TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC

TÔ HOÀI VIỆT - 0012125

NGUYỄN TƯỜNG UYÊN - 0012186

TÌM HIỂU KINH DỊCH - XÂY DỰNG HỆ CHUYÊN GIA DỰ ĐOÁN VÀ KHÁM PHÁ TRI THỨC MỚI

LUẬN VĂN CỬ NHÂN TIN HỌC

GIẢNG VIÊN HƯỚNG DẪN

TS. LÊ HOÀI BẮC

NIÊN KHÓA 2000-2004

Lời cảm ơn

Trước hết chúng em xin chân thành cảm ơn Tiến sĩ Lê Hoài Bắc, người thầy đã giúp gợi mở những ý tưởng ban đầu và tận tâm hướng dẫn chúng em thực hiện khóa luận tốt nghiệp của mình.

Chúng em cũng không quên gởi đến các thầy cô trong Bộ môn Công nghệ tri thức nói riêng, và tất cả các thầy cô khác trong khoa Công nghệ thông tin lời biết ơn chân thành vì đã hết lòng truyền đạt kiến thức trong những năm tháng ở giảng đường đại học.

Và còn một lời cảm ơn nữa xin gởi đến các bạn bè cùng khóa đã chia xẻ những khó khăn trong suốt quá trình học tập và thực hiện khóa luận. Xin chúc các bạn đạt được thành tích tốt nhất.

Thành phố Hồ Chí Minh, tháng 7/ 2004

Danh mục các hình

MỤC LỤC

Tổng quan

Tri thức là tài sản quý giá nhất của nhân loại. Từ xưa, khi con người sống gần gũi và bắt đầu khám phá thiên nhiên, cổ nhân đã phát hiện ra những quy luật vận động của vũ trụ vạn vật. Những tri thức quý báu này vẫn còn được lưu giữ trong Kinh Dịch. Bằng những công cụ toán học hiện đại, người ta đã chứng minh những điều trong Kinh Dịch không phải là mê tín dị đoan mà ngược lại hoàn toàn có căn cứ. Gần đây nhất ở Việt Nam là công trình nghiên cứu của Giáo sư Nguyễn Hoàng Phương, người đã chứng minh Kinh Dịch là một hệ mờ. Thực tế cho thấy, các tri thức Kinh Dịch là những tri thức đã được thống kê, kiểm chứng qua nhiều thế hệ. Những điều này đã đúng trong quá khứ, hiện tại và vẫn đúng trong tương lai vì vũ trụ vẫn muôn đời vận động đúng theo quy luật của nó.

Ngày nay, cùng với sự vận động và phát triển như vũ bão của ngành khoa học máy tính, việc đưa tri thức con người vào máy tính đang là vấn đề được rất nhiều người quan tâm. Ngày càng có nhiều hệ chuyên gia được xây dựng để hỗ trợ hoặc ngay cả thay thế con người trong nhiều lĩnh vực như chẩn đoán bệnh, dự báo thời tiết, các hệ hỗ trợ ra quyết định, các hệ thống tự rút ra tri thức từ dữ liệu đưa vào để bổ sung trở lại vào nguồn tri thức ban đầu… ứng dụng các kỹ thuật trí tuệ nhân tạo.

Từ ý tưởng kết hợp giữa tri thức hiện đại và tri thức cổ, chúng tôi xây dựng một hệ chuyên gia dự đoán. Đây là một hệ thống mở gồm một cơ sở tri thức tách biệt khỏi động cơ suy diễn để người dùng có thể cập nhật tri thức mới bằng tay một cách dễ dàng. Hệ thống còn có khả năng tự động khai thác dữ liệu, rút ra các luật mới làm giàu cơ sở tri thức. Để minh họa cho sự hoạt động của hệ thống, chúng tôi xin tìm hiểu một phần Kinh Dịch và phương pháp dự đoán hôn nhân theo Tứ Trụ, để biểu diễn các luật vào cơ sở tri thức theo cú pháp quy ước sẵn.

Nội dung đề tài:

Chương 1: Trình bày nguồn gốc, các quy luật cơ bản của Kinh Dịch, biểu diễn Kinh Dịch bằng logic mờ, chứng minh không gian Kinh Dịch là một hệ mờ.

Chương 2: Trình bày học thuyết Tứ Trụ - một trong những phương pháp dự đoán của Kinh Dịch, cơ sở khoa học của Tứ Trụ, phương pháp dự đoán hôn nhân theo Tứ Trụ.

Chương 3: Lý thuyết về hệ chuyên gia.

Chương 4: Trình bày một số phương pháp khai thác dữ liệu cơ bản và cải tiến.

Chương 5: Xây dựng chương trình ứng dụng.

Phụ lục: Cách đổi ngày dương lịch sang âm lịch và sang dạng bát tự.

Chương 1: Kinh Dịch – một hệ thống mờ

1.1 Nguồn gốc Kinh Dịch

Kinh Dịch là một loại tài liệu cổ của Trung Quốc xuất hiện cách đây đã mấy ngàn năm. Kinh Dịch có nội dung quy cách hoá sự vận động của tự nhiên, của xã hội theo nhận thức của người Trung Hoa cổ. Hệ thống nhận thức này là một hệ tri thức về không gian, thời gian có tác động và ảnh hưởng tới số phận, hành động của từng người. Con người phải luôn luôn đồng nhất thể và bị chi phối bởi những quy luật vận động trong Không Gian. Cái mốc để xác định sự tác động đó là thời điểm sinh ra sự vật, con người.

Không Gian là nơi con người sinh thành, phát triển. Vị trí tồn tại của con người trong Không Gian Thực sẽ chi phối con người theo một quy luật vận động và phát triển không ngừng. Các sự vật, hiện tượng tồn tại trong Không Gian Thực sẽ ảnh hưởng ràng buộc lẫn nhau. Không Gian Thực được đề cập ở đây là không gian bốn chiều – Không Gian Kinh Dịch.Con người là một đại lượng đặc biệt trong không gian bao la và bị chi phối bởi các Toạ Độ Không Gian (10 can) và Toạ Độ Thời Gian (12 chi) trong suốt quá trình từ lúc sinh ra đến lúc cuối đời.

Chính các nhà Dịch học đã đo đạc và định tính được Không Gian và Thời Gian để tìm ra trị số riêng của từng sự vật, từng con người, từng hiện tượng khi vương vào một không gian cụ thể nào đó. Từ đó suy ra những thông tin làm cơ sở cho dự báo, dự đoán.

Kinh Dịch hướng mỗi người tới sự hòa đồng với tự nhiên theo từng vị trí tồn tại của người đó trong không gian, mỗi người hành động theo đúng quy luật tồn tại của chính mình trong không gian.

Thế giới mà Kinh Dịch diễn tả là thế giới vận động không ngừng. Động lực của sự vận động này, là hai mặt đối lập tồn tại bên nhau và vì nhau trong một khối toàn vẹn và thống nhất, cái mà các nhà Dịch Học cổ gọi là Âm và Dương.

Điều mà sau này đến thế kỷ 18, nhà toán học Đức Leibniz (1646-1716), người sáng lập ra hệ đếm nhị phân đã gán cho ký hiệu biểu thị âm (- -) là con số 0, ký hiệu biểu thị dương (-) là con số 1. Một số tài liệu gọi đây là Lưỡng Nghi (gồm có nghi dương + và nghi âm -)

Cụ thể, cứ sau một thời điểm như vị trí không gian và thời gian (luôn luôn ở dạng động) trước đó là dương thì tiếp ngay sau đó sẽ là âm. Cứ một Tọa độ không gian dương thì có một Tọa độ thời gian tương ứng là dương , nếu là âm thì có một Tọa độ thời gian tương ứng là âm.

TĐKG	Giáp	Ất	Bính	Đinh	Mậu	Kỷ	Canh	Tân	Nhâm	Quý	Giáp	Ất
	+	-	+	-	+	-	+	-	+	-	+	-
TĐTG	Tí	Sửu	Dần	Mão	Thìn	Tị	Ngọ	Mùi	Thân	Dậu	Tuất	Hợi
	+	-	+	-	+	-	+	-	+	-	+	-

Theo Kinh Dịch, không gian nào, thời gian đó. Chính vì vậy khi nói đến sự xuất hiện hay sinh ra một điều gì đó trong một "khu vực" của không gian, bao giờ cũng phải nói đủ cả Tọa độ không gian và Tọa độ thời gian tương ứng như năm Bính Tí, Đinh Sửu… Mỗi một người cụ thể sinh ra từ một Tọa độ không gian với Thời gian tương ứng sẽ có những đặc tính tồn tại và phát triển riêng phù hợp với vị trí của Tọa độ trong Không Gian đó. Chính cái lần sinh độc nhất của mỗi người đã cá biệt hóa số phận của từng người. Chính vì vậy luận thuyết của Dịch học là luận thuyết về nhân sinh, là luận thuyết về vị trí tồn tại của con người trong không gian.

1.2 Học thuyết Âm Dương – Ngũ Hành

1.2.1 Học thuyết Âm Dương

Học thuyết Âm Dương là tư tưởng duy vật biện chứng, là cơ sở lý luận của khoa học tự nhiên và thế giới quan duy vật của Trung Quốc. Sự hình thành, biến hóa và phát triển của vạn vật đều do sự vận động của hai khí âm dương mà ra. Bản thân sự vật, hiện tượng luôn luôn có hai mặt: chất và đối chất, vận động và phản động, vừa mâu thuẫn vừa thống nhất, vừa phủ định vừa khẳng định lẫn nhau.

Âm Dương vừa đối lập vừa thống nhất. Có đối lập mâu thuẫn mới có phát triển vận động; có thống nhất mới có ổn định thành ra vạn vật.

Âm Dương là gốc của nhau, chúng dựa vào nhau để tồn tại. Không có Âm thì không thể xác định Dương và ngược lại.

Âm Dương tiêu giảm và tăng trưởng chỉ sự vận động biến đổi của vạn vật. Mâu thuẫn đối lập của Âm Dương ở trạng thái cái này giảm thì cái kia tăng. Đó là trạng thái cân bằng động, Dương tăng lên thì Âm giảm xuống và ngược lại, chỉ có thế mới giữ được sự phát triển bình thường của sự vật.

Âm Dương có thể chuyển hóa lẫn nhau. Âm đến cực cùng sinh Dương, Dương đến cực cùng sinh Âm.

1.2.2 Học thuyết Ngũ hành

Trong không gian, các đại lượng tồn tại đa hình, đa dạng nhưng tồn tại theo 5 nhóm thuộc tính là tính Kim, tính Mộc, tính Thủy, tính Hỏa, tính Thổ. Các đại lượng trong không gia Kinh Dịch có hay không có 5 thuộc tính nói trên là tùy thuộc vào thời điểm hình thành (sinh vào) tương ứng với các tọa độ không gian (Can) và tọa độ thời gian (Chi).

Hành	Kim	Mộc	Thủy	Hỏa	Thổ

Đặc tính	thanh tĩnh	Mọc lên và phát triển	lạnh rét, hướng xuống dưới	nóng,hướng lên trên	nuôi lớn
Phương	Tây	Đông	Bắc	Nam	Trung
Tương ứng với cơ thể	Phổi, ruột già, khí quản, hệ hô hấp	Gan, mật, gân cốt, tứ chi	Thận, bàng quang, não, hệ bài tiết	Tim, ruột non, mạch máu	Lá lách, dạ dày, hệ tiêu hóa
Màu sắc	Trắng	Xanh	Đen	Hồng	Vàng
Tính tình	Nghĩa	Nhân	Trí	Lễ	Tín

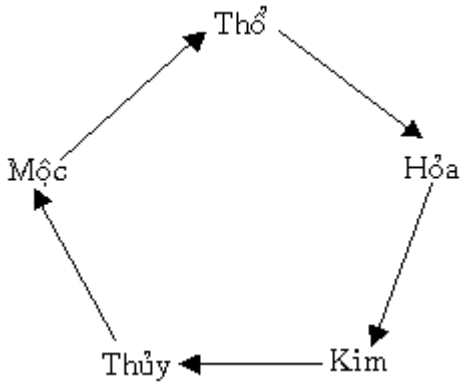
Ngũ hành sinh khắc:

Theo học thuyết Âm Dương và quy luật Nhân quả, khi một hiện tượng xảy ra:

- Do hai nguyên nhân gây ra nó: một nguyên nhân ức chế và một nguyên nhân hưng phấn nó.
- Nó sẽ gây ra hai hậu quả: hưng phấn một hiện tượng khác và ức chế một hiện tượng khác.

Giữa Ngũ Hành tồn tại quy luật tương sinh, tương khắc - quy luật nền tảng của lý thuyết Dịch học. Giống như Âm Dương, tương sinh, tương khắc là hai mặt gắn liền với nhau của sự vật. Không có sinh thì sự vật không thể phát sinh và phát triển. Không có khắc thì không thể duy trì sự cân bằng và điều hòa của sự vật trong quá trình phát triển và biến hóa.

Ngũ Hành tương sinh là:

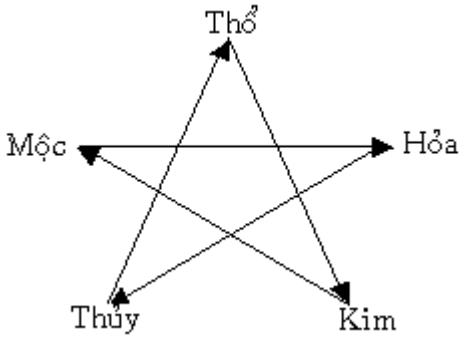


Hình 1: Ngũ hành tương sinh

- Mộc sinh Hỏa: vì Mộc tính ôn hòa, ấm áp tức Hỏa ẩn phục bên trong, xuyên thủng Mộc sẽ sinh ra Hỏa, nên nói Mộc sinh Hỏa.
- Hỏa sinh Thổ: vì Hỏa nóng nên đốt cháy Mộc. Cháy hết biến thành tro tức là Thổ, nên nói Hỏa sinh Thổ.
- Thổ sinh Kim: vì Kim ẩn tàng, vùi lấp trong đá, trong núi, nên nói Thổ sinh Kim.
- Kim sinh Thủy: vì khí của thiếu âm(khí của Kim) chảy ngầm trong núi tức Kim sinh ra Thủy. Làm nóng chảy Kim sẽ biến thành Thủy, nên nói Kim sinh Thủy.
- Thủy sinh Mộc: nhờ Thủy ôn nhuận làm cho cây cối sinh trưởng, nên nói Thủy sinh Mộc.

Ngũ Hành tương khắc:

Ngũ hành tương khắc lẫn nhau đó là bản tính của trời đất: Thủy khắc Hỏa, Hỏa khắc Kim, Kim khắc Mộc, Mộc khắc Thổ, Thổ khắc Thủy.



Hình 2: Ngũ hành tương khắc

Ngũ hành phản sinh, phản khắc:

Ngũ hành phản sinh:

- Mộc sinh Hỏa: Mộc nhiều thì Hỏa không bốc lên được; Hỏa nhiều thì Mộc bị cháy thành than.
- Hỏa sinh Thổ: Hỏa nhiều thì Thổ thành than; Thổ nhiều thì Hỏa chỉ âm ỉ.
- Thổ sinh Kim: Thổ nhiều thì Kim bị vùi lấp; Kim nhiều thì Thổ không còn đáng kể.
- Kim sinh Thủy: Kim nhiều thì nước đục; Thủy nhiều thì Kim chìm xuống.
- Thủy sinh Mộc: Thủy nhiều thì Mộc bị trôi; Mộc thịnh thì Thủy bị co lại.

"Mộc thịnh thì Thủy bị co lại": khi Mộc nhiều thì lấy Kim trị Mộc, lại còn lợi cho Thủy.

"Thổ nhiều thì Hỏa âm ỉ": Thổ nhiều thì lấy Mộc chế Thổ, lại còn lợi cho Hỏa, không được dùng Hỏa vì Hỏa sinh Thổ, Thổ sẽ càng vượng.

Ngũ hành phản khắc:

- Thủy khắc Hỏa: nhưng Hỏa nhiều thì Thủy bị bốc hơi.
- Hỏa khắc Kim: Kim nhiều thì Hỏa tắt.
- Kim khắc Mộc:Mộc quá cứng thì Kim phải mẻ.
- Mộc khắc Thổ: Thổ nặng thì Mộc bị thắt lại.

- Thổ khắc Thủy: Thủy nhiều thì Thổ bị trôi.

1.3 Kinh dịch – một hệ mờ

Kinh Dịch là một đề tài nghiên cứu nghiêm túc đã thu hút được sự quan tâm của nhiều bộ óc vĩ đại của dân tộc như Lê Quí Đôn, Ngô Tất Tố, Nguyễn Duy Cần, Nguyễn Hiến Lê… Trên thế giới cũng có nhiều nhà khoa học lớn nghiên cứu về Kinh Dịch, tiêu biểu như công trình của nhà toán học Đức Leibniz (1646-1716) đã biểu diễn các quẻ của Kinh dịch bằng các dấu hiệu nhị phân (0 và 1). Ở Việt Nam trong những năm gần đây cũng xuất hiện công trình khoa học của Giáo sư tiến sĩ Nguyễn Hoàng Phương ([1]), trong đó, bằng cách hình thức hóa các cấu trúc của Kinh dịch bằng tập mờ, ông đã chứng minh được:

- Triết học cổ Đông Phương mà cốt lõi là Kinh dịch là một loại khoa học tiền đề, lấy khung Thái cực, Lưỡng Nghi, Tứ tượng, Ngũ hành, Bát quái… làm tiền đề
- Triết học cổ Phương Đông là một tập mờ.

Ở dây, chúng tôi xin được trích dẫn một phần công trình của ông với mục đích tham khảo, đó là hình thức quá các cấu trúc của Kinh dịch bằng tập mờ.

1.3.1 Cấu trúc quẻ của triết cổ Đông phương

Thái cực và Lưỡng nghi

Thái cực – xem như Vũ trụ toàn bộ - có thể phân cực thành Âm và Dương, gọi là Lưỡng Nghi, là Nghi Dương và Nghi Âm. Nghi Dương được biểu thị bằng một nét liền liên tục, còn Nghi Âm bằng một nét đứt không liên tục.

Những cách phối hợp đơn giản nhất của các đường liên tục và không liên tục trên lần lượt cho các cấu trúc sau:

Tứ tượng và Ngũ hành

Tứ tượng mô tả quá trình sinh ra, lớn lên, già đi, rồi mất đí hay là quá trình : Thành, Thịnh, Suy, Hủy.

Tứ tượng được biểu thị bằng một tập hợp gồm hai đường liên tục hoặc không liên tục.

Nếu thêm trung tâm vào, ta được cấu trúc Ngũ hành với ý nghĩa: Sinh, Trưởng, Hóa, Thâu, Tàng.

Bát quái

Có hai loại Bát quái : Bái quái tiên thiên và Bát quái Hậu thiên. Bát quái có thể xem là kết quả tổ hợp gồm ba đường với mục đích mô tả những quá trình phức tạp hơn, hoặc trong không gian hoặc trong thời gian, ví dụ mô tả các nửa mùa trong năm về mặt thời gian, hoặc là mô tả tám phương trong không gian.

Cấu trúc Bát quái Tiên thiên hay là Bát quái Đồ Phục Hy mang tính đối xứng tâm cao độ, còn Bát quái Hậu thiên hay Bát quái Đồ Văn Vương thì kém đối xứng hơn. Tính đối xứng cao của Bát quái đồ Tiên thiên biểu hiện những tồn tại tương đối hoàn hảo. Trong lúc đó thì tính kém đối xứng của Bát quái Đồ Văn Vương lại biểu hiện được những tồn tại kém hoàn chỉnh hơn (của cõi trần chúng ta chẳng hạn).

1.3.2 Lý thuyết tập kinh điển:

Hàm thuộc và vũ trụ:

Cho Y là một tập kinh điển, gọi là Vũ trụ (tập mờ) hay Hệ quy chiếu, chẳng hạn là

$$Y= \{a,b,c,d,e\}= \{y\}$$

Ta hãy lấy một số tập con của Y, ví dụ là

$$A= \{a,c,d,e\},\ B= \{ c, d, e\},\ C= \{a, b\}$$

Để xác định các tập con đó, người ta đưa ra khái niệm hàm thuộc hay là hàm thành phần, ký hiệu là μ, định nghĩa như sau:

$$M_A(y) \equiv A(y) = \begin{cases} 0 & \text{khi và chỉ khi y không thuộc A} \\ 1 & \text{khi và chỉ khi y thuộc A} \end{cases}$$

(ký hiệu A(y) dựa vào công trình của Negotia, để viết cho đơn giản).

Tập hai phần tử {0, 1} gọi là tập đánh giá. Với các ví dụ trên, theo định nghĩa của hàm thuộc, ta có chẳng hạn:

$$A(a) = 1,\ A(b) = 0,\ B(a) = 0,\ \ldots$$

Tính chất:

$$Y(y) = 1,\ \varnothing(y) = 0 \text{ với mọi y.}$$

1.3.2.1 Các phép toán và tính chất trong lý thuyết cổ điển

Lý thuyết các tập kinh điển dựa trên ba phép toán sau:

Phép hợp:

Phép hợp ký hiệu là $\cup$ với định nghĩa:

$$(A \cup B)(y) = \text{Max}\{A(y), B(y)\},\ A(y), B(y) \in [0, 1].$$

Phép giao:

Phép giao ký hiệu là $\cap$ với định nghĩa:

$$(A \cap B)(y) = \text{Min}\{A(y), B(y)\},\ A(y), B(y) \in [0, 1].$$

Phép bổ sung:

Phép bổ sung ký hiệu với một dấu gạch ngang trên đầu với định nghĩa:

$$\overline{A}(y) = 1 - A(y), \text{ với mọi y, } A(y) \in [0,1].$$

Tính chất của phép toán

Giữa các phép toán đó chúng ta có các tính chất sau:

a) Tính giao hoán: $A \cap B = B \cap A,\ A \cup B = B \cup A.$

b) Tính kết hợp: $A \cap (B \cap C) = (A \cap B) \cap C$,

$A \cup (B \cup C) = (A \cup B) \cup C$

c) Tính luỹ đẳng: $A \cap A = A$,

$A \cup A = A$.

d) Tính phân phối: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

e) Tính đồng nhất: $A \cup \varnothing = A$, $A \cap Y = A$

f) Tính: $A \cap \varnothing = \varnothing$

$A \cap Y = Y$.

g) Tính hấp thụ: $A \cap (B \cup A) = A$,

$A \cup (B \cap A) = A$.

h) Tính đối hợp: $\overline{\overline{A}} = A$

i) Các qui tắc De Morgan: $\overline{A \cup B} = \overline{A} \cap \overline{B}$

$\overline{A \cap B} = \overline{A} \cup \overline{B}$

j) Nguyên lý bài trung (mâu thuẫn):

$A \cap \overline{A} = \varnothing$

$A \cup \overline{A} = Y$.

1.3.3 Lý thuyết mờ theo Zadeh và nguyên lý phi bài trung:

Năm 1965, A. Zadeh sáng tạo ra lý thuyết tập mờ. Khác với lý thuyết tập kinh điển, ông mở rộng tập đánh giá của hàm thuộc từ tập rời rạc {0,1} sang tập liên tục [0,1], nghĩa là giá trị của hàm thuộc không phải chỉ là 0 hoặc 1, mà trải một cách liên tục từ 0 đến 1.

1.3.3.1 Định nghĩa:

Các định nghĩa về các phép giao, hợp, bổ sung vẫn giữ nguyên như cũ, tức là phép hợp vẫn định nghĩa theo Max, phép giao theo Min và phép bổ sung theo phép trừ trong lý thuyết tập kinh điển.

Cần nói thêm về phép bao và quan hệ bằng nhau, tuân theo định nghĩa sau:

$$A \overset{*}{\supseteq} B \Leftrightarrow \{A(y) \geq B(y)\} \text{ với mọi y.}$$

Cách tiếp cận này của L.A.Zadeh có hai đặc tính: Một là hết sức đơn giản, hai là có tính kế thừa so với lý thuyết tập kinh điển về mặt định nghĩa các phép toán qua Max, Min và phép trừ.

Điểm nổi bật nhất của lý thuyết Zadeh là tất cả các tính chất của các phép toán của lý thuyết tập kinh điển đều được giữ nguyên, trừ một tính chất chủ yếu : Nguyên lý bài trung của Chủ nghĩa duy lý không còn đúng nữa, như sẽ thấy ngay sau đây.

1.3.3.2 Nguyên lý phi bài trung trong lý thuyết Zadeh:

Trong lý thuyết Zadeh, nguyên lý bài trung, gắn bó với chủ nghĩa Duy lý, không còn đúng nữa. Chúng ta hãy lấy vài ví dụ cụ thể:

Cho vũ trụ Y = {a, b, c, d} và A là một tập con của Y, với

$$A(a) = 0{,}2,\ A(b) = 0{,}4,\ A(c) = 0{,}8,\ A(d) = 0$$

Ta được ngay, theo định nghĩa về phép bổ sung:

$$A(a) = 0{,}8,\quad A(b) = 0{,}6,\quad A(c) = 0{,}2,\quad A(d) = 1.$$

Từ đó ta được theo các phép toán Max và Min trong các định nghĩa trên:

$(A \cup \overline{A})(a) = 0{,}8;\ (A \cup \overline{A})(b) = 0{,}6;\ (A \cup \overline{A})(c) = 0{,}8;\ (A \cup \overline{A})(d) = 1$

$(A \cap \overline{A})(a) = 0{,}2;\ (A \cap \overline{A})(b) = 0{,}4;\ (A \cap \overline{A})(c) = 0{,}2;\ (A \cap \overline{A})(d) = 0$

Nhưng vì Y(y) = 1, (y) = 0, với mọi y, nên theo đẳng thức các tập mờ, ta có ngay kết quả hết sức quan trọng sau trong lý thuyết Zadeh:

Nguyên lý phi bài trung

$$A \cap \overline{A} \neq 0\ ,$$

$$A \cup \overline{A} \neq Y.$$

tức là nguyên lý bài trung - từ lâu xem như một chân lý vĩnh cửu - quả thực không còn đúng với định nghĩa của Zadeh.

Kết quả này rất quan trọng trong việc tìm một phương hướng toán học thích hợp cho Triết cổ phương Đông, là khoa học chủ yếu nghiên cứu các quy luật, cấu trúc của tư tưởng, trong đó trong Âm (A) có Dương (A), và trong Dương có Âm, âm dương như thế đều là những khái niệm có "nội dung mờ".

1.3.3.3 Các tính chất của phép toán trên tập mờ:

Như đã đề cập ở trên, hầu hết các tính chất của phép toán trên tập kinh điển vẫn còn đúng trên tập mờ ngoại trừ định lý phi bài trung:

a) Tính giao hoán: $A \cap B = B \cap A$, $A \cup B = B \cup A$.

b) Tính kết hợp: $A \cap (B \cap C) = (A \cap B) \cap C$,

$A \cup (B \cup C) = (A \cup B) \cup C$

c) Tính luỹ đẳng: $A \cap A = A$,

$A \cup A = A$.

d) Tính phân phối: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

e) Tính đồng nhất: $A \cup \varnothing = A$, $A \cap Y = A$

f) Tính: $A \cap \varnothing = \varnothing$

$A \cap Y = Y$.

g) Tính hấp thụ: $A \cap (B \cup A) = A$,

$A \cup (B \cap A) = A$.

h) Tính đối hợp: $A = A$

i) Các qui tắc De Morgan: $\overline{A \cup B} = \overline{A} \cap \overline{B}$

$\overline{A \cap B} = \overline{A} \cup \overline{B}$

j) Nguyên lý phi bài trung (phi mâu thuẫn):

$$A \cap \overline{A} \neq \varnothing$$

$$A \cup \overline{A} \neq Y.$$

1.3.3.4 Phép nhân Max Min và phép nhân Min Max

Trong quá trình vận dụng lý thuyết tập mờ, xuất hiện một số phép nhân đặc biệt gọi là phép nhân Max Min và phép nhân Min Max, định nghĩa như sau:

Phép nhân Max Min

Phép nhân này tương tự như phép nhân ma trận thông thường, chỉ khác một chỗ là: Thay phép nhân thông thường bằng phép lấy Min, còn phép cộng thông thường bằng phép lấy Max.

Ví dụ

Ta chọn trường hợp ma trận 2x2 cho đơn giản. Chẳng hạn ta có trong phép nhân ma trận thông thường

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} m & n \\ p & q \end{pmatrix} = \begin{pmatrix} am + bp & an + bq \\ cm + dp & cn + dq \end{pmatrix}$$

Phép nhân Max Min, kí hiệu là o, theo định nghĩa sẽ có dạng:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} o \begin{pmatrix} m & n \\ p & q \end{pmatrix} = \begin{pmatrix} Max(Min(a, m), Min(b, p)) & Max(Min(a, n), Min(b, q)) \\ Max(Min(c, m), Min(d, p)) & Max(Min(c, n), Min(d, p)) \end{pmatrix}$$

Ví dụ bằng số

$$\begin{pmatrix} 2 & 6 \\ 4 & 5 \end{pmatrix} o \begin{pmatrix} 0 & 1 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 5 & 5 \end{pmatrix}$$

Phép nhân Max Min có tính chất kết hợp:

$$A \text{ o } (B \text{ o } C) = (A \text{ o } B) \text{ o } C.$$

Phép nhân Min Max

Phép nhân này, kí hiệu là $\overline{o}$, suy từ phép nhân Max Min bằng cách thay thế Min và Max cho nhau

Ví dụ:

Phép nhân này cũng có tính chất kết hợp:

$$A \circ (B \circ C) = (A \circ B) \circ C.$$

Phép nhân Max Min và phép nhân Min Max có nhiều ứng dụng sâu xa và nếu có dịp chúng tôi sẽ tiếp tục nghiên cứu.

1.3.4 Sự hình thức hoá cấu trúc lưỡng nghi bằng tập mờ:

Trong phần này trước hết chúng ta tìm cách hình thức hoá cấu trúc Lưỡng Nghi của Kinh Dịch theo lý thuyết mờ Min Max của L.A.Zadeh.

1.3.4.1 Vũ trụ toán học Tây phương - thái cực Đông phương

Vũ trụ, ý hiệu là Y_{AD}, gồm có hai Khí (Nghi) : khí Âm a và khí Dương d,

$$Y_{AD} = \{a,d\} = \text{Thái cực} = \text{Hệ nguyên thuỷ}.$$

Hai khí Âm, Dương này có thể xem là tương ứng với hai quẻ đầu tiên của Kinh Dịch (Trời và Đất) là các Quẻ Kiền và Khôn của Kinh Dịch :

Khí Dương d $\leftrightarrow$ Quẻ Kiền Khí Âm a $\leftrightarrow$ Quẻ Khôn

Theo tinh thần của Kinh Dịch, về sau sẽ có sự phân cực, phân hoá để tạo ra các cõi khác nhau. Mọi tập con A của Thái Cực Y_{AD} đều chứa hai khí đó, với những hàm thuộc có giá trị trong khoảng [0,1].

Ta có

$$A(a,b) \in [0,1] \text{ với mọi } A \in YAD.$$

Đẳng thức và phép bao

Đẳng thức

Hai tập con A và B của Thái cực YAD gọi là bằng nhau, khi và chỉ khi

$$A = B \Leftrightarrow \{\lambda(A) = \lambda(B) \text{ và } A \overset{*}{=} B\}$$

trong đó λ gọi là hàm Âm Dương, với định nghĩa

$$\lambda(A \cap B) = \text{Min } \{\lambda(A), \lambda(B)\},$$

$\lambda(A \cup B) = Max\{\lambda(A), \lambda(B)\}$.

$\lambda(\overline{A}) = -\lambda(A)$, $\lambda(A) = \{1, -1\}$, với mọi A,

còn

$A \overset{*}{=} B \Leftrightarrow \{A(a) = B(a), A(d) = B(d)\}$.

Phép bao

Phép bao định nghĩa như sau:

$A \overset{*}{\supseteq} B \Leftrightarrow A(y) > B(y)$, y= a, d.

Trong công trình này, lý thuyết Zadeh chỉ vận dụng cho loại đẳng thức $\overset{*}{=}$ của Lưỡng Nghi {a, d}, không đi vào phần đẳng thức của hàm Âm Dương λ.

1.3.4.2 Những phương trình cân bằng tĩnh cho Lưỡng Nghi

Có hai loại cân bằng: tĩnh và động

Cân bằng tĩnh

Ta có cân bằng tĩnh cho mọi tập con A khi Âm Dương cân bằng nhau:

Cân bằng tĩnh Âm Dương: $A \overset{*}{=} \overline{A}$ tức là khi.

$A(a) = 1/2$, $A(d) = 1/2$, $\overline{A}(a) = 1/2$, $\overline{A}(d) = 1/2$, $\lambda(A) = -\lambda(\overline{A})$.

Một trong những vấn đề trọng yếu ở đây là xét xem trong cân bằng tĩnh thì, ngoài A và A , còn có xuất hiện những đại lượng nào mới không, quan hệ với A và A bằng các phép tập hợp.

Nếu giả thiết chẳng hạn λ (A) = -1, tức là giả thiết A là Âm thì ta được ngay các kết quả sau:

$\lambda(A \cup \overline{A}) = \lambda(\overline{A}) = 1$,

$\lambda(A \cup \overline{A}) = \lambda(\overline{A}) = -1$,

$A \cup \overline{A} \overset{*}{=} \overline{A}$, $A \cap \overline{A} \overset{*}{=} A$

tức là

$A \cup \overline{A} = \overline{A}$, $A \cap \overline{A} = A$.

Như thế, trong trường hợp cân bằng tĩnh, chúng ta lại chỉ thu được A và $\overline{A}$, nghĩa là không thu được một cái mới ngoài A và $\overline{A}$. Nói cách khác, cân bằng tĩnh không có hiệu lực sản sinh ra cái mới. Vì thế để có thể hiểu được cơ chế Trời Đất sinh ra cái mới, chúng ta hãy quay sang trường hợp cân bằng động.

Cân bằng động

Cân bằng động xảy ra khi hai nhân tố (thế lực) không cân bằng nhau, nhưng lại tạo ra những tình huống nằm dao động xung quanh trạng thái cân bằng tĩnh.

Trong khuôn khổ của Lưỡng Nghi, đó là hai trường hợp Âm thịnh hay Dương thịnh:

$A \overset{*}{\supset} \overline{A}$, $\lambda(A) = -1$, khi Âm thịnh,

hay

$\overline{A} \overset{*}{\supset} A$, $\lambda(\overline{A}) = 1$, khi Dương thịnh

Nguyên lý phi bài trung trong khuôn khổ lý thuyết tập mờ vận dụng cho Lưỡng Nghi

Chúng ta đã nói nhiều về nguyên lý phi bài trung, đó là cốt lõi của nguyên lý Âm Dương của Triết học Đông phương. Vì thế cần có một minh hoạ cụ thể bằng công cụ toán tập mờ.

Ta phân hai trường hợp sau:

Giả sử:

$\lambda(A) = -1$, $\lambda(\overline{A}) = 1$, $A(a) = 0,4$; $A(d) = 0,2$; $\overline{A} \overset{*}{\supset} A$, Âm suy.

Trong trường hợp Âm suy này, ta được:

$\lambda(A \cap \overline{A}) = -1$, $(A \cap \overline{A})(a) = \text{Min}\{0,4; 0,6\} = 0,4$,

$(A \cap \overline{A})(d) = \text{Min}\{0,2; 0,8\} = 0,2$,

tức là:

$A \cap \overline{A} = A$, từ đó $A \cup \overline{A} = \overline{A}$,

nghĩa là không xuất hiện một cái mới thứ ba nào, khác A và $\overline{A}$!

Bây giờ ta giả sử Âm thịnh

$A(a) = 0,8; A(d) = 0,7.$

Thế thì ta được ngay:

$(A \cap \overline{A})(a) = 0,2; (A \cap \overline{A})(d) = 0,3$

tức là

$(A \cap \overline{A}) \neq \overline{A} \neq Ø, (A \cap \overline{A}) \neq A \neq Ø.$

Kết quả thu được là một hiện tượng khác A và $\overline{A}$. Ta được một cái thứ ba nào đó !

Ta cũng thu được một cái thứ ba, khi Âm thịnh một phần, chẳng hạn trong trường hợp cụ thể sau:

$A(a) = 0,6; A(d) = 0,3; \overline{A}(a) = 0,4; \overline{A}(d) = 0,7.$

$(A \cap \overline{A})(a) = 0,4; (A \cap \overline{A})(d) = 0,3; (A \cap \overline{A}) \neq Ø; (A \cap \overline{A}) \neq A, \overline{A}$.

$(A \cup \overline{A})(a) = 0,6; (A \cup \overline{A})(d) = 0,7; (A \cup \overline{A}) \neq Ø; (A \cup \overline{A}) \neq A, \overline{A}$.

Ngày trước Lão Tử nói:

một sinh hai
hai sinh ba
ba sinh vạn vật

Bây giờ ta đã chứng tỏ được cái thứ ba đó bằng toán tập mờ khi Âm thịnh !

Với lập luận và cách hình thức hóa tương tự đối với các cấu trúc Tứ Tượng, Ngũ Hành, Bát Quái, các hệ quẻ… giáo sư Nguyễn Hoàng Phương đã đưa đến cho chúng ta một cái nhìn mới mẻ, lý thú về một vấn đề cổ gắn liền với đời sống của chúng ta. Do giới hạn của đề tài nên chúng tôi xin tạm ngưng phần giới thiệu về lý thuyết Kinh dịch ở đây và nhường lại cho các bạn có hứng thú nghiên cứu tiếp tục và xin chuyển qua phần nghiên cứu về ứng dụng.

1.4 Ứng dụng của Kinh dịch trong đời sống

Kinh dịch từ lâu đã có quan hệ mật thiết đối với đời sống vật chất và tinh thần đối với nhân dân các nước phương Đông trong đó có Việt Nam chúng ta. Kinh dịch là những tri thức cốt lõi, để rồi khi kết hợp với các tri thức khác sẽ giải quyết được một bài toán cụ thể trong đời sống như:

- Khi kết hợp với các tri thức về nhân thể học, thời châm sẽ cho ra đời phương pháp chữa bệnh theo Đông Y, châm cứu…
- Khi kết hợp với học thuyết Phong Thủy sẽ giúp cho ta chọn được các vùng đất thích hợp cho việc xây dựng.
- Khi kết hợp với các học thuyết dự đoán (Độn Giáp, Thái Ất Thần Kinh, Tứ Trụ…) sẽ cho ra các phương pháp dự đoán vận mệnh của một con người hay cho cả cộng đồng người…

Trong phần thực hiện dưới đây, chúng tôi xin áp dụng Kinh dịch trong việc dự đoán vận mệnh, mà cụ thể hơn là các vấn đề về hôn nhân, kết hợp với học thuyết Tứ trụ.

Chương 2: Học thuyết Tứ Trụ

2.1 Thế giới thông tin và con người:

Vũ trụ bao gồm trời đất và vạn vật hòa quyện với nhau, cảm ứng lẫn nhau, do đó con người không thể tách rời khỏi hệ thống mà luôn luôn bị tác động ảnh hưởng từ lúc sinh ra cho đến lúc chết đi.

Sự biến mất của người hay sự vật và sự biến đổi của thiên tượng là do cảm ứng của âm dương ngũ hành mà ra. "Mệnh" của con người thể hiện sự biến đổi, dịch chuyển của vũ trụ. Những thông tin này được biểu diễn bởi âm dương, ngũ hành. Người ta dùng can chi để biểu thị nó. Nhưng sự biến đổi đó là liên tục không ngừng, do đó trong các trạng thái biến đổi khác nhau của vũ trụ, mệnh sẽ biểu hiện thành các "vận" khác nhau.

Do đó thế giới này là thế giới thông tin mà âm dương ngũ hành là biểu tượng của các thông tin đó.

Con người sống trong vũ trụ, nếu biết được vận mạng của mình, thuận theo quy luật, làm chủ được sinh mệnh thì tốt, nghịch lại thì trái với quy luật tự nhiên tất sẽ gặp họa.

Vì sao ta dự đoán được mệnh vận căn cứ vào thời điểm sinh ra?

Khí âm dương, ngũ hành mà thời khắc sinh ra thụ đắc được chính là mức độ phân lượng và tính chất: kim, mộc, thủy, hỏa, thổ được biểu thị bằng các can chi. Can chi của năm, tháng, ngày, giờ sinh đại biểu cho âm dương ngũ hành để tượng trưng mô hình và phản ánh kết cấu nội bộ trong cơ thể.

Cơ thể có cân bằng được với môi trường xung quanh hay không sẽ là căn cứ để giải thích vì sao các tạng phủ trong một người ra đời cùng một lúc, nhưng có

cái bị bệnh, còn những cái khác lại không, đồng thời cũng giải đáp được nguyên nhân vì sao mọi người cùng sống trên trái đất nhưng số phận lại khác nhau.

Ngũ hành đầy đủ, sinh khắc vượng suy hợp lý, đó là mệnh tốt. Ngũ hành lệch nhiều, nếu trong mệnh các vận có sự nhất trí với tuần hoàn biến hóa của vũ trụ, cũng được xem là mệnh tốt. Nếu ngũ hành tổ hợp xấu nhiều hơn tổ hợp tốt, mất cân bằng nhiều lại ngược với khí tuần hoàn của vũ trụ, gọi là mệnh xấu.

Đó là các yếu tố Tiên thiên, trong Hậu thiên, biết mệnh là để hiểu rõ và cải thiện hoàn cảnh của mình trong sự biến đổi của vũ trụ, để tìm được sự yên ổn trong thế giới này. Duy trì sự cân bằng của âm dương, ngũ hành, thuận với quy luật tự nhiên là xu thế cần hướng tới.

2.2 Địa Chi- Tọa độ thời gian

Chiều thời gian trong không gian Kinh Dịch chuyển dịch kế tiếp nhau theo một chu kỳ khép kín - là vòng tròn. Mỗi một thời điểm dịch chuyển là một tọa độ thời gian.

Thời gian của Dịch Học chia thành 4 cấp độ: năm, tháng, ngày, giờ có cùng một loại tên gọi. Ví dụ có năm Giáp Tí, tháng Giáp Tí, ngày Giáp Tí, giờ Giáp Tí.

Mỗi một khái niệm thời gian có thêm 2 khái niệm chỉ tính chất là Tính Âm/Dương và tính Ngũ Hành. Tính Ngũ Hành chỉ có 5 tính là: Kim, Mộc, Thủy, Hỏa, Thổ.

Các nhà dịch học cổ cũng phối 12 chi với hướng và mùa.

Chi	Tí	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi	Thân	Dậu	Tuất	Hợi
Hành	Thủy	Thổ	Mộc	Mộc	Thổ	Hỏa	Hỏa	Thổ	Kim	Kim	Thổ	Thủy
Hướng	Bắc	Trung	Đông	Đông	Trung	Nam	Nam	Trung	Tây	Tây	Trung	Bắc
Mùa	Đông	Đông	Xuân	Xuân	Xuân	Hạ	Hạ	Hạ	Thu	Thu	Thu	Đông
Nghi	+	-	+	-	+	-	+	-	+	-	+	-

Các nhà dịch học cổ xưa khi tạo ra khái niệm 12 Địa Chi (12 Tọa độ thời gian) không phải căn cứ vào 12 con vật như chuột, trâu, hổ, lợn… để đặt.

Chi	Tí	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi	Thân	Dậu	Tuất	Hợi
Tháng(âl)	11	12	1	2	3	4	5	6	7	8	9	10
Giờ	23-1	1-3	3-5	5-7	7-9	9-11	11-13	13-15	15-17	17-19	19-21	21-23

- Lục hợp: do Mão mộc khắc Tuất thổ; Hợi thủy sinh Dần mộc… Lục hợp là:
    - Tí hợp Sửu hóa Thổ (hợp khắc)
    - Dần hợp Hợi hóa Mộc (hợp sinh)
    - Mão hợp Tuất hóa Hỏa (hợp khắc)
    - Thìn hợp Dậu hóa Kim (hợp sinh)
    - Tỵ hợp Thân hóa Thủy (hợp khắc)
    - Ngọ hợp Mùi hóa Thổ (hợp sinh)

Trong Tứ Trụ có lục hợp là tốt. Trong hợp có khắc và trong hợp có sinh. Hợp khắc là trước tốt sau xấu, hợp sinh là càng ngày càng tốt hơn.

- Tương xung của 12 chi: tương xung là đối xung theo phương hướng. Mão mộc Đông xung với Dậu kim Tây, Ngọ hỏa Nam xung với Tí thủy Bắc… Do đó:
    - Tí Ngọ tương xung
    - Sửu Mùi tương xung
    - Dần Thân tương xung
    - Mão Dậu tương xung

- Thìn Tuất tương xung
- Tỵ Hợi tương xung

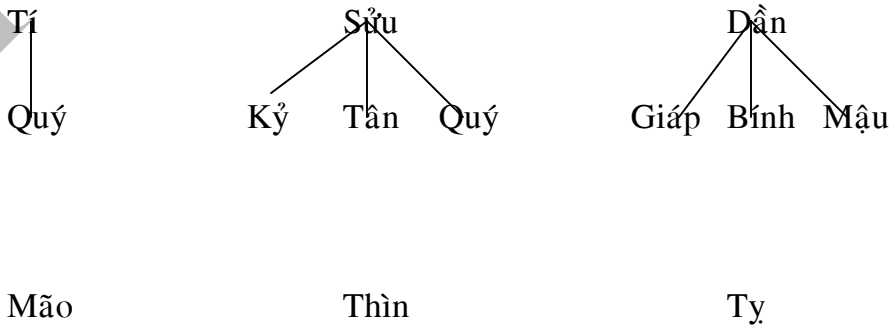
Tứ Trụ có xung có tốt có xấu. Nếu bị xung thần phúc thì hung, xung mất thần khắc là cát.

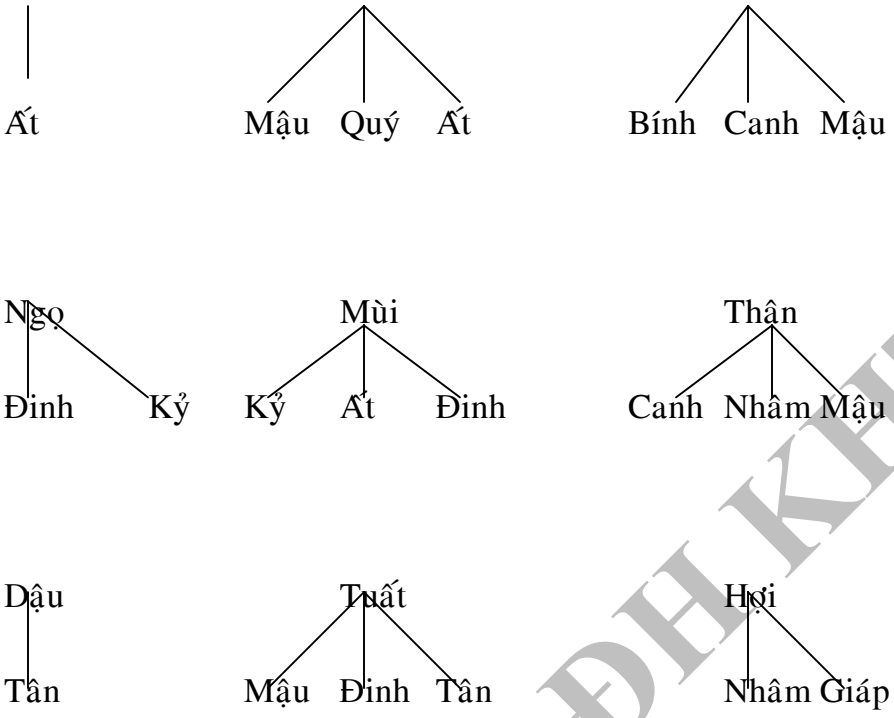
- Tương hại của 12 chi: Tí hợp Sửu, Mùi đến thì xung tan nên Tí Mùi tương hại; Dần hợp Hợi, Tỵ đến xung tan nên Dần Tỵ tương hại… Do đó:
  - Tí Mùi tương hại
  - Sửu Ngo tương hại
  - Dần Tỵ tương hại
  - Mão Thìn tương hại
  - Thân Hợi tương hại
  - Dậu Tuất tương hại

  Tứ Trụ gặp tương hại là bất lợi, còn phải xem có bị chế ngự không.

- Tàng độn của 12 chi: Dịch học căn cứ theo Trường sinh, lâm quan, mộ kho của ngũ hành Thiên can, định ra những địa chi tàng chứa.

Ví dụ: Tí là lâm quan của Quý Thủy nên Quý tàng chứa trong Tí





2.3 Thiên Can- Tọa độ không gian

Các nhà Dịch học cổ đưa ra 10 khái niệm chỉ 10 vị trí không gian gọi là Thập Can, còn chúng ta gọi là Tọa Độ Không Gian. Đó là:

Can	Giáp	Ất	Bính	Đinh	Mậu	Kỷ	Canh	Tân	Nhâm	Quý
Hành	Mộc	Mộc	Hỏa	Hỏa	Thổ	Thổ	Kim	Kim	Thủy	Thủy
Nghi	+	-	+	-	+	-	+	-	+	-
Mùa	Xuân	Xuân	Hạ	Hạ	Hạ trưởng	Hạ trưởng	Thu	Thu	Đông	Đông

- Mười can hóa hợp

Năm Giáp hay năm Kỷ lấy Bính Dần làm tháng giêng. Bính hỏa, hỏa sinh thổ nên Giáp hợp Kỷ hóa thổ.

Tương tự :

Ất hợp Canh hóa kim

Bính hợp Tân hóa thủy

Đinh hợp Nhâm hóa mộc

Mậu hợp Quý hóa hỏa

- Sinh vượng tử tuyệt của 10 can

Người xưa quan niệm có sinh thì có diệt theo một vòng tuần hoàn rồi lại bắt đầu một chu kỳ mới. Đối với con người, tùy vào thời điểm xuất hiện trong không gian mà có sự sinh (phát triển), sự thành đạt mọi mặt (đế vượng), sự suy thoái, bế tắc (mộ) và đến mức cùng ngược với sự phát triển (tuyệt).

Chu kỳ vận động đó như sau: Trường sinh → Mộc dục → Quan đới → Lâm quan → Đế vượng → Suy → Bệnh → Tử → Mộ → Tuyệt → Thai → Dưỡng → Trường sinh …

Trong Trung Y, người thầy thuốc có thể căn cứ vào vòng trường sinh để biết thời gian nào với người sinh ở TĐKG tương ứng bệnh sẽ phát triển hay giảm đi, phục hồi.

Nghi	+	+	+	+	+	-	-	-	-	-
Can	Giáp Mộc	Bính Hỏa	Mậu Thổ	Canh Kim	Nhâm Thủy	Ất Thủy	Đinh Hỏa	Kỷ Thổ	Tân Kim	Qúy Thủy
Sinh	Hợi	Dần	Dần	Tí	Thân	Ngọ	Dậu	Dậu	Tí	Mão
Vượng	Mão	Ngọ	Ngọ	Dậu	Tí	Dần	Tỵ	Tỵ	Thân	Hợi
Mộ	Mùi	Tuất	Tuất	Sửu	Thìn	Tuất	Sửu	Sửu	Thìn	Mùi
Tuyệt	Thân	Hợi	Hợi	Dần	Tỵ	Dậu	Tí	Tí	Mão	Ngọ

- Thông biến của Thiên Can:

Mỗi người sống trên đời này vì sinh vào thời khắc khác nhau trong vũ trụ nên được hưởng khí âm dương bẩm sinh trong đục, vượng suy khác nhau. Tứ trụ lấy sự vượng suy của can ngày trong tứ trụ làm trung tâm, còn những can chi khác sẽ sinh khắc phù trợ hay hạn chế can chi ngày sinh để cấu tạo thành một hệ thống. Tổ hợp của can ngày sinh với các can chi khác trong tứ trụ là biểu tượng của âm

dương ngũ hành cấu tạo thành đặc điểm của một con người cụ thể. Phú quý, phúc họa của con người đều xuất phát từ can ngày sinh và thông qua nó để thể hiện trạng thái được chung kết lại của người đó trong vũ trụ. Do đó can ngày sinh được gọi là "nhật nguyên", "nhật chủ" hoặc "thân".

Quan hệ giữa tính ngũ hành của nhật chủ với tính ngũ hành của các can chi khác trong tứ trụ không ngoài: chính, thiên và sinh, khắc. Can ngày dương gặp các can dương khác trong tứ trụ là sự gặp gỡ đồng tính, là thiên; can ngày dương gặp các can âm khác là dị tính, là chính. Tương tự can ngày âm gặp các can âm khác là đồng tính, là thiên; can ngày âm gặp các can khác dương là dị tính, chính.

Can	Giáp	Ất	Bính	Đinh	Mậu	Kỷ	Canh	Tân	Nhâm	Quý
Giáp	Ngang vai	Kiếp tài	Thực thần	Thương quan	Thiên tài	Chính tài	Thất sát	Chính quan	Thiên ấn	Chính ấn
Ất	Kiếp tài	Ngang vai	Thương quan	Thực thần	Chính tài	Thiên tài	Chính quan	Thất sát	Chính ấn	Thiên ấn
Bính	Thiên ấn	Chính ấn	Ngang vai	Kiếp tài	Thực thần	Thương quan	Thiên tài	Chính tài	Thất sát	Chính quan
Đinh	Chính ấn	Thiên ấn	Kiếp tài	Ngang vai	Thương quan	Thực thần	Chính tài	Thiên tài	Chính quan	Thất sát
Mậu	Thất sát	Chính quan	Thiên ấn	Chính ấn	Ngang vai	Kiếp tài	Thực thần	Thương quan	Thiên tài	Chính tài
Kỷ	Chính quan	Thất sát	Chính ấn	Thiên ấn	Kiếp tài	Ngang vai	Thương quan	Thực thần	Chính tài	Thiên tài
Canh	Thiên tài	Chính tài	Thất sát	Chính quan	Thiên ấn	Chính ấn	Ngang vai	Kiếp tài	Thực thần	Thương quan
Tân	Chính tài	Thiên tài	Chính quan	Thất sát	Chính ấn	Thiên ấn	Kiếp tài	Ngang vai	Thương quan	Thực thần
Nhâm	Thực thần	Thương quan	Thiên tài	Chính tài	Thất sát	Chính quan	Thiên ấn	Chính ấn	Ngang vai	Kiếp tài
Quý	Thương quan	Thực thần	Chính tài	Thiên tài	Chính quan	Thất sát	Chính ấn	Thiên ấn	Kiếp tài	Ngang vai

Ấn = Chính Ấn

Kiêu = Thiên ấn

Thương = Thương quan

Thực = Thực thần

Quan = Chính quan

Sát = Thiên quan

Kiếp = Kiếp tài

Tỉ = Ngang vai

Tài= Chính tài

2.4 Can chi phối hợp

Toạ độ Thời gian và Toạ độ Không gian không tách biệt mà liên kết nhau để xác định một vị trí trong Không Gian. Do vậy bao giờ ta cũng gọi cả Can Chi (TĐKG+TĐTG)

Sự liên kết giữa một Can và một Chi theo quy thức Dương với Dương, Âm với Âm. Chỉ có Can dương phối hợp với chi dương, can âm phối hợp với chi âm, tạo ra các vị trí Không Thời Gian. Như vậy thời gian theo lịch can chi là thời gian lặp, vận động theo đường tròn, thuận theo quy luật của Dịch lý đề ra(1 cặp thuần dương hay thuần âm) như năm Giáp Tí, Quý Mùi… chứ không bao giờ có Đinh Thìn, Giáp Dậu… Sự liên kết này tạo ra một cơ chất ngũ hành nào đó trong không gian Kinh Dịch.

Một đối tượng khi sinh ra vào một tọa độ không-thời gian sẽ phản ánh quy luật vận động của chính nó. Ví dụ người sinh năm Giáp Tí có cơ chất thuộc Kim, nếu gặp tháng hay ngày giờ cũng mang tính Kim thì Kim vượng. Người Kim có chí khí, có nghĩa khí. Nếu tháng hay ngày có Hỏa thì Kim bị khắc. Vượng về một hành nào đó thì tình huống bất lợi hay có lợi đều tăng.

Tứ Trụ luôn lấy sự cân bằng làm tốt, lấy vượng hoặc nhược làm xấu.

Chi-Can	Giáp Ất	Bính Đinh	Mậu Kỷ	Canh Tân	Nhâm Quý
Tí Sửu Ngọ Mùi	Kim	Thủy	Hỏa	Thổ	Mộc
Dần Mão Thân Dậu	Thủy	Hỏa	Thổ	Mộc	Kim
Thìn Tỵ Tuất Hợi	Hỏa	Thổ	Mộc	Kim	Thủy

Mỗi người đều có một khoảnh khắc độc nhất xuất hiện trong Vũ trụ, khoảnh khắc này đã cá biệt hóa số phận của từng người. Để xác định vị trí xuất hiện đó, các nhà Dịch học đã hình thành công cụ đo không gian thời gian trong vũ trụ mênh mông: đó là Lịch Can Chi có kèm theo Lịch Tiết Khí.

2.5 Phương pháp dự đoán hôn nhân theo Tứ Trụ:

Tứ Trụ là dùng thiên can, địa chi của năm, tháng, ngày, giờ sinh để biểu thị thông tin của một người, vận dụng các quy luật của âm dương ngũ hành tìm ra sinh mệnh con người do từ trường quả đất, lực hấp dẫn và các loại trường cảm ứng khác gây nên.

Từ các quy luật cơ bản của âm dương, ngũ hành (như sinh khắc), Tứ Trụ đã cụ thể thành các luật hệ quả áp dụng trong quá trình dự đoán.

Phương pháp Tứ Trụ dự đoán về hôn nhân, về vận trình cả cuộc đời (các đại vận), về lưu niên, lục thân, của cải, tính cách, bệnh tật tai họa… Khóa luận này xin bàn về phương pháp dự đoán theo Tứ Trụ trong lĩnh vực hôn nhân.

Trình tự dự đoán hôn nhân theo Tứ Trụ:

- Lấy ngày giờ sinh thật chính xác.
- Sắp xếp Tứ trụ chính xác theo thứ tự năm-tháng-ngày-giờ. (Nếu cần đổi ngày giờ sinh ra bát tự).
- Tra 10 thần thấu rõ và tàng độn.

- Tính các cát thần, hung sát nào có xuất hiện trên các trụ.
- Áp dụng các luật về hôn nhân để suy ra các kết luận.

Phương pháp dự đoán theo Tứ Trụ lấy can ngày làm chủ, chú trọng sự cân bằng trong Tứ Trụ. Do đó ngoài quy luật thuần sinh và thuần khắc, còn xét thêm quy luật phản sinh và phản khắc.

Chương 3: Hệ chuyên gia

Xây dựng một chương trình dự đoán tương đương với việc đưa các tri thức về dự đoán vào trong máy tính. Việc này bao gồm các bước:

- Biểu diễn các tri thức dự đoán vào trong máy tính
- Sử dụng các tri thức dự đoán.

Chúng ta có thể viết một chương trình máy tính bình thường với các thao tác dòng lệnh để thực hiện các chức năng trên, nhưng hạn chế của một chương trình bình thường là khó thay đổi, bổ sung các tri thức mới. Vì vậy ở đây sẽ xây dựng một hệ chuyên gia dự đoán dựa trên nền tảng là một hệ cơ sở tri thức, chính xác hơn là một cơ sở tri thức dựa vào luật.

3.1 Các khái niệm về cơ sở tri thức:

Chúng ta đã khá quen thuộc với các chương trình máy tính nên ở đây xin nêu ra một điểm khác biệt cơ bản của một chương trình và một hệ cơ sở tri thức, đó là:

- Trong một chương trình truyền thống, cách xử lí hay hành vi của chương trình đã được ấn định sẵn thông qua các dòng lệnh của chương trình dựa trên một thuật giải được ấn định sẵn.

- Trong một hệ CSTT, có hai chức năng tách biệt nhau, trường hợp đơn giản có hai khối, khối tri thức hay còn gọi là cơ sở tri thức, và khối suy diễn hay còn gọi là động cơ suy diễn. Việc tách biệt giữa tri thức khỏi các cơ chế điều khiển giúp ta dễ dàng thêm vào các tri thức mới trong quá trình phát triển của chương trình. Đây là điểm tương tự của động cơ suy diễn trong một hệ CSTT và não bộ con người (điều khiển xử lý),

là không đổi cho dù hành vi của cá nhân có thay đổi theo kinh nghiệm và kiến thức nhận được.

Từ điểm khác biệt trên, ta có thể rút ra một đặc điểm quan trọng của từng loại chương trình trong trường hợp cụ thể này rằng:

- Trong chương trình truyền thống, nếu muốn thay đổi hay bổ sung các tri thức dự đoán mới (có thể xảy ra theo phân tích ở trên), chúng ta cần phải xem xét cấu trúc và thay đổi lại mã lệnh của chương trình.
- Trong khi đó, trong một hệ cơ sở tri thức, ta chỉ cần thay đổi trong khối tri thức (bên ngoài chương trình) mà không cần thay đổi khối suy diễn. Mặt khác hệ cơ sở tri thức còn có tính tái sử dụng đối với các bài toán tương tự (ta có thể xây dựng một khối tri thức cho từng bài toán, và thay đổi đầu vào và đầu ra cho khối suy diễn).

Các hệ CSTT đều có động cơ suy diễn để tiến hành các suy diễn nhằm tạo ra các tri thức mới từ các sự kiện, tri thức cung cấp từ ngoài vào và tri thức có sẵn trong hệ CSTT. Động cơ suy diễn thay đổi theo độ phức tạp của CSTT. Hai kiểu suy diễn chính trong động cơ suy diễn là suy diễn tiến và suy diễn lùi.

Suy diễn tiến: các hệ là việc theo sự điều khiển của dữ liệu (data driven), dựa vào các thông tin có sẵn (các sự kiện cho trước) và tạo sinh ra các sự kiện mới được suy diễn. Do vậy không thể đoán trước được kết quả. Cách tiếp cận này được sử dụng trong các bài toán diễn dịch với mong mỏi của người sử dụng là hệ CSTT sẽ cung cấp các sự kiện mới (kiểu suy diễn này có thể được áp dụng trong việc dự đoán: từ các sự kiện ban đầu về ngày sinh, hệ CSTT sẽ đưa ra các sự kiện dự đoán).

Suy diễn lùi: điều khiển theo mục tiêu nhằm hướng đến các kết luận đã có và đi tìm các dẫn chứng để kiểm định tính đúng đắn của kết luận đó.

Cơ sở tri thức có nhiều dạng khác nhau:

- đối tượng - thuộc tính - giá trị
- thuộc tính- luật dẫn
- mạng ngữ nghĩa
- frame.

Các hệ chuyên gia là một loại hệ CSTT được thiết kế cho một lĩnh vực ứng dụng cụ thể.

Trong trường hợp này chúng tôi chọn xây dựng một hệ chuyên gia dựa trên luật dẫn nên phần dưới xin trình bày một số khái niệm cơ bản trong về hệ dựa trên luật.

3.2 Hệ chuyên gia dựa trên luật

3.2.1 Luật và sự kiện

Một hệ dựa trên luật là một hệ cơ sở tri thức mà cơ sở tri thức được biểu diễn dưới dạng của một tập (hay nhiều tập)luật.

Luật là cấu trúc tri thức dùng để liên kết thông tin đã biết với các thông tin khác giúp đưa ra các suy luận, kết luận từ các thông tin đã biết. Luật là một phương tiện súc tích, có ý nghĩa, không phức tạp và linh hoạt của việc biểu diễn tri thức. Trong hệ thống dựa trên luật, người ta thu thập các tri thức lĩnh vực trong một tập và lưu chúng trong cơ sở tri thức của hệ thống. Hệ thống dùng các luật này cùng với các thông tin trong bộ nhớ để giải bài toán. Việc xử lí các luật trong hệ thống dựa trên các luật được quản lí bằng một module gọi là động cơ suy diễn.

Kiểu đơn giản nhất của luật được gọi là luật sản xuất và có dạng:

If <điều kiện> then <kết quả>

Ví dụ:

If can là Giáp then hành can là Thổ và nghi can là Dương

If can là Ất then hành can là Thổ và nghi can là Âm

Sự kiện: để một luật có thể được thực hiện, và do đó một hệ thống dựa trên luật được sử dụng cho việc nào đó, hệ thống cần truy cập các sự kiện. Sự kiện là những phát biểu được giả định là đúng tại thời điểm sử dụng.

Sự kiện có thể:

- tra cứu từ một cơ sở dữ liệu.
- đã được lưu trữ trong bộ nhớ máy tính
- xác định từ các thiết bị gắn với máy tính
- có được bằng cách nhắc nhở người dùng nhập thông tin
- được dẫn xuất bằng cách áp dụng các luật từ các sự kiện khác.

Chúng ta có thể xem xét một số ví dụ về sự kiện được biểu diễn trong cơ sở tri thức về dự đoán như sau:

- Các sự kiện ban đầu: bao gồm các tri thức về âm dương, ngũ hành, các tri thức về nguyên thần, ... như:
  - "Thuỷ sinh Mộc"
  - "Mộc sinh Hoả"
  - ...
  - "Thuỷ khắc Hoả"
  - "Hoả khắc Kim"
  - …
- Các sự kiện về người dùng: được cung cấp cho động cơ trong mỗi lần dự đoán và sử dụng để suy diễn ra kết quả:

- "Trụ tháng của người nam có can là Giáp"
- "Trụ tháng của người nam có can là Tí"
- …
- "Trụ ngày của người nữ có can là Ất"
- "Trụ ngày của người nữ có chi là Sửu"
- …

- Các sự kiện được suy diễn để cho ra kết quả dự đoán:
  - "Trụ ngày nam nữ tương sinh"
  - "Trụ năm nam nữ tương sinh"
  - …

Biểu diễn luật là một dạng của ngôn ngữ lập trình hướng khai báo bởi vì các luật biểu diễn tri thức có thể được sử dụng bởi máy tính, mà không cần xác định khi nào và bằng cách nào áp dụng tri thức. Bởi thế thứ tự các luật trong một chương trình không quan trọng, và nó phải cho phép thêm những luật mới hoặc sửa chữa những cái có sẵn mà không lo ngại về tác dụng phụ.

3.2.2 Kiểm tra và thực hiện luật:

Cơ sở tri thức đưa ra các phát biểu luật mà không đề cập bằng cách nào các luật sẽ được áp dụng. Nhiệm vụ biểu diễn và áp dụng luật thuộc về động cơ suy diễn. Việc áp dụng của các luật có thể chia như sau:

- lựa chọn các luật để kiểm tra - là những luật sẵn sàng
- xác định những luật nào có thể áp dụng được - những cái này tạo nên tập đối lập
- lựa chọn một luật để thực hiện.

3.2.3 Giả thiết về thế giới đóng:

Nếu chúng ta không biết chắc rằng mệnh đề là đúng, thì trong nhiều hệ thống dựa trên luật mệnh đề được giả định là sai. Giả định này, được biết đến dưới

tên gọi giả định thế giới đóng, đơn giản hóa logic bằng cách cho rằng tất cả các mệnh đề hoặc đúng hoặc sai. Nếu giả định thế giới đóng không được thực hiện, thì một giá trị thứ ba, tên là CHƯA BIẾT, phải được đưa ra.

Trong cơ sở tri thức, chúng ta có các sự kiện về mối quan hệ giữa các hành. Đó là các mối quan hệ tương sinh, tương khắc… giữa 2 hành với nhau. Chúng ta có tất cả 5 hành như vậy sẽ có tất cả C^2_5 nhóm 2 hành với nhau. Mặt khác chúng ta chỉ có 5 quan hệ cho mỗi loại (tương sinh, tương khắc…). Do sử dụng giả thiết về thế giới đóng, chúng ta không cần phải khai báo cho các nhóm không quan hệ. Ví dụ, chúng ta có thể sử dụng các sự kiện sau mà không cần khai báo

"not Thuỷ sinh Hoả"

"not Thuỷ khắc Mộc"

…

Mặt khác, cũng do sử dụng giả thiết trên nên chúng ta cũng phải quan tâm đến thứ tự thực hiện của các luật (phải đảm bảo tính thứ tự cho các luật cần thiết) để tránh những kết quả không mong muốn.

3.2.4 Sử dụng biến số trong luật:

Trong thế giới thực, biến số có thể được sử dụng để làm cho các luật tổng quát hơn, bằng cách giảm số lượng luật cần thiết và giữ cho tập luật có thể kiểm soát được. Kiểu của luật thông thường cần có dạng như sau:

Với tất cả X, IF điều kiện về X THEN kết luận về X.

Ví dụ: nếu như trong cơ sở tri thức của ta có 8 trụ (4 cho nam và 4 cho nữ) thì ta sẽ có 8 luật để tính nghi và hành cho các trụ như sau:

"if trụ ngày của nam có can là Giáp then trụ ngày của nam có hành là Thổ và trụ ngày của nam có nghi là Dương"

…

"if trụ ngày của nữ có can là Giáp then trụ ngày của nam có hành là Thổ và trụ ngày của nam có nghi là Dương"

…

chúng ta sử dụng biểu diễn sau để có thể diễn đạt uyển chuyển và ngắn gọn hơn:

"if trụ ?X có can là Giáp then trụ ?X có hành là Thổ và trụ ?X có nghi là Dương"

Trong trường hợp các giá trị có thể của X được giới hạn, việc sử dụng của một biến số có ý nghĩa tiện lợi hơn là cần thiết. Khi mà các giá trị có thể của một biến số không thể đoán trước khi thực hiện, việc sử dụng biến số trở nên thiết yếu. Đây là trường hợp khi các giá trị là chưa biết và đang được tìm kiếm, có thể trong một cơ sở dữ liệu.

Chúng ta xem ví dụ sau:

"if trụ ngày của nữ có hành là ?X and trụ ngày của nam có hành là ?Y and ?X sinh ?Y then Trụ ngày tương sinh cho nhau"

Giả sử chúng ta có các sự kiện:

"trụ ngày của nữ có hành là Thuỷ"

"trụ ngày của nam có hành là Mộc"

kết hợp với sự kiện ban đầu

"Thuỷ sinh Mộc"

thì ta có thể rút ra được kết luận "trụ ngày tương sinh cho nhau"

Sự kết hợp giữa một giá trị cụ thể (Thuỷ) với một biến số (?X) được gọi là sự hợp giải, cũng tương tự cho giá trị Mộc và biến số ?Y. Thuật ngữ trên bắt nguồn từ cơ chế xử lí của máy tính. Ở đây chúng ta có 2 mẫu thông tin mâu thuẫn:

"trụ ngày của nữ có hành là Thuỷ"

"trụ ngày của nữ có hành là ?X"

Mâu thuẫn trên được xử lí bằng cách nhận ra rằng 1 trong 2 giá trị là tên biến số (bởi vì trong cú pháp của chúng ta nó được đặt sau dấu ?) và bằng cách thực hiện một lệnh gán:

X:= Thuỷ

3.2.5 Sử dụng biến dữ liệu:

Bên cạnh các khái niệm cơ bản trong hệ chuyên gia là luật và sự kiện, chúng tôi đưa thêm khái niệm mở rộng là biến dữ liệu. Biến dữ liệu là những đơn vị có thể chứa dữ liệu, cũng tương tự như khái niệm biến trong các ngôn ngữ lập trình trên máy tính. Cùng với biến dữ liệu đơn giản, chúng tôi còn sử dụng thêm các dữ liệu có cấu trúc (tương tự như biến cấu trúc) để lưu giữ các giá trị có quan hệ với nhau. Sự xuất hiện của biến dữ liệu dẫn đến sự bổ sung một số thao tác mới cũng như thay đổi của một số thao tác cũ như sau:

3.2.5.1 Cung cấp thông tin cho động cơ:

Ngoài việc cung cấp thông tin cho động cơ bằng cách cung cấp các sự kiện, ta có thể cung cấp thông tin cho động cơ bằng cách đặt giá trị cho các biến dữ liệu (đã khai báo trước). Ví dụ, thay vì khai báo sự kiện sau:

"trụ ngày của người nữ có can là Giáp"

ta gán giá trị "Giáp" cho biến "nữ.trụ_ngày.can"

3.2.5.2 Sử dụng giá trị của các biến dữ liệu:

Một điểm thuận tiện trong việc sử dụng biến dữ liệu là ta có thể biểu diễn những luật có sử dụng đến các giá trị biến đổi gọn gàng, súc tích và quen thuộc hơn so với nếu sử dụng cơ chế hợp giải. Để làm điều này, ta cung cấp một kí hiệu để truy xuất lấy giá trị từ biến dữ liệu và thay thế vào biểu thức cần sử dụng. Và ta có thể dùng kí hiệu này trong luật để thay thế cho việc sử dụng một luật cần hợp

giải. Chúng ta có thể thấy điều này bằng cách xem xét ví dụ trên (trong mục biến số):

"if trụ ngày của nữ có hành là ?X and trụ ngày của nam có hành là ?Y and ?X sinh ?Y then Trụ ngày tương sinh cho nhau"

luật trên được thay bằng luật dưới đây:

"if @nữ.trụ_ngày.hành sinh @nam.trụ_ngày.hành then Trụ ngày tương sinh cho nhau"

3.2.6 Sử dụng luật với biến lặp:

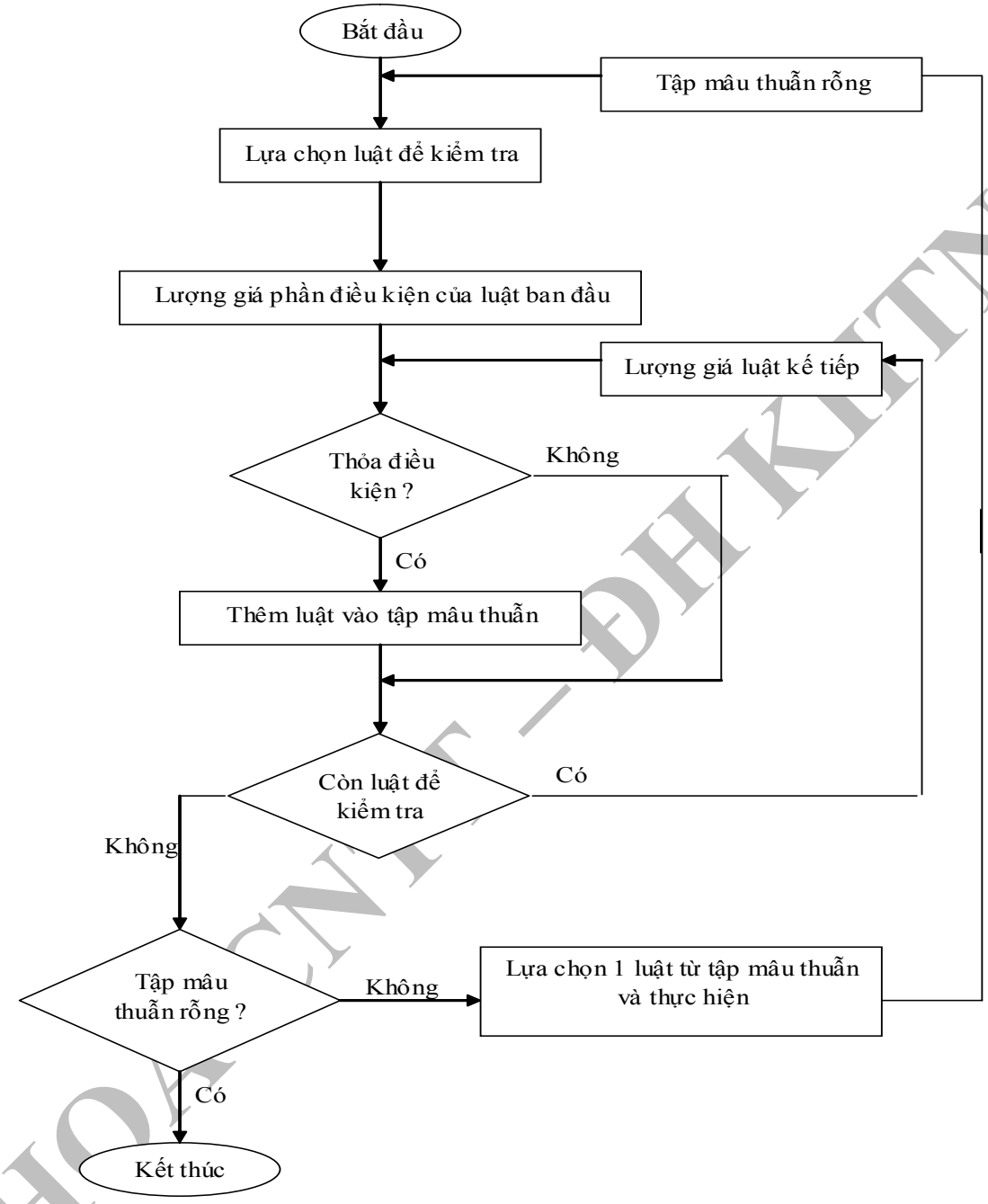
Trong phần biến số, ta cũng đã đề cập đến một vấn đề, đó là áp dụng một luật đối với nhiều đối tượng cùng loại. Trong biến dữ liệu, ta cũng gặp lại một vấn đề tương tự đối với biến dữ liệu. Để sử dụng một luật lặp đối với nhiều biến dữ liệu cùng loại ta có biểu diễn luật tương tự:

"if trụ ?X.can là Giáp then trụ ?X.hành=Thổ và trụ ?X.nghi= Dương"

3.2.7 Suy diễn tiến:

Suy diễn tiến là tên được đặt cho một chiến lược hướng dữ liệu, nghĩa là, những luật được chọn và áp dụng để đáp ứng với cơ sở sự kiện hiện tại. Cơ sở sự kiện bao gồm tất cả các sự kiện đã biết bởi hệ thống, được suy diễn từ luật hay được cung cấp trực tiếp.

Một lược đồ cơ bản của chu trình lựa chọn, kiểm tra và thực hiện của các luật được trình bày trong hình sau:



Hình 3 Mô hình suy diễn tiến

Chu trình của các sự kiện trình bày ở trên chỉ là một thể hiện của suy diễn tiến, và có thể thay đổi. Những điểm chính cần lưu ý trong lược đồ được trình bày là:

- luật được kiểm tra và thực hiện trên cơ sở của các sự kiện hiện tại, độc lập với các đích đã định trước;
- tập hợp các luật sẵn sàng cho kiểm tra có thể bao gồm tất cả hay là một tập con;
- các luật đã sẵn sàng, mà điều kiện được thoả mãn tạo thành tập mâu thuẫn, và phương pháp lựa chọn một luật từ tập mâu thuẫn được gọi là phân giải mâu thuẫn
- mặc dù có nhiều luật trong tập mâu thuẫn, chỉ một luật là được thực hiện trong một chu kỳ. (Điều này bởi vì một khi một luật được thực hiện, các suy luận được lưu trữ có những thay đổi tiềm tàng, và nó không thể đảm bảo rằng các luật khác trong tập mâu thuẫn vẫn thoả mãn điều kiện)

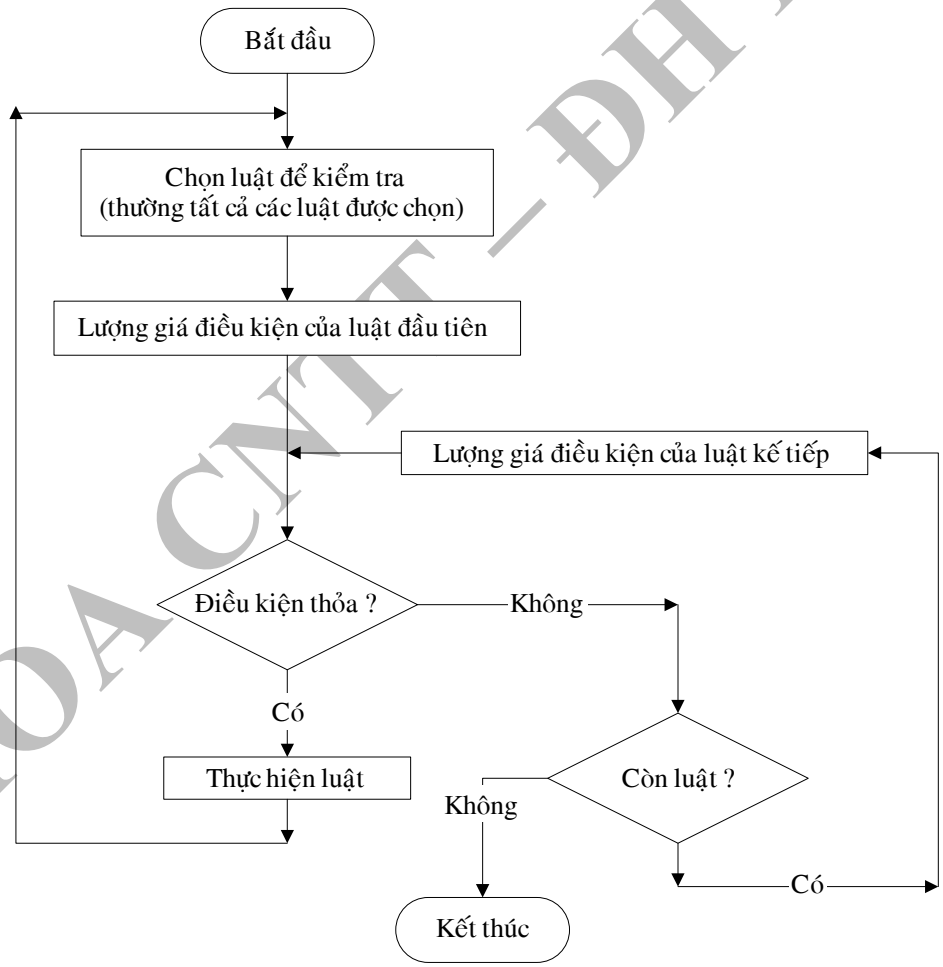
3.2.7.1 Thể hiện đơn trị và thể hiện đa trị trong biến số

Như đã đề cập trước ở trên, biến số trong lưu đồ cơ bản cho suy diễn tiến là được chấp nhận. Khi các biến số được sử dụng trong các luật, những kết luận có thể thực thi chỉ sử dụng tập thể hiện đầu tiên được tìm thấy- đây là thể hiện đơn. Thay vào đó, các kết luận có thể được thực thi lặp đi lặp lại sử dụng tất cả thể hiện- đây là thể hiện đa. Với chương trình hệ chuyên gia này, chúng tôi sử dụng phương thức đa trị cho biến số trong các luật.

3.2.7.2 Phân giải mâu thuẫn

Phân giải mâu thuẫn là phương pháp lựa chọn một luật để thực hiện từ những luật có thể thực hiện, tập mâu thuẫn. Các giải pháp bao gồm vào trước ra trước; sử dụng độ ưu tiên cho các luật; sử dụng các siêu luật (luật về các luật).

Vào trước, ra trước: trong lược đồ ở Hình 3 Mô hình suy diễn tiến, tập mâu thuẫn hoàn toàn được tìm ra trước khi lựa chọn một luật để thực hiện. Bởi vì chỉ một luật từ tập mâu thuẫn có thể thực sự thực hiện trong một chu kỳ, thời gian để lượng giá phần điều kiện của các luật khác bị lãng phí. Một chiến lược để vượt qua điểm không hiệu quả này là thực hiện ngay lập tức luật đầu tiên được tìm thấy mà đủ tiêu chuẩn cho tập mâu thuẫn. Trong lược đồ này, tập mâu thuẫn hoàn toàn không được tạo thành, và thứ tự mà các luật được lựa chọn để kiểm tra xác định sự phân giải mâu thuẫn. Thông thường thứ tự kiểm tra luật là thứ tự xuất hiện của luật trong cơ sở tri thức.



Hình 4: Suy diễn tiến với phân giải mâu thuẫn "vào trước, làm trước"

Siêu luật (luật về các luật): là những luật không liên quan đến bài toán mà liên quan đến cách thức sử dụng của các luật khác, một ví dụ sử dụng siêu luật là dùng để xác định thứ tự thực hiện của các nhóm luật:

refer Tien_de to Dieu_kien

luật trên sẽ bảo đảm cho các luật về Tiền đề được thực hiện trước các luật về Điều kiện.

Để thực hiện suy diễn với sự hướng dẫn của siêu luật, ta bổ sung cho mỗi luật một chỉ số thực hiện và gán giá trị cho chúng dựa vào các siêu luật được khai báo.

```
Thuật giải: gán chỉ số cho các luật dựa trên siêu luật
Dữ liệu vào: bộ luật L, bộ siêu luật M
Dữ liệu ra: chỉ số ind của mỗi luật
Với mỗi luật l ∈ L
      ind(l):= -1
      f:= false
      Với mỗi siêu luật m ∈ M
            Nếu l là luật sau trong m thì
                  f:= true
      Nếu f= false
      ind(l)= 0
f:= true
Lặp khi f= true
      f:= false
      Với mỗi siêu luật m ∈ M
            i:= ind(luật_trước(m))
            Nếu I <> -1 thì
                  Nếu i+ 1 > ind(luật_sau(m)) thì
                        ind(luật_sau(m)):= i+1
                        f:= true
Cuối lặp
```

Sau khi được gán chỉ số thì các luật sẽ được thực hiện lần lượt theo chỉ số từ thấp đến cao.

Chương 4: Khai thác dữ liệu

Ở phần trên chúng ta đã khảo sát và tìm hiểu cách đưa các tri thức từ bên ngoài vào bên trong máy tính và sử dụng các tri thức đó. Các tri thức dự đoán có được là do người xưa đã dựa trên các tiền đề về Âm Dương, Ngũ Hành cộng với việc quan sát thống kê các hiện tượng xảy ra trong cuộc sống mà có được. Vì thế, trong phần này, chúng ta sẽ tiếp cận đến một vấn đề mới trong việc tìm hiểu về Kinh dịch: đó là khám phá thêm các tri thức dự đoán mới và bổ sung chúng vào cơ sở tri thức từ các dữ liệu quan sát được trên người dùng.

Bài toán đặt ra ở đây là: với một cơ sở dữ liệu về người dùng mà ta thu thập được, trong đó dữ liệu của mỗi người dùng là một tập hợp các sự kiện nổi bật đã xảy ra đối với họ, ta tìm cách rút ra được thêm các tri thức dự đoán mới để có thể áp dụng cho những người mới sau này. Cơ sở cho việc khám phá tri thức này là việc tìm ra một quan hệ xác định giữa các tri thức điều kiện trong cơ sở tri thức (được rút ra từ các luật tiền đề về Âm dương Ngũ hành dựa trên các thông tin về ngày tháng năm sinh của một người) và các dữ kiện thu thập được.

Việc khám phá các tri thức có thể đi theo nhiều hướng tiếp cận khác nhau:

- Tiếp cận theo hướng phân lớp: mỗi dữ kiện quan sát được trở thành các thuộc tính phân lớp, phân chia cơ sở dữ liệu thành những lớp tùy theo giá trị của dữ kiện (đơn giản nhất là phân chia cơ sở dữ liệu thành 2 lớp: một lớp có và một lớp không có dữ kiện). Sau đó chúng ta tiến hành khám phá các luật phân lớp và từ đó rút ra được tri thức. Các thuật giải thông dụng cho phân lớp bao gồm: cây định danh và học theo quy nạp (ILA).

- Tiếp cận theo hướng luật kết hợp: áp dụng toàn bộ dữ liệu, chúng ta sử dụng độ phổ biến và độ tin cậy để xác định các luật kết hợp X → Y, trong đó X là các điều kiện và Y là các dữ kiện quan sát được.

4.1 Cây định danh

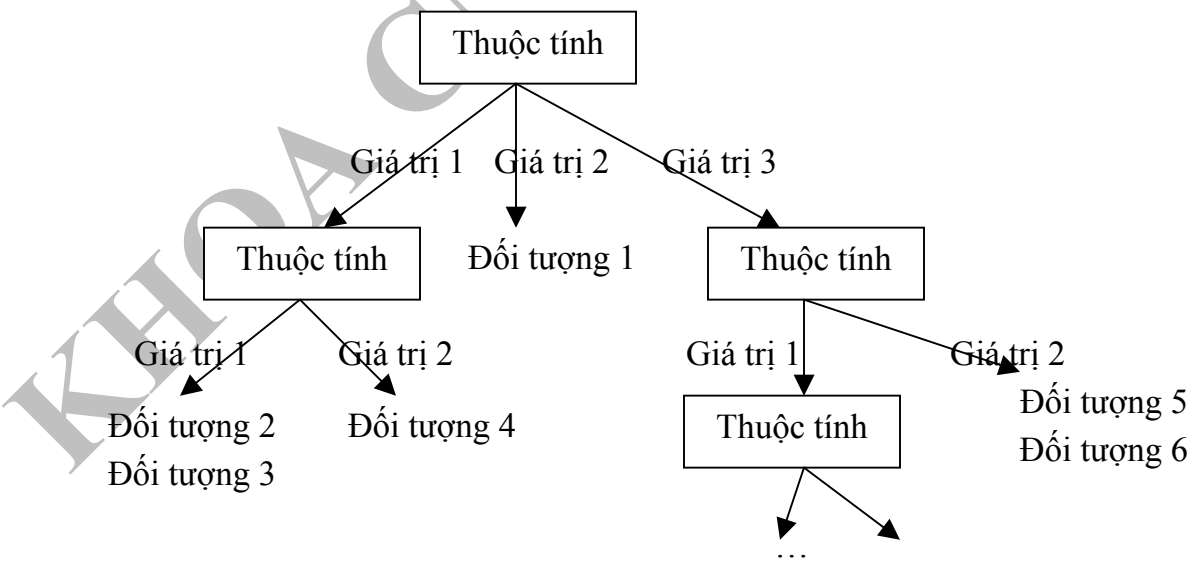
Cây định danh là một công cụ khá phổ biến trong nhiều dạng ứng dụng, với cơ chế rút trích các luật nhân quả xác định các mẫu dữ liệu.

Một cách phù hợp cho phép thực hiện các thủ tục thử nghiệm các thuộc tính là sắp xếp các thử nghiệm trên cây định danh. Do cây định danh thuộc loại cây quyết định, đặc tả của nó như đặc tả cây quyết định.

Định nghĩa: Cây định danh (Identification tree)

Cây định danh có thể hiện như cây quyết định, trong đó mỗi tập các kết luận được thiết lập ngầm định bởi một danh sách đã biết.

Mỗi đối tượng đưa vào định danh đi xuống theo một nhánh cây, tùy theo giá trị thuộc tính của nó.



Phát biểu Occam dùng cho các cây định danh:

Thế giới vốn đơn giản. Do vậy cây định danh gồm các mẫu là cái thích hợp nhất để định danh các đối tượng chưa biết một cách chính xác.

Phân loại đối tượng theo các thuộc tính

Nếu như ta tìm kiếm cây định danh nhỏ nhất khi cần có rất nhiều thử nghiệm thì không thực tế. Chính vì vậy mà cũng nên dừng lại ở thủ tục xây dựng những cây định danh nhỏ, dù rằng nó không phải là nhỏ nhất. Người ta chọn thử nghiệm cho phép chia cơ sở dữ liệu các mẫu thành các tập con. Trong đó nhiều mẫu cùng chung một loại. Đối với mỗi tập có nhiều loại mẫu, dùng thử nghiệm khác để chia các đối tượng không đồng nhất thành các tập chỉ gồm các đối tượng thuần nhất.

Độ lộn xộn của tập hợp

Đối với cơ sở dữ liệu thực, không phải bất kỳ thử nghiệm nào cũng cho ra tập đồng nhất. Với cơ sở dữ liệu này người ta cần đo mức độ lộn xộn của dữ liệu, hay độ không đồng nhất trong các tập con được sinh ra. Công thức đo lý thuyết thông tin về độ lộn xộn trung bình:

$$\sum_b \frac{n_b}{n_t} \sum_c - \frac{n_{bc}}{n_b} \log_2 \frac{n_{bc}}{n_b}$$

Trong đó n_b là số mẫu trong nhánh b, n_t là tổng số các mẫu, và n_{bc} là số mẫu trong nhánh b của lớp c.

Thực chất người ta quan tâm đến số các mẫu tại cuối nhánh. Yêu cầu n_b và n_{bc} là cao khi thử nghiệm sinh ra các tập không đồng nhất, và là thấp khi thử nghiệm sinh ra các tập hoàn toàn thống nhất.

Độ lộn xộn tính bằng

$$\sum_c - \frac{n_{bc}}{n_b} \log_2 \frac{n_{bc}}{n_b}$$

Dù công thức chưa cho thấy "sự lộn xộn", nhưng người ta dùng nó để đo thông tin. Để thấy được các khía cạnh quan tâm, giả sử có một tập gồm các phần tử của hai lớp A và B. Nếu số phần tử của hai lớp là cân bằng, thì độ lộn xộn là 1 và giá trị cực đại về lộn xộn được tính theo công thức:

$$-1/2 \log_2 1/2 - 1/2 \log_2 1/2 = 1/2 + 1/2 = 1$$

Mặt khác nếu phần tử thuộc chỉ một trong A, B, độ lộn xộn là 0

$$-1 \log_2 1 - 0 \log_2 0 = 0$$

Độ lộn xộn bằng 0 khi tập là hoàn toàn thống nhất, và bằng 1 khi tập hoàn toàn không đồng nhất. Độ đo lộn xộn có giá trị từ 0 đến 1.

Bằng công cụ này, người ta có thể tính được độ lộn xộn trung bình của các tập tại cuối các nhánh sau lần thử nghiệm.

$$\text{Độ lộn xộn trung bình} = \sum_b \frac{n_b}{n_t}$$

Để tạo ra cây định danh, người ta dùng thủ tục SINH được trình bày như sau:

Thủ tục SINH dùng cây định danh:

Cho đến khi mỗi nút lá được ghi tến các phần tử của tập đồng nhất, thực hiện:

1. Chọn nút lá ứng với tập mẫu không đồng nhất
2. Thay thế nút này bằng các thử nghiệm cho phép chia tập không đồng nhất thành các tập đồng nhất, dựa theo tính toán độ lôn xộn.

Chuyển cây sang luật

Một khi đã dựng được cây định danh, nếu muốn chuyển các tri thức sang dạng luật thì cũng đơn giản. Người ta đi theo các nhánh của cây, từ gốc đến các nút lá, lấy các thử nghiệm làm giả thiết và phân loại nút lá làm kết luận.

Để chuyển cây định danh về tập các luật, thực hiện thủ tục tên là CAT sau:

Dùng thủ tục CAT cho phép tạo nên các luật từ cây định danh:

- *Tạo một luật từ mỗi nhánh gốc – là của cây định danh.*
- *Đơn giản hóa mỗi luật bằng cách khử các giả thiết không có tác dụng đối với kết luận của luật.*

Thay thế các luật có chung kết luận bằng luật mặc định. Luật này được kích hoạt khi không có luật nào hoạt động. Khi có nhiều khả năng, dùng phép may rủi để chọn luật mặc định.

4.2 Thuật giải ILA

Thuật giải ILA (Inductive Learning Algorithm) được dùng để xác định các luật phân loại cho tập hợp các mẫu học. Thuật giải này thực hiện theo cơ chế lặp, để tìm luật riêng đại diện cho tập mẫu của từng lớp. Sau khi xác định được luật, ILA loại bỏ các mẫu liên quan ra khỏi tập mẫu, đồng thời thêm luật mới này vào tập luật. Kết quả có được là một danh sách có thứ tự các luật chứ không là một cây quyết định. Các ưu điểm của thuật giải này có thể trình bày như sau:

- Dạng các luật sẽ phù hợp cho việc khảo sát dữ liệu, mô tả mỗi lớp một cách đơn giản để dễ phân biệt với các lớp khác.
- Tập luật được sắp thứ tự, riêng biệt – cho phép quan tâm đến một luật tại thời điểm bất kỳ. Khác với việc xử lý luật theo phương pháp cây quyết định, vốn rất phức tạp trong trường hợp các nút cây trở nên khá lớn.

Xác định dữ liệu

Tập mẫu được liệt kê trong một bảng, với mỗi dòng tương ứng với một mẫu, và mỗi cột thể hiện một thuộc tính trong mẫu.

Tập mẫu có m mẫu, mỗi mẫu gồm k thuộc tính, trong đó có một thuộc tính quyết định. Tổng số n các giá trị của thuộc tính này chính là số lớp của tập mẫu.

Tập luật R có giá trị khởi tạo là ∅.

Tất cả các cột trong bảng ban đầu chưa được đánh dấu (kiểm tra).

Thuật giả ILA

Bước 1: Chia bảng m mẫu ban đầu thành n bảng con. Mỗi bảng con ứng với một giá trị của thuộc tính phân lớp của tập mẫu.

(*thực hiện các bước 2 đến 8 cho mỗi bảng con*)

Bước 2: Khởi tạo bộ đếm kết hợp thuộc tính j, j=1

Bước 3: Với mỗi bảng con đang khảo sát, phân chia danh sách các thuộc tính theo các tổ hợp riêng biệt, mỗi tổ hợp ứng với j thuộc tính phân biệt.

Bước 4: Với mỗi tổ hợp các thuộc tính, tính số lượng các giá trị thuộc tính xuất hiện theo cùng tổ hợp thuộc tính trong các dòng chưa đánh dấu của bảng con đang xét (mà đồng thời không xuất hiện với tổ hợp thuộc tính này trên các bảng còn lại). Gọi tổ hợp đầu tiên (trong bảng con) có số lần xuất hiện nhiều nhất là tổ hợp lớn nhất.

Bước 5: Nếu tổ hợp lớn nhất bằng ∅, tăng j lên 1 và quay lại bước 3

Bước 6: Đánh dấu các dòng thỏa tổ hợp lớn nhất của bảng con đang xử lí theo lớp.

Bước 7: Thêm luật mới vào tập luật R, với vế trái là tập các thuộc tính của tổ hợp lớn nhất (kết hợp các thuộc tính bằng toán tử AND) và vế phải là giá trị thuộc tính quyết định tương ứng.

Bước 8: Nếu tất cả các dòng đều đã được đánh dấu phân lớp, tiếp tục thực hiện từ bước 2 cho các bảng con còn lại. Ngược lại (nếu chưa đánh dấu hết các dòng) thì quay lại bước 4. Nếu tất cả các bảng con đã được xét thì kết thúc, kết quả thu được là tập luật cần tìm.

Đánh giá thuật giải

Số lượng các luật thu được xác độ mức độ thành công của thuật giải. Đây chính là mục đích chính của các bài toán phân lớp thông qua một tập mẫu học.

Một vấn đề nữa để đánh giá các hệ học quy nạp là khả năng hệ thống có thể phân lớp các mẫu được đưa vào sau này.

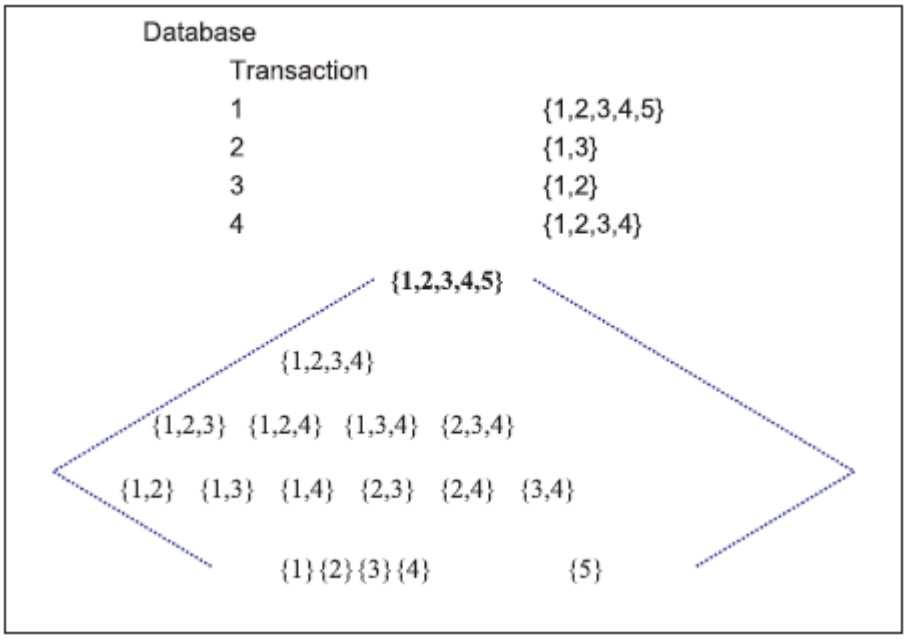
Thuật giải ILA được đánh giá mạnh hơn hai thuật giải khá nổi tiếng về phương pháp học trước đây là ID3 và AQ, đã thử nghiệm trên một số tập mẫu như Balloons, Balance, và Tic-tac-toe.

4.1 Tập phổ biến và luật kết hợp

4.1.1 Phát biểu bài toán

Giới thiệu một cách khái quát về Bài toán tìm kiếm luật kết hợp.

Đặt $I = \{i_1, i_2, ..., i_m\}$ là một tập *m* thuộc tính phân biệt. Ví dụ như tập các mặt hàng khác nhau ở siêu thị hay các tín hiệu báo động khác nhau trong mạng điện thoại. Một giao tác (*transaction*) *T* là một tập các thuộc tính trong *I*. Một giao tác được biểu thị cho việc khách hàng mua một số mặt hàng hay tập các tín hiệu báo động xảy ra trong một thời điểm. Cơ sở dữ liệu *D* tập hợp các giao tác. Tập các thuộc tính được gọi là tập thuộc tính (*itemset*). Chú ý là các thuộc tính trong tập thuộc tính được sắp xếp có thứ tự.



Hình 5. Cơ sở dữ liệu và các giao tác

Một giao tác T được gọi là hỗ trợ (*support*) một tập thuộc tính $X \subseteq I$ nếu và chỉ nếu nó chứa tất cả các thuộc tính của X, nghĩa là $X \subseteq T$. Tỉ lệ các giao tác trong D được hỗ trợ X thì được gọi là *hỗ trợ* của X, ký hiệu là support (X). Có một ngưỡng *hỗ trợ nhỏ nhất – minimum support* được người dùng định nghĩa, trong khoảng [0,1]. Một tập gọi là *phổ biến* (*frequent*) nếu và chỉ nếu nó hỗ trợ lớn hơn hoặc bằng hỗ trợ nhỏ nhất. Nếu không nó không phổ biến (*infrequent*).

Một luật kết hợp (*association rule*) có dạng $R: X \to Y$, nếu X và Y đều không phải là tập rỗng và không phân cách. *Hỗ trợ cho luật R* được định nghĩa như là support($X \cap Y$). Độ tin cậy (*confidence factor*) (biểu diễn bằng phần trăm), định nghĩa như là support($X \cap Y$) / support(X), được dùng để đánh giá độ bền của luật kết hợp.

Đích đến của việc tìm luật kết hợp là tìm tất cả các luật mà có độ hỗ trợ và tin cậy lớn hơn ngưỡng hỗ trợ và tin cậy do người dùng định nghĩa.

Người ta thường tìm luật kết hợp theo hai bước sau:

1. tìm các tập phổ biến, tiếp theo đó là
2. phát sinh các luật kết hợp.

4.1.2 Tập phổ biến cực đại là gì?

Trong tất cả các tập phổ biến một số tập thuộc tính thoả mã tính chất không có tập cha nào của chúng phổ biến, thì đó là các *tập phổ biến tối đại – maximal frequent itemset*.

Do vậy bài toán tìm các tập phổ biến có thể chuyển sang bài toán tìm tập phổ biến cực đại. Tập phổ biến cực đại được xem như là biên giới của các tập phổ biến và không phổ biến. Một khi tập phổ biến cực đại được tìm thấy, các tập phổ biến và không phổ biến sẽ tìm thấy.

4.1.3 Các tính chất của bài toán

Quá trình tìm tập phổ biến là quá trình thực hiện việc phân chia tất cả các tập thuộc tính thành ba tập hợp:

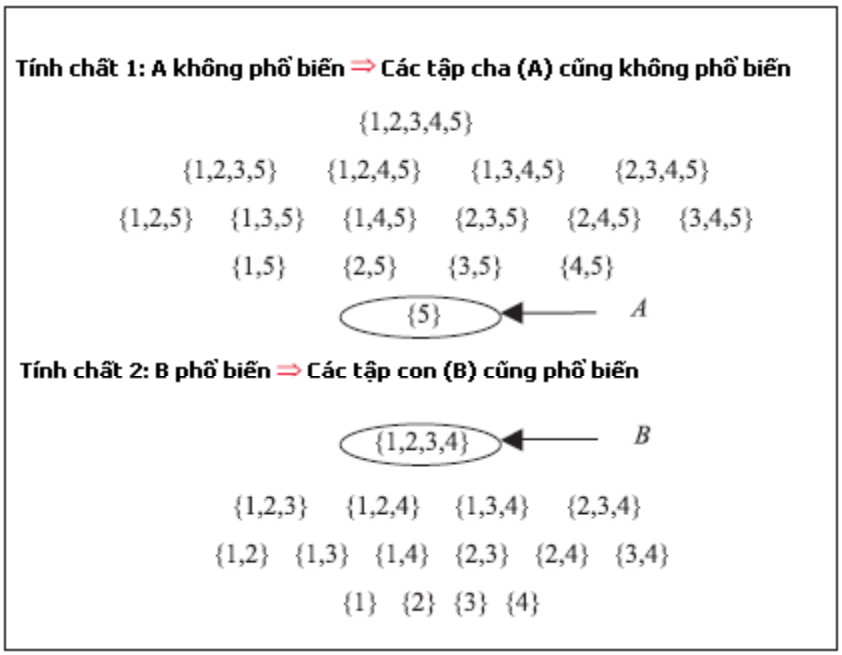
1. *phổ biến (frequent)* : Đây là tập hợp các tập thuộc tính được tìm thấy là phổ biến.
2. *không phổ biến (infrequent)* : Đây là tập hợp các tập thuộc tính được tìm thấy là không phổ biến
3. *chưa phân biệt (unclassified)* : Đây là tập hợp các tập thuộc tính khác còn lại.

Lúc đầu, các tập hợp phổ biến và không phổ biến là rỗng. Qua quá trình thực hiện, các tập hợp phổ biến và không phổ biến được mở rộng từ các tập hợp chưa được phân loại. Việc mở rộng kết thúc khi tập hợp chưa phân loại là rỗng, nghĩa là, khi tất cả các tập thuộc tính hoặc là phổ biến hoặc là không phổ biến. Nói một cách khác, quá trình kết thúc khi tập phổ biến cực đại tìm được.

Xét một *quá trình* phân loại bất kỳ các tập thuộc tính và tại một thời điểm nào đó của trong quá trình, thì một số tập thuộc tính được phân loại thành tập phổ biến, tập không phổ biến và tập chưa phân loại. Có hai tính chất quan trọng đợc sử dụng để phân loại các tập hợp chưa được phân loại:

Tính chất 1: Nếu một tập thuộc tính là không phổ biến, thì tất cả các các tập cha (*superset*) cũng không phổ biến và không cần phải khảo sát tiếp.

Tính chất 2: Nếu một tập thuộc tính là phổ biến, thì tất cả các tập con (*subset*) cũng là phổ biến và không cần phải khảo sát tiếp.



Hình 6. Hai tính chất quan trọng

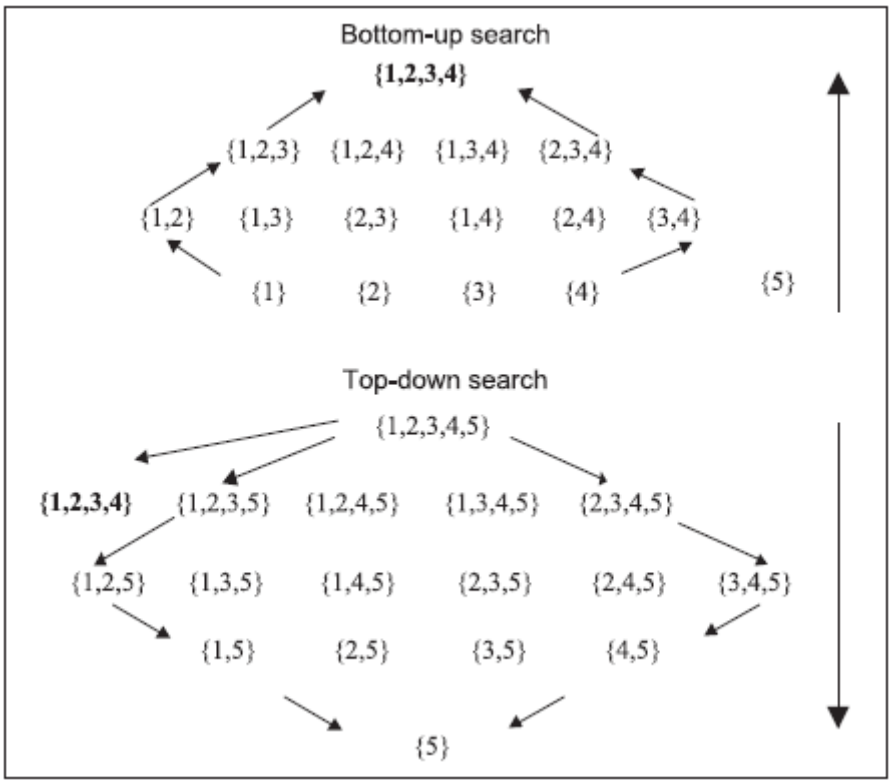
Ví dụ Xét cơ sở dữ liệu như Hình 5. Cơ sở dữ liệu và các giao tác Tập thuộc tính {5} là không phổ biến và như thế tập thuộc tính {2,5} cũng không phổ biến, vì support của {2,5} chắc chắn là sẽ nhỏ hơn hoặc bằng support của {5}. Tương tự như vậy, nếu tập thuộc tính {1,2,3,4} là phổ biến thì tập thuộc tính {1,2,3} phải phổ biến, vì có nhiều hơn hoặc bằng số lượng các giao tác chứa các thuộc tính 1, 2, và 3 hơn là các giao tác chứa các thuộc tính 1, 2, 3, và 4.

Thông thường, có hai cách có thể tìm kiếm tập phổ biến cực đại hoặc là từ dưới lên (*bottom-up*) hay từ trên xuống (*top-down*). Nếu tất cả các tập thuộc tính phổ biến cực đại được xem là ngắn (có kích thước ngắn gần với 1), thì có vẻ sử dụng cách tìm từ dưới lên sẽ hiện quả. Nếu tất cả các tập thuộc tính phổ biến cực đại được xem là lớn (có kích thước dài gần với n) thì có vẻ sử dụng cách tìm kiếm từ trên xuống hiệu quả hơn.

Đầu tiên chúng ta phác họa một cách tiếp cận thông dụng nhất để tìm MFS (tập phổ biến cực đại – **M**aximum **F**requent **S**et). Đây là cách tiếp cận theo hướng từ dưới lên *bottom-up*. Nó áp dụng lập đi lập lại các kỳ (*pass)* gồm hai bước.

Bước đầu tiên là ở kỳ k+1, tập thuộc tính có kích thước k+1 được tạo ra từ hai tập con chập k phần tử có cùng k-1 phần tử giống nhau. Một số tập thuộc tính sẽ được chặt bới (*prune*), vì chúng không cần phải xử lý tiếp nữa. Đặc biệt là những tập thuộc tính là tập cha của các tập thuộc tính không phổ biến được chặt bớt (bỏ đi), vì dĩ nhiên chúng không phổ biến (bởi Tính chất 1). Những tập thuộc tính còn lại hình thành tập hợp các ứng viên (*candidate*) cho kỳ hiện tại.

Bước thứ hai, support của các tập thuộc tính này sẽ được tính và chúng được phân loại thành phổ biến hay không phổ biến. Support của các ứng viên được tính bằng cách đọc cơ sở dữ liệu.



Hình 7. Tìm kiếm một chiều

Ví dụ Xem Hình 7. Tìm kiếm một chiều à một ví dụ của cách tiếp cận từ dưới lên. Xét cơ sở dữ liệu Hình 5. Cơ sở dữ liệu và các giao tác Có tất cả năm tập thuộc tính chập 1 ({1},{2},{3},{4},{5}) được cho là các ứng viên của kỳ đâu tiên. Sau khi tính support, tập thuộc tính {5} là không phổ biến. Bằng Tính chất 1, tất

cả các tập cha của {5} sẽ không được xem xét. Như vậy ở kỳ thứ hai các ứng viên sẽ là {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}. Với chu trình lặp lại như vậy cho đến khi tập phổ biến cực đại (trong ví dụ này là {1,2,3,4}) được tìm thấy.

Trong cách tiếp cận từ dưới lên, *mọi* tập thuộc tính phổ biến đều phải là ứng viên ở một kỳ nào đó. Như chúng ta đã thấy ở ví dụ trên, mọi tập phổ biến (các tập con của {1,2,3,4}) đều cần phải ghé thăm trước khi đến được tập phổ biến cực đại. Như vậy, cách tiếp cận này chỉ hiệu quả khi *tất cả* các tập phổ biến cực đại là ngắn.

Khi có *một số* tập phổ biến cực đại dài xảy ra, cách thức này sẽ không hiệu quả. Trong trường hợp đó, nó cần một cách tìm kiếm hiệu quả hơn cho các tập phổ biến cực đại dài.- sử dụng cách tiếp cận từ trên xuống.

Cách tiếp cận từ trên xuống (*top-down*), bắt đầu với một tập thuộc tính chập *n* và rồi giảm dần kích thước của các ứng viên đi một trong mỗi kỳ. Khi tập thuộc tính chập *k* được quyết định là không phổ biến thì tất cả các tập con chập (*k*-1) sẽ được khảo sát trong kỳ tiếp theo. Nói cách khác, nếu một tập thuộc tính chập *k* là phổ biến thì tất cả các tập con của nó chắc chắn là phổ biến và không cần phải xét tiếp (theo Tính chất 2).

Ví dụ Xem Hình 7. Tìm kiếm một chiều và xét cùng một cơ sở dữ liệu như Hình 5. Cơ sở dữ liệu và các giao tác Tập thuộc tính chập 5 {1,2,3,4,5} là ứng viên duy nhất của kỳ đầu tiên. Sau khi tính support, nó sẽ không phổ biến. Các ứng viên tiếp theo của kỳ thứ hai là tất cả các tập con chập 4 của tập {1,2,3,4,5}. Trong ví dụ này, tập {1,2,3,4} là phổ biến và tất cả các tập thuộc tính khác là không phổ biến. Theo Tính chất 2, tất cả các tập con của {1,2,3,4} sẽ phổ biến và không cần phải xét tiếp. Thủ tục này được lặp lại cho đến khi tập phổ biến cực đại được tìm thấy (nghĩa là, sau khi tất cả các tập thuộc tính không phổ biến được xem xét).
Với cách tiếp cận này, *mọi* tập thuộc tính không phổ biến đều phải xét qua thực sự. Như Hình 7. Tìm kiếm một chiều mọi tập thuộc tính không phổ biến (tập thuộc

tính {5} và các tập cha của nó) cần phải xem xét trước khi các tập phổ biến cực đại được tìm thấy.

4.1.4 Một số thuật giải thông dụng

4.1.4.1 Thuật giải khởi đầu AIS

Bài toán tìm luật kết hợp được giới thiệu đầu tiên trong bài báo của R.Agrawal, T.Imielinski và A.Swami đưa ra. Thuật toán được gọi là AIS được đề xuất để tìm tập phổ biến cực đại. Để tìm các tập thuộc tính phổ biến AIS tạo ra các ứng viên trong lúc đọc cơ sở dữ liệu. Trong suốt mỗi kỳ, toàn bộ cơ sở dữ liệu được đọc. Một ứng viên được tạo ra bằng cách thêm các thuộc tính vào các tập hợp, được gọi là *tập thuộc tính biên* (*frontier*), được tìm thấy là phổ biến trong kỳ vừa mới qua. Để tránh phát sinh các ứng viên không có trong cơ sở dữ liệu, các ứng viên được phát sinh chỉ từ các giao tác. Các ứng viên mới sẽ được phát sinh bằng cách mở rộng tập thuộc tính biên với các thuộc tính còn lại trong một giao tác.

Thí dụ, nếu tập {1,2} là tập thuộc tính phổ biến, trong giao tác {1,2,3,4,6} chúng ta có các ứng viên như sau:

1. {1,2,3} được xem là phổ biến: tiếp tục mở rộng.
2. {1,2,3,4} được xem là không phổ biến: không mở rộng thêm nữa
3. {1,2,3,5} được xem là phổ biến: không thể mở rộng thêm nữa
4. {1,2,4} được xem là không phổ biến: không mở rộng thêm nữa
5. {1,2,5} được xem là phổ biến: không thể mở rộng thêm nữa

Tập thuộc tính {1,2,3,4,6} không được xem xét, vì {1,2,3,4} là tập không phổ biến. Tương tự, {1,2,4,6} không được xem xét, vì {1,2,4} là tập không phổ biến. Các tập thuộc tính {1,2,3,5} và {1,2,5} không xem xét tiếp vì thuộc tính 5 không nằm trong giao tác.

Hai heuristics phức tạp, *tối ưu tập còn lại và tối ưu chức năng chặt tỉa*, được sử dụng để chặt bớt các ứng viên. Thật không may, thuật toán này vẫn phát sinh ra quá nhiều các ứng viên.

4.1.4.2 Thuật giải Apriori

Một thuật giải khá tốt do R.Agrawal và R.Srikant đưa ra. Đây là một cách tiếp cận chuẩn của thuật toán từ dưới lên, với cách thực hiện tốt hơn rất nhiều so với thuật toán AIS. Nó lặp đi lặp lại thuật toán *Apriori-gen* để phát sinh ra các ứng viên và sau đó đếm support của các ứng viên bằng cách đọc toàn bộ cơ sở dữ liệu 1 lần.

Thuật toán Thuật toán Apriori
Dữ liệu vào: cơ sở dữ liệu và Minsupport được người dùng định nghĩa
Dữ liệu ra: các tập phổ biến cực đại
1. L_0 := ∅; k = 1;
2. C_1 := {{i}| $i \in I$ }
3. *MFS* := ∅
4. Trong khi $C_k \neq \emptyset$
5. đọc cơ sở dữ liệu và đếm support của C_k
6. L_k := {các tập thuộc tính phổ biến trong C_k}
7. C_{k+1} := *Apriori-gen*(L_k)
8. $k := k + 1$;
9. *MFS* := (*MFS* $\cup$ L_k) \ {*MFS* | *MFS* $\subseteq$ L_k}
10. trả về *MFS*

Apriori- gen là một thuật toán phát sinh ứng viên. Nó dự vào Tính chất 1 được đề cập ở trên.

Thuật toán Thuật toán Apriori-gen
Dữ liệu vào: L_k, đây là tập các tập thuộc tính phổ biến được tìm thấy trong kỳ k
Dữ liệu ra: tập các ứng viên mới C_{k+1}
1. gọi thủ tục *join* để phát sinh tập ứng viên tạm thời

```
2. gọi thủ tục prune để tìm được tập ứng viên cuối cùng
```

Thủ tục *join* của thuật toán Apriori-gen có nhiệm vụ kết hai tập thuộc tính phổ biến chập k, mà chúng có (k-1) phần tử đầu giống nhau, thành một tập thuộc tính chập (k+1) - một ứng viên tạm thời.

```
Thuật toán Thủ tục join của Thuật toán Apriori-gen
Dữ liệu vào: Lk, đây là tập các tập thuộc tính phổ biến được tìm thấy
             trong kỳ k
Dữ liệu ra: tập các ứng viên mới Ck+1 tạm thời
/* Các tập thuộc tính phải được sắp xếp theo thứ tự từ điển */
1. Với mỗi i từ 1 đến |Lk - 1|
2.   Với mỗi j từ i+1 đến |Lk|
3.     Nếu Lk.itemseti và Lk.itemsetj có cùng (k-1) phần tử đầu
4.       Ck+1 = Ck+1 ∪ {Lk.itemseti ∪ Lk.itemsetj}
5.     Ngược lại
6.       break
```

Tiếp theo đó là thủ tục *prune* (chặt bớt) được dùng để bỏ đi các ứng viên tạm thời c trong C_{k+1} mà có tập con chập k nào đó của c không nằm trong tập phổ biến L_k. Nói cách khác, tập cha của các tập thuộc tính không phổ biến phải được loại bỏ.

```
Thuật toán Thủ tục prune của Thuật toán Apriori-gen
Dữ liệu vào: tập các ứng viên mới Ck+1 tạm thời được phát sinh từ thủ tục
             join ở trên
Dữ liệu ra: tập các ứng viên mới Ck+1 cuối cùng mà không còn chứa bất kỳ
             một tập con không phổ biến nào
1. Với tất cả các tập thuộc tính c trong Ck+1
2.   Với tất cả các tập con chập k s của c
3.     nếu s ∉ Lk
4.       xóa c khỏi Ck+1
```

Thuật toán Apriori-gen rất thành công trong việc giảm bớt số lượng các ứng viên.

4.1.4.3 Thuật giải DHP (sử dụng bảng HASH)

DHP cố gắng cải tiến thuật toán Apriori bằng cách sử dụng bộ lọc băm (hash filter) để đếm support cho các kỳ tiếp theo. Thuật toán này nhắm vào việc giảm bớt số lượng các ứng viên trong kỳ thứ hai, kỳ được xem là rất lớn.

Bằng cách sử dụng bộ lọc, một số các ứng viên có thể được bỏ bớt trước khi đọc cơ sở dữ liệu trong kỳ tiếp theo. Tuy nhiên, một vài nghiên cứu cho thấy sự tối ưu của thuật toán này không tốt bằng cách sử dụng một bảng hai chiều[1].

4.1.4.4 Một số thuật giải khác

4.1.4.4.1 Thuật toán Partition

Ý tưởng thuật toán là chia cơ sở dữ liệu theo chiều ngang (tức là phân tách các giao tác) thành các phần dủ nhỏ để vừa với bộ nhớ. Mỗi phần có một *tập phổ biến cục bộ* (*local frequent set*) ở kỳ đầu tiên. Quá trình xử lý các phần được thực hiện theo cách tiếp cận từ dưới lên giống như thuật toán Apriori như với cấu trúc dữ liệu khác. Sau khi tất cả các tập phổ biến cục bộ được tìm thấy, hợp chúng lại thành một tập hợp, gọi là *tập phổ biến toàn cục* (*global frequent set*) thành một tập cha của các tập phổ biến này. Dựa trên một điều là nếu tập thuộc tính phổ biến thì chắc nó phải phổ biến ít nhất trên một phần. Tương tự, nếu tập thuộc tính không phô biến trong bất kỳ phần nào, thì nó sẽ không phổ biến.

Kỳ thứ hai, dữ liệu được đọc một lần nữa và tính support thực sự cho tập ứng viên toàn cục. Như vậy toàn bộ quá trình chỉ diễn ra trong hai kỳ.

Để đọc cơ sở dữ liệu với mỗi lần kích thước của ứng viên tăng dần, cơ sở dữ liệu phải chuyển thành một cấu trúc dữ liệu mới, được gọi là *danh sách TID*.

[1] Chưa tìm được tài liệu chứng minh

Mỗi một ứng viên sẽ chứa một danh sách các ID giao tác mà ứng viên này support được. Cơ sở dữ liệu cần phải được chia thành nhiều phần có kích thước vừa với bộ nhớ chính. Tuy nhiên, danh sách TID này có thể xảy ra khả năng tràn, vì ID giao tác cho một giao tác chứa m thuộc tính có thể xảy ra, trong trường hợp xấu nhất, là trong $\binom{m}{k}$ danh sách TID trong kỳ lần thứ k.

Có ba vấn đề chính khi tiếp cận thuật toán Partition. Đầu tiên, nó yêu cầu phải quyết định chọn một kích thước các phần phải tốt để biểu diễn. Nếu quá lớn, thì TID sẽ phát triển rất nhanh và sẽ không còn đủ bộ nhớ để chứa. Nhưng nếu quá nhỏ, thì sẽ xảy ra quá nhiều ứng viên toàn cục và hầu hết chúng đều là không phổ biến. Thứ hai, các tập phổ biến cục bộ sẽ rất khác biệt nhau, do vậy mà ứng viên cục bộ sẽ rất lớn. Thứ ba, thuật toán này xem xét nhiều ứng viên hơn thuật toán Apriori. Nếu có một thuộc tính phổ biến cực đại dài, thì thuật toán này không thể thực hiện được.

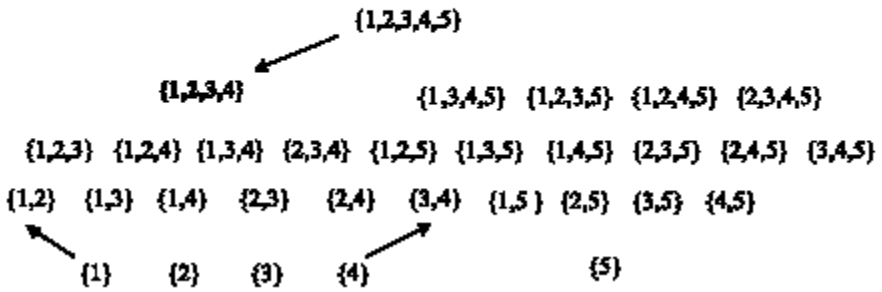
4.1.5 Thuật giải tăng cường

4.1.5.1 Tiếp cận bằng cách giảm số lượng ứng viên và số kỳ duyệt

Như đã đề cập, hướng tiếp cận bottom-up cho kết quả tốt trong trường hợp tất cả các tập phổ biến tối đại là ngắn và phương pháp top-down tốt khi toàn bộ tập các tập phổ biến tối đại là dài (số lượng phần tử trong tâp lớn).Nếu các tập phổ biến vừa ngắn vừa dài thì sử dụng một trong hai phương pháp trên sẽ không cho kết quả tốt. Để thiết kế ra một thuật toán có khả năng tìm các tập phổ biến tối đại cả ngắn lẫn dài một cách hiệu quả, người ta nghĩ đến việc thực thi cả hai chương trình bottom-up và top-down cùng một lúc. Tuy nhiên cách này không cho hiệu quả tốt nhất, chúng ta sẽ cùng tìm hiểu thuật toán sau đây.

Nhắc lại, phương pháp bottom-up sử dụng tính chất 1, còn top-down chỉ dùng tính chất 2 để làm giảm số lượng ứng viên. Tiếp cận theo hướng kết hợp cả hai tính chất để tỉa cành các ứng viên, và sử dụng thông tin vừa tập hợp được trong

khi duyệt theo một chiều để lọc bỏ nhiều ứng viên hơn trong lúc duyệt chiều còn lại. Nếu vài tập phổ biến tối đại nào đó vừa được tìm thấy theo chiều top-down, chúng sẽ được dùng để loại trừ một hoặc nhiều ứng viên trong khi duyệt theo chiều bottom-up. Các tập con của tập phổ biến tối đại này có thể được lọc bỏ (tỉa cành) vì chúng phổ biến (tính chất 2). Dĩ nhiên nếu một tập *không* phổ biến được tìm thấy theo chiều bottom-up, chúng sẽ được dùng để loại bỏ vài ứng viên theo chiều top-down (do tính chất 1). Phương pháp tìm kiếm theo cả hai hướng này sử dụng cả hai tính chất 1 và 2 nên nó cho kết quả tìm kiếm nhanh hơn, được đặt tên là Pincer-Search. Ví dụ sau sẽ làm rõ khái niệm của phương pháp này.



Hình 8: Giảm số lượng ứng viên và số lần duyệt

Hình trên là tiến trình của thuật toán Pincer-Search. Trong bước đầu tiên, tất cả 5 tập hợp (mỗi tập 1 phần tử) là các ứng viên cho bottom-up và tập gồm 5 phần tử {1,2,3,4,5} là ứng viên duy nhất cho top-down. Sau khi tính toán độ phổ biến, tập không phổ biến {5} được bottom-up phát hiện, và thông tin này được chia xẻ cho top-down. Tập không phổ biến {5} này không những cho phép bottom-up loại bớt các tập cha vốn là ứng viên của nó, mà {5} còn cho phép top-down tìm và loại bỏ những tập cha (thường khi sẽ là ứng viên của nó) trong bước thứ hai.

Trong bước 2, các ứng viên cho bottom-up là {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}. Các tập {1,5}, {2,5}, {3,5}, {4,5} không là các ứng viên vì chúng là

tập cha của tập {5}. Ứng viên duy nhất cho top-down trong bước 2 là {1,2,3,4}. Vì tất cả các tập con 4 phần tử khác của {1,2,3,4,5} đều là tập cha của {5}. Sau khi tính toán độ phổ biến lần thứ hai, top-down phát hiện ra {1,2,3,4} là tập phổ biến. Thông tin này được chia xẻ cho bottom-up. Tất cả các tập con của nó là phổ biến và cần được bỏ qua không tính toán. Như vậy {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4} không là ứng viên cho bottom-up và top-down. Sau đó chương trình kết thúc vì không còn ứng viên cho bottom-up và top-down.

Trong ví dụ này, phương pháp tiếp cận 2 chiều xem xét ít ứng viên hơn và số bước đọc cơ sở dữ liệu cũng ít hơn cả hai cách top-down và bottom-up. Trong ví dụ này, bottom-up nguyên thủy phải thực hiện 4 bước và top-down nguyên thủy phải đến 5 bước. Pincer-search chỉ cần 2 bước. Thật vậy, phương pháp 2 chiều có số bước duyệt tối đa cũng chỉ bằng min {số bước của bottom-up, số bước của top-down}. Giảm số lượng ứng viên là nhân tố quan trọng cho việc tăng hiệu năng của giải thuật tìm tập phổ biến, vì chi phí cho toàn bộ tiến trình phát sinh từ việc đọc cơ sở dữ liệu (thời gian I/O) để tính toán độ phổ biến cho ứng viên (thời gian CPU) và phát sinh ứng viên mới (thời gian CPU). Việc đếm độ phổ biến cho các ứng viên là công đoạn tốn kém nhất. Vì thế số lượng ứng viên chi phối toàn bộ thời gian của tiến trình. Giảm số lượng ứng viên không chỉ giảm thời gian I/O mà còn giảm thời gian CPU, vì lượng ứng viên phải tính toán và phát sinh đã ít đi. Phần tiếp theo sẽ trình bày chi tiết hơn.

4.1.5.2 Tìm kiếm hai hướng bằng cách sử dụng MFCS (maximum frequent candidate set)

Chúng ta cần một cấu trúc dữ liệu mới để thuật toán tìm tập phổ biến tối đại (MFCS) có thể hoạt động.

Định nghĩa 1: Xem xét một số điểm trong quá trình thuật toán thực thi để tìm ra tập phổ biến tối đại. Vài tập là phổ biến và vài tập không phổ biến, một số lại không phân loại được. MFCS là một tập hợp của tất cả các tập hợp lớn nhất mà

không bị xem là *không phổ biến*. Cụ thể, nó là tập hợp nhỏ nhất của các tập hợp, phải thỏa điều kiện sau:

Lớp_phổ_biến $\subseteq \cup \{ 2^X \mid X \in MFCS\}$

Lớp_không_phổ_biến $\cap \{ 2^X \mid X \in MFCS\} = \varnothing$

trong đó lớp_phổ_biến và lớp_không_phổ_biến tương ứng là *tập hợp* của các tập phổ biến và các tập không phổ biến. Vì hiển nhiên bất kỳ điểm nào của thuật toán, MFCS là tập cha của MFS, khi tiến trình hoàn tất, MFCS và MFS cân bằng.

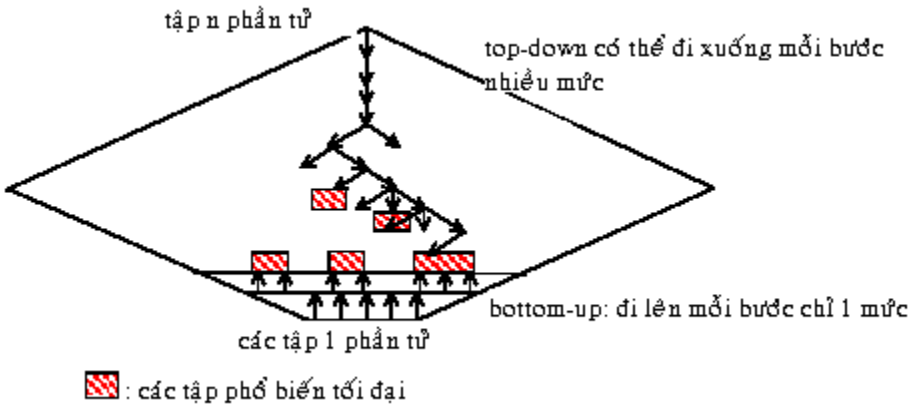
Thuật toán này tính toán theo hướng tiếp cận của bottom-up tìm theo chiều ngang (chiều rộng). Nói tóm lại, ở mỗi bước, để đếm độ phổ biến của các ứng viên theo chiều bottom-up, thuật toán cũng đếm độ phổ biến của các tập hợp trong MFCS: tập hợp này được sửa lại cho hợp với quá trình tìm kiếm theo hướng top-down. Điều này có lợi trong khi tỉa cành các ứng viên, nhưng nó đòi hỏi cách phát sinh ứng viên phải khác đi.

Xét ở bước thứ k, trong lúc phân loại từng tập gồm k phần tử vào lớp thích hợp, nếu có vài tập nào đó là một thành phần của MFCS, X chẳng hạn, được nhận ra là phổ biến, tức toàn bộ các tập con của nó đều phổ biến. Vì vậy tất cả tập con này được loại bỏ khỏi tập ứng viên mà bottom-up phải xét ở bước này. Chúng, và tất cả các tập cha của chúng, sẽ không bao giờ còn là ứng viên cho đến hết tiến trình thực thi, như vậy hiệu năng tăng lên. Dĩ nhiên, khi tính toán tập phổ biến tối đại sau cùng, chúng sẽ không bị bỏ sót.

Tương tự, khi bottom-up tìm thấy một tập không phổ biến mới, thuật toán sẽ dùng nó để cập nhật MFCS. Các tập con của MFCS phải không chứa tập con không phổ biến này.

Hình sau sẽ chỉ ra thuật toán kết hợp tìm kiếm theo cả hai chiều. Khởi tạo MFCS chỉ chứa một phần tử đơn, tập hợp n phần tử chứa toàn bộ thành phần trong cơ sở dữ liệu. Xét bước đầu tiên của bottom-up, nếu có m tập gồm 1 phần tử là tập không phổ biến (sau khi đọc cơ sở dữ liệu 1 lần), MFCS sẽ có các phần tử chủ

chốt của n-m. Tập này được sinh ra do tập MFCS ban đầu loại ra m phần tử không phổ biến. Trong trường hợp này, top-down đi xuống được m mức trong bước 1. Thông thường, không giống như bottom-up, phương pháp tìm kiếm đi lên mỗi bước một mức, top-down có thể đi xuống mỗi bước nhiều mức.



Hình 9: Tìm kiếm theo 2 chiều top-down và bottom-up

Bằng cách dùng MFCS, chúng ta có thể phát hiện ra các tập phổ biến tối đại ở các bước sớm hơn. Việc này có thể làm giảm số lượng ứng viên và số lần đọc cơ sở dữ liệu và do đó giảm thời gian I/O và thời gian CPU. Điều này đặc biệt quan trọng khi các tập phổ biến tối đại được khám phá sớm có nhiều phần tử.

Chúng ta cần quan tâm hai vấn đề, thứ nhất là làm sao nâng cấp hiệu suất của MFCS, thứ hai là mỗi tập con của tập phổ biến tối đại vừa tìm thấy phải được loại bỏ, làm sao phải phát sinh đúng các tập ứng viên cho các bước sau của chiều bottom-up?

4.1.5.3 Cập nhật MFCS một cách hiệu quả

Xét vài tập Y vừa được phân vào lớp không phổ biến, theo định nghĩa của MFCS, nó sẽ là tập con của một hoặc nhiều tập trong MFCS và ta cần cập nhật MFCS sao cho các tập con của nó không chứa Y. Để cập nhật MFCS, ta sẽ làm như sau đối với mọi tập cha của Y (các tập cha này nằm trong MFCS): chúng ta

thay thế mỗi tập X bởi tập |Y|, bằng cách loại bỏ khỏi X 1 phần tử thuộc Y. Tập mới phát sinh sẽ được thêm vào MFCS chỉ khi nó không là tập con của bất kỳ tập nào trong MFCS. Chúng ta thực hiện quá trình này cho tất cả mọi tập không phổ biến mới phát hiện được.

Thuật toán MFCS-gen ở bước k được trình bày như sau:

```
Thuật toán: MFCS-gen
Dữ liệu vào: MFCS cũ và tập không phổ biến S^k tìm thấy ở bước k
Dữ liệu ra:MFCS mới
1. Với mỗi tập s ∈ S_k
2.   Với mỗi tập m ∈ MFCS
3.     Nếu s là tập con của m
4.       MFCS := MFCS \ {m}
5.       Với mỗi phần tử e ∈ tập s
6.         Nếu m \ {e} không  là tập con của bất kỳ tập nào trong
   MFCS
7.            MFCS = MFCS ∪ {m \ {e}}
8. return MFCS
```

Xét ví dụ sau, giả sử {{1,2,3,4,5,6}} là MFCS cũ hiện tại, và 2 tập không phổ biến vừa tìm thấy là {1,6} và {3,6}.

Đầu tiên xét tập không phổ biến {1,6}, vì {{1,2,3,4,5,6}}-thành phần của MFCS chứa 1 và 6, một trong những tập con của nó sẽ là {1,6}. Xóa 1 khỏi tập {1,2,3,4,5,6} ta được {2,3,4,5,6} và nếu xóa 6 ta được {1,2,3,4,5}.

Sau khi xét tập {1,6}, MFCS trở thành {{1,2,3,4,5},{2,3,4,5,6}}.

Sau đó tập {3,6} được dùng để cập nhật MFCS này. Vì {3,6} là con của {2,3,4,5,6}, nên 2 tập {2,3,4,5} và {2,4,5,6} được phát sinh để thay thế cho {2,3,4,5,6}. Tập {2,3,4,5} là con của {1,2,3,4,5} trong MFCS mới, và như vậy nó sẽ không được thêm vào MFCS. Vì vậy MFCS mới trở thành {{1,2,3,4,5},{2,4,5,6}}

Bổ đề 1: Thuật toán MFCS-gen cập nhật đúng MFCS

Chứng minh: thuật toán loại bỏ toàn bộ những tập nào không phổ biến, vì vậy *tập cuối cùng* sẽ không chứa bất kỳ tập không phổ biến nào như là tập con của từng phần tử của *nó*. Bước 7 chỉ xóa đi một phần tử (item) khỏi tập m: tập con dài nhất của m mà không chứa tập không phổ biến s. Vì thuật toán này luôn phát sinh những tập dài nhất, nên cuối cùng số lượng tập hợp sẽ đạt cực tiểu. Do đó thuật toán này cập nhật đúng MFCS.

4.1.5.4 Thuật giải Phát sinh ứng viên cải tiến

Nhắc lại, một tập ứng viên dự bị sẽ được phát sinh sau khi gọi thủ tục tỉa cành (prune). Trong thuật tóan này, sau khi một tập phổ biến tối đại được thêm vào MFS, và tất cả các tập con phổ biến của nó sẽ bị loại bỏ. Qua ví dụ, chúng ta thấy rằng, nếu áp dụng thủ tục join nguyên thủy của thuật toán Apriori-gen, một vài trong số những tập cần thiết có thể bị bỏ sót khỏi tập ứng viên dự bị.

Trở lại ví dụ trên, giả sử các tập phổ biến gồm 3 phần tử ban đầu là L_3 = {{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {2,4,6}, {2,5,6}, {3,4,5}, {4,5,6}}. Giả sử tập {1,2,3,4,5} trong MFCS là phổ biến.

Vì vậy tất cả các tập 3 phần tử đều được xóa khỏi L_3 ngoại trừ {2,4,6}, {2,5,6}, {4,5,6}. Vì thuật toán Apriori-gen dùng k-1 phần tử đầu tiên kiểm tra trên tập phổ biến để phát sinh các ứng viên mới, và không có bất kỳ 2 tập nào trong tập {{2,4,6}, {2,5,6}, {4,5,6}} có chung 2 phần tử đầu tiên, nên không có ứng viên nào được thủ tục join phát sinh trên tập phổ biến này. Tuy nhiên tập ứng viên dự bị đúng ra phải là {{2,4,5,6}}.

Vì vậy ta cần phải tìm lại những ứng viên đã bị bỏ qua này.

4.1.5.4.1 Thủ tục mới để phát sinh tập ứng viên dự bị

Trong thủ tục mới này, join của thuật toán Apriori-gen cũ được gọi lần đầu để phát sinh tập ứng viên tạm thời, có thể không hoàn thành được. Khi nó không

hoàn thành, một thủ tục tái tìm kiếm sẽ được gọi để tìm lại những ứng viên vừa bị bỏ qua.

Ta có thể tìm lại tất cả các ứng viên bị bỏ sót bằng cách đem một vài tập trở vào tập phổ biến hiện tại.

Các tập vừa được đưa trở lại này được rút trích từ tập MFS, cho đến nay đang chứa đựng toàn bộ các tập phổ biến.

Xét bước k, một tập X trong MFS và một tập Y trong tập phổ biến hiện tại, sao cho |X| > k. Giả sử k-1 phần tử đầu tiên của Y nằm trong X, và phần tử thứ k-1 của Y bằng với phần tử thứ j của X. Ta có được các tập gồm k phần tử của X mà có k-1 phần tử đầu tiên giống Y bằng cách đặt 1 phần tử của X có số thứ tự lớn hơn j và kết hợp nó với k-1 phần tử đầu tiên của Y, được một trong những tập con k phần tử này. Sau khi tìm được các tập k phần tử rồi, chúng ta khôi phục các ứng viên bằng cách kết hợp chúng với tập Y theo giải thuật sau:

Thuật toán: Thủ tục khôi phục ứng viên
Dữ liệu vào: C_{k+1} sau thủ tục *join* , L_k , MFS hiện tại
Dữ liệu ra: Tập ứng viên C_{k+1} hoàn hảo

```
1.  Với mỗi tập l trong Lk
2.    Với mỗi tập m trong MFS
3.      Nếu k-1 phần tử đầu trong trong l cũng thuộc m
4.      /*giả sử phần tử j của m = phần tử k-1 của l (hay m.item_j
    = l.item_k-1)*/
5.        Cho i= j+1 đến |m|
6.            Ck+1 = C k+1 ∪ {{l.item1, l.item2,…, l.itemk,,m.item i
    }}
```

Ví dụ: MFS = {{1,2,3,4,5}} và tập phổ biến hiện nay là {{2,4,6}, {2,5,6}, {4,5,6}}. Tập con 3 phần tử duy nhất của {{1,2,3,4,5}} cần được khôi phục cho tập {2,4,6} để phát sinh một ứng viên mới là {2,4,5}. Vì {2,4,5} là tập con duy nhất của {{1,2,3,4,5}} mà có cùng chiều dài và cùng giống 2 phần tử đầu tiên như

tập {2,4,6}. Kết hợp {2,4,5} và {2,4,6}, ta khôi phục lại ứng viên đã mất là {2,4,5,6}. Không có tập nào cần khôi phục cho {2,5,6} và {4,5,6}.

Nhóm thứ hai các tập cần được khôi phục gồm có các tập con k phần tử này của MFS có k-1 phần tử đầu giống nhau nhưng lại không có chung tập cha nào trong MFS. Có thể áp dụng một thủ tục khôi phục tương tự sau khi chúng đã được phục hồi.

4.1.5.4.2 Thủ tục prune (tỉa cành) mới

Sau bước phục hồi, một tập ứng viên dự bị sẽ được phát sinh. Chúng ta có thể tiến hành thủ tục prune. Thay vì kiểm tra xem tất cả các tập con k phần tử của tập X có nằm trong L_k hay không, đơn giản hơn chúng ta có thể kiểm tra xem liệu X có là tập con của một tập nào đó trong MFCS hiện thời hay không. So với thủ tục prune cũ, ta dùng ít hơn 1 vòng lặp trong thuật toán mới như sau:

Thuật toán: thủ tục prune mới
Dữ liệu vào: MFCS hiện tại và C_{k+1} sau các thủ tục join và recovery (khôi phục)
Dữ liệu ra: Tập ứng viên cuối cùng C_{k+1}
1. Với mỗi tập c trong C_{k+1}
2. Nếu c không là tập con của bất kỳ tập nào trong MFCS hiện tại
3. Xóa c khỏi C_{k+1}

4.1.5.4.3 Thuật toán phát sinh ứng viên mới hoạt động đúng

Tóm lại, quá trình phát sinh ứng viên mới của chúng ta gồm 3 bước như sau:

Thuật toán: Phát sinh ứng viên mới
Dữ liệu vào: L_k, MFCS hiện tại và MFS hiện tại
Dữ liệu ra: Tập ứng viên C_{k+1} mới
1. Gọi thủ tục *join* như trong thuật toán Apriori
2. Gọi thủ tục *recovery* nếu cần
3. Gọi thủ tục *prune* mới

Bổ đề 2: Thuật toán mới để phát sinh ứng viên sẽ sinh ra một tập ứng viên đúng đắn

Chứng minh: Nhắc lại quá trình phát sinh ứng viên trong Apriori-gen. Có 4 trường hợp xảy ra khi ta kết hợp 2 tập phổ biến k phần tử là I và J mà chúng có k-1 phần tử đầu giống nhau, để tạo một tập k+1 phần tử làm ứng viên dự bị mới. Ngay cả khi chúng ta đã xóa đi các tập con của MFS khỏi tập phổ biến hiện tại, thuật toán vẫn xử lý đúng các trường hợp:

Trường hợp 1: {I không là tập con của X | với mọi X trong MFS} và {J không là tập con của Y | với mọi Y trong MFS}. Cả hai tập đều thuộc tập phổ biến hiện tại, thủ tục *join* sẽ kết hợp chúng và sinh ra một ứng viên dự bị.

Trường hợp 2: {I là tập con của X | X thuộc MFS} và {J không là tập con của Y | với mọi Y trong MFS}. I bị xóa khỏi tập phổ biến hiện tại. Tuy nhiên kết hợp I và J lại sẽ sinh ra 1 ứng viên cần kiểm tra. Thủ tục *recovery* sẽ tìm lại các ứng viên trong trường hợp này.

Trường hợp 3: {cả I và J đều là các tập con của vài tập X nào đó | X thuộc MFS}. Không kết hợp chúng, vì ứng viên mà chúng phát sinh sẽ là tập con của X, và do đó sẽ phổ biến.

Trường hợp 4: Không thuộc trường hợp 3 và {I là tập con của vài X | $X \in$ MFS} và {J là tập con của vài Y | $Y \in$ MFS} và $X \neq Y$. Cả I và J đều bị xóa khỏi tập phổ biến hiện tại. Tuy nhiên, nếu kết hợp chúng, ta sẽ sinh được 1 ứng viên cần thiết. Tương tự *recovery* sẽ tìm lại các ứng viên bị mất.

Thuật toán này kết hợp các tập phổ biến lại giống như cách Apriori-gen đã làm, cũng phát sinh tất cả các ứng viên ngoại trừ các tập là con của MFS. Ở bước *recovery* , nếu cần, vài tập con của MFS sẽ được hồi phục lai cho bước tiếp theo.

Bổ đề 1 chỉ ra MFCS sẽ được duy trì đúng trong mỗi bước. Vì vậy, thủ tục *prune* mới sẽ bảo đảm chắc chắn tập ứng viên dự bị sẽ không chứa các tập cha của các tập không phổ biến. Do đó thuật toán mới để phát sinh ứng viên là đúng.

4.1.5.5 Thuật toán Pincer-search thuần túy

Thuật toán như sau dựa trên hướng tiếp cận kết hợp để quyết định tập phổ biến tối đại:

Thuật toán: Pincer-search
Dữ liệu vào: cơ sở dữ liệu và ngưỡng min-supp do người dùng định nghĩa
Dữ liệu ra: MFS chứa tất cả các tập phổ biến tối đại

1. $L_0 := \emptyset$; k:= 1; $C_1 := \{\{i\} \mid i \in I\}$
2. MFCS := {{1,2,3,…,n}}; MFS := ∅
3. Trong khi $C_k \neq \emptyset$
4. Đọc CSDL và đếm số support cho C_k và MFCS
5. Xóa các tập phổ biến khỏi MFCS và thêm chúng vào MFS
6. L_k := {các tập phổ biến trong C_k} \ {các tập con của MFS}
7. S_k := {các tập không phổ biến trong C_k}
8. Nếu $S_k \neq \emptyset$, gọi thuật toán *MFCS-gen*
9. Gọi thủ tục *join* để phát sinh C_{k+1}
10. Nếu có bất kỳ tập không phổ biến nào của C_k bị xóa trong bước 6
11. Gọi thủ tục *recovery* để phục hồi các ứng viên cho C_{k+1}
12. Gọi thủ tục *prune* mới để tỉa cành các ứng viên cho C_{k+1}
13. k := k+1
14. Hết lặp
15. Trả về MFS

Trong đó dòng 9 đến 12 tạo nên thủ tục phát sinh ứng viên mới. Khởi tạo MFCS chỉ chứa 1 tập mà gồm tất cả thành phần trong cơ sở dữ liệu. MFCS được cập nhật mỗi lần có tập không phổ biến được tìm thấy (dòng 8). Nếu 1 tập trong MFCS là phổ biến, thì các tập con của nó sẽ không tham gia vào bước tính độ phổ biến và phát sinh tập ứng viên. Dòng 6 loại trừ những tập nào là con của bất kỳ tập nào trong MFS hiện tại (MFS chứa các tập phổ biến tìm thấy trong MFCS). Nếu

xóa đi vài tập trong L_k , thuật toán sẽ gọi thủ tục recovery để tìm lại những ứng viên đã mất (dòng 11).

Định lý 1: Thuật toán Pincer-search phát sinh tất cả tập phổ biến tối đại

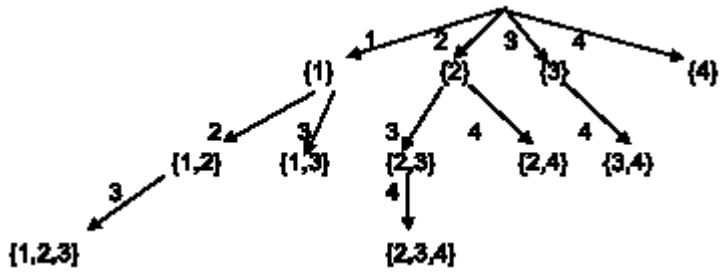
Chứng minh: Bổ đề 1 chỉ ra rằng thuật toán của chúng ta sẽ phát sinh đúng tập ứng viên. Thuật toán Pincer-search sẽ phát sinh các tập phổ biến. Các tập phổ biến chắc chắn được phát hiện khi bottom-up phát hiện ra chúng. Các tập phổ biến sẽ chưa chắc là tập phổ biến trong kết quả khi top-down tìm thấy tập cha của chúng là phổ biến (top-down tìm thấy tập cha trước khi bottom-up tìm thấy các tập con). Hơn nữa, chỉ có các tập phổ biến tối đại mới được thêm vào MFS (dòng 5).Vì vậy, Pincer-search phát sinh tất cả các tập phổ biến tối đại.

4.1.5.6 Đếm số support của tất cả các tập phổ biến

Trong những trường hợp cần đếm số hỗ trợ của các tập phổ biến, ta có thể xây dựng một *complete hash-tree* cho tập phổ biến tối đại. Trong thuật toán Apriori, mỗi nút lá là một tập duy nhất. Không như các hash-tree khác chỉ chứa các tập có độ dài bằng nhau ở các nút lá, *complete hash-tree* chứa tất cả tập con của tập phổ biến tối đại (có độ dài khác nhau).

Tiến trình đếm số hỗ trợ chỉ cần một thay đổi nhỏ, chúng ta thêm một trường để lưu trữ số hỗ trợ của bản thân các nút. Số hỗ trợ trong mỗi nút sẽ tăng lên mỗi khi duyệt đến nút đó theo chiều tiến (Apriori chỉ tăng đối với các nút lá). Để tăng số hỗ trợ của các ứng viên được hỗ trợ bởi một giao tác (transaction) , tất cả các mối liên hệ của các phần tử (item) trong một giao tác đều được xem xét đến.

Ví dụ minh họa cách thức hoạt động của tiến trình đếm số hỗ trợ:



Hình 10: Đếm số hỗ trợ của các tập phổ biến

Giả sử tập phổ biến tối đại là {{1.2.3}, {2.3.4}}. Cơ sở dữ liệu chứa 3 giao tác: {1,2,3,5}, {1,2,3,4}, {2,3,4}. Tất cả các mối liên hệ của các phần tử trong mỗi giao tác đều được liệt kê tăng dần theo thứ tự tự điển và ta sẽ duyệt cây theo thứ tự này. Nếu mỗi phần tử trong giao tác không thuộc về cây, ngừng và thử tiếp theo thứ tự. Nói cách khác, cây được duyệt theo chiều sâu. Số hỗ trợ được tăng lên 1 đơn vị mỗi khi duyệt đến 1 nút theo chiều tiến (nói cách khác, ta không tăng nó khi quay lui).

Thứ tự tự điển cho giao tác {1,2,3,5} là 1, 12, 123, 1235, 125, 13, 135, 2, 23, 235, 25, 3, và 35. Điều này có nghĩa là đường tắt của cây là nút {1}, {2}, {3}. Từ nút {3} không có nhánh cho 5, vì vậy quay lui trở về nút {2}. Từ nút {2} cũng không có nhánh cho 5, vì vậy quay về {1}. Sau đó xét tiếp nhánh 3, quay lui về gốc, sau đó đến nút {2}, cứ thế tiếp tục. Tương tự, thứ tự tự điển của giao tác {1,2,3,4} là 1, 12, 123, 1234, 124,13, 134, 2, 23, 234, 24, 3, 34, và 4. Đối với giao tác {2,3,4} là 2, 23, 234, 24, 3, 34, 4.

Số hỗ trợ của tất cả các tập phổ biến được đếm hiệu quả hơn vì chỉ cần đọc cơ sở dữ liệu 1 lần duy nhất.

4.2 Nhận xét và sử dụng các hướng tiếp cận:

4.2.1 Hướng tiếp cận phân lớp:

Sau khi khảo sát hai hướng tiếp cận, ta nhận thấy hướng tiếp cận phân lớp cũng tương tự như hướng xác định luật kết hợp trong các điều kiện sau:

- Các dữ liệu quan sát được khảo sát lần lượt.
- Độ tin cậy của luật bằng 1.

Các luật tìm ra sẽ phân lớp một cách chính xác trên dữ liệu quan sát: điều này cũng gây ra hạn chế khi rút luật:

- Kết quả nhạy với nhiễu: nếu một trong số các mẫu dữ liệu là không chính xác thì các luật rút ra được sẽ sai lệch so với các luật thật sự.

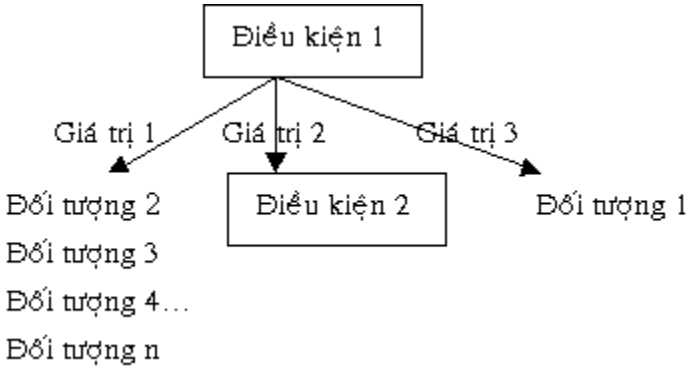
Ví dụ: với tập dữ liệu như sau

	Điều kiện 1	Điều kiện 2	…	Kết luận
Đối tượng 1	0(nhiễu)	0	…	1
Đối tượng 2	1	0	…	1
Đối tượng 3	1	0	…	1

Nếu *không bị nhiễu*, luật "if DieuKien1 then KetLuan" sẽ được rút ra. Nhưng do xuất hiện một thông tin nhiễu nên luật không thể được rút ra

- Các luật kết quả mang tính chi tiết cao làm giảm tính tổng quát của chúng: ngược với điều chúng ta mong muốn là thu được những luật phổ biến để áp dụng trên các đối tượng sau này.

Ví dụ:



Có 2 luật được rút ra:

If DieuKien1=GiaTri3 then KetQua=True (luật 1)

If DieuKien1=GiaTri1 then KetQua=False (luật 2)

Luật 1 chỉ cá biệt trên đối tượng 1, trong khi luật 2 phổ biến trên nhiều đối tượng hơn. Tuy nhiên hai luật này lại có độ ưu tiên như nhau.

4.2.2 Hướng tiếp cận theo độ phổ biến và luật kết hợp:

Hướng tiếp cận này cũng gặp phải một số hạn chế khi áp dụng như sau:

- Các dữ liệu phải đạt được một độ phổ biến trước khi có thể được sử dụng tìm luật kết hợp: nếu trong một không gian các sự kiện lớn, trong đó ở mỗi đối tượng chỉ xuất hiện một số sự kiện và khảo sát trên nhiều đối tượng với phân bố xác suất đều, thì phần lớn các sự kiện sẽ không đạt được ngưỡng phổ biến cần thiết. Và vì thế, chúng ta đã bỏ qua một số lượng tri thức.
- Bùng nổ dữ liệu không cần thiết: không phải tất cả các tập phổ biến tìm được đều là luật kết hợp nên quá trình tìm kiếm trên các dữ liệu này sẽ gây lãng phí tài nguyên. Điều này xảy ra do chúng ta tìm kiếm trên không gian tất cả các sự kiện mà không hướng đến kết quả (như bên phân lớp).

	Điều kiện 1	Điều kiện 2	Điều kiện 3	…	Kết luận
Đối tượng 1				…	
…				…	

Đối tượng n				…	

Độ tin cậy của *kết luận* Y vào *điều kiện* X :

$$Con(X \rightarrow Y) = \frac{Sup(\{X,Y\})}{Sup(\{X\})}$$

Chương trình phải duyệt qua tất cả các tổ hợp 2 phần tử được phát sinh là:

{Điều kiện 1, Điều kiện 2}(1),{Điều kiện 1, Điều kiện 3}(2),{Điều kiện 2, Điều kiện 3}(3), {Điều kiện 1, kết luận}(4),{Điều kiện 2, kết luận}(5),{Điều kiện 3, kết luận}(6).

Trong đó chỉ có 3 tổ hợp 4,5,6 là có nghĩa, 3 tổ hợp 1,2,3 gây lãng phí tài nguyên.

4.2.3 Áp dụng để giải quyết bài toán khai thác dữ liệu

4.2.3.1 Bài toán

Cho một cơ dữ liệu người dùng bao gồm các thông tin về tứ trụ và các sự kiện về người đó cùng với một cơ sở tri thức áp dụng cho dự đoán. Hãy tìm và bổ sung thêm các luật sinh từ cơ sở dữ liệu và cơ sở tri thức nói trên.

4.2.3.2 Phương hướng giải quyết bài toán.

Kết hợp giữa hai hướng tiếp cận vừa nêu, ta sẽ đi tìm các luật phân lớp trên các sự kiện kết quả dựa trên một độ phổ biến và một độ tin cậy cho trước.

Bước đầu tiên, tương tự như thuật giải phân lớp, với mỗi sự kiện kết quả Y, ta chia cơ sở dữ liệu thành các lớp dựa trên giá trị của kết quả. Dữ liệu của ta bao gồm N dòng (với N là số người dùng), nội dung các cột dữ liệu là thuộc tính xác định các thuộc tính của lá số, số lượng và nội dung các cột do hệ cơ sở tri thức qui định.

Với giá trị của kết luận Y là tập {true; false}, cho biết người dùng có sự kiện tương ứng hay không, ta chia dữ liệu thành 2 lớp: lớp tDB và fDB (các lớp dữ liệu này chỉ bao gồm dữ liệu điều kiện).

Bước thứ hai ta xác định các tập phổ biến trong tBD dựa trên một ngưỡng cho trước.

Lưu ý là trong tập tDB, khi kết luận Y là true, mỗi tập phổ biến X tương ứng với một tập {X,Y} trong toàn bộ cơ sở dữ liệu. Tuy nhiên ngưỡng phổ biến mà ta dùng để xác định tập X là cục bộ, bên trong tập tDB, vì thế tập X được tìm thấy chỉ phụ thuộc vào độ phổ biến của {X,Y}/{Y} mà không cần quan tâm đến độ phổ biến của tập {Y} và do đó ta tránh được khuyết điểm 1 đã nêu ở phần trước.

Mặt khác, ta tìm các tập phổ biến {X} khi đã xác định được giá trị của Y nên cũng tránh được việc tìm kiếm các tập phổ biến không cần thiết như đã nêu ở nhận xét thứ 2.

Để thực hiện bước này, ta chấp nhận một tiền đề là nếu tồn tại một luật kết hợp $X \rightarrow Y$ thật sự thì tập thuộc tính {X} phải đạt một ngưỡng hỗ trợ nào đó khi đã biết được Y.

Bước thứ 3, xác định độ tin cậy của luật

Gọi a là số hỗ trợ của X trong tDB, b là số hỗ trợ của X trong fDB. Công thức xác định độ tin cậy:

$$Con(X \rightarrow Y) = \frac{\rho(X \cap Y)}{\rho(X)} = \frac{\rho(X \cap Y)}{\rho(X \cap Y) + \rho(X \cap \overline{Y})} = \frac{a}{a+b}$$

Bổ sung luật: các luật sau khi xác định độ tin cậy sẽ được giữ lại nếu tập X có độ tin cậy lớn hơn một ngưỡng cho trước và các luật được sắp xếp dựa trên độ tin cậy của mình.

Loại bỏ luật: các luật khi được bổ sung vào cơ sở tri thức sẽ có quan hệ nối rời với nhau, mặc khác, nội dung điều kiện bên trong mỗi luật có quan hệ nối liền, vì thế ta có thể xóa bớt các luật dựa trên định luật hút.

Định luật hút: $(a \wedge b) \vee a = a$

$(a \vee b) \wedge a = a$

Hai điều kiện A, B nếu có một quan hệ thuộc về sẽ được loại bớt một luật và chỉ giữ lại luật nào có độ tin cậy cao hơn.

Chương 5: Xây dựng chương trình

Chương trình được viết bằng ngôn ngữ lập trình C# trên môi trường .NET FrameWork 1.1.

5.1 Động cơ suy diễn

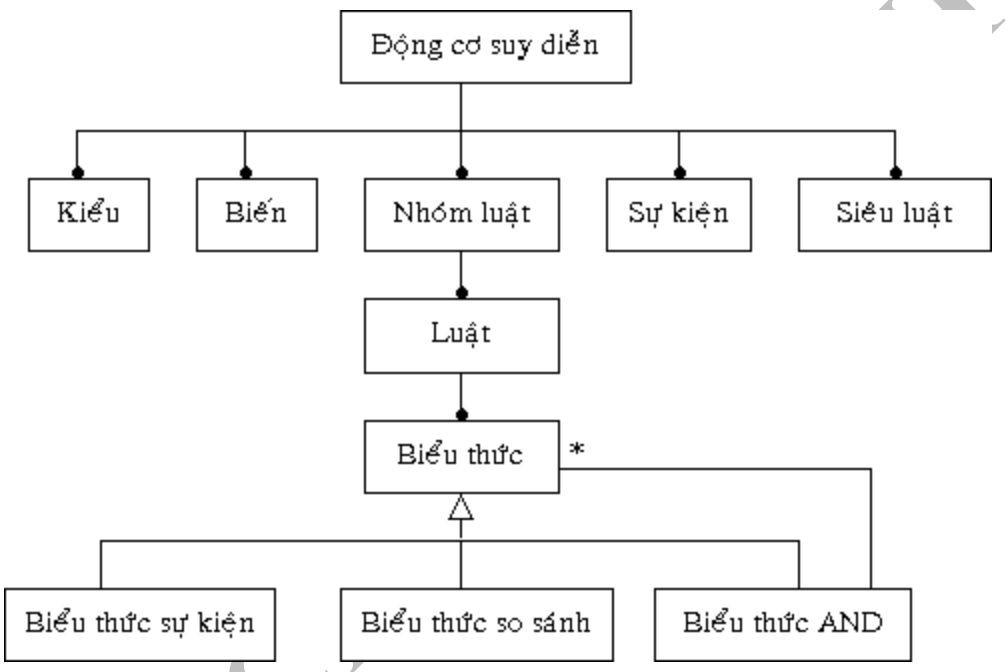
Động cơ suy diễn có các chức năng chính như sau:

- *Chức năng giao tiếp người dùng*: cho phép người dùng
  - nhập giá trị cho các biến dữ liệu (đã khai báo bên trong cơ sở tri thức)
  - đặt giá trị cho các sự kiện
  - lấy giá trị của các biến dữ liệu
  - lấy giá trị (cũng như phần định nghĩa) của các sự kiện
- *Chức năng biên dịch cơ sở tri thức*: như đã đề cập ở trên, hệ chuyên gia cung cấp cho người dùng một cú pháp để biểu diễn tri thức tương tự như một ngôn ngữ lập trình hướng khai báo. Vì thế, hệ chuyên gia có chức năng phân tích nội dung của một cơ sở tri thức (được khai báo dưới dạng văn bản thô) để thiết lập tập sự kiện, tập luật, các biến dữ liệu và các thông tin có liên quan.
- *Chức năng suy diễn*: để rút ra các tri thức dự đoán dựa trên các thông tin đưa vào, hệ chuyên gia thực hiện quá trình suy diễn tiến. Trong quá trình suy diễn, hệ thực hiện các hành động sau:
  - Tìm kiếm các biến dữ liệu và phân giải các luật lặp
  - Kiểm tra các biểu thức điều kiện (và cả kết quả) dựa trên tập sự kiện và giá trị của các biến dữ liệu hiện tại
  - Thực hiện các biểu thức kết quả nếu luật thỏa.

Các thành phần của hệ cơ sở tri thức:

- Sự kiện và mô tả.
- Luật và nhóm luật.
- Biến dữ liệu và kiểu dữ liệu.
- Siêu luật.

5.1.1 Sơ đồ các lớp chính của động cơ:



Hình 11: Sơ đồ các lớp chính của động cơ

Lớp Kieu:

```
class Kieu
{
      private string tenKieu;       // tên của kiểu
      private ArrayList arrayTruong;// danh sách các trường
}
```

Lớp Bien

```
class Bien
{
      private string ten;           // tên biến
      private Kieu kieu;            // kiểu của biến
```

```
      private string []duLieu;      // các giá trị dữ liệu của biến
      private bool []f;       // cho biết 1 trường đã có giá trị chưa

      public void PhanTich(Nguon nguon, ArrayList mk);
            //đọc và cấp phát bộ nhớ cho dữ liệu
      public bool LayGiaTri(string truong, out string giatri)
            // lấy giá trị (kiểu string) theo tên trường
      public bool GanGiaTri(string truong, ref string giatri)
            // gán giá trị cho 1 trường trong biến theo tên trường
      public int LayViTri(ref string truong)
            // tính vị trí tương đối của 1 trường trong biến
}
```

Lớp SuKien

```
public class SuKien
{
      static private DongCoSuyDien dongCo;   // động cơ suy diễn
      static private string kieuHt; // kiểu sự kiện hiện tại
      private string tenSuKien;     //tên của sự kiện
      private string dinhNghia;     // nội dung sự kiện
      private int giaTri;           // giá trị của SK (1: có,0: không)
      private LoaiSK loai;    // loại sự kiện (BanDau hoac PhatSinh)
      private string kieu;    //kiểu sự kiện (Tien_de, Dieu_kien,
                              Ket_qua)

      public void PhanTich(Nguon nguon);        //đọc và đưa vào tênSk,
                                                dinhNghia
}
```

Lớp SieuLuat

```
public class SieuLuat
{
      private string luatTruoc;     // luật đứng trước
      private string luatSau;       // luật đứng sau

      public void PhanTich(Nguon nguon); //đọc và tạo luatTruoc,luatSau
}
```

Lớp BieuThuc (lớp trừu tượng)

```
abstract class BieuThuc
{
      public abstract void PhanTich(Nguon nguon);
      public abstract bool LuongGia();
      public abstract bool LuongGiaPhai();
      public abstract void ThucHien();
      public abstract bool LayThamBien(ref string tenLap);
      public abstract void ThayTheBien(string goc, string thayThe);
      public abstract KBieuThuc LoaiBieuThuc();
            // loại BieuThucAnd, BieuThucSoSanh, BieuThucSuKien
      private static DongCoSuyDien dongCo;
           //động cơ suy diễn
      public static BieuThuc TaoBieuThuc(BieuThuc bt)
            // trả ra 1 BieuThuc tùy theo bt là loại biểu thức nào
}
```

Lớp BieuThucSuKien

```
class BieuThucSuKien : Du_Doan.BieuThuc
{
      private string ten;           // tên biểu thức
      private bool giaTri;          // giá trị biểu thức

      public override void PhanTich(Nguon nguon)
            //đọc và lấy tên sự kiện, không có "not": giaTri= 1
      public override bool LuongGia()
            // lượng giá biểu thức sự kiện
      public override bool LuongGiaPhai()
            // trả về not(LượngGia)
      public override void ThucHien()
            // thêm sự kiện vào mảng các sự kiện
      public override bool LayThamBien(ref string tenLap);
            // lấy tên tham biến
      public override void ThayTheBien(string goc, string t)
            // thay thế t cho goc trong tên biểu thức
}
```

Lớp BieuThucSoSanh

```
class BieuThucSoSanh : Du_Doan.BieuThuc
{
      private string veTrai;  //vế trái của biểu thức
      private string vePhai;  // vế phải của biểu thức
      private string toanTu;  // tóan tử trong biểu thức so sánh

      public override void PhanTich(Nguon nguon)
            // đọc và tạo các biến
      public override bool LuongGia()
            // lượng giá biểu thức
      public override bool LuongGiaPhai()
            // trả về not(LuongGia)
      public override void ThucHien()
            // gán giá trị cho vế phải
}
```

Lớp BieuThucAnd

```
public class BieuThucAnd : Du_Doan.BieuThuc
{
      private ArrayList cacBieuThuc; // mảng các biểu thức cần AND
      public override void PhanTich(Nguon nguon)
            // đọc và thêm các biểu thức vào mảng
      public override bool LuongGia()
            // lượng giá
      public override void ThucHien()
            // gọi ThucHien của các biểu thức trong mảng theo thứ tự
      public override bool LayThamBien(ref string tenLap)
            // gọi LayThamBien của các biểu thức trong mảng
      public override void ThayTheBien(string goc, string thayThe)
            // gọi ThayTheBien của các biểu thức trong mảng
}
```

Lớp Luat

```
class Luat
{
      private BieuThuc veTrai;                  // biểu thức if …
      private BieuThuc vePhai;                  // biểu thức then …
```

```
      private static DongCoSuyDien dongCo;      // động cơ suy diễn
      public void PhanTich(Nguon nguon)         //  đọc và tạo các biểu
                                          thức ở vế trái và vế phải
      public bool SuyDien()// thay thế,lượng giá 2 vế và thực hiện luật
}
```

Lớp NhomLuat

```
class NhomLuat
{
      private string ten;     //tên nhóm luật
      private CacLuat luat;   // mảng các luật cùng nhóm
      private SuKien sk;      // sự kiện mặc định, nếu không có luật
                              // nào thỏa thì dùng sự kiện
}
```

Lớp DongCoSuyDien

```
class DongCoSuyDien
{
      private Nguon nguon;          // nguồn cơ sở tri thức
      private string curKeyWord;    // từ khóa hiện tại
      private int curType;          // kiểu hiện tại

      private CacSuKien cacSuKien;  // mảng các sự kiện
      private CacKieu cacKieu;      // mảng các kiểu
      private CacBien cacBien;      // mảng các biến
      private CacNhomLuat cacNhomLuat;    // mảng các nhóm luật
      private Cache bienLap;        // lưu lại các biến lặp đã dùng
      private CacSieuLuat cacSieuLuat;    // mảng các siêu luật
      public void DocCoSoTriThuc(string fn)     // lấy dữ liệu từ file
                              chứa cơ sở tri thức và gọi hàm PhanTich()
      public void PhanTich()              //Hàm phân tích chính cho
                                          động cơ
      public void TaoDoiTuong()           //tùy theo curType, giao cho
                                          các đối tượng thực hiện hàm
                                          PhanTich() tương ứng với từng
                                          đối tượng
      public void SuyDien()               //các nhóm luật suy diễn theo
                                          thứ tự trong các siêu luật
```

```
      public ADuLieu LapBangDieuKien(UserList danhsach, bool gt, ref
            int val)                // lập bảng dữ liệu cho quá trình
            mining
      public void RefreshAll()//làm mới lại các sự kiện và các biến
}
```

5.1.2 Cú pháp khai báo hệ cơ sở tri thức:

5.1.2.1 Khai báo sự kiện:

5.1.2.1.1 Khai báo giá trị cho sự kiện:

Cú pháp:

fact [not] Tên_sự_kiện; [not] Tên_sự_kiện(Tham_số_1, Tham_số_2, …);

Mục đích: khai báo tên và giá trị ban đầu cho sự kiện. Giá trị ở đây là giá trị ban đầu của sự kiện hay biến dữ liệu (và do đó đây là giá trị của sự kiện hay biến dữ liệu trong suốt quá trình suy diễn). Chúng ta sử dụng cú pháp trên để khai báo cho các sự kiện nguyên thủy ban đầu trong hệ cơ sở tri thức hay gán các giá trị định danh cho biến dữ liệu định danh.

Để khai báo các sự kiện quạn hệ 2 hay nhiều ngôi, ta đưa quan hệ ra làm tên của sự kiện và đưa các đối tượng quan hệ làm các tham số cho sự kiện đó, sử dụng cú pháp khai báo số 2.

Ví dụ: khi cần khai báo cho các sự kiện

- "Thủy sinh Mộc" ,
- "Thủy khắc Hỏa"
- …

ta thực hiện khai báo sau:

Sinh(Thủy, Mộc)

Khắc(Thủy,Hỏa)

…

Tương tự như thế cho các sự kiện nguyên thủy ban đầu.

5.1.2.1.2 Khai báo mô tả cho sự kiện:

Cú pháp

```
fact_define
        Tên_sự_kiện: Định nghĩa;
```

Mục đích: dùng để khai báo một định nghĩa cho một sự kiện. Mô tả trong trường hợp này là một câu có nghĩa (có sử dụng khoảng trắng và dấu tiếng Việt), chuỗi này dùng để giải thích cho sự kiện. Ta dùng cú pháp trên để khai báo các giải thích cho những sự kiện suy diễn được (những sự kiện kết luận), như thế, sau quá trình suy diễn ta sẽ thu được những câu giải thích cho việc dữ đoán của mình. Và chương trình sẽ lấy và hiển thị kết luận từ bên trong động cơ.

5.1.2.2 Khai báo kiểu dữ liệu:

Cú pháp

```
type
        Tên_kiểu= (Tên_Kiểu_1 Trường_1, Tên_Kiểu_2 Trường_2,…)
```

Mục đích: khai báo một kiểu có cấu trúc. Các kiểu bên trong có thể là một kiểu có cấu trúc khác.

Ví dụ:

- khai báo cho kiểu can (chứa thông tin về can của bát tự), các thông tin bao gồm tên của can, hành, nghi và thần của trụ.

  kcan= (symbol ten, symbol hanh, symbol nghi, symbol than)

- khai báo cho kiểu trụ: bao gồm 1 thiên can và 1 địa chi cho mỗi trụ

  kchi= (kcan can, kchi chi)

5.1.2.3 Khai báo biến dữ liệu

Cú pháp:

```
var
        Tên_kiểu Tên_biến;
```

Mục đích: khai báo 1 biến dữ liệu. Các biến dữ liệu có thể được gán (lấy) giá trị từ bên ngoài, hoặc được gán (lấy) giá trị bên trong động cơ trong quá trình suy diễn.

5.1.2.4 Khai báo luật:

Cú pháp:

```
rule
[group Tên_nhóm_luật]
        if      <Điều kiện>  then    <Kết quả>
```

Mục đích: khai báo 1 luật. <Điều kiện> và <Kết quả> là những biểu thức logic. Khi suy diễn, động cơ sẽ lượng giá điều kiện (và cả kết quả) của luật, nếu thỏa điều kiện thì biểu thức kết quả sẽ được thực hiện. Luật trên thỏa khi:

- <điều kiện> của luật là đúng
- và <kết luận> phải thỏa theo cách lượng giá đặc biệt trình bày bên dưới.

Biểu thức logic có thể có một trong các dạng sau:

- Dạng 1:

[not] Tên_sự_kiện hay

[not] Tên_sự_kiện(Tham_số_1, Tham_số_2…)

Các tham số có thể là ký hiệu hay là tên biến dữ liệu. Nếu là tên biến dữ liệu thì giá trị của dữ liệu sẽ được thay vào khi tính toán.

  - Đối với biểu thức điều kiện: khi lượng giá, động cơ sẽ tìm kiếm sự kiện trên trong tập sự kiện và lấy giá trị của sự kiện thay vào biểu

thức lượng giá. Nếu như động cơ không tìm thấy sự kiện trên thì giá trị mặc nhiên false sẽ được thay vào vị trí biểu thức (giả thiết về thế giới đóng).

- Đối với biểu thức kết quả:
  - kết luận của luật được xem là thỏa khi và chỉ khi sự kiện khai báo trong kết luận chưa được gán giá trị.
  - nếu luật thỏa, <kết luận> được thực hiện bằng cách gán giá trị cho sự kiện.

- Dạng 2: *Vế_trái Toán_tử Vế_phải;*
  - Vế trái, vế phải là những biểu thức symbol.
  - Toán tử bao gồm các phép so sánh {=,<>}
  - Biểu_thức_symbol có thể là một ký hiệu (symbol), một tên biến hay một biểu thức có chứa biến…
  - Đối với biểu thức điều kiện: điều kiện được lượng giá dựa trên phép so sánh giá trị giữa hai vế.
  - Đối với biểu thức kết quả: vế_trái phải là một tên biến, toán tử chỉ nhận phép so sánh =
    - kết luận chỉ được xem là thỏa khi biến dữ liệu chưa có giá trị (từ bên ngoài đưa vào hay trong quá trình suy diễn).
    - nếu luật thỏa, giá trị của vế phải được gán cho biến ở vế trái.

- Dạng 3: dạng nối liền

  Điều_kiện_1 and Điều_kiện_2 …
  - *Điều_kiện_i* là những biểu thức dạng 1 hay dạng 2, được nối với nhau bằng phép nối and.
  - Đối với biểu thức điều kiện: giá trị của biểu thức được tính toán dựa trên phép nối and giữa giá trị các vế.
  - Đối với biểu thức kết luận: nếu luật thỏa thì tất cả các biểu thức trong kết luận sẽ được thực hiện.

5.1.2.5 Khai báo các siêu luật

Cú pháp:

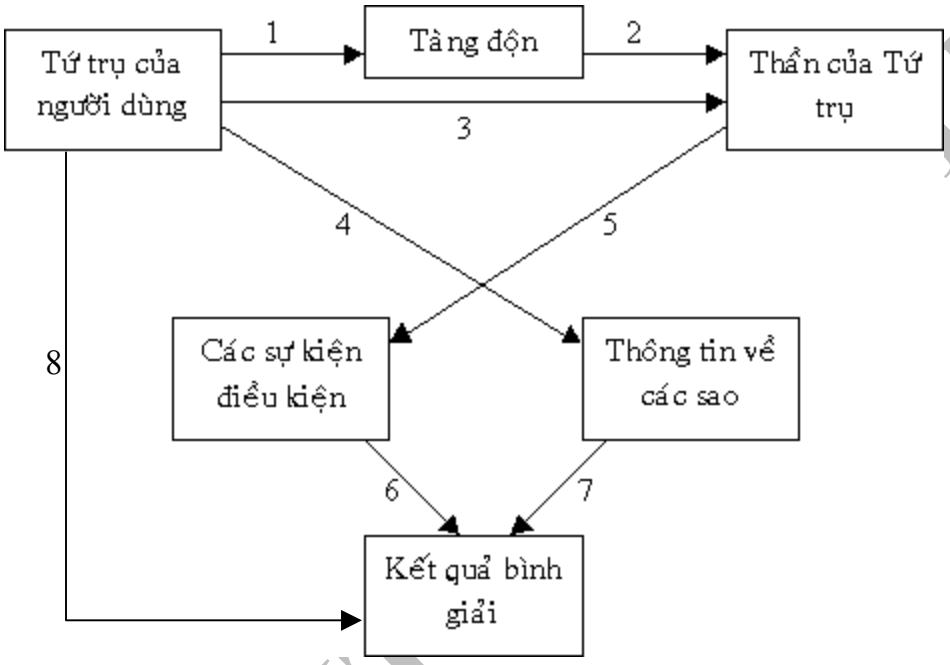
```
meta
refer Nhóm_luật_trước to Nhóm_luật_sau
```

5.1.3 Nội dung khai báo trong cơ sở tri thức:

- Các kiểu dữ liệu. Ví dụ:
  - ktru= (kcan can, kchi chi);
- Các biến dữ liệu: là nơi đưa các thông tin đầu vào để suy diễn. Ví dụ:
  - ktu_tru nam; // khai báo 1 biến để lưu thông tin về tứ trụ cho nam
- Các sự kiện và các luật tiền đề:
  - Sinh, khắc cũa ngũ hành. Ví dụ:
    - Khac(Kim, Moc);…
  - Tính sao trên trụ. Ví dụ:
    - DaoHoa(Ty,Ngo);
  - Các thần sinh, khắc nhau. Ví dụ:
    - ThanKhac (Thuong_quan, Thien_quan);
  - Các luật tính tàng độn. Ví dụ:
    - if (Sinh(?(ktu_tru X).?(ktru Y).?(kcan Z).hanh, ?X.ngay.can.hanh) and (?X.?Y.?Z.nghi = ?X.ngay.can.nghi)) then (?X.?Y.?Z.than= Thien_an);
- Các sự kiện điều kiện: dùng để đánh dấu các thuộc tính điều kiện xuất hiện trên lá số của người dùng.
  - Ví dụ: Nam_ChinhAn_Gio (Trụ giờ của nam có Chính_ấn)
- Các sự kiện kết luận cùng với định nghĩa: mô tả những lời bình giải cho lá số tương ứng.
  - Ví dụ: WChungThuy: vợ chung thủy;
- Các luật suy ra kết luận.

- Ví dụ: if nu.?(kcan X).than=Thien_quan and ThanKhac(nu.?(kcan Y).than,nu.?X.than) then WChungThuy;

5.1.4 Sơ đồ các khối tri thức suy diễn:



Hình 12: Sơ đồ các khối tri thức

(1): Các tri thức về tàng độn

(2)(3): Các tri thức về tính thần của tứ trụ

(4): Các tri thức tính sao

(5): Các tri thức điều kiện

(6)(7)(8): Các tri thức bình giải

5.1.5 Nội dung của cơ sở tri thức

- Các biến và kiểu dữ liệu

  Biến tứ trụ của hai người nam, nữ:

```
ktu_tru nam
```

ktu_tru nu

Các kiểu dữ liệu:

```
kcan= (symbol ten, symbol nghi, symbol hanh, symbol than);
ktang= (kcan tang1, kcan tang2, kcan tang3);
kchi= (symbol ten, symbol nghi, symbol hanh, ktang tang);
ktru= (kcan can, kchi chi);
ktu_tru= (ktru nam, ktru thang, ktru ngay, ktru gio);
```

- Các tri thức tiền đề
  - Các tri thức ngũ hành và sao

```
Sinh(Thuy, Moc);
Sinh(Moc, Hoa);
Sinh(Hoa, Tho);
Sinh(Tho, Kim);
Sinh(Kim, Thuy);

Khac(Thuy, Hoa);
Khac(Hoa, Kim);
Khac(Kim, Moc);
Khac(Moc, Tho);
Khac(Tho, Thuy);

LucHop(Ti,Suu);
LucHop(Ngo,Mui);
LucHop(Dan,Hoi);
LucHop(Mao,Tuat);
LucHop(Thin,Dau);
LucHop(Ty,Than);
```

```
LucXung(Ti,Ngo);
LucXung(Suu,Mui);
LucXung(Dan,Than);
LucXung(Mao,Dau);
LucXung(Thin,Tuat);
LucXung(Ty,Hoi);

DaoHoa(Dan,Mao);
DaoHoa(Ngo,Mao);
DaoHoa(Tuat,Mao);
DaoHoa(Than,Dau);
DaoHoa(Ti,Dau);
DaoHoa(Thin,Dau);
DaoHoa(Ty,Ngo);
DaoHoa(Dau,Ngo);
DaoHoa(Suu,Ngo);
DaoHoa(Hoi,Ti);
DaoHoa(Mao,Ti);
DaoHoa(Mui,Ti);

AmDuongLech(Binh,Ti);
AmDuongLech(Dinh,Suu);
AmDuongLech(Mau,Dan);
AmDuongLech(Tan,Mao);
AmDuongLech(Nham,Thin);
AmDuongLech(Quy,Ty);
AmDuongLech(Binh,Ngo);
AmDuongLech(Dinh,Mui);
```

```
AmDuongLech(Mau,Than);
AmDuongLech(Tan,Dau);
AmDuongLech(Nham,Tuat);
AmDuongLech(Quy,Hoi);

HungSatKinhDuong(Giap,Mao);
HungSatKinhDuong(At,Dan);
HungSatKinhDuong(Binh,Ngo);
HungSatKinhDuong(Dinh,Ty);
HungSatKinhDuong(Mau,Ngo);
HungSatKinhDuong(Ky,Ty);
HungSatKinhDuong(Canh,Dau);
HungSatKinhDuong(Tan,Than);
HungSatKinhDuong(Nham,Ti);
HungSatKinhDuong(Quy,Hoi);

TuongTinh(Ti,Ti);
TuongTinh(Suu,Dau);
TuongTinh(Dan,Ngo);
TuongTinh(Mao,Mao);
TuongTinh(Thin,Ti);
TuongTinh(Ty,Dau);
TuongTinh(Ngo,Ngo);
TuongTinh(Mui,Mao);
TuongTinh(Than,Ti);
TuongTinh(Dau,Dau);
TuongTinh(Tuat,Ngo);
TuongTinh(Hoi,Mao);
```

```
LocThan(Giap,Dan);
LocThan(At,Mao);
LocThan(Binh,Ty);
LocThan(Dinh,Ngo);
LocThan(Mau,Ty);
LocThan(Ky,Ngo);
LocThan(Canh,Than);
LocThan(Tan,Dau);
LocThan(Nham,Hoi);
LocThan(Quy,Ti);

TrachMa(Ti,Dan);
TrachMa(Suu,Hoi);
TrachMa(Dan,Than);
TrachMa(Mao,Ty);
TrachMa(Thin,Dan);
TrachMa(Ty,Hoi);
TrachMa(Ngo,Than);
TrachMa(Mui,Ty);
TrachMa(Than,Dan);
TrachMa(Dau,Hoi);
TrachMa(Tuat,Than);
TrachMa(Hoi,Ty);

ThienDuc(Ti,Ty);
ThienDuc(Suu,Canh);
ThienDuc(Dan,Dinh);
```

```
ThienDuc(Mao,Than);
ThienDuc(Thin,Nham);
ThienDuc(Ty,Tan);
ThienDuc(Ngo,Hoi);
ThienDuc(Mui,Giap);
ThienDuc(Than,Quy);
ThienDuc(Dau,Dan);
ThienDuc(Tuat,Binh);
ThienDuc(Hoi,At);

NguyetDuc(Ti,Nham);
NguyetDuc(Suu,Canh);
NguyetDuc(Dan,Binh);
NguyetDuc(Mao,Giap);
NguyetDuc(Thin,Nham);
NguyetDuc(Ty,Canh);
NguyetDuc(Ngo,Binh);
NguyetDuc(Mui,Giap);
NguyetDuc(Than,Nham);
NguyetDuc(Dau,Canh);
NguyetDuc(Tuat,Binh);
NguyetDuc(Hoi,Giap);

KiepSat(Ti,Ty);
KiepSat(Suu,Dan);
KiepSat(Dan,Hoi);
KiepSat(Mao,Than);
KiepSat(Thin,Ty);
```

```
KiepSat(Ty,Dan);
KiepSat(Ngo,Hoi);
KiepSat(Mui,Than);
KiepSat(Than,Ty);
KiepSat(Dau,Dan);
KiepSat(Tuat,Hoi);
KiepSat(Hoi,Than);

HoaCai(Ti,Thin);
HoaCai(Suu,Suu);
HoaCai(Dan,Tuat);
HoaCai(Mao,Mui);
HoaCai(Thin,Thin);
HoaCai(Ty,Suu);
HoaCai(Ngo,Tuat);
HoaCai(Mui,Mui);
HoaCai(Than,Thin);
HoaCai(Dau,Suu);
HoaCai(Tuat,Tuat);
HoaCai(Hoi,Mui);
KimThan(At,Suu);
KimThan(Quy,Dau);
KimThan(Ky,Ty);

ThanSinh(ChinhTai,ChinhQuan);
ThanSinh(ChinhTai,ThienQuan);
ThanSinh(ChinhTai,ThatSat);
ThanSinh(ThienTai,ChinhQuan);
```

```
ThanSinh(ThienTai,ThienQuan);
ThanSinh(ThienTai,ThatSat);

ThanSinh(ChinhQuan,ChinhAn);
ThanSinh(ChinhQuan,ThienAn);
ThanSinh(ThatSat,ChinhAn);
ThanSinh(ThatSat,ThienAn);

ThanKhac(ThatSat,ThuongQuan);
ThanKhac(ThatSat,ThucThan);
ThanKhac(Chinh_an,Thuong_quan);
ThanKhac(Thien_an,Thuong_quan);
ThanKhac(Kiep_tai,Chinh_tai);
ThanKhac(Ngang_vai,Chinh_tai);
ThanKhac(Kiep_tai,Thien_tai);
ThanKhac(Ngang_vai,Thien_tai);
ThanKhac(Chinh_tai,Chinh_an);
ThanKhac(Thien_tai,Chinh_an);
ThanKhac(Thuong_quan,Thien_quan);
ThanKhac(Thuc_than,Thien_quan);
```

- Các tri thức tính tàng độn:

```
if ?(kchi X).ten= Ti then ?X.tang.tang1.ten= Quy;
if ?(kchi X).ten= Suu then ?X.tang.tang1.ten= Ky and ?X.tang.tang2.ten= Tan and ?X.tang.tang2.ten= Quy ;
if ?(kchi X).ten= Dan then ?X.tang.tang1.ten= Giap and ?X.tang.tang2.ten= Binh and ?X.tang.tang3.ten= Mau;
if ?(kchi X).ten= Mao then ?X.tang.tang1.ten= At;
```

if ?(kchi X).ten= Thin then ?X.tang.tang1.ten= Mau and ?X.tang.tang2.ten= Quy and ?X.tang.tang3.ten= At;

if ?(kchi X).ten= Ty then ?X.tang.tang1.ten= Binh and ?X.tang.tang2.ten= Canh and ?X.tang.tang3.ten= Mau;

if ?(kchi X).ten= Ngo then ?X.tang.tang1.ten= Dinh and ?X.tang.tang2.ten= Ky;

if ?(kchi X).ten= Mui then ?X.tang.tang1.ten= Ky and ?X.tang.tang2.ten= At and ?X.tang.tang3.ten= Dinh;

if ?(kchi X).ten= Than then ?X.tang.tang1.ten= Canh and ?X.tang.tang2.ten= Nham and ?X.tang.tang3.ten= Mau;

if ?(kchi X).ten= Dau then ?X.tang.tang1.ten= Tan;

if ?(kchi X).ten= Tuat then ?X.tang.tang1.ten= Mau and ?X.tang.tang2.ten= Dinh and ?X.tang.tang3.ten= Tan;

if ?(kchi X).ten= Hoi then ?X.tang.tang1.ten= Nham and ?X.tang.tang2.ten= Giap;

- Các tri thức tính can chi, thần và sao

if (?(kcan X).ten = Giap) then (?X.nghi= Duong) and (?X.hanh= Moc);

if (?(kcan X).ten = At) then (?X.nghi= Am) and (?X.hanh= Moc);

if (?(kcan X).ten = Binh) then (?X.nghi= Duong) and (?X.hanh= Hoa);

if (?(kcan X).ten = Dinh) then (?X.nghi= Am) and (?X.hanh= Hoa);

if (?(kcan X).ten = Mau) then (?X.nghi= Duong) and (?X.hanh= Tho);

if (?(kcan X).ten = Ky) then (?X.nghi= Am) and (?X.hanh= Tho);

if (?(kcan X).ten = Canh) then (?X.nghi= Duong) and (?X.hanh= Kim);

if (?(kcan X).ten = Tan) then (?X.nghi= Am) and (?X.hanh= Kim);

if (?(kcan X).ten = Nham) then (?X.nghi= Duong) and (?X.hanh= Thuy);

if (?(kcan X).ten = Quy) then (?X.nghi= Am) and (?X.hanh= Thuy);

if (?(kchi X).ten = Ti) then (?X.nghi= Duong) and (?X.hanh= Thuy);

```
    if (?(kchi X).ten = Suu) then (?X.nghi= Am) and (?X.hanh= Tho);
    if (?(kchi X).ten = Dan) then (?X.nghi= Duong) and (?X.hanh= Moc);
    if (?(kchi X).ten = Mao) then (?X.nghi= Am) and (?X.hanh= Moc);
    if (?(kchi X).ten = Thin) then (?X.nghi= Duong) and (?X.hanh= Tho);
    if (?(kchi X).ten = Ty) then (?X.nghi= Am) and (?X.hanh= Hoa);
    if (?(kchi X).ten = Ngo) then (?X.nghi= Duong) and (?X.hanh= Hoa);
    if (?(kchi X).ten = Mui) then (?X.nghi= Am) and (?X.hanh= Tho);
    if (?(kchi X).ten = Than) then (?X.nghi= Duong) and (?X.hanh= Kim);
    if (?(kchi X).ten = Dau) then (?X.nghi= Am) and (?X.hanh= Kim);
    if (?(kchi X).ten = Tuat) then (?X.nghi= Duong) and (?X.hanh= Tho);
    if (?(kchi X).ten = Hoi) then (?X.nghi= Am) and (?X.hanh= Thuy);


    if (Khac(?(ktu_tru X).?(ktru Y).?(kcan Z).hanh, ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi = ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Thien_quan);
    if (Khac(?(ktu_tru X).?(ktru Y).?(kcan Z).hanh, ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi <> ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Chinh_quan);
    if (Sinh(?(ktu_tru X).?(ktru Y).?(kcan Z).hanh, ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi = ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Thien_an);
    if (Sinh(?(ktu_tru X).?(ktru Y).?(kcan Z).hanh, ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi <> ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Chinh_an);
    if (?(ktu_tru X).?(ktru Y).?(kcan Z).hanh = ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi = ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Ngang_vai);
```

```
    if (?(ktu_tru X).?(ktru Y).?(kcan Z).hanh = ?X.ngay.can.hanh) and
(?X.?Y.?Z.nghi <> ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Kiep_tai);
    if (Sinh(?(ktu_tru X).ngay.can.hanh,?X.?(ktru Y).?(kcan Z).hanh) and
(?X.?Y.?Z.nghi = ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Thuc_than);
    if (Sinh(?(ktu_tru X).ngay.can.hanh,?X.?(ktru Y).?(kcan Z).hanh) and
(?X.?Y.?Z.nghi <> ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Thuong_quan);
    if (Khac(?(ktu_tru X).ngay.can.hanh,?X.?(ktru Y).?(kcan Z).hanh) and
(?X.?Y.?Z.nghi = ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Thien_tai);
    if (Khac(?(ktu_tru X).ngay.can.hanh,?X.?(ktru Y).?(kcan Z).hanh) and
(?X.?Y.?Z.nghi <> ?X.ngay.can.nghi))
    then (?X.?Y.?Z.than= Chinh_tai);
```

- Các tri thức bình giải (trích):

```
facttype KetQua
define
        Nam_NgayLocMa:;
        Nam_GioLocMa:;
        Nu_NhatThienDuc:;
        Nu_NhatNguyetDuc:;
rule
group _TrungGian /*bắt buộc suy diễn trước tiên*/
        if LocThan(nam.ngay.can.ten,nam.ngay.chi.ten) then Nam_NgayLocMa;
        if TrachMa(nam.nam.chi.ten,nam.ngay.chi.ten) then Nam_NgayLocMa;
        if LocThan(nam.ngay.can.ten,nam.gio.chi.ten) then Nam_GioLocMa;
        if TrachMa(nam.nam.chi.ten,nam.gio.chi.ten) then Nam_GioLocMa;
```

```
        if TrachMa(nam.ngay.chi.ten,nam.gio.chi.ten) then Nam_GioLocMa;


        if ThienDuc(nu.thang.chi.ten,nu.ngay.chi.ten) then Nu_NhatThienDuc;
        if ThienDuc(nu.thang.chi.ten,nu.ngay.can.ten) then Nu_NhatThienDuc;
        if NguyetDuc(nu.thang.chi.ten,nu.ngay.chi.ten) then Nu_NhatNguyetDuc;
        if NguyetDuc(nu.thang.chi.ten,nu.ngay.can.ten) then Nu_NhatNguyetDuc;


/*Các sự kiện về Tiền tài 1*/
define
MVoPhatTai:;
MPhatTai:;
MHaoTai:;
WHuongChaMe:;
WKhongPhuc:;
WPhuc:;
WNgheo:;
CKhongLoi:;


CKhongLoinotMVoPhatTai:Vì Can chi ngày giống nhau nên Vợ chồng hao tổn tài;
MVoPhatTainotCKhongLoi:Vì Kiếp tài đóng ở ngày  hay  Chính tài, thiên tài đóng ở ngày lại không bị khắc pha  hay Hàm trì lâm nhật nên  Nam nhờ vợ mà phát tài;
CKhongLoiandMVoPhatTai:Vì Can chi ngày giống nhau   hay  Kiếp tài đóng ở ngày hay  Chính tài, thiên tài đóng ở ngày lại không bị khắc phá  hay  Hàm trì lâm nhật nên Hai vợ chồng làm ăn ban đầu khó phát đạt, hao tổn tài nhưng về sau chồng nhờ vợ mà phát tài;


MPhatTainotMHaoTai:Vì Thực thần nếu gặp Ấn nên Nam tiền tài ngày càng nhiều;
```

MHaoTainotMPhatTai:Vì Nam gặp ngang vai, kiếp tài hay Tứ trụ có kiếp tài, kình dương hay Trụ ngày có kình dương hay Trụ giờ thiên tài, tỉ kiếp nên Nam phá tài, phá tổ nghiệp, làm tổn hao tiền của;

MHaoTaiandMphatTai: Vì Nam gặp ngang vai, kiếp tài hay Thực thần nếu gặp Ấn hay Tứ trụ có kiếp tài, kình dương hay Trụ ngày có kình dương hay Trụ giờ thiên tài, tỉ kiếp nên Ban đầu nam phá tài, làm tổn hao tiền của nhưng về sau tiền tài ngày càng nhiều;

WPhucnotWNgheo:Vì Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ phúc đức ngày càng tăng;

WNgheonotWPhuc:Vì Khôngcó tài và không có ấn nên Nữ cuối đời cô đơn nghèo đói;

WNgheoandWPhuc: Vì Khôngcó tài và không có ấn hay Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ nếu không nhờ phúc đức dòng họ, sẽ cô đơn nghèo đói;

WHuongChaMenotWNgheo:Vì Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ hưởng của cải tiền bạc của cha mẹ;

WNgheonotWHuongChaMe:Vì Khôngcó tài và không có ấn nên Nữ cô đơn nghèo đói;

WNgheoandWHuongChaMe:Vì Khôngcó tài và không có ấn hay Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ hưởng của cải tiền bạc của cha mẹ, không biết gìn giữ cuối cùng sẽ cô đơn nghèo đói;

WKhongPhucnotWPhuc:Vì Thương Quan nên Nữ không hưởng được phúc;

WPhucnotWKhongPhuc: Vì Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ ngày càng tu nhân tích đức;

WPhucandWKhongPhuc:Vì Thương Quan hay Ấn thụ không gặp tài làm tổn thương, thiên đức nguyệt đức trên trụ ngày nên Nữ không hưởng được phúc của tổ tông nhưng biết tu thân nên phúc đức ngày càng tăng;

DefaultTienTai:Tiền tài trung bình;

rule

group _TienTai1

if (nam.ngay.can.ten=nu.ngay.can.ten) and (nam.ngay.chi.ten=nu.ngay.chi.ten) then CKhongLoi;

if Nam_KiepTai_Ngay then MVoPhatTai,

if DaoHoa(nam.nam.chi.ten,nam.ngay.chi.ten) then MVoPhatTai;

/**/ if nam.?(ktru X).?(kcan Y).than=Thuc_than and nam.?X.?(kcan Z).than=Chinh_an then MPhatTai;

if nam.?(ktru X).?(kcan Y).than=Thuc_than and nam.?X.?(kcan Z).than=Thien_an then MPhatTai;

if nu.?(kcan X).than=Chinh_an and not ThanKhac(nu.?(kcan Y).than,nu.?X.than) and Nu_NhatThienDuc then WHuongChaMe and WPhuc;

if nu.?(kcan X).than=Chinh_an and not ThanKhac(nu.?(kcan Y).than,nu.?X.than) and Nu_NhatNguyetDuc then WHuongChaMe and WPhuc;/**/

if nam.thang.?(kcan X).than=Ngang_vai and nam.?(kcan Y).than=Kiep_tai then MHaoTai and MTonHaiVo;

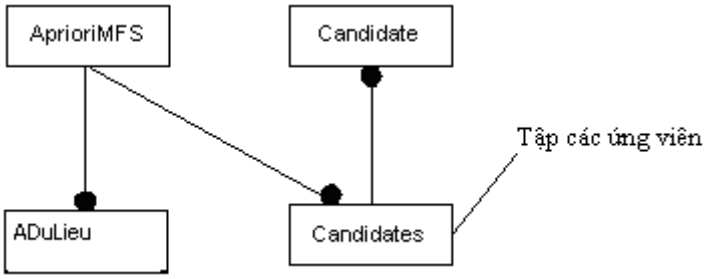
if nam.nam.?(kcan X).than=Ngang_vai and nam.?(kcan Y).than=Kiep_tai then MHaoTai;

if nam.gio.?(kcan X).than=Ngang_vai and nam.?(kcan Y).than=Kiep_tai then MHaoTai;

```
        if Nam_KiepTai and HungSatKinhDuong(nam.ngay.can.ten,nam.thang.chi.ten)
then MHaoTai;
        if Nam_KiepTai and HungSatKinhDuong(nam.ngay.can.ten,nam.gio.chi.ten)
then MHaoTai;
        if Nam_KiepTai and HungSatKinhDuong(nam.ngay.can.ten,nam.nam.chi.ten)
then MHaoTai;
        if HungSatKinhDuong(nam.ngay.can.ten,nam.ngay.chi.ten) then MHaoTai;
        if Nam_ThienTai_Gio and Nam_NgangVai_Gio then MHaoTai;
        if Nam_ThienTai_Gio and Nam_KiepTai_Gio then MHaoTai;
        if Nu_ThuongQuan then WKhongPhuc;
        if not (nu.?(kcan X).than = Chinh_tai) and not Nu_ThienTai and not
Nu_ChinhAn and not Nu_ThienAn then WNgheo; /**/

/*Mâu thuẫn tien tai*/
        if CKhongLoi and not MVoPhatTai then CKhongLoinotMVoPhatTai;
        if not CKhongLoi and MVoPhatTai then MVoPhatTainotCKhongLoi;
        if CKhongLoi and MVoPhatTai then CKhongLoiandMVoPhatTai;

        if MPhatTai and not MHaoTai then MPhatTainotMHaoTai;
        if MHaoTai and not MPhatTai then MHaoTainotMPhatTai;
        if  MHaoTai and MPhatTai then MHaoTaiandMphatTai;

        if WPhuc and WNgheo then WNgheoandWPhuc;

        if WHuongChaMe and not WNgheo then WHuongChaMenotWNgheo;
        if WHuongChaMe and WNgheo then WNgheoandWHuongChaMe;
```

```
if WKhongPhuc and not WPhuc then WKhongPhucnotWPhuc;
if WKhongPhuc and WPhuc then WPhucandWKhongPhuc;

if WPhuc and not WNgheo and not WKhongPhuc then WPhucnotWNgheo;
if not WHuongChaMe and WNgheo and not WPhuc then WNgheonotWHuongChaMe;
default DefaultTienTai;
```

5.2 Khai thác dữ liệu



Hình 13 Các lớp chính trong khai thác dữ liệu

Lớp Candidate

```
class Candidate
{
      public ArrayList SetOfItems;  // Tập thuộc tính của ứng viên
      public int NumOfFrequentSet;  // Đếm số lượng các giao tác được
            support của ứng viên này
      public int NumOfInfrequentSet;// Đếm số lượng các giao tác không
      được support của ứng viên này
}
```

Lớp Candidates

```
class Candidates
{
      public ArrayList SetOfCandidate// Tập các ứng viên
}
```

Lớp ADuLieu

```
class ADuLieu
{
      public ArrayList dong;        // mảng các dòng dữ liệu
      public ArrayList TenDieuKien; // mảng tên các sự kiện

      public void CountSup(ref Candidates Can)
      // đếm số hỗ trợ của các ứng viên trong tập Can
      public void CountCon(Candidates Can)
      // đếm độ tin cậy của các ứng viên trong tập Can
      public ArrayList LayKetQua(Candidates c, int ketQua)
      // tạo mảng các luật kết quả
      public string LayDieuKien(int n)
      // lấy tên điều kiện thứ n trong mảng điều kiện
}
```

Lớp AprioriMFS

```
class AprioriMFS
{
      private int PassK;            // bước k
      private Candidates Len;       // tập các tập phổ biến
      private Candidates Can;       // tập các ứng viên
      private Candidates MFS;       // tập các tập phổ biến cực đại
      private int NumOfItems;       // số lượng thuộc tính
      private double MinSup;  //ngưỡng MinSupport người dùng định nghĩa
      private double MinCon; //ngưỡng MinConf do người dùng định nghĩa

      public void AprioriSearchMFS(ADuLieu tDb, ADuLieu fDb)
      // hàm tìm MFS theo thuật toán Apriori
      public void Join()
      // hàm phát sinh ứng viên tạm thời
      public void Prune()
      // hàm bỏ các ứng viên không phổ biến
      public bool IsSubset(Candidate set1,Candidate set2)
      // cho biết set1 có là tập con của set2 hay không
}
```

Tổng kết

Khóa luận đã thực hiện được các mục tiêu đề ra ban đầu:

- Tìm hiểu và cài đặt một hệ chuyên gia
- Tìm hiểu và cài đặt thuật toán khai thác dữ liệu để khám phá các tri thức mới bổ sung trở lại vào quá trình dự đoán
- Tìm hiểu nguồn gốc, các quy luật cơ bản của Kinh Dịch, biểu diễn Kinh Dịch bằng logic mờ.
- Tìm hiểu học thuyết Tứ Trụ - một trong những phương pháp dự đoán của Kinh Dịch, cơ sở khoa học của Tứ Trụ
- Sử dụng phương pháp dự đoán hôn nhân theo Tứ Trụ làm ví dụ cho hoạt động của hệ chuyên gia.

Tuy nhiên, vẫn còn một số công việc khóa luận chưa kịp thực hiện vì nhiều lý do khách quan, xin dành lại cho các công trình khác sau này:

- Cài đặt thêm thuật toán khai thác dữ liệu cải tiến để tăng tốc độ
- Thu thập thêm dữ liệu cho quá trình mining
- Phương pháp dự đoán theo Tứ Trụ còn rất nhiều ứng dụng như dự đoán vận trình cả cuộc đời, công danh, tiền tài, con cái…

Phụ lục

Trong quá trình thực hiện đề tài, chúng tôi có gặp một số khó khăn trong việc xử lí liên quan đến ngày tháng âm lịch. Để tiện cho các bạn sau dễ dàng tìm kiếm và sử dụng, chúng tôi xin giới thiệu quy luật và cách tính âm lịch Việt Nam trong phần phụ lục này.

Quy luật của âm lịch Việt Nam

Âm lịch Việt Nam là một loại lịch thiên văn. Nó được tính toán dựa trên sự chuyển động của mặt trời, trái đất và mặt trăng. Ngày tháng âm lịch được tính dựa theo các nguyên tắc sau:

1. Ngày đầu tiên của tháng âm lịch là ngày chứa điểm Sóc
2. Một năm bình thường có 12 tháng âm lịch, một năm nhuận có 13 tháng âm lịch
3. Đông chí luôn rơi vào tháng 11 âm lịch
4. Trong một năm nhuận, nếu có 1 tháng không có Trung khí thì tháng đó là tháng nhuận. Nếu nhiều tháng trong năm nhuận đều không có Trung khí thì chỉ tháng đầu tiên sau Đông chí là tháng nhuận
5. Việc tính toán dựa trên kinh tuyến 105° đông.

Sóc là thời điểm hội diện, đó là khi trái đất, mặt trăng và mặt trời nằm trên một đường thẳng và mặt trăng nằm giữa trái đất và mặt trời. (Như thế góc giữa mặt trăng và mặt trời bằng 0 độ). Gọi là "hội diện" vì mặt trăng và mặt trời ở cùng một hướng đối với trái đất. Chu kỳ của điểm Sóc là khoảng 29,5 ngày. Ngày chứa điểm Sóc được gọi là ngày Sóc, và đó là ngày bắt đầu tháng âm lịch.

Trung khí là các điểm chia đường hoàng đạo thành 12 phần bằng nhau. Trong đó, bốn Trung khí giữa bốn mùa là đặc biệt nhất: Xuân phân (khoảng 20/3), Hạ chí (khoảng 22/6), Thu phân (khoảng 23/9) và Đông chí (khoảng 22/12).

Bởi vì dựa trên cả mặt trời và mặt trăng nên lịch Việt Nam không phải là thuần âm lịch mà là âm-dương-lịch. Theo các nguyên tắc trên, để tính ngày tháng âm lịch cho một năm bất kỳ chúng ta cần xác định những ngày nào trong năm chứa các thời điểm Sóc (New moon) và các thời điểm Trung khí (Major solar term). Một khi bạn đã tính được ngày Sóc, bạn đã biết được ngày bắt đầu và kết thúc của một tháng âm lịch: ngày mùng một của tháng âm lịch là ngày chứa điểm sóc.

Sau khi đã biết ngày bắt đầu/kết thúc các tháng âm lịch, bạn có thể xác định tên các tháng và tìm tháng nhuận theo cách sau. Đông chí luôn rơi vào tháng 11 của năm âm lịch. Bởi vậy chúng ta cần tính 2 điểm sóc: Sóc A ngay trước ngày Đông chí thứ nhất và Sóc B ngay trước ngày Đông chí thứ hai. Nếu khoảng cách giữa A và B là dưới 365 ngày thì năm âm lịch có 12 tháng, và những tháng đó có tên là: tháng 11, tháng 12, tháng 1, tháng 2, …, tháng 10. Ngược lại, nếu khoảng cách giữa hai sóc A và B là trên 365 ngày thì năm âm lịch này là năm nhuận, và chúng ta cần tìm xem đâu là tháng nhuận. Để làm việc này ta xem xét tất cả các tháng giữa A và B, tháng đầu tiên không chứa Trung khí sau ngày Đông chí thứ nhất là tháng nhuận. Tháng đó sẽ được mang tên của tháng trước nó kèm chữ "nhuận".

Khi tính ngày Sóc và ngày chứa Trung khí bạn cần lưu ý xem xét chính xác múi giờ. Đây là lý do tại sao có một vài điểm khác nhau giữa lịch Việt Nam và lịch Trung Quốc.Ví dụ, nếu bạn biết thời điểm hội diện là vào lúc yyyy-02-18 16:24:45 GMT thì ngày Sóc của lịch Việt Nam là 18 tháng 2, bởi vì 16:24:45 GMT là 23:24:45 cùng ngày, giờ Hà nội (GMT+7, kinh tuyến 105° đông). Tuy nhiên theo giờ Bắc Kinh (GMT+8, kinh tuyến 120° đông) thì Sóc là lúc 00:24:45

ngày yyyy-02-19, do đó tháng âm lịch của Trung Quốc lại bắt đầu ngày yyyy-02-19, chậm hơn lịch Việt Nam 1 ngày.

Ví dụ 1: Âm lịch năm 1984

Chúng ta áp dụng quy luật trên để tính âm lịch Việt nam năm 1984.

- Sóc A (ngay trước Đông chí năm 1983) rơi vào ngày 4/12/1983, Sóc B (ngay trước Đông chí năm 1984) vào ngày 23/11/1984.
- Giữa A và B là khoảng 355 ngày, như thế năm âm lịch 1984 là năm thường. Tháng 11 âm lịch của năm trước kéo dài từ 4/12/1983 đến 2/01/1984, tháng 12 âm từ 3/1/1984 đến 1/2/1984, tháng Giêng từ 2/2/1984 đến 1/3/1984 v.v.

Ví dụ 2: Âm lịch năm 2004

- Sóc A - điểm sóc cuối cùng trước Đông chí 2003 - rơi vào ngày 23/11/2003. Sóc B (ngay trước Đông chí năm 2004) rơi vào ngày 12/12/2004.
- Giữa 2 ngày này là khoảng 385 ngày, như vậy năm âm lịch 2004 là năm nhuận. Tháng 11 âm của năm 2003 bắt đầu vào ngày chứa Sóc A, tức ngày 23/11/2003.
- Tháng âm lịch đầu tiên sau đó mà không chứa Trung khí là tháng từ 21/3/2004 đến 18/4/2004 (Xuân phân rơi vào 20/3/2004, còn Cốc vũ là 19/4/2004). Như thế tháng ấy là tháng nhuận.
- Từ 23/11/2003 đến 21/3/2004 là khoảng 120 ngày, tức 4 tháng âm lịch: tháng 11, 12, 1 và 2. Như vậy năm 2004 có tháng 2 nhuận.

Một số công thức hỗ trợ

Để tiện cho việc tính toán chúng ta sử dụng 2 hàm hỗ trợ để tính số ngày đã trôi qua kể từ một điểm gốc qui ước và ngược lại. Trong tính toán thiên văn người ta thường lấy ngày 1/1/4713 trước công nguyên của lịch Julius (tức ngày 24/11/4714 trước CN theo lịch Gregory) làm điểm gốc. Số ngày tính từ điểm gốc này gọi là ngày Julius (Julian day hoặc Julian day number) của một ngày. Ví dụ, số ngày Julius của 1/1/2000 là 2451545.

Trong các công thức sau, phép chia a/b được hiểu như sau. Nếu cả a và b là số nguyên (int) thì / là chia số nguyên, bỏ phần dư. Ví dụ: 7 / 12 = 0; -5 / 12 = 0; -13 / 12 = -1. Nếu ít nhất một trong hai số là số thực (real, float, double) thì / là phép chia số thực, ví dụ: 7.25 / 2.4 = 3.0208333333333335. Chú ý: khi viết 7.0 ta hiểu là tính toán với số thực 7.0, còn nếu chỉ viết 7 sẽ tính với số nguyên 7.

Ngày dương lịch được tính theo lịch Gregory (Gregorian calendar), kể cả cho các năm trước 1582 khi lịch này được đưa vào sử dụng.

Đổi ngày dương lịch ra số ngày Julius

Viết ngày dương lịch dưới dạng d/m/y (ngày = d, tháng = m, năm = y). d là một số nguyên từ 1 đến 31, m từ 1 đến 12 và y là một số nguyên lớn hơn -4712 (âm 4712).

Gregorian2JD(int d, int m, int y)

```
1461 * ( y + 4800 + ( m - 14 ) / 12 ) ) / 4 +
( 367 * ( m - 2 - 12 * ( ( m - 14 ) / 12 ) ) ) / 12 -
( 3 * ( ( y + 4900 + ( m - 14 ) / 12 ) / 100 ) ) / 4 +
d - 32075
```

Đổi số ngày Julius ra ngày dương lịch

Cho jd là một số nguyên dương. Công thức sau đổi ngày Julius jd ra ngày dương lịch m/d/y:

JD2Gregorian(int jd)

```
l = jd + 68569;
n = ( 4 * l ) / 146097;
l = l - ( 146097 * n + 3 ) / 4;
i = ( 4000 * ( l + 1 ) ) / 1461001;
l = l - ( 1461 * i ) / 4 + 31;
j = ( 80 * l ) / 2447;
d = l - ( 2447 * j ) / 80;
l = j / 11;
m = j + 2 - ( 12 * l );
y = 100 * ( n - 49 ) + i + l;
```

Giải phương trình Kepler

Để tính vị trí thực của một vật thể quay trên một quĩ đạo hình ê-líp với độ lệch tâm ecc và vị trí trung bình mean, ta cần giải phương trình Kepler. Trong thuật toán sau, PI = 3.14159265358979323846, sqrt(x) là căn bậc hai của x, abs(x) là giá trị tuyệt đối của x, còn sin, cos, tan, atan và atan2 là các hàm lượng giác quen thuộc.

kepler(double mean, double ecc)

```
double delta;
double E = mean;
do {
  delta = E - ecc * sin(E) - mean;
  E = E - delta / (1 - ecc * cos(E));
} while (abs(delta) > 0.0001);
return 2.0 * atan(tan(E / 2) * sqrt((1 + ecc) / (1 - ecc)));
```

Tính vị trí mặt trời tại một thời điểm

Thuật toán sau cho phép tính vị trí của mặt trời trên quĩ đạo của nó vào lúc 0h sáng ngày d/m/y dương lịch tại múi giờ tz. (tz là khoảng cách tính bằng giờ giữa giờ địa phương và giờ GMT.) Giờ Việt Nam trước GMT 7h, như thế tz = 7.0. Tại Nhật Bản lấy tz = 9.0, còn tại California lấy tz = -8.0. Kết quả mà thuật toán

trả lại là một góc Rađian giữa 0 và 2*PI. Vị trí mặt trời được sử dụng để chia năm thời tiết thành 24 Khí (12 Tiết khí và 12 Trung khí). Một Trung khí là thời điểm mà kinh độ mặt trời có một trong những giá trị sau: 0*PI/6, 1*PI/6, 2*PI/6, 3*PI/6, 4*PI/6, 5*PI/6, 6*PI/6, 7*PI/6, 8*PI/6, 9*PI/6, 10*PI/6, 11*PI/6. Các điểm "Phân-Chí" được định nghĩa bằng các tọa độ sau: Xuân phân: 0, Hạ chí: PI/2, Thu phân: PI, Đông chí: 3*PI/2.

Trong các công thức sau, fixAngle là hàm dùng để chuẩn hoá một góc Rađian về khoảng từ 0 đến 2*PI: fixAngle(x) = x - PI*2*floor(x/(PI*2)), trong đó floor(x) là số nguyên lớn nhất không lớn hơn x. Ví dụ: fixAngle(7.5*PI) = 7.5*PI - 2*PI*3 = 1.5*PI

SunLongitude(int d, int m, int y, double tz)

```
int x = toJD(d, m, y);
double x2 = x - 2447892 - tz / 24.0;
double z = fixAngle(PI * 2 * x2 / 365.242191);
double angle = fixAngle(z - 0.05873240627141918);
angle = kepler(angle, 0.016713) + 4.935239984568769;
return fixAngle(angle);
```

Tính tuổi trăng tại một thời điểm

Tuổi trăng được hiểu là góc giữa mặt trăng và mặt trời (nhìn từ trái đất) tại một thời điểm. Thuật toán sau tính tuổi trăng vào lúc 0h sáng ngày d/m/y dương lịch tại múi giờ tz. Kết quả của thuật toán là một góc Rađian giữa 0 và 2*PI. Ngày đầu tháng âm lịch là ngày chứa điểm hội diện (điểm Sóc), đó là thời điểm khi tuổi trăng bằng 0.

MoonAge(int d, int m, int y, double tz)

```
double sunLongitude = getSunLongitude(d, m, y, tz);
int x = toJD(d, m, y);
double day = x - 2447892 - tz / 24.0;
```

```
double meanLongitude = fixAngle(0.22997150421858628 * day +
5.556284436750021);
double meanAnomalyMoon = fixAngle(meanLongitude -
0.001944368345221015 * day - 0.6342598060246725);
double epochAngle = fixAngle(PI * 2 * day / 365.242191);
double meanAnomalySun = fixAngle(epochAngle -
0.05873240627141918);
double evection = 0.022233749341155764 * sin(2 *
(meanLongitude - sunLongitude) - meanAnomalyMoon);
double annual = 0.003242821750205464 * sin(meanAnomalySun);
double a3 = 0.00645771823237902 * sin(meanAnomalySun);
meanAnomalyMoon = meanAnomalyMoon + evection - annual - a3;
double center = 0.10975677534091541 * sin(meanAnomalyMoon);
double a4 = 0.0037350045992678655 * sin(2 *
meanAnomalyMoon);
double moonLongitude = meanLongitude + evection + center -
annual + a4;
double variation = 0.011489502465878671 * sin(2 *
(moonLongitude - sunLongitude));
moonLongitude = moonLongitude + variation;
double nodeLongitude = fixAngle(5.559050068029439 -
0.0009242199067718253 * day);
nodeLongitude = nodeLongitude - 0.0027925268031909274 *
sin(meanAnomalySun);
double y2 = sin(moonLongitude - nodeLongitude);
double x2 = cos(moonLongitude - nodeLongitude);
double moonEclipLong = atan2(y2 * cos(0.08980357792017056),
x2) + nodeLongitude;
double ret = fixAngle(moonEclipLong - sunLongitude);
return ret;
```

Đổi ngày dương lịch ra ngày âm lịch

Với các công thức và thuật toán hỗ trợ trên chúng ta bây giờ có thể đổi ngày dương lịch ra ngày âm lịch.

Tính ngày đầu tháng âm lịch

Cho ngày dương lịch d/m/y và múi giờ tz, ta tính xem tháng âm lịch chứa ngày d/m/y bắt đầu vào ngày nào. Trước hết, tính tuổi trăng moonAge1 tại lúc 0h và moonAge2 lúc 24h ngày d/m/y. Nếu moonAge1 > moonAge2 thì d/m/y là ngày chứa Sóc, nếu không lùi lại cho tới khi nào moonAge1 > moonAge2. (moonAge1 > moonAge2 chỉ xảy ra khi moonAge1 xấp xỉ 2*PI và moonAge2 xấp xỉ 0, khi đó trong ngày có một thời điểm mà tuổi trăng bằng 0.)

NewMoonBefore(int d, int m, int y, double tz)

```
int jdn = Gregorian2JD(d, m, y);
do {
  (d1, m1, y1) = JD2Gregorian(jdn);
  (d2, m2, y2) = JD2Gregorian(jdn+1);
  double moonAge1 = MoonAge(d1, m1, y1, tz);
  double moonAge2 = MoonAge(d2, m2, y2, tz);
  jdn = jdn - 1;
} while (moonAge2 > moonAge1);
return (d1, m1, y1);
```

Chú ý: cách tính này không nhanh lắm, cần tối ưu hóa khi lập trình. Chẳng hạn, khi biết một ngày không phải là ngày Sóc thì không nên nhảy lùi từng ngày một và kiểm tra ngày ngay trước đó mà có thể đoán là ngày Sóc cách ngày hiện tại khoảng bao nhiêu ngày để lùi lại tới sát ngày dầu tháng. Mặt trăng cần khoảng 29.5 ngày để đi hết một vòng tròn (2*PI), như thế để đạt được tuổi trăng moonAge thì cần khoảng moonAge/(2*PI/29.5) ngày.

Khi biết (d1, m1, y1) là ngày Sóc ngay trước (d, m, y), ta tính được khoảng cách giữa 2 ngày này là Gregorian2JD(d, m, y) - Gregorian2JD(d1, m1, y1) như thế biết (d, m, y) là ngày mùng mấy âm lịch. Nếu hiệu số này là 0 thì (d, m, y) chính là ngày mùng 1 âm lịch, ngày chứa điểm Sóc.

Tính tháng âm lịch chứa ngày Đông chí

Đông chí của một năm y thường rơi vào khoảng ngày 20-22 tháng 12 năm đó. Chúng ta nhớ lại rằng Đông chí là thời điểm mà kinh độ của mặt trời trên đường hoàng đạo là 3*PI/2. Trước hết ta tính ngày (d1, m1, y1) bắt đầu tháng âm lịch chứa ngày 24/12 dương lịch bằng phương thức NewMoonBefore nói trên. Đông chí chỉ rơi vào tháng âm lịch bắt đầu ngày (d1, m1, y1) nếu kinh độ mặt trời vào lúc 0h ngày (d1, m1, y1) nhỏ hơn hoặc bằng 3*PI/2. Nếu không thì tháng âm lịch trước đó là tháng có Đông chí.

LunarMonth11(int y, double tz)

```
int jdn = getNewMoonBefore(24, 12, y, tz);
(d1, m1, y1) = JD2Gregorian(jdn);
double sunLong = getSunLongitude(d1, m1, y1, tz);
if (sunLong > 3*PI/2) {
    jdn = getNewMoonBefore(25, 11, y, tz);
}
return JD2Gregorian(jdn);
```

Xác định năm nhuận, tháng nhuận

Theo lịch pháp thì tháng 11 âm lịch (tháng Tý) là tháng chứa ngày Đông chí. Với hai hàm NewMoonBefore và LunarMonth11 ta có thể tính được ngày dương lịch (d,m,y) là ngày mùng mấy âm lịch và tháng âm lịch chứa ngày này cách tháng 11 âm lịch bao nhiêu tháng. Để xác định tên của tháng này ta phải tính xem tháng đó hoặc một tháng trước đó có phải tháng nhuận không. Để làm điều này ta tính (d1, m1, y1) = NewMoonBefore(d, m, y, tz) và chọn hai ngày (d2, m2, y2) và (d3, m3, y3) sao cho (d2,m2,y2)<=(d1,m1,y1)<(d3,m3,y3) và (d2, m2, y2) và (d3, m3, y3) là những ngày bắt đầu tháng 11 âm lịch của 2 năm liền nhau.

Khoảng cách giữa (d1, m1, y1) và (d2, m2, y2) là X=(Gregorian2JD(d1,m1,y1)-Gregorian2JD(d2,m2,y2))/29 tháng âm lịch. Nếu khoảng cách giữa (d2, m2, y2) và (d3, m3, y3) nhỏ hơn 365 ngày thì giữa chúng

có 12 tháng âm lịch, như vậy không có tháng nhuận nào. Như thế (d, m, y) là ngày Gregorian2JD(d, m, y) - Gregorian2JD(d1, m1, y1) + 1 tháng T âm lịch với T = 11+X nếu X=0 hoặc X=1 và T=11+X-12 cho X từ 2 đến 11.

Nếu (d3, m3, y3) cách (d2, m2, y2) một khoảng lớn hơn 365 ngày thì ta tìm tháng âm lịch đầu tiên không có Trung khí giữa (d2, m2, y2) và (d3, m3, y3). Một tháng âm lịch có Trung khí nếu một trong các góc định nghĩa Trung khí (0*PI/6, PI/6, 2*PI/6, ..., 11*PI/6) nằm giữa kinh độ mặt trời (Sun Longitude) tại ngày đầu tháng và kinh độ mặt trời tại ngày đầu tháng sau đó. Chẳng hạn, tháng âm lịch chứa ngày (d,m,y) bắt đầu vào ngày (d1,m1,y1). Tháng âm lịch sau đó bắt đầu vào ngày (d4,m4,y4). Nếu trong khoảng [SunLongitude(d1,m1,y1,tz), SunLongitude(d4,m4,y4,tz)) không có điểm định nghĩa Trung khí nào thì tháng âm lịch bắt đầu ngày (d1,m1,y1) không chứa Trung khí. Nếu đó là tháng đầu tiên không có Trung khí thì đó là tháng nhuận, như thế ngày (d,m,y) nằm trong tháng T nhuận với T = 10+X nếu X=1,2 và T=10+X-12 với X=3,4,...,12.

Tài liệu tham khảo

[1] Nguyễn Hoàng Phương, *Tích hợp đa văn hóa Đông Tây cho một chiến lược giáo dục tương lai*, NXB Giáo Dục, 1995

[2] Hoàng Kiếm, *Bài giảng chuyên đề Hệ cơ sở tri thức*, ĐH Khoa học Tự nhiên Tp Hồ Chí Minh, 2001

[3] Đỗ Phúc, *Bài giảng chuyên đề Khai thác dữ liệu*, ĐH Khoa học Tự nhiên Tp Hồ Chí Minh, 2003

[4] Thiệu Vĩ Hoa, *Dự đoán theo Tứ trụ*, Nhà xuất bản Văn hóa thông tin, Hà Nội, 2002

[5] Bùi Biên Hòa, *Không gian Kinh dịch với dự báo qua Bát tự Hà Lạc*, Nhà xuất bản Văn hóa thông tin. Hà Nội, 2002.

[6] Vipin Kumar và Mahesh Joshi, *Tutorial on High Performance Data Mining*, University of Minnessota, Minneapolis, USA, 2000.

[7] Adrian A. Hopgood, *Knowledge-based systems for Engineers and Scientists*, CRC press, 1993

[8] Dao-I Lin, *Fast Algorithms for Discovering the Maximum Frequent Set*, New York University, 1998

[9] Nguyễn Thiên Bảo, *Biểu diễn tri thức Kinh dịch và ứng dụng trong Phong thủy học*, ĐH Khoa học Tự nhiên, 2002

[10] Jiawei Han và Micheline Kamber, *Data Mining:Concepts and Techniques — Slides for Textbook — Chapter 7—* ,Department of Computer Science University of Illinois at Urbana-Champaign, 2003

[11] Hồ Ngọc Đức, *Thuật toán tính Âm lịch*, http://come.to/duc