Chương 3

TRI THỨC VÀ LẬP LUẬN

Chương 3 trình bày các đặc trưng của ngôn ngữ biểu diễn tri thức. Chúng ta sẽ nghiên cứu logic mệnh đề, một ngôn ngữ biểu diễn tri thức rất đơn giản, có khả năng biểu diễn hẹp, nhưng thuận lợi cho ta làm quen với nhiều khái niệm quan trọng trong logic, đặc biệt trong logic vị từ cấp một.

3.1. LOGIC MỆNH ĐỀ

3.1.1. Tri thức và biểu diễn tri thức

Tri thức được mô tả dưới dạng các câu trong ngôn ngữ biểu diễn tri thức. Mỗi câu có thể xem như sự mã hóa của một sự hiểu biết của chúng ta về thế giới hiện thực. Ngôn ngữ biểu diễn tri thức (cũng như mọi ngôn ngữ hình thức khác) gồm hai thành phần cơ bản là cú pháp và ngữ nghĩa.

Cú pháp của một ngôn ngữ bao gồm các ký hiệu về các quy tắc liên kết các ký hiệu (các luật cú pháp) để tạo thành các câu (công thức) trong ngôn ngữ. Các câu ở đây là biểu diễn ngoài, cần phân biệt với biểu diễn bên trong máy tính. Các câu sẽ được chuyển thành các cấu trúc dữ liệu thích hợp được cài đặt trong một vùng nhớ nào đó của máy tính, đó là biểu diễn bên trong. Bản thân các câu chưa chứa đựng một nội dung nào cả, chưa mang một ý nghĩa nào cả.

Ngoài hai thành phần cú pháp và ngữ nghĩa, ngôn ngữ biểu diễn tri thức cần được cung cấp cơ chế suy diễn. Một luật suy diễn (rule of inference) cho phép ta suy ra một công thức từ một tập nào đó các công thức. Chẳng hạn, trong logic mệnh đề, luật Modus Ponens từ hai công thức A và A⇒ịB suy ra công thức B. Chúng ta sẽ hiểu lập luận hoặc suy diễn là một quá trình áp dụng các luật suy diễn để từ các tri thức trong cơ sở tri thức và các sự kiện ta nhận được các tri thức mới. Như vậy chúng ta xác định:

- ✓ Ngôn ngữ biểu diễn tri thức = Cú pháp + Ngữ nghĩa + Cơ chế suy diễn.
- ✓ Một ngôn ngữ biểu diễn tri thức tốt cần phải có khả năng biểu diễn rộng, tức là có thể mô tả được mọi điều mà chúng ta muốn nói. Nó cần phải hiệu quả theo nghĩa là, để đi tới các kết luận, thủ tục suy diễn đòi hỏi ít thời gian tính toán và ít không gian nhớ. Người ta cũng mong muốn ngôn ngữ biểu diễn tri thức gần với ngôn ngữ tự nhiên.

Mệnh đề là câu, thường mang giá trị hoặc đúng hoặc sai. Giá trị này được gọi là chân lý của mệnh đề. Logic mệnh đề gán một biến kí hiệu cho mệnh đề, như phép gán sau:

A= "Trời mưa"

Khi một bài toán phát biểu theo logic mệnh đề đòi hỏi phải kiểm tra tính đúng đắn của câu trên, người ta kiểm tra giá trị của A.

3.1.2. Cú pháp và ngữ nghĩa của logic mệnh đề

3.1.2.1. Cú pháp

Cú pháp của logic mệnh đề rất đơn giản, nó cho phép xây dựng nên các công thức. Cú pháp của logic mệnh đề bao gồm tập các ký hiệu và tập các luật xây dựng công thức.

Các ký hiệu

- ✓ Hai hằng logic True và False.

- ✓ Các ký hiệu mệnh đề (còn được gọi là các biến mệnh đề): P, Q, …
- ✓ Các kết nối logic ∧Ù, ∨Ú, ¬, ⇒ị, ⇔Û.
- ✓ Các dấu mở ngoặc (và đóng ngoặc).

Các quy tắc xây dựng các công thức

- ✓ Các biến mệnh đề là công thức.
- ✓ Nếu A và B là công thức thì:
- ✓ (A∧ÙB) (đọc "A hội B" hoặc "A và B")
- ✓ (A∨ÚB) (đọc "A tuyển B" hoặc "A hoặc B")
- ✓ (¬A) (đọc "phủ định A")
- ✓ (A⇒ịB) (đọc "A kéo theo B" hoặc "nếu A thì B")
- ✓ (A⇔ÛB) (đọc "A và B kéo theo nhau") là các công thức.

Sau này để cho ngắn gọn, ta sẽ bỏ đi các cặp dấu ngoặc không cần thiết. Chẳng hạn, thay cho ((A∨ÚB)∧ÙC) ta sẽ viết là (A∨ÚB)∧ÙC.

Các công thức là các ký hiệu mệnh đề sẽ được gọi là các câu đơn hoặc câu phân tử. Các công thức không phải là câu đơn sẽ được gọi là câu phức hợp. Nếu P là ký hiệu mệnh đề thì P và TP được gọi là literal, P là literal dương, còn TP là literal âm. Câu phức hợp có dạng A1∨Ú...∨ÚAm trong đó Ai là các literal sẽ được gọi là câu tuyển (clause).

Ví dụ 3.1:

Mệnh đề thực tế	**Mệnh đề logic**
✓ "Nếu trời mưa thì bầu trời có mây" ✓ Trời đang mưa Vậy ⇒ị Bầu trời có mây	✓ P="Trời mưa" ✓ Q= "Bầu trời có mây" Ta có hai phát biểu sau đúng: ✓ P⇒ị Q ✓ P Vậy theo luật suy diễn ⇒ị Q là đúng. Nghĩa là: "Bầu trời có mây"

Ví dụ 3.2:

Mệnh đề thực tế	**Mệnh đề logic**
✓ "Nếu NAM có nhiều tiền thì NAM đi mua sắm" ✓ "Nam KHÔNG đi mua sắm" Vậy ⇒ị Nam KHÔNG có nhiều tiền	✓ P= "Nam có nhiều tiền" ✓ Q= "Nam đi mua sắm" Ta có hai phát biểu sau đúng: ✓ P⇒ị Q ✓ ¬ Q Vậy theo luật suy diễn ⇒ị ¬ P là đúng. Nghĩa là: "Nam KHÔNG có nhiều tiền"

3.1.2.2. Ngữ nghĩa

Ngữ nghĩa của logic mệnh đề cho phép ta xác định thiết lập ý nghĩa của các công thức trong thế giới thực. Điều này được thực hiện bằng cách kết hợp mệnh đề với sự kiện nào đó trong thế giới hiện thực. Chẳng hạn, ký hiệu mệnh đề P có thể ứng với sự kiện

"Paris là thủ đô nước Pháp" hoặc bất kỳ một sự kiện nào khác. Bất kỳ một sự kết hợp các kí hiệu mệnh đề với các sự kiện trong thế giới thực được gọi là một minh họa (interpretation). Ví dụ minh họa của kí hiệu mệnh đề P có thể là một sự kiện (mệnh đề) "Paris là thủ đô nước Pháp". Một sự kiện chỉ có thể đúng hoặc sai. Chẳng hạn, sự kiện "Paris là thủ đô nước Pháp" là đúng, còn sự kiện "Hà Nội là thủ đô nước Pháp" là sai.

Một cách chính xác hơn, cho ta hiểu một minh họa là một cách gán cho mỗi ký hiệu mệnh đề một giá trị chân lý True hoặc False. Trong một minh họa, nếu kí hiệu mệnh đề P được gán giá trị chân lý True/False (P ←True/ P←False) thì ta nói mệnh đề P đúng/sai trong minh họa đó. Trong một minh họa, ý nghĩa của các câu phức hợp được xác định bởi ý nghĩa của các kết nối logic. Chúng ta xác định ý nghĩa của các kết nối logic trong các bảng chân lý 3.1.

Bảng 3.1: Bảng chân lý của các kết nối logic

P	Q	¬P	P∧ÙQ	P ∨ Q	P ⇒ịQ	P <=>Q
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Ý nghĩa của các kết nối logic ∧Ù, ∨ và ¬ được xác định như các từ "và","hoặc" và "phủ định" trong ngôn ngữ tự nhiên.

Phép kéo theo P ⇒ị Q (P kéo theo Q) với P là giả thiết, còn Q là kết luận. Khi P là đúng và Q là đúng thì câu "P kéo theo Q" là đúng, còn khi P là đúng Q là sai thì câu "P kéo theo Q" là sai. Nhưng nếu P sai và Q đúng, hoặc P sai Q sai thì "P kéo theo Q" là đúng hay sai? Nếu xuất phát từ giả thiết sai, thì không thể khẳng định gì về kết luận. Không có lý do gì để nói rằng, nếu P sai và Q đúng hoặc P sai và Q sai thì "P kéo theo Q" là sai. Do đó trong trường hợp P sai thì "P kéo theo Q" là đúng dù Q là đúng hay Q là sai.

Bảng chân lý cho phép ta xác định ngẫu nhiên các câu phức hợp. Chẳng hạn ngữ nghĩa của các câu P∧ÙQ trong minh họa {P ← True , Q←False } là False. Việc xác định ngữ nghĩa của một câu (P ∨ Q) ∧Ù ¬S trong một minh họa được tiến hành như sau: đầu tiên ta xác định giá trị chân lý của P ∨ Q và ¬S , sau đó ta sử dụng bảng chân lý ∧Ù để xác định giá trị (P∨Q) ∧Ù¬S

- ✓ Một công thức được gọi là thoả được (satisfiable) nếu nó đúng trong một minh họa nào đó. Chẳng hạn công thức (PvQ) ∧Ù¬S là thoả được, vì nó có giá trị True trong minh họa {P ← True, Q←False, S←True}.
- ✓ Một công thức được gọi là vững chắc nếu nó đúng trong mọi minh họa chẳng hạn câu P ∨¬P là vững chắc.
- ✓ Một công thức được gọi là không thoả được, nếu nó là sai trong mọi minh họa. Chẳng hạn công thức P ∧Ù ¬P.

Một mô hình của một công thức là một minh họa sao cho công thức là đúng trong minh họa này. Như vậy một công thức thoả được là công thức có một mô hình. Chẳng hạn, minh họa {P ← False , Q ← False , S←True } là một mô hình của công thức (P ⇒ịQ) ∧Ù S.

Bằng cách lập bảng chân lý (phương pháp bảng chân lý) là ta có thể xác định được một công thức có thoả được hay không. Trong bảng này, mỗi biến mệnh đề đứng đầu với một cột, công thức cần kiểm tra đứng đầu một cột, mỗi dòng tương ứng với một minh họa. Bảng 3.2 là bảng chân lý cho công thức (P=>Q) ∧ÙS. Trong bảng chân lý này ta cần đưa vào các cột phụ ứng với các công thức con của các công thức cần kiểm tra để việc tính giá trị của công thức này được dễ dàng. Từ bảng chân lý ta thấy rằng công thức (P⇒ịQ) ∧ÙS là thoả được nhưng không vững chắc.

Bảng 3.2: Bảng chân lý cho công thức (P⇒ịQ) ∧ÙS

P	Q	S	P⇒ịQ	(P⇒ịQ) ∧ÙS
False	False	False	True	False
False	False	True	True	True
False	True	False	True	False
False	True	True	True	True
True	False	False	False	False
True	False	True	False	False
True	True	False	True	False
True	True	True	True	True

Một công thức chứa n biến, thì số các minh họa của nó là 2^n, tức là bảng chân lý có 2^n dòng. Như vậy việc kiểm tra một công thức có thoả được hay không bằng phương pháp bảng chân lý, đòi hỏi thời gian mũ.

Chúng ta sẽ nói rằng (thoả được, không thoả được) nếu hội của chúng G1∧Ù.....∧ÙGm là vững chắc (thoả được, không thoả được). Một mô hình của tập công thức G là mô hình của tập công thức G1∧Ù.....∧ÙGm.

3.1.3. Dạng chuẩn tắc

Đưa các công thức về dạng chuẩn tắc sẽ thuận lợi cho việc lập luận, suy diễn. Sử dụng các phép biến đổi tương đương, có thể đưa một công thức bất kỳ về các dạng chuẩn tắc.

- ✓ Dạng chuẩn là kết xuất chuẩn của các giải thuật làm việc với phép toán mệnh đề.
- ✓ Tuyển cơ bản: là thành phần cơ bản hay sự kết hợp của các thành phần cơ bản bằng phép tuyển (∨).
- ✓ Hội cơ bản: là thành phần cơ bản hay sự kết hợp của các thành phần cơ bản bằng phép hội (∧Ù).
- ✓ Dạng chuẩn hội – CNF: là thành phần tuyển cơ bản hay các tuyển cơ bản kết hợp bởi phép hội.
- ✓ Dạng chuẩn tuyển – DNF: là thành phần hội cơ bản hay các hội cơ bản kết hợp bởi phép tuyển.

3.1.3.1. Sự tương đương của các công thức

Hai công thức A và B được xem là tương đương nếu chúng có cùng một giá trị chân lý trong mọi minh họa. Để chỉ A tương đương với B ta viết A≡º B bằng phương pháp bảng chân lý, dễ dàng chứng minh được sự tương đương của các công thức sau đây:

- ✓ A⇒ịB ≡º ¬A ∨ B
- ✓ A<=> B ≡º (A⇒ịB) ∧Ù (B⇒ịA)
- ✓ ¬ (¬A) ≡º A

Các luật logic:

Luật logic
Cho các biến mệnh đề p, q, r và các hằng đúng T, hằng sai F
1. Luật giao hoán
$p \wedge q \equiv q \wedge p$ $\quad\quad$ $p \vee q \equiv q \vee p$
2. Luật kết hợp
$(p \wedge q) \wedge r \equiv q \wedge (p \wedge r)$ $\quad\quad$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Luật phân phối
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $\quad\quad$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Luật phủ định của phủ định
$\neg(\neg p) \equiv p$
5. Luật De Morgan
$\neg(p \vee q) \equiv \neg p \wedge \neg q$ $\quad\quad$ $\neg(p \wedge q) \equiv \neg p \vee \neg q$
6. Luật kéo theo
$p \to q \equiv \neg p \vee q$
7. Luật tương đương
$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
8. Luật trung hòa
$p \wedge T \equiv p$ $\quad\quad$ $p \vee F \equiv p$
9. Luật thống trị
$p \wedge F \equiv F$ $\quad\quad$ $p \vee T \equiv T$
10. Luật lũy đẳng
$p \wedge p \equiv p$ $\quad\quad$ $p \vee p \equiv p$
11. Luật phần tử bù
$p \wedge \neg p \equiv F$ $\quad\quad$ $p \vee \neg p \equiv T$

Việc chứng minh các luật trên dựa vào việc lập bảng giá trị chân lý

3.1.3.2. Dạng chuẩn tắc

Các công thức tương đương có thể xem như các biểu diễn khác nhau của cùng một sự kiện. Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta sẽ chuẩn hóa các công thức, đưa chúng về dạng biểu diễn chuẩn được gọi là dạng chuẩn hội. Một công thức ở dạng chuẩn hội, có dạng A1 $\vee$ ... . $\vee$ Am trong đó các Ai là literal. Chúng ta có thể biến đổi một công thức bất kỳ về công thức ở dạng chuẩn hội bằng cách áp dụng các bước sau:

- ✓ Bỏ các dấu kéo theo (=>) bằng cách thay (A=>B) bởi (¬A$\vee$B).
- ✓ Chuyển các dấu phủ định (¬) vào sát các kết hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay ¬ (¬A) bởi A .
- ✓ Áp dụng luật phân phối, thay các công thức có dạng A$\vee$ (B$\wedge$ÙC) bởi (A $\vee$ B) $\wedge$Ù (A $\vee$ B) .

Ví dụ 3.3:

Chuẩn hóa công thức (P => Q) ∨ ¬ (R ∨ ¬S)

(P => Q) ∨ ¬ (R ∨ ¬S) ≡º (¬P ∨ Q) v (¬R ∧Ù S) ≡º ((¬P v Q) ∨¬R) ∧Ù ((¬P ∨ Q) ∨ S) ≡º (¬ P ∨ Q ∨ ¬R) ∧Ù (¬P ∨ Q ∨ S). Như vậy công thức (P=> Q) ∨ ¬ (R ∨ ¬S) được đưa về dạng chuẩn hội (¬P ∨ Q ∨ ¬R) ∧Ù (¬P ∨ Q ∨ S).

Khi biểu diễn tri thức bởi các công thức trong logic mệnh đề, cơ sở tri thức là một tập nào đó các công thức. Bằng cách chuẩn hoá các công thức, cơ sở tri thức là một tập nào đó các câu tuyển.

3.1.4. Luật suy diễn

Là quá trình dùng trong hệ chuyên gia để rút ra thông tin mới từ các thông tin cũ.

Một công thức H được xem là hệ quả logic của một tập công thức G ={G1, ...,Gm} nếu trong bất kỳ minh họa nào mà {G1,.....,Gm} đúng thì H cũng đúng, hay nói cách khác bất kỳ một mô hình nào của G cũng là mô hình của H.

Khi có một cơ sở tri thức, ta muốn sử dụng các tri thức trong cơ sở này để suy ra tri thức mới mà nó là hệ quả logic của các công thức trong cơ sở tri thức. Điều đó được thực hiện bằng các thực hiện các luật suy diễn (rule of inference). Luật suy diễn giống như một thủ tục mà chúng ta sử dụng để sinh ra một công thức mới từ các công thức đã có. Một luật suy diễn gồm hai phần: một tập các điều kiện và một kết luận. Trong một số tài liệu, luật suy diễn được biểu diễn dưới dạng "phân số", trong đó tử số là danh sách các điều kiện, còn mẫu số là kết luận của luật, tức là mẫu số là công thức mới được suy ra từ các công thức ở tử số.

Luật Modus Ponens: Nếu mệnh đề P là đúng và P ⇒ Q là đúng thì giá trị của Q là đúng.

Luật Modus Tollens: Nếu mệnh đề P ⇒ Q là đúng và mệnh đề Q là sai thì giá trị của P sẽ sai

Bảng sau trình bày một số luật suy diễn quan trọng trong logic mệnh đề. Trong các luật này P, P_i , Q, S là các công thức:

Bảng 3.3: Một số luật suy diễn

Tổng quát hóa (Generalization) - (còn được gọi là luật cộng)

$$\frac{p}{\therefore p \vee q} \qquad \frac{q}{\therefore p \vee q}$$

Cơ sở toán học

$$p \rightarrow ¿)$$
$$q \rightarrow (p \vee q)$$

Chuyên biệt hóa (Specialization) – (còn được gọi là luật rút gọn)

$$\frac{p \wedge q}{\therefore p} \qquad \frac{p \wedge q}{\therefore q}$$

Cơ sở toán học

$$p \wedge q \rightarrow p$$

$$p \wedge q \rightarrow q$$

Tam đoạn luận

Suy luận chứa hai tiền đề (tiền đề thứ nhất gọi là tiền đề chính, tiền đề thứ hai gọi là tiền đề phụ) và một kết luận được gọi là tam đoạn luận.

Modus Ponens – Tam đoạn luận khẳng định

$$\begin{array}{l} p \rightarrow q \\ \underline{p} \\ \therefore q \end{array}$$

Cơ sở toán học:

$$((p \rightarrow q) \wedge p) \rightarrow q$$

Modus Tollens – Tam đoạn luận phủ định

$$\begin{array}{l} p \rightarrow q \\ \underline{\neg q} \\ \therefore \neg p \end{array}$$

Cơ sở toán học của phương pháp này:

$$((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$$

Tam đoạn luận rời (Quy tắc loại trừ)

$$\begin{array}{l} p \vee q \\ \underline{\neg p} \\ \therefore q \end{array}$$

$$\begin{array}{l} p \vee q \\ \underline{\neg q} \\ \therefore p \end{array}$$

Cơ sở toán học

$$((p \vee q) \wedge \neg q) \rightarrow p$$
$$((p \vee q) \wedge \neg p) \rightarrow q$$

Tam đoạn luận giả định

$$\begin{array}{l} p \rightarrow q \\ \underline{q \rightarrow r} \\ \therefore p \rightarrow r \end{array}$$ **Cơ sở toán học**

$$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

Chứng minh bằng phân chia trường hợp

$$\begin{array}{l} p \vee q \\ p \rightarrow r \\ \underline{q \rightarrow r} \\ \therefore r \end{array}$$

Cơ sở toán học

$$((p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)) \rightarrow r$$

Quy tắc mâu thuẫn

(Được dựa trên nguyên tắc: *"Nếu một giả định dẫn đến mâu thuẫn, thì giả định đó phải sai"*)

p là một mệnh đề cần kiểm tra giá trị chân lý. Nếu có thể chỉ ra rằng giả sử p là sai, dẫn đến mâu thuẫn, thì có thể kết luận p đúng.

$\neg p \rightarrow c$ với c là một mâu thuẫn.

$\therefore p$

Một luật suy diễn được xem là tin cậy (secured) nếu bất kỳ một mô hình nào của giả thiết của luật cũng là mô hình kết luận của luật. Ta chỉ quan tâm đến các luật suy diễn tin cậy.

Bằng phương pháp bảng chân lý, ta có thể kiểm chứng được các luật suy diễn nêu trên đều là tin cậy. Bảng chân lý của luật giải được cho trong Bảng 3.X. Từ bảng này ta thấy rằng , trong bất kỳ một minh họa nào mà cả hai giả thiết P ∨ Q , ¬Q ∨ S đúng thì kết luận P ∨ S cũng đúng. Do đó luật giải là luật suy diễn tin cậy.

Bảng 3.3: Bảng chân lý chứng minh tính tin cậy của luật giải

P	Q	S	P ∨ Q	¬Q ∨ S	P ∨ S
False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

Ta có nhận xét rằng, luật giải là một luật suy diễn tổng quát, nó bao gồm luật Modus Ponens, luật Modus Tollens, luật bắc cầu như các trường hợp riêng.

Ví dụ 3.4:

Ta có các biểu thức sau: A∨B, A∨C,và ¬A là TRUE

Chứng minh B∧C có trị TRUE

1	A∨B	P (tiên đề)
2	A∨C	P (tiên đề)
3	¬A	P (tiên đề)
4	B	1,3, tam đoạn luận tuyển
5	C	2,3, tam đoạn luận tuyển
6	B∧C	4,5, Luật hội

Đã chứng minh xong

Ví dụ 3.5:

Ta có các biểu thức sau là đúng:

A∨B, A⇒ C, B⇒ D, ¬D

Chứng minh C

Sử dụng phương pháp chứng minh bác bỏ (refutation proof hoặc proof by contradiction). Để chứng minh P đúng, sẽ giả sử P sai (thêm ¬P vào các giả thiết) và dẫn tới một mâu thuẫn.

Ta giả thiết ¬C dẫn đến false

1	A∨B	P (tiên đề)
2	A⇒C	P (tiên đề)
3	B⇒ D	P (tiên đề)
4	¬D	P (tiên đề)
5	¬C	P (giả thiết phản chứng)
6	¬B	3,4,Modus Tollens
7	A	1,6, Tam đoạn luận tuyển
8	¬A	2,5,Modus Tollens
9	A ^¬A	7,8, Luật hội
10	False	Mâu thuẫn với luật hội

Đã chứng minh xong

3.1.5. Luật phân giải, chứng minh bác bỏ bằng luật giải

Clause: là tuyển của không hay nhiều thành phần cơ bản.

Dạng clause: là hội của một hay nhiều Clause

Luật phân giải mệnh đề:

$P \vee D_1, \neg P \vee D_2 \therefore (D_1\text{-}P) \text{v} (D_2\text{-}\neg P)$

- ✓ D_1, D_2 là tuyển của không hay một thành phần cơ bản.
- ✓ P là mệnh đề
- ✓ D_1-P: là một clause thu được bằng cách xóa bỏ các P trong D_1
- ✓ D_2- $\neg P$: là một clause thu được bằng cách xóa bỏ các P trong D_2

Luật phân giải bảo toàn tính Unsatisfiable

S là unsatisfiable $\Leftrightarrow R_n(S)$cũng unsatisfiable

R: luật phân giải, n số lần áp dụng R trên S, $n>0$

Ứng dụng của luật phân giải: dùng để chứng minh: Có S là tập các clause, dùng S chứng minh biểu thức mệnh đề W

Phương pháp:

- ✓ Thành lập phủ định của W
- ✓ Đưa W về dạng clause
- ✓ Thêm clause trong bước 2 vào S thành lập S_1
- ✓ Dùng luật phân giải trên S_1 để dẫn ra clause rỗng.

Ví dụ 3.6:

Cho đoạn sau: "Nam đẹp trai, giàu có. Do vậy, Nam hoặc là phung phí hoặc là nhân từ và giúp người. Thực tế, Nam không phung phí hoặc cũng không kiêu căng."

Suy luận: "Có thể nói Nam là người nhân từ". Kiểm chứng kết quả suy luận trên, bằng luật phân giải.

Chuyển sang mệnh đề

- ✓ P1 = “Nam đẹp trai.”
- ✓ P2 = “Nam giàu có.”
- ✓ P3 = “Nam phung phí.”
- ✓ P4 = “Nam kiêu căng.”
- ✓ P5 = “Nam nhân từ.”
- ✓ P6 = “Nam giúp người.”

Các biểu thức thành lập được từ đoạn trên:

- ✓ Wff1 = P1 ^ P2
- ✓ Wff2 = (P1 ^ P2) => (P3 ^ ¬(P5 ^ P6)) v (¬P3 ^ (P5 ^ P6))
- ✓ Wff3 = ¬P3 ^ ¬P4

Wff4 = P5 Biểu thức cần chứng minh.

Đưa về dạng clause

- ✓ Wff1, sinh ra hai clause: C1 = P1 C2 = P2
- ✓ Wff2 = ¬(P1 ^ P2) v ((P3 ^ ¬(P5 ^ P6)) ∨ (¬P3 ^ (P5 ^ P6)))

= (¬P1 ∨ ¬P2 ∨ P3 v ¬P3 ∨ ¬P6) ^ (¬P1 ∨ ¬P2 ∨ P5 v ¬P3 ∨ ¬P6)^(¬P1 ∨ ¬P2 ∨ P3 ∨ P3 ∨ P5) ^ (¬P1 ∨ ¬P2 ∨ P5 ∨ P3 ∨ P5) ^ (¬P1 ∨ ¬P2 ∨ P3 ∨ P5 ∨ ¬P6)^ (¬P1 ∨ ¬P2 ∨ P5 ∨ P5 ∨ ¬P6) ^(¬P1 ∨ ¬P2 ∨ P3 ∨ P3 ∨ P6) ^ (¬P1 v ¬P2 ∨ P5 ∨ P3 ∨ P6)

Sinh ra các clause:

C3 = (¬P1 ∨ ¬P2 ∨ ¬P6) C4 = (¬P1 ∨ ¬P2 ∨ P5 ∨ ¬P3 ∨ ¬P6)

C5 = (¬P1 ∨ ¬P2 ∨ P3 v P5) C6 = (¬P1 ∨ ¬P2 ∨ P3 ∨ P5)

C7 = (¬P1 ∨ ¬P2 ∨ P3 v P5 v ¬P6) C8 = (¬P1 ∨ ¬P2 ∨ P5 ∨ ¬P6)

C9 = (¬P1 ∨ ¬P2 ∨ P3 v P6) C10 =(¬P1 ∨ ¬P2 ∨ P5 ∨ P3 ∨ P6)

Wff3 sinh ra các clause: C11 = ¬P3 C12 = ¬P4 C13 = ¬P5 (gồm cả bước lấy phủ định kết luận)

Áp dụng luật phân giải trên các clause

TT	***Clause***	***Luật áp dụng***
1	P1	P
2	P2	P
3	¬P1 ∨ ¬P2 ∨ ¬P6	P
4	¬P1 ∨ ¬P2 ∨ P5 ∨ ¬P3 ∨ ¬P6	P
5	¬P1 ∨ ¬P2 ∨ P3 ∨ P5	P
6	¬P1 ∨ ¬P2 ∨ P3 ∨ P5	P
7	¬P1 v ¬P2 ∨ P3 ∨ P5 ∨ ¬P6	P
8	¬P1 v ¬P2 ∨ P5 ∨ ¬P6	P
9	¬P1 v ¬P2 ∨ P3 ∨ P6	P
10	¬P1 v ¬P2 ∨ P5 ∨ P3 ∨ P6	P
11	¬P3	P (giả thiết phản chứng)
12	¬P4	P
13	¬P5	P
14	¬P2 ∨ ¬P6	1,3, R

15	¬P6	2, 14, R
16	¬P1 ∨ ¬P2 ∨ P5 ∨ P3	10,15,R
17	¬P2 ∨ P5 ∨ P3	1,16,R
18	P5 ∨ P3	2,17, R
19	P3	13, 18, R
20	Đã được CM	11, 19, R

Đã được chứng minh

Định lý giải:

Một tập câu tuyển là không thỏa được nếu và chỉ nếu câu rỗng [] ∈ R(G).

Định lý giải có nghĩa rằng, nếu từ các câu thuộc G, bằng cách áp dụng luật giải ta dẫn tới câu rỗng thì G là không thỏa được, còn nếu không thể sinh ra câu rỗng bằng luật giải thì G thỏa được. Lưu ý rằng, việc dẫn tới câu rỗng có nghĩa là ta đã dẫn tới hai literal đối lập nhau P và ¬ P (tức là dẫn tới mâu thuẫn).

Nếu G là tập hữu hạn các câu thì các literal có mặt trong các câu của G là hữu hạn. Do đó số các câu tuyển thành lập được từ các literal đó là hữu hạn. Vì vậy, chỉ có một số hữu hạn câu được sinh ra bằng luật giải. Thủ tục giải sẽ dừng lại sau một số hữu hạn bước.

Chỉ sử dụng luật giải ta không thể suy ra mọi công thức là hệ quả logic của một tập công thức đã cho. Tuy nhiên, sử dụng luật giải ta có thể chứng minh được một công thức bất kì có là hệ quả của một tập công thức đã cho hay không bằng phương pháp chứng minh bác bỏ. Vì vậy luật giải được xem là luật đầy đủ cho bác bỏ.

3.2. LOGIC VỊ TỪ CẤP MỘT

Logic mệnh đề cho phép ta biểu diễn các sự kiện, mỗi kí hiệu trong logic mệnh đề được minh họa như là một sự kiện trong thế giới hiện thực, sử dụng các kết nối logic ta có thể tạo ra các câu phức hợp biểu diễn các sự kiện mang ý nghĩa phức tạp hơn. Như vậy khả năng biểu diễn của logic mệnh đề chỉ giới hạn trong phạm vi thế giới các sự kiện.

Logic mệnh đề cung cấp công cụ thu nhận các sự kiện và các luật theo các dạng kí hiệu và dùng các phép toán logic để thao tác trên các sự kiện và luật. Tiếp cận logic hình thức này đảm bảo một phương pháp chính xác để quản lý các câu có giá trị đúng hay sai. Tuy nhiên, biểu diễn tri thức bằng mệnh đề gặp phải trở ngại lớn là không thể can thiệp vào cấu trúc của mệnh đề. Nói một cách khác, mệnh đề không có cấu trúc. Điều này làm hạn chế rất nhiều các thao tác suy luận. Do đó, người ta đã đưa vào khái niệm vị từ và lượng từ để tăng cường tính cấu trúc của một mệnh đề.

Logic vị từ là sự mở rộng của phép toán mệnh đề để thể hiện rõ hơn các tri thức.

Logic vị từ cấp một là mở rộng của logic mệnh đề. Logic vị từ cấp một cho phép mô tả thế giới với các đối tượng, các thuộc tính của đối tượng và các mối quan hệ giữa các đối tượng. Nó sử dụng các biến (biến đối tượng) để chỉ một đối tượng trong một miền đối tượng nào đó. Để mô tả các thuộc tính của đối tượng, các quan hệ giữa các đối tượng, trong logic vị từ, người ta dựa vào các vị từ (predicate). Ngoài các kết nối logic như trong logic mệnh đề, logic vị từ cấp một còn sử dụng các lượng tử. Ví dụ, lượng tử ∀" (với mọi) cho phép ta tạo ra các câu nói tới mọi đối tượng trong một miền đối tượng nào đó.

Logic vị từ cấp một có vai trò quan trọng trong biểu diễn tri thức, vì khả năng biểu diễn của nó (nó cho phép ta biểu diễn tri thức về thế giới với các đối tượng, các thuộc tính

của đối tượng và các quan hệ của đối tượng), và hơn nữa, nó là cơ sở cho nhiều ngôn ngữ logic khác.

3.2.1. Cú pháp và ngữ nghĩa của logic vị từ cấp một

3.2.1.1. Cú pháp

Thể hiện toàn bộ câu bằng một ký hiệu đơn. Trong logic vị từ, một mệnh đề được cấu tạo bởi hai thành phần là các đối tượng tri thức và mối liên hệ giữa chúng (gọi là vị từ). Các mệnh đề sẽ được biểu diễn dưới dạng:

Vị từ (<đối tượng 1>,<đối tượng 2>,…<đối tượng n>)

Ví dụ 3.7

Để biểu diễn vị của các trái cây như "Cam có vị ngọt", mệnh đề sẽ được viết lại thành: **Vị (Cam, Ngọt)**

A = "Chiếc áo màu xanh".

Phép toán vị từ cho phép mô tả theo quan hệ của tri thức theo dạng: ***màu (chiếc áo, xanh).***

Cách thể hiện này thuận tiện đối với việc dùng biến và hàm trong xử lý tri thức. Vì kiểu biểu diễn này tương tự như hàm trong các ngôn ngữ lập trình, các đối tượng tri thức chính là các tham số của hàm, giá trị mệnh đề chính là kết quả của hàm.

Phép toán vị từ

Phép toán vị từ dùng các kí hiệu để thể hiện tri thức. Những kí hiệu này gồm hằng số, vị từ, biến và hàm. Phép toán vị từ cho phép thực hiện các phép toán logic mệnh đề trên các kí hiệu.

Các hằng số: Dùng để đặt tên các đối tượng đặc biệt hay thuộc tính. Các hằng số được kí hiệu bằng chữ viết thường, ví dụ: Nam. Hằng số "Nam" có thể được dùng để thể hiện đối tượng Nam, là một người được xét.

Các vị từ: Một sự kiện hay mệnh đề trong phép toán vị từ được chia làm hai phần: vị từ và tham số. Tham số thể hiện một hay nhiều đối tượng của mệnh đề, còn vị từ dùng để khẳng định về đối tượng. Ví dụ: mệnh đề "Nam thích Mai" được viết dưới dạng vị từ có dạng:

Thích(Nam, Mai)

Biến: Các biến dùng để thể hiện các lớp tổng quát của các đối tượng hay các thuộc tính. Như vậy, có thể dùng vị từ có biến để thể hiện nhiều vị từ tương tự.

Ví dụ 3.8:

Có hai mệnh đề: "Nam thích Mai" và "Dũng thích Lan". Hai biến X, Y được dùng trong mệnh đề:

Thích(X,Y)

Trong phép toán vị từ, người ta dùng biến như đối số của biểu thức vị từ hay của hàm.

Hàm: được thể hiện bằng kí hiệu, cho biết quan hệ hàm số.

Ví dụ 3.9:

Có các mệnh đề sau "Nam là bố của Dũng", "Lan là mẹ của Sơn", "Dũng và Sơn là bạn của nhau". Hàm số được viết để thể hiện các quan hệ này như sau:

Bố(Dũng)=Nam

Mẹ(Sơn)=Lan

Bạn(Dũng, Sơn)

Bạn (Bố(Dũng), Mẹ(Sơn))

Các phép toán:

¬	Phủ định	- một ngôi.
∀X	Với mọi	- một ngôi
∃X	Tồn tại	- một ngôi
^	Hội	- hai ngôi.
∨	Tuyển	- hai ngôi.
⇒	Suy ra	- hai ngôi.
⇔	Tương đương	- hai ngôi.

Biểu thức vị từ đúng được ký hiệu wff.

Biểu thức cơ bản: Có thể là một vị từ, một đại diện trị TRUE (trị là T - đúng), một đại diện trị FALSE (trị là F - sai).

Một biểu thức đúng cú pháp được định nghĩa như sau:

wff = "Biểu thức cơ bản" |¬ wff |wff ^ wff |wff ∨ wff |wff=>wff |wff

= wff |(wff) |∀X wff |∃X wff

Với

- ✓ X: Là một biến.
- ✓ ∀: Lượng từ với mọi.
- ✓ ∃: Lượng từ tồn tại.

Giả sử có: Nam là học sinh khá. Lan là học sinh trung bình. Mai học sinh khá

Xét tập D = [Nam, Lan, Mai]

Gọi p(X) cho biết: "X là học sinh khá" ta có các vị từ

p("Nam"): trị là T. p("Lan"): trị là F. p("Mai"): trị là T.

Lượng từ tồn tại:

Xét mệnh đề p("Nam") ∨ p("Lan") ∨ p("Mai") có thể biểu diễn bằng vị từ:

∃X ∈ D: p(X)

"Tồn tại X thuộc tập D, mà X là học sinh khá"

Lượng từ với mọi:

Xét mệnh đề p("Nam") ^ p("Lan") ^ p("Mai") có thể biểu diễn bằng vị từ:

∀X ∈ D: p(X)

"Mọi X thuộc tập D đều là học sinh khá"

Ví dụ 3.10:

Chuyển các câu sau sang biểu thức vị từ:

"Mọi sinh viên trường ĐH đều có bằng tú tài.
Lan không có bằng tú tài.
Do vậy, Lan không là sinh viên trường ĐH"

Với sv_bk(X) cho biết: "X là sinh viên trường ĐH"
tu_tai(X) cho biết: "X có bằng tú tài"

Các câu trên được chuyển qua vị từ là:

∀X(sv_bk(X) ⇒ tu_tai(X)).
¬tu_tai("Lan").
Do vậy, ¬sv_bk("Lan").

Ví dụ 3.11:

"Chỉ vài sinh viên máy tính lập trình tốt."

với sv_mt(X) : "X là sinh viên máy tính"

laptrinh_tot(X) : "X lập trình tốt"

Câu trên chuyển sang vị từ là: ∃X(sv_mt(X) ^ laptrinh_tot(X))

"Không một sinh viên máy tính nào không cần cù."

với: sv_mt(X) : "X là sinh viên máy tính

can_cu(X) : "X cần cù"

Câu trên chuyển sang là: ∀X (sv_mt(X) => can_cu(X))

"Không phải tất cả các sinh viên máy tính đều thông minh"

với thong_minh(X): "X thông minh"

Câu trên chuyển sang là: ∃X(sv_mt(X) ^ ¬thong_minh(X))

- **Vấn đề:** Nếu chúng ta có biểu thức sau: ∀X∃Y p(X,Y). Để hiểu được biểu thức cần sự diễn dịch.

+ Cách hiểu 1:

X, Y : là con người.

p(X,Y) cho biết : "X là cha của Y"

Do vậy:

∀X∃Y p(X,Y) có thể hiểu là:

"Mọi người X, tồn tại người Y để X là cha của Y"

-> wff = ∀X∃Y p(X,Y) có trị là F (sai)

+ Cách hiểu 2:

X, Y : là con người.

p(X,Y) cho biết : "Y là cha của X"

Do vậy:

∀X∃Y p(X,Y) có thể hiểu là:

"Mọi người X, tồn tại người Y là cha của X"

-> wff = ∀X∃Y p(X,Y) có trị là T (đúng)

+ Cách hiểu 3:

X, Y : là số nguyên.

p(X,Y) cho biết : "Y bằng bình phương của X"

-> wff = ∀X∃Y p(X,Y) có trị là T (đúng)

- **Diễn dịch:** gồm

- Tập D, không rỗng, miền diễn dịch.
- Các phép gán:
  - Vị từ : Quan hệ trên D
  - Hàm : Hàm (ánh xạ) trên D
  - Biến tự do : Một trị trên D, cùng một trị cho các xuất hiện
  - Hằng : Một trị trên D, cùng một trị cho các xuất hiện

- **Ngữ nghĩa:**

Có diễn dịch I trên miền D của wff.

- Wff không có lượng từ:
  Ngữ nghĩa = trị sự thật (T|F) của wff khi áp dụng diễn dịch
- wff có lượng từ:

∃XW là T, nếu: W(X/d) là T cho một d thuộc D

ngược lại: ∃XW là F

∀XW là T, nếu: W(X/d) là T cho mọi d thuộc D

ngược lại: ∀XW là F

Có I: diễn dịch, E là wff

- **Model:**

I là cho E có trị T ---> I là Model của E

Ngược lại: ---> I là **CounterModel** của E

- **Valid:**

  E là valid nếu mọi diễn dịch I đều là Model.

  Ngược lại là : **Invalid**

- **Unsatisfiable:**

  E là unsatisfiable : mọi I đều là CounterModel

  Ngược lại :**Satisfiable**

- Từ tương đương của mệnh đề:

Nếu chúng ta thay thế các mệnh đề bởi các biểu thức vị từ, các mệnh đề cùng tên thì được thay cùng một biểu thức vị từ, thì được một tương đương của vị từ.

Ví dụ 3.12:

Mệnh đề:

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

Vị từ:

P bởi: $\forall X \exists Y p(X,Y)$, Q bởi: q(X)

tương đương:

$(\forall X \exists Y p(X,Y) \Rightarrow q(X)) = (\neg(\forall X \exists Y p(X,Y)) \vee q(X))$

- Lượng từ:

$\neg(\forall X\ W) = \exists X(\neg W)$

$\neg(\exists X\ W) = \forall X(\neg W)$

Với W là một wff

- Tương đương có ràng buộc:

  Sau đây: Y: biến, W(X): wff *có chứa* biến X, C là wff không chứa X

  Ràng buộc: Y không xuất hiện trong W(X)

  Tương đương:

  - **$\forall X\ W(X) = \forall Y\ W(Y)$**
  - **$\exists X\ W(X) = \exists\ Y\ W(Y)$**

Tương đương:

Dạng tuyển:

- **$C \vee \forall X A(X) = \forall X(C \vee A(X))$**
- **$C \vee \exists X A(X) = \exists X(C \vee A(X))$**

Dạng hội:

- **$C \wedge \forall X A(X) = \forall X(C \wedge A(X))$**
- **$C \wedge \exists X A(X) = \exists X(C \wedge A(X))$**

Dạng suy ra:

- **$C \Rightarrow \forall X A(X) = \forall X(C \Rightarrow A(X))$**
- **$C \Rightarrow \exists X A(X) = \exists X(C \Rightarrow A(X))$**
- **$\forall X A(X) \Rightarrow C = \forall X(A(X) \Rightarrow C)$**
- **$\exists X A(X) \Rightarrow C = \exists X(A(X) \Rightarrow C)$**

- Dạng Chuẩn Prenex:

  $Q_1X_1Q_2X_2 \ldots QnXnM$

  Qi : $\forall$, $\exists$.

  M : wff không có lượng từ.

Ví dụ 3.13:

- sv_bk(x)
- $\forall X(sv\_bk(X) \wedge hoc\_te(X))$

- ∀X∃Ycha(X,Y)

- Giải thuật đưa wff về chuẩn Prenex:
  - Đổi tên biến --> wff không còn lượng từ cùng tên biến, biến lượng từ không trùng tên biến tự do.
  - Đưa lượng từ sang trái dùng tương đương.
- Dạng chuẩn Tuyển Prenex:

  $Q1X_1Q_2X_2\ldots QnXn(C_1 \vee \ldots \vee Ck)$

  Ci : Thành phần hội cơ bản.
- Dạng chuẩn Hội Prenex:

  $Q1X_1Q_2X_2\ldots QnXn(D_1 \vee \ldots \vee Dk)$

  Di : Thành phần tuyển cơ bản.
- Giải thuật:
  - Đổi tên biến.
  - Loại bỏ => bởi : A => B = ¬A ∨ B
  - Chuyển ¬ sang phải dùng De Morgan và phủ định kép.
  - Chuyển lượng từ sang trái dùng tương đương.
  - Phân phối v trên ^ (CNF), hay ^ trên ∨ (DNF)

Các ký hiệu.

Logic vị từ cấp một sử dụng các loại ký hiệu sau đây.

- ✓ Các ký hiệu hằng: a, b, c, An, Ba, John,...
- ✓ Các ký hiệu biến: x, y, z, u, v, w,...
- ✓ Các ký hiệu vị từ: P, Q, R, S, Like, Havecolor, Prime,...

Mỗi vị từ là vị từ của n biến (n≥³0). Chẳng hạn Like là vị từ của hai biến, Prime là vị từ một biến. Các ký hiệu vị từ không biến là các ký hiệu mệnh đề.

- ✓ Các ký hiệu hàm: f, g, cos, sin, mother, husband, distance,...
- ✓ Mỗi hàm là hàm của n biến (n≥³1). Chẳng hạn, cos, sin là hàm một biến, distance là hàm của ba biến.
- ✓ Các ký hiệu kết nối logic: ∧Ù (hội), ∨Ú (tuyển), ⎤(phủ định), ⇒ị(kéo theo), ⇔Û (kéo theo nhau).
- ✓ Các ký hiệu lượng tử: ∀" (với mọi), ∃$ (tồn tại).
- ✓ Các ký hiệu ngăn cách: dấu phẩy, dấu mở ngoặc và dấu đóng ngoặc.

Các hạng thức

Các hạng thức (term) là các biểu thức mô tả các đối tượng. Các hạng thức được xác định đệ quy như sau.

- ✓ Các ký hiệu hằng và các ký hiệu biến là hạng thức.
- ✓ Nếu t1, t2, t3, ..., tn là n hạng thức và f là một ký hiệu hàm n biến thì f(t1, t2, ..., tn) là hạng thức. Một hạng thức không chứa biến được gọi là một hạng thức cụ thể (ground term).

Chẳng hạn, An là ký hiệu hằng, mother là ký hiệu hàm một biến, thì mother (An) là một hạng thức cụ thể.

Các công thức phân tử

Chúng ta sẽ biểu diễn các tính chất của đối tượng, hoặc các quan hệ của đối tượng bởi các *công thức phân tử* (*câu đơn*).

Các công thức phân tử (câu đơn) được xác định đệ quy như sau.

- ✓ Các ký hiệu vị từ không biến (các ký hiệu mệnh đề) là câu đơn.

✓ Nếu t1, t2,...,tn là n hạng thức và p là vị từ của n biến thì p(t1,t2,...,tn) là câu đơn.

Chẳng hạn, Hoa là một ký hiệu hằng, Love là một vị từ của hai biến, husband là hàm của một biến, thì Love (Hoa, husband(Hoa)) là một câu đơn.

Các công thức

Từ công thức phần tử, sử dụng các kết nối logic và các lượng tử, ta xây dựng nên các công thức (các câu).

Các công thức được xác định đệ quy như sau:

✓ Các công thức phân tử là công thức.

✓ Nếu G và H là các công thức, thì các biểu thức (G ∧Ù H), (G ∨Ú H), (¬ G), (G⇒ịH), (G⇔ÛH) là công thức.

✓ Nếu G là một công thức và x là biến thì các biểu thức (∀" x G), (∃$ x G) là công thức.

Các công thức không phải là công thức phân tử sẽ được gọi là các câu phức hợp. Các công thức không chứa biến sẽ được gọi là công thức cụ thể. Khi viết các công thức ta sẽ bỏ đi các dấu ngoặc không cần thiết, chẳng hạn các dấu ngoặc ngoài cùng.

Lượng tử phổ dụng (∀") cho phép mô tả tính chất của cả một lớp các đối tượng, chứ không phải của một đối tượng, mà không cần phải liệt kê ra tất cả các đối tượng trong lớp. Chẳng hạn sử dụng vị từ Elephant(x) (đối tượng x là con voi) và vị từ Color(x, Gray) (đối tượng x có mầu xám) thì câu " tất cả các con voi đều có mầu xám" có thể biểu diễn bởi công thức ∀"x (Elephant(x) ⇒ị Color(x, Gray)).

Lượng tử tồn tại (∃$) cho phép ta tạo ra các câu nói đến một đối tượng nào đó trong một lớp đối tượng mà nó có một tính chất hoặc thoả mãn một quan hệ nào đó. Chẳng hạn bằng cách sử dụng các câu đơn Student(x) (x là sinh viên) và Inside(x, P301), (x ở trong phòng 301), ta có thể biểu diễn câu " Có một sinh viên ở phòng 301" bởi biểu thức ∃$x (Student(x) ∧Ù Inside(x,P301).

Một công thức là công thức phân tử hoặc phủ định của công thức phân tử được gọi là literal. Chẳng hạn, Play(x, Football), ⎤ Like(Lan, Rose) là các literal. Một công thức là tuyển của các literal sẽ được gọi là câu tuyển. Chẳng hạn, Male(x) ∨Ú ¬ Like(x, Foodball) là câu tuyển.

Trong công thức (∀"x G), hoặc ∃$x G trong đó G là một công thức nào đó, thì mỗi xuất hiện của biến x trong công thức G được gọi là xuất hiện buộc. Một công thức mà tất cả các biến đều là xuất hiện buộc thì được gọi là công thức đóng.

Ví dụ 3.14:

Công thức ∀"xP(x, f(a, x)) ∧ ∃y Q(y) là công thức đóng, còn công thức ∀"x P(x, f(y, x)) không phải là công thức đóng, vì sự xuất hiện của biến y trong công thức này không chịu ràng buộc bởi một lượng tử nào cả (Sự xuất hiện của y gọi là sự xuất hiện tự do).

Sau này chúng ta chỉ quan tâm tới các công thức đóng.

Ngữ nghĩa.

Cũng như trong logic mệnh đề, nói đến ngữ nghĩa là chúng ta nói đến ý nghĩa của các công thức trong một thế giới hiện thực nào đó mà chúng ta sẽ gọi là một minh họa.

Để xác định một minh hoạ, trước hết ta cần xác định một miền đối tượng (nó bao gồm tất cả các đối tượng trong thế giới hiện thực mà ta quan tâm).

Trong một minh hoạ, các ký hiệu hằng sẽ được gắn với các đối tượng cụ thể trong miền đối tượng các ký hiệu hàm sẽ được gắn với một hàm cụ thể nào đó. Khi đó, mỗi hạng thức cụ thể sẽ chỉ định một đối tượng cụ thể trong miền đối tượng. Chẳng hạn, nếu An là một ký hiệu hằng, Father là một ký hiệu hàm, nếu trong minh hoạ An ứng với một người cụ thể nào đó, còn Father(x) gắn với hàm; ứng với mỗi x là cha của nó, thì hạng thức Father(An) sẽ chỉ người cha của An.

Ngữ nghĩa của các câu đơn

Trong một minh hoạ, các ký hiệu vị từ sẽ được gắn với một thuộc tính, hoặc một quan hệ cụ thể nào đó. Khi đó mỗi công thức phân tử (không chứa biến) sẽ chỉ định một sự kiện cụ thể. Đương nhiên sự kiện này có thể là đúng (True) hoặc sai (False). Chẳng hạn, nếu trong minh hoạ, ký hiệu hằng Lan ứng với một cô gái cụ thể nào đó, còn Student(x) ứng với thuộc tính "x là sinh viên" thì câu Student (Lan) có giá trị chân lý là True hoặc False tuỳ thuộc trong thực tế Lan có phải là sinh viên hay không.

Ngữ nghĩa của các câu phức hợp.

Khi đã xác định được ngữ nghĩa của các câu đơn, ta có thể thực hiện được ngữ nghĩa của các câu phức hợp (được tạo thành từ các câu đơn bằng cách liên kết các câu đơn bởi các kết nối logic) như trong logic mệnh đề.

Ví dụ: Câu Student(Lan) ∧ Student(An) nhận giá trị True nếu cả hai câu Student(Lan) và Student(An) đều có giá trị True, tức là cả Lan và An đều là sinh viên.

Câu Like(Lan, Rose) ∨Ú Like(An, Tulip) là đúng nếu câu Like(Lan, Rose) là đúng hoặc câu Like(An, Tulip) là đúng.

Ngữ nghĩa của các câu chứa các lượng tử.

Ngữ nghĩa của các câu ∀"x G, trong đó G là một công thức nào đó, được xác định như là ngữ nghĩa của công thức là hội của tất cả các công thức nhận được từ công thức G bằng cách thay x bởi một đối tượng trong miền đối tượng. Chẳng hạn, nếu miền đối tượng gồm ba người {Lan, An, Hoa} thì ngữ nghĩa của câu ∀"x Student(x) được xác định là ngữ nghĩa của câu Student(Lan) ∧Ù Student(An) ∧Ù Student(Hoa). Câu này đúng khi và chỉ khi cả ba câu thành phần đều đúng, tức là cả Lan, An, Hoa đều là sinh viên.

Như vậy, công thức ∀"x G là đúng nếu và chỉ nếu mọi công thức nhận được từ G bằng cách thay x bởi một đối tượng trong miền đối tượng đều đúng, tức là G đúng cho tất cả các đối tượng x trong miền đối tượng.

Ngữ nghĩa của công thức ∃$x G được xác định như là ngữ nghĩa của công thức là tuyển của tất cả các công thức nhận được từ G bằng cách thay x bởi một đối tượng trong miền đối tượng. Chẳng hạn, nếu ngữ nghĩa của câu Younger(x,20) là " x trẻ hơn 30 tuổi " và miền đối tượng gồm ba người {Lan, An, Hoa} thì ngữ nghĩa của câu ∃$x Yourger(x,20) là ngữ nghĩa của câu Yourger(Lan,20) ∨Ú Yourger(An,20) ∨Ú Yourger(Hoa,20). Câu này nhận giá trị True nếu và chỉ nếu ít nhất một trong ba người Lan, An, Hoa trẻ hơn 20.

Như vậy công thức ∃$x G là đúng nếu và chỉ nếu một trong các công thức nhận được từ G bằng cách thay x bằng một đối tượng trong miền đối tượng là đúng.

Bằng các phương pháp đã trình bày ở trên, ta có thể xác định được giá trị chân lý (True, False) của một công thức bất kỳ trong một minh hoạ. (Lưu ý rằng, ta chỉ quan tâm tới các công thức đúng).

Sau khi đã xác định khái niệm minh hoạ và giá trị chân lý của một công thức trong một minh hoạ, có thể đưa ra các khái niệm công thức vững chắc (thoả được, không thoả được), mô hình của công thức giống như trong logic mệnh đề.

Các công thức tương đương

Cũng như trong logic mệnh đề, ta nói hai công thức G và H tương đương (viết là $G \equiv^o H$) nếu chúng cùng đúng hoặc cùng sai trong một minh hoạ. Ngoài các tương đương đã biết trong logic mệnh đề, trong logic vị từ cấp một còn có các tương đương khác liên quan tới các lượng tử. Giả sử G là một công thức, cách viết G(x) nói rằng công thức G có chứa các xuất hiện của biến x. Khi đó công thức G(y) là công thức nhận được từ G(x) bằng cách thay tất cả các xuất hiện của x bởi y. Ta nói G(y) là công thức nhận được từ G(x) bằngcách đặt tên lại (biến x được đổi tên lại là y).

Chúng ta có các tương đương sau đây:

1. $\forall$"x G(x) $\equiv^o$ $\forall$"y G(y)

   $\exists$\$x G(x) $\equiv^o$ $\exists$\$y G(y)

Đặt tên lại biến đi sau lượng tử phổ dụng (tồn tại), ta nhận được công thức tương đương.

2. $\neg$ ($\forall$"x G(x)) $\equiv^o$ $\exists$\$x ($\neg$ G(x))

   $\neg$ ($\exists$\$x G(x)) $\equiv^o$ $\forall$"x ($\neg$ G(x))

3. $\forall$"x (G(x) $\wedge$Ù H(x)) $\equiv^o$ $\forall$"x G(x) $\wedge$Ù $\forall$"x H(x)

   $\exists$\$x (G(x) $\vee$Ú H(x)) $\equiv^o$ $\exists$\$x G(x) $\vee$Ú $\exists$\$x H(x)

ví dụ : $\forall$"x Love(x, Husband(x)) $\equiv^o$ $\forall$"y Love(y, Husband(y)).

3.2.2. Quy tắc chuẩn hóa các công thức

Từ các câu phân tử, bằng cách sử dụng các kết nối logic và các lượng từ, ta có thể tạo ra các câu phức hợp có cấu trúc rất phức tạp. Để dễ dàng cho việc lưu trữ các câu trong bộ nhớ và thuận lợi cho việc xây dựng các thủ tục suy diễn, chúng ta cần chuẩn hoá các câu bằng cách đưa chúng về dạng chuẩn tắc hội (hội của các câu tuyển).

Trong mục này, chúng ta sẽ trình bày thủ tục chuyển một câu phức hợp thành một câu ở dạng chuẩn tắc hội tương đương.

Thủ tục chuẩn hoá các công thức bao gồm các bước sau:

a. Loại bỏ các kéo theo

Để loại bỏ các kéo theo, ta chỉ cần thay công thức $P \Rightarrow Q$ bởi công thức tương đương $\neg P \vee Q$, thay $P \Leftrightarrow Q$ bởi $(\neg P \vee Q) \wedge (P \vee \neg Q)$

b. Chuyển các phủ định tới các phân tử

Điều này được thực hiện bằng cách thay công thức ở vế trái bởi công thức ở vế phải trong các tương đương sau đây:

$\neg (\neg P) \equiv P$

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

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

$\neg (\forall x (P)) \equiv \exists x (\neg P)$

$\neg (\exists x (P)) \equiv \forall x (\neg P)$

c. Loại bỏ các lượng từ tồn tại

Giả sử p(X,Y) là các vị từ có nghĩa rằng "Y lớn hơn X" trong miền các số. Khi đó công thức $\forall x (\exists y (P(x,y)))$ có nghĩa là "với mọi số X, tồn tại Y sao cho số Y lớn hơn X". Ta có thể xem Y trong công thức đó là hàm của đối số X, chẳng hạn f(X) và loại bỏ lượng

tử ∃y, công thức đang xét trở thành ∀x (P(x,f(x))). Ví dụ: f(X)=X+1, khi đó ∀x (P(x,f(x))) ≡ ∀x (P(x, x+1)) nghĩa là với mọi giá trị của X thì X+1 lớn hơn X.

Một cách tổng quát, giả sử ∃y(G) là một công thức con của công thức đang xét và nằm trong miền tác dụng của các lượng từ ∀x1,...,∀xn. Khi đó ta có thể xem Y là hàm của n biến x1,...,xn, chẳng hạn f(x1,...,xn). Sau đó ta thay các xuất hiện của Y trong công thức G bởi hạng thức f(x1,...,xn) và loại bỏ các lượng từ tồn tại. Các hàm f được đưa vào để loại bỏ các lượng từ tồn tại được gọi là hàm Scholem.

Ví dụ 3.15: Xét công thức sau

$$\forall x\,(\exists y\,(P(x,y)) \vee \forall u\,(\exists v\,(Q(a,v) \wedge \exists y\,(\neg\, R(x,y)))) \qquad (1)$$

Công thức con ∃y (P(x,y) nằm trong miền tác dụng của lượng từ ∀x, ta xem Y là hàm của X: f(X). Các công thức con ∃v (Q(a,V)) và ∃y (¬ R(x,y)) nằm trong miền tác dụng của các lượng tử ∀x, ∀u ta xem V là hàm g(X,U) và Y là hàm h(X,U) của hai biến X, U. Thay các xuất hiện của Y và V bởi các hàm tương ứng, sau đó loại bỏ các lượng từ tồn tại, từ công thức (1) ta nhận được công thức:

$$\forall x\,(P(x,f(x)) \vee \forall u\,(Q(a,g(x,u)) \wedge \neg\, R(x,h(x,u)))) \qquad (2)$$

d. Loại bỏ các lượng từ phổ dụng

Sau bước 3 trong công thức chỉ còn lại các lượng từ phổ dụng và mọi xuất hiện của biến đều nằm trong miền tác dụng của các lượng từ phổ dụng. Ta có thể loại bỏ tất cả lượng từ phổ dụng, công thức (2) trở thành công thức:

$$P(x,f(x)) \vee Q(a,g(x,u)) \wedge \neg\, R(x,h(x,u))) \qquad (3)$$

Cần chú ý rằng, sau khi được thực hiện bước này tất cả các biến trong công thức được xem là chịu tác dụng của các lượng tử phổ dụng.

e. Chuyển các tuyển tới các literal

Bước này được thực hiện bằng cách thay các công thức dạng: P∨(Q∧R) bởi (P∨Q)∧(P∨R) và thay (P∧Q)∨R bởi (P∨Q)∧(P∨R). Sau bước này công thức trở thành hội của các câu tuyển nghĩa là ta nhận được các công thức ở dạng chuẩn tắc hội. Chẳng hạn, công thức (3) được chuyển thành công thức sau:

$$(P(x,f(x)) \vee Q(a,g(x,u))) \wedge (P(x,f(x)) \vee \neg\, R(x,h(x,u))) \qquad (4)$$

f. Loại bỏ các hội

Một câu hội là đúng nếu và chỉ nếu tất cả các thành phần của nó đều là đúng. Do đó công thức ở dạng chuẩn tắc hội tương đương với tập các thành phần. Chẳng hạn, công thức (4) tương đương với tập hai câu tuyển sau:

$$P(f(x)) \vee Q(a,g(x,u))$$
$$P(f(x)) \vee \neg\, R(x,h(x,u)) \qquad (5)$$

g. Đặt tên lại các biến

Đặt tên lại các biến sao cho các biến trong các câu khác nhau có tên khác nhau, chẳng hạn hai câu (5) có hai biến cùng tên là X, ta cần đổi tên biến X trong câu hai thành Z, khi đó các câu (5) tương đương với các câu sau:

$$P(f(x)) \vee Q(a,g(x,u))$$
$$P(f(Z)) \vee \neg\, R(Z,h(Z,u)) \qquad (5')$$

Như vậy, khi tri thức là một tập hợp nào đó các công thức trong logic vị từ, bằng cách áp dụng thủ tục trên ta nhận được cơ sở tri thức chỉ gồm các câu tuyển (tức là ta luôn luôn có thể xem mỗi câu trong cơ sở tri thức là tuyển của các literal). Hoàn toàn tương tự

như trong logic mệnh đề, mỗi câu tuyển có thể biểu diễn dưới dạng một kéo theo, vế trái của kéo theo là hội của các câu phân tử, còn vế phải là tuyển của các câu phân tử. Dạng câu này được gọi là câu Kowlski, một trường hợp quan trọng của câu Kowlski là câu Horn (luật if- then)

3.2.3. Các luật suy diễn

Tất cả các luật suy diễn đã được đưa ra trong logic mệnh đề đều đúng trong logic vị từ cấp một. Bây giờ ta đưa ra một luật suy diễn quan trọng trong logic vị từ liên quan tới lượng tử phổ dụng

a. Luật thay thế phổ dụng

Giả sử G là một câu, câu ∀x(G) là đúng trong một minh hoạ nào đó nếu và chỉ nếu G đúng với tất cả các đối tượng nằm trong miền đối tượng của minh hoạ đó. Mỗi hạng thức t ứng với một đối tượng, vì thế nếu câu ∀x (G) đúng thì khi thay tất cả các xuất hiện của biến X bởi hạng thức t ta nhận được câu đúng. Công thức nhận được từ công thức G bằng cách thay tất cả các xuất hiện của x bởi t được ký hiệu là G[x/t]. Luật thay thế phổ dụng (universal instatiation) phát biểu rằng, từ công thức ∀x (G) suy ra công thức G[x/t]:

Chẳng hạn, từ câu ∀x, like(x, "Football") (mọi người đều thích bóng đá), bằng cách thay X bởi An ta suy ra câu like("An", "Football") (An thích bóng đá).

b. Hợp nhất

Trong luật thay thế phổ dụng, ta cần sử dụng phép thế các biến bởi các hạng thức để nhận được các công thức mới từ công thức chứa các lượng tử phổ dụng. Ta có thể sử dụng phép thế để hợp nhất các câu phân tử (tức là để các câu trả lời thành đồng nhất). Chẳng hạn xét hai câu phân tử like("An",Y) và ∀x, like(X, "Football") mà để cho đơn giản ta bỏ đi các lượng tử phổ dụng. Sử dụng phép thế [X/An. Y/Football] hai câu trên trở thành đồng nhất like("An", "Football"). Trong các suy diễn, ta cần sử dụng phép hợp nhất các câu bởi các phép thế. Chẳng hạn, cho trước hai câu

friend(x,"Ba")⇒ good(x) (mọi bạn của Ba đều là tốt)

friend("Lan",y) (Lan là bạn của của tất cả mọi người)

Ta có thể hợp nhất hai câu friend(x,"Ba")⇒ good(x) và friend("Lan",y) bởi phép thay thế [X/Lan, Y/Ba]

friend("Lan","Ba")⇒ good("Lan")

friend("Lan","Ba")

Từ hai câu này, theo luật Modus Ponens, ta suy ra câu good("Lan") (Lan là người tốt)

Một cách tổng quát, một phép thế θ là một dãy các cặp Xi/ti, θ = [X1/t1 X2/t2.... Xn/tn] trong đó các Xi là các biến khác nhau, các ti là các hạng thức và các Xi không có mặt trong ti (i=1, ..., n). áp dụng phép thế θ vào công thức G, ta nhận được công thức G0, đó là công thức nhận được từ công thức G bằng cách thay mỗi sự xuất hiện của các Xi bởi ti. Chẳng hạn, nếu G=p(X,Y,f(a,X)) và θ=[X/b, Y/g(Z)] thì G0=P(b,g(Z),f(a,b)).

Với hai câu phân tử G và H mà tồn tại phép thế θ sao cho G0 và H0 trở thành đồng nhất (G0=H0) thì G và H được gọi là hợp nhất được, phép thế θ được gọi là hợp nhất tử của G và H. Chẳng hạn, hai câu Like(An,y) và Like(x, Football) là hợp nhất được bởi hợp nhất tử [X/An, Y/Football]. Vấn đề đặt ra là với hai câu phân tử bất kỳ G và H, chúng có hợp nhất được không và nếu có thì làm thế nào tìm được hợp nhất tử?

c. Luật Modus Ponens tổng quát

Giả sử Pi, Pi' (i=1,...,n) và Q là các công thức phân tử sao cho tất cả các cặp câu Pi, Pi' hợp nhất được bởi phép thế θ, tức là Pi0=Pi0' (i=1,...,n). Khi đó ta có luật:

$$\frac{(Pi \ldots PnQ), Pi', \ldots, Pn'}{Q'}$$

Trong đó Q'=Qθ

Ví dụ 3.16: Giả sử ta có các câu student(x) ∧ male(x) ⇒ like(x,"Football") và student("Anh"), male("Anh"). Với phép thế θ = [x/Anh], các cặp câu student(X), student("Anh") và male(X), male("Anh") hợp nhất được. Do đó ta suy ra câu like("Anh", "Football").

d. Luật phân giải tổng quát

Luật phân giải trên các câu tuyển

Giả sử ta có hai câu tuyển A1 ∨ ... ∨ Am ∨ C và B1 ∨ ... ∨ Bn ∨ ¬ D, trong đó Ai (i=1,..., m) và Bj (j=1,..., n) là các literal, còn C và D là các câu phân tử có thể hợp nhất được bởi phép thế θ, Cθθ=Dθ. Khi đó ta có luật:

$$\frac{A1 \ldots Am C, B1 \ldots Bn D}{A1' \ldots Am' B1' \ldots Bn'}$$

Trong đó Ai'=Aiθ (i=1,..., m) và Bj'=Bjθ (j=1,..., n)

Trong luật phân giải này, hai câu ở tử số (giả thiết) của luật được gọi là hai câu phân giải được, còn hai câu ở mẫu số (kết luận) của luật được gọi là phân giải thức của hai câu ở tử số. Ta sẽ ký hiệu của hai câu A và B là Res(A,B).

Ví dụ 3.17: Giả sử ta có hai câu A=hear(x, "Music") ∨ play(x, "Tennis") và B=¬play("An",y) ∨ study("An"). Hai câu play(X,"Tennis") và play("An", Y) hợp nhất được bởi phép thế θ=[x/An, y/Tennis]. Do đó từ hai câu đã cho, ta suy ra câu hear("An", "Music") ∨ study("An"). Trong ví dụ này, hai câu A= hear(x, "Music") ∨ play(x, "Tennis") và B=¬ play("An", y) ∨ study("An") là phân giải được và phân giải thức của chúng là hear("An", "Music") ∨ study("An").

Luật phân giải trên các câu Horn

Câu Horn (luật If- then) là câu có dạng:

P1 ∧ ... ∧ Pm ⇒ Q

Trong đó Pi (i=1,..., m; m≥ 0) và Q là các câu phân tử.

Giả sử, ta có hai câu Horn P1 ∧ ... ∧ Pm ∧ S ⇒ Q và R1 ∧ ... ∧ Rn ⇒ T, trong đó hai câu S và T hợp nhất được bởi phép thế θ, Sθ=Tθ. khi đó ta có luật:

$$\frac{P1 \ldots Pm S Q, \quad R1 \ldots Rn T}{P1' \ldots Pm' R1' \ldots Rn' Q'}$$

Trong đó, Pi'=Piθ (i=1,..., m), Rj'=Rjθ (j=1,..., n), Q'=Qθ

Trong thực tế, chúng ta thường sử dụng trường hợp riêng sau đây. Giả sử S và T là hai câu phân tử, hợp nhất được bởi phép thế θ. Khi đó ta có luật:

$$\frac{P1 \ldots Pm S Q, \quad T}{P1' \ldots Pm' Q'}$$

Trong đó Pi'=Piθ (i=1,..., m) và Q'=Qθ

Ví dụ 3.18: Xét hai câu student(x) ∧ male(x) ⇒ play(x, "Football") và male("Ba"). Hai câu male("Ba") và male(X) hợp nhất được với phép thế [X/Ba], do đó từ hai câu trên ta suy ra student("Ba") ⇒ play("Ba", "Football").

3.3. BIỂU DIỄN LOGIC VÀ LẬP LUẬN

3.3.1. Biểu diễn tri thức và lập luận với logic mệnh đề

Mệnh đề là một khẳng định, một phát biểu mà giá trị của nó chỉ có thể hoặc là đúng hoặc là sai.

Ví dụ 3.19:

Phát biểu "1+1=2" có giá trị đúng.

Phát biểu "Mọi loại cá có thể sống trên bờ" có giá trị sai.

Giá trị của mệnh đề không chỉ phụ thuộc vào bản thân mệnh đề đó. Có những mệnh đề mà giá trị của nó luôn đúng hoặc sai bất chấp thời gian nhưng cũng có những mệnh đề mà giá trị của nó lại phụ thuộc vào thời gian, không gian và nhiều yếu tố khác quan khác. Chẳng hạn như mệnh đề: "Con người không thể nhảy cao hơn 5m với chân trần" là đúng khi ở trái đất, còn ở những hành tinh có lực hấp dẫn yếu thì có thể sai.

Ta ký hiệu mệnh đề bằng những chữ cái la tinh như a, b, c, ...

Có 3 phép nối cơ bản để tạo ra những mệnh đề mới từ những mệnh đề cơ sở là phép hội (∨), giao (∧) và phủ định (¬)

Bạn đọc chắn hẳn đã từng sử dụng logic mệnh đề trong chương trình rất nhiều lần (như trong cấu trúc lệnh IF ... THEN ... ELSE) để biểu diễn các tri thức "cứng" trong máy tính.

Bên cạnh các thao tác tính ra giá trị các mệnh đề phức từ giá trị những mệnh đề con, chúng ta có được một cơ chế suy diễn như sau:

Modus Ponens: Nếu mệnh đề A là đúng và mệnh đề A→ B là đúng thì giá trị của B sẽ là đúng.

Modus Tollens: Nếu mệnh đề A→ B là đúng và mệnh đề B là sai thì giá trị của A sẽ là sai.

Tri thức là một khái niệm rất trừu tương, và gần như không có một định nghĩa tường minh nào phù hợp mà đươjc chấp nhận. Vì vậy, chúng ta sẽ không cố gắng đưa ra một định nghĩa chính xác về tri thức mà chỉ cố gắng "cảm nhận" khái niệm "tri thức" một cách gần gũi nhất thông qua hai khái niệm khác, đó là thông tin và dữ liệu.

Trong khoa học máy tính, người ta thường quan niệm dữ liệu là các con số, chữ cái, hình ảnh, âm thanh …mà máy tính có thể tiếp nhận và xử lý. Bản thân dữ liệu thường không có ý nghĩa đối với con người. Còn thông tin là tất cả những gì mà còn người có thể cảm nhận được thông qua các phương tiện kỹ thuật như truyền thanh, truyền hình, một cách trực tiếp nhờ các giác quan của mình. Thông tin đối với con người luôn có một ý nghĩa nhất định nào đó.

Các dữ liệu được sắp xếp theo một thứ tự nhất định hoặc được tập hợp lại theo một quan hệ nào đó sẽ chứa đựng thông tin. Nếu các quan hệ này được chỉ ra một cách rõ ràng thì đó là tri thức.

Mục tiêu của Trí tuệ nhân tạo là tạo ra các sản phẩm có khả năng nhận thức, suy luận và phản ứng. Nhận thức được hiểu là khả năng quan sát, học hỏi, hiểu biết cũng như

có kinh nghiệm về thế giới xung quanh. Quá trình nhận thức giúp con người có tri thức. Suy luận là khả năng vận dụng những tri thức sẵn có để phản ứng với những tình huống thực tế. Tri thức tuy không quyết định sự thông minh, nhưng nó là một yếu tố cơ bản cấu thành trí thông minh. Chính vì vậy, để xây dựng một trí thông minh nhân tạo, cần phải có yếu tố cơ bản này.

Giữa tri thức và dữ liệu có mối liên hệ chặt chẽ và không có ranh giới rõ ràng. Một cách hình thức, có thể phân biệt giữa tri thức và dữ liệu dựa vào các yếu tố như là: Tri thức thường không có cấu trúc trong khi dữ liệu thì có cấu trúc hoặc tri thức thì định tính trong khi dữ liệu thì định lương.

Để máy tính có thể sử dụng được tri thức, có thể xử lý được tri thức, chúng ta cần phải biểu diễn tri thức dưới dạng thuận tiện cho máy tính. Đó là mục tiêu của biểu diễn tri thức. Có một số cách biểu diễn tri thức như sau:

Biểu diễn tri thức bằng logic.

Mọi tri thức được diễn đạt dưới dạng các biểu thức logic (mệnh đề hay vị từ).

Biểu diễn tri thức bằng mạng ngữ nghĩa.

Phương pháp biểu diễn tri thức bằng cách dùng một đồ thị G = (V, E) gồm tập đỉnh V và tập cung E. trong đó các đỉnh ứng với các đối tượng, khái niệm hay sự kiện cụ thể, các cung thể hiện quan hệ giữa các đối tượng. Có một cung nối giữa hai đối tượng a và đối tượng b (ký hiệu a$\rightarrow$ b) nếu có một quan hệ nào đó giữa hai đối tượng a, b.

Biểu diễn tri thức bằng khung (Frame)

Khung thực chất là sự tổng quát hoá của cấu trúc bản ghi trong Pascal và tương tự như cấu trúc đối tượng trong C++. Một khung được mô tả bởi cấu trúc:

- ✓ Tên khung: Định danh đối tượng mô tả
- ✓ Các khe (slot): trên mỗi khe lưu trữ các thông tin, n\miền giá trị, thuộc tính và chiều mũi tên chỉ đến các khung khác

Biểu diễn tri thức bằng các luật sản xuất

Phương pháp biểu diễn tri thức nhờ logic (logic mệnh đề và logic vị từ) khá trực quan song chỉ phù hợp khi không có quá nhiều luật suy diễn. Một tri thức được thể hiện bằng một câu Horn dạng chuẩn:

$p1 \wedge p2 \wedge ....\wedge pn \rightarrow q$

các câu Horn dạng này còn được gọi là luật if- then và được biểu diễn như sau:

if P_1 and....and P_n then Q

Một câu Horn dạng tổng quát:

$$p_1 \wedge p_2 \wedge ....\wedge p_n \rightarrow q_1 \vee q_2 \vee ....\vee q_m$$

Lưu ý:

Nếu có luật dạng: $p_1 \wedge p_2 \wedge ....\wedge p_n \rightarrow q_1 \vee q_2 \vee ....\vee q_m$ thì có nghĩa là có m luật sau:

$$p_1 \wedge p_2 \wedge ....\wedge p_n \wedge \neg q_2 \wedge ....\wedge \neg q_m \rightarrow q_1$$

$$p_1 \wedge p_2 \wedge ....\wedge p_n \wedge \neg q_1 \wedge \neg q_3 ...\wedge \neg q_m \rightarrow q_2$$

. . .

$$p_1 \wedge p_2 \wedge ....\wedge p_n \wedge \neg q_1 ....\wedge \neg q_{m-1} \rightarrow q_m$$

Tuy nhiên ta chỉ xét câu Horn dạng chuẩn (m=1)

- Nếu n=0, m=1: câu Horn có dạng: $\rightarrow$ q , q được gọi là sự kiện (fact).
- Nếu n>0, m=1: câu Horn có dạng: $p_1 \wedge p_2 \wedge ....\wedge p_n \rightarrow$ q được gọi là luật (rule).

Trong các hệ chuyên gia, cơ sở tri thức gồm 2 phần: tập các sự kiện (facts) và tập luật (rules).

3.3.2. Biểu diễn tri thức và lập luận với logic vị từ

Biểu diễn tri thức bằng mệnh đề gặp phải một trở ngại cơ bản là ta không thể can thiệp vào cấu trúc của một mệnh đề. Hay nói một cách khác là mệnh đề không có cấu trúc. Điều này làm hạn chế rất nhiều thao tác suy luận. Do đó, người ta đã đưa vào khái niệm vị từ và lượng từ ($\forall$ - với mọi, $\exists$ - tồn tại) để tăng cường tính cấu trúc của một mệnh đề.

Trong logic vị từ, một mệnh đề được cấu tạo bởi hai thành phần là các đối tượng tri thức và mối liên hệ giữa chúng (gọi là vị từ). Các mệnh đề sẽ được biểu diễn dưới dạng:

Vị từ (<đối tượng 1>, <đối tượng 2>, …, <đối tượng n>)

Như vậy để biểu diễn vị của các trái cây, các mệnh đề sẽ được viết lại thành:

Cam có vị Ngọt $\Rightarrow$ Vị (Cam, Ngọt)

Cam có màu Xanh $\Rightarrow$ Màu (Cam, Xanh)

Kiểu biểu diễn này có hình thức tương tự như hàm trong các ngôn ngữ lập trình, các đối tượng tri thức chính là các tham số của hàm, giá trị mệnh đề chính là kết quả của hàm (thuộc kiểu BOOLEAN).

Với vị từ, ta có thể biểu diễn các tri thức dưới dạng các mệnh đề tổng quát, là những mệnh đề mà giá trị của nó được xác định thông qua các đối tượng tri thức cấu tạo nên nó.

Chẳng hạn tri thức: "A là bố của B nếu B là anh hoặc em của một người con của A" có thể được biểu diễn dưới dạng vị từ như sau:

Bố (A, B) = Tồn tại Z sao cho: Bố (A, Z) và (Anh(Z, B) hoặc Anh(B,Z))

Trong trường hợp này, mệnh đề Bố(A,B) là một mệnh đề tổng quát

Như vậy nếu ta có các mệnh đề cơ sở là:

a) Bố ("An", "Bình") có giá trị đúng (An

b) là bố của Bình)

b) Anh("Tú", "Bình") có giá trị đúng (Tú là anh của Bình)

thì mệnh đề c) Bố ("An", "Tú") sẽ có giá trị là đúng. (An là bố của Tú).

Rõ ràng là nếu chỉ sử dụng logic mệnh đề thông thường thì ta sẽ không thể tìm được một mối liên hệ nào giữa c và a, b bằng các phép nối mệnh đề $\wedge$, $\vee$, $\neg$. Từ đó, ta cũng không thể tính ra được giá trị của mệnh đề c. Sở dĩ như vậy vì ta không thể thể hiện tường minh tri thức "(A là bố của B) nếu có Z sao cho (A là bố của Z) và (Z anh hoặc em C)" dưới dạng các mệnh đề thông thường. Chính đặc trưng của vị từ đã cho phép chúng ta thể hiện được các tri thức dạng tổng quát như trên.

Phát biểu "Không có vật gì là lớn nhất và không có vật gì là bé nhất!" có thể được biểu diễn dưới dạng vị từ như sau:

LớnHơn(x,y) = x>y

NhỏHơn(x,y) = x<y

$\forall$ x, $\exists$ y : LớnHơn(y,x) và $\forall$ x, $\exists$ y : NhỏHơn(y,x)

Câu châm ngôn "Gần mực thì đen, gần đèn thì sáng" được hiểu là "chơi với bạn xấu nào thì ta cũng sẽ thành người xấu" có thể được biểu diễn bằng vị từ như sau:

NgườiXấu (x) = $\exists$ y : Bạn(x,y) và NgườiXấu(y)

3.3.3. Biểu diễn tri thức bằng luật sinh và cơ chế suy diễn trên tập luật sinh

Ý tưởng cơ bản là tri thức có thể được cấu trúc bằng một cặp điều kiện – hành động: "NẾU điều kiện xảy ra THÌ hành động sẽ được thi hành". Chẳng hạn: NẾU đèn giao thông là đỏ THÌ bạn không được đi thẳng, NẾU máy tính đã mở mà không khởi động được THÌ kiểm tra nguồn điện, …

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

Một cách tổng quát luật sinh có dạng như sau:

$$\mathbf{P_1 \wedge P_2 \wedge ... \wedge Pn \rightarrow Q}$$

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

Trong logic vị từ : P_1, P_2, ..., Pn, Q là những biểu thức logic.

Trong ngôn ngữ lập trình, mỗi một luật sinh là một câu lệnh.

IF (P_1 AND P_2 AND .. AND Pn) THEN Q.

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

ONE $\rightarrow$ một.

TWO $\rightarrow$ hai.

JANUARY $\rightarrow$ tháng một

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

(1) Tập các sự kiện F(Facts)

F = { f_1, f_2, ... fn }

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

f_1 ^ f_2 ^ ... ^ fi $\rightarrow$ q

Trong đó, các fi , q đều thuộc F

Ví dụ 3.20: Cho 1 cơ sở tri thức được xác định như sau :

Các sự kiện : A, B, C, D, E, F, G, H, K

Tập các quy tắc hay luật sinh (rule)

R1 : A $\rightarrow$ E

R2 : B $\rightarrow$ D

R3 : H $\rightarrow$ A

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

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

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

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

Cơ chế suy luận trên các luật sinh

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

Sự kiện ban đầu : H, K

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

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

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

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

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

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

Ví dụ 3.21:

Tập các sự kiện :

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

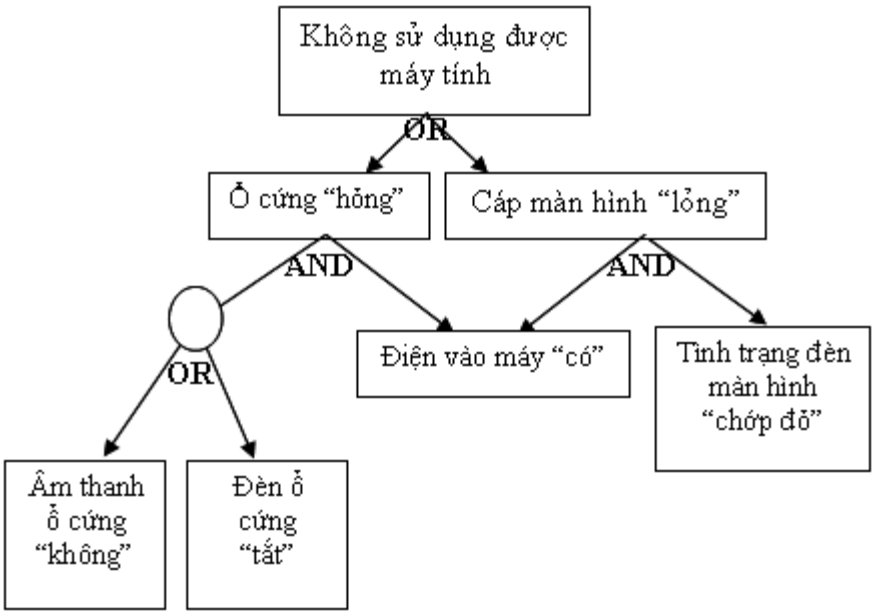
Tập các luật:

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

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

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

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



Hình 3.1: Biểu diễn tri thức bằng luật sinh

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

Vấn đề tối ưu luật

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

Rút gọn bên phải

Luật sau hiển nhiên đúng:

$\mathbf{A} \wedge B \rightarrow \mathbf{A}$

Do đó luật

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

Là hoàn toàn tương đương với

$A \wedge B \rightarrow C$

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

Rút gọn bên trái

Xét các luật :

(L1) $A, B \rightarrow C$

(L2) $A \rightarrow X$

(L3) $X \rightarrow C$

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

Phân rã và kết hợp luật

Luật $A \vee B \rightarrow C$

Tương đương với hai luật

$A \rightarrow C$

$B \rightarrow C$

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

Luật thừa

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

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

Thuật toán tối ưu tập luật dẫn

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

Thuật toán bao gồm các bước chính

B1 : Rút gọn vế phải
Với mỗi luật r trong R
Với mỗi sự kiện A A ∈ VếPhải(r)
Nếu A ∈ VếTrái(r) thì Loại A ra khỏi vế phải của R.
Nếu VếPhải(r) rỗng thì loại bỏ r ra khỏi hệ luật dẫn : R = R – {r}
B2 : Phân rã các luật
Với mỗi luật r : $X_1 \vee X_2 \vee \ldots \vee Xn \rightarrow Y$ trong R
Với mỗi i từ 1 đến n R := R + { Xi → Y }
R := R – {r}
B3 : Loại bỏ luật thừa
Với mỗi luật r thuộc R
Nếu VếPhải(r) ∈ BaoĐóng(VếTrái(r), R-{r}) thì R := R – {r}
B4 : Rút gọn vế trái
Với mỗi luật dẫn r : X : $A_1 \wedge A_2$, …, An → Y thuộc R
Với mỗi sự kiện Ai thuộc r
Gọi luật r_1 : X – Ai → Y
S = (R – {r}) U {r_1}
Nếu BaoĐóng(X – Ai , S) ≡ BaoĐóng(X, R) thì loại sự kiện A ra khỏi X

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

Ưu điểm

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

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

Nhược điểm

Các tri thức phức tạp đôi lúc đòi hỏi quá nhiều (hàng ngàn) luật sinh. Điều này sẽ làm nảy sinh nhiều vấn đề liên quan đến tốc độ lẫn quản trị hệ thống.

Thống kê cho thấy, người xây dựng hệ thống trí tuệ nhân tạo thích sử dụng luật sinh hơn tất cả phương pháp khác (dễ hiểu, dễ cài đặt) nên họ thường tìm mọi cách để biểu diễn tri thức bằng luật sinh cho dù có phương pháp khác thích hợp hơn! Đây là nhược điểm mang tính chủ quan của con người.

Cơ sở tri thức luật sinh lớn sẽ làm giới hạn khả năng tìm kiếm của chương trình điều khiển. Nhiều hệ thống gặp khó khăn trong việc đánh giá các hệ dựa trên luật sinh cũng như gặp khó khăn khi suy luận trên luật sinh.

3.3.4. Lập luận xấp xỉ và suy diễn sác xuất

3.3.4.1. Vấn đề thông tin không chắc chắn khi suy diễn và giải quyết vấn đề

Các nguyên nhân của sự không chắc chắn:

- ✓ Dữ liệu/thông tin/tri thức có thể: không đủ, không đáng tin cậy, không đúng, không chính xác
- ✓ Các phép suy luận có thể không hợp logic: suy luận ngược từ kết luận về điều kiện (abduction reasoning)

Xử lý trường hợp không chắc chắn:

- ✓ Tiếp cận thống kê: quan tâm đến mức độ tin tưởng của một khẳng định.
  - o Lý thuyết xác suất Bayesian (Bayesian Probability Theory)
  - o Đại số chắc chắn Stanford (The Stanford Certainty Algebra)
- ✓ Suy luận theo logic mờ (Fuzzy Logic) quan tâm đến mức độ đúng (truth) của một khẳng định.

3.3.4.2. Phương pháp Bayes và các khái niệm xác suất liên quan

Phương pháp Bayes là các phương pháp suy luận xấp xỉ dựa trên lý thuyết về xác suất.

Được sử dụng để:

- ✓ Mô tả một thế giới hoàn toàn ngẫu nhiên (chơi bài,…)
- ✓ Mô tả một thế giới bình thường (mối tương quan thống kê,…)
- ✓ Mô tả các ngoại lệ (tỉ lệ xuất hiện lỗi,…)
- ✓ Làm cơ sở cho việc học của máy (quy nạp cây quyết định,…)

Thường xác suất được dùng cho:

- ✓ Sự kiện: xác suất của việc quan sát một chứng cớ nào đó.
- ✓ Giả thuyết: xác suất để giả thuyết đúng.

Theo xác suất truyền thống: tần số xuất hiện tương đối của một sự kiện trong một thời gian dài sẽ tiến đến xác suất của nó.

$P(e) \in [0,1]$

$P(e_1) + P(e_2) + \ldots + P(e_n) = 1$

Ví dụ 3.22: đồng xu tốt khi P(mặt_sấp) = P(mặt_ngửa) = 0.5

đồng xu không đều khi P(mặt_sấp) =0.7 P(mặt_ngửa) = 0.3

Nếu sự kiện e_1 và e_2 độc lập nhau:

$P(e_1 \text{ and } e_2) = P(e_1) * P(e_2)$

$P(e_1 \text{ or } e_2) = P(e_1) + P(e_2) - P(e_1) * P(e_2)$

$P(\text{not } e) = 1 – P(e)$

Ví dụ 3.23:

Tung 2 đồng xu (có mặt sấp S và ngửa N): các khả năng có thể xảy ra là SS SN NS NN, suy ra:

P(S and N) = ¼ = 0.25 P(S or S) = ¾ = 0.75

Xác suất tiên nghiệm (prior probability) hay xác suất vô điều kiện (unconditional probability): là xs của một sự kiện trong điều kiện không có tri thức bổ sung cho sự có mặt hay vắng mặt của nó.

Xác suất hậu nghiệm (posterior probability) hay xác suất có điều kiện (conditional probability): là xác suất của một sự kiện khi biết trước một hay nhiều sự kiện khác

$$P(e1|e2) = \frac{|e1 \wedge e2|}{|e2|}$$

Ví dụ 3.24:

P(cúm) = 0.001 P(sốt) = 0.003
P(cúm and sốt) = 0.000003
nhưng cúm và sốt là cá sự kiện không độc lập
các chuyên gia cho biết: P(sốt | cúm) = 0.9

P(h|e) là xác suất khẳng định *giả thuyết* h đúng cho trước *bằng chứng* e.

$$P(h|e)=\frac{P(e|h).P(h)}{P(e)}$$ (luật Bayes)

Công thức này nói rằng xác suất đúng của giả thuyết h khi quan sát được bằng chứng e, bằng với xác xuất cho rằng chúng ta sẽ quan sát được bằng chứng e nếu giả thuyết h là đúng, nhân với xác suất tiên nghiệm của h, tất cả chia cho xác suất tiên nghiệm của việc quan sát được bằng chứng e.

Ví dụ 3.25:

Bằng chứng (triệu chứng): bệnh nhân bị sốt

Giả thuyết (bệnh): bệnh nhân bị cảm cúm

$$P(cúm|sốt)=\frac{P(sốt|cúm).P(cúm)}{P(sốt)}=\frac{0.001.0.9}{0.003}=0.3$$

Các con số ở vế phải thì dễ đạt được hơn con số ở vế trái

Khi nào bằng chứng e không làm tăng xác suất đúng của giả thuyết h?

Khi xác suất của giả thuyết h đã là 1.0

Khi bằng chứng e không liên quan gì đến giả thuyết h

Tại sao sử dụng luật Bayes?

Tri thức về nguyên nhân (knowledge of causes): P (sốt | cúm) thì dễ dàng có được hơn là tri thức về chẩn đoán (diagnostic knowledge): P (cúm | sốt).

Luật Bayes cho phép chúng ta sử dụng tri thức về nguyên nhân để suy ra tri thức về chẩn đoán.

Các vấn đề trong suy luận Bayes: Việc tính toán các xác suất tiên nghiêm và hậu nghiệm liên quan đòi hỏi một sự thu thập dữ liệu rất lớn.

Trong thực tế phải xử lý nhiều triệu chứng

- Chỉ có vài triệu chứng là độc lập nhau:
  $P(s_i|s_j) = P(s_i)$
- Nếu chúng không độc lập nhau:
- $$P(d|s1\wedge s2\wedge\ldots\wedge sn)=\frac{P(d).P(s1\wedge s2\wedge\ldots\wedge sn|d)}{P(s1\wedge s2\wedge\ldots\wedge sn)}$$
- Đối với thông tin phủ định:

P(not s) = 1 – P(s) và P(not d | s) = 1 – P(d | s)

Sự độc lập của các điều kiện trong luật Bayes

Trong thực tế có nhiều giả thuyết canh tranh nhau, vì vậy công thức Bayes tổng quát nhất là:

$$P(h_i|e)=\frac{P(e|h_i).P(h_i)}{\sum_k P(e|h_k).P(h_k)}$$

Đòi hỏi tất cả các $P(e | h_k)$ phải độc lập nhau.

Giả sử các chấm đỏ và sốt là độc lập về điều kiện khi cho trước bệnh sởi:

P(các chấm đỏ, sốt | sởi) = P(các chấm đỏ| sởi) P (sốt| sởi)

Khi đó ta có thể kết luận:

P(các chấm đỏ, sốt, sởi) = P(các chấm đỏ, sốt | sởi) P(sởi) = P(các chấm đỏ | sởi) P(sốt | sởi) P(sởi)

CÂU HỎI, BÀI TẬP

1.

Cho cơ sở tri thức sau:

R1: P^Q => R^S

R2: U => P

R3: H =>Q

R4: H

R5: U

Áp dụng thủ tục chứng minh bác bỏ bằng luật phân giải trong logic mệnh đề để chứng minh R.

2.

Cho cơ sở tri thức sau:

1: Mèo(X) ^ Mẹ(X, Y) ^ Đẹp(Y) => Ngoan(X)

2: Mèo(mimi)

3: Mèo(titi)

4: Mèo(kiki)

5: Mẹ(mimi, kiki)

6: Mẹ(kiki, titi)

7: Đẹp(titi)

8: Đẹp(kiki)

Áp dụng suy diễn lùi vào cơ sở tri thức trên để chứng minh Ngoan(kiki).

3.

Cho cơ sở tri thức sau:

1: Bố(X,Y) =>Con(Y,X)

2: Chồng(X, Z) => Vợ(Z, X)

3: Vợ(Z, X) ^ Con(Y, X) => Mẹ(Z, Y)

4: Bố(nam, lan)

5: Chồng(nam, hương)

Áp dụng thủ tục chứng minh bác bỏ bằng luật phân giải trong logic vị từ cấp 1 để chứng minh Mẹ(lan, hương).

4.

Cho cơ sở tri thức và tập luật như sau:

1: a => c

2: d^c =>e

3: a^b => f

4: b => d

5: a^h => i

6: e^f => g

7: a

8: b

Áp dụng thuật toán suy diễn lùi vào cơ sở tri thức trên để chứng minh: g

5.
Cho cơ sở tri thức sau:
R1: P^Q => R^S
R2: U => P
R3: H =>Q
R4: H
R5: U
Áp dụng thủ tục chứng minh bác bỏ bằng luật phân giải trong logic mệnh đề để chứng minh S.
6.
Cho cơ sở tri thức sau:
1: Anh(X, Y) ^ Cưới(Y, Z) => Anh chồng(X, Z)
2: Anh(Minh, Việt)
3: Anh(Long, Nam)
4: Cưới(Việt, Hoa)
5: Cưới (Nam, Hương)
Áp dụng thuật toán suy diễn lùi chứng minh Anh chồng(Minh, Hoa).