Trí Tuệ Nhân Tạo

Nguyễn Nhật Quang

quangnn-fit@mail.hut.edu.vn

Viện Công nghệ Thông tin và Truyền thông
Trường Đại học Bách Khoa Hà Nội

Năm học 2009-2010

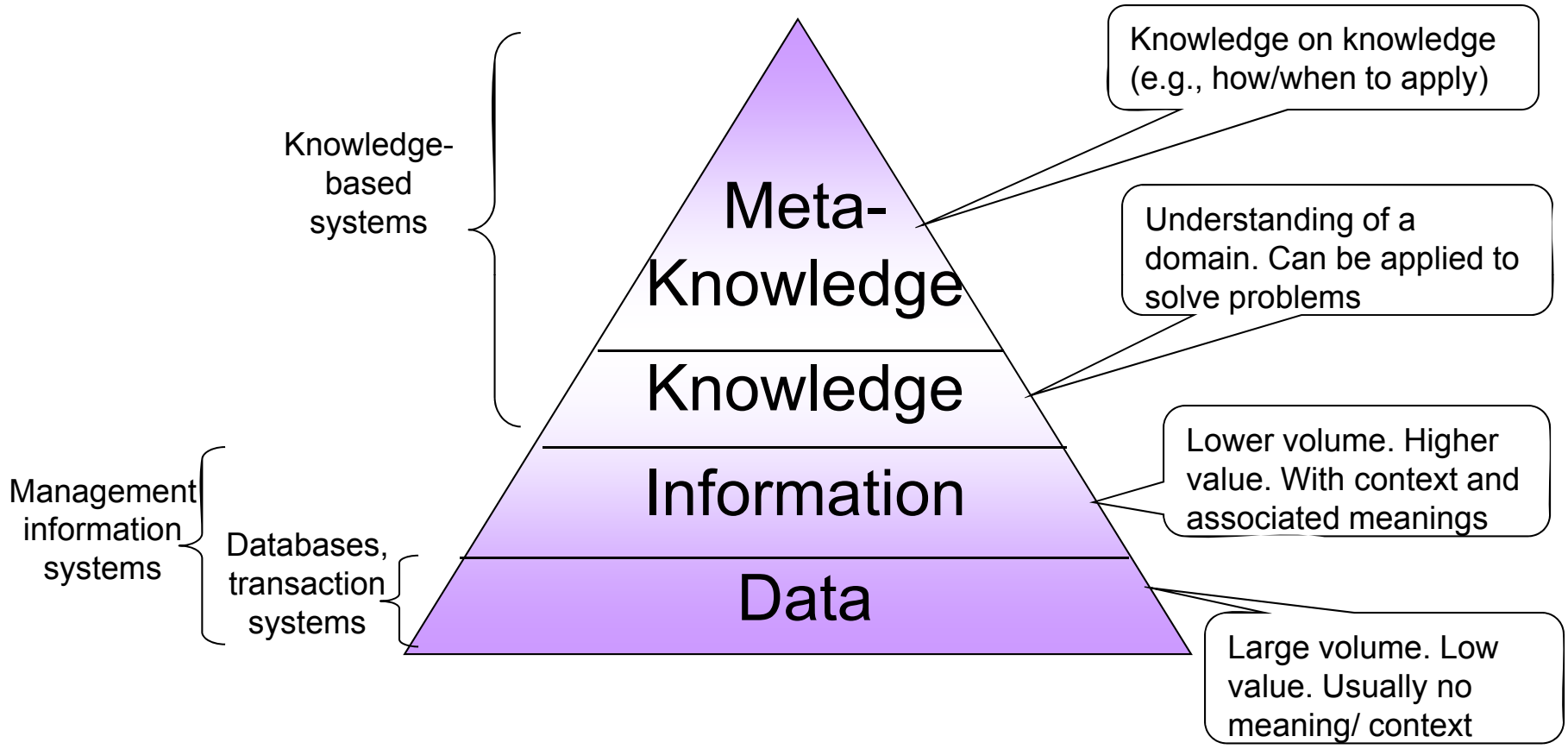
Nội dung môn học:

- Giới thiệu về Trí tuệ nhân tạo
- Tác tử
- Giải quyết vấn đề: Tìm kiếm, Thỏa mãn ràng buộc
- Logic và suy diễn
- **Biểu diễn tri thức**
  - **Luật sản xuất**
  - **Khung**
  - **Mạng ngữ nghĩa**
  - **Ontology**
- Suy diễn với tri thức không chắc chắn
- Học máy
- Lập kế hoạch

Dữ liệu, Thông tin, Tri thức (1)

- **Dữ liệu (data)** thường được định nghĩa là các sự kiện (facts) hoặc các ký hiệu (symbols)
- **Thông tin (information)** thường được định nghĩa là dữ liệu đã được xử lý hoặc chuyển đổi thành những dạng hoặc cấu trúc phù hợp cho việc sử dụng của con người
- Thông tin có được sau (chứ không xuất hiện trước) dữ liệu
- **Tri thức (knowledge)** thường được định nghĩa là sự hiểu biết (nhận thức) về thông tin

Dữ liệu, Thông tin, Tri thức (2)



(Adapted from "Knowledge Engineering course (CM3016), by K. Hui 2008-2009")

Dữ liệu, Thông tin, Tri thức (3)

- Dữ liệu
  - Nhiệt độ ngoài trời là 5 độ C
- Thông tin
  - Ngoài trời thời tiết lạnh
- Tri thức
  - Nếu ngoài trời thời tiết lạnh thì bạn nên mặc áo choàng ấm (khi đi ra ngoài)

→Giá trị (sử dụng) của dữ liệu tăng lên khi nó được "chuyển đổi" thành tri thức

→Sử dụng tri thức sẽ cho phép đưa ra các quyết định phù hợp và hiệu quả

Biểu diễn tri thức (1)

- Biểu diễn tri thức (Knowlegde representation) là một lĩnh vực nghiên cứu quan trọng của Trí tuệ nhân tạo
  - Nhằm phát triển các phương pháp, cách thức biểu diễn tri thức và các công cụ hỗ trợ việc biểu diễn tri thức
- Tồn tại nhiều phương pháp biểu diễn tri thức
  - **Luật sản xuất (Production rules)**
  - **Khung (Frames)**
  - **Mạng ngữ nghĩa (Semantic networks)**
  - Ontology
  - Các mô hình xác suất
  - …

Biểu diễn tri thức (2)

- Tính hoàn chỉnh (Completeness)
  - Phương pháp biểu diễn có hỗ trợ việc thu thập và thể hiện mọi khía cạnh của tri thức (của một lĩnh vực cụ thể)?
- Tính ngắn gọn (Conciseness)
  - Phương pháp biểu diễn có cho phép việc thu thập tri thức một cách hiệu quả?
  - Phương pháp biểu diễn có cho phép việc lưu trữ và truy nhập dễ dàng tri thức không?
- Tính hiệu quả về tính toán (Computational efficiency)
- Tính rõ ràng, dễ hiểu (Transparency)
  - Phương pháp biểu diễn có cho phép diễn giải (để người dùng hiểu) về các hoạt động và các kết luận của hệ thống?

Biểu diễn tri thức bằng luật (1)

- Biểu diễn tri thức bằng các luật (rules) là cách biểu diễn phổ biến nhất trong các hệ cơ sở tri thức
  - Một luật chứa đựng (biểu diễn) tri thức về việc giải quyết một vấn đề nào đó
  - Các luật được tạo nên khá dễ dàng, và dễ hiểu
- Một luật được biểu diễn ở dạng:

$$\textbf{IF } A_1 \textbf{ AND } A_2 \textbf{ AND } \ldots \textbf{ AND } A_n \textbf{ THEN } B$$

- A_i
  - Là các **điều kiện (conditions, antecedents, premises)**
- **B**
  - Là **kết luận (conclusion, consequence, action)**

Biểu diễn tri thức bằng luật (2)

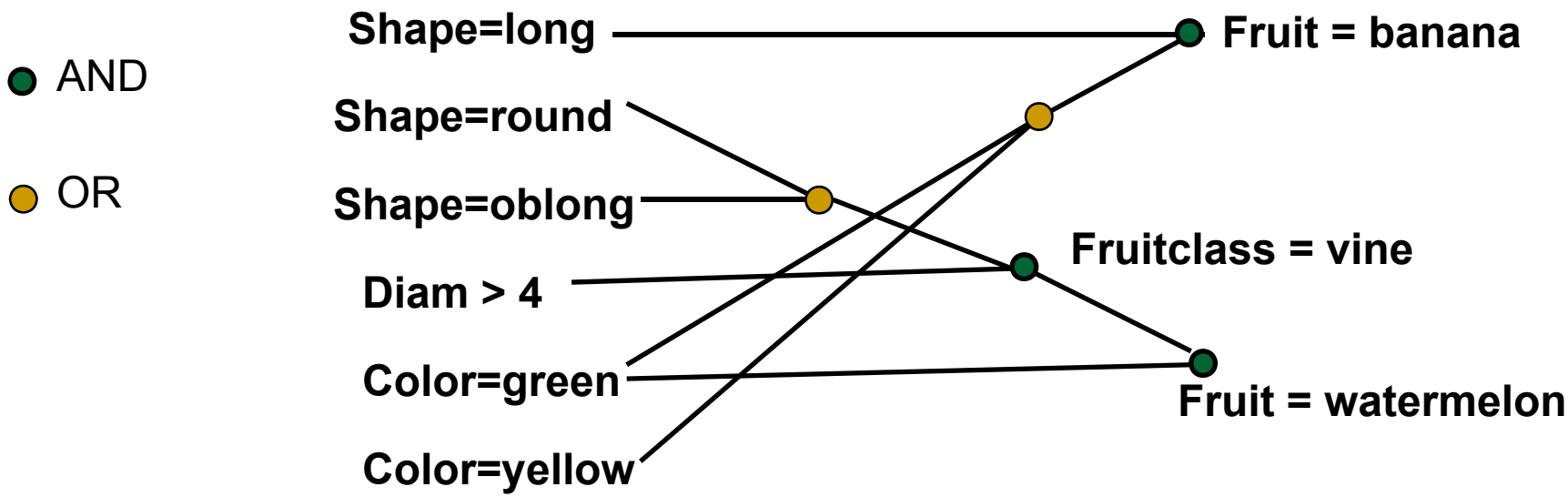
- Mệnh đề điều kiện của một luật
  - Không cần sử dụng toán tử logic OR
  - Một luật với toán tử logic OR trong mệnh đề điều kiện, thì sẽ được chuyển thành một tập các luật tương ứng không chứa OR
  - Ví dụ: Luật (IF $A_1 \vee A_2$ THEN B) được chuyển thành 2 luật (IF A_1 THEN B) và (IF A_2 THEN B)
- Mệnh đề kết luận của một luật
  - Không cần sử dụng toán tử logic AND
  - Một luật với toán tử logic AND trong mệnh đề kết luận, thì sẽ được chuyển thành một tập các luật tương ứng không chứa AND
  - Ví dụ: Luật (IF … THEN $B_1 \wedge B_2$) được chuyển thành 2 luật (IF … THEN B_1) and (IF … THEN B_2)
  - Không cho phép sử dụng toán tử OR!

Các kiểu luật

- Các kiểu luật khác nhau để biểu diễn các kiểu tri thức khác nhau
- Quan hệ liên kết
  - `IF addressAt(x, Hospital) THEN heathIs(x, Bad)`
- Quan hệ nguyên nhân (kết quả)
  - `IF diseaseType(x, Infection) THEN tempIs(x, High)`
- Tình huống và hành động (gợi ý)
  - `IF diseaseType(x, Infection) THEN takeMedicine(x, Antibiotic)`
- Quan hệ logic
  - `IF tempGreater(x, 37) THEN isFever(x)`

Đồ thị AND/OR (1)

- IF (Shape=long) **AND** (Color=(green **OR** yellow)) THEN (Fruit=banana)
- IF (Shape=(round **OR** oblong)) **AND** (Diam > 4) THEN (Fruitclass=vine)
- IF (Fruitclass=vine) **AND** (Color=green) THEN (Fruit=watermelon)



Đồ thị AND/OR (2)

- Luật *IF (Shape=long)* ***AND*** *(Color=(green* ***OR*** *yellow)) THEN (Fruit=banana)* được tạo nên bởi các luật:
  - IF (Shape=long) **AND** (Color=green) THEN (Fruit=banana)
  - IF (Shape=long) AND (Color=yellow) THEN (Fruit=banana)
- Luật *IF (Shape=(round* ***OR*** *oblong))* ***AND*** *(Diam > 4) THEN (Fruitclass=vine)* được tạo nên bởi các luật:
  - IF (Shape=round) **AND** (Diam > 4) THEN (Fruitclass=vine)
  - IF (Shape=oblong) **AND** (Diam > 4) THEN (Fruitclass=vine)

Các vấn đề với biểu diễn luật

- Các luật có chứa các vòng lặp
  - IF A THEN A
  - {IF A THEN B, IF B THEN C, IF C THEN A}
- Các luật có chứa mâu thuẫn
  - {IF A THEN B, IF B THEN C, IF A AND D THEN $\neg$C}
- Các kết luận không thể suy ra được (từ các luật hiện có)
- Khó khăn trong việc thay đổi (cập nhật) cơ sở tri thức
  - Cơ sở tri thức cũ: {IF A_1 THEN B_1, IF A_2 THEN B_2, …, IF A_n THEN B_n}
  - Cần bổ sung thêm điều kiện *C* vào tất cả các luật
  - Cơ sở tri thức mới: {IF A_1 AND C THEN B_1, IF A_2 AND C THEN B_2, …, IF A_n AND C THEN B_n}

Sử dụng các luật trong suy diễn

- So khớp mẫu (Pattern matching)
  - Để kiểm tra một luật có thể được sử dụng (áp dụng) hay không
  - Ví dụ: Nếu cơ sở tri thức chứa đựng tập các luật {IF A_1 THEN B_1, IF A_1 AND A_2 THEN B_2, IF A_2 AND A_3 THEN B_3} và các sự kiện (được lưu trong bộ nhớ làm việc) bao gồm A_1 và A_2, thì 2 luật đầu tiên có thể được sử dụng
- Chuỗi suy diễn (chuỗi áp dụng các luật)
  - Xác định trật tự áp dụng các luật trong quá trình suy diễn
  - Với một tập các luật và một tập các sự kiện (các giả thiết), các luật nào nên được sử dụng, và theo trật tự nào, để đạt tới (suy ra) một kết luật cần chứng minh?
  - 2 chiến lược suy diễn: tiến (forward) vs. lùi (backward)
  - 2 chiến lược suy diễn này đã được trình bày trong bài trước!

Giải quyết xung đột

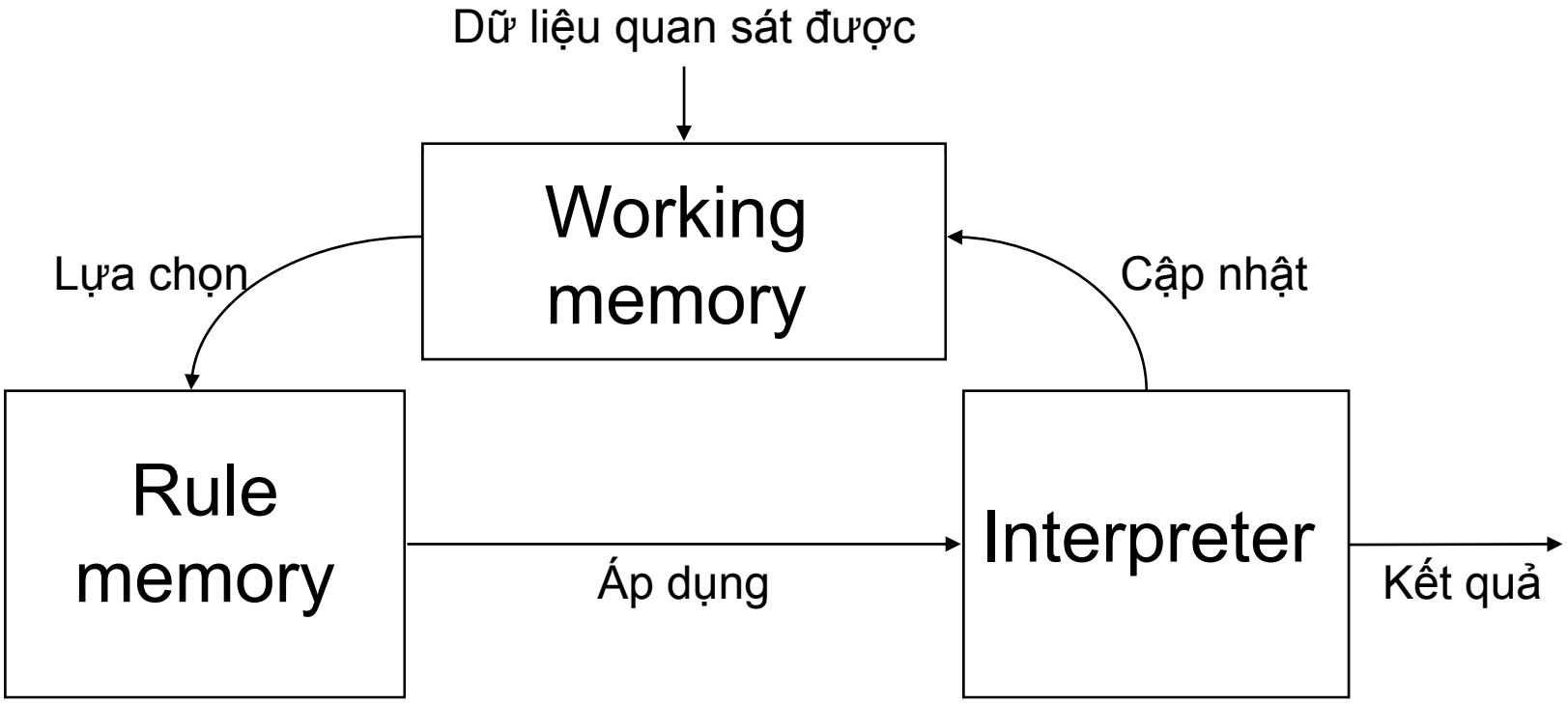
- Một xung đột (conflict) xảy ra khi có nhiều hơn một luật có thể áp dụng được (phù hợp với các sự kiện trong bộ nhớ làm việc)
  - Lưu ý, một xung đột không phải là một mâu thuẫn của tập luật
- Trong trường hợp xảy ra xung đột, cần một chiến lược giải quyết xung đột (conflict resolution strategy - CRS) để quyết định luật nào được (ưu tiên) áp dụng
- Sự lựa chọn thích hợp một chiến lược giải quyết xung đột có thể mang lại những cải thiện đáng kể đối với quá trình suy diễn của hệ thống

Chiến lược giải quyết xung đột

- Áp dụng luật xuất hiện đầu tiên (theo thứ tự) trong cơ sở tri thức
- Không áp dụng các luật sinh ra các kết quả (sự kiện) đã có trong bộ nhớ làm việc
- Áp dụng luật cụ thể nhất (luật có nhiều điều kiện nhất)
- Áp dụng các luật phù hợp với các sự kiện được đưa vào trong bộ nhớ làm việc gần thời điểm hiện tại nhất
- Không áp dụng lại một luật, nếu nó vẫn sinh ra cùng một tập các sự kiện (giống như lần áp dụng trước của nó)
- Áp dụng luật có độ tin cậy (chắc chắn) cao nhất
- …
- *Kết hợp* của các chiến lược trên

Hệ thống suy diễn dựa trên luật (1)

Kiến trúc điển hình của một hệ thống suy diễn dựa trên luật (Rule-based system – RBS)



(http://www.cwa.mdx.ac.uk/bis2040/johnlect.html)

Hệ thống suy diễn dựa trên luật (2)

- Bộ nhớ làm việc (Working memory)
  - Lưu giữ các sự kiện (các giả thiết đúng, đã được chứng minh)
  - Các sự kiện này sẽ quyết định những luật nào được áp dụng (bởi thành phần Interpreter)
- Bộ nhớ các luật (Rule memory)
  - Chính là cơ sở tri thức của hệ thống
  - Lưu giữ các luật có thể áp dụng
- Bộ diễn dịch (Interpreter)
  - Hệ thống bắt đầu bằng việc đưa một sự kiện (dữ liệu) phù hợp vào bộ nhớ làm việc
  - Khi sự kiện (dữ liệu) trong bộ nhớ làm việc phù hợp với các điều kiện của một luật trong bộ nhớ các luật, thì luật đó sẽ được áp dụng

RBS – Ưu điểm (1)

- Cách biểu diễn (diễn đạt) phù hợp
  - Rất gần với cách diễn đạt trong ngôn ngữ tự nhiên
  - Rất dễ dàng để diễn đạt các tri thức bởi các luật
- Dễ hiểu
  - Các luật dạng IF-THEN rất dễ hiểu (có lẽ là dễ hiểu nhất) đối với người sử dụng
  - Trong một lĩnh vực (bài toán) cụ thể, cách biểu diễn bằng luật giúp các chuyên gia trong lĩnh vực này có thể đánh giá và cải tiến các luật

RBS – Ưu điểm (2)

- Một cách biểu diễn tri thức theo kiểu khai báo (declarative)
  - Kỹ sư tri thức thu thập các tri thức (ở dạng các luật IF-THEN) về một lĩnh vực cụ thể, và đưa chúng vào trong một cơ sở các luật (rule base)
  - Kỹ sư tri thức (có thể) không cần phải quan tâm đến khi nào, làm thế nào, và theo trật tự nào mà các luật được sử dụng – Hệ thống sẽ tự động đảm nhận các nhiệm vụ này
- Dễ dàng mở rộng cơ sở tri thức
  - Chỉ việc bổ sung thêm các luật mới (các tri thức mới) vào cuối của cơ sở các luật

RBS – Nhược điểm

- Khả năng biểu diễn (diễn đạt) bị giới hạn
  - Trong nhiều lĩnh vực bài toán thực tế, tri thức của lĩnh vực bài toán đó không phù hợp với cách biểu diễn dạng (IF-THEN)
- Sự tương tác giữa các luật và trật tự của các luật trong cơ sở luật có thể gây ra các hiệu ứng không mong muốn
  - Trong quá trình thiết kế (design) và bảo trì (maintenance)một cơ sở luật, mỗi luật mới được đưa vào cần phải được cân nhắc (kiểm tra) với các luật đã có từ trước
  - Rất khó khăn và chi phí tốn kém để xem xét tất cả các tương tác (interactions) có thể giữa các luật

Biểu diễn tri thức bằng khung (1)

- Làm thế nào để biểu diễn tri thức "Xe buýt màu vàng"?
- Giải pháp thứ 1. `Yellow(bus)`
  - Câu hỏi "Cái gì là màu vàng?" có thể trả lời được
  - Nhưng câu hỏi "Màu của xe buýt là gì?" thì không thể trả lời được
- Giải pháp thứ 2. `Color(bus, yellow)`
  - Câu hỏi "Cái gì là màu vàng?" có thể trả lời được
  - Câu hỏi "Màu của xe buýt là gì?" có thể trả lời được
  - Nhưng câu hỏi "Thuộc tính nào của xe buýt có giá trị là màu vàng?" thì không thể trả lời được
- Giải pháp thứ 3. `Prop(bus, color, yellow)`
  - Tất cả 3 câu hỏi trên đều có thể trả lời được

Biểu diễn tri thức bằng khung (2)

- Một đối tượng được biểu diễn bởi:

  (Object, Property, Value)

  - Được gọi là cách biểu diễn bằng bộ ba *đối tượng-thuộc tính-giá trị (object-property-value)*

- Nếu chúng ta gộp nhiều thuộc tính của cùng một kiểu đối tượng thành một cấu trúc, thì chúng ta có cách biểu diễn hướng đối tượng (object-centered representation)

  Prop(Object, Property$_1$, Value$_1$)

  Prop(Object, Property$_2$, Value$_2$)

  …

  Prop(Object, Property$_n$, Value$_n$)

Object

Property$_1$
Property$_2$
. . .
Property$_n$

Biểu diễn hướng đối tượng

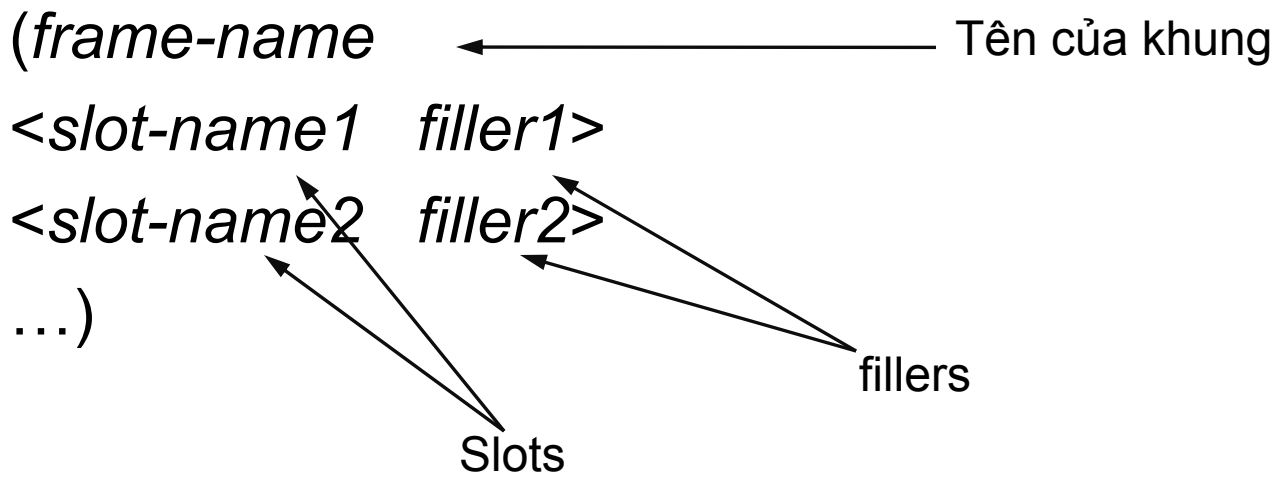
- Cách biểu diễn bằng bộ ba *object-property-value* là một cách tự nhiên để biểu diễn các đối tượng
- Các đối tượng cụ thể (Physical objects)
  - Ví dụ: Một *cái bàn* có các thuộc tính chất liệu bề mặt, số ngăn kéo, độ rộng, độ dài, độ cao, màu sắc, ...
- Các tình huống (Situations)
  - Ví dụ: Một *lớp học* có các thuộc tính mã số phòng học, danh sách sinh viên tham dự, giáo viên, ngày học, thời gian học, ...
  - Ví dụ: Một *chuyến đi nghỉ mát* có các thuộc tính nơi khởi hành, nơi đến, phương tiện di chuyển, phòng nghỉ, …

Khung (Frame)

- Có 2 kiểu khung: cụ thể (individual) và tổng quát (generic)
- **Khung cụ thể (Individual frames)**. Để biểu diễn một đối tượng cụ thể, chẳng hạn như một người cụ thể, một chuyến đi nghỉ mát cụ thể, ...
- **Khung tổng quát (Generic frames)**. Để biểu diễn một lớp (loại) các đối tượng, chẳng hạn như các sinh viên, các chuyến đi nghỉ mát, ...
- Ví dụ
  - Khung tổng quát: Europian_City
  - Khung cụ thể: City_Paris

Biểu diễn của một khung

- Một khung được biểu diễn bằng một danh sách định danh các thuộc tính được gọi là các **slots**
- Giá trị gán cho một thuộc tính được gọi là **filler of the slot**



Khung đơn

- Một khung đơn (individual frame) có một thuộc tính đặc biệt có tên là **INSTANCE-OF**, và giá trị của thuộc tính đặc biệt này là tên của một khung tổng quát (generic frame)

- Ví dụ

(toronto % sử dụng chữ thường cho các khung đơn
<:**INSTANCE-OF** CanadianCity>
<:Province ontario>
<:Population 4.5M>
…)

Khung tổng quát

- Một khung tổng quát (generic frames) có thể có một thuộc tính đặc biệt là **IS-A** mà giá trị của thuộc tính đặc biệt này là tên của một khung tổng quát khác

- Ví dụ

```
(CanadianCity          % sử dụng tên bắt đầu chữ hoa cho khung tổng quát
<:IS-A City>
<:Province CanadianProvince>
<:Country canada>
…)
```

Suy diễn với khung (1)

- Các thuộc tính (slots) trong các khung tổng quát có thể được gắn (liên kết) với các thủ tục để thực hiện và điều khiển việc suy diễn
- Có 2 kiểu thủ tục: **IF-NEEDED** và **IF-ADDED**
- Thủ tục **IF-NEEDED**
  - Được thực hiện khi không có giá trị cần thiết được gán cho một thuộc tính (no slot filler)
  - Ví dụ: (Table
    <:Clearance [**IF-NEEDED** computeClearance]>
    …)

    computeClearance là một thủ tục để tính toán (xác định) mức độ sạch sẽ của cái bàn

Suy diễn với khung (2)

- Thủ tục **IF-ADDED**
  - Được thực hiện khi một thuộc tính được gán giá trị, để cho phép *lan truyền ảnh hưởng của việc gán giá trị của thuộc tính* đó đối với các khung khác (ví dụ, để đảm bảo các ràng buộc trong bài toán)
  - Ví dụ: (Lecture

    <:DayOfWeek WeekDay>

    <:Date [**IF-ADDED** computeDayOfWeek]>

    …)

    Giá trị của thuộc tính *:DayOfWeek* sẽ được tính toán (lại) khi thuộc tính *:Date* được gán giá trị

Khung – Giá trị mặc định (1)

- Hãy xét khung tổng quát (generic frame) sau đây

  (CanadianCity
  <:**IS-A** City>
  <:Province CanadianProvince>
  <:Country canada>
  …)

- Hãy xét khung cụ thể (individual frame) sau đây

  (city134
  <:**INSTANCE-OF** CanadianCity>
  …)

- Đối với khung `city134`, giá trị (mặc định) cho thuộc tính *:Country* là canada

Khung – Giá trị mặc định (2)

- Hãy xét khung cụ thể (individual frame) sau đây

  (city135
  <:**INSTANCE-OF** CanadianCity>
  <:Country holland>
  …)

- Đối với khung `city135`, thì giá trị cho thuộc tính *:Country* là holland (chứ không phải là giá trị mặc định canada)

Khung – Tính kế thừa (1)

- Các thủ tục và các giá trị thuộc tính của một khung tổng quát hơn sẽ được áp dụng (kế thừa) bởi một khung cụ thể hơn, thông qua cơ chế kế thừa

- Ví dụ

(CoffeeTable
<:**IS-A** Table>
...)

(MahoganyCoffeeTable
<:**IS-A** CoffeeTable>
...)

Khung – Tính kế thừa (2)

- Ví dụ

(Elephant
 <:**IS-A** Mammal>
 <:Colour gray>
 ...)

(RoyalElephant
 <:**IS-A** Elephant>
 <:Colour white>
 …)

(clyde
 <:INSTANCE-OF RoyalElephant>
 …)

Khung – Suy diễn

- Quá trình suy diễn trong phương pháp biểu diễn bằng khung sẽ diễn ra như sau
  1. Người dùng khởi tạo một khung (tương đương với việc khai báo sự tồn tại của một đối tượng hay một tình huống)
  2. Các giá trị của các thuộc tính sẽ được kế thừa (từ các khung tổng quát hơn)
  3. Các thủ tục IF-ADDED sẽ được thực hiện. Việc này có thể sẽ dẫn đến việc khởi tạo của các khung khác, và việc gán giá trị của các thuộc tính
- Nếu người dùng hoặc một thủ tục yêu cầu việc gán giá trị cho một thuộc tính, thì:
  - Nếu có giá trị cho thuộc tính, thì giá trị đó sẽ được gán
  - Nếu không, thủ tục IF-NEEDED sẽ được thực hiện

Biểu diễn bằng khung – Ưu điểm

- Kết hợp được cả tri thức khai báo (declarative knowledge) và tri thức thủ tục (procedural knowledge) trong cùng một phương pháp biểu diễn
- Các khung được tổ chức có cấu trúc phân cấp, cho phép dễ dàng phân loại (phân lớp) tri thức
- Cấu trúc phân cấp các khung cho phép giảm bớt sự phức tạp (và chi phí) trong quá trình xây dựng cơ sở tri thức
- Cho phép thiết lập các ràng buộc đối với các giá trị được gán cho các thuộc tính (ví dụ: ràng buộc giá trị nhập vào phải nằm trong một khoảng giá trị cụ thể)
- Cho phép lưu giữ các giá trị mặc định (sử dụng thuộc tính đặc biệt IS-A, các giá trị của các thuộc tính của một khung tổng quát hơn được sử dụng để gán cho các thuộc tính của một khung cụ thể hơn)

Biểu diễn bằng khung – Nhược điểm

- Trong quá trình thiết kế cấu trúc phân cấp của các khung, cần rất để ý đến sự hợp lý của việc phân loại (các khung)
- Có thể gặp vấn đề chi phí cao cho việc thiết kế các thủ tục (IF-ADDED và IF-NEEDED) – Quá nhiều công sức dành cho việc thiết kế các thủ tục phù hợp, thay vì tập trung vào việc kiểm tra cấu trúc và nội dung của các khung
- Quá trình khai thác các khung có thể không hiệu quả, vì không có phương pháp hiệu quả để lưu trữ dữ liệu (của các khung) trong máy tính

Mạng ngữ nghĩa (1)

- **Mạng ngữ nghĩa (Semantic Network)** được đề cử bởi Quillian vào năm 1966 như là một mô hình biểu diễn bộ nhớ của con người
- Các động cơ thúc đẩy sự phát triển của mạng ngữ nghĩa
  - Để hiểu về cấu trúc của bộ nhớ con người, và việc sử dụng cấu trúc bộ nhớ này trong xử lý (hiểu) ngôn ngữ
  - Kiểu biểu diễn nào cho phép lưu giữ các ý nghĩa (meanings) của các từ để có thể sử dụng lại các ngữ nghĩa này (như trong bộ nhớ con người)?
  - Giả sử rằng con người sử dụng cùng cấu trúc bộ nhớ cho các công việc
- Các chứng minh về tâm lý học đã chỉ ra rằng bộ nhớ của con người sử dụng các liên kết trong việc xử lý (hiểu) các từ
- Yêu cầu: cần biểu diễn định nghĩa từ điển của các từ, để
  - So sánh và phân biệt ý nghĩa của 2 từ
  - Sinh ra các câu "tựa" (gần giống) tiếng Anh để mô tả sự so sánh này

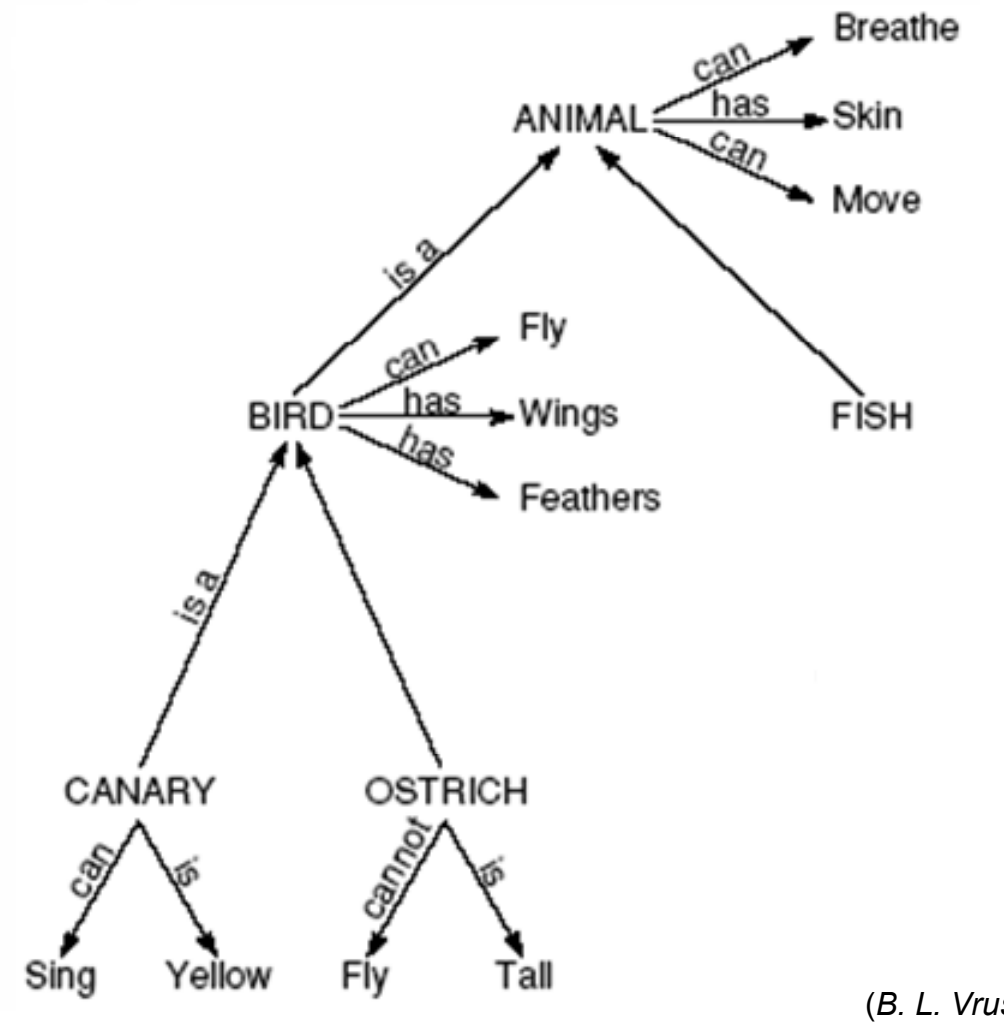
Mạng ngữ nghĩa (2)

- Mạng ngữ nghĩa (Semantic Network - SN) là phương pháp biểu diễn dựa trên đồ thị (graph-based representation)
- Một mạng ngữ nghĩa bao gồm một tập **các nút (nodes)** và **các liên kết (links)** để biểu diễn định nghĩa của một khái niệm (hoặc của một tập các khái niệm)
  - Các nút biểu diễn các khái niệm
  - Các liên kết biểu diễn các mối quan hệ (liên hệ) giữa các khái niệm
- Quá trình suy diễn (reasoning/inference) trong mạng ngữ nghĩa được thực hiện thông qua cơ chế lan truyền
  - Tác động (Activation)
  - Kế thừa (Inheritance)

Mạng ngữ nghĩa – Cú pháp

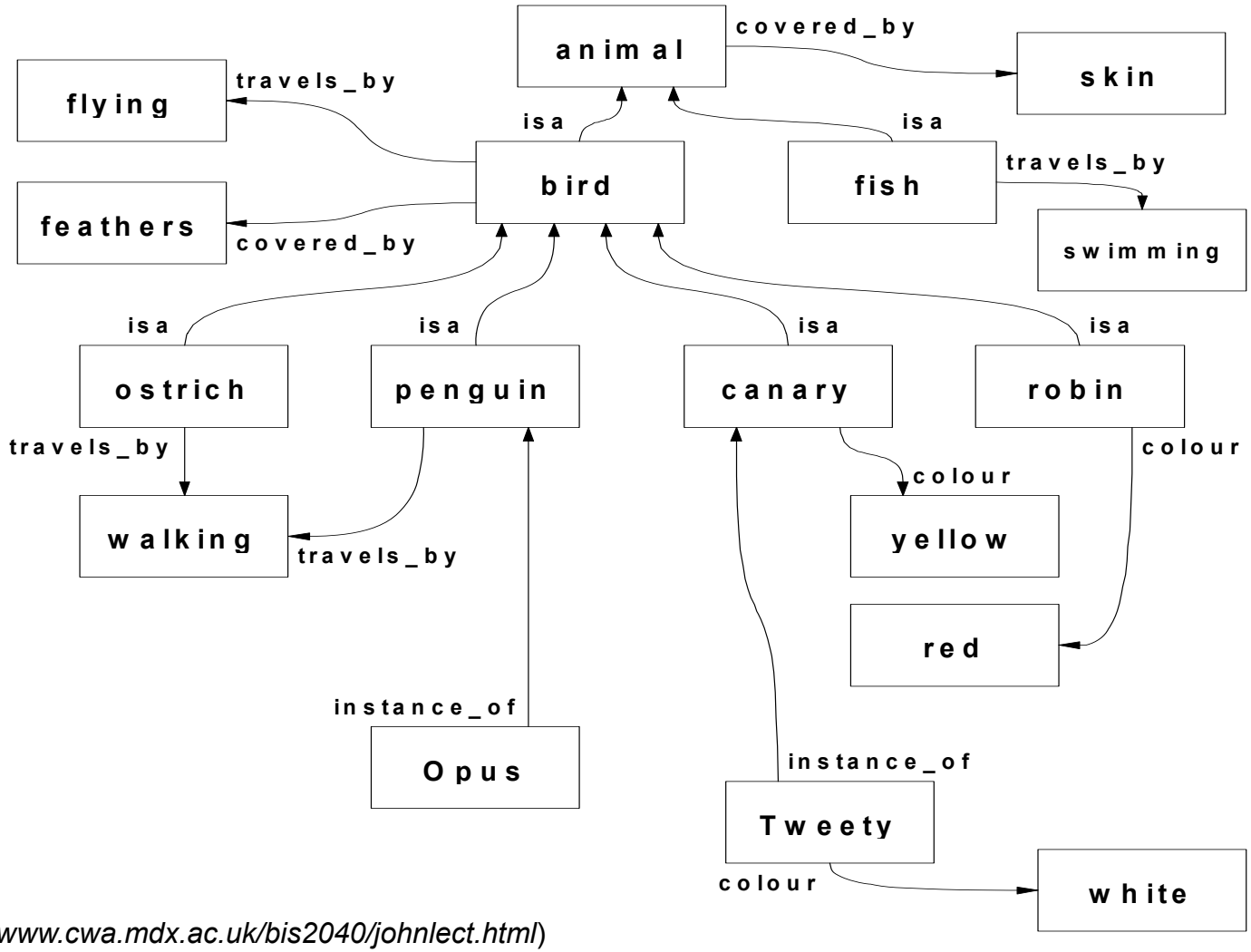
- **Các nút (nodes)** biểu diễn các *khái niệm (concepts)*, *hành động (actions)* hoặc *đối tượng (objects)* trong lĩnh vực bài toán đang xét
- **Các liên kết (links)** là các quan hệ được gán nhãn và có chiều (directional and labeled) giữa các nút
- Hai kiểu liên kết: **kế thừa** và **cụ thể**
- **Liên kết kế thừa (Inheritance-oriented link)** biểu diễn:
  - Nút *A* là một lớp (loại) con của nút *B* (vd: liên kết *IS-A*)
  - Nút A là một ví dụ (instance) của nút *B* (vd: liên kết *INSTANCE-OF*)
- **Liên kết cụ thể (Domain-specific link)** biểu diễn:
  - Nút A liên quan tới (có quan hệ với) nút *B*
  - Ví dụ: *HAS*, *CAN*, *HAS-PART*, *CAUSES*, *HAS-COLOR*, …

Mạng ngữ nghĩa – Ví dụ (1)



(*B. L. Vrusias, course AI-CS289*)

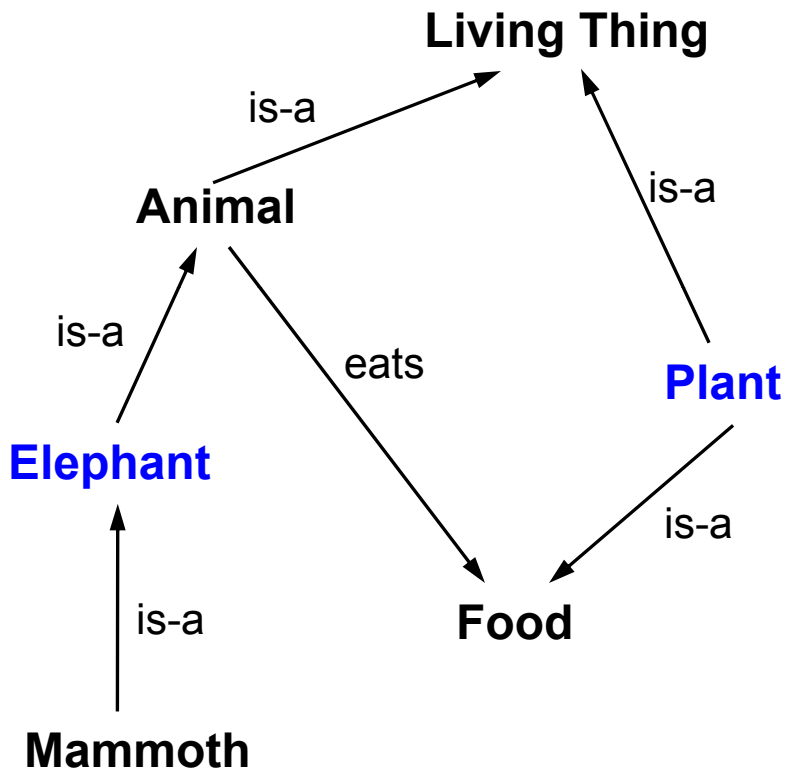
Mạng ngữ nghĩa – Ví dụ (2)



(*http://www.cwa.mdx.ac.uk/bis2040/johnlect.html*)

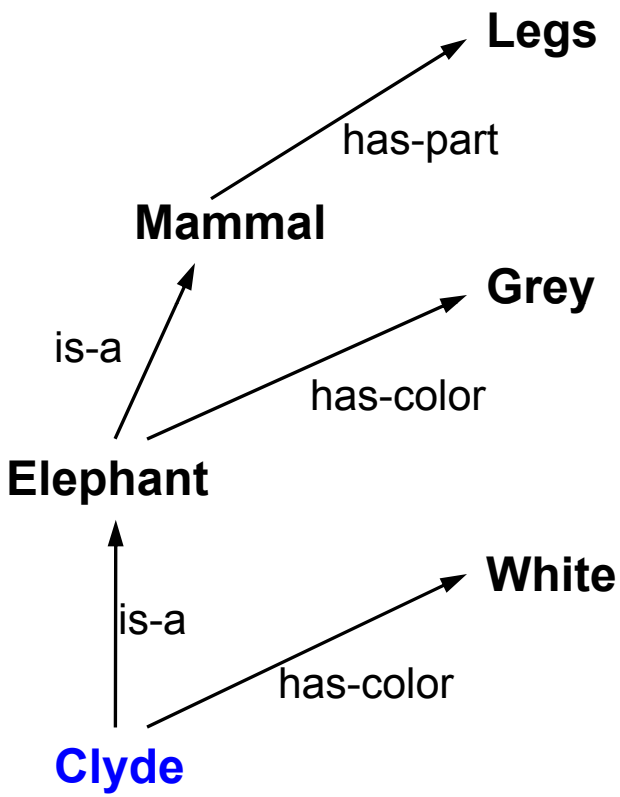
SN – Lan truyền tác động

- Đối với 2 khái niệm, việc lan truyền tác động (spreading activation) sẽ kích hoạt tác động từ khái niệm này tới khái niệm kia, hoặc theo cả 2 hướng
- Cho phép xác định các khái niệm "nằm giữa" liên quan đến cả 2 khái niệm đó
  - Ví dụ: Xét việc lan truyền tác động giữa 2 khái niệm "Elephant" và "Plant"



SN – Tính kế thừa

- Các thuộc tính (properties) của lớp (loại) cha được kế thừa cho các lớp (loại) con
- **Kế thừa toàn bộ (Universal inheritance)**: Tất cả các quan hệ được kế thừa
- **Kế thừa mặc định (Default inheritance)**: Các quan hệ được kế thừa, trừ khi có các thông tin mâu thuẫn (với nút cha) ở một nút con
- Các nghiên cứu tâm lý học chỉ ra rằng con người nhận thức "Clyde has-color White" nhanh hơn "Clyde has-part Legs"

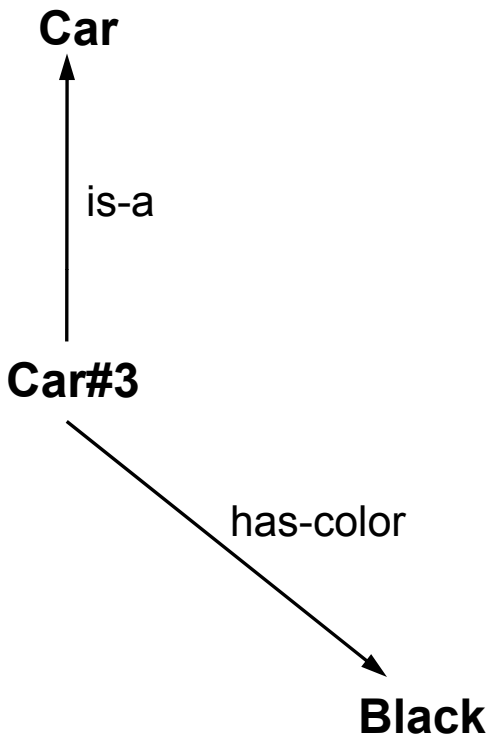


SN – Ngữ nghĩa (1)

- Biểu diễn bằng mạng ngữ nghĩa rất trực quan, và gần với nhận thức (cách biểu diễn) của con người
- Nhưng: Đối với *cùng một đồ thị (mạng) ngữ nghĩa*, các hệ thống khác nhau có thể có *các cách diễn giải (interpretations)khác nhau*
- Ngữ nghĩa (sematics) của các mạng ngữ nghĩa?
  - ”Since the semantics of any given language is dependent of the interpretation of the primitive elements....., the well-definedness of a network language rests heavily on the set of node and link types that it provides” (Brachman, p204, Readings in KR)

SN – Ngữ nghĩa (2)

- Ý nghĩa của mạng ngữ nghĩa sau là gì?
  - Thể hiện định nghĩa của một cái ô-tô màu đen (**Thông tin định nghĩa**)
  - Thể hiện rằng tồn tại một ô-tô màu đen (**Thông tin xác nhận**)
  - Thể hiện rằng một cái ô-tô cụ thể (Car#3) là màu đen (**Xác nhận sự tồn tại**)



SN – Ngữ nghĩa (3)

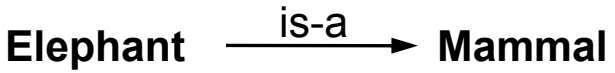
- Các liên kết có thể là...
  - **Liên kết xác nhận (Assertional links)**
    - Lưu giữ các thông tin về không gian bài toán đang xét
    - *Có thể thay đổi khi không gian bài toán thay đổi*
    - Ví dụ: "Jon hit Mary" (một *sự kiện* cụ thể đã xảy ra)
  - **Liên kết định nghĩa (Definitional links)**
    - Lưu giữ các ý nghĩa của các khái niệm
    - *Không thay đổi khi không gian bài toán thay đổi*
    - Ví dụ: "apple *is-a* fruit", "apple *has-color* red"

SN – Chuyển đổi sang logic (1)

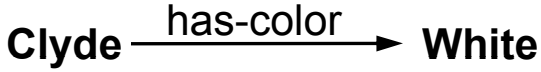
- Một giải pháp có thể đối với vấn đề xác định ngữ nghĩa của các mạng ngữ nghĩa: Chuyển đổi (transform) các mạng ngữ nghĩa sang logic
- Ý tưởng: Ngữ nghĩa của logic đã được định nghĩa chuẩn – Chuyển đổi ngữ nghĩa sang logic sẽ cho phép biểu diễn chính xác ngữ nghĩa của các mạng ngữ nghĩa
- Việc chuyển đổi chỉ đơn giản là biểu diễn các nút là các hằng (constants) và các liên kết là các vị từ hai ngôi (binary predicates)?
  - Không quá đơn giản như vậy
  - Tuy nhiên, cú pháp của các mạng ngữ nghĩa có thể được viết lại trong logic

SN – Chuyển đổi sang logic (2)

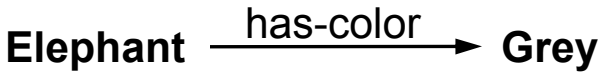
- Đối với các liên kết *is-a*
  - $\forall x$: elephant(x) $\rightarrow$ mammal(x)



- Đối với các liên kết (thuộc tính) của một ví dụ (instance)
  - hasColor(clyde, white)



- Đối với các liên kết (thuộc tính) của một lớp (class)
  - $\forall x$: elephant(x) $\rightarrow$ hasColor(x, grey)



- Xét việc suy diễn logic, nếu các phát biểu trên là đúng + *"elephant(clyde)"*?
  - Suy diễn mặc định (default reasoning) sẽ không còn đúng!

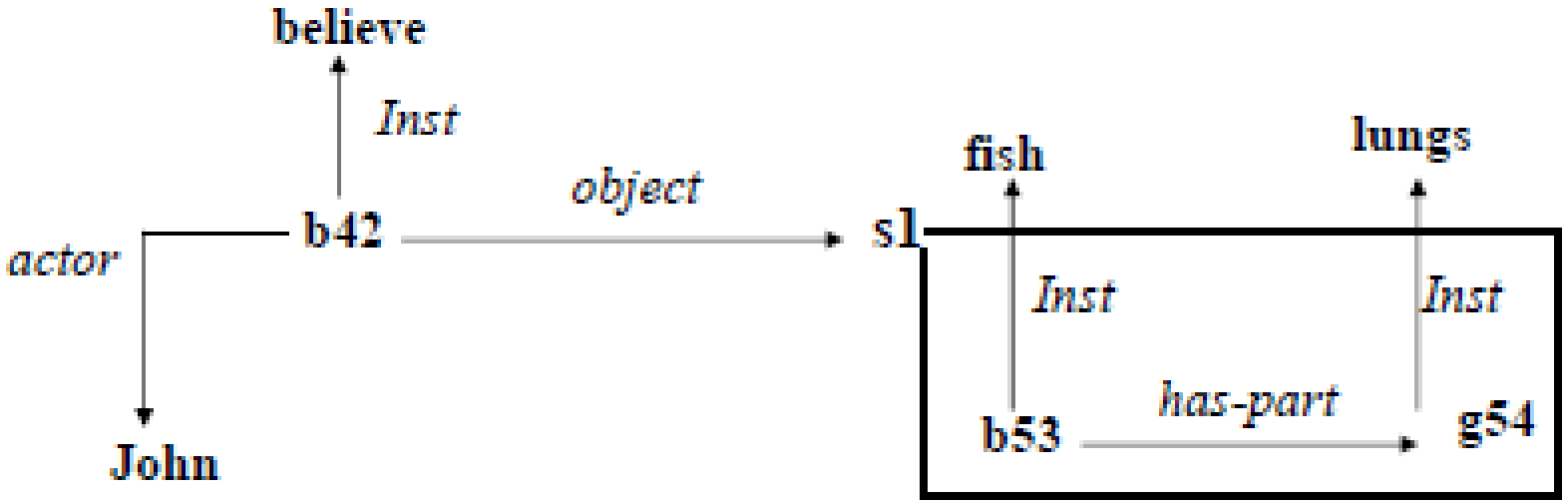
SN – Khả năng biểu diễn

- Chúng ta có thể biểu diễn các suy nghĩ (beliefs) của một người, mà không cần hệ thống phải xác nhận rằng các suy nghĩ này là đúng?
  - Ví dụ: "John believes that fish has lungs"
- Chúng ta có thể biểu diễn các lượng từ logic (∀,∃)?
  - Ví dụ: "Every bird has feathers"
- Trong cách biểu diễn các mạng ngữ nghĩa "chuẩn", các yêu cầu về biểu diễn nêu trên không được hỗ trợ

Các mạng được phân tách (1)

- Các mạng được phân tách (Partitioned networks) là sự cải tiến dựa trên cách biểu diễn SN chuẩn, được đề cử bởi Hendrix
- Cho phép biểu diễn các quan hệ giữa các phần đồ thị (sub-graphs) của mạng ngữ nghĩa

"*John believes that fish has lungs*"

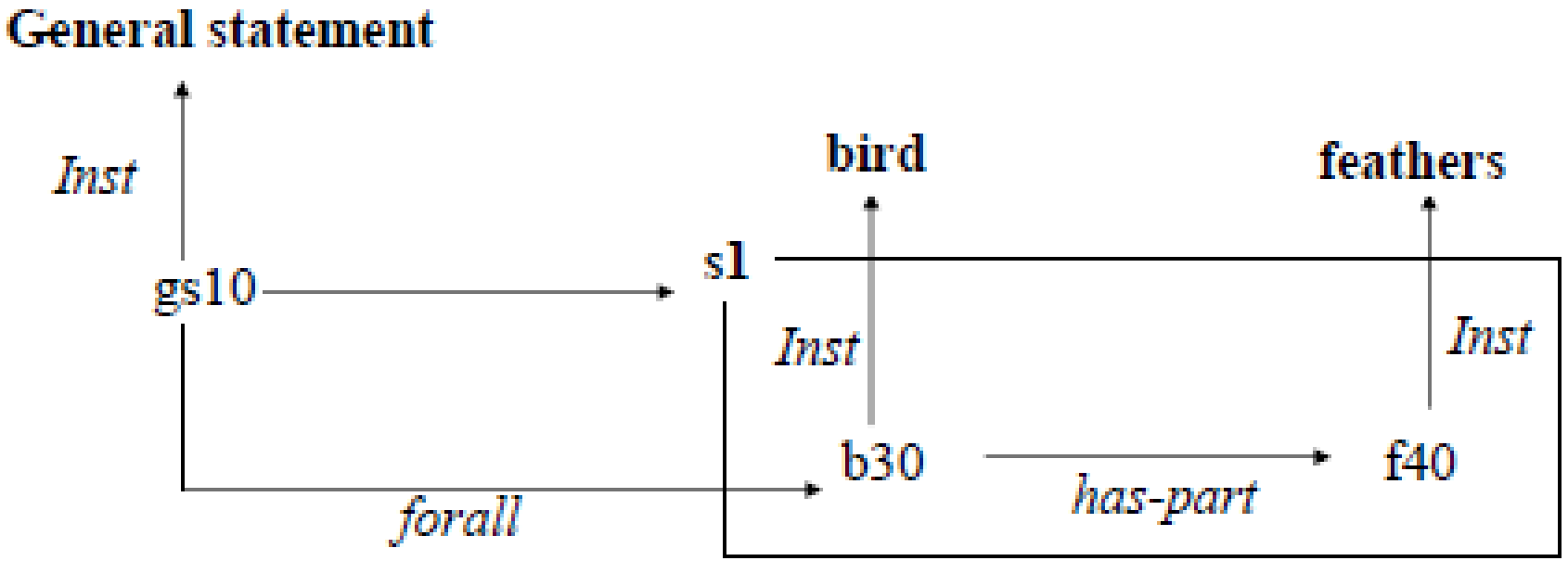


(*http://www.idi.ntnu.no/emner/it3706*)

Các mạng được phân tách (2)

- Cho phép mạng ngữ nghĩa phân tách các suy nghĩ (beliefs) và nhóm lại các xác nhận/định nghĩa
- Cho phép biểu diễn các lượng từ logic

"*Every bird has feathers*"



(*http://www.idi.ntnu.no/emner/it3706*)

Các mạng được phân tách – Vấn đề

- Không trực quan, rất khó đọc
- Tính trực quan của cách biểu diễn mạng ngữ nghĩa chuẩn đã bị giảm đi

- Ngữ nghĩa không rõ ràng của các mạng được phân tách
  - Ngữ nghĩa của các liên kết định nghĩa (definitional links) là gì?
  - Ngữ nghĩa của các liên kết xác nhận (Assertional links) là gì?

Mạng ngữ nghĩa – Ưu điểm

- Rõ ràng (trực quan) trong hiển thị, dễ hiểu đối với người dùng
  - SNs thường được sử dụng như là một công cụ trao đổi (làm việc) giữa các kỹ sư tri thức (knowledge engineers) và các chuyên gia (experts) trong giai đoạn thu thập tri thức
- SNs là rất phù hợp đối với các bài toán biểu diễn tri thức ở dạng phân cấp các khái niệm
  - Tri thức được phân loại (phân lớp) thành một cấu trúc phân cấp
- Cơ chế biểu diễn phân cấp của SNs hỗ trợ quá trình suy diễn nhanh chóng
- Hỗ trợ cơ chế suy diễn mặc định (default reasoning)
- Tập trung vào các thành phần chính của tri thức và liên kết giữa chúng

Mạng ngữ nghĩa – Nhược điểm

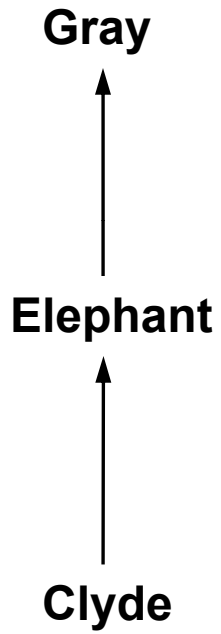
- Không tồn tại cách diễn dịch (interpretation) chung (chuẩn) – Ngữ nghĩa của các mạng ngữ nghĩa không được định nghĩa một cách chuẩn tắc
- Gặp vấn đề trong việc biểu diễn các thông tin tin phủ định (negation) và tuyển (disjunction)
  - Vd: "John **does not** go fishing", "John eats pizza **or** fish and chips"
- Khó khăn trong việc diễn đạt các suy nghĩ (beliefs) và các lượng từ (quantifiers)
  - Nếu hỗ trợ việc diễn đạt này, thì lại làm cho SNs trở nên khó đọc
- Khó khăn trong việc chọn các thành phần cơ bản (primitives) phù hợp
- Khả năng suy diễn hạn chế

Sự kế thừa – Phân loại phân cấp

- Phân loại một cách phân cấp (Hierarchical taxonomy) là một cách tự nhiên trong việc biểu diễn
  - Được sử dụng trong cả 2 cách biểu diễn tri thức: Khung (frames) và Mạng ngữ nghĩa (semantic networks)
- Sự quan trọng của việc **khái quát hóa (abstraction)** trong việc nhớ và suy diễn
  - Nhóm các đối tượng có cùng chung các thuộc tính
  - Không phải lặp lại các biểu diễn cho từng đối tượng riêng lẻ
- **Sự kế thừa (Inheritance)** là kết quả của việc suy diễn theo các hướng (paths) trong một cấu trúc phân cấp
  - *"A có kế thừa từ B không?"* tương đương với *"B có quan hệ bắc cầu kiểu is-a (subsumption) với A không?"*

Biểu diễn của sự kế thừa

- Các quan hệ ***IS***
  - Clyde is an Elephant, Elephant is Gray
- Suy diễn (reasoning) với các hướng (paths) và các kết luận (conclusions) mà các hướng đó biểu diễn
  - Các quan hệ bắc cầu
- Tính bắc cầu của các quan hệ
  - Clyde is Elephant
  - Elephant is Gray
  - Clyde is Gray

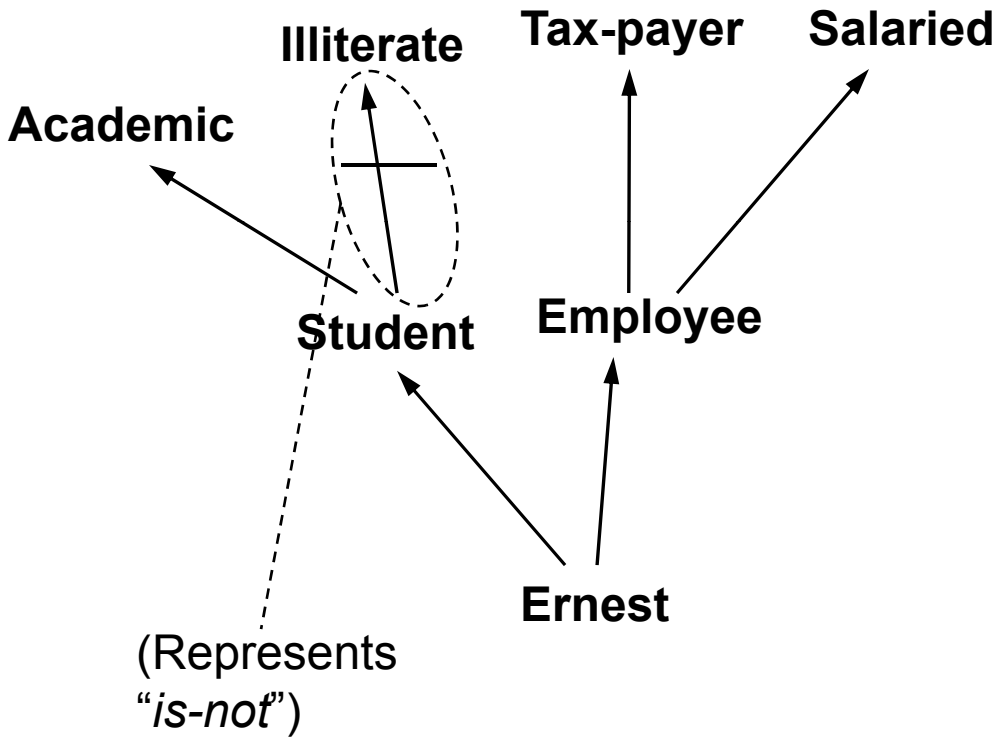


Mạng kế thừa (1)

- **Cấu trúc cây (tree structure) biểu diễn các quan hệ kế thừa**
  - Các kết luận được đưa ra, bằng cách áp dụng bắc cầu các quan hệ kế thừa đối với tất cả các hướng
  - Xét đến tất cả các nút có liên kết trong mạng

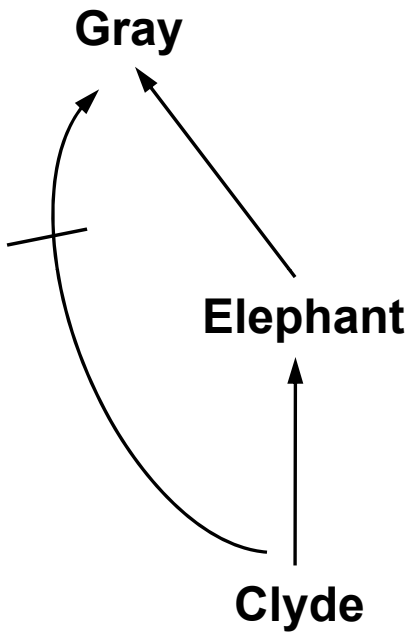
Mạng kế thừa (2)

- **Cấu trúc lưới (Lattice structure) biểu diễn các quan hệ kế thừa**
  - Một nút con có thể có nhiều nút cha (nhiều kế thừa)
- Giống như cấu trúc cây: Tất cả các kết luận được đưa ra theo tất cả các hướng



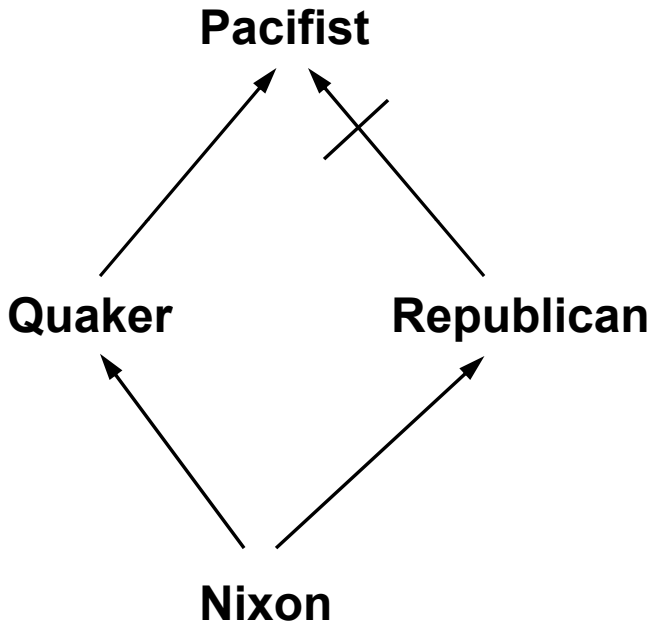
Mạng kế thừa (3)

- **Kế thừa có thể bị hủy bỏ (Defeasible inheritance)**
  - Giống như đối với cách biểu diễn tri thức bằng khung (frame-based representation)
  - Các thuộc tính không phải luôn luôn được kế thừa – chúng có thể *bị hủy bỏ (defeated)*
  - Các kết luận được xác định bằng cách duyệt từ nút hiện thời và chọn **phiên bản đầu tiên của thuộc tính** (**first version of property)** cần kết luận



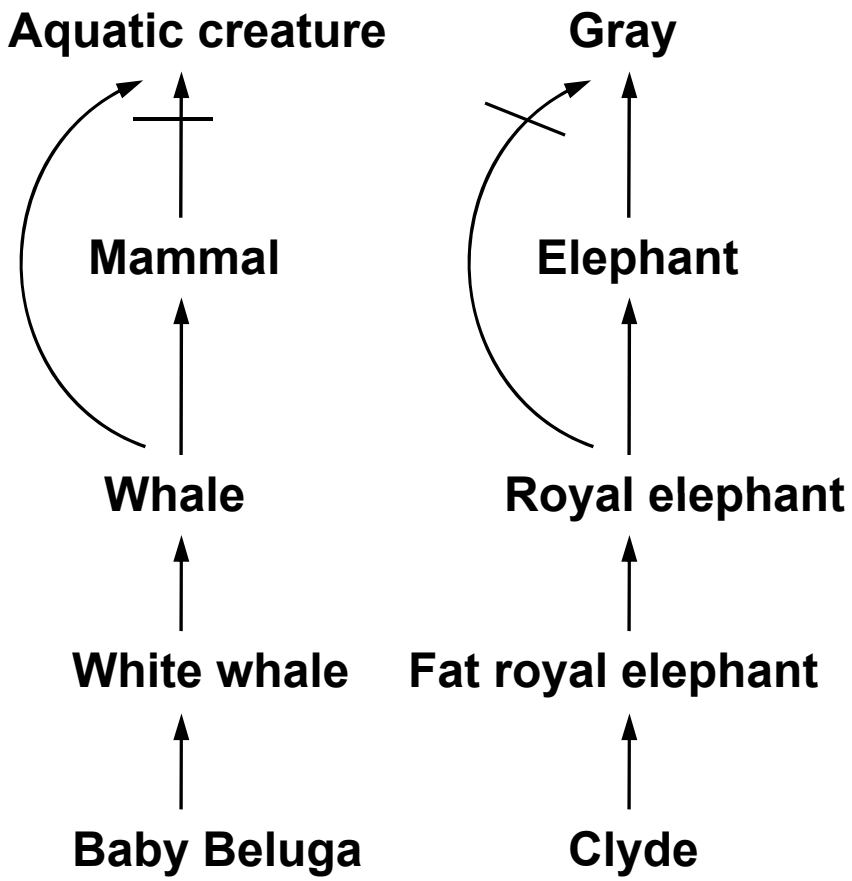
Mạng kế thừa (4)

- **Tính nhập nhằng trong kế thừa (Ambiguity in inheritance)**
- Hãy xét ví dụ sau đây:
  - Nixon là người yêu hòa bình (pacifist) hay không?



Mạng kế thừa (5)

- **Kế thừa có thể bị hủy bỏ (Defeasible inheritance)**
  - Các liên kết có cực (dương hoặc âm)
  - Sử dụng quy tắc **đường đi ngắn nhất** để xác định cực
  - Vì vậy, không phải tất cả các hướng đều sinh ra kết luận – Một số hướng (path) sẽ *bị ngăn chặn (pre-empted)*, nhưng một số hướng *được chấp nhận (admissible)*
  - Các hướng (paths) được xem như là các *lý lẽ (arguments)* đối với các kết luận



Phân cấp kế thừa – Định nghĩa (1)

- Một **phân cấp kế thừa (inheritance hierarchy)** G=<V,E> là một đồ thị có hướng và không chứa vòng lặp (directed and acyclic graph - DAG) với các cạnh dương và âm để biểu diễn tương ứng các quan hệ "*is-a*" và "*is-not-a*"
  - Các cạnh dương được biểu diễn là: $a \bullet x$
  - Các cạnh âm được biểu diễn là: $a \bullet \neg x$
- Một **hướng** (**path)** là một chuỗi các cạnh
  - Một **hướng dương (positive path)** là một chuỗi của một hoặc nhiều cạnh dương: $a \bullet \ldots \bullet x$
  - Một **hướng âm (negative path)** là một chuỗi của các cạnh dương và kết thúc bởi (chỉ) một cạnh âm: $a \bullet \ldots \bullet v \bullet \neg x$

Phân cấp kế thừa – Định nghĩa (2)

- Không tồn tại bất kỳ hướng (path) nào có nhiều hơn một cạnh âm
- Một hướng (path) có thể chẳng có cạnh dương nào
- Mỗi hướng biểu diễn một lý lẽ (argument) đối với một kết luận (được đưa ra theo hướng đó)
  - Hướng $a \bullet \ldots \bullet x$ biểu diễn kết luận "*a is an x*"
  - Hướng $a \bullet \ldots \bullet v \bullet \neg x$ biểu diễn kết luận "*a is not an x*"
- Một kết luận có thể được xác nhận bằng nhiều lý lẽ (đạt được bằng nhiều hướng)
- Tuy nhiên, không phải tất cả các lý lẽ (tất cả các hướng) đều có cùng mức độ tin cậy

Sự xác nhận và sự chấp nhận (1)

- Phân cấp phân loại G **xác nhận (supports) một hướng** $a \bullet s_1 \bullet \ldots \bullet s_n \bullet x$ nếu tập các cạnh tương ứng $\{a \bullet s_1 , s_1 \bullet s_2 , \ldots , s_{n-1} \bullet s_n , s_n \bullet x\}$ thuộc vào E, và hướng đó là chấp nhận được (admissible)
  - Định nghĩa tương tự, đối với một hướng âm $a \bullet s_1 \bullet \ldots \bullet s_n \bullet \neg x$
- Phân cấp phân loại G **xác nhận một kết luận "*a is x*"** nếu G xác nhận một hướng nào đó để đạt được kết luận
  - Định nghĩa tương tự, đối với G **xác nhận một kết luận "*a is not x*"**
- Một **hướng là chấp nhận được (admissible path)** nếu tất cả các cạnh của hướng đó đều là chấp nhận được (admissible edges)

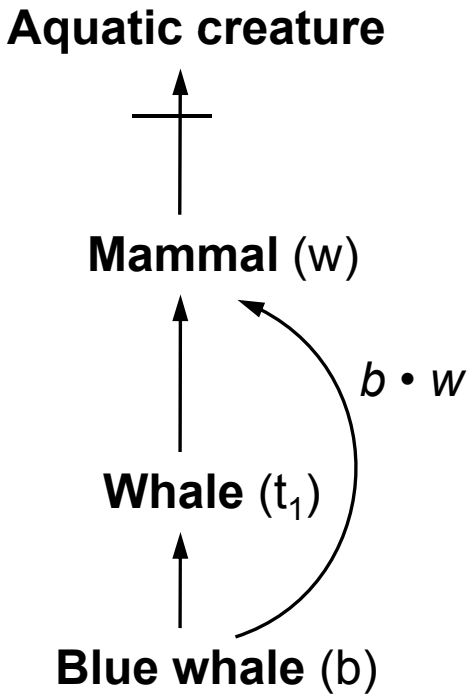
Sự xác nhận và sự chấp nhận (2)

- Một **cạnh dương** ***v*** • ***x*** **là chấp nhận được** trong G đối với một nút *a* nếu tồn tại một hướng dương *a* • s_1 • … • s_n • *v* (n ≥ 0) trong E và:
  1. (Định nghĩa truy hồi) Mỗi cạnh trong hướng *a* • s_1 • … • s_n • *v* đều là *chấp nhận được* trong G đối với nút *a*;
  2. *Không có* cạnh nào trong hướng *a* • s_1 • … • s_n • *v* là *dư thừa* (*redundant)* trong G đối với nút *a* (Định nghĩa về "dư thừa" sẽ được giải thích ở slide sau);
  3. *Không* có nút trung gian trong hướng *a*, s_1, …, s_n là một *nút ngăn chặn (pre-emptor)* của cạnh *v* • *x* đối với nút *a* (Định nghĩa "nút ngăn chặn" sẽ được giải thích ở slide sau)
- Định nghĩa tương tự, đối với một **cạnh âm** ***v*** • ¬***x*** **là chấp nhận được**

Sự dư thừa

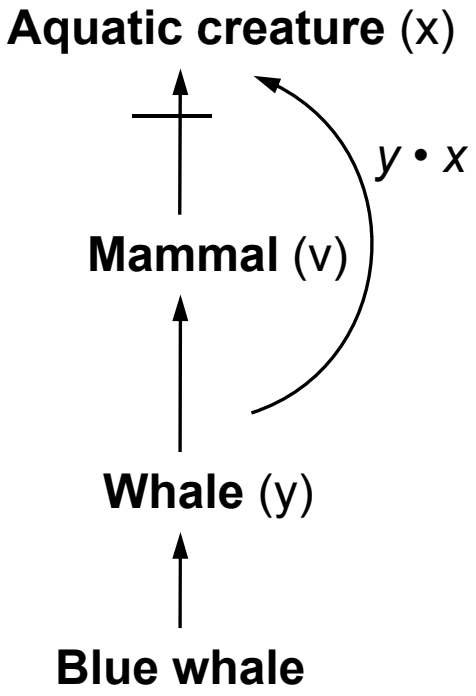
- Một **cạnh dương** ***b*** • ***w*** **là dư thừa (redundant)** trong G đối với nút *a* nếu tồn tại một hướng dương ($b \bullet t_1 \bullet \ldots \bullet t_m \bullet w$) ∈ E (m ≥ 1), sao cho:
  1. Mỗi cạnh của hướng $b \bullet t_1 \bullet \ldots \bullet t_m$ là chấp nhận được trong G đối với nút *a*;
  2. Không tồn tại *c* và t_i sao cho $c \bullet \neg t_i$ là chấp nhận được trong G đối với nút *a*;
  3. Không tồn tại *c* sao cho $c \bullet \neg w$ là chấp nhận được trong G đối với nút *a*

- Định nghĩa tương tự, đối với một **cạnh âm** ***b*** • ¬***w*** **là dư thừa**



Nút ngăn chặn

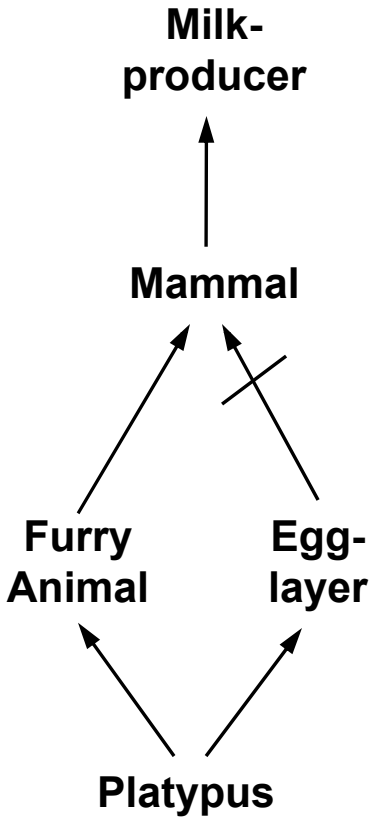
- Một nút *y* trong hướng *a* • ... • *y* • ... • *v* được gọi là **nút ngăn chặn (pre-emptor) của cạnh dương *v* • *x*** đối với nút *a* nếu $(y \bullet \neg x) \in E$
- Một nút *y* trong hướng *a* • ... • *y* • ... • *v* được gọi là **nút ngăn chặn (pre-emptor) của cạnh âm *v* • ¬*x*** đối với nút *a* nếu $(y \bullet x) \in E$
  - Xem ví dụ bên!

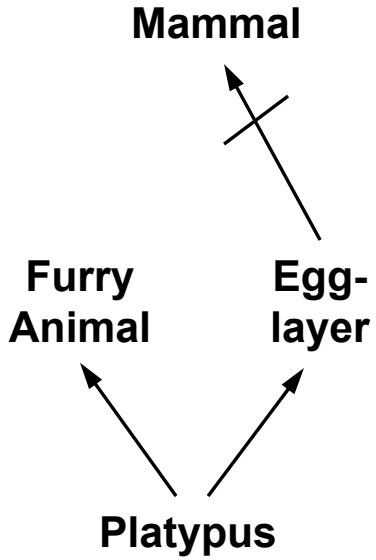


Sự mở rộng

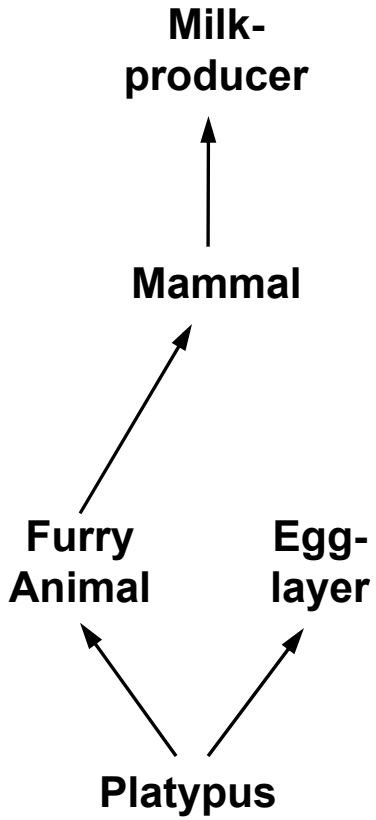
- Phân cấp phân loại G được gọi là ***a-connected*** nếu và chỉ nếu với mọi nút *x* trong G luôn tồn tại một hướng từ *a* đến *x*, và đối với mọi cạnh dương (hoặc âm) $v \bullet x$ (hoặc $v \bullet \neg x$) trong G luôn tồn tại một hướng dương từ *a* tới *v*
- Nói cách khác, mọi nút và mọi cạnh đều có thể đạt tới (reachable) từ *a*
- G được gọi là (có thể) **nhập nhằng (ambiguous)** đối với nút *a* nếu tồn tại một nút $x \in V$ sao cho cả 2 hướng $(a \bullet s_1 \bullet \ldots \bullet s_n \bullet x)$ và $(a \bullet t_1 \bullet \ldots \bullet t_m \bullet \neg x)$ đều thuộc G
- Một sự **mở rộng (extension)** của G đối với nút *a* là một cấu trúc con của G, sao cho cấu trúc con đó là *a-connected* và chứa tối thiểu (ít hơn) các nhập nhằng

Sự mở rộng – Ví dụ





Extension 1



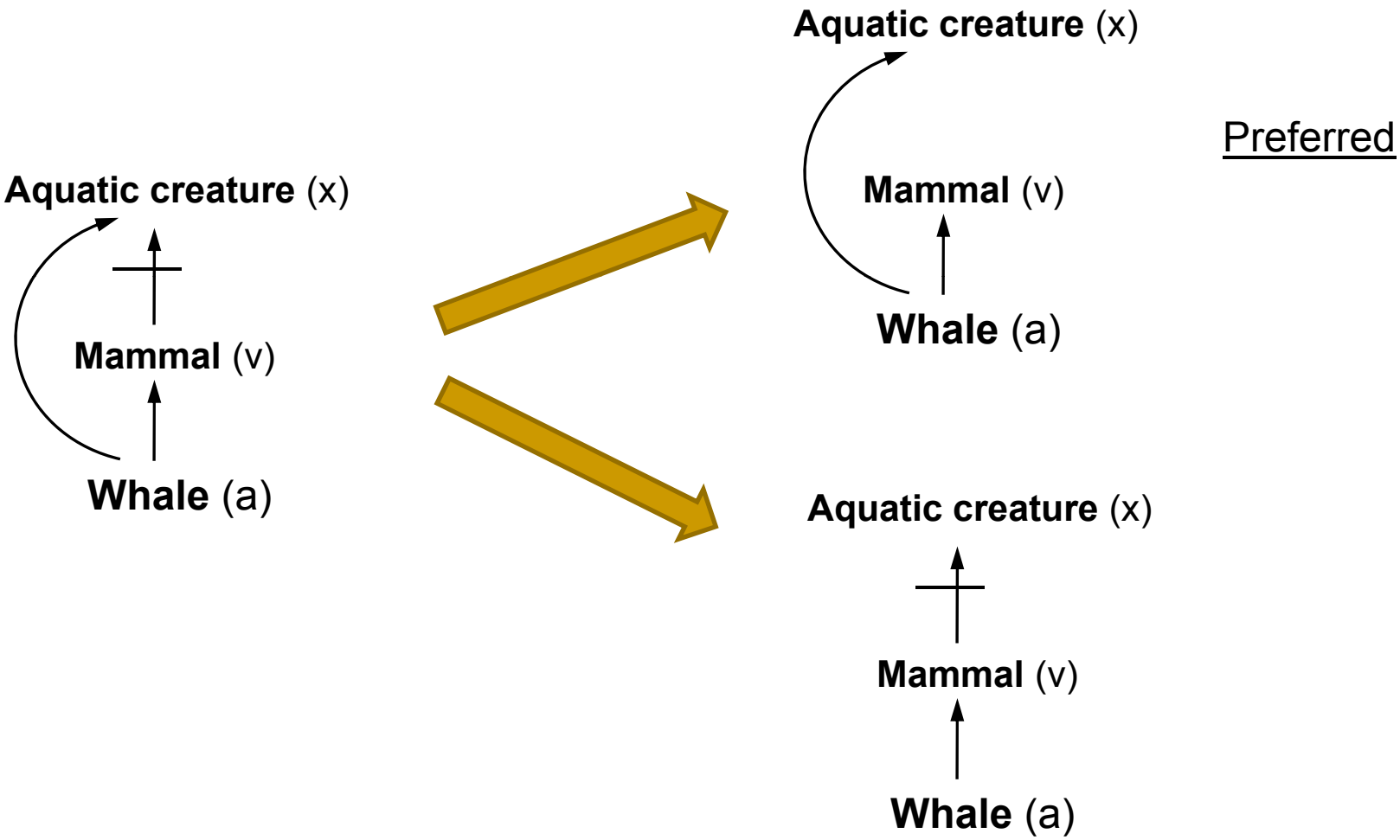
Extension 2

Sự mở rộng được ưu tiên

- Các mở rộng (extensions) không tính đến các khía cạnh về sự chấp nhận được (admissibility) hay sự ngăn chặn (pre-emption)

- Giả sử *X* và *Y* là các mở rộng của *G* đối với nút *a.* ***X được gọi là ưu tiên (preferred) so với Y*** nếu và chỉ nếu tồn tại các nút *v* và *x* sao cho:
  - *X* và *Y* là như nhau (về các kết luận) đối với tất cả các cạnh kết thúc ở nút trước *v*,
  - Tồn tại một cạnh dương $v \bullet x$ (hoặc cạnh âm $v \bullet \neg x$) là *không chấp nhận được (inadmissible)* trong G,
  - Cạnh không chấp nhận được này có trong *Y*, nhưng không có trong *X*

Sự mở rộng được ưu tiên – Ví dụ



Ontology

- Một **ontology** là một đặc tả (biểu diễn) hình thức và rõ ràng về các khái niệm
- Một **ontology** là một từ vựng dùng chung, được dùng để biểu diễn (mô hình) một lĩnh vực cụ thể
  - Các đối tượng và/hoặc các khái niệm
  - Các thuộc tính và các quan hệ của chúng
- Một ontology có thể được xem như là một cơ sở tri thức
- Một ontology có thể phục vụ các mục đích khác nhau
  - Ví dụ: Mô hình cơ sở dữ liệu (database schema) là một ontology
- Cơ sở tri thức chứa đựng các tri thức cụ thể cần thiết cho việc giải quyết vấn đề của một lĩnh vực

Ontology – Các động cơ thúc đẩy

- Các động cơ thúc đẩy về mặt công nghệ
  - Rất nhiều các hệ thống dựa trên tri thức sử dụng một ontology mô tả các tri thức của lĩnh vực liên quan
  - Việc xây dựng ontology là một nhiệm vụ rất tốn kém (về thời gian và công sức) trong quá trình phát triển một hệ thống dựa trên tri thức
  - Tại sao không giảm chi phí bằng cách chia sẻ các ontology?
    - Ví dụ: Các ontology "cơ bản" mô tả không gian, thời gian, số lượng, …
- Động cơ thúc đẩy về mặt khoa học
  - Để hiểu được các vấn đề nền tảng trong quá trình khái niệm hóa (nhận thức các khái niệm) của con người

Các khía cạnh của một ontology (1)

- Nội dung của ontology
  - Các kiểu đối tượng, các kiểu quan hệ
  - Ví dụ, bài toán sắp xếp các khối (blocks)
    - Các lớp đối tượng: Blocks, Robot Hands
    - Các thuộc tính: shapes of blocks, color of blocks
    - Các quan hệ: On, Above, Below, Grasp
    - Các quá trình: kế hoặc xây nên một tòa tháp
- Kiểu của ontology
  - Các quan hệ phân loại cơ bản (vd: instance-of, subclass) là gì?
  - Các định nghĩa của các khái niệm (và các ràng buộc đối với chúng)?
  - Khả năng diễn đạt của ngôn ngữ định nghĩa các khái niệm?
  - Hướng quá trình (process-centric) hay hướng đối tượng (object-centric)?

Các khía cạnh của một ontology (2)

- Mục đích sử dụng của ontology
  - Chia sẻ tri thức
    - Ví dụ: Giữa những người sử dụng, giữa các hệ thống, …
  - Sử dụng lại tri thức
    - Ví dụ: Sử dụng lại (một phần) tri thức khi các mô hình hoặc hệ thống thay đổi
  - Mục đích chung (tổng quát) hay cụ thể cho một lĩnh vực
- Việc xây dựng ontology
  - Ontology sẽ được thu thập hay xây dựng?
  - Nếu được thu thập, thì cần kiểm tra:
    - Chất lượng của tri thức?
    - Sự khác biệt về nội dung?
    - Sự tin cậy của tri thức?
    - Các khả năng sử dụng trong tương lai?

Xây dựng ontology

- Xác định giới hạn (phạm vi) và mục đích sử dụng
- Cân nhắc việc sử dụng lại các ontology đã có và liên quan đến lĩnh vực đang xét
- Liệt kê các khái niệm
- Định nghĩa sự phân loại
- Định nghĩa các thuộc tính
- Định nghĩa các khía cạnh (các ràng buộc)
- Xác định các ví dụ cụ thể (instances)
- Kiểm tra các bất thường (anomalies)

Phạm vi và mục đích sử dụng

- Không tồn tại một ontology "chuẩn" cho một lĩnh vực cụ thể
  - Một ontology là một sự khái quát hóa về một lĩnh vực cụ thể, và luôn tồn tại nhiều ontologies phù hợp
- Việc khái quát hóa này nên tính đến:
  - Việc sử dụng trong tương lai của ontology (mục đích sử dụng)
  - Các khả năng mở rộng (của ontology) có thể dự đoán trước
- Các câu hỏi cần phải được trả lời ở giai đoạn này:
  - Ontology này được dùng cho lĩnh vực nào?
  - Ontology này sẽ được dùng để làm gì?
  - Những kiểu câu hỏi nào mà ontology này có thể đưa ra câu trả lời?
  - Ai sẽ dùng và bảo trì ontology này?

Cân nhắc sử dụng lại các ontology

- Với sự phát triển rất nhanh chóng của Internet và của lĩnh vực Semantic Web, sẽ tồn tại (có) rất nhiều ontologies có thể khai thác
- Ít khi chúng ta phải bắt đầu việc xây dựng một ontology từ đầu (từ chỗ chưa có gì cả)
  - Gần như luôn tồn tại một số ontology có thể khai thác, hoặc là để sử dụng luôn, hoặc là để bắt đầu cho việc xây dựng ontology mong muốn

Liệt kê các khái niệm

- Xác định (theo một danh sách) tất cả các khái niệm liên quan sẽ xuất hiện trong ontology
  - Các danh từ (nouns) thường là cơ sở để xác định các tên lớp (class names)
  - Các động từ (verbs) hoặc cụm động từ (verb phrases) thường là cơ sở để xác định các tên thuộc tính (property names)
- Nên sử dụng các công cụ xây dựng tri thức, để thu được
  - Tập các khái niệm
  - Cấu trúc (phân cấp) ban đầu của các khái niệm này

Định nghĩa sự phân loại

- Các khái niệm liên quan cần được tổ chức lại với nhau thành một cấu trúc phân cấp phân loại (a taxonomic hierarchy)
  - Hai chiến lược thường được dùng: top-down vs. bottom-up
  - Cần cân nhắc đến hai yếu tố: sự tin cậy (chính xác) và hiệu quả (suy diễn)
- Đảm bảo rằng phân cấp thu được thực sự là một phân loại khái niệm
  - Nếu A là một lớp con của B, thì mọi ví dụ của A cũng phải là ví dụ của B

Định nghĩa các thuộc tính

- Thường được thực hiện kết hợp với bước trước (định nghĩa sự phân loại)
- Yêu cầu: Nếu *A* là một lớp con của *B*, thì mọi thuộc tính của *B* đều phải được áp dụng cho *A*
  - Nên gắn các thuộc tính với lớp cao nhất (phù hợp) trong cấu trúc phân cấp
- Khi xác định một thuộc tính cho một lớp, cần ngay lập tức xác định các thông tin về miền và giá trị của thuộc tính đó

Định nghĩa các ràng buộc

- Các ràng buộc về số chiều (số phần tử của một tập hợp)
- Các giá trị có thể được gán cho
- Các đặc điểm của quan hệ
  - Tính đối xứng, bắc cầu, đảo, …

Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

(*protege.stanford.edu/amia2003/AMIA2003Tutorial.ppt*)

Xác định các ví dụ

- Bổ sung vào ontology các ví dụ (instances) cụ thể
- Số lượng các ví dụ thường lớn hơn (nhiều) số lượng các lớp (classes)
- Vì vậy, việc bổ sung các ví dụ vào một ontology thường được làm thủ công (manually)
  - Lấy từ các nguồn dữ liệu sẵn có (vd: từ các cơ sở dữ liệu)
  - Trích tự động từ một tập các dữ liệu văn bản

Kiểm tra các bất thường

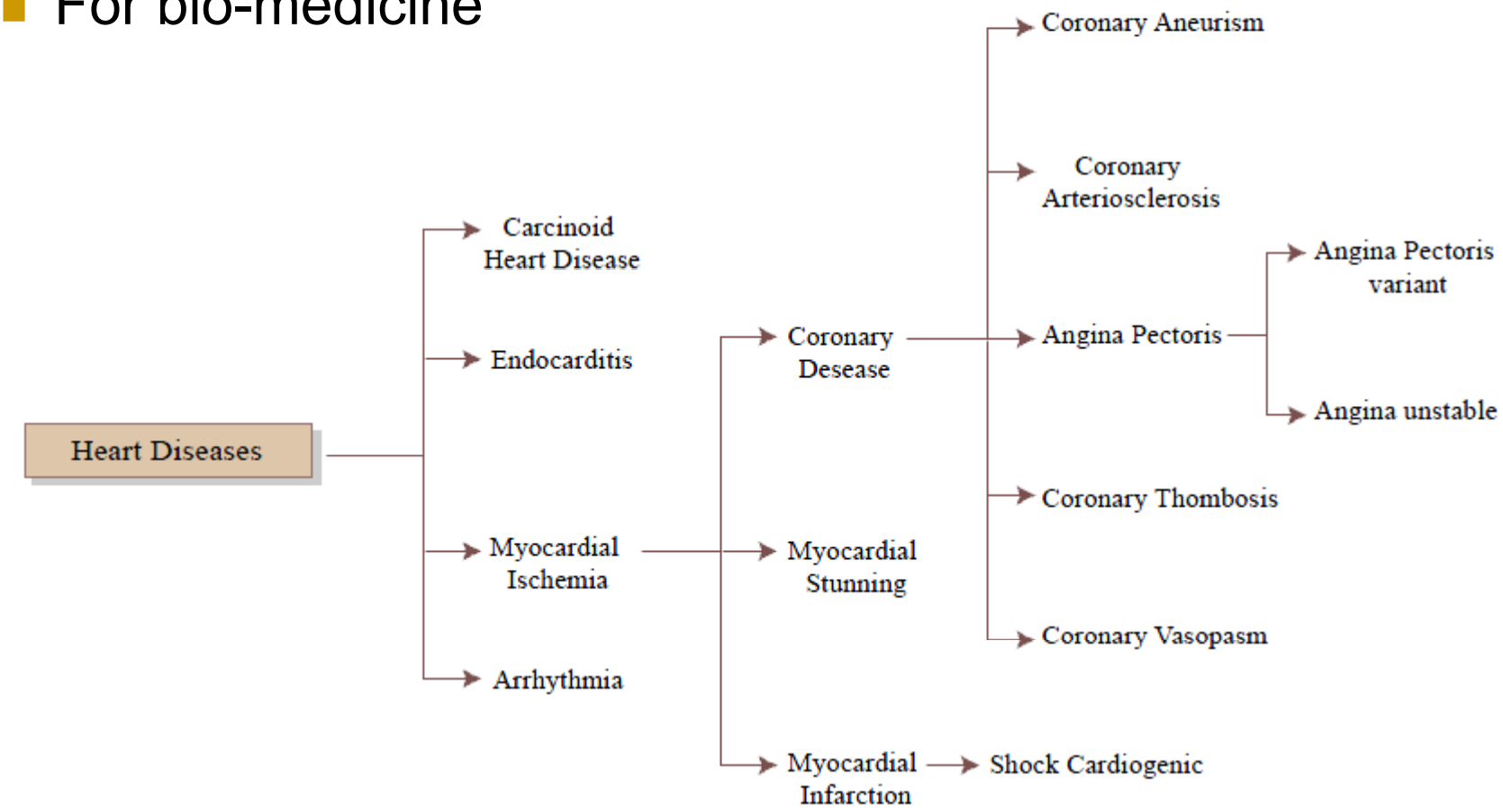
- Để kiểm tra các mâu thuẫn có thể có trong một ontology, (hoặc có trong một ontology đi kèm với các ví dụ)
- Các ví dụ của các mâu thuẫn
  - Không tương thích về miền, về giá trị, về các đặc tính bắc cầu, đối xứng, đảo,...
  - Không tương thích với ràng buộc về số chiều của một thuộc tính

Các ví dụ về ontology

- CYC (dùng chung, tổng quát)
  - 10^5 kiểu khái niệm, 10^6 tiên đề
- SENSUS (mở rộng của WordNet, dùng cho xử lý văn bản)
  - 70.000 khái niệm
- SUMO (tổng quát)
  - 1000 khái niệm, 4200 đánh giá
- UMLS (lĩnh vực y-sinh)
  - 135 kiểu ngữ nghĩa, 54 quan hệ ngữ nghĩa, 975.354 khái niệm
- WordNet (từ vựng)
  - 152.059 dạng từ, 115,424 cụm từ

UMLS ontology

- For bio-medicine



(*Figure by MIT OCW*)

CYC ontology

- Encode all of human common sense knowledge