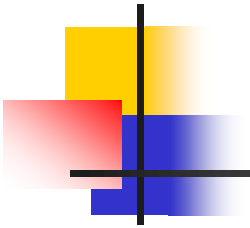
Chương 3 : Các Phương Pháp Biểu Diễn Tri Thức



- **Biểu Diễn Tri Thức Là Gì** ?
- **Biểu Diễn Tri Thức Nhờ Logic Vị Từ**
- **Biểu Diễn Tri Thức Nhờ Mạng Ngữ Nghĩa**
- **Biểu Diễn Tri Thức Nhờ Frame**
- **Giới Thiệu Về Ngôn Ngữ Lập Prolog**

4.1) Biểu Diễn Tri Thức Là Gì ?

- Biểu diễn tri thức là cách thể hiện tri thức trong máy dưới dạng sao cho bài tóan có thể giải được tốt nhất và hiệu quả nhất. Biểu diễn tri thức trong máy phải sao cho

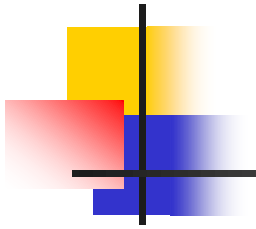
  + Thể hiện được tất cả mọi thông tin cần thiết và đầy đủ về bài tóan.

  + Cho phép tri thức mới được suy diễn ra từ các tri thức sẵn có.

  + Cho phép biểu diễn thể hiện các nguyên lý dưới dạng tổng quát nhất đến dạng đặc trưng nhất.

  + Nắm bắt được ý nghĩa ngữ nghĩa phức tạp nhất.

  + Cho phép lý giải ở mức tri hức cơ bản đến mức tri thức cao hơn.

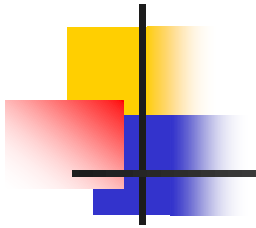


- Có hai loại tri thức của bài tóan cần phải được biểu diễn đó là tri thức mô tả và tri thức thủ tục.

+ **Tri thức mô tả** là loại tri thức mô tả những gì được biết về bài tóan. Loại tri thức này bao gồm sự kiện, đối tượng, lớp của các đối tượng và quan hệ giữa các đối tượng.

+ **Tri thức thủ tục** là thủ tục tổng quát mô tả cách giải quyết bài tóan. Loại tri thức này bao gồm thủ tục tìm kiếm và luật suy diễn.

Có ba phương pháp biểu diễn tri thức mô tả cơ bản đó là phương pháp biểu diễn tri thức nhờ logic vị từ, phương pháp biểu diễn tri thức nhờ mạng ngữ nghĩa và phương pháp biểu diễn tri thức nhờ khung.



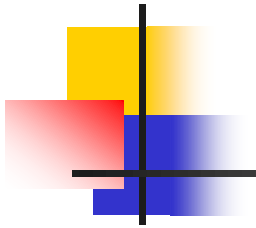
- **Biểu diễn tri thức nhờ logic vị từ** :

+ Tri thức được thể hiện dưới dạng lớp của các biểu thức logic và cơ sở tri thức giải bài tóan được thiết lập trên cơ sở lớp của các biểu thức logic này.

+ Luật suy diễn và thủ tục chứng minh tri thức được lập luận trên cơ sở tóan học logic với các yêu cầu đặt ra của bài tóan.

+ Với phương pháp biểu diễn này cung cấp ý tưởng để tiếp cận với ngôn ngữ lập trình Prolog trong lĩnh vực trí tuệ nhân tạo.

+ Biểu diễn tri thức nhờ logic vị từ còn được gọi là một ngôn ngữ biểu diễn dùng để mã hóa tri thức dưới dạng sao cho dễ lập trình với ngôn ngữ lập trình Prolog.



- **Biểu diễn tri thức nhờ mạng ngữ nghĩa :**

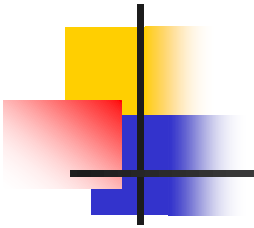
  + Ý tưởng cung cấp một cái nhìn tri thức của bài tóan bằng đồ thị và đồ thị còn được gọi là đồ thị tri thức.

  + Cách biểu diễn này cho phép vạch ra được các ràng buộc vốn sẵn có tự nhiên trong bài tóan và hàm chứa các thành phần của biểu diễn như ngữ từ học, cấu trúc, thủ tục và ngữ nghĩa.

  - **Ngữ từ học** : Các từ vựng hợp lệ sử dụng trong biểu diễn.

  - **Cấu trúc** : Sự liên kết giữa các nút thông qua các cạnh trên đồ thị.

  - **Thủ tục** : Vạch ra hướng giải quyết bài tóan.



- **Ngữ nghĩa** : Hàm chứa ý nghĩa của nút và cạnh liên kết trên đồ thị.

+ Nhờ cách biểu diễn tri thức này, các sự kiện được công thức hóa thành các luật suy diễn giải quyết mọi tình huống đặt ra cần được xử lý của bài tóan. Đồ thị biểu diễn tri thức trong cách này được gọi là mạng suy diễn diễn tri thức.

+ Quá trình biểu diễn, mạng có thể dễ dàng bổ sung thêm tri thức còn thiếu hoặc bớt đi tri thức không cần thiết của bài tóan nhờ thông qua nút và cạnh liên kết.

+ Tri thức được biểu diễn nhờ mạng cũng dễ dàng cho phép được mã hóa bằng ký hiệu nhờ ngôn ngữ biểu diễn logic hoặc ngôn ngữ biểu diễn khung.

- **Biểu diễn tri thức nhờ khung :**

  + Cung cấp ý tưởng tiếp cận với hướng lập trình định hướng đối tượng.

  + Phương pháp biểu diễn này còn được gọi là ngôn ngữ biểu diễn dùng để mã hóa tri thức bằng ký hiệu.

  + Khung còn được gọi là một biểu diễn có cấu trúc với các điểm mạnh như sau :

    - Tính đóng gói : Thể hiện tri thức của bài tóan dưới dạng lớp của nhiều đối tượng chứa nhiều thuộc tính.

    - Tính thừa kế : khung tổng quát nhất được gọi là khung cơ sở và khung đặc trưng hơn được gọi là khung dẫn xuất. Các thuộc tính của khung dẫn xuất có quyền được thừa kế từ các thuộc tính của khung cơ sở.

- Tính đa hình : Thể hiện thông qua sự liên kết của các khung hiện hữu chia sẻ thông tin với một hoặc nhiều khung khác bên trong không hiện hữu.

+ Khung chứa các lọai thông tin như thông tin mặc định, thông tin khung,thông tin mô tả, thông tin thủ tục.

- Thông tin mặc định : Thông tin trả về giá trị mặc định đúng hoặc sai.

- Thông tin khung : Khung này có thể chứa khung khác.

- Thông tin mô tả : Thông tin mô tả về các thành phần của đối tượng.

- Thông tin thủ tục : Khung có thể chứa thông tin thủ tục giải quyết bài tóan.

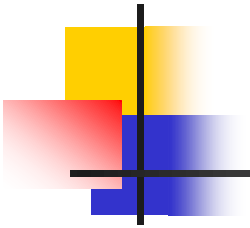
4.2) Biểu Diễn Tri Thức Nhờ Logic Vị Từ

1) Logic đề xuất :

- Logic đề xuất là tập của các đề xuất, trong đó mỗi đề xuất là một phát biểu mà nội dung của nó có thể là đúng hoặc là sai .
- Cú pháp của logic đề xuất gồm có ký hiệu chân lý, ký hiệu đề xuất và tóan tử logic.

+ **Ký hiệu chân lý** : Ký hiệu chân lý là hai chữ cái in hoa T và F, trong đó T xác định nội dung của phát biểu là đúng và F xác định nội dung của phát biểu là sai.

+ **Ký hiệu đề xuất** : Ký hiệu đề xuất là các chữ cái in hoa như A, B, C, D, ... được sử dụng để biểu diễn đề xuất.

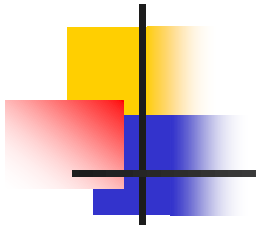


+ **Tóan tử logic** : Tóan tử logic gồm có các lọai như :

- $\wedge$: tóan tử logic liên từ và.
- $\vee$: tóan tử logic giới từ hoặc.
- $\neg$: tóan tử logic phủ định.
- $\rightarrow$: Tóan tử logic kéo nếu.
- $\leftrightarrow$: tóan tử logic tương đương nếu và chỉ nếu.

+ **Câu đề xuất** : Câu đề xuất được định nghĩa như sau :

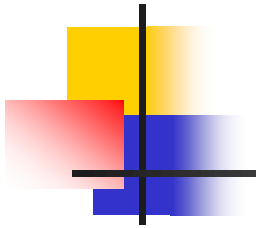
- Mọi ký hiệu đề xuất và ký hiệu chân lý là một câu điển hình là T,F,R,Q,hoặc R là câu.
- Phủ định của câu là câu điển hình $\neg$P là câu.
- Liên kết liên từ và của hai câu là một câu điển hình P$\wedge$Q là câu.



- Liên kết giới từ hoặc của hai câu là một câu điển hình $P \vee Q$ là câu.
- Một câu kéo theo một câu khác là một câu điển hình $P \rightarrow Q$ là câu.
- Một câu nếu và chỉ nếu kéo theo một câu khác cũng là câu điển hình $P \leftrightarrow Q$ là câu.

- Câu đề xuất hợp lệ được xem như một công thức logic hòan thiện (WFFs).

- Biểu thức $P \wedge Q$, trong đó P và Q được gọi là các thành phần liên từ.
- Biểu thức $P \vee Q$, trong P và Q được gọi là các thành phần giới từ.



- Biểu thức $P \rightarrow Q$, trong đó P được gọi là thành phần tiền điều kiện và Q được gọi là thành phần kết luận.

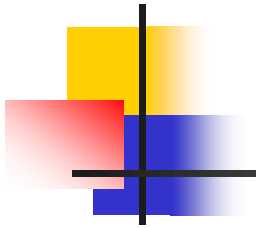
- Trong câu logic đề xuất, các ký hiệu ( ) và [ ] được sử dụng để nhóm các biểu thức con trong câu.
- **Ví dụ** : Cho biểu thức logic là

$$((P \wedge Q) \rightarrow R) = \neg P \vee \neg Q \vee R.$$

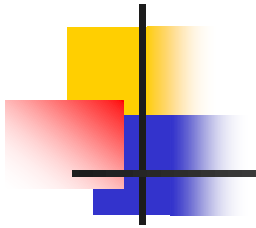
Hãy chứng minh rằng đó là câu hòan thiện ?

- P, Q và R là các ký hiệu đề xuất và vì thế chúng là các câu hòan thiện.

- $P \wedge Q$ là liên từ của hai câu và vì thế nó là một câu hòan thiện.



- $(P \wedge Q) \rightarrow R$ là kéo theo của một câu cho một câu khác và vì thế nó là một câu.
- $\neg P$ và $\neg Q$ là phủ định của câu và vì thế chúng là câu
- $\neg P \vee \neg Q$ là giới từ của hai câu và vì thế nó là một câu hòan thiện.
- $\neg P \vee \neg Q \vee R$ là giới từ của hai câu và vì thế nó cũng là một câu hòan thiện.
- $((P \wedge Q) \rightarrow R) = \neg P \vee \neg Q \vee R$ là sự tương của hai câu và vì thế nó là một câu hòan thiện.

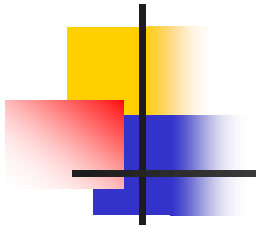


+ **Ngữ nghĩa của logic đề xuất** : Ngữ nghĩa của logic đề xuất đó chính là giá trị chân lý của các ký hiệu đề xuất. Giá trị chân lý đúng của một đề xuất được ký hiệu là T và giá trị chân lý sai của một đề xuất được ký hiệu là F.

  - Giá trị chân lý của phủ định $\neg$, $\neg$P là logic F nếu P là logic T và $\neg$P là logic T nếu P là logic F.

  - Giá trị chân lý của liên từ $\wedge$, là logic T chỉ khi nào giá trị chân lý của cả hai thành phần của nó là logic T; mặt khác giá trị chân lý của nó là logic F.

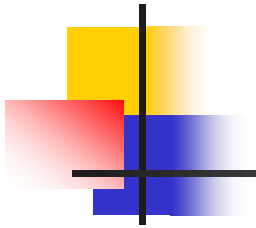
  - Giá trị chân lý của giới từ $\vee$, là logic F chỉ khi nào giá trị chân lý của cả hai thành phần của nó là logic F; mặt khác giá trị chân lý của nó là logic T.



- Giá trị chân lý của phép kéo theo $\rightarrow$, là logic F nếu giá trị chân lý của vế tiền điều kiện là logic T và giá trị chân lý của vế kết luận là logic F; mặt khác giá trị chân lý của nó là logic T.

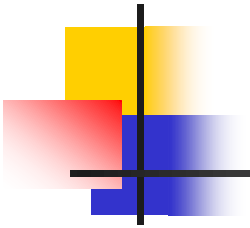
- Giá trị chân lý của phép tương đương $\leftrightarrow$, là T chỉ khi nào hai thành phần của nó là có cùng giá trị chân lý; mặt khác giá trị chân lý của nó là F.

- Cho P, Q và R là các ký hiệu đề xuất, các biểu thức logic sau đây là các biểu thức logic tương đương :
  - $\neg(\neg P) = P$.
  - $(P \vee Q) = (\neg P \rightarrow Q)$.
  - Luật de Morgan : $\neg(P \vee Q) = (\neg P \wedge \neg Q)$.



- Luật de Morgan : $\neg(P \wedge Q) = (\neg P \vee \neg Q)$.
- Luật phân bố : $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$.
- Luật phân bố : $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$.
- Luật giao hóan : $(P \wedge Q) = (Q \wedge P)$.
- Luật giao hóan : $(P \vee Q) = (Q \vee P)$.
- Luật kết hợp : $((P \wedge Q) \wedge R)) = (P \wedge (Q \wedge R))$.
- Luật kết hợp : $((P \vee Q) \vee R)) = (P \vee (Q \vee R))$.
- Luật tương phản : $(P \rightarrow Q) = (\neg Q \rightarrow \neg P)$.

- Hai biểu thức logic được gọi là tương đương khi hai biểu thức có cùng giá trị chân lý.



2) Logic vị từ :

- Logic vị từ là sự mở rộng của logic đề xuất, đôi lúc nó còn được gọi là logic bậc nhất.
- Cách biểu diễn tri thức nhờ logic với các ký hiệu đề xuất không cho phép truy cập hoặc thay thế các thành phần của tri thức và vì vậy, logic đề xuất được mở rộng đến logic vị từ.
- Cách biểu diễn tri thức nhờ logic vị từ cho phép truy cập hoặc thay thế các thành phần của tri thức nhờ thông qua các hàm tính vị từ.

  Ví dụ : Cho đề xuất là "It rained on Tuesday".
- Cách biểu diễn đề xuất này dùng logic đề xuất là

R = It rained on Tuesday.

Với cách biểu diễn này, R có thể được xác minh bằng giá trị chân lý của nó nhưng không thể truy cập được các thành phần cá thể trong đề xuất như rained và tuesday mô tả tính thời tiết và thời gian.

- Cách biểu diễn đề xuất trên dùng logic vị từ là

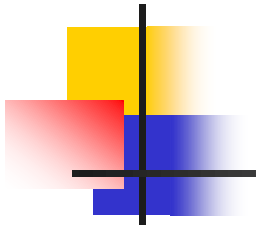
weather(tuesday,rain).

Với cách biểu diễn này, ta có thể truy cập các thành phần cá thể trong đề xuất như tuesday và rain .

- Trong cách biểu diễn này, cũng cho phép thành phần của tri thức được thể hiện dưới dạng biến điển hình là

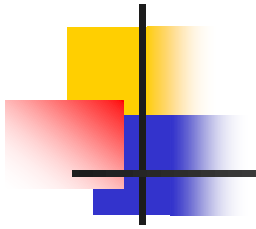
weather(X,rain).

Trong đó,X biểu diễn lớp của các đối tượng trong tuần.



- **Cú pháp của logic vị từ** : gồm có ký hiệu chân lý, ký hiệu vị từ và các phép tóan logic.
- Ký hiệu chân lý và các phép tóan logic của logic vị từ là giống như logic đề xuất.
- Sự khác nhau của hai loại logic này là ký hiệu đề xuất và ký hiệu vị từ.
- Ký hiệu vị từ gồm có hằng vị từ, biến vị từ, hàm vị từ, vị từ và vị từ định lượng.

  + Hằng vị từ : là chuổi của các chữ cái in thường dùng để biểu diễn các thành phần cá thể đặc trưng của đề xuất.



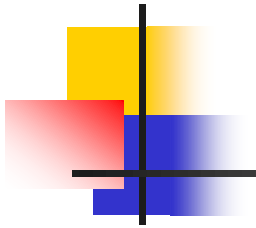
Ví dụ : john, tree, tall, blue là các hằng vị từ hợp lệ.

+ Biến vị từ : là chuỗi của các chữ cái với ít nhất chữ cái đầu tiên của chuỗi phải là chữ cái in hoa dùng để biểu diễn lớp của nhiều đối tượng trong thành phần cá thể của đề xuất.

Ví dụ : X là biến vị từ dùng để biểu diễn lớp của các đối tượng ngày trong tuần hoặc Breaker là biến vị từ dùng để biểu diễn lớp của các đối tượng máy cắt điện.

+ Hàm vị từ : là ánh xạ từ một hoặc nhiều phần tử của tập hợp này đến một phần tử duy nhất trong một tập hợp khác. Hàm phải có tên riêng và các đối số vào.

- Theo qui ước, tên hàm là chuỗi của các chữ cái in thường.



- Cú pháp tổng quát của hàm là

Tên_hàm(< đối số vào >).

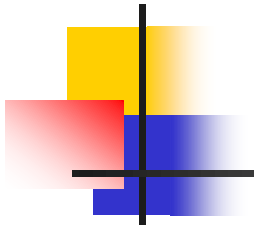
- Hàm nhận các đối số vào từ một tập hợp này và trả về duy nhất một đối số ra trong một tập hợp khác.
- Đối số vào của hàm vị từ có thể là hằng vị từ hoặc biến vị từ.
- Hàm được phép gán cho một hoặc nhiều biến vị từ khác nhau.

Ví dụ : Cho đề xuất là “George is father of David”.

Đề xuất này có thể được biểu diễn bằng hàm vị từ father là

father(david).

Hàm trả về giá trị ra của nó là george.



Cho một đề xuất khác là “2 plus 3 is equal to 5”.

Đề xuất này có thể được biểu diễn bằng hàm vị từ plus là

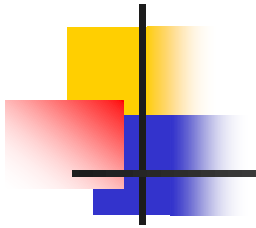
X = plus(2,3).

Hàm sẽ trả về giá trị ra là 5 gán cho biến X.

+ Vị từ tính: Vị từ là một dạng đặc biệt của hàm vị từ. Vị từ cũng phải có tên riêng và các đối số vào của chính nó.

  - Theo qui ước, tên vị từ là chuổi của các chữ cái in thường đó là tên của mối quan hệ giữa hai hoặc nhiều đối tượng trong một đề xuất.

  - Vị từ nhận ít nhất một hoặc nhiều đối số vào từ một tập hợp và trả về đối số ra trong tập hợp khác với giá trị logic đúng T hoặc sai F.



- Đối số vào của vị từ tính có thể là hằng vị từ, biến vị từ hoặc hàm vị từ.
- Vị từ tính không được phép gán cho biến giống như hàm vị từ.

Ví dụ : Cho đề xuất là "George likes Kate".

Đề xuất này có thể được biểu diễn bằng vị từ likes là

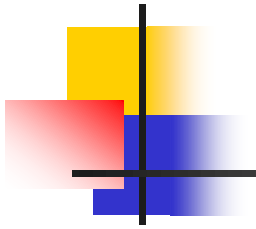
likes(george,kate).

Vị từ likes sẽ trả về logic true(T) nếu George thích Kate; mặt khác vị từ trả về giá trị logic false(F).

Vị từ likes cũng có thể được thể hiện dưới dạng biến là

likes(X, kate).

trong đó X là lớp của các đối tượng thích Kate.



+ Vị từ định lượng : Khi đối số vào của hàm vị từ hoặc vị từ tính là biến vị từ khi đó để xác định phạm vi giá trị của biến, hai đại lượng đứng trước biến $\forall$ và $\exists$, hai đại lượng này được gọi là các vị từ định lượng.

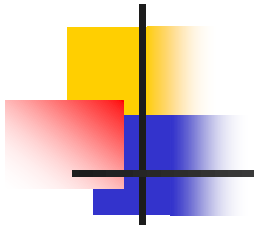
- Vị từ định lượng $\forall$, đứng trước biến để xác minh rằng biểu thức là đúng cho mọi giá trị của biến.

- Vị từ định lượng $\exists$, đứng trước biến để xác minh rằng biểu thức là đúng cho một vài giá trị của biến.

Ví dụ : Cho đề xuất là "All humans are mortal".

Đề xuất này có thể được biểu diễn là

$$(\forall X)\ (human(X) \rightarrow mortal(X)).$$



Cho một đề xuất khác là

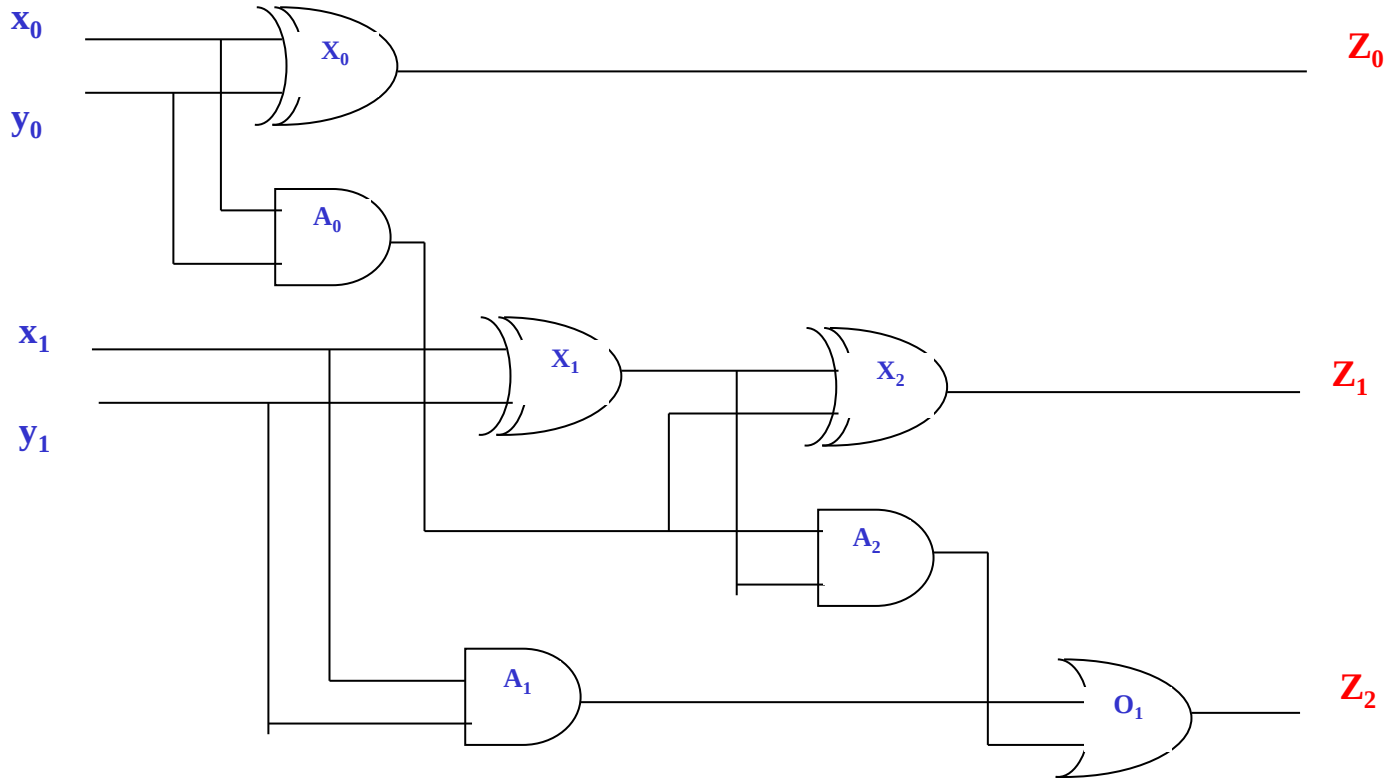
"There is a student who is smart".

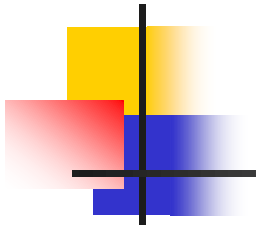
Đề xuất này có thể được biểu diễn là

$(\exists X)(student(X) \wedge smart(X))$.

- **Ví dụ ứng dụng** : Cho mạch số cộng hai bit có nhớ gồm có bốn ngõ vào x_0 , y_0 , x_1, y_1 và ba ngõ ra ngõ ra z_0, z_1, z_2, mỗi của chúng tương ứng với một chữ số 0 hoặc 1. Mạch chứa các thành phần cổng số như X_1, X_2 là các cổng xor, A_1, A_2 là các cổng and, và O_1 là cổng or như hình vẽ.

Mạch số cộng hai bit có nhớ





Hãy xây dựng cơ sở tri thức vận hành mạch được mã hóa bằng logic vị từ ?

- **Cơ sở luật suy diễn vận hành mạch được mã hóa**.

  1. If two terminals are connected, then they have the same signals.

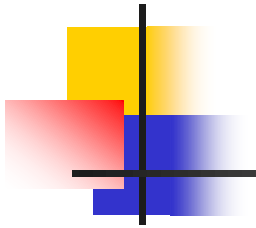
  $\forall T_1 \forall T_2$ connected(T_1 ,T_2) $\rightarrow$ signal(T_1) = signal(T_2).

  2. The signal at every terminal is either on or off (but not both).

  $\forall T$ signal(T) = on $\vee$ signal(T) = off.

  3. Connected is a commutative predicate :

  $\forall T_1 \forall T_2$ connected(T_1 ,T_2) $\leftrightarrow$ connected(T_2 ,T_1).



4. An OR gate's output is on if and only if any of its inputs are on.

$$\forall G \ \ type(G) = or \rightarrow signal(out(L, G)) = on \leftrightarrow$$
$$\exists N \ signal(in(N,G)) = on.$$

5. An AND gate's output is off if and only if any of its inputs are off.

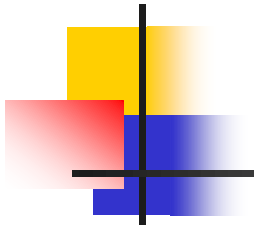
$$\forall G \ \ type(G) = and \rightarrow signal(out(L, G)) = off \leftrightarrow$$
$$\exists N \ signal(in(N,G)) = off.$$

6. An XOR gate's output is on if and only if its inputs are different.

$$\forall G \ \ type(G) = xor \rightarrow signal(out(L, G)) = on \leftrightarrow$$
$$signal(in(1,G)) \neq signal(in(2,G)).$$

7. A NOT gate's output is different from its input.

$$\forall G \ \ type(G) = not \rightarrow signal(out(L, G)) \neq signal(in(L,G)).$$



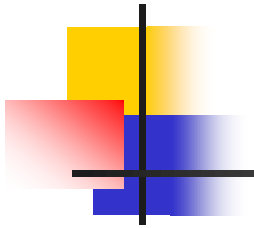
- **Cơ sở dữ liệu của mạch được mã hóa.**

  + Mô tả các loại cổng :

$$type(X_0) = xor,\ type(X_1) = xor,$$
$$type(X_2) = xor,\ type(A_0) = and,$$
$$type(A_1) = and,\ type(A_2) = and,$$
$$type(O_1) = or.$$

  + Mô tả sự liên kết giữa các ngõ vào ra.

$$connected(in(x_0, C), in(1, X_0)).$$
$$connected(in(y_0, C), in(2, X_0)).$$



connected(in(x_0 ,C),in(1,A_0)).

connected(in(y_0 ,C),in(2,A_0)).

connected(in(x_1 ,C),in(1,X_1)).

connected(in(y_1 ,C),in(2,X_1)).

connected(in(x_1 ,C),in(1,A_1)).

connected(in(y_1 ,C),in(2,A_1)).

connected(out(1 ,X_0),out(z_0,C)).

connected(out(1 ,A_0),in(1,A_2)).

connected(out(1 ,A_0),in(2,X_2)).

connected(out(1 ,X_1),in(1,X_2)).
connected(out(1 ,X_1),in(2,A_2)).
connected(out(1 ,X_2),out(z_1,C)).
connected(out(1 ,A_1),in(1,O_1)).
connected(out(1 ,A_2),in(2,O_1)).
connected(out(1 ,O_1),out(z_2,C)).

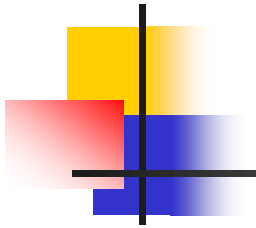
Các yêu cầu đặt ra của bài tóan là cho các tín hiệu vào

x_0 = on, y_0 = on, x_1 = on và y_1 = on.

Hãy lý giải các ngõ ra của mạch ?

3) Giải bài tóan bằng phương pháp hợp giải :

- Có hai hướng giải bài tóan trong lĩnh vực trí tuệ nhân tạo đó là giải bài tóan theo hướng thuận và giải bài tóan theo hướng nghịch.



- Một trong các phương pháp giải bài bài tóan theo hướng nghịc đó là phương pháp hợp giải.
- Phương pháp hợp giải là phương pháp giải bài tóan bằng phương pháp chứng minh phản đề.
- Phương pháp chứng minh phản đề của đề xuất P là đúng từ tập các tiên đề là giả sử rằng phủ định của đề xuất P có giá trị logic đúng và cộng nó vào tập các tiên đề.
- Lần lượt hợp giải cặp hai tiên đề loại bỏ các thành phần mâu thuẩn cho đến khi mệnh tiên đề rỗng xuất hiện.
- Tiên đề rỗng chứng tỏ rằng có sự mâu thuẩn trong tập các tiên đề. Điều này dẫn đến kết luận rằng giả sử phủ định của P là tương thích và P là tương thích với tập các tiên đề.

- Qui trình hợp giải được mô tả gồm các bước như sau :

Bước 1 : Chuyển tất cả các tiên đề ở dạng công thức logic hòan thiện sang dạng logic mệnh đề với dạng tổng quát là $a_1 \vee a_2 \vee a_3 \vee ..... \vee a_n$ sử dụng các biểu thức thức logic tương đương là

$\neg(\neg P) = P.$

$P \rightarrow Q = \neg P \vee Q.$

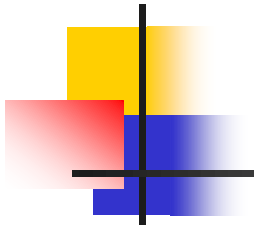
$P \leftrightarrow Q = P \rightarrow Q \wedge Q \rightarrow P.$

$\neg(P \wedge Q) = \neg P \vee \neg Q$

$\neg(P \vee Q) = \neg P \wedge \neg Q.$

$\neg(\exists X)\ P(X) = (\forall X) \neg P(X).$

$\neg(\forall X)\ P(X) = (\exists X) \neg P(X).$

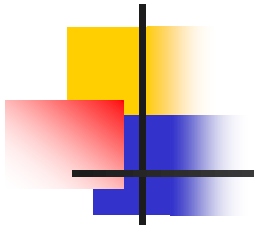


Bước 2 : Để chứng minh đề xuất P là đúng, ta giả sử P là sai, điều đó có nghĩa là phủ định của P là đúng và cộng phủ định này vào tập các tiên đề.

Bước 3 : Đưa tất cả các vị từ định lượng về đứng trước các mệnh đề và loại bỏ chúng khỏi tập các tiên đề.

Bước 4 : Chọn cặp mệnh đề, một có chứa P và một mệnh đề khác chứa ¬P, khử bỏ cặp phân tử này, vì chúng có mâu thuẩn. Các phần tử còn lại của hai mệnh đề hợp nhau tạo thành mệnh đề mới.

Bước 5 : Lặp lại bước 4 cho đến khi nào có mệnh đề rỗng xuất hiện thì dừng thủ tục chứng minh.

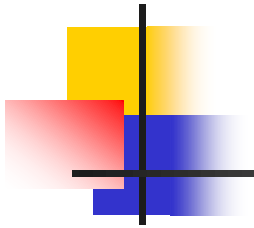


- Mệnh đề rỗng là mệnh đề không chứa bất kỳ một phân tử nào. Điều này chứng tỏ rằng có sự mâu thuẩn trong tập các tiên đề.
- Do đó kết luận rằng đề xuất P là tương thích với tập các tiên đề và phủ định của đề xuất P là không tương thích với tập các tiên đề.

Ví dụ : Cho tập các tiên đề là

1) All dogs are animals.
2) Fido is a dog.
3) All animals will die.

Hãy chứng minh rằng Fido will die bằng phương pháp hợp giải ?



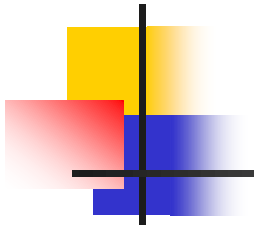
Giải

- Các tiên đề được biểu diễn bằng logic vị từ ở dạng hòan thiện là

  1) $\forall(X)\ (dog(x) \rightarrow animal(X))$.
  2) $dog(fido)$.
  3) $\forall(Y)(animal(Y) \rightarrow die(Y))$.

- Cộng phủ định của đề xuất $\neg die(fido)$ vào tập các tên đề.

  1) $\forall(X)\ (dog(x) \rightarrow animal(X))$.
  2) $dog(fido)$.
  3) $\forall(Y)(animal(Y) \rightarrow die(Y))$.
  4) $\neg die(fido)$.



- Chuyển các công thức logic dạng hòan thiện sang dạng mệnh đề.

1) $\forall(X)(\neg dog(X) \vee animal(X))$.

2) $dog(fido)$.

3) $\forall(Y)(\neg animal(Y) \vee die(Y))$.

4) $\neg die(fido)$.

- Khử bỏ các vị từ định lượng.

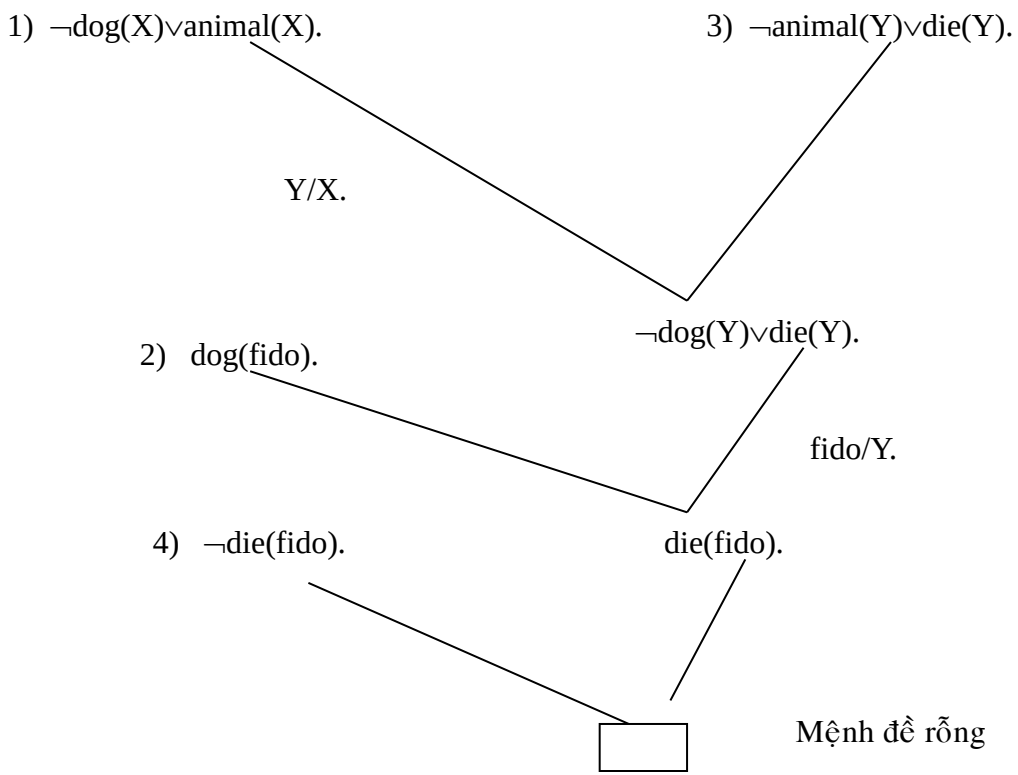
1) $\neg dog(X) \vee animal(X)$.

2) $dog(fido)$.

3) $\neg animal(Y) \vee die(Y)$.

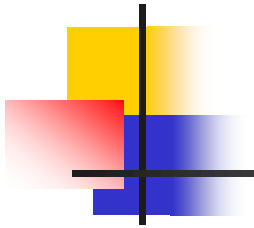
4) $\neg die(fido)$.

- Qui trình chứng minh bằng cây hợp giải nhị phân được mô tả như hình.



4.3) Biểu Diễn Tri Thức Nhờ Mạng Ngữ Nghĩa

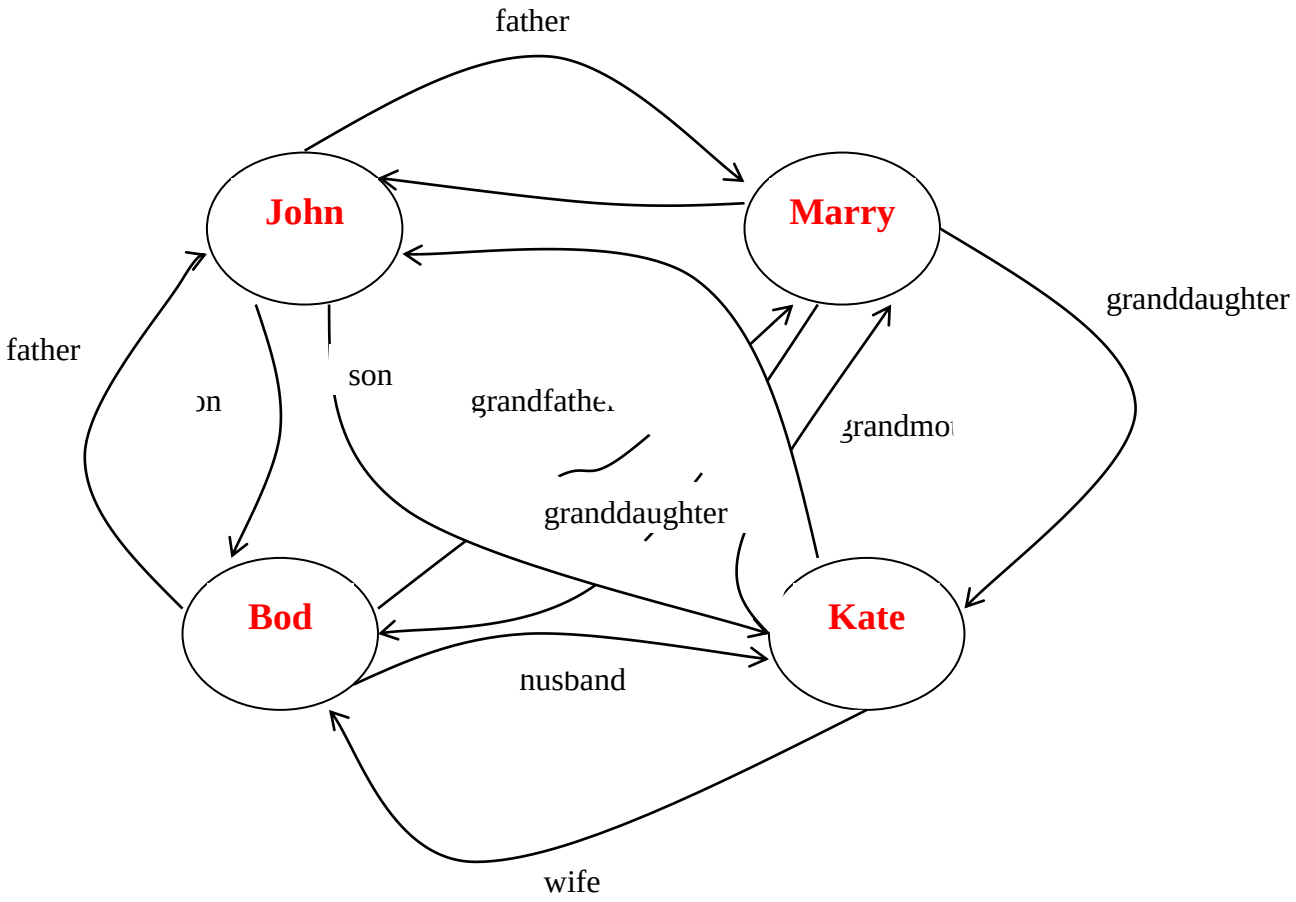
- Một cách biểu diễn tri thức khác đó là mạng ngữ nghĩa.
- Mạng ngữ nghĩa là một đồ thị định hướng gồm tập nút và tập cạnh, trong đó,
  - Mỗi nút có đánh nhãn để biểu diễn một đối tượng.
  - Mỗi cạnh đường mũi tên có đánh nhãn liên kết giữa hai nút chỉ mối quan hệ giữa hai đối tượng.
- Cách biểu diễn này cho ta cái nhìn tổng thể về tri thức của bài tóan bằng bằng đồ thị.
- Nhờ cách biểu diễn này, luật suy diễn được thiết lập thông qua nút và cạnh liên kết.
- Biểu diễn được xem như một cấu trúc dữ liệu mà các ràng buộc của bài tóan có thể được vạch ra .



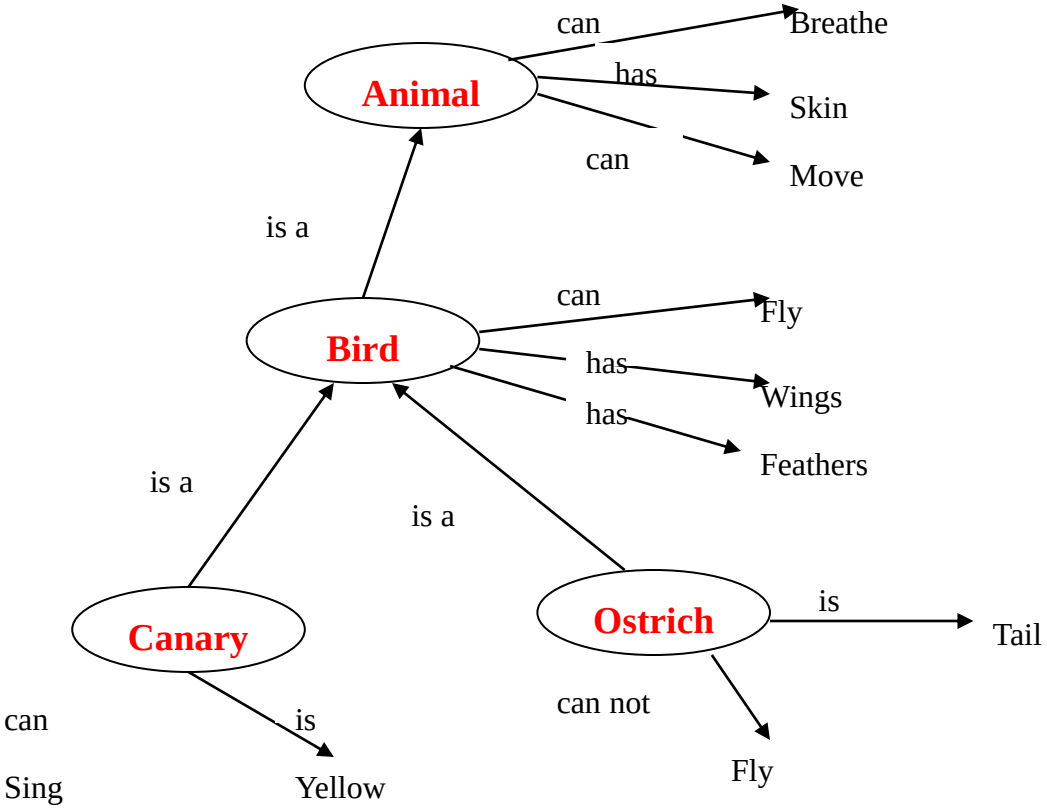
Ví dụ : Cho bài tóan quan hệ gia đình với các sự kiện là

1) John is father of Marry.
2) Mary is daughter of John.
3) Bod is father of John.
4) John is son of Bod.
5) Bod is husband of Kate.
6) Kate is wife of Bod.
7) John is son of Kate.
8) Bod is grandfather of Marry.
9) Kate is grandmother of Marry.
10) Marry is granddaughter of Bod and Kate.

Quan hệ gia đình được biểu diễn nhờ mạng ngữ nghĩa như hình

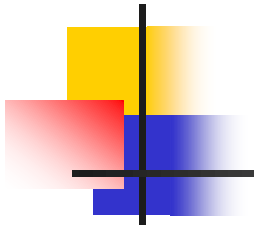


Một biểu diễn khác lớp của các động vật như hình



4.4) Biểu Diễn Tri Thức Nhờ Frame :

- Mộp phương pháp biểu diễn tri thức khác đó là frame.
- Frame là sự mở rộng của mạng ngữ nghĩa trong đó, mỗi nút của mạng là một cấu trúc dữ liệu.
- Frame chứa các slot được xem như là các thuộc tính và các giá trị của các slot được kèm theo.
- Biểu diễn tri thức nhờ Frame cung cấp ý tưởng lập trình hướng đối tượng trong ngôn ngữ lập trình Prolog hoặc các ngôn ngữ lập trình khác như C++ và Visual Basic.
- Frame cho phép truy cập các thành phần của Frame đó là các slot và cho phép hưởng quyền thừa kế giữa Frame dữ liệu này và Frame dữ liệu khác.
- Cấu trúc tổng quát của một frame dữ liệu được mô tả như hình



Frame <frame_name>

Slot : <property_name_1>

Value : <value of property_name_1>

Slot : <property_name_2>

Value : value of property_name_2>

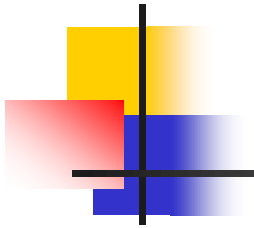
Slot : <property_name_N>

Value : <value of property_name_N>.

Các slot trong mỗi frame chứa các thông tin như sau :

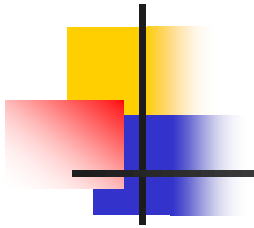
1) Thông tin nhận dạng frame.
2) Thông tin quan hệ của frame này với frame khác.
3) Các thành phần mô tả của frame.
4) Thông tin thủ thủ tục.
5) Thông tin mặc định frame.
6) Thông tin đề xuất mới.

Ví dụ : Cho mạng ngữ nghĩa biểu diễn các sự kiện về động vật như hình trên, các sự kiện này được tổ chức trong các khung như sau :

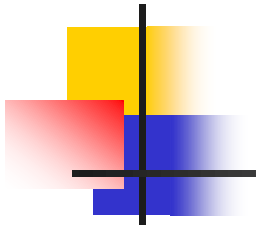


- Frame animal
  Slot : can
  Value : breathe, move
  Slot : has
  Value : wings

- Frame bird
  Slot : is_a
  Value : animal
  Slot : can
  Value : fly
  Slot : has
  Value : wings, feathers



- Frame canary
  - Slot : is_a
  - Value : bird
  - Slot : can
  - Value : sing
  - Slot : is
  - Value : yellow
- Frame ostrich
  - Slot : is_a
  - Value : bird
  - Slot : can_not
  - Value : fly
  - Slot : is
  - Value : tail

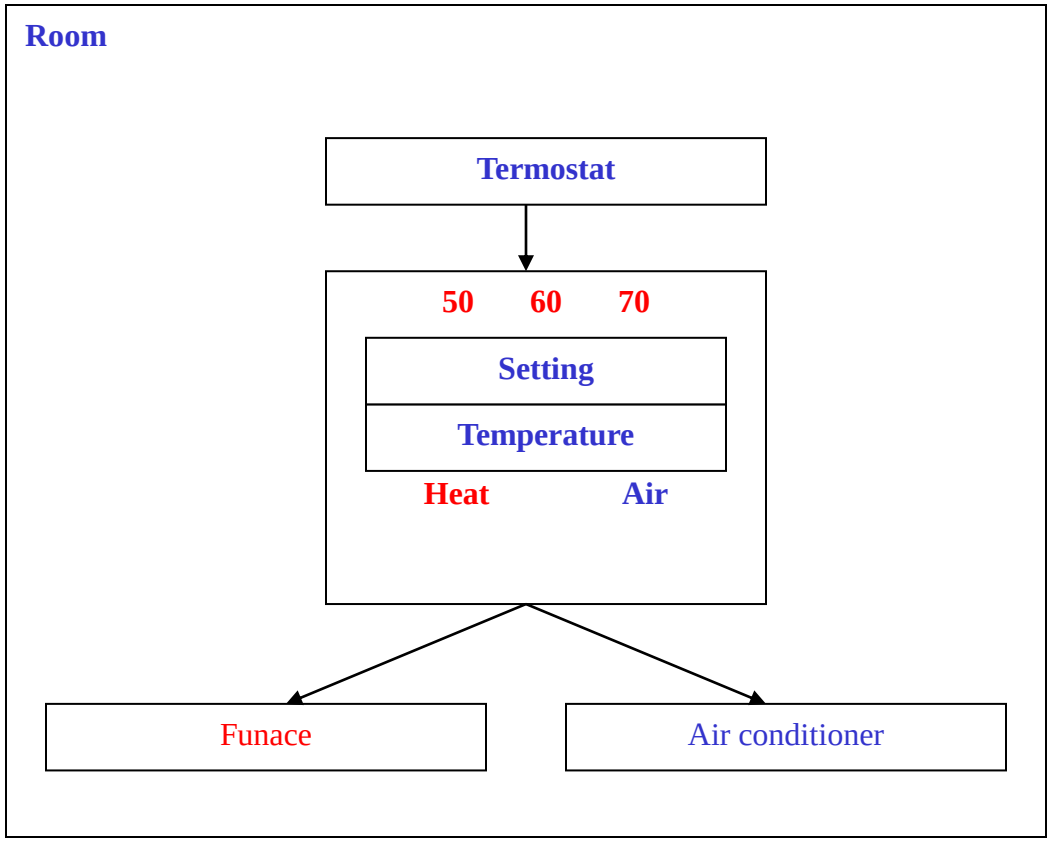


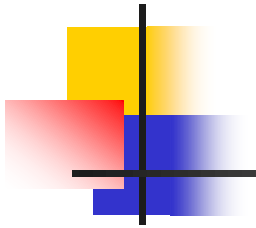
- **Ví dụ ứng dụng** : Thiết kế hệ chuyên gia điều khiển nhiệt độ môi trường trên cơ sở hệ thống Frame.
- Các yêu cầu đặt ra của bài tóan là

  + Điều khiển nhiệt độ môi trường trong một căn nhà nhỏ gồm ba phòng : khách, ăn và ngũ.

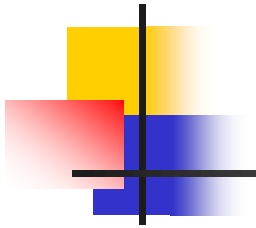
  + Mỗi phòng được trang bị một lò sưởi, một máy điều hòa và một nhiệt kế.

  + Lò sưởi làm ấm, máy điều hòa làm mát và nhiệt kế đo nhiệt độ môi trường.
- Cấu trúc hệ thống điều khiển nhiệt độ trong mỗi phòng được mô tả như hình.

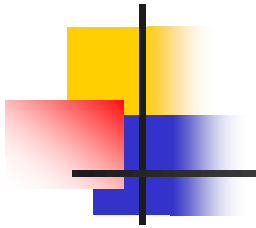

Room
Termostat
50
60
70
Setting
Temperature
Heat
Air
Funace
Air conditioner



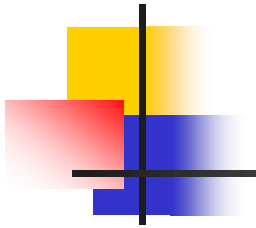
- Hệ thống gồm các frame như room, thermostat, funace và air conditioner được mô tả như sau :
- Frame room
  Slot : funace
  Value : < funac1,funace2,funace3 >
  Slot : air_conditioner
  Value : < air_conditioner1,air_conditioner2, air_conditioner3 >
  Slot : thermostat
  Value : < thermostat1, thermost2, thermostat3 >
  Slot : occupany
  Value : < yes,no >



- Frame thermostat
  Slot : air_conditioner
  Value : < air_conditioner1,air_conditioner2, air_conditioner3 >
  Slot : funace
  Value : < funac1,funace2,funace3 >
  Slot : mode
  Value : <heat,air>
  Slot : setting
  Value : 60
  Slot : temperature
  Value : 65
  Slot : room
  Value : <livingroom, bedroom, kitchenroom >



- Frame air_conditioner
  Slot : room
  Value : <livingroom, bedroom, kitchenroom >
  Slot : state
  Value : <on,off>
  Slot : thermostat
  Value : < thermostat1,thermostat2, thermostat3 >
- Frame funace
  Slot : room
  Value : <livingroom, bedroom, kitchenroom >
  Slot : state
  Value : <on,off>
  Slot : thermostat
  Value : < thermostat1,thermostat2, thermostat3 >

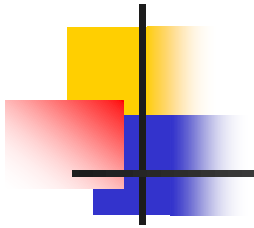


- Cơ sở luật điều khiển nhiệt độ môi trường trong nhà trên cơ sở các thành phần của hệ thống frame được thiết lập gồm các luật là

Luật 1 : if (temperature < setting) and (funace state ≠ off) and (mode ≠ heat)and (roomoccupancy ≠ yes) then send message (funace state = on).

Luật 2 : if (temperature < setting - 5) and (funace state ≠ off) and (mode ≠ heat)and (roomoccupancy ≠ no) then send message (funace state = on).

Luật 3 : if (temperature >= setting) and (funace state ≠ on) and (mode ≠ heat) and (roomoccupancy ≠ yes) then send message (funace state = off).

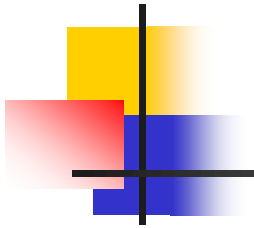


Luật 4 : if (temperature >= setting - 5) and (funace state ≠ on) and (mode ≠ heat) and (roomoccupancy ≠ no) then send message (funace state = off).

Luật 5 : if (temperature < setting) and (air_conditioner state ≠ off) and (mode ≠ air) and (roomoccupancy ≠ yes) then send message (air_conditioner state = on).

Luật 6 : if (temperature < setting - 5) and (air_conditioner state ≠ off) and (mode ≠ air) and (roomoccupancy ≠ no) then send message (air_conditioner state = on).

Luật 7 : if (temperature >= setting) and (air_conditioner state ≠ on) and (mode ≠ air) and (roomoccupancy ≠ yes) then send message (air_conditioner state = off).



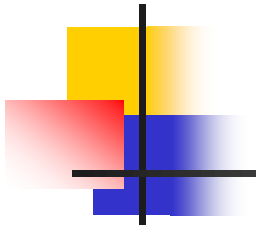
Luật 8 : if (temperature >= setting - 5) and (air_conditioner state ≠ on) and (mode ≠ air) and (roomoccupancy ≠ no) then send message (air_conditioner state = off).

4.5) Giới Thiệu Về Ngôn Ngữ Lập Prolog :

1) Cấu trúc chương trình :

- Ngôn ngữ Prolog là ngôn ngữ lập trình suy luận trên cơ sở tóan học logic để giải quyết các bài tóan trong lĩnh vực trí tuệ nhân tạo.
- Đặc điểm của ngôn ngữ là xử lý tri thức của các bài tóan được mã hóa bằng ký hiệu.
- Một điểm mạnh khác của ngôn ngữ là xử lý danh sách trên cơ sở xử lý song song và đệ qui với các thuật tóan tìm kiếm.
- Ngôn ngữ cho phép liên kết với các ngôn ngữ khác như C, Pascal và Assempler.

Cấu trúc cơ bản của ngôn ngữ lập trình Prolog được mô tả như sau :



- **domains**

  /* domain declarations*/
- **predicates**

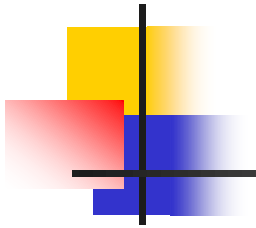
  /* predicate declarations */
- **clauses**

  /*clauses (rules and facts) */
- **goal**

  /* subgoal_1

  subgoal_2

  /*

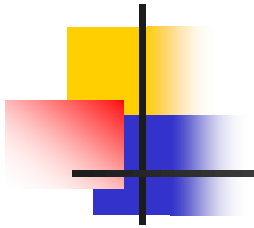


- **domains** : là miền để khai báo kiểu dữ liệu không chuẩn trong các hàm hoặc các vị từ tính predicate. Giống như các ngôn ngữ lập trình khác, các miền kiểu dữ liệu chuẩn của Prolog là short, ushort, word, integer, unsigned, long, ulong, dword, real, string và symbol. Cú pháp khai báo biến của các miền kiểu dữ liệu là

domains

<Variable_name> = <Data_type>

Ví dụ : Khai báo biến d là kiểu dữ liệu số nguyên và List là biến danh sách chứa các số nguyên, cú pháp khai báo hai biến này ở miền domains là



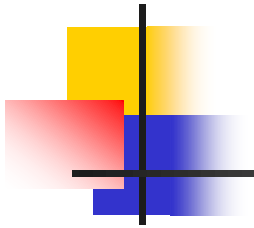
domains

d = integer

List = d*

predicates: là vùng để khai báo các hàm hoặc các vị từ tính predicates. Vị từ tính phải có tên riêng và các đối số vào. Theo qui ước, tên của vị từ tính phải là chuổi của các chữ cái in thường và các đối số khai báo trong các vị từ tính phải là các biến số đã được khai báo ở miền domains, đó là chuổi của các chữ cái phải bắt đầu từ chữ cái in hoa.

Ví dụ : Khai báo vị từ tính father với X là cha của Y, trong đó X và Y là các đối số kiểu dữ liệu symbol. Cú pháp khai báo vị từ tính father với các đối số này là

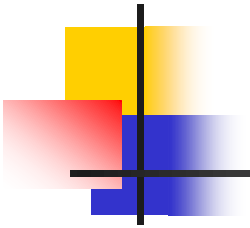


```
domains
        x = symbol
predicates
        father(X,X)
```

- **clauses** : là vùng cho phép thể hiện các mệnh đề sự kiện và mệnh đề luật suy diễn.

  + Mệnh đề sự kiện : đó chính là vị từ tính tính với các đối số của nó là các hằng vị từ mô tả các sự kện hiện có của bài tóan. Cú pháp tổng quát của các mệnh đề sự kiện là

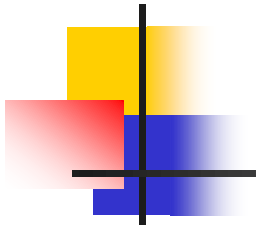
```
clauses
    <predicate_name>( <constant_1>, ...,<constant_N>).
```



Ví dụ : Sự kiện cho là John là cha của Marry. Sự kiện này được thể hiện dưới dạng mệnh đề sự kiện với vị từ tính father là

```
domains
        x = symbol
predicates
        father(X,X)
clauses
        father(john,marry).
```

+ Mệnh đề luật suy diễn : là mệnh đề được thiết lập để thực thi tác vụ của luật suy diễn **If < conditions > Then <conclusion>** với cú pháp tổng quát của Prolog là



clauses

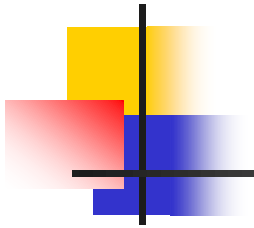
predicateconclusionname(<Arg_1,…,Arg_N>):-
predicatecoditionname_1(<Ar_1>,…, <Arg_N>),
predicatecoditionname_N(<Ar_1>,…, <Arg_N>).

Trong đó, các vị từ tính ở vế phải của mệnh đề là các vị từ điều kiện của luật suy diễn và vị từ tính ở vế trái của mệnh đề là vị từ tính kết luận của luật suy diễn.

Ví dụ : Cho luật suy diễn là

If X is father of Y and Y is father of Z
Then X is grandfather of Z.

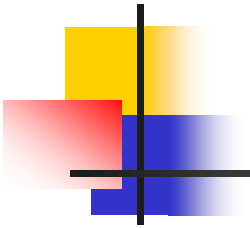
Luật suy diễn này được thể hiện dưới dạng mệnh đề luật suy diễn trong prolog là



```
domains
        x = symbol
    predicates
        father(X,X)
        grandfather(X,X)
    clauses
        father(john,marry).
        grandfather(X,Z):-
            father(X,Y),father(Y,Z).
```

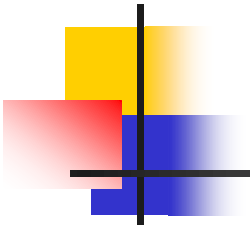
- **goal** : là vùng để thực hiện tác vụ gọi của các đích con, cú pháp tổng quát là

```
goal
        subgoal_1, ... , subgoal_n.
```



Ví dụ: Chương trình sau là một ví dụ minh chứng sử dụng goal.

```
domains
        x = symbol
predicates
        father(X,X)
        grandfather(X,X)
clauses
        father(john,marry).
        grandfather(X,Z):- father(X,Y),father(Y,Z).
test:- grandfather(X,Z),write(X," grandfather of ",Z),nl.
goal
        test.
```

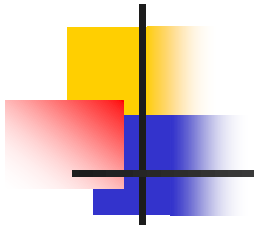


Chạy chương trình này cho kết quả là

bod is Grandfather of marry, yes.

2) Các loại tóan tử trong prolog:

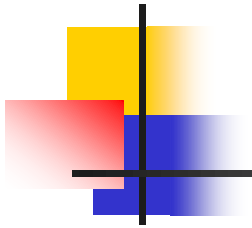
- **Tóan tử logic :**
  - + Logic and: , (dấu phảy).
  - + Logic or : ; (dấu chấm phảy).
  - + Logic not : not.
- **Tóan tử quan hệ** :
  - + Nhỏ hơn : <
  - + Nhỏ hơn hoặc bằng : <=
  - + Bằng nhau : =
  - + Lớn hơn : >
  - + Lớn hơn hoặc bằng : >=
  - + Tóan tử không bằng nhau : <> hoặc ><



+ Tóan tử cut : **!**

Ví dụ : Chương trình Prolog giải phương trình bậc 2 $ax^2 + bx + c = 0$ là một ví dụ minh chứng.

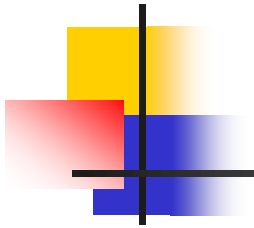
```
predicates
        solve(real,real,real)
        reply(real,real,real)
clauses
        solve(A,B,C):-
                D = B*B - 4*A*C,
                reply(A,B,D),nl.
        reply(_,_,D):-
                D < 0,
                write("No solution"),!.
```



```
reply(A,B,D):-
        D = 0,
        X = -B/(2*A),
        write("X = ",X),!.
reply(A,B,D):-
        X1 = (-B +sqrt(D))/(2*A),
        X2 = (-B -sqrt(D))/(2*A),
write("X1 = ",X1," and X2 = ",X2).
```

goal

```
write("Enter cofficient A:"),readreal(A),
write("Enter cofficient B :"),readreal(B),
write("Enter cofficient C :"),readreal(C),
solve(A,B,C).
```



Chạy chương trình cho kết là

Enter cofficient A:1

Enter cofficient B :3

Enter cofficient C :1

X1 = -0.3819660113 and X2 = -2.618033989

A=1, B=3, C=1

1 Solution

Lưu ý : tóan tử cut sử dụng trong các mệnh đề luật suy diễn để bỏ qua các mệnh đề khác nếu mệnh đề đó thỏa mãn điều kiện. Nếu không có tóan tử cut, prolog sẽ yêu cầu từ khóa nondeterm đứng trước các vị từ tính khai báo ở vùng predicates.

3 3)Xử lý danh sách :

Danh sách là lớp chứa nhiều đối tượng khác nhau tùy ý.

Ví dụ : Danh sách chứa các số nguyên là

[1, 2, 3, 4, 5].

Khai báo danh sách :

domains

<List_name> = <Datatype>*

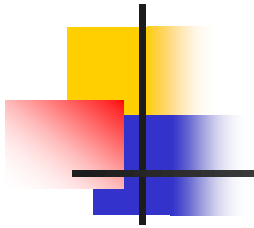
- **Thành viên của danh sách** : Nếu X là thành viên của danh sách L thì mệnh đề luật suy diễn được thiết lập là

clauses

member(X,[X|-]):- !.

Member(X,[_|L]) :- member(X,L).

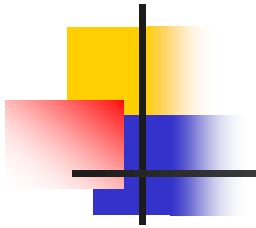
Mệnh đề này sẽ trả về giá trị logic đúng nếu X là thành viên của danh sách L; mặt khác sẽ trả về logic sai.



Ví dụ : Chương trình sau là một ví dụ minh chứng.

```
domains
        x = symbol
        List = symbol*
predicates
        member(X,List)
clauses
        member(X,[X|_]):-!.
        member(X,[_|T]) :- member(X,T).
goal
        member(b,[a,b,c]).
```

Chạy chương trình sẽ cho đáp án là Yes , vì a là thành viên của danh sách; mặt khác đáp án là No.

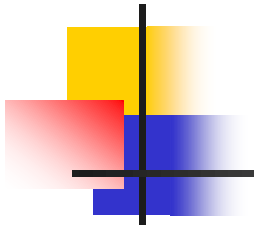


- **Nối danh sách** : Mệnh đề nối hai danh sách được thiết lập là

```
clauses
        append([],List,List).
        Append([X|L1],L2,[X|L3]) :-
                append(L1,L2,L3).
```

Ví dụ : Chương trình sau là một ví dụ minh chứng nối hai danh sách.

```
domains
        s = symbol
        List = symbol*
predicates
        append(List,List,List)
clauses
        append([],List,List).
        append([X|T],L,[X|NL]) :- append(T,L,NL).
goal
        append([a,b,c],[d,e],Y).
```



Chạy chương trình này cho kết quả là

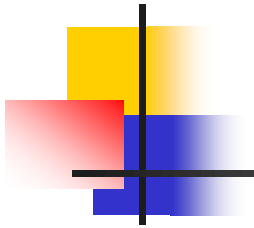
Y = [“a”,”b”,”c”,”d”,”e”].

- Hiển thị danh sách : Mệnh đề hiển thị danh sách ra màn hình được thiết lập là

```
writelist([]).
writelist([H|T) :-  write(H),nl,writelist(T).
```

- Hiển thị đảo ngược danh sách : mệnh đề hiển thị đảo ngược thứ tự các phần tử trong danh sách được thiết lập là

```
reverse_writelist([]).
Reverse_writelist([H|T]):-
        reverse_writelist(T),write(H),nl.
```



- Mệnh đề khử bỏ phần tử đầu tiên của danh sách :

  dequeue(E,[E|T],T).

- Mệnh đề cộng phần tử vào cuối danh sách :

  add_to_queue(E,[],[E]).
  add_to_queue(E,[H|T],[H|Tnew]):-
  add_to_queue(E,T,Tnew).

- Mệnh đề khẳng định thành viên là của tập hợp :

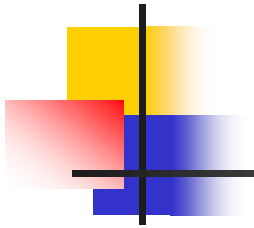
  member_set(S,L):- member(S,L).

- Stack : stack(Top,Stack,[Top|Stack]).

Cộng đối tượng Top vào đầu danh sách Stack.

Ví dụ :

stack(e,[a,b,c,d],Y), trả về danh sách Y = [e,a,b,c,d].



- Cộng danh sách vào cuối stack :

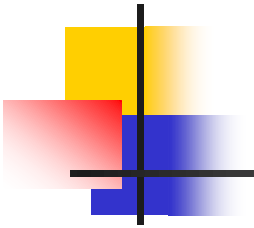
```
add_list_to_stack(List,Stack,Result) :-
                        append(List,Stack,Result).
```

- Thành viên của stack :

```
member_stack(Element,Stack):-
                        member(Element,Stack).
```

- Cộng đối tượng X vào danh sách S nếu nó không thuộc thành viên của S :

```
add_if_not_in_set(X,S,S):-
                member(X,S),!.
                add_if_not_in_set(X,S,[X|S]).
```



Mệnh đề một xác định nếu X là thành viên của S thì bỏ qua; mặt khác mệnh đề 2 cộng thành viên X vào danh sách S.

- Subset : Xác định T là tập con của L.

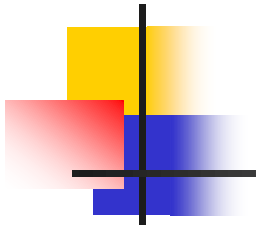
```
subset([],_).
subset([H|T],L):- member_set(H,L),subset(T,L).
```

Ví dụ : subset([1,2],[1,2,3,4,5]). Trả về logic đúng.

+ **Cộng Thành viên vào cuối danh sách :**

```
add_to_list(D,[],[D]).
add_to_list(D,[H|T],[H|Tnew]):-
        add_to_list(D,T,Tnew).
```

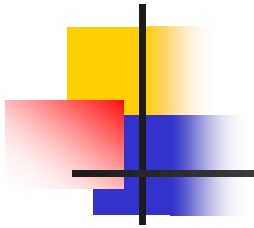
Ví dụ : add_to_list(6,[1,2,3,4,5],L), cho kết quả là L=[1,2,3,4,5,6].



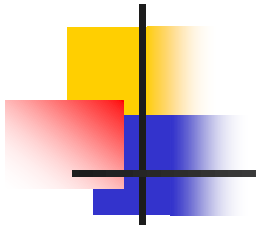
- **Sử dụng lệnh findall** : Lệnh tìm tất cả các giá trị của một lớp đối số nào đó trong vị từ tính nhốt vào một danh sách.

Ví dụ : Chương trình sau là một ví dụ minh chứng cách sử dụng lệnh findall.

```
domains
        name, address = string
        age = integer
        list = age*
predicates
        nondeterm person(name,address,age)
        sumlist(list,age,integer)
```



```
clauses
        sumlist([],0,0).
        sumlist([H|T],Sum,N):-
                sumlist(T,S1,N1),
                Sum = H + S1, N = 1 + N1.
    person("Sherlock Holmes", "22B Baker Street",42).
    person("Petter Spiers", "Apt. 22, 21st Street",36).
    person("Mary Darrow", "Suite 2, Omega Home",51).
goal
    findall(Age,person(_,_,Age),L),sumlist(L,Sum,N),
        Ave = Sum/N, write("Averages = ",Ave),nl.
```



- Chạy chương trình này cho kết quả là

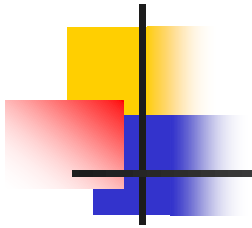
  Averages = 43

  L=[42,36,51], Sum=129, N=3, Ave=43

  1 Solution.

- **Hủy bỏ phần tử bất kỳ trong danh sách** : Chương trình sau là một ví dụ minh chứng.

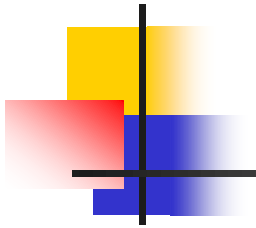
```
domains
        d = integer
        L = d*
predicates
        nondeterm delete(D,L,L)
```



```
clauses
        delete(_,[],[]).
        delete(X,[X|L],M):-
        delete(X,L,M).
        delete(X,[Y|L],[Y|M]):-
        not(X = Y),delete(X,L,M).
goal
        delete(2,[1,2,3,4,5],M).
```

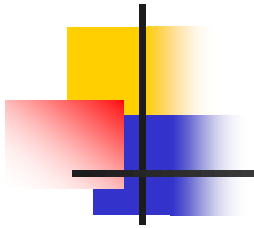
Chạy chương trình cho kết quả là M=[1,3,4,5]

1 Solution.

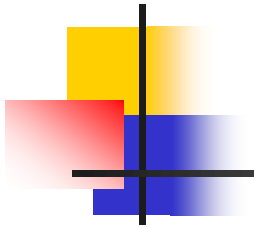


- Khai báo tổng hợp : Giả sử trạng thái của bài tóan là một danh sách chứa nhiều đối tượng với nhiều kiểu dữ liệu khác nhau, chương trình sau là một ví dụ minh chứng cách khai báo dữ liệu hổn hợp này.

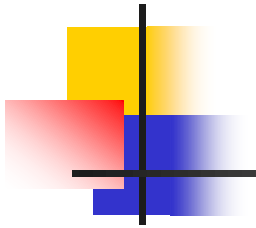
```
domains
llist = table(symbol);on(symbol,symbol);clear(symbol)
state = llist*
path = state*
predicates
append(path,path,path)
reverse_writelist(path)
```



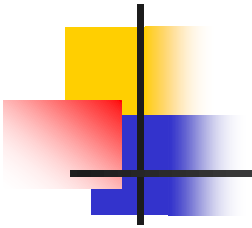
```
nondeterm add(state,path,path)
nondeterm del(state,path,path)
nondeterm test
clauses
append([],L,L).
append([X|L1],L2,[X|L3]):-
append(L1,L2,L3).
add(D,[],[D]).
add(D,[H|T],[H|Tnew]):-
add(D,T,Tnew).
```



```
del(_,[],[]).
del(X,[X|L],M):-
del(X,L,M).
del(X,[Y|L],[Y|M]):-
not(X = Y),del(X,L,M).
reverse_writelist([]).
reverse_writelist([H|T]):-
reverse_writelist(T),write(H),nl.
test :-State =[table(a),on(b,a),clear(b)],
Next = [table(b),clear(a)],
Path1 = [State,Next],
```



```
add(State,Path1,Path2),
reverse_writelist(Path2),nl,
del(State,Path2,Path3),
reverse_writelist(Path3),nl,
append(Path2,Path3,Path),
reverse_writelist(Path).
goal
test.
```

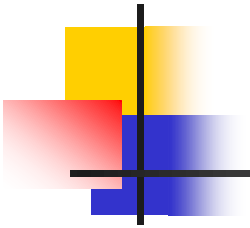


Chạy chương trình trên cho kết quả là

[table("a"),on("b","a"),clear("b")]
[table("b"),clear("a")]

[table("a"),on("b","a"),clear("b")]
[table("b"),clear("a")]

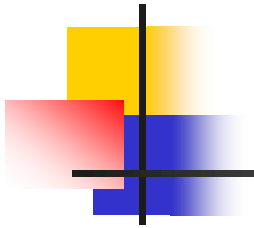
[table("b"),clear("a")]
[table("a"),on("b","a"),clear("b")]
[table("b"),clear("a")]
[table("a"),on("b","a"),clear("b")]
yes.



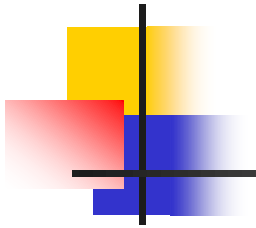
4) Ví dụ ứng dụng :

Ví dụ 1 : Chương trình giải bài tóan nông dân, chồn, ngỗng và ngũ cốc.

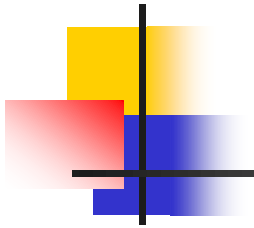
```
domains
        LOC   = east;
                west
 STATE = state(LOC farmer,LOC wolf,LOC goat,LOC cabbage)
  PATH  = STATE*
predicates
         go(STATE,STATE)
        path(STATE,STATE,PATH,PATH)
```



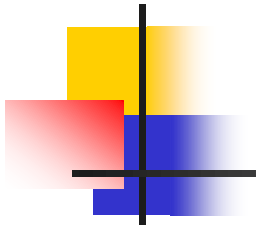
```
  nondeterm move(STATE,STATE)
  opposite(LOC,LOC)
  nondeterm unsafe(STATE)
  nondeterm member(STATE,PATH)
  write_path(PATH)
  write_move(STATE,STATE)
clauses
      go(StartState,GoalState):-
      path(StartState,GoalState,[StartState],Path),
      write("A solution is:\n"),
      write_path(Path).
```



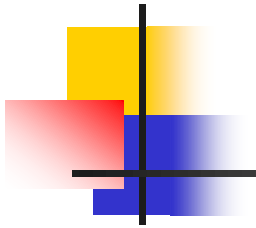
```
path(StartState,GoalState,VisitedPath,Path):-
move(StartState,NextState),
not(member(NextState,VisitedPath)),
path(NextState,GoalState,[NextState|VisitedPath],Path),
        !.
 path(GoalState,GoalState,Path,Path).
move(state(X,X,G,C),state(Y,Y,G,C)):-
opposite(X,Y),not(unsafe(state(Y,Y,G,C))).
move(state(X,W,X,C),state(Y,W,Y,C)):-
opposite(X,Y),not(unsafe(state(Y,W,Y,C))).
 move(state(X,W,G,X),state(Y,W,G,Y)):-
opposite(X,Y),not(unsafe(state(Y,W,G,Y))).
move(state(X,W,G,C),state(Y,W,G,C)):-
opposite(X,Y),not(unsafe(state(Y,W,G,C))).
```



```
 opposite(east,west).
 opposite(west,east).
 unsafe(state(F,X,X,_)):- % The wolf eats the goat
 opposite(F,X),!.
 unsafe(state(F,_,X,X)):- % The goat eats the cabbage
opposite(F,X),!.
 member(X,[X|_]):- !.
 member(X,[_|L]):-
 member(X,L).
write_path([H1,H2|T]):-
      write_move(H1,H2), write_path([H2|T]).
write_path([]).
```



```
write_move(state(X,W,G,C),state(Y,W,G,C)):- !,
write("The farmer crosses the river from ",X," to ",Y),nl.
write_move(state(X,X,G,C),state(Y,Y,G,C)):- !,
write("The farmer takes the Wolf from ",X," of the river
to ",Y),nl.
write_move(state(X,W,X,C),state(Y,W,Y,C)):- !,
write("The farmer takes the Goat from ",X," of the river
to ",Y),nl.
 write_move(state(X,W,G,X),state(Y,W,G,Y)):- !,
write("The farmer takes the cabbage from ",X," of the
river to ",Y),nl.
```

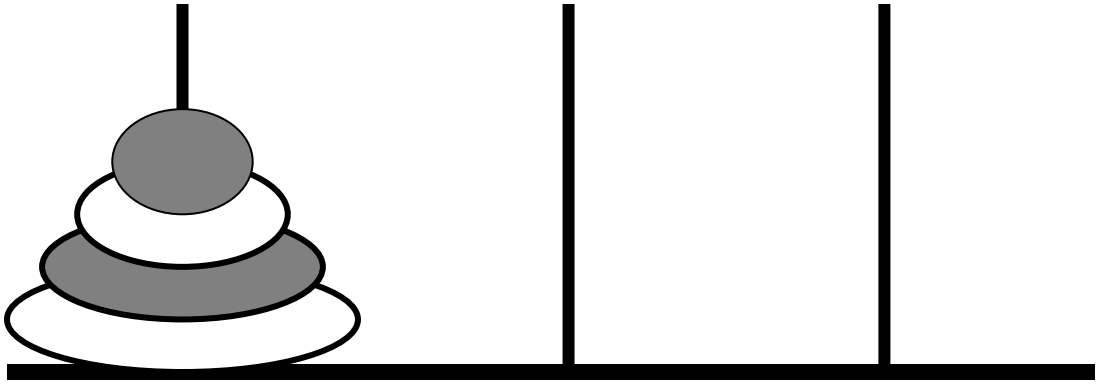


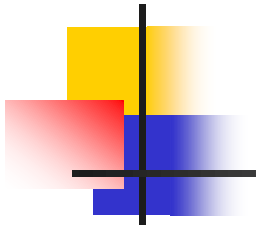
```
goal
    go(state(east,east,east,east),state(west,west,west,west)),
  write("solved").
```

Chạy chương trình này cho kết quả là

```
A solution is:
The farmer takes the Goat from west of the river to east
The farmer crosses the river from east to west
The farmer takes the cabbage from west of the river to east
The farmer takes the Goat from east of the river to west
The farmer takes the Wolf from west of the river to east
The farmer crosses the river from east to west
The farmer takes the Goat from west of the river to east
no
```

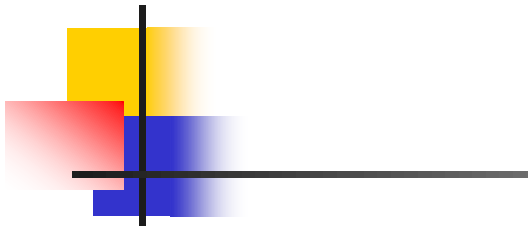
Ví dụ 2 : Bài tóan tháp Hà Nội.





- Chương trình Prolog sau là một ví dụ minh chứng giải bài tóan tháp Hà Nội này.

```
domains
        loc =right;middle;left
predicates
        hanoi(integer)
        move(integer,loc,loc,loc)
        inform(loc,loc)
clauses
      hanoi(N):-
              move(N,left,middle,right).
              move(1,A,_,C):-
              inform(A,C),!.
```

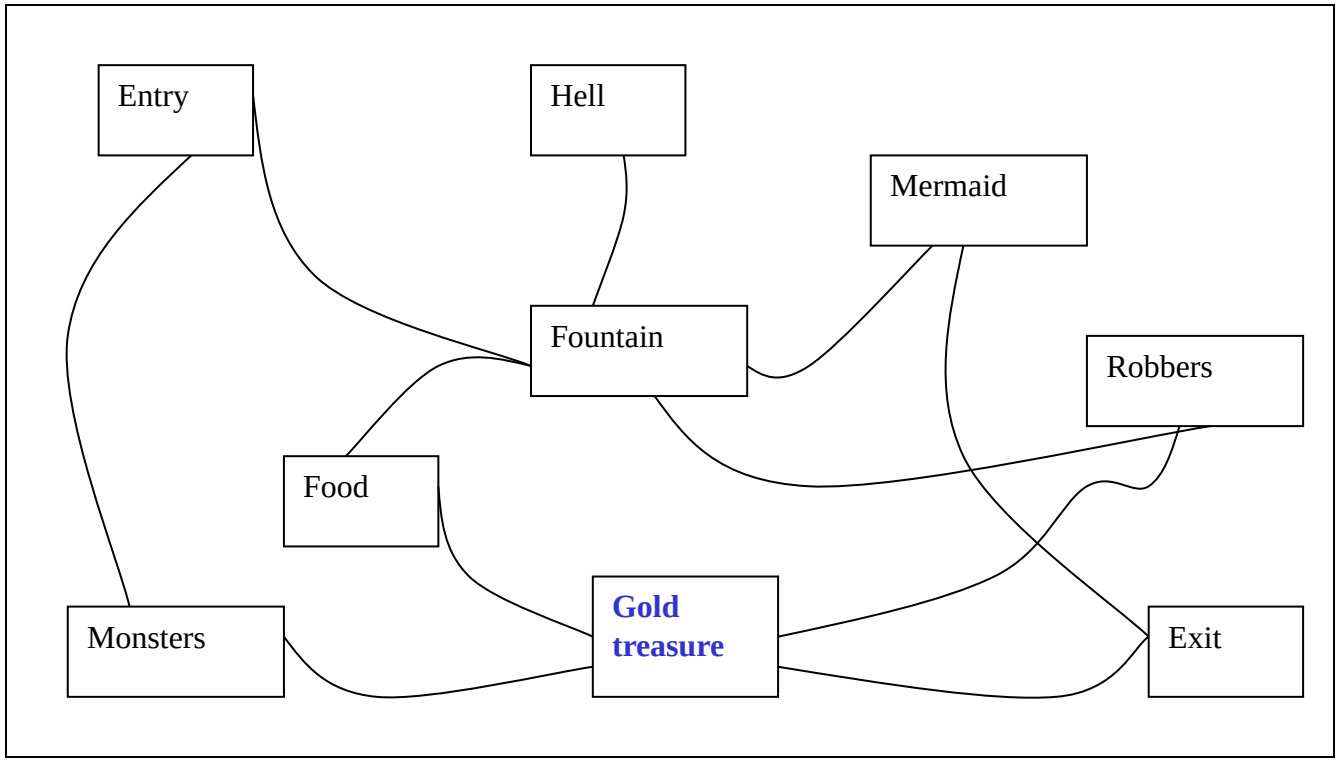


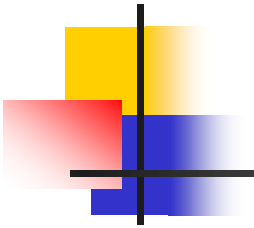
```
move(N,A,B,C):-
        N1=N-1,move(N1,A,C,B),inform(A,C), move(N1,B,A,C).
        inform(Loc1, Loc2):-
        write("\nMove a disk from ", Loc1, " to ", Loc2).
goal
        hanoi(3).
```

Chạy chương trình này cho kết quả là

```
Move a disk from left to right
Move a disk from left to middle
Move a disk from right to middle
Move a disk from left to right
Move a disk from middle to left
Move a disk from middle to right
Move a disk from left to right   yes
```

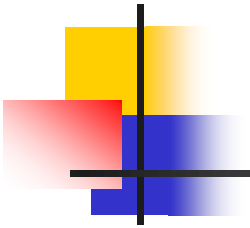
Ví dụ 3 : Bản đồ tìm kho báu.





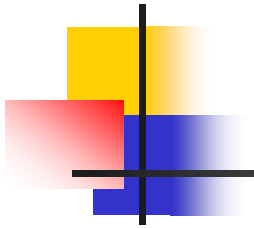
Chương trình Prolog sau là một ví dụ minh chứng tìm đường an tòan đến hang động kho báu chứa vàng.

```
domains
    room = symbol
    roomlist = room*
predicates
    nondeterm gallery(room,room)
    nondeterm neighborroom(room,room)
    avoid(roomlist)
    nondeterm go(room,room)
    nondeterm route(room,room,roomlist)
    nondeterm member(room,roomlist)
```

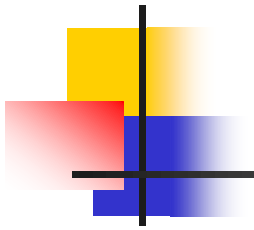


clauses

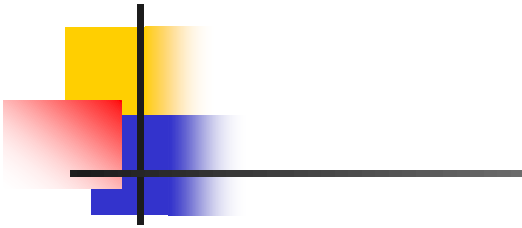
```
gallery(entry,monsters).
gallery(entry,fountain).
gallery(fountain,hell).
gallery(fountain,food).
gallery(exit,gold_treasure).
gallery(fountain,mermaid).
gallery(robbers,go_treasure).
gallery(fountain,robbers).
gallery(food,gold_treasure).
```



```
gallery(mermaid,exit).
gallery(monsters,gold_treasure).
gallery(gold_treasure,exit).
gallery(mermaid,gold_treasure).
neighborroom(X,Y) :- gallery(X,Y).
neighborroom(X,Y) :- gallery(Y,X).
avoid([monsters,robbers]).
go(Here,There) :- route(Here,There,[Here]).
go(_,_).
```



```
route(Room,Room,VisitedRooms)                                    :-
    member(gold_treasure,VisitedRooms),
                    write(VisitedRooms),nl,fail.
route(Room,Way_out,VisitedRooms)                                 :-
    neighborroom(Room,Nextroom),
          avoid(DangerousRooms),
          not(member(NextRoom,DangerousRooms)),
          not(member(NextRoom,VisitedRooms)),
route(NextRoom,Way_out,[NextRoom|VisitedRooms]).
member(X,[X|_]):-!.
 member(X,[_|L]):-
   member(X,L).
```



goal

go(entry,exit).

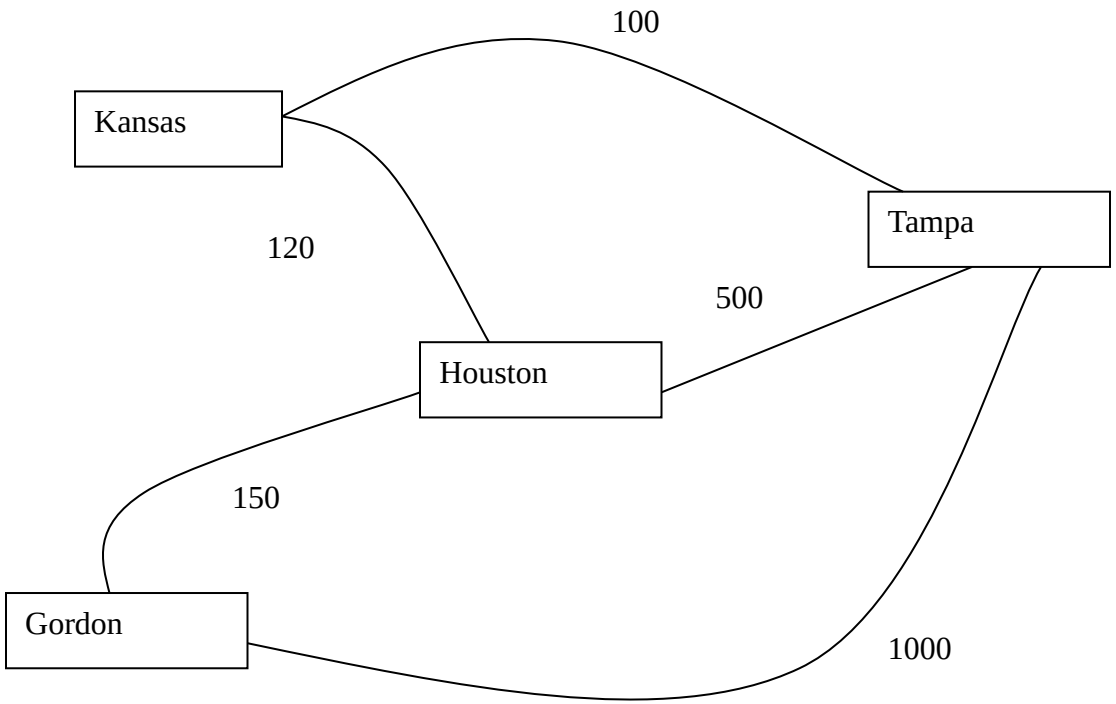
Chạy chương trình này cho kết quả là

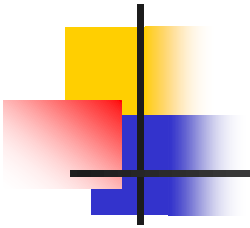
["exit","gold_treasure","food","fountain","entry"]

["exit","gold_treasure","food","fountain","entry"]

yes

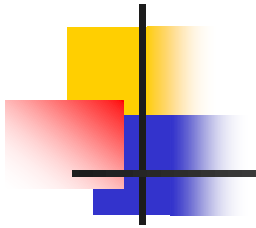
Ví dụ 4 : Tìm đường đi ngắn nhất



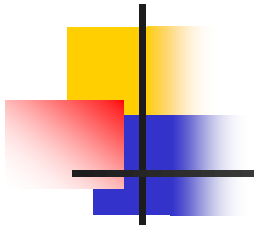


Chương trình Prolog sau là một ví dụ minh chứng giải bài tóan tìm đường đi ngắn nhất từ thành phố Gordon đến thành phố Tampa.

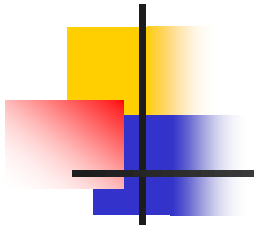
```
domains
  town     = symbol
  townlist = town*
  distance = integer

predicates
  nondeterm road(town,town,distance)
clauses
  road(tampa,houston,500).
     road(tampa,kansas_city,100).
```



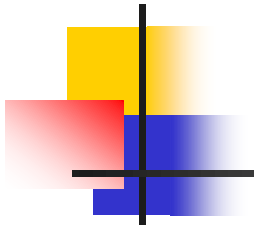
```
road(gordon,tampa,1000).
 road(houston,gordon,150).
 road(houston,kansas_city,120).
predicates
  nondeterm connected(town,town,distance)
clauses
  connected(X,Y,Dist):-
    road(X,Y,Dist).
  connected(X,Y,Dist):-
    road(Y,X,Dist).
```



```
predicates
  determ member(town,townlist)
clauses
  member(X,[X|_]):-!.
  member(X,[_|L]):-
    member(X,L).
 predicates
  nondeterm route(town,town,townList,townList,distance)
clauses
  route(Town,Town,VisitedTowns, VisitedTowns, 0) :-
    !.
```

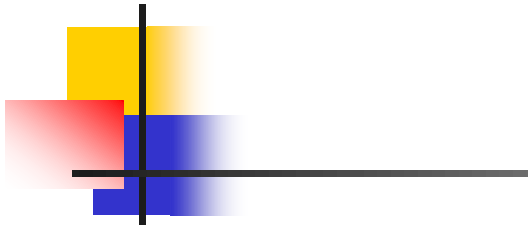


```
route(Town1,Town2,VisitedTowns,ResultVisitedTowns,Dis
    tance):-
   connected(Town1,X,Dist1),
   not(member(X,VisitedTowns)),

    route(X,Town2,[X|VisitedTowns],ResultVisitedTowns,D
    ist2),
   Distance=Dist1+Dist2.

predicates
  showAllRoutes(town,town)
  write_rote(town FirstTown,townList,distance)
  reverse_list(townList InList, townList Tmp, townList
    Reversed)
```

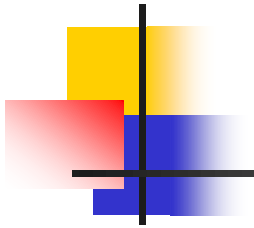


clauses

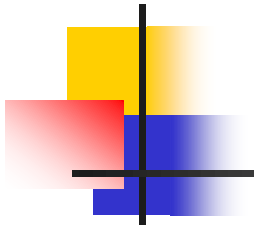
```
showAllRoutes(Town1,Town2):-
  write("All routes from ",Town1," to ",Town2," are:\n"),
  route(Town1,Town2, [Town1] ,VisitedTowns, Dist),
  write_rote(Town1,VisitedTowns,Dist),nl,
  fail.
showAllRoutes(_,_).
write_rote(Town1,[Town1|VisitedTowns],Dist):-
  !,
  Towns = [Town1|VisitedTowns],
  write("  ",Towns," --> ",Dist),nl.
```



```
write_rote(_,VisitedTowns,Dist):-
    reverse_list(VisitedTowns, [], VisitedTowns_Reversed),
    write("  ",VisitedTowns_Reversed," --> ",Dist),nl.

 reverse_List([],LIST,LIST):-!.
 reverse_List([H|SeenListRest],Interm,SeenList):-
    reverse_List(SeenListRest,[H|Interm],SeenList).

predicates
 showShortestRoutes(town,town)
 determ shorterRouteExist(town,town,distance)
```



```
clauses
  showShortestRoutes(Town1,Town2):-
    write("Shortest routes between ",Town1," to ",Town2,"
    is:\n"),
    route(Town1,Town2, [Town1] ,VisitedTowns, Dist),
    not(shorterRouteExist(Town1,Town2,Dist)),
    write_rote(Town1,VisitedTowns,Dist),nl,
    fail.
  showShortestRoutes(_,_).

  shorterRouteExist(Town1,Town2,Dist):-
    route(Town1,Town2, [Town1] ,_, Dist1),
    Dist1<Dist,!.
```

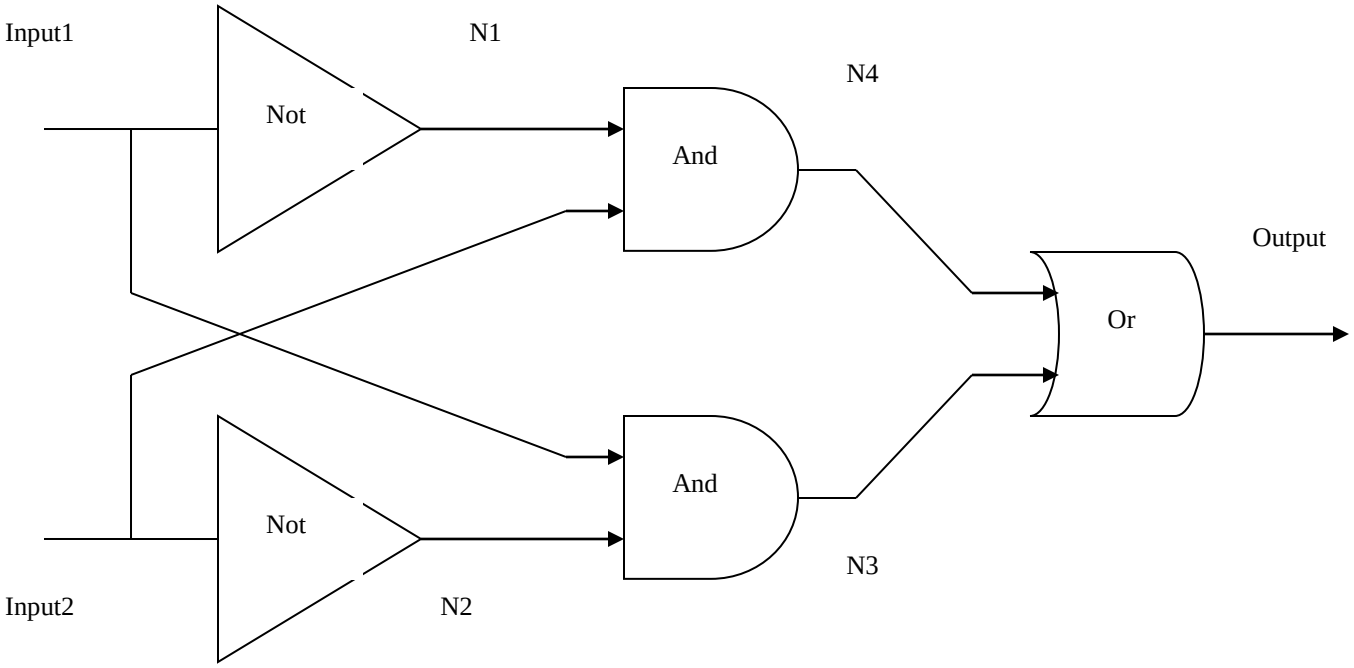


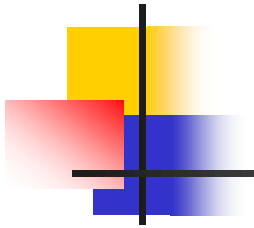
```
goal
  showAllRoutes("gordon", "tampa"),nl,
  showShortestRoutes("gordon", "tampa").
```

Chạy chương trình này cho kết quả là

```
All routes from gordon to tampa are:
  ["gordon","tampa"] --> 1000

  ["gordon","houston","kansas_city","tampa"] --> 370
  ["gordon","houston","tampa"] --> 650

Shortest routes between gordon to tampa is:
  ["gordon","houston","kansas_city","tampa"] --> 370
yes
```

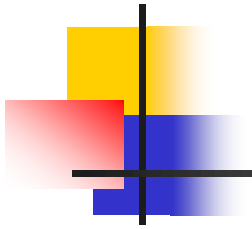
Ví dụ 5 : Mạch số xor



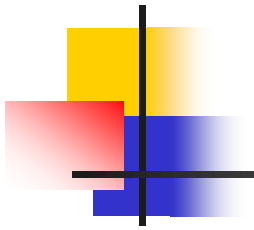


Chương trình Prolog sau là một ví dụ minh chứng kiểm tra cách vận hành của mạch.

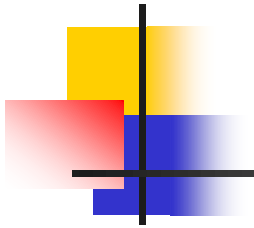
```
domains
  d = integer
 predicates
  not_(D,D)
  and_(D,D,D)
  or_(D,D,D)
  xor_(D,D,D)
```



```
clauses
  not_(1,0).
  not_(0,1).
  and_(0,0,0).
  and_(0,1,0).
  and_(1,0,0).
  and_(1,1,1).
  or_(0,0,0).
  or_(0,1,1).
  or_(1,0,1).
  or_(1,1,1).
```



```
xor_(Input1,Input2,Output):-
    not_(Input1,N1),
    not_(Input2,N2),
    and_(Input1,N2,N3),
    and_(Input2,N1,N4),
    or_(N3,N4,Output).

goal
  xor_(Input1,Input2,Output),    % Use GOAL mode to see
    results !!!
  format(Msg," xor_(%,%,%)",Input1,Input2,Output),
  write(Msg).
```



Chạy chương trình này cho kết quả là

xor_(1,1,0)Input1=1, Input2=1, Output=0,

Msg= xor_(1,1,0)

xor_(1,0,1)Input1=1, Input2=0, Output=1,

Msg= xor_(1,0,1)

xor_(0,1,1)Input1=0, Input2=1, Output=1,

Msg= xor_(0,1,1)

xor_(0,0,0)Input1=0, Input2=0, Output=0,

Msg= xor_(0,0,0)

4 Solutions .