Chương 7: Logic vị từ

Giảng viên: Nguyễn Văn Hòa

Khoa CNTT - ĐH An Giang

Nội dung

- Logic bậc nhất (First Order Logic – FOL)
- Cú pháp và ngữ nghĩa
- Các lượng từ
- Hợp giải với logic vị từ
- Dạng mệnh đề
- Lập trình logic Turbo Prolog

Tại sao sử dụng logic vị từ

- Logic mệnh đề chỉ xử lý trên các sự kiện, là các khẳng định có giá trị đúng hoặc sai
- Logic vị từ (logic bậc nhất) cho phép chúng ta nói về các đối tượng, tính chất của chúng, quan hệ giữa chúng, phát biểu về một hay tất cả các đối tượng nào đó mà không cần liệt kê trong chương trình
- Các câu không thể biểu diễn bằng logic mệnh đề nhưng có thể biểu diễn bằng logic vị từ
  - Socrates là người nên socrates phải chết

Cú pháp của logic vị từ

- Term (Tham chiếu đối tượng)
  - Ký kiệu bằng: Lan, Tuan, DHAG
  - Biến: x, y, a
  - Ký hiệu hàm áp dụng cho một hay nhiều term: f(x), tuoi(Lan), sinh-vien(Tuan)
- Câu:
  - Một ký hiệu vi từ (predicate) áp dụng cho một hay nhiều term: thuoc(Lan, DHAG), la-anh-em(Lan,Tuan)…
  - t1=t2
  - Nếu x là biến và ϕ là câu thì $\forall x\ \phi$, $\exists x\ \phi$ cũng là câu
  - Các câu tạo từ các câu khác với phép nối câu $\neg\ \wedge\ \vee\ \rightarrow$

Ngữ nghĩa của logic vị từ

- Có một tập ngầm định U các đối tượng mà một câu FOL phát biểu trên đó, U được gọi là tập phát biểu
- Term tham chiếu đến đối tượng trong U
  - Hằng tham chiếu đến một đối tượng cụ thể
  - Biến tham chiếu đến một đối tượng nào đó (cần lượng từ hóa)
  - Hàm tham chiếu đối một đối tượng thông qua đối tượng khác
- Câu là phát biểu trên các đối tượng
  - Vị từ thể hiện tính chất, hoặc quan hệ giữa các đối tượng
  - Lượng từ giúp phát biểu số lượng đối tượng: $\forall$, $\exists$

Lượng từ với mọi

- Cú pháp: ∀ <biến> <câu>
  - Sinh viên CNTT thì thông minh

  ∀x sinhvien(x, CNTT) → thongminh(x)
- ∀x P đúng trong một miềm phát biểu U, nếu và chỉ nếu P đúng với **mọi** x thuộc U
- Nghĩa là tương đương với phép nối liền (∧) của các đối tượng trong U

  sinhvien(Lan, CNTT) → thongminh(Lan)

  ∧ sinhvien(Tuan, CNTT) → thongminh(Tuan)

  ∧ sinhvien(Nam, CNTT) → thongminh(Nam)

Lượng từ tồn tại

- Cú pháp: ∃ <biến> <câu>
  - Sinh viên CNTT thì thông minh

  ∃ x sinhvien(x, CNTT) → thongminh(x)
- ∃x. P đúng trong một miềm phát biểu U, nếu và chỉ nếu P đúng với **một** x thuộc U
- Nghĩa là tương đương với phép nối rời (∨) của các đối tượng trong U

  sinhvien(Lan, CNTT) → thongminh(Lan)

  ∨ sinhvien(Tuan, CNTT) → thongminh(Tuan)

  ∨ sinhvien(Nam, CNTT) → thongminh(Nam)

Viết FOL

- Mèo là động vật có vú

  $\forall x.\text{Mèo}(x) \rightarrow \text{Động-vật-có-vú}(x)$
- Lan là sinh viên học giỏi

  $\text{Sinh-viên}(\text{Lan}) \wedge \text{Học-giỏi}(\text{Lan})$
- Cháu là con của anh em

  $\forall x,y.\text{Cháu}(x,y) \leftrightarrow \exists z.(\text{Anh-em}(z,y) \wedge \text{Con}(x,z))$
- Bà ngoại là mẹ của mẹ

  $\forall x, \exists y.\ \text{Bà-ngoại}(x,y) \leftrightarrow \exists z.(\text{Mẹ}(x,z) \wedge \text{Mẹ}(z,y))$
- Mọi người đều yêu ai đó

  $\forall x, \exists y.\text{Yêu}(x, y)$

Chứng minh và suy dẫn

- Suy dẫn xuất pháp từ khái niệm tổng quát của phép “kéo theo”
- Không thể tính toán trực tiếp từ cách liệt kê
- Do đó ta sẽ phải làm theo cách chứng minh
- Trong FOL, nếu KB suy dẫn được S thì sẽ có một tập hữu hạn chứng minh của s từ KB

Hợp giải bậc nhất

$\forall x , P(x) \rightarrow Q(x)$

$P(A)$

$Q(A)$

Tam đoạn luận:
Mọi người đều chết
Socrate là người
Socrate chết

$\forall x, \neg P(x) \vee Q(x)$

$P(A)$

$Q(A)$

Tương đương theo định nghĩa của phép
Suy ra

$\neg P(A) \vee Q(A)$

$P(A)$

$Q(A)$

Thay A vào x, dẫn đúng khi đó
Hợp giải mệnh đề

Hai vấn đề mới đặt ra:
- Biến đổi FOL thành dạng mệnh đề (clausal form)
- Hợp giải biến

Dạng mệnh đề (clausal form)

- Cấu trúc ngoài giống CNF
- Không có lượng từ

$$\forall x.\ \exists y.\ P(x) \Rightarrow R(x,y)$$

$$\neg P(x) \ \vee R(x,F(y))$$

Biến đổi về dạng mệnh đề

- Câu sau được dùng làm ví dụ trong thủ tục đưa về clause form.
  - "All Romans who know Marcus either hate Caesar or think that anyone who hates anyone is crazy"
  - $\forall$ X: [roman(X) ^ know(X, Marcus)] $\rightarrow$
    [hate(X, Ceasar) v
    ($\forall$ Y: $\exists$ Z: hate(Y,Z) $\rightarrow$ thinkcrazy(X,Y))]

Biến đổi về dạng mệnh đề…

1. Loại bỏ →

   dùng tương đương: a→b = ¬a v b

- Ví dụ
- ∀ X: [roman(X) ^ know(X, Marcus)] →
  [hate(X, Ceasar) v
  (∀ Y: ∃ Z: hate(Y,Z) → thinkcrazy(X,Y))]
- ∀ X: ¬[roman(X) ^ know(X, Marcus)] v
  [hate(X, Ceasar) v
  (∀ Y: ∃ Z: hate(Y,Z) → thinkcrazy(X,Y))]

Biến đổi về dạng mệnh đề…

2. Thu giảm tầm vực của ¬ vào đến mức term.
   - Dùng tương đương:
     ¬(¬p) = p
   - De Morgan:
     ¬(a v b) = ¬a ^ ¬b
     ¬(a ^ b) = ¬a v ¬b
   - Tương đương lượng từ:
     ¬ ∀ X: P(X) = ∃ X: P(X)
     ¬ ∃: P(X) = ∀ X: P(X)
- Áp dung cho ví dụ trước
   - ∀ X: [¬roman(X) v ¬know(X, Marcus)] v [hate(X,Ceasar)
     v
     (∀ Y: ∃ Z: ¬hate(Y,Z) v thinkcrazy(X,Y))]

Biến đổi về dạng mệnh đề…

3. Chuẩn hoá các biến để các lượng từ chỉ ràng buộc 1 biến duy nhất.
   - Biến đổi như VD sau:

   $\forall X: P(X) \vee \forall X: Q(X) = \forall X: P(X) \vee \forall Y: Q(Y)$

4. Chuyển lượng từ về bên trái. Chú ý, không chuyển thứ tự của chúng
   - Ví dụ: tiếp bước 2.

   $\forall X: \forall Y: \exists Z: [\neg roman(X) \vee \neg know(X, Marcus)] \vee [hate(X, Ceasar) \vee (\neg hate(Y,Z) \vee thinkcrazy(X,Y))]$

Biến đổi về dạng mệnh đề…

- 5. Loại bỏ lượng từ tồn tại : Sử dụng hàm skolem
  - Hàm skolem:

    $\forall X: \forall Y: \exists Z : P(X,Y,Z) = \forall X: \forall Y: P(X,Y,f(X,Y))$
  - Biến của lượng từ tồn tại được thay là hàm theo những biến của lượng từ với mọi trước nó
- Bỏ qua các lượng từ (với mọi) còn lại ở bước 5. xem như mọi biến đều bị tác động bởi lượng từ với mọi ($\forall$)
  - Ví dụ: tiếp bước 4

    [¬roman(X) v ¬know(X, Marcus)] v [hate(X, Ceasar) v (¬hate(Y,Z) v thinkcrazy(X,Y))]

Biến đổi về dạng mệnh đề…

7. Chuyển hội chuẩn (Conjunctive Normal Form - CNF)

- Một chuỗi các mệnh đề kết nối nhau bằng quan hệ AND (^). Mỗi mệnh đề có dạng một tuyển OR (v) của các biến mệnh đề.
- Dùng phép phân phố giữa v và ^
- Dạng thường gặp:

  (a ^ b) v c = (a v c) ^ (b v c)

  (a ^ b) v (c ^ d) = (a v c) ^ (a v d) ^ (b v c) ^ (b v d)
- Ví dụ: tiếp bước 6

  ¬roman(X) v ¬know(X, Marcus) v

  hate(X, Ceasar) v ¬hate(Y,Z) v thinkcrazy(X,Y)

Biến đổi về dạng mệnh đề…

8. Tách riêng các clause trong CNF ở trên
   - Nếu có clause form:

     (a v ¬b) ^ (¬a v c v d) ^ (a v ¬c v e)
   - Thì được tách riêng thành các clause:

   1. (a v ¬b)
   2. (¬a v c v d)
   3. (a v ¬c v e)
- Đưa các lượng từ về từng clause
  - (∀ X: P(X) ^ Q(X)) = ∀ X: P(X) ^ ∀ X: Q(X)

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

```
predicates
  ktnt(integer,integer)
  tieptuc(integer,real,real,integer)
clauses
  ktnt(1,_):-write("Day la so nguyen to").
  ktnt(2,_):-write("Day la so nguyen to").
  ktnt(N,M):-N1=N-1,
    N2=M/N1,N3=round(M/N1),tieptuc(N1,N2,N3,M).
  tieptuc(_,N2,N3,_):-N2=N3, write("Day khong phai la so
    nguyen to").
  tieptuc(N1,N2,N3,M):-N2<>N3,ktnt(N1,M).
goal
  clearwindow,write("Nhap N:"),readint(N),ktnt(N,N).
```

Các nguyên tắc của NN Prolog

- Có 2 nguyên tắc: đồng nhất và quay lui
- Đồng nhất
  - Một quan hệ có thể đồng nhất với một quan hệ nào đó cùng tên, cùng số lượng tham số, các đại lượng con cũng đồng nhất theo từng cặp
  - Một hằng có thể đồng nhất với một hằng
  - Một biến có thể đồng nhất với một hằng nào đó và có thể nhận luôn giá trị hằng đó

Các nguyên tắc của NN Prolog (tt)

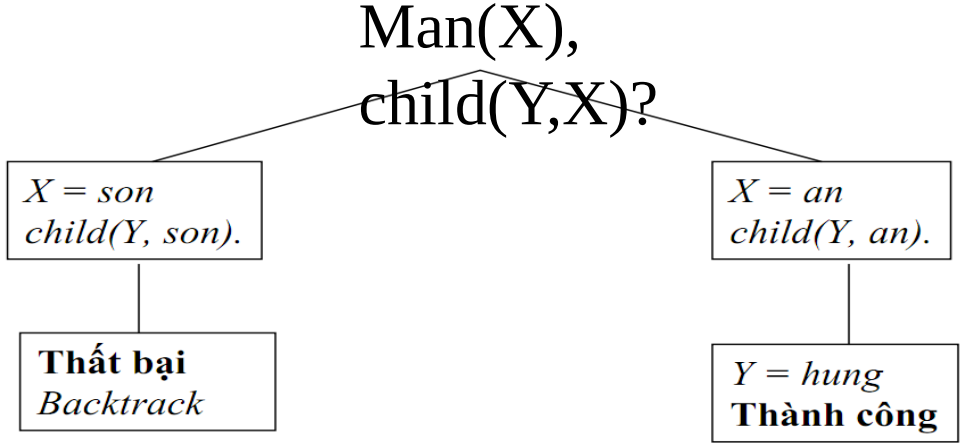
- Nguyên tắc quay lui (backtract, backtracting)
  - Cần chứng minh Goal : :- g_1, g_2, …, g_{j-1}, g_j, …, g_n
  - Kiểm chứng từ trái sang phải, đến g_i là sai, hệ thống cần phải qui lui lại g_{i-1}
  - Ví dụ

    man(son)

    man(an)

    child(hung,an)

    goal: man(X), child(Y,X)?



Cây hợp giải

- Xét các mệnh đề sau đây

sister(X,Y) :- child(X,P),child(Y,P),

woman(X), X<>Y.

woman(hien).

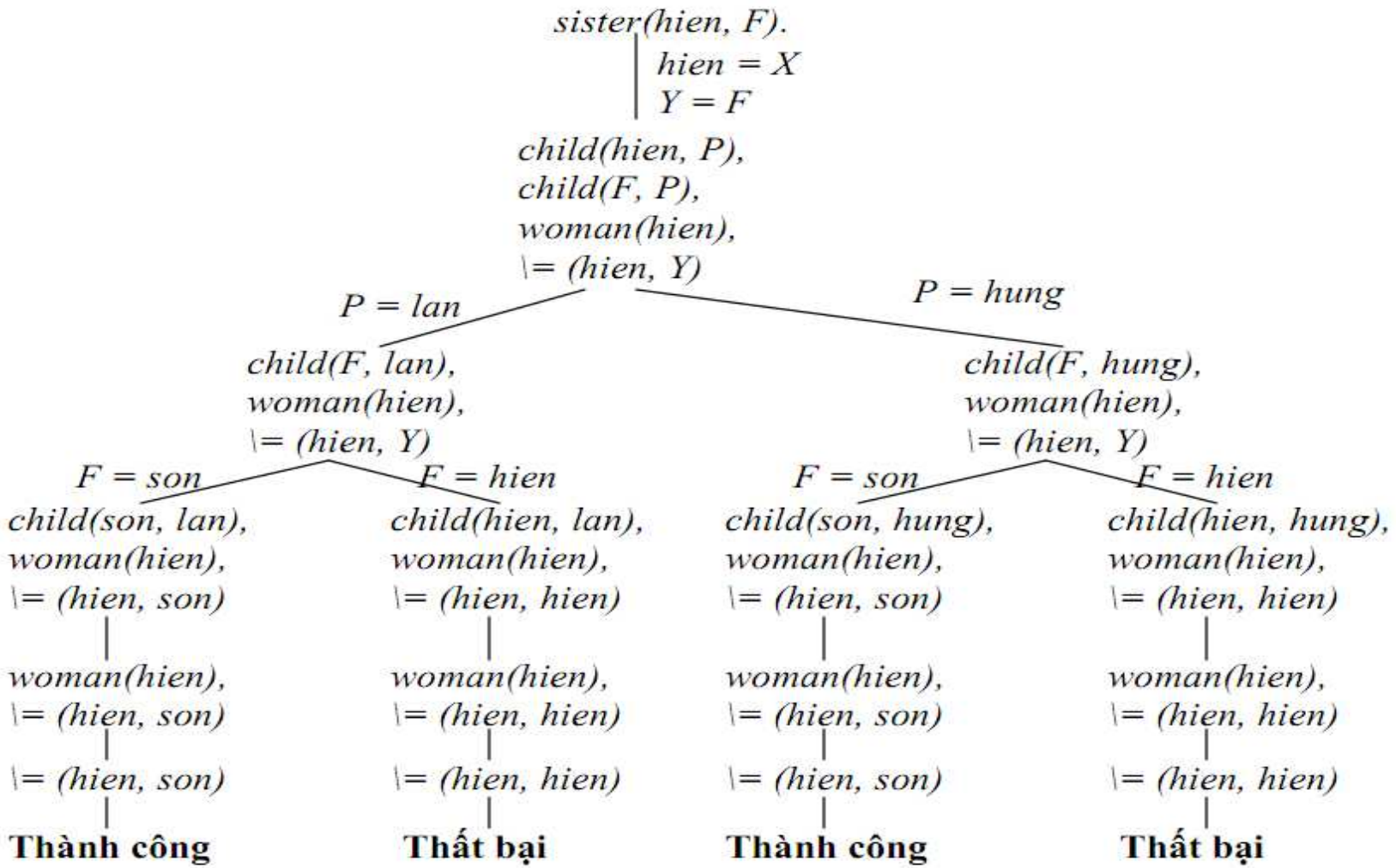
child(son,lan).

child(hien,lan).

child(son,hung).

child(hien,hung).

goal: sister(hien, F)?


sister(hien, F).
hien = X
Y = F
child(hien, P),
child(F, P),
woman(hien),
\= (hien, Y)
P = lan
P = hung
child(F, lan),
woman(hien),
\= (hien, Y)
child(F, hung),
woman(hien),
\= (hien, Y)
F = son
F = hien
F = son
F = hien
child(son, lan),
woman(hien),
\= (hien, son)
child(hien, lan),
woman(hien),
\= (hien, hien)
child(son, hung),
woman(hien),
\= (hien, son)
child(hien, hung),
woman(hien),
\= (hien, hien)
woman(hien),
\= (hien, son)
woman(hien),
\= (hien, hien)
woman(hien),
\= (hien, son)
woman(hien),
\= (hien, hien)
\= (hien, son)
\= (hien, hien)
\= (hien, son)
\= (hien, hien)
Thành công
Thất bại
Thành công
Thất bại

Cây hợp giải

- Trong ví dụ trên, chúng ta tìm thấy 2 câu trả lời giống hệt nhau được thực hiện bởi 2 con đường khác nhau (cha và mẹ).
- Để tránh điều đó, chúng ta viết lại

sister(X,Y) :- mother(M,X), father(F,X), mother(M,Y),
father(F,Y), woman(X), X<>Y.

woman(hien).

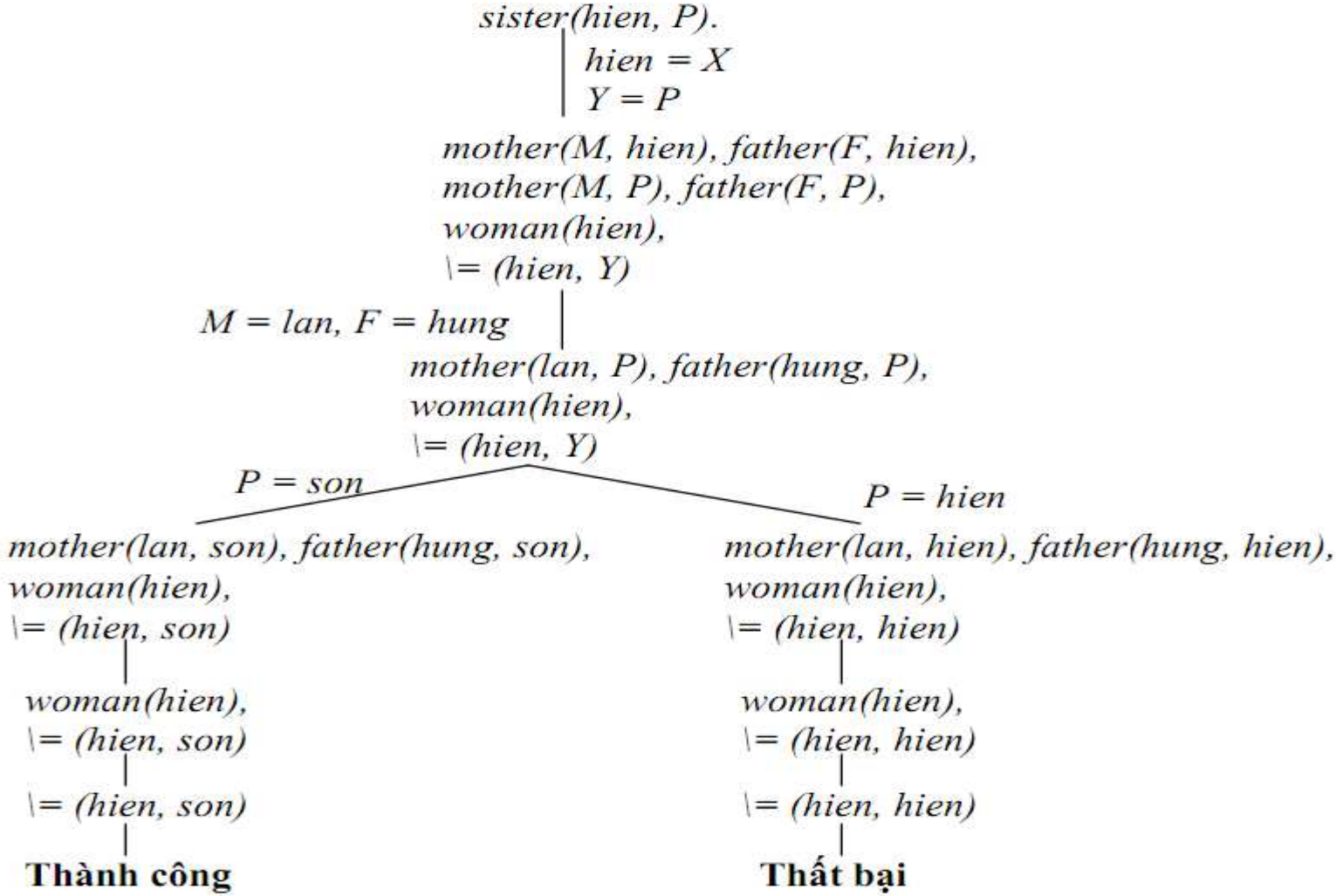
mother(lan,son).

mother(lan,hien).

father(hung,son).

father(hung,hien).

goal: sister(hien, P)


sister(hien, P).
hien = X
Y = P
mother(M, hien), father(F, hien),
mother(M, P), father(F, P),
woman(hien),
\= (hien, Y)
M = lan, F = hung
mother(lan, P), father(hung, P),
woman(hien),
\= (hien, Y)
P = son
P = hien
mother(lan, son), father(hung, son),
woman(hien),
\= (hien, son)
woman(hien),
\= (hien, son)
\= (hien, son)
Thành công
mother(lan, hien), father(hung, hien),
woman(hien),
\= (hien, hien)
woman(hien),
\= (hien, hien)
\= (hien, hien)
Thất bại

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

Link demo & software

- Clause Deduction
  - AISpace
- SWI-Prolog
  - www.swi-prolog.org
- GNU-Prolog
  - gnu-prolog.inria.fr