# Chương 6:
# Biểu diễn tri thức và sử dụng luật

# Nội dung

- Tri thức khai báo và thủ tục
- Suy diễn tiến, suy diễn lùi
- Lập trình logic
- Giới thiệu ngôn ngữ Prolog

# Tri thức khai báo và thủ tục

- Biểu diễn dạng khai báo
  - Là một dạng biểu diễn mà ở đó tri thức được đặc tả nhưng sử dụng nó không được nêu ra.
  - Để sử dụng nó cần bổ sung một chương trình đặc tả cái gì sẽ được làm với tri thức và bằng cách nào
  - Ví dụ:
    - **Dạng đặc tả: một tập các "logical assertion"**
    - **Bộ phân giải có thể được hiểu như là cách để làm việc với tập assertions trên.**
  - Tập assertions như là DATA vào BỘ PHÂN GIẢI.
  - Một cách nhìn khác: tập assertions trên như là một PROGRAM. Ở đó: Luật giúp cho sự suy diễn xảy ra. Các con đường suy diễn khác nhau từ START – GOAL (hay ngược lại) được quan niệm như con đường thực thi trong chương trình.

# Tri thức khai báo và thủ tục (tt)

- Biểu diễn dạng thủ tục
  - Là một dạng biểu diễn mà thông tin điều khiển cần thiết cho việc sử dụng tri thức được nhúng vào chính tri thức đó.
  - Để sử dụng cần bổ sung một bộ thông dịch có thể thực thi các chỉ thị chứa trong tri thức.
  - Sự khác nhau cơ bản giữa tri thức thủ tục và khai báo nằm ở chổ: Thông tin điều khiển nằm ở đâu?

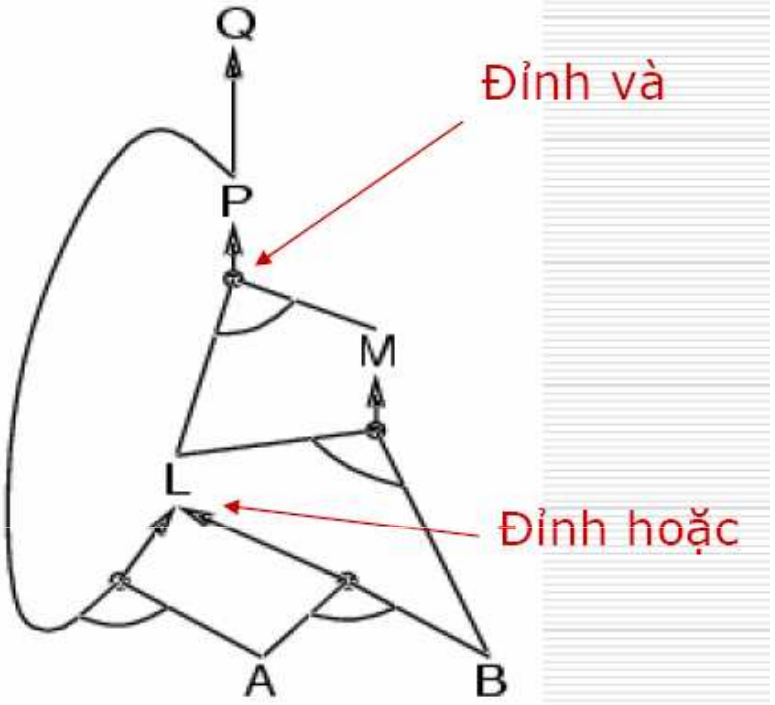
# Suy diễn tiến & suy diễn lùi

- Suy diễn tiến
  - Cho một tập luật (câu có dạng): $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$ và một tập các sự kiện $\{p,r,\ldots\}$
  - Hỏi một sự kiện $q$ có phải là một hệ quả của tập luật và tập sự kiện hay không?
  - Tìm tất cả các luật có giả thiết thuộc tập các sự kiện
    - Thêm kết luận vào tập các sự kiện
    - Tiếp tục các dẫn xuất khác

# Suy diễn tiến & suy diễn lùi (tt)

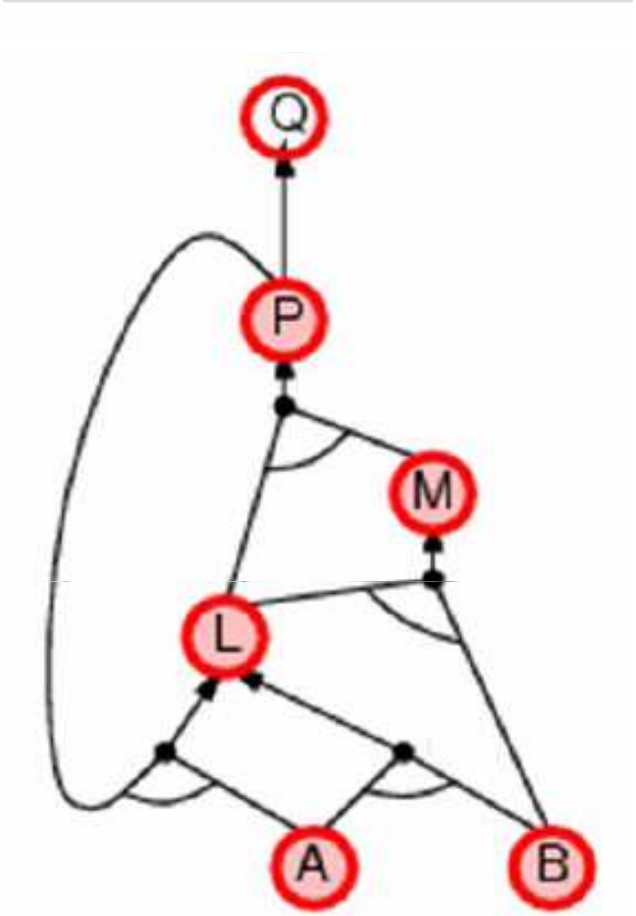
- Suy diễn tiến: ví dụ

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



# Suy diễn tiến & suy diễn lùi (tt)

- Suy diễn tiến: ví dụ



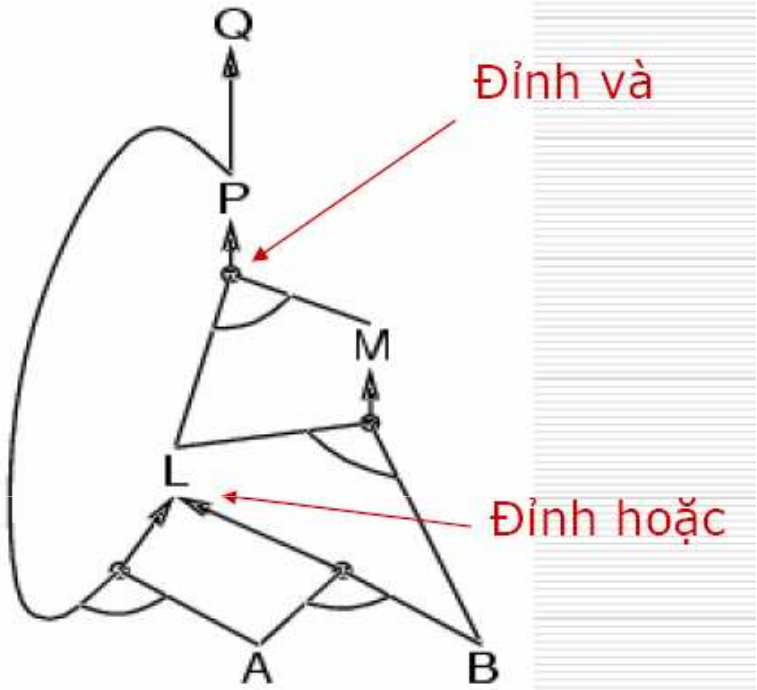
# Suy diễn tiến & suy diễn lùi (tt)

- Suy diễn lùi
  - Cho một tập luật (câu có dạng): $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$ và một tập các sự kiện {*p*,*r*,…}
  - Hỏi một sự kiện *p* có phải là một hệ quả của tập luật và tập sự kiện hay không?
  - Kiểm tra xem *p* có thuộc tập các sự kiện hay không
  - Nếu không tìm tất cả các luật có kết luận là p
    - Nếu giả thiết của các luật này là một hội, tiếp tục thủ tục (đệ quy) với từng thứa số của phép hội

# Suy diễn tiến & suy diễn lùi (tt)

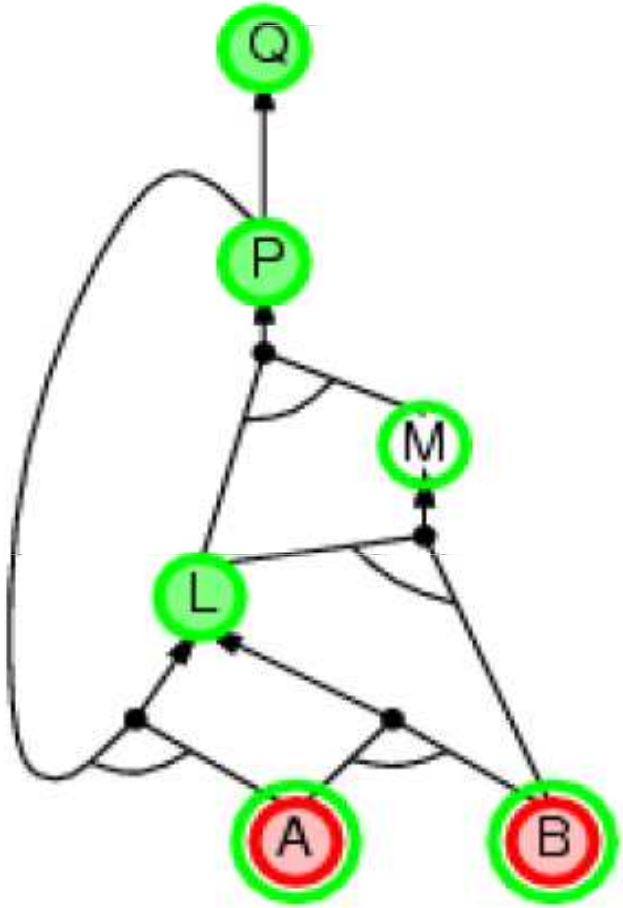
- Suy diễn lùi : ví dụ

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



# Suy diễn tiến & suy diễn lùi (tt)

- Suy diễn lùi : ví dụ



# Giới thiệu về ngôn ngữ Prolog

- Cấu trúc chương trình
  - Ngôn ngữ Prolog là ngôn ngữ lập trình suy luận trên cơ sở logic toán học để giải quyết các bài toán trong lĩnh vực trí tuệ nhân tạo.
  - Đặc điểm của ngôn ngữ là xử lý tri thức của các bài toán 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 toán tìm kiếm.
  - Ngôn ngữ cho phép liên kết với các ngôn ngữ khác như C, Pascal và Assembler.

# Giới thiệu về ngôn ngữ Prolog

- Cấu trúc chương trình (tt)

```
Domains
   /* domain declarations*/
Predicates
  /* predicate declarations */
clauses
    /*clauses ( rules and facts) */
goal
    /*subgoal_1
       subgoal_2 */
```

# Chương trình Prolog mẫu

```
domains
  nguoi = string
predicates
  cha(nguoi,nguoi)
  me(nguoi,nguoi)
  ong_noi(nguoi,nguoi)
  ong_ngoai(nguoi,nguoi)
clauses
  /*cac qui tac */
  ong_noi(X,Y):- cha(X,Z),cha(Z,Y).
  ong_ngoai(X,Y):- cha(X,Z),me(Z,Y).
  /* cac su kien */
  cha(nam,minh).
  cha(minh,lam).
  cha(long,giang).
  cha(long,thu).
  me(thu,phi).
```

# Phần domains : miền xác định

- Là phần định nghĩa kiểu mới dựa vào các kiểu đã biết
- Cú pháp định nghĩa kiểu
  - <DS kiểu mới> = <kiểu đã biết> hoặc
  - <DS kiểu mới> = <DS kiểu đã biết>

  Trong đó các kiểu mới phân cách bởi dấu «,», các kiểu đã biết phân cách bởi dấu «;»

# Phần domains (tt)

- VD

  Domains

  ten, tac_gia, nha_xb, dia_chi = string

  nam, thang, so_luong = integer

  dien_tich = real

  nam_xb = nxb(thang, nam)

  do_vat = sach(tac_gia, ten, nha_xb, nam_xb); xe(ten, so_luong); nha(dia_chi, dien_tich)

# Phần Predicates : vị từ

- Là phần bắt buộc phải có
- Phần predicates cần phải khai báo đầy đủ các vị từ sử dụng trong phần Clauses
- Cú pháp

  <Tên vị từ> (<danh sách các kiểu>)

  Các kiểu được phân cách nhau bởi «,»
- VD

  Predicates

  so_huu (ten, do_vat)

  so_nguyen_to(integer)

# Phần Clauses : luật

- Là phần bắt buộc phải có, dùng để mô tả các sự kiện và các luật
- Sử dụng các vị từ đã khai báo trong phần predicates
- Cú pháp

  <Tên vị từ>(<danh sách các tham số>) <kí hiệu>

  <Tên vị từ 1>(<danh sách các tham số 1>) <kí hiệu>

  ……

  <Tên vị từ N>(<danh sách các tham số N>) <kí hiệu>

  Các ký hiệu bao gồm :- (điều kiện nếu);

  , (điều kiện và)

  ; (điều kiện hoặc)

  . (kết thúc vị từ)

# Phần Clauses (tt)

- VD

  Clauses

  so_nguyen_to(2):-!.

  so_nguyen_to(N):-N>0,

  so_nguyen_to(M),

  M<N,

  N MOD M <>0.

  so_huu(“Nguyen Van A”, sach(“Do Xuan Loi”, “Cau truc DL”, “Khoa hoc Ky thuat”, nxb(8,1985))).

# Phần goal

- Bao gồm các mục tiêu mà ta yêu cầu Prolog xác định và tìm kết quả
- Không bắt buộc phải có
- Nếu được viết sẵn trong CT thì đó gọi là goal nội; Nếu không, khi chạy CT Prolog sẽ yêu cầu ta nhập goal vào, goal ngoại
- VD
  - Constants
  - Pi = 3.141592653

# VD chương trình prolog

```
domains
  so_nguyen = integer
predicates
  so_nguyen_to(so_nguyen)
Clauses
  so_nguyen_to(2):-!.
  so_nguyen_to(N):-N>0,
  so_nguyen_to(M),
  M<N,
  N MOD M <>0.
goal
  so_nguyen_to(13). /* goal nội */
```

# Bộ ký tự và từ khóa

- Prolog dùng bộ ký tự sau:
  - Các chữ cái và chữ số (A – Z, a – z, 0 – 9);
  - Các toán tử (+, -, *, /, <, =, >)
  - Các ký hiệu đặc biệt
- Một vài từ khóa
  - Trace: Khi có từ khoá này ở đầu chương trình, thì chương trình được thực hiện từng bước để theo dõi
  - Fail: Khi ta dùng goal nội, để nhận về tất cả các kết quả khi chạy goal nội, ta dùng toán tử Fail
  - ! hay còn gọi là nhát cắt, nhận chỉ một kết quả từ goal ngoại, ta dùng ký hiệu !

# Kiểu dữ liệu chuẩn

- Kiểu do prolog định nghĩa sẵn: char,integer, real string và symbol
- char: ký tự, hằng phải nằm trong dấu nháy: ‘a’, ‘#’
- integer: -32768 đến 32767
- real: số thực
- string: chuỗi ký tự, hằng chuỗi ký tự nằm trong dấu nháy kép; ”prolog”

# Kiểu do người dùng định nghĩa

- Kiểu mẩu tin
  - Cú pháp <tên kiểu mẩu tin> = tên mẩu tin (danh sách các kiểu phần tử)

```
Domains
ten, tac_gia, nha_xb, dia_chi = string
nam, thang, so_luong = integer
dien_tich = real
nam_xb = nxb(thang, nam)
```

# Kỹ thuật đệ quy

- Sử dụng đệ quy khi một vị từ được định nghĩa nhờ vào chính vị từ đó
- Trường hợp dừng được thể hiện bằng một sự kiện
- VD

```
Predicates
    Facto (integer, integer)
Clauses
    Facto(0,1):- !.
    Facto(N, Y) :- N>0,M = N–1, facto(M, Z), Y=N*Z.
```

# Các hàm xuất nhập chuẩn

- Xuất ra màn hình
  - write( Arg1, Arg2, … ,Argn) in ra màn hình giá trị của các đối số.
  - writef(đinh_dang, Arg1, Arg2, … ,Argn) in ra màn hình giá trị của các đối số theo định_dạng
  - Các định_dạng
    - “%d”: In số thập phân bình thường; đối số phải là char hoặc integer
    - “%c”: Đối số là một số integer, in ký tự có mã Ascci là đối số đó, chẳng hạn writef(“%c”,65) được A
    - “%e”: In số thực dưới dạng lũy thừa của 10
    - “%x”: In số Hexa; đối số phải là char hoặc integer
    - “%s”: In một chuỗi hoặc một symbol

# Các hàm xuất nhập chuẩn (tt)

- Nhập vào từ bàn phím
  - Readln(X): Nhập một chuỗi ký tự vào biến X
  - ReadInt(X): Nhập một số nguyên vào biến X
  - ReadReal(X): Nhập một số thực vào biến X
  - ReadChar(X): Nhập vào một ký tự vào biến X

# Ví dụ …

- Tháp Hà nội
- Người nông dân
- Random
- …