

# CÁC MÔ HÌNH BIỂU DIỄN TRI THỨC

## 1.1 Biểu diễn tri thức bằng luật dẫn( luật sinh)

### 1.1.1 Khái niệm:

Phương pháp biểu diễn tri thức bằng luật sinh được phát minh bởi Newell và Simon trong lúc hai ông đang cố gắng xây dựng một hệ giải bài toán tổng quát. Đây là một kiểu biểu diễn tri thức có cấu trúc. Ý tưởng cơ bản là tri thức có thể được cấu trúc bằng một cặp điều kiện & hành động : "NẾU *điều kiện* xảy ra THÌ *hành động* sẽ được thi hành". Chẳng hạn : NẾU đèn giao thông là đỏ THÌ bạn không được đi thẳng, NẾU máy tính đã mở mà không khởi động được THÌ kiểm tra nguồn điện, v.v...

Ngày nay, các luật sinh đã trở nên phổ biến và được áp dụng rộng rãi trong nhiều hệ thống trí tuệ nhân tạo khác nhau. Luật sinh có thể là một công cụ mô tả để giải quyết các vấn đề thực tế thay cho các kiểu phân tích vấn đề truyền thống. Trong trường hợp này, các luật được dùng như là những chỉ dẫn (tuy có thể không hoàn chỉnh) nhưng rất hữu ích để trợ giúp cho các quyết định trong quá trình tìm kiếm, từ đó làm giảm không gian tìm kiếm. Một ví dụ khác là luật sinh có thể được dùng để bắt chước hành vi của những chuyên gia. Theo cách này, luật sinh không chỉ đơn thuần là một kiểu biểu diễn tri thức trong máy tính mà là một kiểu biểu diễn các hành vi của con người.

Một cách tổng quát luật sinh có dạng như sau:

$$P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$$

Tùy vào các vấn đề đang quan tâm mà luật sinh có những ngữ nghĩa hay cấu tạo khác nhau :

- Trong logic vị từ :  $P_1, P_2, \dots, P_n, Q$  là những biểu thức logic.
- Trong ngôn ngữ lập trình, mỗi một luật sinh là một câu lệnh.

IF ( $P_1$  AND  $P_2$  AND .. AND  $P_n$ ) THEN Q.

- Trong lý thuyết hiểu ngôn ngữ tự nhiên, mỗi luật sinh là một phép dịch:

ONE  $\rightarrow$  một.

TWO  $\rightarrow$  hai.

JANUARY  $\rightarrow$  tháng một.

Để biểu diễn một tập luật sinh, người ta thường phải chỉ rõ hai thành phần chính sau :

(1) Tập các sự kiện F(Facts)

$$F = \{ f_1, f_2, \dots, f_n \}$$

(2) Tập các quy tắc R (Rules) áp dụng trên các sự kiện dạng như sau :

$$f_1 \wedge f_2 \wedge \dots \wedge f_i \rightarrow q$$

Trong đó, các  $f_i$ ,  $q$  đều thuộc F

Ví dụ : Cho 1 cơ sở tri thức được xác định như sau :

- Các sự kiện : A, B, C, D, E, F, G, H, K
- Tập các quy tắc hay luật sinh (rule):

$$R1 : A \rightarrow E$$

$$R2 : B \rightarrow D$$

$$R3 : H \rightarrow A$$


$$R4 : E \wedge G \rightarrow C$$

$$R5 : E \wedge K \rightarrow B$$

$$R6 : D \wedge E \wedge K \rightarrow C$$

$$R7 : G \wedge K \wedge F \rightarrow A$$

### 1.1.2 Cơ chế suy luận trên các luật sinh

 **Suy diễn tiến** : là quá trình suy luận xuất phát từ một số sự kiện ban đầu, xác định các sự kiện có thể được "sinh" ra từ sự kiện này.

*Sự kiện ban đầu* : H, K


$R3 : H \rightarrow A \{ A, H, K \}$

$R1 : A \rightarrow E \{ A, E, H, K \}$

$R5 : E \wedge K \rightarrow B \{ A, B, E, H, K \}$

$R2 : B \rightarrow D \{ A, B, D, E, H, K \}$

$R6 : D \wedge E \wedge K \rightarrow C \{ A, B, C, D, E, H, K \}$

 **Suy diễn lùi** : là quá trình suy luận ngược xuất phát từ một số sự kiện ban đầu, ta tìm kiếm các sự kiện đã "sinh" ra sự kiện này. Một ví dụ thường gặp trong thực tế là xuất phát từ các tình trạng của máy tính, chẩn đoán xem máy tính đã bị hỏng hóc ở đâu.

Ví dụ :

Tập các sự kiện :

Ổ cứng là "hỏng" hay "hoạt động bình thường"

- Hỏng màn hình.
- Lỏng cáp màn hình.
- Tình trạng đèn ổ cứng là "tắt" hoặc "sáng"
- Có âm thanh đọc ổ cứng.
- Tình trạng đèn màn hình "xanh" hoặc "chớp đỏ"
- Không sử dụng được máy tính.
- Điện vào máy tính "có" hay "không".

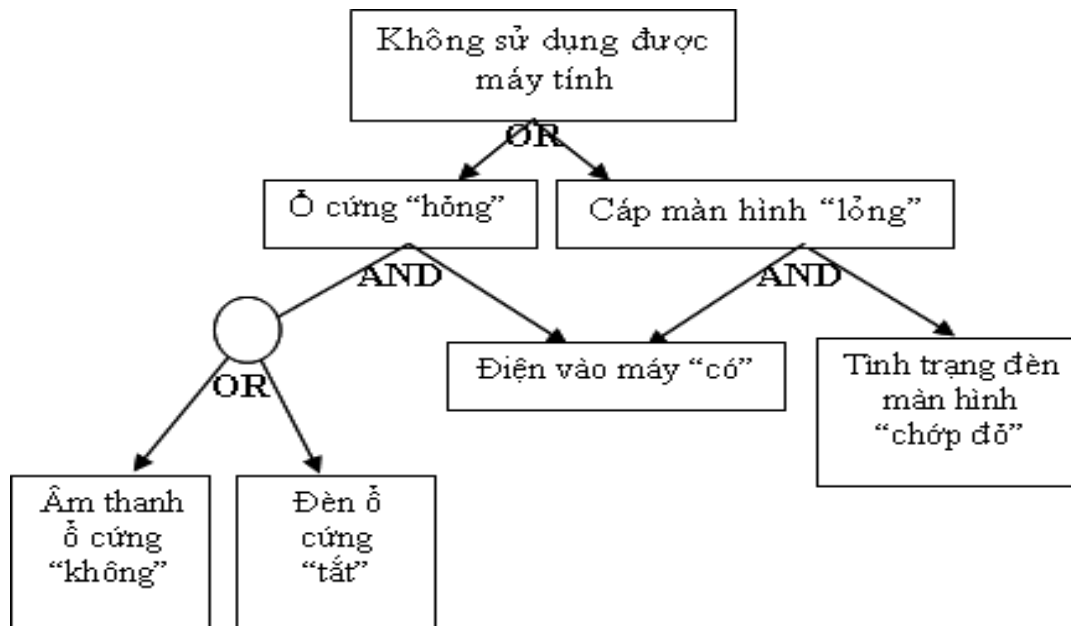
*Tập các luật :*

R1. Nếu ( ổ cứng "hỏng" ) hoặc ( cáp màn hình "lỏng" ) thì không sử dụng được máy tính.

R2. Nếu ( điện vào máy là "có" ) và ( âm thanh đọc ổ cứng là "không" ) hoặc tình trạng đèn ổ cứng là "tắt" ) thì ( ổ cứng "hỏng" ).

R3. Nếu ( điện vào máy là "có" ) và ( tình trạng đèn màn hình là "chớp đỏ" ) thì ( cáp màn hình "lỏng" ).

Để xác định được các nguyên nhân gây ra sự kiện "không sử dụng được máy tính", ta phải xây dựng một cấu trúc đồ thị gọi là đồ thị AND/OR như sau :



Hình 0-1: Cơ chế suy diễn của suy diễn lùi.

Như vậy là để xác định được nguyên nhân gây ra hỏng hóc là do ổ cứng hỏng hay cáp màn hình lỏng, hệ thống phải lần lượt đi vào các nhánh để kiểm tra các điều kiện như điện vào máy "có", âm thanh ổ cứng "không". Tại một bước, nếu giá trị cần xác định không thể được suy ra từ bất kỳ một luật nào, hệ thống sẽ yêu cầu người dùng trực tiếp nhập vào. Chẳng hạn như để biết máy tính có điện không, hệ thống sẽ hiện ra màn hình câu hỏi "*Bạn kiểm tra xem có điện vào máy tính không (kiểm tra đèn nguồn)? (C/K)*". Để thực hiện được cơ chế suy luận lùi, người ta thường sử dụng ngăn xếp (để ghi nhận lại những nhánh chưa kiểm tra).

### 1.1.3 Vấn đề tối ưu luật

Tập các luật trong một cơ sở tri thức rất có khả năng thừa, trùng lặp hoặc mâu thuẫn. Dĩ nhiên là hệ thống có thể *đổ lỗi* cho người dùng về việc đưa vào hệ thống những tri thức như vậy. Tuy việc tối ưu một cơ sở tri thức về mặt tổng quát là một thao tác khó (vì giữa các tri thức thường có quan hệ không tường minh), nhưng trong giới hạn cơ sở tri thức dưới dạng luật, ta vẫn có một số thuật toán đơn giản để loại bỏ các vấn đề này. .

#### 1.1.3.1 Rút gọn bên phải

Luật sau hiển nhiên đúng :

$$A \wedge B \rightarrow A \text{ (1)}$$

Do đó luật:

$$A \wedge B \rightarrow A \wedge C$$

Là hoàn toàn tương đương với

$$A \wedge B \rightarrow C$$

Quy tắc rút gọn : Có thể loại bỏ những sự kiện bên vế phải nếu những sự kiện đó đã xuất hiện bên vế trái. Nếu sau khi rút gọn mà vế phải trở thành rỗng thì luật đó là luật hiển nhiên. Ta có thể loại bỏ các luật hiển nhiên ra khỏi tri thức.

#### 1.1.3.2 Rút gọn bên trái

Xét các luật :

$$(L1) A, B \rightarrow C \text{ (L2) } A \rightarrow X \text{ (L3) } X \rightarrow C$$

Rõ ràng là luật  $A, B \rightarrow C$  có thể được thay thế bằng luật  $A \rightarrow C$  mà không làm ảnh hưởng đến các kết luận trong mọi trường hợp. Ta nói rằng sự kiện B trong luật (1) là dư thừa và có thể được loại bỏ khỏi luật dẫn trên.

#### 1.1.3.3 Phân rã và kết hợp luật:

Luật:  $A \wedge B \rightarrow C$

Tương đương với hai luật

$$A \rightarrow C$$

$$B \rightarrow C$$

Với quy tắc này, ta có thể loại bỏ hoàn toàn các luật có phép nối HOẶC. Các luật có phép nối này thường làm cho thao tác xử lý trở nên phức tạp.

#### 1.1.3.4 Luật thừa

Một luật dẫn  $A \rightarrow B$  được gọi là thừa nếu có thể suy ra luật này từ những luật còn lại.

Ví dụ : trong tập các luật gồm  $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$  thì luật thứ 3 là luật thừa vì nó có thể được suy ra từ 2 luật còn lại.

#### 1.1.3.5 Thuật toán tối ưu tập luật dẫn

Thuật toán này sẽ tối ưu hóa tập luật đã cho bằng cách loại đi các luật có phép nối HOẶC, các luật hiển nhiên hoặc các luật thừa.

Thuật toán bao gồm các bước chính:

##### **B1** : Rút gọn về phải

Với mỗi luật  $r$  trong  $R$

Với mỗi sự kiện  $A \in \text{VếPhải}(r)$

Nếu  $A \in \text{VếTrái}(r)$  thì Loại  $A$  ra khỏi vế phải của  $R$ .

Nếu  $\text{VếPhải}(r)$  rỗng thì loại bỏ  $r$  ra khỏi hệ luật dẫn :  $R = R \setminus \{r\}$

##### **B2** : Phân rã các luật

Với mỗi luật  $r : X_1 \vee X_2 \vee \dots \vee X_n \rightarrow Y$  trong  $R$

Với mỗi  $i$  từ 1 đến  $n$   $R := R + \{ X_i \rightarrow Y \}$

$R := R \setminus \{r\}$

##### **B3** : Loại bỏ luật thừa

Với mỗi luật  $r$  thuộc  $R$

Nếu  $\text{VếPhải}(r) \in \text{BaoĐóng}(\text{VếTrái}(r), R - \{r\})$  thì  $R := R \setminus \{r\}$

##### **B4** : Rút gọn về trái

Với mỗi luật dẫn  $r : X : A_1 \wedge A_2, \dots, A_n \rightarrow Y$  thuộc  $R$

Với mỗi sự kiện  $A_i \in r$

Gọi luật  $r_1 : X - A_i \rightarrow Y$

$S = (R - \{r\}) \cup \{r_1\}$

Nếu  $\text{BaoĐóng}(X - A_i, S) \equiv \text{BaoĐóng}(X, R)$  thì loại sự kiện  $A$  ra khỏi  $X$

### 1.1.4 Ưu điểm và nhược điểm của biểu diễn tri thức bằng luật

 **Ưu điểm:**

Biểu diễn tri thức bằng luật đặc biệt hữu hiệu trong những tình huống hệ thống cần đưa ra những hành động dựa vào những sự kiện có thể quan sát được. Nó có những ưu điểm chính yếu sau đây:

- Các luật rất dễ hiểu nên có thể dễ dàng dùng để trao đổi với người dùng (vì nó là một trong những dạng tự nhiên của ngôn ngữ).
- Có thể dễ dàng xây dựng được cơ chế suy luận và giải thích từ các luật.
- Việc hiệu chỉnh và bảo trì hệ thống là tương đối dễ dàng.
- Có thể cải tiến dễ dàng để tích hợp các luật mờ.
- Các luật thường ít phụ thuộc vào nhau.

#### **Nhược điểm:**

- Các tri thức phức tạp đôi lúc đòi hỏi quá nhiều (hàng ngàn) luật sinh. Điều này sẽ làm nảy sinh nhiều vấn đề liên quan đến tốc độ lẫn quản trị hệ thống.
- Thống kê cho thấy, người xây dựng hệ thống trí tuệ nhân tạo thích sử dụng luật sinh hơn tất cả phương pháp khác (dễ hiểu, dễ cài đặt) nên họ thường tìm mọi cách để biểu diễn tri thức bằng luật sinh cho dù có phương pháp khác thích hợp hơn! Đây là nhược điểm mang tính chủ quan của con người.
- Cơ sở tri thức luật sinh lớn sẽ làm giới hạn khả năng tìm kiếm của chương trình điều khiển. Nhiều hệ thống gặp khó khăn trong việc đánh giá các hệ dựa trên luật sinh cũng như gặp khó khăn khi suy luận trên luật sinh.

## **1.2 Mạng suy diễn tính toán**

### **1.2.1 Khái niệm:**

Mạng tính toán [ là một dạng biểu diễn tri thức có thể dùng biểu diễn các tri thức về các vấn đề tính toán và được áp dụng một cách có hiệu quả để giải một số dạng bài toán. *Mỗi mạng tính toán là một mạng ngữ nghĩa chứa các biến và những quan hệ có thể cài đặt và sử dụng được cho việc tính toán.* Chúng ta xét một mạng tính toán gồm một tập hợp các biến cùng với một tập các quan hệ (chẳng hạn các công thức) tính toán giữa các biến.

Trong ứng dụng cụ thể mỗi biến và giá trị của nó thường gắn liền với một khái niệm cụ thể về sự vật, mỗi quan hệ thể hiện một sự tri thức về sự vật.

### 1.2.2 Các quan hệ

Cho  $M = \{x_1, x_2, \dots, x_m\}$  là một tập hợp các biến có thể lấy giá trị trong các miền xác định tương ứng  $D_1, D_2, \dots, D_m$ . Đối với mỗi quan hệ  $R \subseteq D_1 \times D_2 \times \dots \times D_m$  trên các tập hợp  $D_1, D_2, \dots, D_m$  ta nói rằng quan hệ này liên kết các biến  $x_1, x_2, \dots, x_m$ , và ký hiệu là  $R(x_1, x_2, \dots, x_m)$  hay vắn tắt là  $R(x)$  (ký hiệu  $x$  dùng để chỉ bộ biến  $\langle x_1, x_2, \dots, x_m \rangle$ ). Ta có thể thấy rằng quan hệ  $R(x)$  có thể được biểu diễn bởi một ánh xạ  $f_{R,u,v}$  với  $u \cup v = x$ , và ta viết:  $f_{R,u,v} : u \rightarrow v$ , hay vắn tắt là  $f : u \rightarrow v$ .

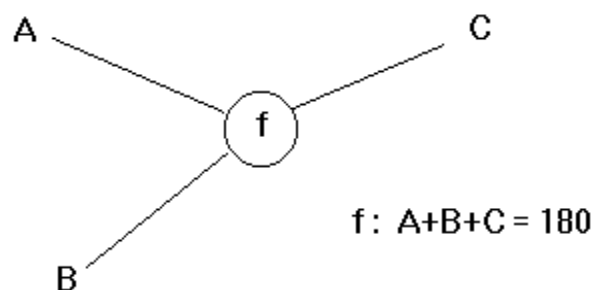
Đối với các quan hệ dùng cho việc tính toán, cách ký hiệu trên bao hàm ý nghĩa như là một hàm: ta có thể tính được giá trị của các biến thuộc  $v$  khi biết được giá trị của các biến thuộc  $u$ .

Trong phần sau ta xét các quan hệ xác định bởi các hàm có dạng  $f : u \rightarrow v$ , trong đó  $u \cap v = \emptyset$  (tập rỗng). Đặc biệt là các *quan hệ đối xứng* có hạng (rank) bằng một số nguyên dương  $k$ . Đó là các quan hệ mà ta có thể tính được  $k$  biến bất kỳ từ  $m-k$  biến kia (ở đây  $x$  là bộ gồm  $m$  biến  $\langle x_1, x_2, \dots, x_m \rangle$ ). Ngoài ra, trong trường hợp cần nói rõ ta viết  $u(f)$  thay cho  $u$ ,  $v(f)$  thay cho  $v$ . Đối với các quan hệ không phải là đối xứng có hạng  $k$ , không làm mất tính tổng quát, ta có thể giả sử quan hệ xác định duy nhất một hàm  $f$  với tập biến vào là  $u(f)$  và tập biến ra là  $v(f)$ ; ta gọi loại quan hệ này là *quan hệ không đối xứng xác định một hàm*, hay gọi vắn tắt là *quan hệ không đối xứng*.

Ví dụ: quan hệ  $f$  giữa 3 góc  $A, B, C$  trong tam giác  $ABC$  cho bởi hệ thức:

$$A+B+C = 180 \quad (\text{đơn vị: độ})$$





Hình 0-2: Quan hệ 3 góc trong tam giác ABC

### 1.2.3 Mạng tính toán và các kí hiệu

Như đã nói ở trên, ta sẽ xem xét các mạng tính toán bao gồm một tập hợp các biến  $M$  và một tập hợp các quan hệ (tính toán)  $F$  trên các biến. Trong trường hợp tổng quát có thể viết:

$$M = \{x_1, x_2, \dots, x_n\},$$

$$F = \{f_1, f_2, \dots, f_m\}.$$

Đối với mỗi  $f \in F$ , ta ký hiệu  $M(f)$  là tập các biến có liên hệ trong quan hệ  $f$ . Dĩ nhiên  $M(f)$  là một tập con của  $M$ :  $M(f) \subseteq M$ . Nếu viết  $f$  dưới dạng:

$$f: u(f) \rightarrow v(f)$$

thì ta có:  $M(f) = u(f) \cup v(f)$ .

### 1.2.4 Bài toán trên mạng suy diễn tính toán

Cho một mạng tính toán  $(M, F)$ ,  $M$  là tập các biến và  $F$  là tập các quan hệ. Giả sử có một tập biến  $A \subseteq M$  đã được xác định và  $B$  là một tập biến bất kỳ trong  $M$ .

**Các vấn đề đặt ra là:**

- ❖ Có thể xác định được tập B từ tập A nhờ các quan hệ trong F hay không? Nói cách khác, ta có thể tính được giá trị của các biến thuộc B với giả thiết đã biết giá trị của các biến thuộc A hay không?
- ❖ Nếu có thể xác định được B từ A thì quá trình tính toán giá trị của các biến thuộc B như thế nào?
- ❖ Trong trường hợp không thể xác định được B, thì cần cho thêm điều kiện gì để có thể xác định được B.

Bài toán xác định B từ A trên mạng tính toán (M,F) được viết dưới dạng:

$$A \rightarrow B,$$

trong đó A được gọi là giả thiết, B được gọi là mục tiêu tính toán của bài toán.

### 1.2.5 Ưu điểm & khuyết điểm của mạng suy diễn tính toán

#### Ưu điểm:

- Giải được hầu hết các bài toán GT  $\rightarrow$  KL nếu như đáp ứng đầy đủ các giả thiết cần thiết.
- Thuật toán đơn giản dễ cài đặt cho nên việc bảo trì hệ thống tương đối đơn giản.
- Có thể xây dựng hệ thống suy luận và giải thích một cách rõ ràng và dễ hiểu.

#### Khuyết điểm

- Do hệ thống chỉ bao gồm 1 cặp (M, F) để biểu diễn tri thức nên khi gặp phải những bài toán phức tạp thì có thể xảy ra việc lưu trữ khó khăn và nhập nhằng khi quản lý. Đồng thời việc xây dựng lại thuật toán là một việc tương đối khó khăn  $\rightarrow$  phải bảo trì lại toàn bộ hệ thống.
- Đối với các bài toán mà sử dụng nhiều các đối tượng tính toán bài toán trở nên phức tạp, việc giải quyết bài toán bằng mạng tính toán trở nên khó khăn cho người lập trình.

## 1.3 Biểu diễn tri thức bằng Frame

### 1.3.1 Khái niệm:

Frame là một cấu trúc dữ liệu chứa đựng tất cả những tri thức liên quan đến một đối tượng cụ thể nào đó. Frames có liên hệ chặt chẽ đến khái niệm hướng đối tượng (thực ra frame là nguồn gốc của lập trình hướng đối tượng). Ngược lại với các phương pháp biểu diễn tri thức đã được đề cập đến, frame "đóng gói" toàn bộ một đối tượng, tình huống hoặc cả một vấn đề phức tạp thành một thực thể duy nhất có cấu trúc. Một frame bao hàm trong nó một khối lượng tương đối lớn tri thức về một đối tượng, sự kiện, vị trí, tình huống hoặc những yếu tố khác. Do đó, frame có thể giúp ta mô tả khá chi tiết một đối tượng.

Dưới một khía cạnh nào đó, người ta có thể xem phương pháp biểu diễn tri thức bằng frame chính là nguồn gốc của ngôn ngữ lập trình hướng đối tượng. Ý tưởng của phương pháp này là *"thay vì bắt người dùng sử dụng các công cụ phụ như dao mở để đồ hộp, ngày nay các hãng sản xuất đồ hộp thường gắn kèm các nắp mở đồ hộp ngay bên trên vỏ lon. Như vậy, người dùng sẽ không bao giờ phải lo lắng đến việc tìm một thiết bị để mở đồ hộp nữa!"*. Cũng vậy, ý tưởng chính của frame (hay của phương pháp lập trình hướng đối tượng) là khi biểu diễn một tri thức, ta sẽ "gắn kèm" những thao tác thường gặp trên tri thức này. Chẳng hạn như khi mô tả khái niệm về hình chữ nhật, ta sẽ gắn kèm *cách tính chu vi, diện tích*.

Frame thường được dùng để biểu diễn những tri thức "chuẩn" hoặc những tri thức được xây dựng dựa trên những kinh nghiệm hoặc các đặc điểm đã được hiểu biết cặn kẽ. Bộ não của con người chúng ta vẫn luôn "lưu trữ" rất nhiều các tri thức chung mà khi cần, chúng ta có thể "lấy ra" để vận dụng nó trong những vấn đề cần phải giải quyết. Frame là một công cụ thích hợp để biểu diễn những kiểu tri thức này.

<p>Frame : <b>XE HƠI</b></p> <p><b>Thuộc lớp</b> : phương tiện vận chuyển.</p> <p>Tên nhà sản xuất : Audi</p> <p>Quốc gia của nhà sản xuất : Đức</p> <p>Model : 5000 Turbo</p> <p>Loại xe : Sedan</p> <p>Trọng lượng : 3300lb</p> <p>Số lượng cửa : 4 (default)</p> <p>Hộp số : 3 số tự động</p> <p>Số lượng bánh : 4 (default)</p> <p><b>Máy (tham chiếu đến frame Máy)</b></p> <p>Kiểu : In-line, overhead cam</p> <p>Số xy-lanh : 5</p> <p>Khả năng tăng tốc</p> <p>0-60 : 10.4 giây</p> <p>¼ dặm : 17.1 giây, 85 mph.</p>	<table><tr><td>Frame <b>MÁY</b></td></tr><tr><td>Xy-lanh : 3.19 inch</td></tr><tr><td>Tỷ lệ nén : 3.4 inche</td></tr><tr><td>Xăng :</td></tr><tr><td>TurboCharger</td></tr><tr><td>Mã lực : 140 hp</td></tr></table>	Frame <b>MÁY</b>	Xy-lanh : 3.19 inch	Tỷ lệ nén : 3.4 inche	Xăng :	TurboCharger	Mã lực : 140 hp
Frame <b>MÁY</b>							
Xy-lanh : 3.19 inch							
Tỷ lệ nén : 3.4 inche							
Xăng :							
TurboCharger							
Mã lực : 140 hp							

Hình 0-3: Cấu trúc một Frame xe hơi

### 1.3.2 Cấu trúc của Frame

Mỗi một frame mô tả một *đối tượng (object)*. Một frame bao gồm 2 thành phần cơ bản là slot và facet. Một slot là một thuộc tính đặc tả đối tượng được biểu diễn bởi frame. Ví dụ : trong frame mô tả xe hơi, có hai slot là *trọng lượng* và *loại máy*.

Mỗi slot có thể chứa một hoặc nhiều facet. Các facet (đôi lúc được gọi là slot "con") đặc tả một số thông tin hoặc thủ tục liên quan đến thuộc tính được mô tả bởi slot. Facet có nhiều loại khác nhau, sau đây là một số facet thường gặp.

- *Value (giá trị)* : cho biết *giá trị* của thuộc tính đó (như xanh, đỏ, tím vàng nếu slot là màu xe).
- *Default (giá trị mặc định)* : hệ thống sẽ tự động sử dụng giá trị trong facet này nếu slot là rỗng (nghĩa là chẳng có đặc tả nào!). Chẳng hạn trong frame về xe, xét slot về *số lượng bánh*. Slot này sẽ có giá trị 4. Nghĩa là, mặc định một chiếc xe hơi sẽ có 4 bánh!
- *Range (miền giá trị)* : (tương tự như kiểu biến), cho biết giá trị slot có thể nhận những loại giá trị gì (như số nguyên, số thực, chữ cái, ...)
- *If added*: mô tả một hành động sẽ được thi hành khi một giá trị trong slot được thêm vào (hoặc được hiệu chỉnh). Thủ tục thường được viết dưới dạng một script.
- *If needed* : được sử dụng khi slot không có giá trị nào. Facet mô tả một hàm để tính ra giá trị của slot.

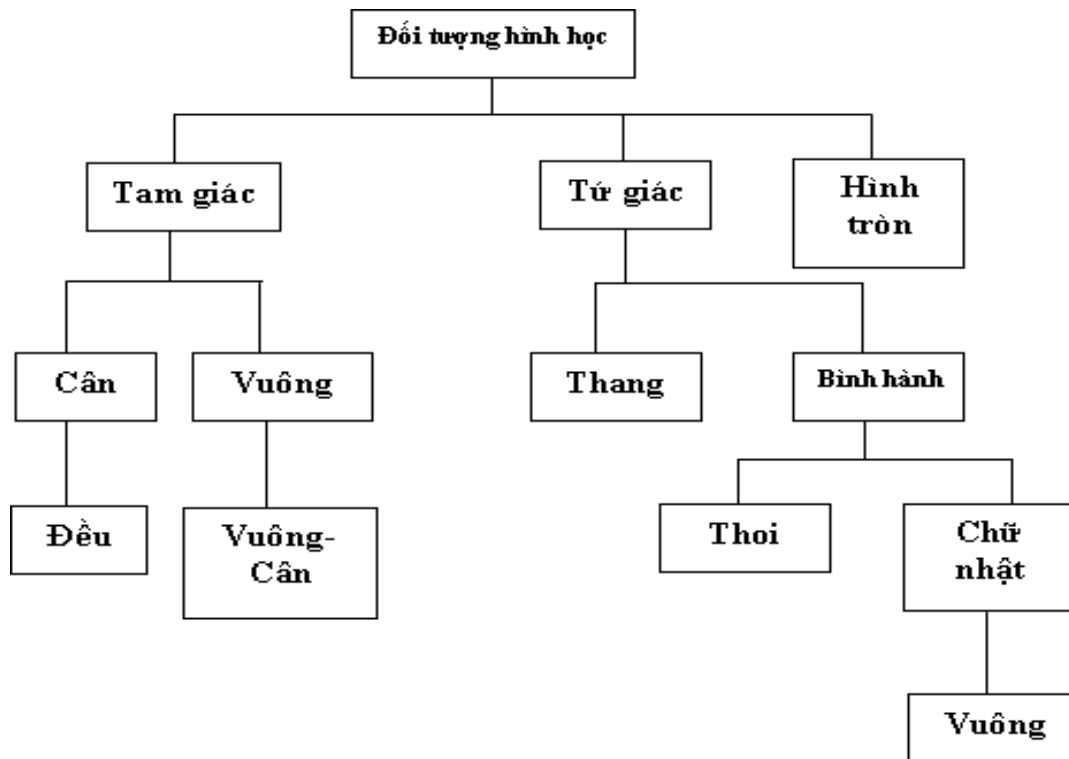
### 1.3.3 Tính kế thừa

cuu duong than cong. com

Trong thực tế, một hệ thống trí tuệ nhân tạo thường sử dụng nhiều frame được liên kết với nhau theo một cách nào đó. Một trong những điểm thú vị của frame là tính phân cấp. Đặc tính này cho phép kế thừa các tính chất giữa các frame.

Hình sau đây cho thấy cấu trúc phân cấp của các loại hình hình học cơ bản. Gốc của cây ở trên cùng tương ứng với mức độ trừu tượng cao nhất. Các frame nằm ở dưới cùng (không có frame con nào) gọi là lá. Những frame nằm ở mức thấp hơn có thể thừa kế tất cả những tính chất của những frame cao hơn.

Các frame cha sẽ cung cấp những mô tả tổng quát về thực thể. Frame có cấp càng cao thì mức độ tổng quát càng cao. Thông thường, frame cha sẽ bao gồm các *định nghĩa* của các thuộc tính. Còn các frame con sẽ chứa đựng giá trị thực sự của các thuộc tính này.



Hình 0-4: Quan hệ giữa các đối tượng hình học phẳng

## 1.4 Mô hình COKB

### 1.4.1 Định nghĩa về mô hình COKB

Mô hình biểu diễn tri thức COKB(Computational Objects Knowledge Base) [4] là một mô hình tri thức của các đối tượng tính toán. Mô hình COKB là một hệ thống gồm 6 thành phần chính được ký hiệu bởi bộ 6 như sau:

**(C,H,R,Opts, Funcs,Rules)**

#### 1.4.1.1 Tập hợp C (các khái niệm về các C\_Object):

Các khái niệm được xây dựng dựa trên các đối tượng. Mỗi khái niệm là một lớp các đối tượng tính toán có cấu trúc nhất định và được phân cấp theo sự thiết lập của cấu trúc đối tượng, bao gồm:

- **Các đối tượng (hay khái niệm) nền:** là các đối tượng (hay khái niệm) được mặc nhiên thừa nhận. Ví dụ: như một số đối tượng kiểu boolean (logic), số tự nhiên (natural), số nguyên (integer), số thực (real), tập hợp (set), danh sách (list) hay một số kiểu tự định nghĩa.
- **Các đối tượng cơ bản (hay khái niệm) cơ bản cấp 0:** có cấu trúc rỗng hoặc có cấu trúc thiết lập trên một số thuộc tính kiểu khái niệm nền: Các đối tượng(hay khái niệm) này làm nền cho các đối tượng(hay các khái niệm) cấp cao hơn. Ví dụ: đối tượng DIEM có kiểu mô tả không có cấu trúc thiết lập.
- **Các đối tượng (hay khái niệm) cấp 1:** Các đối tượng này chỉ có các thuộc tính kiểu khái niệm nền và có thể được thiết lập trên một danh sách nền các đối tượng cơ bản. Ví dụ: đối tượng DOAN[A,B] trong đó A, B là các đối tượng cơ bản loại DIEM, thuộc tính a biểu thị độ dài đoạn thẳng có kiểu tương ứng là “real”.
- **Các đối tượng (hay khái niệm) cấp 2:** Các đối tượng này có các thuộc tính kiểu khái niệm nền và các thuộc tính loại đối tượng cấp 1, có thể được thiết lập trên một danh sách nền các đối tượng cơ bản. Ví dụ: đối tượng TAMGIAC[A,B,C] trong đó A, B, C là các đối tượng cơ bản loại DIEM, các thuộc tính như GocA, a, S có kiểu tương ứng là “GOC[C,A,B]”, “DOAN[B,C]”, “real”.
- **Các đối tượng (hay khái niệm) cấp  $n > 0$ :** Các đối tượng này có các thuộc tính kiểu khái niệm nền và các thuộc tính loại đối tượng cấp thấp hơn, có thể được thiết lập trên một danh sách nền các đối tượng cấp thấp hơn.

### Cấu trúc bên trong của mỗi lớp đối tượng:

- Kiểu đối tượng: Kiểu này có thể là kiểu thiết lập trên một danh sách nền các đối tượng cấp thấp hơn.
- Danh sách các thuộc tính của đối tượng: Mỗi thuộc tính có kiểu thực, kiểu đối tượng cơ bản hay kiểu đối tượng cấp thấp hơn. Phân ra làm 2 loại là tập các thuộc tính thiết lập của đối tượng và tập các thuộc tính khác (còn gọi là tập thuộc tính).

- Tập hợp các điều kiện ràng buộc trên các thuộc tính.
- Tập hợp các tính chất nội tại hay sự kiện vốn có liên quan đến các thuộc tính của đối tượng.
- Tập hợp các quan hệ suy diễn - tính toán trên các thuộc tính của đối tượng. Các quan hệ này thể hiện các luật suy diễn và cho phép ta có thể tính toán một hay một số thuộc tính từ các thuộc tính khác của đối tượng.
- Tập hợp các luật suy diễn trên các loại sự kiện khác nhau liên quan đến các thuộc tính của đối tượng hay bản thân đối tượng. Mỗi luật suy diễn có dạng: {các sự kiện giả thiết}  $\Rightarrow$  {các sự kiện kết luận}.

#### 1.4.1.2 Mô hình cho một đối tượng tính toán (C-Object)

Một C-Object có thể được mô hình hóa bởi một bộ 6 thành phần chính:

**(BasicO, Attrs, CRela, Rules, Prop, Cons)**

Trong đó:

- BasicO: là tập hợp các đối tượng nền của một đối tượng.
- Attrs: là tập hợp các thuộc tính của đối tượng.
- CRela: là tập hợp các quan hệ suy diễn tính toán.
- Rules: là tập hợp các luật suy diễn trên các sự kiện liên quan đến các thuộc tính cũng như liên quan đến bản thân đối tượng.
- Prop: là tập hợp các tính chất hay sự kiện vốn có của đối tượng.
- Cons: là tập hợp các điều kiện ràng buộc.

#### 1.4.1.3 Tập hợp H (các quan hệ phân cấp giữa các đối tượng)

Trong tập C, ta có các quan hệ mà theo đó có thể có những khái niệm là sự đặc biệt hoá của những khái niệm khác. Có thể nói, H là một biểu đồ Hasse trên C khi xem quan hệ phân cấp là một quan hệ thứ tự trên C.

Cấu trúc của một quan hệ phân cấp:



**[<tên lớp đối tượng cấp cao>, <tên lớp đối tượng cấp thấp> ]**

#### 1.4.1.4 Tập hợp R các khái niệm về các loại quan hệ trên các C-Object

Mỗi quan hệ được xác định bởi tên quan hệ và danh sách các loại đối tượng của quan hệ. Đối với quan hệ 2 hay 3 ngôi thì quan hệ có thể có các tính chất như tính phản xạ, tính phản xứng, tính đối xứng và tính bắc cầu.

Cấu trúc của một quan hệ:

**[ < tên quan hệ > , < loại đối tượng > , < loại đối tượng > , ... ] , { < tính chất > , < tính chất > }.**

#### 1.4.1.5 Tập hợp Opts các toán tử

Các toán tử thể hiện các qui tắc tính toán nhất định trên các biến thực cũng như trên các đối tượng. Chẳng hạn như các phép toán số học, các phép tính toán trên các đối tượng đoạn, góc tương tự như đối với các biến thực hay các phép tính toán vectơ, tính toán ma trận,... Trong trường hợp các phép toán 2 ngôi thì phép toán có thể có các tính chất như tính giao hoán, tính kết hợp, tính nghịch đảo, tính trung hoà.

#### 1.4.1.6 Tập hợp Funcs các hàm

Tập hợp Funcs trong mô hình COKB thể hiện tri thức về các hàm hay nói cách khác là thể hiện tri thức về các khái niệm và các qui tắc tính toán trên các biến thực cũng như trên các loại C-Object, được xây dựng thông qua các quan hệ tính toán dạng hàm. Mỗi hàm được xác định bởi <tên hàm>, danh sách các đối số và một qui tắc định nghĩa hàm về phương diện toán học.

#### 1.4.1.7 Tập hợp Rules các luật

Mỗi luật cho ta một qui tắc suy luận để từ các sự kiện đang biết suy ra được các sự kiện mới thông qua việc áp dụng các định luật, định lý hay các qui tắc tính toán nào đó. Mỗi luật suy diễn  $r$  có thể được mô hình hoá dưới dạng :

$$r : \{sk_1, sk_2, \dots, sk_m\} \Rightarrow \{sk_{m+1}, sk_{m+2}, \dots, sk_n\}.$$

Cấu trúc của một luật:

[ **Kind**, **BasicO**, **Hypos**, **Goals**]

Trong đó:

- **Kind**: loại luật.
- **BaseO**: tập các đối tượng cơ bản.
- **Hypos**: tập các sự kiện giả thiết của một luật.
- **Goals**: tập các sự kiện kết luận của một luật.

#### 1.4.2 Tổ chức cơ sở tri thức theo COKB

Cơ sở tri thức được tổ chức bởi một hệ thống tập tin văn bản có cấu trúc dựa trên một số từ khoá và qui ước về cú pháp, thể hiện các thành phần trong mô hình tri thức COKB. Hệ thống này bao gồm các tập tin như sau:

- Tập tin **OBJECT.txt** : Lưu trữ tất cả các khái niệm đối tượng của cơ sở tri thức.
- Tập tin **HIERARCHY.txt**: Lưu lại các biểu đồ Hasse thể hiện quan hệ phân cấp đặc biệt hoá giữa các loại đối tượng C-Object.
- Tập tin **RELATIONS.txt**: Lưu trữ tất cả các quan hệ cũng như các tính chất giữa các loại đối tượng C-Object.
- Tập tin **OPERATORS.txt**: lưu trữ các thông tin, cơ sở tri thức của thành phần toán tử trên các đối tượng C-Object.

- Tập tin **OPERATORS\_DEF.txt**: Lưu trữ định nghĩa về các loại toán tử hay định nghĩa của các thủ tục tính toán phục vụ toán tử.
- Tập tin **RULES.txt**: Lưu trữ các hệ luật trên các loại đối tượng và các sự kiện trong cơ sở tri thức.
- Tập tin **FUNCTIONS.txt**: Lưu trữ cách khai báo hàm, thông tin về hàm trên các C-Object.
- Tập tin **FUNCTIONS\_DEF.txt**: Lưu trữ định nghĩa về các hàm trên các đối tượng và các sự kiện.
- Các tập tin có tên **<tên các C-OBJECT>.txt**: Lưu trữ cấu trúc của đối tượng <tên khái niệm C-Object>.

## Cấu trúc của các tập tin lưu trữ các thành phần trong mô hình COKB

### 1.4.2.1 Cấu trúc tập tin “OBJECTS.txt”

```
begin_objects
    <tên loại đối tượng 1>
    <tên loại đối tượng 2>
    -----
end_objects
```

### 1.4.2.2 Cấu trúc tập tin “RELATIONS.txt”

```
begin_relations
    [<tên quan hệ 1>,<tên loại đối tượng 1>,<tên loại đối tượng 2>,...],{“<tính chất 1>”,“<tính chất 2>”,...}
    [<tên quan hệ 2>,<tên loại đối tượng 1>,<tên loại đối tượng 2>,...],{“<tính chất 1>”,“<tính chất 2>”,...}
    -----
end_relations
```

#### 1.4.2.3 Cấu trúc tập tin “HIERARCHY.txt”

begin\_hierarchy

[<tên lớp đối tượng cấp cao>,<tên lớp đối tượng cấp thấp>]

[<tên lớp đối tượng cấp cao>,<tên lớp đối tượng cấp thấp>]

-----

end\_hierarchy

#### 1.4.2.4 Cấu trúc tập tin “OPERATORS.txt”

begin\_operators

[<toán tử 1>,<kiểu kết quả>,<kiểu toán hạng 1>,<kiểu toán hạng 2>,...]

[<toán tử 2>,<kiểu kết quả>,<kiểu toán hạng 1>,<kiểu toán hạng 2>,...]

[<toán tử 3>,<kiểu kết quả>,<kiểu toán hạng 1>,<kiểu toán hạng 2>,...]

-----

end\_operators

cuu duong than cong. com

#### Cấu trúc tập tin “OPERATORS\_DEF.txt”

begin\_operators\_def

begin\_define\_operator: <toán tử 1(ký hiệu)>(<toán hạng 1>,<toán hạng 2>,...)

<các tên toán hạng> : <kiểu toán hạng>

<các tên toán hạng> : <kiểu toán hạng>

-----

return <kiểu đối tượng trả về>

begin\_proc

<các qui tắc tính toán> hay <thủ tục tính toán>

end\_proc

end\_operators\_def

cuu duong than cong. com

### Cấu trúc tập tin “FUNCTIONS.txt”

begin\_functions

<kiểu trả về của hàm 1> <tên hàm 1>(<loại của đối số 1>,....) {tính chất của hàm}

<kiểu trả về của hàm 2> <tên hàm 2>(<loại của đối số 2>,....) {tính chất của hàm}

-----

end\_functions

### Cấu trúc tập tin “<FUNCTIONS\_DEF.txt>”

begin\_functions

begin\_function 1: <tên hàm>(<đối số 1>,....)

<các tên đối số > : <kiểu của đối số>

<các tên đối số > : <kiểu của đối số>

-----

result <đối tượng>: kiểu của đối tượng trả về

begin\_proc

end\_proc

properties

<các qui tắc tính toán> hay <thủ tục tính toán>

end\_properties

end\_function

begin\_function 2

-----

end\_function

-----

end\_functions

### Cấu trúc tập tin “<RULES.txt>”

begin\_rules

begin\_rule 1

kind\_rule = “<loại luật>”

<các tên đối tượng > : <kiểu đối tượng>;

<các tên đối tượng > : <kiểu đối tượng>;

-----

hypothesis\_part:

{các sự kiện giả thiết của luật}

end\_hypothesis\_part

goal\_part:

{các sự kiện kết luận}

end\_goal\_part

end\_rule

begin\_rule 2

-----

end\_rule

-----

end\_rules

### Cấu trúc tập tin “<Tên khái niệm C-Object>.txt”

begin\_object: <tên khái niệm C-Object>[<đối tượng nền 1>,<đối tượng nền 2>,...]

<các đối tượng nền> : <kiểu đối tượng>

<các đối tượng nền> : <kiểu đối tượng>

-----

begin\_variables

<các tên thuộc tính > : <kiểu thuộc tính>

<các tên thuộc tính> : <kiểu thuộc tính>

-----

```

end_variables
begin_constraints
    <điều kiện ràng buộc>
    <điều kiện ràng buộc>
    -----
end_constraints
begin_construct_relations
    <sự kiện quan hệ thiết lập>
    <sự kiện quan hệ thiết lập>
    -----
end_construct_relations
begin_properties
    <sự kiện tính chất>
    <sự kiện tính chất>
    -----
end_properties
begin_computation_relations
begin_relation 1
    flag = 0 hoặc 1
     $M_f = \{\text{các thuộc tính}\}$ 
     $rf = 1$ 
     $vf = \{\text{thuộc tính kết quả nếu flag} = 0\}$ 
    expf = ` biểu thức tính toán `
    cost = <trọng số của sự tính toán>
end_relation
begin_relation 2
    -----
end_relation
    -----
end_computation_relations
begin_rules

```

```

begin_rule 1
    kind_rule = "<loại luật>"
    <các tên đối tượng> : <kiểu đối tượng>;
    <các tên đối tượng> : <kiểu đối tượng>;
    -----
    hypothesis_part:
        {các sự kiện giả thiết của luật}
    end_hypothesis_part
    goal_part:
        {các sự kiện kết luận}
    end_goal_part
end_rule 1

begin_rule 2
    -----

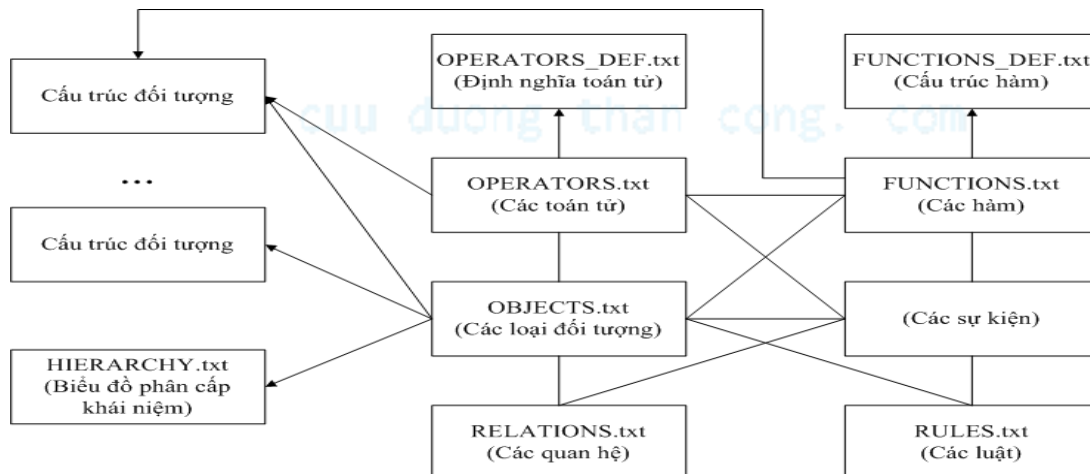
end_rule
-----

end_rules
end_object

```

### 1.4.3 Sơ đồ tổ chức cơ sở tri thức

Mối liên hệ về cấu trúc thông tin trong cơ sở tri thức có thể được minh họa trên sơ đồ sau đây:





Hình 0-5: Sơ đồ tổ chức theo mô hình COKB

#### 1.4.4 Ngôn ngữ đặc tả theo mô hình COKB:

Ngôn ngữ đặc tả theo mô hình COKB được xây dựng để biểu diễn cho các tri thức có dạng COKB. Ngôn ngữ này bao gồm các thành phần :

- Tập hợp các kí tự : chữ, số và các ký tự đặc biệt.
- Từ vựng : từ khóa và tên.
- Các kiểu dữ liệu : Các kiểu dữ liệu cơ bản và các loại có cấu trúc.
- Các biểu thức và câu.
- Các câu lệnh.
- Cú pháp quy định cho các thành phần của mô hình COKB.

Một số cấu trúc của các định nghĩa cho các biểu thức, C-Object,relations, facts và functions.

#### Định nghĩa của các biểu thức :

```
expr      ::=  expr |  
              rel-expr |  
              logic-expr  
expr      ::=  expr add-operator term |  
              term  
term      ::=  term mul-operator factor |  
              factor  
factor    ::=  factor |  
              element ^ factor |  
              element  
element   ::=  (expr ) |  
              name |  
              number |  
              function-call  
rel-expr  ::=  expr rel - operator expr
```

$\text{logic-expr} ::= \text{logic-expr } \mathbf{OR} \text{ logic-term} \mid$   
 $\text{logic-expr } \mathbf{IMPLIES} \text{ logic-term} \mid$   
 $\mathbf{NOT} \text{ logic-term} \mid$   
 $\text{logic-term} \mid$   
 $\text{logic-term} ::= \text{logic-term } \mathbf{AND} \text{ logic-primary} \mid$   
 $\text{logic-primary} \mid$   
 $\text{logic-primary} ::= \text{expr} \mid$   
 $\text{rel-expr} \mid$   
 $\text{function-call} \mid$   
 $\text{quantify-expr} \mid$   
 $\mathbf{TRUE} \mid \mathbf{FALSE}$   
 $\text{quantify-expr} ::= \mathbf{FORALL}(\text{name} <, \text{name}>^*), \text{logic-expr} \mid$   
 $\mathbf{EXISTS}(\text{name}), \text{logic-expr}$

### Định nghĩa của C-Object

$\text{cobject-type} ::= \mathbf{OBJECT} \text{ name};$   
 $\text{[isa]}$   
 $\text{[hasa]}$   
 $\text{[constructs]}$   
 $\text{[attributes]}$   
 $\text{[constraints]}$   
 $\text{[crelations]}$   
 $\text{[rules]}$   
 $\mathbf{ENDCOBJECT};$

### Định nghĩa của các quan hệ đặc biệt:

$\text{isa} ::= \mathbf{ISA}: \text{name} <, \text{name}>^*;$   
 $\text{hasa} ::= \mathbf{HASA}: [\text{fact-def}]$

### Định nghĩa của các sự kiện:

facts ::= **FACT**: fact-def+  
fact-def ::= object-type | attribute | name |  
equation | relation | expression  
object-type ::= cobject-type (name) |  
cobject-type (name <, name>\* )  
relation ::= relation ( name <, name>+ )

### Định nghĩa của các quan hệ cơ sở trong các sự kiện:

relation-def ::= **RELATION** name;  
**ARGUMENT**: argument-def+  
[facts]  
**ENDRELATION**;  
argument-def ::= name <, name>\* : type;

### Định nghĩa của các hàm – dạng 1:

function-def ::= **FUNCTION** name;  
**ARGUMENT**: argument-def+  
**RETURN**: return-def;  
[constraint]  
[facts]  
**ENDFUNCTION**;  
return-def ::= name: type;

### Định nghĩa của các hàm – dạng 2:

function-def ::= **FUNCTION** name;  
**ARGUMENT**: argument-def+  
**RETURN**: return-def;  
[constraint]  
[variables]

```

[statements]
ENDFUNCTION;
statements ::= statement-def+
statement-def ::= assign-stmt | if-stmt | for-stmt
assign-stmt ::= name := expr;
if-stmt ::= IF logic-expr THEN statements+
ENDIF; |
IF logic-expr THEN statements+
ELSE statements+
ENDIF;
for-stmt ::= FOR name IN [range] DO
statements+
ENDFOR;

```

cuu duong than cong. com

#### 1.4.5 Các loại sự kiện trong mô hình COKB:

Sự kiện loại 1: Sự kiện thông tin về loại của đối tượng.

Cấu trúc sự kiện:

[<đối tượng>, <loại đối tượng>]

Ví dụ: Tam giác cân ABC được định nghĩa như sau :

[TAMGIAC[A,B,C], “TAMGIACCAN”].

Sự kiện loại 2: Sự kiện về tính xác định của một đối tượng hay của một thuộc tính của đối tượng.

Cấu trúc sự kiện:

<đối tượng>|<đối tượng >.<thuộc tính>

Ví dụ: Trong tam giác ABC ta có các cạnh AB, AC, BC và các góc ABC, góc BAC và góc ACB được định nghĩa như sau:

- Các cạnh: DOAN[A,B], DOAN[A,C], DOAN[B,C].
- Các góc: GOC[A,B,C], GOC[B,A,C], GOC[A,C,B].

Sự kiện loại 3: Sự kiện về tính xác định của một đối tượng hay của một thuộc tính của đối tượng thông qua biểu thức hằng.

Cấu trúc sự kiện:

$$\langle \text{đối tượng} \rangle | \langle \text{đối tượng} \rangle . \langle \text{thuộc tính} \rangle = \langle \text{biểu thức hằng} \rangle$$

Ví dụ: DOAN[A,B].a = 5; GOC[A,B,C] = Pi/2.

Sự kiện loại 4: Sự kiện về sự bằng nhau của một đối tượng hay một thuộc tính của đối tượng với một đối tượng hay một thuộc tính khác.

Cấu trúc sự kiện:

$$\langle \text{đối tượng} \rangle | \langle \text{đối tượng} \rangle . \langle \text{thuộc tính} \rangle = \langle \text{đối tượng} \rangle | \langle \text{đối tượng} \rangle . \langle \text{thuộc tính} \rangle$$

Ví dụ: DOAN[A,B].a = DOAN[B,C].a, GOC[A,B,C].a = GOC[A,C,B].a.

Sự kiện loại 5: Sự kiện về sự phụ thuộc giữa các đối tượng và các thuộc tính của các đối tượng thông qua một công thức tính toán hay một đẳng thức theo các đối tượng hay các thuộc tính.

Cấu trúc sự kiện:

$$\langle \text{đối tượng} \rangle | \langle \text{đối tượng} \rangle . \langle \text{thuộc tính} \rangle = \langle \text{biểu thức theo các đối tượng hay thuộc tính} \rangle$$

Ví dụ: CV = DOAN[A,B].a + DOAN[A,C].a + DOAN[B,C].a

Sự kiện loại 6: Sự kiện về một quan hệ trên các đối tượng hay trên các thuộc tính của các đối tượng.

Cấu trúc sự kiện:

$$[\langle \text{tên quan hệ} \rangle, \langle \text{object1} \rangle, \langle \text{object2} \rangle, \dots]$$

Ví dụ: ["THUOC", M, DOAN[A,B]] → Điểm M thuộc đoạn AB.

Sự kiện loại 7: Sự kiện về tính xác định của một hàm.

Cấu trúc sự kiện: **<hàm>**

Ví dụ: TRUNGDIEM(A,B) → Hàm xác định trung điểm của 2 điểm A,B.

Sự kiện loại 8: Sự kiện về tính xác định của một hàm thông qua một biểu thức hằng.

Cấu trúc sự kiện:

**<hàm> = <biểu thức hằng>**

Ví dụ: KHOANGCACH(d1, d2) = 9 → Khoảng cách giữa 2 đường thẳng d1 và d2 bằng 9.

Sự kiện loại 9: Sự kiện về sự bằng nhau giữa một đối tượng hay thuộc tính với một hàm.

Cấu trúc sự kiện:

**<đối tượng> | <đối tượng>.<thuộc tính> = <hàm>**

Ví dụ : GOC[A,B,C] = GOC(d1, d2), H1 = HINHCHIEU(A, d)

Sự kiện loại 10: Sự kiện về sự bằng nhau của một hàm với một hàm khác.

Cấu trúc sự kiện:

**<hàm> = <hàm>**

Ví dụ: KHOANGCACH(d,d1) = KHOANGCACH(d1,d2)

Sự kiện loại 11: Sự kiện về sự phụ thuộc của một hàm theo các hàm hay các đối tượng khác thông qua một công thức tính toán.

Cấu trúc sự kiện:

**<hàm> = <biểu thức theo các hàm hay các đối tượng>**

Ví dụ: GOC(d,d1) = GOC(d,d2) + GOC(d,d3).

Sự kiện loại 12: Sự kiện về sự phụ thuộc giữa các hàm hay các đối tượng thông qua một đẳng thức theo các hàm hay các đối tượng.

Cấu trúc sự kiện:

**<đẳng thức theo các hàm hay các đối tượng>**

Ví dụ:  $GOC(d,d1) + GOC(d,d3) = GOC[A,B,C].a + GOC(d,d2)$ .

#### 1.4.6 Định nghĩa các bước giải cho mô hình COKB:

1. **Deduce\_from3s:** suy ra các sự kiện loại 2 từ các sự kiện loại 1
2. **Deduce\_from43s:** suy ra các sự kiện mới loại 3 từ các sự kiện loại 3 và 4 bằng cách thay thế các biến trong sự kiện loại 3 vào sự kiện loại 4.
3. **Deduce\_from53s:** suy ra các sự kiện mới loại 3, 4, 5 từ các sự kiện loại 3 và 5 bằng cách thay thế các biến trong sự kiện loại 3 vào sự kiện loại 5.
4. **Deduce\_from45s:** suy ra các sự kiện mới loại 3 từ các sự kiện loại 4 và 5 bằng cách giải hệ phương trình.
5. **Deduce\_from8s:** suy ra các sự kiện loại 7 từ các sự kiện loại 8.
6. **Deduce\_from983s:** suy ra các sự kiện loại 3, 8 từ các sự kiện loại 3, 8, 9 bằng cách thay thế các sự kiện loại 8 (hay sự kiện loại 3) vào các sự kiện loại 9.
7. **Deduce\_Objects:** thực hiện suy diễn và tính toán bên trong cấu trúc của từng đối tượng. Các đối tượng tham gia vào bước giải có khả năng tham gia vào các bước giải có khả năng thực hiện các hành vi nhất định để phát sinh sự kiện mới, thực hiện suy diễn tính toán trên các thuộc tính của đối tượng, bản thân đối tượng hay các đối tượng liên quan được thiết lập trên nền của đối tượng.
8. **Deduce\_from9s:** suy ra các sự kiện loại 2, 3, 6, 7, 8 từ các sự kiện loại 9 bằng cách thực hiện tính toán hàm.
9. **Deduce\_Rules:** dò tìm luật có thể áp dụng được.
10. **Deduce\_Funcs:** dò tìm hàm có thể áp dụng được.
11. **Deduce\_EqsGoal:** giải hệ phương trình đơn giản gồm n phương trình n ẩn.

#### 1.4.7 Ưu điểm của mô hình COKB

Thông qua những khái niệm về các mô hình biểu diễn tri thức tiêu biểu đã được biết ta đã thấy được một số ưu điểm cũng như những khuyết điểm của chúng. Để làm rõ hơn ta có bảng liệt kê ưu khuyết của các phương pháp biểu diễn tri thức:

P.Pháp	Ưu điểm	Nhược điểm
--------	---------	------------

<b>Luật sinh</b>	Cú pháp đơn giản, dễ hiểu, diễn dịch đơn giản, tính đơn thể cao, linh động (dễ điều chỉnh).	Rất khó theo dõi sự phân cấp, không hiệu quả trong những hệ thống lớn, không thể biểu diễn được mọi loại tri thức, rất yếu trong việc biểu diễn các tri thức dạng mô tả, có cấu trúc.
<b>Mạng ngữ nghĩa</b>	Dễ theo dõi sự phân cấp, sẽ dò theo các mối liên hệ, linh động	Ngữ nghĩa gắn liền với mỗi đỉnh có thể nhập nhằng, khó xử lý các ngoại lệ, khó lập trình.
<b>Mạng tính toán</b>	Giải được hầu hết các bài toán GT → KL nếu như đáp ứng đầy đủ các giả thiết cần thiết. Thuật toán đơn giản dễ cài đặt cho nên việc bảo trì hệ thống tương đối đơn giản. Có thể xây dựng hệ thống suy luận và giải thích một cách rõ ràng và dễ hiểu.	Không giải được các tri thức phức tạp, lưu trữ khó khăn và nhập nhằng khi quản lý. Đồng thời việc xây dựng lại thuật toán là một việc tương đối khó khăn → bảo trì lại toàn bộ hệ thống.
<b>Frame</b>	Có sức mạnh diễn đạt tốt, dễ cài đặt các thuộc tính cho các slot cũng như các mối liên hệ, dễ dàng tạo ra các thủ tục chuyên biệt hóa, dễ đưa vào các thông tin mặc định và dễ thực hiện các thao tác phát hiện các giá trị bị thiếu sót.	Khó lập trình, khó suy diễn, thiếu phần mềm hỗ trợ.

Bảng 0-1: Bảng liệt kê các ưu khuyết của các phương pháp biểu diễn tri thức

Ta nhận thấy “mô hình biểu diễn tri thức bằng Frame” là mô hình biểu diễn tri thức tương đối đối hoàn thiện nhất trong tất cả các phương pháp. Nhưng khuyết điểm của mô hình đó là khó lập trình và thiếu phần mềm hỗ trợ. Trong khi ưu điểm của mô hình COKB là:

- Cấu trúc tường minh giúp dễ dàng thiết kế các môđun truy cập cơ sở tri thức.
- Thích hợp cho việc thiết kế một cơ sở tri thức với các khái niệm có thể được biểu diễn bởi các đối tượng tính toán.
- Tiện lợi cho việc thiết kế các môđun giải bài toán tự động.
- Thích hợp cho việc định dạng ra một ngôn ngữ khai báo bài toán và đặc tả bài toán một cách tự nhiên.



Với những ưu điểm trên mô hình COKB là mô hình lý tưởng để biểu diễn tri thức thay thế cho các mô hình biểu diễn tri thức thông thường. Ngoài ra, với sự hỗ trợ của công cụ Maple phần mềm đại số tính toán là ngôn ngữ lập trình chính đã hỗ trợ một phần rất lớn cho mô hình COKB. Và dựa vào mô hình COKB được trình bày ở trên sẽ là nền tảng để xây dựng cho đề tài sẽ được trình bày vào chương tiếp theo.

cuu duong than cong. com

cuu duong than cong. com