot P(A|S)$$

b) Phân tích độc lập (1.0 điểm):

- **1. `Rain` và `Accident` là độc lập. Sai**. Việc biết trời mưa làm tăng khả năng đường trơn, và do đó làm tăng khả năng có tai nạn. Thông tin về `Rain` có ảnh hưởng đến `Accident` thông qua `Slippery`. $P(A|R) \neq P(A)$.
- **2. `Rain` và `Accident` là độc lập có điều kiện khi biết `Slippery`. Đúng**. (0.5đ cho lựa chọn đúng, 0.5đ cho giải thích). Theo cấu trúc của chuỗi Markov, một nút chỉ phụ thuộc vào nút cha trực tiếp của nó. Khi ta đã biết trạng thái của `Slippery` (ví dụ, ta biết chắc chắn đường đang trơn), thì nguyên nhân gây ra nó (`Rain`) không còn cung cấp thêm thông tin gì về hậu quả của nó (`Accident`). Nói cách khác, `Slippery` đã "chặn" dòng ảnh hưởng từ `Rain` đến `Accident`. Ký hiệu: $P(A|S, R) = P(A|S)$.