MỤC LỤC

PHẦN II

TRI THỨC VÀ LẬP LUẬN

LỜI NÓI ĐẦU

Trí tuệ nhân tạo (TTNT) là một lĩnh vực của khoa học máy tính, nghiên cứu sự thiết kế của các tác nhân thông minh *("Computational intelligence is the studi of the degn of intelligens")*. Các áp dụng của TTNT rất đa dạng và phong phú, hiện nay đã có nhiều hệ thông minh ra đời: các hệ chuyên gia, các hệ điều khiển tự động, các robot, các hệ dịch tự động các ngôn ngữ tự nhiên , các hệ nhận dạng, các chương trình chơi cờ,… Kỹ thuật của TTNT đã được sử dụng trong việc xây dựng các hệ mềm, nhằm tạo ra các hệ mềm mang yếu tố thông minh, linh hoạt và tiện dụng. Ở nước ta, trong những năm gần đây, TTNT đã được đưa vào chương trình giảng dạy cho sinh viên các năm cuối ngành Tin học và Công nghệ thông tin. Cuốn sách này được hình thành trên cơ sở giáo trình TTNT mà chúng tôi đã giảng dạy cho sinh viên và các lớp cao học ngành Tin học và ngành Công nghệ thông tin trong các năm học từ 1997 tới nay, tại khoa Công nghệ thông tin, Đại học Khoa học tự nhiên, nay là khoa Công nghệ, Đại học Quốc gia Hà nội.

Cuốn sách này được viết như một cuốn nhập môn về TTNT, đối tượng phục vụ chủ yếu của nó là sinh viên các ngành Tin học và Công nghệ thông tin. Nội dung cuốn sách gồm hai phần:

- ***Phần 1:*** Giải quyết vấn đề bằng tìm kiếm. Trong phần này, chúng tôi trình bày các phương pháp biểu diễn các vấn đề và các kỹ thuật tìm kiếm. các kỹ thuật tìm kiếm, đặc biệt là tìm kiếm kinh nghiệm (heuristic serch), được sử dụng thường xuyên trong nhiều lĩnh vực nghiên cứu của TTNT.
- ***Phần 2:*** Biểu diễn tri thức và lập luận. Phần này đề cập đến các ngôn ngữ biểu diễn tri thức, đặc biệt là các logic và các phương pháp luận trong mỗi ngôn ngữ biểu diễn tri thức. Các kỹ thuật biểu diễn tri thức và lập luận đóng vai trò quan trọng trong việc thiết kế các hệ thông minh.

Tuy nhiên với mong muốn cuốn sách này có thể dùng làm tài liệu tham khảo cho một phạm vi rộng rãi các đọc giả, chúng tôi cố gắng trình bày một cách hệ thống các khái niệm và các kỹ thuật cơ bản của TTNT, nhằm giúp cho đọc giả có cơ sở để đi vàonghiên cứu các lĩnh vực chuyên sâu của TTNT, chẳng hạn như lập kế hoạch (planning),học máy(machine learning),

nhìn máy (computer viison), hiểu ngôn ngữ tự nhiên (natural language understanding).

Hai ngôn ngữ thao tác ký hiệu được sử dụng nhiều trong lập trình TTNT là Lisp và Prolog. Trong các sách viết về TTNT các năm gần đây, một số tác giả, chẳng hạn trong [5] và [7] , đã sử dụng common Lisp để mô tả thuật toán. Trong 20, các tác giả lại sử dụng Prolog để biểu diễn thuật toán. Ở nước ta, các ngôn ngữ Lisp và Prolog được ít người biết đến, vì vậy chúng tôi biểu diễn các thuật toán trong sách này theo cách truyền thống. Tức là chúng tôi sử dụng các cấu trúc điều khiển (tuần tự, điều kiện, lặp) mà mọi người quen biết. Đặc biệt, chúng tôi sử dụng cấu trúc loop do <dãy câu> để biểu diễn rằng, <dãy câu> được thực hiện lặp. Toán tử exit để thoát khỏi vòng lặp, còn toán tử stop đẻ dừng sự thực hiện thuật toán, các bạn có thể lựa chọn một trong các ngôn ngữ sau để sử dụng: Common Lisp, Scheme, Prolog, Smalltalk, C** hoặc ML (xem [28]).

Chúng tôi xin chân thành cảm ơn giáo sư Nguyễn Văn Hiệu, chủ nhiệm khoa Công nghệ, Đại học Quốc gia Hà nội đã tạo điều kiện thuận lợi cho chúng tôi viết cuốn sách này.

Cuốn sách chắc chắn không tránh khỏi những thiếu sót. Chúng tôi mong nhận được sự góp ý của độc giả. Thư góp ý xin gửi về Bộ môn Khoa Học Máy Tính, Khoa Công Nghệ, Đại Học Quốc Gia Hà Nội.

Tác giả

NHẬP MÔN

TRÍ TUỆ NHÂN TẠO LÀ GÌ?

Thuật ngữ **trí tuệ nhân tạo** (artificial intelligence) được Jonh McCarthy đưa ra trong hội thảo ở Dartmouth vào mùa hè 1956. Trong hội thảo này có mặt các tên tuổi nổi tiếng như Marvin Minsky, Claude Shannon, Nathaniel Rochester, Arthur Samuel, Allen Newell và Herbert Simon. Trước hội thảo này, từ năm 1952 Arthur Samuel đã viết chương trình chơi cờ. Samuel đã bác bỏ tư tưởng cho rằng máy tính chỉ có thể làm được cái mà người ta bảo nó làm, vì chương trình của Samuel có thể học để chơi tốt hơn người viết ra nó. Đến hội thảo này, Allen Newell và Herbert Simon cũng đã viết chương trình lập luận với tên gọi "the logic theorist". Chương trình của các ông có khả năng chứng minh hầu hết các định lý trong chương 2 cuốn "Principia Mathematics" của Russell và Whitehead. Trong hội thảo ở Dartmouth, các nhà nghiên cứu đã thảo luận và vạch ra các phương hướng nghiên cứu của lĩnh vực Trí tuệ nhân tạo (*TTNT*). Vì vậy, hội thảo ở Dartmouth, mùa hè năm 1956 được xem là thời điểm ra đời thực sự của lĩnh vực nghiên cứu TTNT.

Trong các sách viết về TTNT các năm gần đây, các tác giả đưa ra nhiều định nghĩa về TTNT. Chúng tôi dẫn ra đây một số định nghĩa:

- "Sự nghiên cứu các năng lực trí tuệ thông qua việc sử dụng các mô hình tính toán" ("The study ò mental faculties through the use ò computational models" – Charniak and McDormott, 1985)
- "Nghệ thuật tạo ra các máy thực hiện các chức năng đòi hỏi sự thông minh khi được thực hiện bởi con người" ("The art of creating machies that perform functions that require intelligence when performed by people" – Kurzweil. 1990).
- "Lĩnh vực nghiên cứu tìm cách giải quyết và mô phỏng các hành vi thông minh trong thuật ngữ các quá trình tính toán" ("A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" – Schalkoff, 1990).
- "Sự nghiên cứu các tính toán để có thể nhận thức, lập luận và hành động" ("The study of computations that make it possible to perceive, reason, and act" – Winston, 1992).

- "Một nhánh của khoa học máy tính liên quan tới sự tự động hoá các hành vi thông minh" ("The branch of computer science that is concerned with the automation of intelligent behavior" – Luger and Stubblefield, 1993)

Sau đây là một số định nghĩa gần đây nhất:

- "TTNT là sự thiết kế và nghiên cứu các chương trình máy tính ứng xử một cách thông minh. Các chương trình này được xây dựng để thực hiện các hành vi mà khi ở người hoặc động vật chúng ta xem là thông minh" ("Artificial Intelligence is the design and study of computer programs that behave intelligently.These programs are constructed to perform as would a human or an animal whose behvior we consider intelligent" – Dean, Allen and Aloimonos, 1995).
- " TTNT là sự nghiên cứu các tác nhân tồn tại trong môi trường, nhận thức và hành động" ("Artificial Intelligence is the design of agents that exists in an environment and act" – Russell and Norvig, 1995).
- " TTNT là sự nghiên cứu ác thiết kế các tác nhân thông minh" ("Computational Intelligence is the study of the design of Intelligent agents" – Pulle, Mackworth and Goebel, 1998).

Hiện nay nhiều nhà nghiên cứu quan niệm rằng, TTNT là lĩnh vực nghiên cứu sự nghiên cứu các tác nhân thông minh (intelligent agent). Tác nhân thông minh là bất cứ ấci gì tồn tại trong môi trường và hành động một cách thông minh (Một câu hỏi được đặt ra: hành động như thế nào thì được xem là thông minh?).

Mục tiêu khoa học của TTNT là hiểu được bản chất của các hành vi thông minh. mục tiêu thực tiễn, công nghệ của TTNT là xây dựng nân các hệ thông minh. Phương pháp luận nghiên cứu ở đây cũng tương tự như khi chúng ta nghiên cứu để hiểu được các nguyên lý bay, rồi thiết kế nên các máy biết bay (máy bay). Máy bay không phải là sự mô phỏng con chim, song nó có khả năng baybay tốt hơn chim.

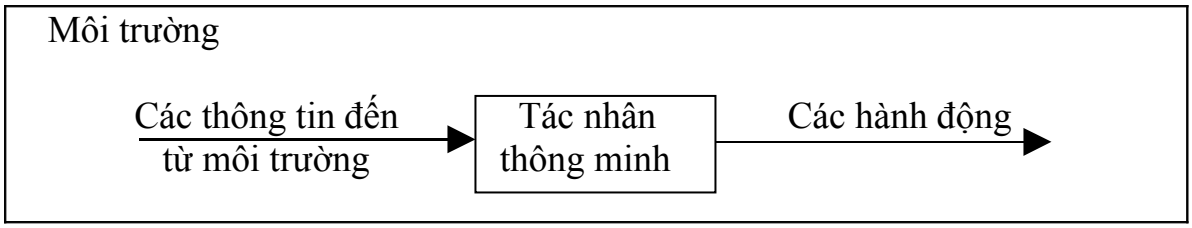
Một số ngành khoa học khác, chẳng hạn như Triết học, Tâm lý học cũng quan tâm nghiên cứu các năng lực được xem của con người. Song khác với Triết học và Tâm lý học, TTNT là một ngành của khoa học máy tính – nghiên cứu sử lý thông tin bằng máy tính, do đó TTNT đặt ra mục tiêu nghiên cứu: làm thế nào thể hiện được các hành vi thông minh bằng thuật toán, rồi nghiên cứu các phương pháp cài đặt các chương trình có thể thực hiện được các hành vi thông minh. Như vậy chúng ta cần xây dựng các mô hình tính toán (Computational modeles phục vụ cho việc giải thích, mô tả

các hành vi thông minh bằng thuật toán, tiếp theo chúng ta cần chỉ ra tính hiệu quả, tính khả thi của thuật toánthực hiện một nhiệm vụ, và đưa ra các phương pháp cài đặt.

TÁC NHÂN THÔNG MINH

Tác nhân (agent) là bất cứ cái gì hành động trong môi trường. các tác nhân có thể là con người, con sâu, con con chó, robot infobot, … Mục tiêu của TTNT là nhiên cứu và thiết kế các tác nhân thông minh: các tác nhân tồn tại trong môi trường và hành động một cách thông minh.

Tồn tại trong môi trường, nên tác nhân thông minh cần có khả năng nhận thức được môi trường. Chẳng hạn, con người có thẻ nhận thức được môi trường nhờ tai, mắt và các giác quan khác. Để nhận thức được môi trường các robot sẽ được trang bị các **bộ cảm nhận** (sensors), đó là các thiết bị vật lý, chẳng hạn camera, các máy đo đạc,…Các tác nhân thông minh cần có các bộ tác động (effectors) để đưa ra các hành động đáp ứng môi trường. Với con người, đó là chân, tay và các bộ phận khác của thân thể. Với ác robot, đó có thể là các cánh tay robot,…



Chúng ta có thể xem tác nhân như một hộp đen, đầu vào là các thông tin nhận thức từ môi trường, đầu ra là các hành động thích ứng với môi trường, như trong hình trên. Bây giờ chúng ta xét xem cần phải cài đặt vào bên trong hộp đen cái gì để hành động đầu ra là hợp lý, thích ứng với các thông tin đầu vào.

Tác nhân cần có bộ nhớ để lưu giữ các tri thức chung về lĩnh vực, về môi trường mà nó được thiết kế để hoạt động trong lĩnh vực đó. Chẳng hạn, đối với robot lái taxi, tri thức về môi trường mà robot cần có là các tri thức về mạng lưới giao thông trong thành phố, về luật lệ giao thông,… Đối với các hệ chuyên gia trợ giúp chẩn đoán bệnh, các tri thức cần lưu là các tri thức của các bác sĩ về bệnh lý, về các phương án điều trị,…Bộ nhớ của tác nhân còn để ghi lại các tri thức mới mà tác nhân mới rút ra được trong quá trình hoạt động trong môi trường. Trong nhiều trường hợp, nó cũng cần lưu lại vết của các trạng thái của môi trường , bởi vì hành động thích hợp mà nó

cần đưa ra không chỉ phụ thuộc vào trạng thái hiện tại của môi trường mà còn phụ thuộc vào trạng thái của môi trường trong quá khứ.

Chúng ta cần trang bị cho tác nhân một chương trình: chương trình tác nhân (agent program). Chương trình này là sự mô tả thuật toán kết hợp với các thông tin về trạng thái của môi trường với ác tri thức đã được lưuđể cho ra hành động thích ứng.

Bây giờ, chúng ta trả lời câu hỏi: "các tác nhân như thế nào thì được xem là thông minh?" bằng định nghĩa về thông minh của Turing: thử nghiệm Turing (Turing test).

THỬ NGHIỆM TURING

Alan Turing (1950) đã xác định các hành vi thông minh như là các hành vi trong các nhiệm vụ nhận thức đạt tới mức độ có thể đánh lừa được con người.

Sau đây là dạng tổng quát của thử nghiệm Turing. Một người hỏi là con người ngồi ở một phòng. Đối tác của người hỏi là một máy tính được đặt ở một phòng khác, hai bên trao đổi thông tin với nhau thông qua các phương tiện truyền tin hiện đại. Nếu máy tính có thể làm cho người hỏi tưởng lầm là có người khác đang nói chuyện với máy mình thì máy tính được xem là thông minh.

Bây giờ chúng ta xét xem, để thực hiện được ác hành vi được xem là thông minh, các tác nhân thông minh (TNTM) cần có khả năng gì?

- TNTM cần có khả năng ghi nhớ và lập luận. nó sử dụng các tri thức đã lưu trữ, và bằng lập luận để rút ra những kết luận đáp ứng những câu hỏi mà người đặt ra .

Biểu diễn tri thức và lập luận là lĩnh vực nghiên cứu trung tâm của TTNT.

- Người hỏi có thể hỏi bằng ngôn ngữ tự nhiên, chẳng hạn tiếng Anh. Vì vậy, TNTM cần có khả năng biểu diễn ngôn ngữ tự nhiên.

Hiểu ngôn ngữ tự nhiên (natural language understanding) là một lĩnh vực nghiên cứu quan trọng của TTNT.

- Người hỏi có thể đưa ra một số hoàn cảnh và các hành động cần tiến hành tương ứng với mỗi hoàn cảnh đó, rồi đưa ra một hoàn cảnh mới và hỏi tác nhân cần phải làm gì trong hoàn cảnh đó. Để trả lời được câu hỏi này,

TNTM cần có khả năng học để có thể đưa ra hành động thích ứng với hoàn cảnh mới.

Học máy (machine learning) là một lĩnh vực nghiên cứu của TTNT đang phát tiển mạnh mẽ và có nhiều ứng dụng trong các lĩnh vực khác nhau, chẳng hạn trong khám phá tri thức và khai thác dữ liệu.

- Người hỏi có thể đưa ra một số hình ảnh về các đối tượng và kiểm tra khả năng nhận biết của các đối tượng đó của TNTM.

Nhìn máy (computer vision) là lĩnh vực nghiên cứu để máy tính có thể hiểu được cấu trúc và các tính chất của các đối tượng trong không gian ba chiều từ các hình ảnh hai chiều.

- Khi được cho các nhiệm vụ cần thực hiện, TNTM cần có khả năng đưa ra một dãy các hành động mà nó cần thực hiện để đạt được mục đích đó. Quá trình này được gọi là lập kế hoạch.

Lập kế hoạch (planning) là một lĩnh vực nghiên cứu quan trọng của TTNT.

BIỂU DIỄN VÀ LẬP LUẬN

Khi gặp một vấn đề cần giải quyết, một nhiệm vụ cần thực hiện, điều đầu tiên chúng ta phải làm là cần phải biểu diễn vấn đề, cần phải xác định cái gì tạo thành lời giải (nghiệm) của vấn đề, và sau đó mới đi tìm thuật toán tính nghiệm. Cần lưu ý rằng, biểu diễn vấn đề đóng vai trò quan trọng trong việc giải quyết một vấn đề. Trong nhiều trường hợp, khi chúng ta chọn được cách biểu diễn thích hợp cho một vấn đề thì vấn đề hầu như đã được giải quyết. Chúng ta có thể biểu diễn nhiều vấn đề bằng cách xác định trạng thái đích cần đạt tới và các hành động làm biến đổi các trạng thái của thế giới. Khi đó việc tìm nghiệm của vấn đề được quy về việc tìm kiếm một dãy các hành động dẫn tới trạng thái đích. Chú ý rằng, tìm kiếm đóng vai trò quan trọng trong việc giải quyết vấn đề và trong nhiều lĩnh vực nghiên cứu của TTNT, chẳng hạn như trong học máy và lập kế hoạch. Vì vậy, phần I của sách này dành cho việc trình bày phương pháp biểu diễn vấn đề trong không gian trạng thái và nghiên cứu các chiến lược tìm kiếm.

Thực tế cho thấy rằng, Để thực hiện được các nhiệm vụ khó khăn và phức tạp, con người cần sử dụng một khối lượng lớn tri thức về lĩnh vực liên quan. Một câu hỏi được đặt ra: tri thức là gì? Một cách không hình thức, chúng ta hiểu tri thức là thông tinliên quan tới một đối tượng, một hoàn cảnh, một lĩnh vực. Để máy tính có thể lưu trữ được tri thức, sử dụng được tri thức, chúng ta cần tìm các phương pháp biểu diễn tri thức.

Lập luận tự động được hiểu là quá trình tính toán trên các biểu diễn tri thức, mà khi cho đầu vào là các biểu diễn tri thức, chúng ta nhận được các đầu ra là các biểu diễn tri thức mới. Mục tiêu trung tâm của TTNT là nghiên cứu thiết kế các hệ thông minh, nó lưu trữ tri thức về lĩnh vực và có khả năng đưa ra hành động thích ứng bằng lập luận dựa tren các tri thức đã lưu trữ và các thông tin thu nhận từ môi trường. Do đó, vấn đề trung tâm mà chúng ta đề cập đến trong sách này là vấn đề biểu diễn tri thức và lập luận. Chúng ta sẽ nghiên cứu các mô hình biểu diễn tri thức khác nhau và các phương pháp lập luận trong mỗi mô hình đó.

CÁC ÁP DỤNG

Các áp dụng của TTNT là rất đa dạng: các hệ điều khiển tự động các quá trình sản xuất công nghiệp; các robot làm việc trong các môi trường đặc biệt, chẳng hạn các robot thám hiểm các hành tinh; các robot làm việc trong các điều kiện nguy hiểm đến tính mạng con người; các hệ chuyên gia trong các lĩnh vực; các hệ dịch tự động; các hệ nhận dạng; các infobot làm việc trong môi trường thuần tuý thông tin; các chương trình chơi cờ,…

Sau đây chúng ta sẽ trình bày vài cách áp dụng. Qua các ví dụ này, chúng ta muốn chỉ ra những vấn đề cần giải quyết khi chúng ta định xây dựng một hệ thông minh.

- Robot đưa thư

Giả sử chúng ta muốn thiết kế một robot để phân phát thư, các gói nhỏ và phục vụ cà phê trong toà nhà làm việc của một công ty. Đương nhiên là robot phải được trang bịcác bộ cảm nhận có thể "nhìn" và "nghe", chẳng hạn video camera, các thiết bị thu thanh,…Nó cần có các bộ phậnthực hiện hành động, chẳng hạn bánh xe chuyển động, "cánh tay" để nhặt lên, đặt xuống các đồ vật,…

Các thông tin đến từ môi trường ở đây là các hình ảnh và âm thanh, … mà robot thu được nhờ các bộ cảm nhận. Khi nhận được mệnh lệnh, chẳng hạn "mang cà phê cho giám đốc", nó phải đưa ra một dãy hành động đáp ứng môi trường nhằm đạt được mục đích: có cà phê cho giám đốc.

Các tri thức mà robot cần biết trước và được lưu trong cơ sở tri thức của robot là cấu trúc của toà nhà: Gồm có mấy tầng, các phòng trong mỗi tầng và hành lang được bố trí ra sao, thang máy ở đâu; các đối tượng mà robot có thể gặp trong toà nhà, ai làm việc ở đâu, cà phê ở phòng nào, …Cơ sở tri thức của robot còn có thể chứa các tri thức cho biết các hoàn cảnh mà robot

thường gặp và các hành động mà robot cần thực hiện cách thích hợp sao cho thuận tiện cho việc lưu trữ, tìm kiếm và suy diễn.

Từ các thông tin hình ảnh, robot cần có khả năng hiểu được các đối tượng mà nó gặp để có thể suy ra các hành động thích hợp cần thực hiện.

Robot cần có khả năng hiểu được các mệnh lệnh bằng ngôn ngữ tự nhiên, từ đó suy ra các mục đích mà nó cần đạt tới.

Khi có nhiều mục đích cần đạt tới, robot cần có khả năng lập kế hoạch để đạt được các mục đích đó.

Robot cần có khả năng tìm đường đi (tối ưu nhất) giữa các vị trí trong toà nhà, từ các thông tin về sự bố trí và chức năng của các phòng và các bộ phận trong toà nhà.

Chúng ta cũng cần trang bị cho robot có khả năng học để có thể rút kinh nghiệm, khái quát hoá từ thực tiễn trong quá trình phục vụ, và do đó, khi gặp các hoàn cảnh mới nó có thể đưa ra các hành động thích hợp.

- ***Hệ chuyên gia trong y học***

Các hệ chuyên gia y học có mục đích trợ giúp các bác sĩ trong chẩn đoán bệnh và điều trị. Môi trường của các hệ chuyên gia này là người bệnh. Các thông tin đến từ môi trường là các triệu chứng của người bệnh, các kết quả xét nghiệm từ người bệnh.

Cơ sở tri thức của các hệ chuyên gia này là các tri thức của các bác sĩ về bệnh lý, các tri thức này thường được biểu diễn dưới dạng các luật ***if-then***. Chẳng hạn, Feigenbaur và Shortiffe đã phát triển hệ MYCIN để chẩn đoán bệnh nhiễm trùng máu. Cơ sở luật của hệ MYCIN chứa khoảng 450 luật, mỗi luật gắn với một mức độ chắc chắn (điều này phù hợp với tính chất của các kết luận y học). Hệ MYCIN có khả năng làm việc tốt như các bác sĩ giỏi trong lĩnh vực.

Các hệ chuyên gia này có khả năng suy diễn để từ các triệu chứng, các kết quả xét nghiệm người bệnh và sử dụng các luật trong cơ sở luật để suy ra các kết luận về bệnh. Nó cũng có khả năng gợi ý cần kiểm tra cái gì, cần xét nghiệm cái gì ở người bệnh. Cũng như các bác sĩ, các hệ chuyên gia này có khả năng giải thích các kết luận mà nó đã đưa ra, nó có thể cho chúng ta biết tại sao nó đi đến kết luận như thế.

PHẦN I

GIẢI QUYẾT VẤN ĐỀ BẰNG TÌM KIẾM

Vấn đề tìm kiếm, một cách tổng quát, có thể hiểu là tìm một đối tượng thỏa mãn một số đòi hỏi nào đó, trong một tập hợp rộng lớn các đối tượng. Chúng ta có thể kể ra rất nhiều vấn đề mà việc giải quyết nó được quy về vấn đề tìm kiếm.

Các trò chơi, chẳng hạn cờ vua, cờ carô có thể xem như vấn đề tìm kiếm. Trong số rất nhiều nước đi được phép thực hiện, ta phải tìm ra các nước đi dẫn tới tình thế kết cuộc mà ta là người thắng.

Chứng minh định lý cũng có thể xem như vấn đề tìm kiếm. Cho một tập các tiên đề và các luật suy diễn, trong trường hợp này mục tiêu của ta là tìm ra một chứng minh (một dãy các luật suy diễn được áp dụng) để đưa đến công thức mà ta cần chứng minh.

Trong các lĩnh vực nghiên cứu của **Trí Tuệ Nhân Tạo**, chúng ta thường xuyên phải đối đầu với vấn đề tìm kiếm. Đặc biệt trong lập kế hoạch và học máy, tìm kiếm đóng vai trò quan trọng.

Trong phần này chúng ta sẽ nghiên cứu các kỹ thuật tìm kiếm cơ bản được áp dụng để giải quyết các vấn đề và được áp dụng rộng rãi trong các lĩnh vực nghiên cứu khác của **Trí Tuệ Nhân Tạo**. Chúng ta lần lượt nghiên cứu các kỹ thuật sau:

- Các kỹ thuật tìm kiếm mù, trong đó chúng ta không có hiểu biết gì về các đối tượng để hướng dẫn tìm kiếm mà chỉ đơn thuần là xem xét theo một hệ thống nào đó tất cả các đối tượng để phát hiện ra đối tượng cần tìm.
- Các kỹ thuật tìm kiếm kinh nghiệm (tìm kiếm heuristic) trong đó chúng ta dựa vào kinh nghiệm và sự hiểu biết của chúng ta về vấn đề cần giải quyết để xây dựng nên hàm đánh giá hướng dẫn sự tìm kiếm.
- Các kỹ thuật tìm kiếm tối ưu.
- Các phương pháp tìm kiếm có đối thủ, tức là các chiến lược tìm kiếm nước đi trong các trò chơi hai người, chẳng hạn cờ vua, cờ tướng, cờ carô.

CHƯƠNG 1
CÁC CHIẾN LƯỢC TÌM KIẾM MÙ

Trong chương này, chúng ta sẽ nghiên cứu các chiến lược tìm kiếm mù (blind search): tìm kiếm theo bề rộng (breadth-first search) và tìm kiếm theo độ sâu (depth-first search). Hiệu quả của các phương pháp tìm kiếm này cũng sẽ được đánh giá.

1.1 . BIỂU DIỄN VẤN ĐỀ TRONG KHÔNG GIAN TRẠNG THÁI

Một khi chúng ta muốn giải quyết một vấn đề nào đó bằng tìm kiếm, đầu tiên ta phải xác định không gian tìm kiếm. Không gian tìm kiếm bao gồm tất cả các đối tượng mà ta cần quan tâm tìm kiếm. Nó có thể là không gian liên tục, chẳng hạn không gian các véctơ thực n chiều; nó cũng có thể là không gian các đối tượng rời rạc.

Trong mục này ta sẽ xét việc biểu diễn một vấn đề trong không gian trạng thái sao cho việc giải quyết vấn đề được quy về việc tìm kiếm trong không gian trạng thái.

Một phạm vi rộng lớn các vấn đề, đặc biệt các câu đố, các trò chơi, có thể mô tả bằng cách sử dụng khái niệm trạng thái và toán tử (phép biến đổi trạng thái). Chẳng hạn, một khách du lịch có trong tay bản đồ mạng lưới giao thông nối các thành phố trong một vùng lãnh thổ (hình 1.1), du khách đang ở thành phố A và anh ta muốn tìm đường đi tới thăm thành phố B. Trong bài toán này, các thành phố có trong bản đồ là các trạng thái, thành phố A là trạng thái ban đầu, B là trạng thái kết thúc. Khi đang ở một thành phố, chẳng hạn ở thành phố D anh ta có thể đi theo các con đường để tới các thành phố C, F và G. Các con đường nối các thành phố sẽ được biểu diễn bởi các toán tử. Một toán tử biến đổi một trạng thái thành một trạng thái khác. Chẳng hạn, ở trạng thái D sẽ có ba toán tử dẫn trạng thái D tới các trạng thái C, F và G. Vấn đề của du khách bây giờ sẽ là tìm một dãy toán tử để đưa trạng thái ban đầu A tới trạng thái kết thúc B.

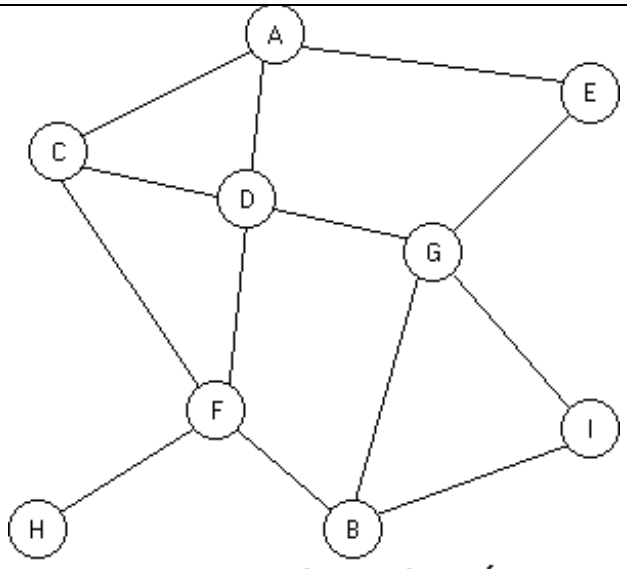
Một ví dụ khác, trong trò chơi cờ vua, mỗi cách bố trí các quân trên bàn cờ là một trạng thái. Trạng thái ban đầu là sự sắp xếp các quân lúc bắt đầu cuộc chơi. Mỗi nước đi hợp lệ là một toán tử, nó biến đổi một cảnh huống trên bàn cờ thành một cảnh huống khác.

Như vậy muốn biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:

- Trạng thái ban đầu.
- Một tập hợp các toán tử. Trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.
  Tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu bằng cách áp dụng một dãy toán tử, lập thành không gian trạng thái của vấn đề.
  Ta sẽ ký hiệu không gian trạng thái là U, trạng thái ban đầu là u_0 ($u_0 \in U$). Mỗi toán tử R có thể xem như một ánh xạ R: U→U. Nói chung R là một ánh xạ không xác định khắp nơi trên U.
- Một tập hợp T các trạng thái kết thúc (trạng thái đích). T là tập con của không gian U. Trong vấn đề của du khách trên, chỉ có một trạng thái đích, đó là thành phố B. Nhưng trong nhiều vấn đề (chẳng hạn các loại cờ) có thể có nhiều trạng thái đích và ta không thể xác định trước được các trạng thái đích. Nói chung trong phần lớn các vấn đề hay, ta chỉ có thể mô tả các trạng thái đích là các trạng thái thỏa mãn một số điều kiện nào đó.

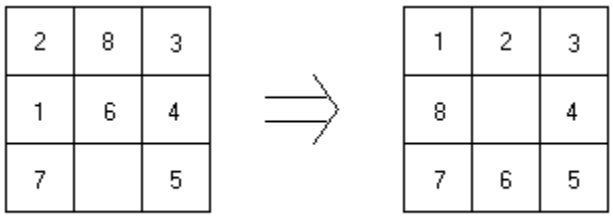
Khi chúng ta biểu diễn một vấn đề thông qua các trạng thái và các toán tử, thì việc tìm nghiệm của bài toán được quy về việc tìm đường đi từ trạng thái ban đầu tới trạng thái đích. (Một đường đi trong không gian trạng thái là một dãy toán tử dẫn một trạng thái tới một trạng thái khác).

Chúng ta có thể biểu diễn không gian trạng thái bằng đồ thị định hướng, trong đó mỗi đỉnh của đồ thị tương ứng với một trạng thái. Nếu có toán tử R biến đổi trạng thái u thành trạng thái v, thì có cung gán nhãn R đi từ đỉnh u tới đỉnh v. Khi đó một đường đi trong không gian trạng thái sẽ là một đường đi trong đồ thị này.



Hình 1.1 Tìm đường đi từ A đến B

Sau đây chúng ta sẽ xét một số ví dụ về các không gian trạng thái được xây dựng cho một số vấn đề.



Hình 1.2 Trạng thái ban đầu và trạng thái kết thúc của bài toán số.

Ví dụ 1: Bài toán 8 số. Chúng ta có bảng 3x3 ô và tám quân mang số hiệu từ 1 đến 8 được xếp vào tám ô, còn lại một ô trống, chẳng hạn như trong hình 1.2 bên trái. Trong trò chơi này, bạn có thể chuyển dịch các quân ở cạch ô trống tới ô trống đó. Vấn đề của bạn là tìm ra một dãy các chuyển dịch để biến đổi cảnh huống ban đầu (hình 1.2 bên trái) thành một cảnh huống xác định nào đó, chẳng hạn cảnh huống trong hình 1.2 bên phải.

Trong bài toán này, trạng thái ban đầu là cảnh huống ở bên trái hình 1.2, còn trạng thái kết thúc ở bên phải hình 1.2. Tương ứng với các quy tắc chuyển dịch các quân, ta có bốn toán tử: ***up*** (đẩy quân lên trên), ***down*** (đẩy quân xuống dưới), ***left*** (đẩy quân sang trái), ***right*** (đẩy quân sang phải). Rõ ràng là, các toán tử này chỉ là các toán tử bộ phận; chẳng hạn, từ trạng thái ban đầu (hình 1.2 bên trái), ta chỉ có thể áp dụng các toán tử ***down, left, right***.

Trong các ví dụ trên việc tìm ra một biểu diễn thích hợp để mô tả các trạng thái của vấn đề là khá dễ dàng và tự nhiên. Song trong nhiều vấn đề

việc tìm được biểu diễn thích hợp cho các trạng thái của vấn đề là hoàn toàn không đơn giản. Việc tìm ra dạng biểu diễn tốt cho các trạng thái đóng vai trò hết sức quan trọng trong quá trình giải quyết một vấn đề. Có thể nói rằng, nếu ta tìm được dạng biểu diễn tốt cho các trạng thái của vấn đề, thì vấn đề hầu như đã được giải quyết.

Ví dụ 2: Vấn đề triệu phú và kẻ cướp. Có ba nhà triệu phú và ba tên cướp ở bên bờ tả ngạn một con sông, cùng một chiếc thuyền chở được một hoặc hai người. Hãy tìm cách đưa mọi người qua sông sao cho không để lại ở bên bờ sông kẻ cướp nhiều hơn triệu phú. Đương nhiên trong bài toán này, các toán tử tương ứng với các hành động chở 1 hoặc 2 người qua sông. Nhưng ở đây ta cần lưu ý rằng, khi hành động xẩy ra (lúc thuyền đang bơi qua sông) thì ở bên bờ sông thuyền vừa dời chỗ, số kẻ cướp không được nhiều hơn số triệu phú. Tiếp theo ta cần quyết định cái gì là trạng thái của vấn đề. Ở đây ta không cần phân biệt các nhà triệu phú và các tên cướp, mà chỉ số lượng của họ ở bên bờ sông là quan trọng. Để biểu diễn các trạng thái, ta sử dụng bộ ba (a, b, k), trong đó a là số triệu phú, b là số kẻ cướp ở bên bờ tả ngạn vào các thời điểm mà thuyền ở bờ này hoặc bờ kia, k = 1 nếu thuyền ở bờ tả ngạn và k = 0 nếu thuyền ở bờ hữu ngạn. Như vậy, không gian trạng thái cho bài toán triệu phú và kẻ cướp được xác định như sau:

- Trạng thái ban đầu là (3, 3, 1).
- Các toán tử. Có năm toán tử tương ứng với hành động thuyền chở qua sông 1 triệu phú, hoặc 1 kẻ cướp, hoặc 2 triệu phú, hoặc 2 kẻ cướp, hoặc 1 triệu phú và 1 kẻ cướp.
- Trạng thái kết thúc là (0, 0, 0).

1.2. CÁC CHIẾN LƯỢC TÌM KIẾM

Như ta đã thấy trong mục 1.1, để giải quyết một vấn đề bằng tìm kiếm trong không gian trạng thái, đầu tiên ta cần tìm dạng thích hợp mô tả các trạng thái của vấn đề. Sau đó cần xác định:

- Trạng thái ban đầu.
- Tập các toán tử.
- Tập T các trạng thái kết thúc. (T có thể không được xác định cụ thể gồm các trạng thái nào mà chỉ được chỉ định bởi một số điều kiện nào đó).

Giả sử u là một trạng thái nào đó và R là một toán tử biến đổi u thành v. Ta sẽ gọi v là trạng thái kề u, hoặc v được sinh ra từ trạng thái u bởi toán tử

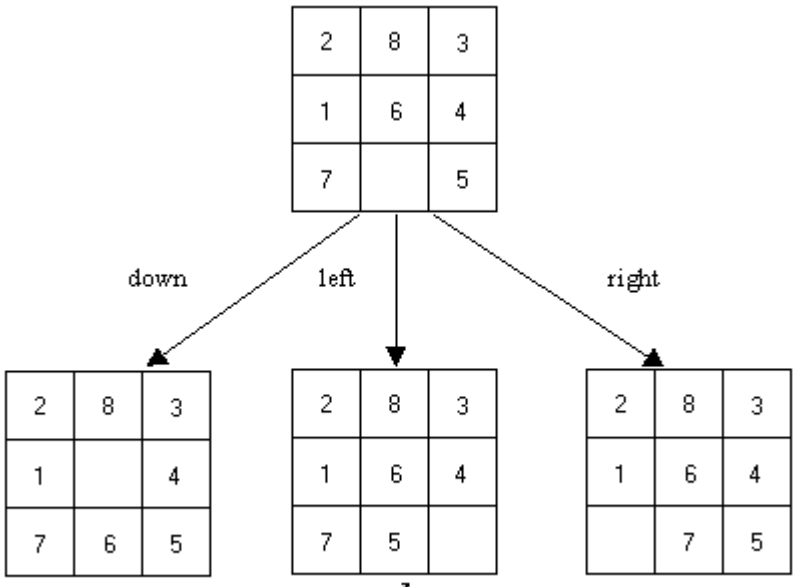
R. Quá trình áp dụng các toán tử để sinh ra các trạng thái kề u được gọi là phát triển trạng thái u. Chẳng hạn, trong bài toán tám số, phát triển trạng thái ban đầu (hình 1.2 bên trái), ta nhận được ba trạng thái kề (hình 1.3).

Khi chúng ta biểu diễn một vấn đề cần giải quyết thông qua các trạng thái và các toán tử thì việc tìm lời giải của vấn đề được quy về việc tìm đường đi từ trạng thái ban đầu tới một trạng thái kết thúc nào đó.

Có thể phân các chiến lược tìm kiếm thành hai loại:

- Các chiến lược tìm kiếm mù. Trong các chiến lược tìm kiếm này, không có một sự hướng dẫn nào cho sự tìm kiếm, mà ta chỉ phát triển các trạng thái ban đầu cho tới khi gặp một trạng thái đích nào đó. Có hai kỹ thuật tìm kiếm mù, đó là tìm kiếm theo bề rộng và tìm kiếm theo độ sâu.

Tư tưởng của tìm kiếm theo bề rộng là các trạng thái được phát triển theo thứ tự mà chúng được sinh ra, tức là trạng thái nào được sinh ra trước sẽ được phát triển trước. Còn trong tìm kiếm theo độ sâu, trạng tháI sinh ra



Hình 1.3 Phát triển một trạng thái

sau cùng sẽ được phát triển trước.

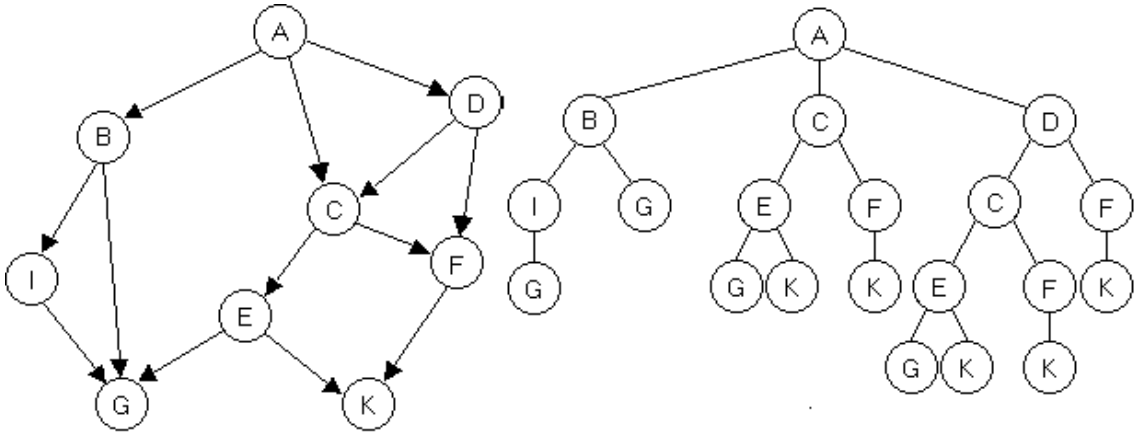
Trong nhiều vấn đề, dù chúng ta phát triển các trạng thái theo hệ thống nào (theo bề rộng hoặc theo độ sâu) thì số lượng các trạng thái được sinh ra trước khi ta gặp trạng thái đích thường là cực kỳ lớn. Do đó các thuật toán tìm kiếm mù kém hiệu quả, đòi hỏi rất nhiều không gian và thời gian. Trong thực tế, nhiều vấn đề không thể giải quyết được bằng tìm kiếm mù.

- Tìm kiếm kinh nghiệm (tìm kiếm heuristic). Trong rất nhiều vấn đề, chúng ta có thể dựa vào sự hiểu biết của chúng ta về vấn đề, dựa vào kinh nghiệm, trực giác, để đánh giá các trạng thái. Sử dụng sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm: trong quá trình phát triển các trạng thái, ta sẽ chọn trong số các trạng thái chờ phát triển, trạng thái được đánh giá là tốt nhất để phát triển. Do đó tốc độ tìm kiếm sẽ nhanh hơn. Các phương pháp tìm kiếm dựa vào sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm gọi chung là các phương pháp tìm kiếm kinh nghiệm.

Như vậy chiến lược tìm kiếm được xác định bởi chiến lược chọn trạng thái để phát triển ở mỗi bước. Trong tìm kiếm mù, ta chọn trạng thái để phát triển theo thứ tự mà chúng được sinh ra; còn trong tìm kiếm kinh nghiệm ta chọn trạng thái dựa vào sự đánh giá các trạng thái.

CÂY TÌM KIẾM

Chúng ta có thể nghĩ đến quá trình tìm kiếm như quá trình xây dựng ***cây tìm kiếm***. Cây tìm kiếm là cây mà các đỉnh được gắn bởi các trạng thái của không gian trạng thái. Gốc của cây tìm kiếm tương ứng với trạng thái ban đầu. Nếu một đỉnh ứng với trạng thái u, thì các đỉnh con của nó ứng với các trạng thái v kề u. Hình 1.4a là đồ thị biểu diễn một không gian trạng thái với trạng thái ban đầu là A, hình 1.4b là cây tìm kiếm tương ứng với không gian trạng thái đó.



a) Đồ thị không gian trạng thái b) Cây tìm kiếm tương ứng

Hình 1.4

Mỗi chiến lược tìm kiếm trong không gian trạng thái tương ứng với một phương pháp xây dựng cây tìm kiếm. Quá trình xây dựng cây bắt đầu từ cây chỉ có một đỉnh là trạng thái ban đầu. Giả sử tới một bước nào đó trong chiến lược tìm kiếm, ta đã xây dựng được một cây nào đó, các lá của cây

tương ứng với các trạng thái chưa được phát triển. Bước tiếp theo phụ thuộc vào chiến lược tìm kiếm mà một đỉnh nào đó trong các lá được chọn để phát triển. Khi phát triển đỉnh đó, cây tìm kiếm được mở rộng bằng cách thêm vào các đỉnh con của đỉnh đó. Kỹ thuật tìm kiếm theo bề rộng (theo độ sâu) tương ứng với phương pháp xây dựng cây tìm kiếm theo bề rộng (theo độ sâu).

1.3. CÁC CHIẾN LƯỢC TÌM KIẾM MÙ

Trong mục này chúng ta sẽ trình bày hai chiến lược tìm kiếm mù: tìm kiếm theo bề rộng và tìm kiếm theo độ sâu. Trong tìm kiếm theo bề rộng, tại mỗi bước ta sẽ chọn trạng thái để phát triển là trạng thái được sinh ra trước các trạng thái chờ phát triển khác. Còn trong tìm kiếm theo độ sâu, trạng thái được chọn để phát triển là trạng thái được sinh ra sau cùng trong số các trạng thái chờ phát triển.

Chúng ta sử dụng danh sách L để lưu các trạng thái đã được sinh ra và chờ được phát triển. Mục tiêu của tìm kiếm trong không gian trạng thái là tìm đường đi từ trạng thái ban đầu tới trạng thái đích, do đó ta cần lưu lại vết của đường đi. Ta có thể sử dụng hàm father để lưu lại cha của mỗi đỉnh trên đường đi, ***father***(v) = u nếu cha của đỉnh v là u.

1.3.1. Tìm kiếm theo bề rộng

Thuật toán tìm kiếm theo bề rộng được mô tả bởi thủ tục sau:

```
PROCEDURE     BREADTH_FIRST_SEARCH;
BEGIN
1. KHỞI TẠO DANH SÁCH L CHỈ CHỨA TRẠNG THÁI BAN ĐẦU;
2. LOOP DO
    2.1 IF L RỖNG THEN
        {THÔNG BÁO TÌM KIẾM THẤT BẠI; STOP};
    2.2 LOẠI TRẠNG THÁI U Ở ĐẦU DANH SÁCH L;
    2.3 IF U LÀ TRẠNG THÁI KẾT THÚC THEN
        {THÔNG BÁO TÌM KIẾM THÀNH CÔNG; STOP};
    2.4 FOR MỖI TRẠNG THÁI V KỀ U DO {
        ĐẶT V VÀO CUỐI DANH SÁCH L;
        FATHER(V) ← U}
END;
```

Chúng ta có một số nhận xét sau đây về thuật toán tìm kiếm theo bề rộng:

- Trong tìm kiếm theo bề rộng, trạng thái nào được sinh ra trước sẽ được phát triển trước, do đó danh sách L được xử lý như hàng đợi. Trong bước 2.3, ta cần kiểm tra xem u có là trạng thái kết thúc hay không. Nói chung các trạng thái kết thúc được xác định bởi một số điều kiện nào đó, khi đó ta cần kiểm tra xem u có thỏa mãn các điều kiện đó hay không.
- Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái ban đầu tới trạng thái đích), thì thuật toán tìm kiếm theo bề rộng sẽ tìm ra nghiệm, đồng thời đường đi tìm được sẽ là ngắn nhất. Trong trường hợp bài toán vô nghiệm và không gian trạng thái hữu hạn, thuật toán sẽ dừng và cho thông báo vô nghiệm.

ĐÁNH GIÁ TÌM KIẾM THEO BỀ RỘNG

Bây giờ ta đánh giá thời gian và bộ nhớ mà tìm kiếm theo bề rộng đòi hỏi. Giả sử rằng, mỗi trạng thái khi được phát triển sẽ sinh ra b trạng thái kề. Ta sẽ gọi b là ***nhân tố nhánh***. Giả sử rằng, nghiệm của bài toán là đường đi có độ dài d. Bởi vì nghiệm có thể được tìm ra tại một đỉnh bất kỳ ở mức d của cây tìm kiếm, do đó số đỉnh cần xem xét để tìm ra nghiệm là:

$$\mathbf{1 + b + b^2 + ... + b^{d-1} + k}$$

Trong đó k có thể là 1, 2,..., b^d. Do đó số lớn nhất các đỉnh cần xem xét là:

$$\mathbf{1 + b + b^2 + ... + b^d}$$

Như vậy, độ phức tạp thời gian của thuật toán tìm kiếm theo bề rộng là $O(b^d)$. Độ phức tạp không gian cũng là $O(b^d)$, bởi vì ta cần lưu vào danh sách L tất cả các đỉnh của cây tìm kiếm ở mức d, số các đỉnh này là b^d.

Để thấy rõ tìm kiếm theo bề rộng đòi hỏi thời gian và không gian lớn tới mức nào, ta xét trường hợp nhân tố nhánh b = 10 và độ sâu d thay đổi. Giả sử để phát hiện và kiểm tra 1000 trạng thái cần 1 giây, và lưu giữ 1 trạng thái cần 100 bytes. Khi đó thời gian và không gian mà thuật toán đòi hỏi được cho trong bảng sau:

Độ sâu d	Thời gian	Không gian
4	11 giây	1 megabyte
6	18 giây	111 megabytes
8	31 giờ	11 gigabytes
10	128 ngày	1 terabyte
12	35 năm	111 terabytes
14	3500 năm	11.111 terabytes

1.3.2. Tìm kiếm theo độ sâu

Như ta đã biết, tư tưởng của chiến lược tìm kiếm theo độ sâu là, tại mỗi bước, trạng thái được chọn để phát triển là trạng thái được sinh ra sau cùng trong số các trạng thái chờ phát triển. Do đó thuật toán tìm kiếm theo độ sâu là hoàn toàn tương tự như thuật toán tìm kiếm theo bề rộng, chỉ có một điều khác là, ta xử lý danh sách L các trạng thái chờ phát triển không phải như hàng đợi mà như ngăn xếp. Cụ thể là trong bước 2.4 của thuật toán tìm kiếm theo bề rộng, ta cần sửa lại là "Đặt v vào ***đầu*** danh sách L".

Sau đây chúng ta sẽ đưa ra các nhận xét so sánh hai chiến lược tìm kiếm mù:

- Thuật toán tìm kiếm theo bề rộng luôn luôn tìm ra nghiệm nếu bài toán có nghiệm. Song không phải với bất kỳ bài toán có nghiệm nào thuật toán tìm kiếm theo độ sâu cũng tìm ra nghiệm! Nếu bài toán có nghiệm và không gian trạng thái hữu hạn, thì thuật toán tìm kiếm theo độ sâu sẽ tìm ra nghiệm. Tuy nhiên, trong trường hợp không gian trạng thái vô hạn, thì có thể nó không tìm ra nghiệm, lý do là ta luôn luôn đi xuống theo độ sâu, nếu ta đi theo một nhánh vô hạn mà nghiệm không nằm trên nhánh đó thì thuật toán sẽ không dừng. Do đó người ta khuyên rằng, không nên áp dụng tìm kiếm theo dộ sâu cho các bài toán có cây tìm kiếm chứa các nhánh vô hạn.
- Độ phức tạp của thuật toán tìm kiếm theo độ sâu.

Giả sử rằng, nghiệm của bài toán là đường đi có độ dài d, cây tìm kiếm có nhân tố nhánh là b và có chiều cao là d. Có thể xảy ra, nghiệm là đỉnh ngoài cùng bên phải trên mức d của cây tìm kiếm, do đó độ phức tạp thời gian của tìm kiếm theo độ sâu trong trường hợp xấu nhất là $O(b^d)$, tức là cũng như tìm kiếm theo bề rộng. Tuy nhiên, trên thực tế đối với nhiều bài toán, tìm kiếm theo độ sâu thực sự nhanh hơn tìm kiếm theo bề rộng. Lý do là tìm kiếm theo bề rộng phải xem xét toàn bộ cây tìm kiếm tới mức d-1, rồi mới xem xét các đỉnh ở mức d. Còn trong tìm kiếm theo độ sâu, có thể ta chỉ cần xem xét một bộ phận nhỏ của cây tìm kiếm thì đã tìm ra nghiệm.

Để đánh giá độ phức tạp không gian của tìm kiếm theo độ sâu ta có nhận xét rằng, khi ta phát triển một đỉnh u trên cây tìm kiếm theo độ sâu, ta chỉ cần lưu các đỉnh chưa được phát triển mà chúng là các đỉnh con của các đỉnh nằm trên đường đi từ gốc tới đỉnh u. Như vậy đối với cây tìm kiếm có nhân tố nhánh b và độ sâu lớn nhất là d, ta chỉ cần lưu ít hơn db đỉnh. Do đó độ phức tạp không gian của tìm kiếm theo độ sâu là O(db), trong khi đó tìm kiếm theo bề rộng đòi hỏi không gian nhớ $O(b^d)$!

1.3.3. Các trạng thái lặp

Như ta thấy trong mục 1.2, cây tìm kiếm có thể chứa nhiều đỉnh ứng với cùng một trạng thái, các trạng thái này được gọi là trạng thái lặp. Chẳng hạn, trong cây tìm kiếm hình 4b, các trạng thái C, E, F là các trạng thái lặp. Trong đồ thị biểu diễn không gian trạng thái, các trạng thái lặp ứng với các đỉnh có nhiều đường đi dẫn tới nó từ trạng thái ban đầu. Nếu đồ thị có chu trình thì cây tìm kiếm sẽ chứa các nhánh với một số đỉnh lặp lại vô hạn lần. Trong các thuật toán tìm kiếm sẽ lãng phí rất nhiều thời gian để phát triển lại các trạng thái mà ta đã gặp và đã phát triển. Vì vậy trong quá trình tìm kiếm ta cần tránh phát sinh ra các trạng thái mà ta đã phát triển. Chúng ta có thể áp dụng một trong các giải pháp sau đây:

1. Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với cha của u.
2. Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường đi dẫn tới u.
3. Không sinh ra các đỉnh mà nó đã được sinh ra, tức là chỉ sinh ra các đỉnh mới.

Hai giải pháp đầu dễ cài đặt và không tốn nhiều không gian nhớ, tuy nhiên các giải pháp này không tránh được hết các trạng thái lặp.

Để thực hiện giải pháp thứ 3 ta cần lưu các trạng thái đã phát triển vào tập Q, lưu các trạng thái chờ phát triển vào danh sách L. Đương nhiên, trạng

thái v lần đầu được sinh ra nếu nó không có trong Q và L. Việc lưu các trạng thái đã phát triển và kiểm tra xem một trạng thái có phải lần đầu được sinh ra không đòi hỏi rất nhiều không gian và thời gian. Chúng ta có thể cài đặt tập Q bởi bảng băm (xem [8]).

1.3.4. Tìm kiếm sâu lặp

Như chúng ta đã nhận xét, nếu cây tìm kiếm chứa nhánh vô hạn, khi sử dụng tìm kiếm theo độ sâu, ta có thể mắc kẹt ở nhánh đó và không tìm ra nghiệm. Để khắc phục hoàn cảnh đó, ta tìm kiếm theo độ sâu chỉ tới mức d nào đó; nếu không tìm ra nghiệm, ta tăng độ sâu lên d+1 và lại tìm kiếm theo độ sâu tới mức d+1. Quá trình trên được lặp lại với d lần lượt là 1, 2,... dến một độ sâu max nào đó. Như vậy, thuật toán tìm kiếm sâu lặp (iterative deepening search) sẽ sử dụng thủ tục tìm kiếm sâu hạn chế (depth_limited search) như thủ tục con. Đó là thủ tục tìm kiếm theo độ sâu, nhưng chỉ đi tới độ sâu d nào đó rồi quay lên.

Trong thủ tục tìm kiếm sâu hạn chế, d là tham số độ sâu, hàm depth ghi lại độ sâu của mỗi đỉnh

```
procedure Depth_Limited_Search(d);
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu u0;
  depth(u0)← 0;
2. loop do
     2.1 if L rỗng then
           {thông báo thất bại; stop};
     2.2 Loại trạng thái u ở đầu danh sách L;
     2.3 if u là trạng thái kết thúc then
           {thông báo thành công; stop};
     2.4 if depth(u) <= d then
           for mỗi trạng thái v kề u do
                 {Đặt v vào đầu danh sách L;
                 depth(v)← depth(u) + 1};
end;

procedure DEPTH_DEEPENING_SEARCH;
begin
     for D ← 0 to MAX do
           {Depth_Limited_Search(d);
```

if *thành công* **then exit**}

end;

Kỹ thuật tìm kiếm sâu lặp kết hợp được các ưu điểm của tìm kiếm theo bề rộng và tìm kiếm theo độ sâu. Chúng ta có một số nhận xét sau:

- Cũng như tìm kiếm theo bề rộng, tìm kiếm sâu lặp luôn luôn tìm ra nghiệm (nếu bài toán có nghiệm), miễn là ta chọn độ sâu max đủ lớn.
- Tìm kiếm sâu lặp chỉ cần không gian nhớ như tìm kiếm theo độ sâu.
- Trong tìm kiếm sâu lặp, ta phải phát triển lặp lại nhiều lần cùng một trạng thái. Điều đó làm cho ta có cảm giác rằng, tìm kiếm sâu lặp lãng phí nhiều thời gian. Thực ra thời gian tiêu tốn cho phát triển lặp lại các trạng thái là không đáng kể so với thời gian tìm kiếm theo bề rộng. Thật vậy, mỗi lần gọi thủ tục tìm kiếm sâu hạn chế tới mức d, nếu cây tìm kiếm có nhân tố nhánh là b, thì số đỉnh cần phát triển là:

  $\mathbf{1 + b + b^2 + ... + b^d}$

Nếu nghiệm ở độ sâu d, thì trong tìm kiếm sâu lặp, ta phải gọi thủ tục tìm kiếm sâu hạn chế với độ sâu lần lượt là 0, 1, 2,..., d. Do đó các đỉnh ở mức 1 phải phát triển lặp d lần, các đỉnh ở mức 2 lặp d-1 lần,..., các đỉnh ở mức d lặp 1 lần. Như vậy tổng số đỉnh cần phát triển trong tìm kiếm sâu lặp là:

$\mathbf{(d+1)1 + db + (d-1)b^2 + ... + 2b^{d-1} + 1b^d}$

Do đó thời gian tìm kiếm sâu lặp là $O(b^d)$.

Tóm lại, tìm kiếm sâu lặp có độ phức tạp thời gian là $O(b^d)$ (như tìm kiếm theo bề rộng), và có độ phức tạp không gian là O(bd) (như tìm kiếm theo độ sâu). Nói chung, chúng ta nên áp dụng tìm kiếm sâu lặp cho các vấn đề có không gian trạng thái lớn và độ sâu của nghiệm không biết trước.

1.4. QUY VẤN ĐỀ VỀ CÁC VẤN ĐỀ CON. TÌM KIẾM TRÊN ĐỒ THỊ VÀ/HOẶC.

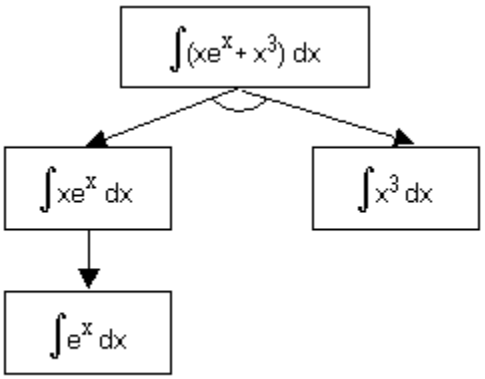
1.4.1. Quy vấn đề về các vấn đề con:

Trong mục 1.1, chúng ta đã nghiên cứu việc biểu diễn vấn đề thông qua các trạng thái và các toán tử. Khi đó việc tìm nghiệm của vấn đề được quy về việc tìm đường trong không gian trạng thái. Trong mục này chúng ta sẽ nghiên cứu một phương pháp luận khác để giải quyết vấn đề, dựa trên việc quy vấn đề về các vấn đề con. Quy vấn đề về các vấn đề con (còn gọi là rút gọn vấn đề) là một phương pháp được sử dụng rộng rãi nhất để giải quyết

các vấn đề. Trong đời sống hàng ngày, cũng như trong khoa học kỹ thuật, mỗi khi gặp một vấn đề cần giải quyết, ta vẫn thường cố gắng tìm cách đưa nó về các vấn đề đơn giản hơn. Quá trình rút gọn vấn đề sẽ được tiếp tục cho tới khi ta dẫn tới các vấn đề con có thể giải quyết được dễ dàng. Sau đây chúng ta xét một số vấn đề.

VẤN ĐỀ TÍNH TÍCH PHÂN BẤT ĐỊNH

Giả sử ta cần tính một tích phân bất định, chẳng hạn $\int(xe^x + x^3)dx$. Quá trình chúng ta vẫn thường làm để tính tích phân bất định là như sau. Sử dụng các quy tắc tính tích phân (quy tắc tính tích phân của một tổng, quy tắc tính tích phân từng phần...), sử dụng các phép biến đổi biến số, các phép biến đổi các hàm (chẳng hạn, các phép biến đổi lượng giác),... để đưa tích phân cần tính về tích phân của các hàm số sơ cấp mà chúng ta đã biết cách tính. Chẳng hạn, đối với tích phân $\int(xe^x + x^3)dx$, áp dụng quy tắc tích phân của tổng ta đưa về hai tích phân $\int xe^x dx$ và $\int x^3 dx$. Áp dụng quy tắc tích phân từng phần ta đưa tích phân $\int xe^x dx$ về tích phân $\int e^x dx$. Quá trình trên có thể biểu diễn bởi đồ thị trong hình 1.5.



Hình 1.5 Quy một số tích phân về các tích phân cơ bản.

Các tích phân $\int e^x dx$ và $\int x^3 dx$ là các tích phân cơ bản đã có trong bảng tích phân. Kết hợp các kết quả của các tích phân cơ bản, ta nhận được kết quả của tích phân đã cho.

Chúng ta có thể biểu diễn việc quy một vấn đề về các vấn đề cơ bởi các trạng thái và các toán tử. Ở đây, bài toán cần giải là trạng thái ban đầu. Mỗi cách quy bài toán về các bài toán con được biểu diễn bởi một toán tử, toán tử $A \rightarrow B, C$ biểu diễn việc quy bài toán A về hai bài toán B và C. Chẳng hạn, đối với bài toán tính tích phân bất định, ta có thể xác định các toán tử dạng:

$\int (f_1 + f_2)\, dx \rightarrow \int f_1\, dx, \int f_2\, dx$ và $\int u\, dv \rightarrow \int v\, du$

Các trạng thái kết thúc là các bài toán sơ cấp (các bài toán đã biết cách giải). Chẳng hạn, trong bài toán tính tích phân, các tích phân cơ bản là các trạng thái kết thúc. Một điều cần lưu ý là, trong không gian trạng thái biểu diễn việc quy vấn đề về các vấn đề con, các toán tử có thể là đa trị, nó biến đổi một trạng thái thành nhiều trạng thái khác.

VẤN ĐỀ TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ GIAO THÔNG

Bài toán này đã được phát biểu như bài toán tìm đường đi trong không gian trạng thái (xem 1.1), trong đó mỗi trạng thái ứng với một thành phố, mỗi toán tử ứng với một con đường, nối thành phố này với thành phố khác. Bây giờ ta đưa ra một cách biểu diễn khác dựa trên việc quy vấn đề về các vấn đề con. Giả sử ta có bản đồ giao thông trong một vùng lãnh thổ (xem hình 1.6). Giả sử ta cần tìm đường đi từ thành phố A tới thành phố B. Có con sông chảy qua hai thành phố E và G và có cầu qua sông ở mỗi thành phố đó. Mọi đường đi từ A đến B chỉ có thể qua E hoặc G. Như vậy bài toán tìm đường đi từ A đến B được quy về:

1) Bài toán tìm đường đi từ A đến B qua E (hoặc)

2) Bài toán tìm đường đi từ A đến b qua G.

Mỗi một trong hai bài toán trên lại có thể phân nhỏ như sau:

1) Bài toán tìm đường đi từ A đến B qua E được quy về:

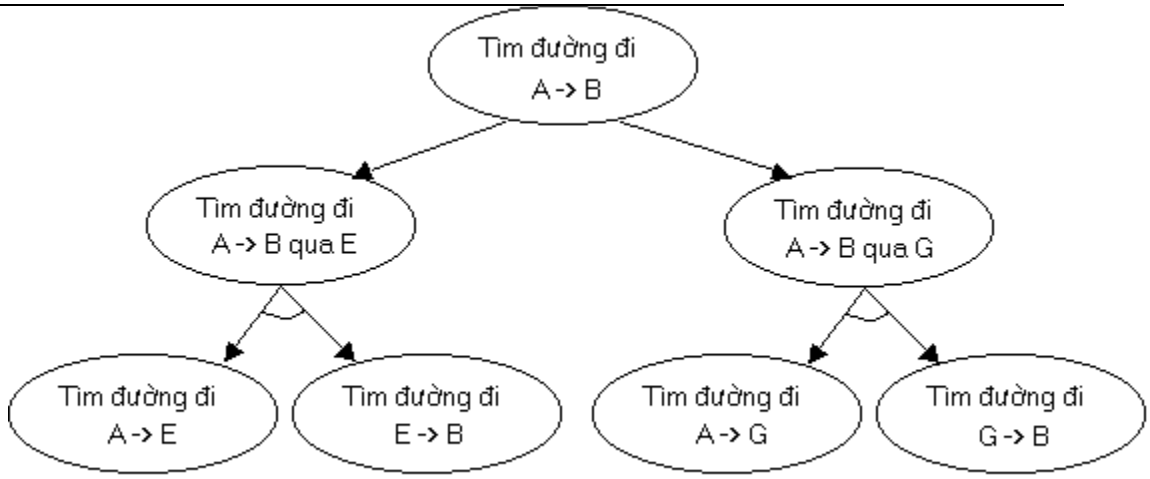
1.1 Tìm đường đi từ A đến E (và)

1.2 Tìm đường đi từ E đến B.

2) Bài toán tìm đường đi từ A đến B qua G được quy về:

2.1 Tìm đường đi từ A đến G (và)
2.2 Tìm đường đi từ G đến B.

Quá trình rút gọn vấn đề như trên có thể biểu diễn dưới dạng đồ thị (đồ thị và/hoặc) trong hình 1.7. Ở đây mỗi bài toán tìm đường đi từ một thành phố tới một thành phố khác ứng với một trạng thái. Các trạng thái kết thúc là các trạng thái ứng với các bài toán tìm đường đi, chẳng hạn từ A đến C, hoặc từ D đến E, bởi vì đã có đường nối A với C, nối D với E.

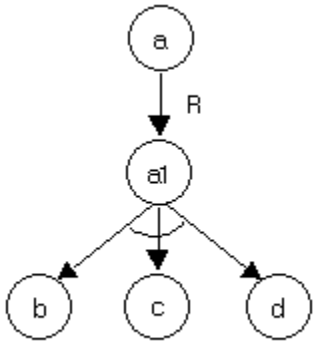


Hình 1.7 Đồ thị và/hoặc của vấn đề tìm đường đi.

1.4.2. Đồ thị *và/hoặc*

Không gian trạng thái mô tả việc quy vấn đề về các vấn đề con có thể biểu diễn dưới dạng đồ thị định hướng đặc biệt được gọi là đồ thị và/hoặc. Đồ thị này được xây dựng như sau:

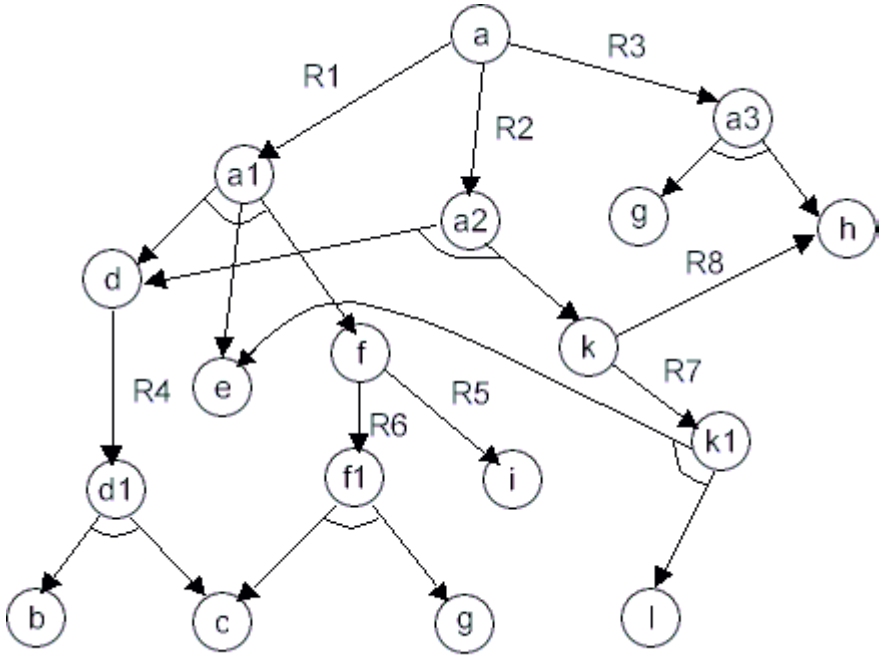
Mỗi bài toán ứng với một đỉnh của đồ thị. Nếu có một toán tử quy một bài toán về một bài toán khác, chẳng hạn R: a →b, thì trong đồ thị sẽ có cung gán nhãn đi từ đỉnh a tới đỉnh b. Đối với mỗi toán tử quy một bài toán về một số bài toán con, chẳng hạn R: a →b, c, d ta đưa vào một đỉnh mới a_1, đỉnh này biểu diễn tập các bài toán con {b, c, d} và toán tử R: a →b, c, d được biểu diễn bởi đồ thị hình 1.8.



Hình 1.8 Đồ thị biểu diễn toán tử R : a->b, c, d

Ví dụ: Giả sử chúng ta có không gian trạng thái sau:

- Trạng thái ban đầu (bài toán cần giải) là a.



Hình 1.9 Một đồ thị và/hoặc

- Tập các toán tử quy gồm:

R_1: a →d, e, f

R_2: a →d, k

R_3: a →g, h

R_4: d →b, c

R_5: f →i

R_6: f →c, j

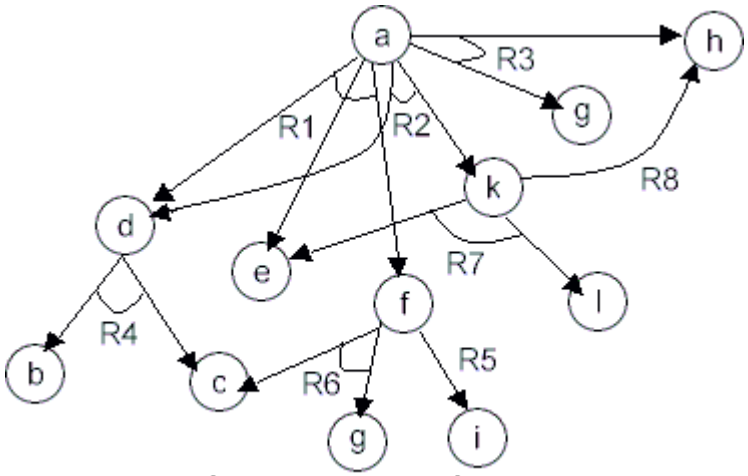
R_7: k →e, l

R_8: k →h

- Tập các trạng thái kết thúc (các bài toán sơ cấp) là T = {b, c, e, j, l}.

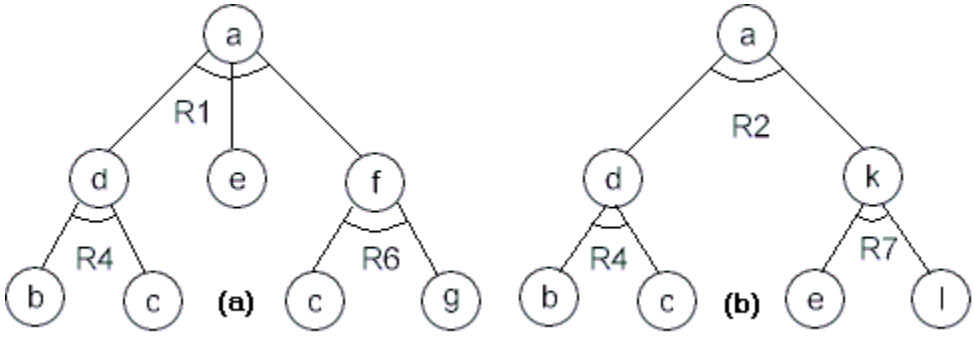
Không gian trạng thái trên có thể biểu diễn bởi đồ thị và/hoặc trong hình 1.9. Trong đồ thị đó, các đỉnh, chẳng hạn a_1, a_2, a_3 được gọi là đỉnh ***và***, các đỉnh chẳng hạn a, f, k được gọi là đỉnh ***hoặc***. Lý do là, đỉnh a_1 biểu diễn tập các bài toán {d, e, f} và a_1 được giải quyết nếu d và e và f được giải quyết. Còn tại đỉnh a, ta có các toán tử R_1, R_2, R_3 quy bài toán a về các bài toán con khác nhau, do đó a được giải quyết nếu hoặc a_1 = {d, e, f}, hoặc a_2 = {d, k}, hoặc a_3 = {g, h} được giải quyết.

Người ta thường sử dụng đồ thị và/hoặc ở dạng rút gọn. Chẳng hạn, đồ thị và/hoặc trong hình 1.9 có thể rút gọn thành đồ thị trong hình 1.10. Trong đồ thị rút gọn này, ta sẽ nói chẳng hạn d, e, f là các đỉnh kề đỉnh a theo toán tử R_1, còn d, k là các đỉnh kề a theo toán tử R_2.



Hình 1.10 Đồ thị rút gọn của đồ thị trong hình 1.9

Khi đã có các toán tử rút gọn vấn đề, thì bằng cách áp dụng liên tiếp các toán tử, ta có thể đưa bài toán cần giải về một tập các bài toán con. Chẳng hạn, trong ví dụ trên nếu ta áp dụng các toán tử R_1, R_4, R_6, ta sẽ quy bài toán a về tập các bài toán con {b, c, e, f}, tất cả các bài toán con này đều là sơ cấp. Từ các toán tử R_1, R_4 và R_6 ta xây dựng được một cây trong hình 1.11a, cây này được gọi là cây nghiệm. Cây nghiệm được định nghĩa như sau:



Hình 1.11 Các cây nghiệm

Cây nghiệm là một cây, trong đó:

- Gốc của cây ứng với bài toán cần giải.
- Tất cả các lá của cây là các đỉnh kết thúc (đỉnh ứng với các bài toán sơ cấp).

- Nếu u là đỉnh trong của cây, thì các đỉnh con của u là tất cả các đỉnh kề u theo một toán tử nào đó.

Các đỉnh của đồ thị và/hoặc sẽ được gắn nhãn giải được hoặc không giải được.

Các đỉnh ***giải được*** được xác định đệ quy như sau:

- Các đỉnh kết thúc là các đỉnh ***giải được***.
- Nếu u không phải là đỉnh kết thúc, nhưng có một toán tử R sao cho tất cả các đỉnh kề u theo R đều giải được thì u ***giải được***.

Các đỉnh ***không giải được*** được xác định đệ quy như sau:

- Các đỉnh không phải là đỉnh kết thúc và không có đỉnh kề, là các đỉnh ***không giải được***.
- Nếu u không phải là đỉnh kết thúc và với mọi toán tử R áp dụng được tại u đều có một đỉnh v kề u theo R không giải được, thì u ***không giải được***.

Ta có nhận xét rằng, nếu bài toán a ***giải được*** thì sẽ có một cây nghiệm gốc a, và ngược lại nếu có một cây nghiệm gốc a thì a ***giải được***. Hiển nhiên là, một bài toán giải được có thể có nhiều cây nghiệm, mỗi cây nghiệm biểu diễn một cách giải bài toán đó. Chẳng hạn trong ví dụ đã nêu, bài toán a có hai cây nghiệm trong hình 1.11.

Thứ tự giải các bài toán con trong một cây nghiệm là như sau. Bài toán ứng với đỉnh u chỉ được giải sau khi tất cả các bài toán ứng với các đỉnh con của u đã được giải. Chẳng hạn, với cây nghiệm trong hình 1.11a, thứ tự giải các bài toán có thể là b, c, d, j, f, e, a. Ta có thể sử dụng thủ tục sắp xếp topo (xem [8]) để sắp xếp thứ tự các bài toán trong một cây nghiệm. Đương nhiên ta cũng có thể giải quyết đồng thời các bài toán con ở cùng một mức trong cây nghiệm.

Vấn đề của chúng ta bây giờ là, tìm kiếm trên đồ thị và/hoặc để xác định được đỉnh ứng với bài toán ban đầu là giải được hay không giải được, và nếu nó giải được thì xây dựng một cây nghiệm cho nó.

1.4.3. Tìm kiếm trên đồ thị và/hoặc

Ta sẽ sử dụng kỹ thuật tìm kiếm theo độ sâu trên đồ thị và/hoặc để đánh dấu các đỉnh. Các đỉnh sẽ được đánh dấu giải được hoặc không giải được theo định nghĩa đệ quy về đỉnh giải được và không giải được. Xuất phát từ đỉnh ứng với bài toán ban đầu, đi xuống theo độ sâu, nếu gặp đỉnh u

là đỉnh kết thúc thì nó được đánh dấu giải được. Nếu gặp đỉnh u không phải là đỉnh kết thúc và từ u không đi tiếp được, thì u được đánh dấu không giải được. Khi đi tới đỉnh u, thì từ u ta lần lượt đi xuống các đỉnh v kề u theo một toán tử R nào đó. Nếu đánh dấu được một đỉnh v không giải được thì không cần đi tiếp xuống các đỉnh v còn lại. Tiếp tục đi xuống các đỉnh kề u theo một toán tử khác. Nếu tất cả các đỉnh kề u theo một toán tử nào đó được đánh dấu giải được thì u sẽ được đánh dấu giải được và quay lên cha của u. Còn nếu từ u đi xuống các đỉnh kề nó theo mọi toán tử đều gặp các đỉnh kề được đánh dấu không giải được, thì u được đánh dấu không giải được và quay lên cha của u.

Ta sẽ biểu diễn thủ tục tìm kiếm theo độ sâu và đánh dấu các đỉnh đã trình bày trên bởi hàm đệ quy Solvable(u). Hàm này nhận giá trị true nếu u giải được và nhận giá trị ***false*** nếu u không giải được. Trong hàm Solvable(u), ta sẽ sử dụng:

- Biến Ok. Với mỗi toán tử R áp dụng được tại u, biến Ok nhận giá trị ***true*** nếu tất cả các đỉnh v kề u theo R đều giải được, và Ok nhận giá trị ***false*** nếu có một đỉnh v kề u theo R không giải được.
- Hàm Operator(u) ghi lại toán tử áp dụng thành công tại u, tức là Operator(u) = R nếu mọi đỉnh v kề u theo R đều giải được.

```
function Solvable(u);
begin
1. if u là đỉnh kết thúc then
        {Solvable ← true; stop};
2. if u không là đỉnh kết thúc và không có đỉnh kề then
        {Solvable(u) ← false; stop};
3. for mỗi toán tử R áp dụng được tại u do
        {Ok ← true;
        for mỗi v kề u theo R do
                if Solvable(v) = false then
                        {Ok ← false; exit};
                if Ok then
                        {Solvable(u) ← true;
                        Operator(u) ← R; stop}}
4. Solvable(u) ← false;
end;
```

Nhận xét

- Hoàn toàn tương tự như thuật toán tìm kiếm theo độ sâu trong không gian trạng thái (mục 1.3.2), thuật toán tìm kiếm theo độ sâu trên đồ thị và/hoặc sẽ xác định được bài toán ban đầu là giải được hay không giải được, nếu cây tìm kiếm không có nhánh vô hạn. Nếu cây tìm kiếm có nhánh vô hạn thì chưa chắc thuật toán đã dừng, vì có thể nó bị xa lầy khi đi xuống nhánh vô hạn. Trong trường hợp này ta nên sử dụng thuật toán tìm kiếm sâu lặp (mục 1.3.3).

Nếu bài toán ban đầu giải được, thì bằng cách sử dụng hàm Operator ta sẽ xây dựng được cây nghiệm.

CHƯƠNG 2

CÁC CHIẾN LƯỢC TÌM KIẾM KINH NGHIỆM

Trong chương I, chúng ta đã nghiên cứu việc biểu diễn vấn đề trong không gian trạng thái và các kỹ thuật tìm kiếm mù. Các kỹ thuật tìm kiếm mù rất kém hiệu quả và trong nhiều trường hợp không thể áp dụng được. Trong chương này, chúng ta sẽ nghiên cứu các phương pháp tìm kiếm kinh nghiệm (tìm kiếm heuristic), đó là các phương pháp sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm.

2.1. HÀM ĐÁNH GIÁ VÀ TÌM KIẾM KINH NGHIỆM:

Trong nhiều vấn đề, ta có thể sử dụng kinh nghiệm, tri thức của chúng ta về vấn đề để đánh giá các trạng thái của vấn đề. Với mỗi trạng thái u, chúng ta sẽ xác định một giá trị số h(u), số này đánh giá "sự gần đích"của trạng thái u. Hàm h(u) được gọi là ***hàm đánh giá***. Chúng ta sẽ sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm. Trong quá trình tìm kiếm, tại mỗi bước ta sẽ chọn trạng thái để phát triển là trạng thái có giá trị hàm đánh giá nhỏ nhất, trạng thái này được xem là trạng thái có nhiều hứa hẹn nhất dẫn tới đích.

Các kỹ thuật tìm kiếm sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm được gọi chung là các kỹ thuật tìm kiếm kinh nghiệm (heuristic search). Các giai đoạn cơ bản để giải quyết vấn đề bằng tìm kiếm kinh nghiệm như sau:

1. Tìm biểu diễn thích hợp mô tả các trạng thái và các toán tử của vấn đề.
2. Xây dựng hàm đánh giá.
3. Thiết kế chiến lược chọn trạng thái để phát triển ở mỗi bước.

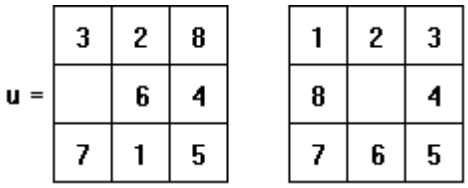
HÀM ĐÁNH GIÁ

Trong tìm kiếm kinh nghiệm, hàm đánh giá đóng vai trò cực kỳ quan trọng. Chúng ta có xây dựng được hàm đánh giá cho ta sự đánh giá đúng các trạng thái thì tìm kiếm mới hiệu quả. Nếu hàm đánh giá không chính xác, nó có thể dẫn ta đi chệch hướng và do đó tìm kiếm kém hiệu quả.

Hàm đánh giá được xây dựng tùy thuộc vào vấn đề. Sau đây là một số ví dụ về hàm đánh giá:

- Trong bài toán tìm kiếm đường đi trên bản đồ giao thông, ta có thể lấy độ dài của đường chim bay từ một thành phố tới một thành phố đích làm giá trị của hàm đánh giá.
- Bài toán 8 số. Chúng ta có thể đưa ra hai cách xây dựng hàm đánh giá.

Hàm h_1: Với mỗi trạng thái u thì $h_1(u)$ là số quân không nằm đúng vị trí của nó trong trạng thái đích. Chẳng hạn trạng thái đích ở bên phải hình 2.1, và u là trạng thái ở bên trái hình 2.1, thì $h_1(u) = 4$, vì các quân không



$h_1(u) = 4$ $h_2(u) = 9$

Hình 2.1 Đánh giá trạng thái u.

đúng vị trí là 3, 8, 6 và 1.

Hàm h_2: $h_2(u)$ là tổng khoảng cách giữa vị trí của các quân trong trạng thái u và vị trí của nó trong trạng thái đích. Ở đây khoảng cách được hiểu là số ít nhất các dịch chuyển theo hàng hoặc cột để đưa một quân tới vị trí của nó trong trạng thái đích. Chẳng hạn với trạng thái u và trạng thái đích như trong hình 2.1, ta có:

$$\mathbf{h_2(u) = 2 + 3 + 1 + 3 = 9}$$

Vì quân 3 cần ít nhất 2 dịch chuyển, quân 8 cần ít nhất 3 dịch chuyển, quân 6 cần ít nhất 1 dịch chuyển và quân 1 cần ít nhất 3 dịch chuyển.

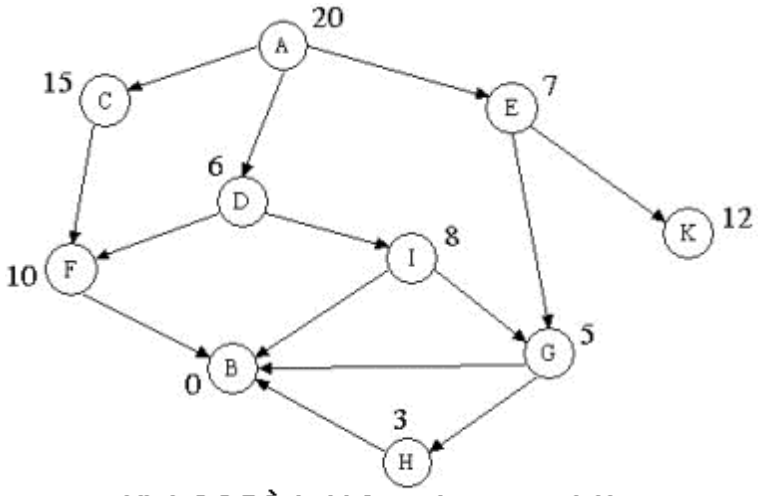
Hai chiến lược tìm kiếm kinh nghiệm quan trọng nhất là tìm kiếm tốt nhất - đầu tiên (best-first search) và tìm kiếm leo đồi (hill-climbing search). Có thể xác định các chiến lược này như sau:

Tìm kiếm tốt nhất đầu tiên = Tìm kiếm theo bề rộng + Hàm đánh giá

Tìm kiếm leo đồi = Tìm kiếm theo độ sâu + Hàm đánh giá

Chúng ta sẽ lần lượt nghiên cứu các kỹ thuật tìm kiếm này trong các mục sau.

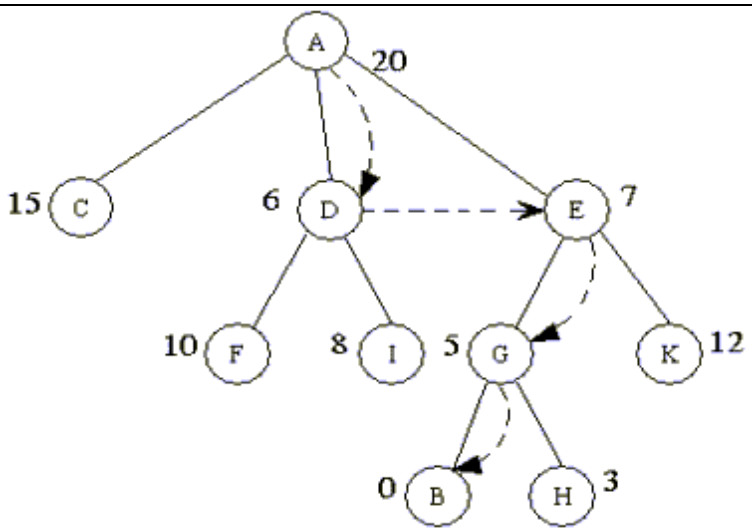
2.2. TÌM KIẾM TỐT NHẤT - ĐẦU TIÊN:



Hình 2.2 Đồ thị không gian trạng thái

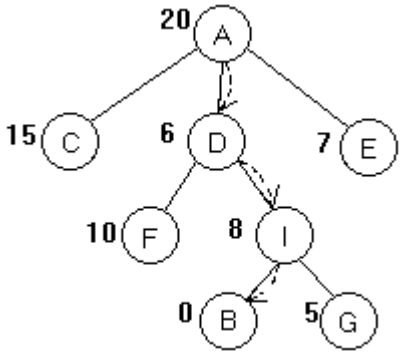
Tìm kiếm tốt nhất - đầu tiên (best-first search) là tìm kiếm theo bề rộng được hướng dẫn bởi hàm đánh giá. Nhưng nó khác với tìm kiếm theo bề rộng ở chỗ, trong tìm kiếm theo bề rộng ta lần lượt phát triển tất cả các đỉnh ở mức hiện tại để sinh ra các đỉnh ở mức tiếp theo, còn trong tìm kiếm tốt nhất - đầu tiên ta chọn đỉnh để phát triển là đỉnh tốt nhất được xác định bởi hàm đánh giá (tức là đỉnh có giá trị hàm đánh giá là nhỏ nhất), đỉnh này có thể ở mức hiện tại hoặc ở các mức trên.

Ví dụ: Xét không gian trạng thái được biểu diễn bởi đồ thị trong hình 2.2, trong đó trạng thái ban đầu là A, trạng thái kết thúc là B. Giá trị của hàm đánh giá là các số ghi cạnh mỗi đỉnh. Quá trình tìm kiếm tốt nhất - đầu tiên diễn ra như sau: Đầu tiên phát triển đỉnh A sinh ra các đỉnh kề là C, D và E. Trong ba đỉnh này, đỉnh D có giá trị hàm đánh giá nhỏ nhất, nó được chọn để phát triển và sinh ra F, I. Trong số các đỉnh chưa được phát triển C, E, F, I thì đỉnh E có giá trị đánh giá nhỏ nhất, nó được chọn để phát triển và sinh ra các đỉnh G, K. Trong số các đỉnh chưa được phát triển thì G tốt nhất, phát triển G sinh ra B, H. Đến đây ta đã đạt tới trạng thái kết thúc. Cây tìm kiếm tốt nhất - đầu tiên được biểu diễn trong hình 2.3.



Hình 2.3 Cây tìm kiếm tốt nhất - đầu tiên

Sau đây là thủ tục tìm kiếm tốt nhất - đầu tiên. Trong thủ tục này, chúng ta sử dụng danh sách L để lưu các trạng thái chờ phát triển, danh sách được sắp theo thứ tự tăng dần của hàm đánh giá sao cho trạng thái có giá trị



Hình 2.4 Cây tìm kiếm leo đồi

hàm đánh giá nhỏ nhất ở đầu danh sách.

procedure *Best_First_Search*;
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
2. **loop do**
2.1 ***if*** *L rỗng* ***then***
*{thông báo thất bại****; stop};***
2.2 Loại trạng thái u ở đầu danh sách L;
2.3 ***if*** *u là trạng thái kết thúc* ***then***
*{thông báo thành công****; stop}***
2.4 ***for*** *mỗi trạng thái v kề u* ***do***

Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần của hàm đánh giá;

end;

2.3. TÌM KIẾM LEO ĐỒI:

Tìm kiếm leo đồi (hill-climbing search) là tìm kiếm theo độ sâu được hướng dẫn bởi hàm đánh giá. Song khác với tìm kiếm theo độ sâu, khi ta phát triển một đỉnh u thì bước tiếp theo, ta chọn trong số các đỉnh con của u, đỉnh có nhiều hứa hẹn nhất để phát triển, đỉnh này được xác định bởi hàm đánh giá.

Ví dụ: Ta lại xét đồ thị không gian trạng thái trong hình 2.2. Quá trình tìm kiếm leo đồi được tiến hành như sau. Đầu tiên phát triển đỉnh A sinh ra các đỉnh con C, D, E. Trong các đỉnh này chọn D để phát triển (vì h(D)= 6 nhỏ nhất),sinh ra F, I. Trong hai đỉnh này, ta chọn I ddeer phát triển, và nó sinh ra các đỉnh con B, G. Quá trình tìm kiếm kết thúc. Cây tìm kiếm leo đồi được cho trong hình 2.4.

Trong thủ tục tìm kiếm leo đồi được trình bày dưới đây, ngoài danh sách L lưu các trạng thái chờ được phát triển, chúng ta sử dụng danh sách L_1 để lưu giữ tạm thời các trạng thái kề trạng thái u, khi ta phát triển u. Danh sách L_1 được sắp xếp theo thứ tự tăng dần của hàm đánh giá, rồi được chuyển vào danh sách L sao trạng thái tốt nhất kề u đứng ở danh sách L.

procedure Hill_Climbing_Search;

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

2. **loop** ***do***

2.1 ***if*** *L rỗng* ***then***

*{thông báo thất bại****; stop};***

2.2 Loại trạng thái u ở đầu danh sách L;

2.3 if *u là trạng thái kết thúc* **then**

{*thông báo thành công***; stop};**

2.3 ***for*** *mỗi trạng thái v kề u* ***do*** *đặt v vào* L_1***;***

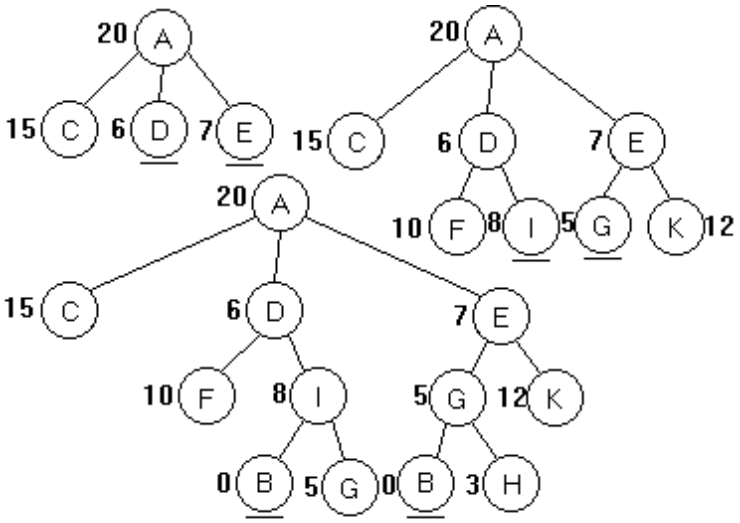
2.5 Sắp xếp L_1 *theo thứ tự tăng dần của hàm đánh giá;*

2.6 Chuyển danh sách L_1 *vào đầu danh sách L;*

end;

2.4. TÌM KIẾM BEAM

Tìm kiếm beam (beam search) giống như tìm kiếm theo bề rộng, nó phát triển các đỉnh ở một mức rồi phát triển các đỉnh ở mức tiếp theo. Tuy nhiên, trong tìm kiếm theo bề rộng, ta phát triển tất cả các đỉnh ở một mức, còn trong tìm kiếm beam, ta hạn chế chỉ phát triển k đỉnh tốt nhất (các đỉnh này được xác định bởi hàm đánh giá). Do đó trong tìm kiếm beam, ở bất kỳ mức nào cũng chỉ có nhiều nhất k đỉnh được phát triển, trong khi tìm kiếm theo bề rộng, số đỉnh cần phát triển ở mức d là b^d (b là nhân tố nhánh).



Hình 2.5 Cây tìm kiếm beam.

Ví dụ: Chúng ta lại xét đồ thị không gian trạng thái trong hình 2.2. Chọn k = 2. Khi đó cây tìm kiếm beam được cho như hình 2.5. Các đỉnh được gạch dưới là các đỉnh được chọn để phát triển ở mỗi mức.

CHƯƠNG 3

CÁC CHIẾN LƯỢC TÌM KIẾM TỐI ƯU

Vấn đề tìm kiếm tối ưu, một cách tổng quát, có thể phát biểu như sau. Mỗi đối tượng x trong không gian tìm kiếm được gắn với một số đo giá trị của đối tượng đó f(x), mục tiêu của ta là tìm đối tượng có giá trị f(x) lớn nhất (hoặc nhỏ nhất) trong không gian tìm kiếm. Hàm f(x) được gọi là hàm mục tiêu. Trong chương này chúng ta sẽ nghiên cứu các thuật toán tìm kiếm sau:

- Các kỹ thuật tìm đường đi ngắn nhất trong không gian trạng thái: Thuật toán A*, thuật toán nhánh_và_cận.
- Các kỹ thuật tìm kiếm đối tượng tốt nhất: Tìm kiếm leo đồi, tìm kiếm gradient, tìm kiếm mô phỏng luyện kim.
- Tìm kiếm bắt chước sự tiến hóa: thuật toán di truyền.

3.1. TÌM ĐƯỜNG ĐI NGẮN NHẤT.

Trong các chương trước chúng ta đã nghiên cứu vấn đề tìm kiếm đường đi từ trạng thái ban đầu tới trạng thái kết thúc trong không gian trạng thái. Trong mục này, ta giả sử rằng, giá phải trả để đưa trạng thái a tới trạng thái b (bởi một toán tử nào đó) là một số $k(a,b) \geq 0$, ta sẽ gọi số này là độ dài cung (a,b) hoặc giá trị của cung (a,b) trong đồ thị không gian trạng thái. Độ dài của các cung được xác định tùy thuộc vào vấn đề. Chẳng hạn, trong bài toán tìm đường đi trong bản đồ giao thông, giá của cung (a,b) chính là độ dài của đường nối thành phố a với thành phố b. Độ dài đường đí được xác định là tổng độ dài của các cung trên đường đi. Vấn đề của chúng ta trong mục này, tìm đường đi ngắn nhất từ trạng thái ban đầu tới trạng thái đích. Không gian tìm kiếm ở đây bao gồm tất cả các đường đi từ trạng thái ban đầu tới trạng thái kết thúc, hàm mục tiêu được xác định ở đây là độ dài của đường đi.

Chúng ta có thể giải quyết vấn đề đặt ra bằng cách tìm tất cả các đường đi có thể có từ trạng thái ban đầu tới trạng thái đích (chẳng hạn, sử sụng các ký thuật tìm kiếm mù), sau đó so sánh độ dài của chúng, ta sẽ tìm

ra đường đi ngắn nhất. Thủ tục tìm kiếm này thường được gọi là thủ tục bảo tàng Anh Quốc (British Museum Procedure). Trong thực tế, kỹ thuật này không thể áp dụng được, vì cây tìm kiếm thường rất lớn, việc tìm ra tất cả các đường đi có thể có đòi hỏi rất nhiều thời gian. Do đó chỉ có một cách tăng hiệu quả tìm kiếm là sử dụng các hàm đánh giá đề hướng dẫn sử tìm kiếm. Các phương pháp tìm kiếm đường đi ngắn nhất mà chúng ta sẽ trình bày đều là các phương pháp tìm kiếm heuristic.

Giả sử u là một ***trạng thái đạt tới*** (có dường đi từ trạng thái ban đầu u_0 tới u). Ta xác định hai hàm đánh giá sau:

- g(u) là đánh giá độ dài đường đi ngắn nhất từ u_0 tới u (Đường đi từ u_0 tới trạng thái u không phải là trạng thái đích được gọi là ***đường đi một phần***, để phân biệt với ***đường đi đầy đủ***, là đường đi từ u_0 tới trạng thái đích).
- h(u) là đánh giá độ dài đường đi ngắn nhất từ u tới trạng thái đích.

Hàm h(u) được gọi là ***chấp nhận được*** (hoặc đánh giá thấp***)*** nếu với mọi trạng thái u, h(u) ≤ độ dài đường đi ngắn nhất thực tế từ u tới trạng thái đích. Chẳng hạn trong bài toán tìm đường đi ngắn nhất trên bản đồ giao thông, ta có thể xác định h(u) là độ dài đường chim bay từ u tới đích.

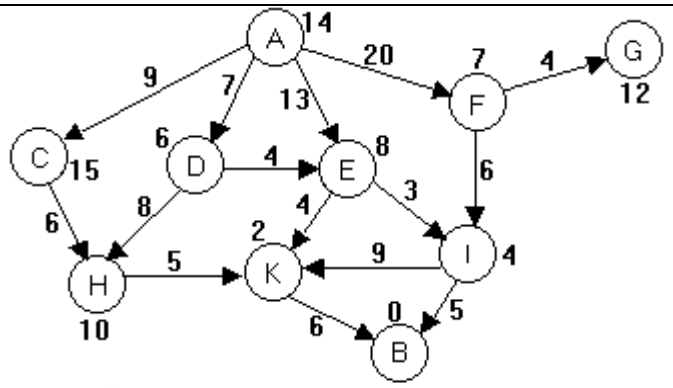
Ta có thể sử dụng kỹ thuật tìm kiếm leo đồi với hàm đánh giá h(u). Tất nhiên phương pháp này chỉ cho phép ta tìm được đường đi tương đối tốt, chưa chắc đã là đường đi tối ưu.

Ta cũng có thể sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(u). Phương pháp này sẽ tìm ra đường đi ngắn nhất, tuy nhiên nó có thể kém hiệu quả.

Để tăng hiệu quả tìm kiếm, ta sử dụng hàm đánh giá mới:

f(u) = g(u) + h(u)

Tức là, f(u) là đánh giá độ dài đường đi ngắn nhất qua u từ trạng thái ban đầu tới trạng thái kết thúc.



Hình 3.1 Đồ thị không gian trạng thái với hàm đánh giá.

3.1.1. Thuật toán A*

Thuật toán A* là thuật toán sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh giá f(u).

Để thấy được thuật toán A* làm việc như thế nào, ta xét đồ thị không gian trạng thái trong hình 3.1. Trong đó, trạng thái ban đầu là trạng thái A, trạng thái đích là B, các số ghi cạnh các cung là độ dài cung đó, các số cạnh các đỉnh là giá trị của hàm h.Đầu tiên, phát triển đỉnh A sinh ra các đỉnh con C, D, E và F. Tính giá trị của hàm f tại các đỉnh này ta có:

g(C) = 9, f(C) = 9 + 15 = 24, g(D) = 7, f(D) = 7 + 6 = 13,

g(E) = 13, f(E) = 13 + 8 = 21, g(F) = 20, f(F) = 20 +7 = 27

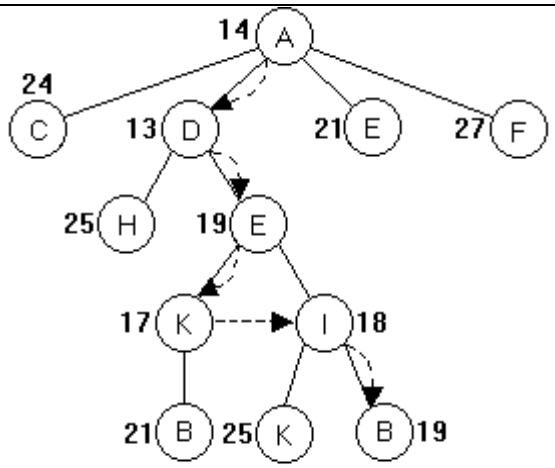
Như vậy đỉnh tốt nhất là D (vì f(D) = 13 là nhỏ nhất). Phát triển D, ta nhận được các đỉnh con H và E. Ta đánh giá H và E (mới):

g(H) = g(D) + Độ dài cung (D, H) = 7 + 8 = 15, f(H) = 15 + 10 = 25.

Đường đi tới E qua D có độ dài:

g(E) = g(D) + Độ dài cung (D, E) = 7 + 4 = 11.

Vậy đỉnh E mới có đánh giá là f(E) = g(E) + h(E) = 11 + 8 = 19. Trong số các đỉnh chờ phát triển, thì đỉnh E với đánh giá f(E) = 19 là đỉnh tốt nhất. Phát triển đỉnh này, ta nhận được các đỉnh con của nó là K và I. Chúng ta tiếp tục quá trình trên cho tới khi đỉnh được chọn để phát triển là đỉnh kết thúc B, độ dài đường đi ngắn nhất tới B là g(B) = 19. Quá trình tìm kiếm trên được mô tả bởi cây tìm kiếm trong hình 3.2, trong đó các số cạnh các đỉnh là các giá trị của hàm đánh giá f(u).



Hình 3.2 Cây tìm kiếm theo thuật toán A*

procedure A*;
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
2. loop do
2.1 if *L rỗng* **then**
{thông báo thất bại; **stop***};*
2.2 Loại trạng thái u ở đầu danh sách L;
2.3 if *u là trạng thái đích* **then**
{thông báo thành công; **stop***}*
2.4 **for** *mỗi trạng thái v kề u* **do**
{g(v) ← g(u) + k(u,v);
f(v) ← g(v) + h(v);
Đặt v vào danh sách L;}
2.5 Sắp xếp L theo thứ tự tăng dần của hàm f sao cho
trạng thái có giá trị của hàm f nhỏ nhất
ở đầu danh sách;
end*;*

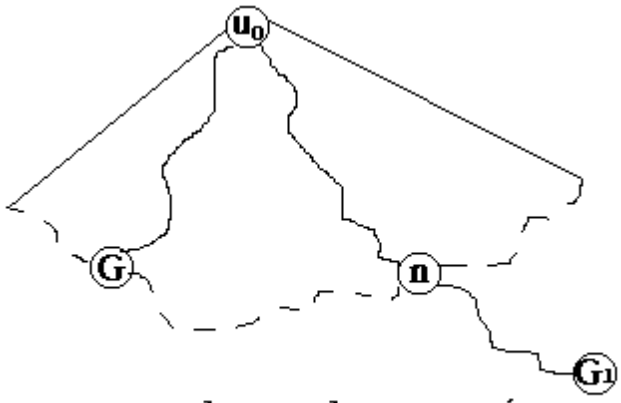
Chúng ta đưa ra một số nhận xét về thuật toán A*.

- Người ta chứng minh được rằng, nếu hàm đánh giá h(u) là đánh giá thấp nhất (trường hợp đặc biệt, h(u) = 0 với mọi trạng thái u) thì thuật toán A* là thuật toán ***tối ưu,*** tức là nghiệm mà nó tìm ra là nghiệm tối ưu. Ngoài ra, nếu độ dài của các cung không nhỏ hơn một số dương δ

nào đó thì thuật toán A* là thuật toán ***đầy đủ*** theo nghĩa rằng, nó luôn dừng và tìm ra nghiệm.

Chúng ta chứng minh tính tối ưu của thuật toán A*.

Giả sử thuật toán dừng lại ở đỉnh kết thúc G với độ dài đường đi từ



Hình 3.3 Đỉnh lá n của cây tìm kiếm nằm trên đường đi tối ưu.

trạng thái ban đầu u_0 tới G là g(G). Vì G là đỉnh kết thúc, ta có h(G) = 0 và f(G) = g(G) + h(G) = g(G). Giả sử nghiệm tối ưu là đường đi từ u_0 tới đỉnh kết thúc G_1 với độ dài l. Giả sử đường đi này "thoát ra"khỏi cây tìm kiếm tại đỉnh lá n (Xem hình 3.3). Có thể xẩy ra hai khả năng: n trùng với G_1 hoặc không. Nếu n là G_1 thì vì G được chọn để phát triển trước G_1, nên $f(G) \leq f(G_1)$, do đó $g(G) \leq g(G_1) = l$. Nếu $n \neq G_1$ thì do h(u) là hàm đánh giá thấp, nên $f(n) = g(n) + h(n) \leq l$. Mặt khác, cũng do G được chọn để phát triển trước n, nên $f(G) \leq f(n)$, do đó, $g(G) \leq l$. Như vậy, ta đã chứng minh được rằng độ dài của đường đi mà thuật toán tìm ra g(G) không dài hơn độ dài l của đường đi tối ưu. Vậy nó là độ dài đường đi tối ưu.

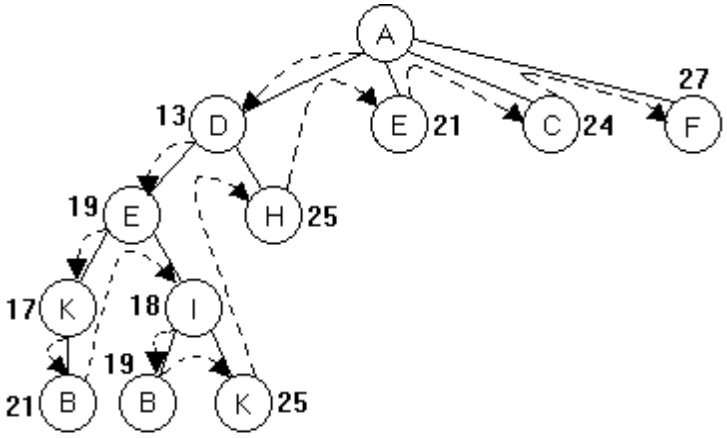
- Trong trường hợp hàm đánh giá h(u) = 0 với mọi u, thuật toán A* chính là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(u) mà ta đã nói đến.
- Thuật toán A* đã được chứng tỏ là thuật toán hiệu quả nhất trong số các thuật toán đầy đủ và tối ưu cho vấn đề tìm kiếm đường đi ngắn nhất.

3.1.2. Thuật toán tìm kiếm nhánh-và-cận.

Thuật toán nhánh_và_cận là thuật toán sử dụng tìm kiếm leo đồi với hàm đánh giá f(u).

Trong thuật toán này, tại mỗi bước khi phát triển trạng thái u, thì ta sẽ chọn trạng thái tốt nhất v (f(v) nhỏ nhất) trong số các trạng thái kề u đề phát

triển ở bước sau. Đi xuống cho tới khi gặp trạng thái v là đích, hoặc gặp trạng thái v không có đỉnh kề, hoặc gặp trạng thái v mà f(v) lớn hơn độ dài đường đi tối ưu tạm thời, tức là đường đi đầy đủ ngắn nhất trong số các đường đi đầy đủ mà ta đã tìm ra. Trong các trường hợp này, ta không phát triển đỉnh v nữa, hay nói cách khác, ta cắt đi các nhánh cây xuất phát từ v, và quay lên cha của v để tiếp tục đi xuống trạng thái tốt nhất trong các trạng thái còn lại chưa được phát triển.



Hình 3.4 Cây tìm kiếm nhánh_và_cận.

Ví dụ: Chúng ta lại xét không gian trạng thái trong hình 3.1. Phát triển đỉnh A, ta nhận được các đỉnh con C, D, E và F, f(C) = 24, f(D) = 13, f(E) = 21, f(F) = 27. Trong số này D là tốt nhất, phát triển D, sinh ra các đỉnh con H và E, f(H) = 25, f(E) = 19. Đi xuống phát triển E, sinh ra các đỉnh con là K và I, f(K) = 17, f(I) = 18. Đi xuống phát triển K sinh ra đỉnh B với f(B) = g(B) = 21. Đi xuống B, vì B là đỉnh đích, vậy ta tìm được đường đi tối ưu tạm thời với độ dài 21. Từ B quay lên K, rồi từ K quay lên cha nó là E. Từ E đi xuống J, f(J) = 18 nhỏ hơn độ dài đường đi tạm thời (là 21). Phát triển I sinh ra các con K và B, f(K) = 25, f(B) = g(B) = 19. Đi xuống đỉnh B, vì đỉnh B là đích ta tìm được đường đi đầy đủ mới với độ dài là 19 nhỏ hơn độ dài đường đi tối ưu tạm thời cũ (21). Vậy độ dài đường đi tối ưu tạm thời bây giờ là 19. Bây giờ từ B ta lại quay lên các đỉnh còn lại chưa được phát triển. Song các đỉnh này đều có giá trị hàm đánh giá lớn hơn 19, do đó không có đỉnh nào được phát triển nữa. Như vậy, ta tìm được đường đi tối ưu với độ dài 19. Cây tìm kiếm được biểu diễn trong hình 3.4.

Thuật toán nhánh_và_cận sẽ được biểu diễn bởi thủ tục Branch_and_Bound. Trong thủ tục này, biến cost được dùng để lưu độ dài đường đi ngắn nhất. Giá trị ban đầu của cost là số đủ lớn, hoặc độ dài của một đường đi đầy đủ mà ta đã biết.

procedure *Branch_and_Bound*;

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

Gán giá trị ban đầu cho cost;

2. **loop** ***do***

2.1 ***if*** *L rỗng* ***then stop;***

2.2 Loại trạng thái u ở đầu danh sách L;

2.3 ***if*** *u là trạng thái kết thúc* ***then***

if *g(u) ≤ cost* ***then {cost*** *← g(u);* ***Quay lại 2.1};***

2.4 ***if*** *f(u) > cost* ***then*** *Quay lại 2.1****;***

2.5 ***for*** *mỗi trạng thái v kề u* ***do***

{g(v) ← g(u) + k(u,v);

f(v) ← g(v) + h(v);

Đặt v vào danh sách L_1};

2.6 Sắp xếp L_1 theo thứ tự tăng của hàm f;

2.7 Chuyển L_1 vào đầu danh sách L sao cho trạng thái ở đầu L_1 trở thành ở đầu L;

end;

Người ta chứng minh được rằng, thuật toán nhánh_và_cận cũng là thuật toán đầy đủ và tối ưu nếu hàm đánh giá h(u) là đánh giá thấp và có độ dài các cung không nhỏ hơn một số dương δ nào đó.

3.2. TÌM ĐỐI TƯỢNG TỐT NHẤT

Trong mục này chúng ta sẽ xét vấn đề tìm kiếm sau. Trên không gian tìm kiếm U được xác định hàm giá (hàm mục tiêu) cost, ứng với mỗi đối tượng x ∈ U với một giá trị số cost(x), số này được gọi là giá trị của x. Chúng ta cần tìm một đối tượng mà tại đó hàm giá đạt giá trị lớn nhất, ta gọi đối tượng đó là ***đối tượng tốt nhất***. Giả sử không gian tìm kiếm có cấu trúc cho phép ta xác định được khái niệm lân cận của mỗi đối tượng. Chẳng hạn, U là không gian trạng thái thì lân cận của trạng thái u gồm tất cả các trạng thái v kề u; nếu U là không gian các vectơ thực n-chiều thì lân cận của vectơ $x = (x_1, x_2, ... x_n)$ gồm tất cả các vectơ ở gần x theo khoảng cách Ơcơlit thông thường.

Trong mục này, ta sẽ xét kỹ thuật tìm kiếm leo đồi để tìm đối tượng tốt nhất. Sau đó ta sẽ xét kỹ thuật tìm kiếm gradient (gradient search). Đó là kỹ thuật leo đồi áp dụng cho không gian tìm kiếm là không gian các vectơ thực n-chiều và hàm giá là hàm khả vi liên tục. Cuối cùng ta sẽ nghiên cứu kỹ thuật tìm kiếm mô phỏng luyện kim(simulated annealing).

3.2.1. Tìm kiếm leo đồi

Kỹ thuật tìm kiếm leo đồi để tìm kiếm đối tượng tốt nhất hoàn toàn giống như kỹ thuật tìm kiếm leo đồi để tìm trạng thái kết thúc đã xét trong mục 2.3. Chỉ khác là trong thuật toán leo đồi ở mục 2.3, từ một trạng thái ta "leo lên"trạng thái kề tốt nhất (được xác định bởi hàm giá), tiếp tục cho tới khi đạt tới trạng thái đích; nếu chưa đạt tới trạng thái đích mà không leo lên được nữa, thì ta tiếp tục "tụt xuống"trạng thái trước nó, rồi lại leo lên trạng thái tốt nhất còn lại. Còn ở đây, từ một đỉnh u ta chỉ leo lên đỉnh tốt nhất v (được xác định bởi hàm giá cost) trong lân cận u nếu đỉnh này "cao hơn"đỉnh u, tức là cost(v) > cost(u). Quá trình tìm kiếm sẽ dừng lại ngay khi ta không leo lên đỉnh cao hơn được nữa.

Trong thủ tục leo đồi dưới đây, biến u lưu đỉnh hiện thời, biến v lưu đỉnh tốt nhất (cost(v) nhỏ nhất) trong các đỉnh ở lân cận u. Khi thuật toán dừng, biến u sẽ lưu đối tượng tìm được.

procedure *Hill_Climbing;*

begin

1. u ← một đối tượng ban đầu nào đó;

*2.***loop do**

2.1 v← đối tượng tốt nhất (được xác định bởi hàm giá)trong lân cận u;

2.2 **if** *cost(u) < cost(v)* **then** *u ←v else* **stop***;*

end*;*

TỐI ƯU ĐỊA PHƯƠNG VÀ TỐI ƯU TOÀN CỤC

Rõ ràng là, khi thuật toán leo đồi dừng lại tại đối tương u*, thì giá của nó cost(u*) lớn hơn giá của tất cả các đối tượng nằm trong lân cận của tất cả các đối tượng trên đường đi từ đối tượng ban đầu tới trạng thái u*. Do đó nghiệm u* mà thuật toán leo đồi tìm được là ***tối ưu địa phương***. Cần nhấn mạnh rằng không có gì đảm bảo nghiệm đó là ***tối ưu toàn cục*** theo nghĩa là cost(u*) là lớn nhất trên toàn bộ không gian tìm kiếm.

Để nhận được nghiệm tốt hơn bằng thuật toán leo đồi, ta có thể áp dụng lặp lại nhiều lần thủ tục leo đồi xuất phát từ một dãy các đối tượng ban đầu được chọn ngẫu nhiên và lưu lại nghiệm tốt nhất qua mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm được nghiệm tối ưu.

Kết quả của thuật toán leo đồi phụ thuộc rất nhiều vào hình dáng của "mặt cong"của hàm giá. Nếu mặt cong chỉ có một số ít cực đại địa phương, thì kỹ thuật leo đồi sẽ tìm ra rất nhanh cực đại toàn cục. Song có những vấn

đề mà mặt cong của hàm giá tựa như lông nhím vậy, khi đó sử dụng kỹ thuật leo đồi đòi hỏi rất nhiều thời gian.

3.2.2. Tìm kiếm gradient

Tìm kiếm gradient là kỹ thuật tìm kiếm leo đồi để tìm giá trị lớn nhất (hoặc nhỏ nhất) của hàm khả vi liên tục f(x) trong không gian các vectơ thực n-chiều. Như ta đã biết, trong lân cận đủ nhỏ của điểm $x = (x_1,...,x_n)$, thì hàm

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

f tăng nhanh nhất theo hướng của vectơ gradient:

Do đó tư tưởng của tìm kiếm gradient là từ một điểm ta đi tới điểm ở lân cận nó theo hướng của vectơ gradient.

```
procedure Gradient_Search;
begin
x ← điểm xuất phát nào đó;
repeat
x ← x + α∇f(x);
until |∇f| < ε;
end;
```

Trong thủ tục trên, α là hằng số dương nhỏ xác định tỉ lệ của các bước, còn ε là hằng số dương nhỏ xác định tiêu chuẩn dừng. Bằng cách lấy các bước đủ nhỏ theo hướng của vectơ gradient chúng ta sẽ tìm được điểm cực đại địa phương, đó là điểm mà tại đó $\nabla f = 0$, hoặc tìm được điểm rất gần với cực đại địa phương.

3.2.3. Tìm kiếm mô phỏng luyện kim:

Như đã nhấn mạnh ở trên, tìm kiếm leo đồi không đảm bảo cho ta tìm được nghiệm tối ưu toàn cục. Để cho nghiệm tìm được gần với tối ưu toàn cục, ta áp dụng kỹ thuật leo đồi lặp xuất phát từ các điểm được lựa chọn ngẫu nhiên. Bây giờ thay cho việc luôn luôn "leo lên đồi"xuất phát từ các điểm khác nhau, ta thực hiện một số bước "tụt xuống"nhằm thoát ra khỏi các điểm cực đại địa phương. Đó chính là tư tưởng của kỹ thuật tìm kiếm mô phỏng luyện kim.

Trong tìm kiếm leo đồi, khi ở một trạng thái u ta luôn luôn đi tới trạng thái tốt nhất trong lân cận nó. Còn bây giờ, trong tìm kiếm mô phỏng luyện kim, ta chọn ngẫu nhiên một trạng thái v trong lân cận u. Nếu trạng thái v

được chọn tốt hơn u (cost(v) > cost(u)) thì ta đi tới v, còn nếu không ta chỉ đi tới v với một xác suất nào đó. Xác suất này giảm theo hàm mũ của "độ xấu"của trạng thái v. Xác suất này còn phụ thuộc vào tham số nhiệt độ T. Nhiệt độ T càng cao thì bước đi tới trạng thái xấu càng có khả năng được thực hiện. Trong quá trình tìm kiếm, tham số nhiệt độ T giảm dần tới không. Khi T gần không, thuật toán hoạt động gần giống như leo đồi, hầu như nó không thực hiện bước tụt xuống. Cụ thể ta xác định xác suất đi tới trạng thái xấu v từ u là $e^{\Delta/T}$, ở đây Δ = cost(v) - cost(u).

Sau đây là thủ tục mô phỏng luyện kim.

procedure *Simulated_Anneaning;*
begin
  t ← 0;
  u ← trạng thái ban đầu nào đó;
  T ← nhiệt độ ban đầu;
repeat
  v ← trạng thái được chọn nhẫu nhiên trong lân cận u;
if *cost(v) > cost(u)* **then** *u ← v*
    else *u ← v với xác suất* $e^{\Delta/T}$*;*
    T ← g(T, t);
    t ← t + 1;
until *T đủ nhỏ*
end*;*

Trong thủ tục trên, hàm g(T, t) thỏa mãn điều kiện g(T, t) < T với mọi t, nó xác định tốc độ giảm của nhiệt độ T. Người ta chứng minh được rằng, nếu nhiêt độ T giảm đủ chậm, thì thuật toán sẽ tìm được nghiệm tối ưu toàn cục. Thuật toán mô phỏng luyện kim đã được áp dụng thành công cho các bài toán tối ưu cỡ lớn.

3.3. TÌM KIẾM MÔ PHỎNG SỰ TIẾN HÓA. THUẬT TOÁN DI TRUYỀN

Thuật toán di truyền (TTDT) là thuật toán bắt chước sự chọn lọc tự nhiên và di truyền. Trong tự nhiên, các cá thể khỏe, có khả năng thích nghi tốt với môi trường sẽ được tái sinh và nhân bản ở các thế hệ sau. Mỗi cá thể có cấu trúc gien đặc trưng cho phẩm chất của cá thể đó. Trong quá trình sinh sản, các cá thể con có thể thừa hưởng các phẩm chất của cả cha và mẹ, cấu trúc gien của nó mang một phần cấu trúc gien của cha và mẹ. Ngoài ra, trong quá trình tiến hóa, có thể xảy ra hiện tượng đột biến, cấu trúc gien của cá thể con có thể chứa các gien mà cả cha và mẹ đều không có.

Trong TTDT, mỗi cá thể được mã hóa bởi một cấu trúc dữ liệu mô tả cấu trúc gien của cá thể đó, ta sẽ gọi nó là ***nhiễm sắc thể (chroniosome).*** Mỗi nhiễm sắc thể được tạo thành từ các đơn vị được gọi là gien. Chẳng hạn, trong các TTDT cổ điển, các nhiễm sắc thể là các chuỗi nhị phân, tức là mỗi cá thể được biểu diễn bởi một chuỗi nhị phân.

TTDT sẽ làm việc trên các quần thể gồm nhiều cá thể. Một quần thể ứng với một giai đoạn phát triển sẽ được gọi là một ***thế hệ.*** Từ thế hệ ban đầu được tạo ra, TTDT bắt chước chọn lọc tự nhiên và di truyền để biến đổi các thế hệ. TTDT sử dụng các toán tử cơ bản sau đây để biến đổi các thế hệ.

- ***Toán tử tái sinh (reproduction)*** (còn được gọi là ***toán tử chọn lọc (selection)***). Các cá thể tốt được chọn lọc để đưa vào thế hệ sau. Sự lựa chọn này được thực hiện dựa vào độ thích nghi với môi trường của mỗi cá thể. Ta sẽ gọi hàm ứng mỗi cá thể với độ thích nghi của nó là ***hàm thích nghi (fitness function).***
- ***Toán tử lai ghép (crossover).*** Hai cá thể cha và mẹ trao đổi các gien để tạo ra hai cá thể con.
- ***Toán tử đột biến (mutation).*** Một cá thể thay đổi một số gien để tạo thành cá thể mới.

Tất cả các toán tử trên khi thực hiện đều mang tính ngẫu nhiên. Cấu trúc cơ bản của TTDT là như sau:

procedure *Genetic_Algorithm;*
begin
    t ← 0;
    Khởi tạo thế hệ ban đầu P(t);
    Đánh giá P(t) (theo hàm thích nghi);
    repeat

t ← t + 1;
Sinh ra thế hệ mới P(t) từ P(t-1) bởi
Chọn lọc
Lai ghép
Đột biến;
Đánh giá P(t);
until *điều kiện kết thúc được thỏa mãn;*
end;

Trong thủ tục trên, điều kiện kết thúc vòng lặp có thể là một số thế hệ đủ lớn nào đó, hoặc độ thích nghi của các cá thể tốt nhất trong các thế hệ kế tiếp nhau khác nhau không đáng kể. Khi thuật toán dừng, cá thể tốt nhất trong thế hệ cuối cùng được chọn làm nghiệm cần tìm.

Bây giờ ta sẽ xét chi tiết hơn toán tử chọn lọc và các toán tử di truyền (lai ghép, đột biến) trong các TTDT cổ điển.

1. Chọn lọc: Việc chọn lọc các cá thể từ một quần thể dựa trên độ thích nghi của mỗi cá thể. Các cá thể có độ thích nghi cao có nhiều khả năng được chọn. Cần nhấn mạnh rằng, hàm thích nghi chỉ cần là ***một hàm thực dương***, nó có thể không tuyến tính, không liên tục, không khả vi. Quá trình chọn lọc được thực hiện theo kỹ thuật quay bánh xe.

Giả sử thế hệ hiện thời P(t) gồm có n cá thể $\{x_1,..,x_n\}$. Số n được gọi là cỡ của quần thể. Với mỗi cá thể x_i, ta tính độ thích nghi của nó $f(x_i)$. Tính

$$F = \sum_{i=1}^{n} f(x_i)$$

tổng các độ thích nghi của tất cả các cá thể trong quần thể:

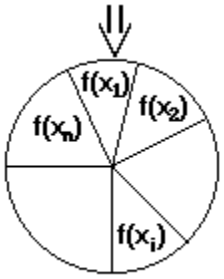
Mỗi lần chọn lọc, ta thực hiện hai bước sau:

1) Sinh ra một số thực ngẫu nhiên q trong khoảng (0, F);

2) x_k là cá thể được chọn, nếu k là số nhỏ nhất sao cho

$$\sum_{i=1}^{k} f(x_i) \geq 4$$

Việc chọn lọc theo hai bước trên có thể minh họa như sau: Ta có một bánh xe được chia thành n phần, mỗi phần ứng với độ thích nghi của một cá thể (hình 3.5). Một mũi tên chỉ vào bánh xe. Quay bánh xe, khi bánh xe dừng, mũi tên chỉ vào phần nào, cá thể ứng với phần đó được chọn.



Hình 3.5 Kỹ thuật quay bánh xe.

Rõ ràng là với cách chọn này, các cá thể có độ thích nghi càng cao càng có khả năng được chọn. Các cá thể có độ thích nghi cao có thể có một hay nhiều bản sao, các cá thể có độ thích nghi thấp có thể không có mặt ở thế hệ sau (nó bị chết đi).

2. Lai ghép: Trên các cá thể được chọn lọc, ta tiến hành toán tử lai ghép. Đầu tiên ta cần đưa ra xác suất lai ghép p_c. xác suất này cho ta hy vọng có p_c.n cá thể được lai ghép (n là cỡ của quần thể).

Với mỗi cá thể ta thực hiện hai bước sau:

1) Sinh ra số thực ngẫu nhiên r trong đoạn [0, 1];
2) Nếu $r < p_c$ thì cá thể đó được chọn để lai ghép

Từ các cá thể được chọn để lai ghép, người ta cặp đôi chúng một cách ngẫu nhiên. Trong trường hợp các nhiễm sắc thể là các chuỗi nhị phân có độ dài cố định m, ta có thể thực hiện lai ghép như sau: Với mỗi cặp, sinh ra một số nguyên ngẫu nhiên p trên đoạn [0, m -1], p là vị trí điểm ghép. Cặp gồm hai nhiễm sắc thể

$$\mathbf{a} = (a_1,..., a_p, a_{p+1},..., a_m)$$

$$\mathbf{a} = (b_1,..., b_p, b_{p+1},..., b_m)$$

được thay bởi hai con là:

$$\mathbf{a'} = (a_1,..., a_p, b_{p+1},..., b_m)$$

$$\mathbf{b'} = (b_1,..., b_p, a_{p+1},..., a_m)$$

3. Đột biến: Ta thực hiện toán tử đột biến trên các cá thể có được sau quá trình lai ghép. Đột biến là thay đổi trạng thái một số gien nào đó trong nhiễm sắc thể. Mỗi gien chịu đột biến với xác suất p_m. Xác suất đột biến p_m do ta xác định và là xác suất thấp. Sau đây là toán tử đột biến trên các nhiễm sắc thể chuỗi nhị phân.

Với mỗi vị trí i trong nhiễm sắc thể:

$$\mathbf{a = (a_1,..., a_i,..., a_m)}$$

Ta sinh ra một số thực ngẫu nhiên p_i trong [0,1]. Qua đột biến a được biến thành a' như sau:

$$\mathbf{a' = (a'_1,..., a'_i,..., a'_m)}$$

Trong đó:

$$\mathbf{a'_i} = \begin{cases} \mathbf{a_i} & \text{nếu } p_i \geq p_m \\ \mathbf{1 - a_i} & \text{nếu } p_i < p_m \end{cases}$$

Sau quá trình chọn lọc, lai ghép, đột biến, một thế hệ mới được sinh ra. Công việc còn lại của thuật toán di truyền bây giờ chỉ là lặp lại các bước trên.

Ví dụ: Xét bài toán tìm max của hàm $f(x) = x^2$ với x là số nguyên trên đoạn [0,31]. Để sử dụng TTDT, ta mã hoá mỗi số nguyên x trong đoạn [0,31] bởi một số nhị phân độ dài 5, chẳng hạn, chuỗi 11000 là mã của số nguyên 24. Hàm thích nghi được xác định là chính hàm $f(x) = x^2$. Quần thể ban đầu gồm 4 cá thể (cỡ của quần thể là n = 4). Thực hiện quá trình chọn lọc, ta nhận được kết quả trong bảng sau. Trong bảng này, ta thấy cá thể 2 có độ thích nghi cao nhất (576) nên nó được chọn 2 lần, cá thể 3 có độ thích nghi thấp nhất (64) không được chọn lần nào. Mỗi cá thể 1 và 4 được chọn 1 lần.

Bảng kết quả chọn lọc

Số liệu cá thể	Quần thể ban đầu	x	Độ thích nghi $f(x) = x^2$	Số lần được chọn
1	0 1 1 0 1	13	169	1
2	1 1 0 0 0	24	576	2
3	0 1 0 0 0	8	64	0
4	1 0 0 1 1	19	361	1

Thực hiện qúa trình lai ghép với xác suất lai ghép $p_c = 1$, cả 4 cá thể sau chọn lọc đều được lai ghép. Kết quả lai ghép được cho trong bảng sau.

Trong bảng này, chuỗi thứ nhất được lai ghép với chuỗi thứ hai với điểm ghép là 4, hai chuỗi còn lại được lai ghép với nhau với điểm ghép là 2.

Bảng kết quả lai ghép

Quần thể sau chọn lọc	Điểm ghép	Quần thể sau lai ghép	x	Độ thích nghi $f(x) = x^2$
0 1 1 0 | 1	4	0 1 1 0 0	2	144
1 1 0 0 | 0	4	1 1 0 0 1	5	625
1 1 | 0 0 0	2	1 1 0 1 1	7	729
1 0 | 0 1 1	2	1 0 0 0 0	6	256

Để thực hiện quá trình đột biến, ta chọn xác suất đột biến p_m= 0,001, tức là ta hy vọng có 5.4.0,001 = 0,02 bit được đột biến. Thực tế sẽ không có bit nào được đột biến. Như vậy thế hệ mới là quần thể sau lai ghép. Trong thế hệ ban đầu, độ thích nghi cao nhất là 576, độ thích nghi trung bình 292. Trong thế hệ sau, độ thích nghi cao nhất là 729, trung bình là 438. Chỉ qua một thế hệ, các cá thể đã "tốt lên"rất nhiều.

Thuật toán di truyền khác với các thuật toán tối ưu khác ở các điểm sau:

- TTDT chỉ sử dụng hàm thích nghi để hướng dẫn sự tìm kiếm, hàm thích nghi chỉ cần là hàm thực dương. Ngoài ra, nó không đòi hỏi không gian tìm kiếm phải có cấu trúc nào cả.
- TTDT làm việc trên các nhiễm sắc thể là mã của các cá thể cần tìm.
- TTDT tìm kiếm từ một quần thể gồm nhiều cá thể.
- Các toán tử trong TTDT đều mang tính ngẫu nhiên.

Để giải quyết một vấn đề bằng TTDT, chúng ta cần thực hiện các bước sau đây:

- Trước hết ta cần mã hóa các đối tượng cần tìm bởi một cấu trúc dữ liệu nào đó. Chẳng hạn, trong các TTDT cổ điển, như trong ví dụ trên, ta sử dụng mã nhị phân.
- Thiết kế hàm thích nghi. Trong các bài toán tối ưu, hàm thích nghi được xác định dựa vào hàm mục tiêu.
- Trên cơ sở cấu trúc của nhiễm sắc thể, thiết kế các toán tử di truyền (lai ghép, đột biến) cho phù hợp với các vấn đề cần giải quyết.

- Xác định cỡ của quần thể và khởi tạo quần thể ban đầu.
- Xác định xác suất lai ghép p_c và xác suất đột biến p_m. Xác suất đột biến cần là xác suất thấp. Người ta (Goldberg, 1989) khuyên rằng nên chọn xác suất lai ghép là 0,6 và xác suất đột biến là 0,03. Tuy nhiên cần qua thử nghiệm để tìm ra các xác suất thích hợp cho vấn đề cần giải quyết.

Nói chung thuật ngữ TTDT là để chỉ TTDT cổ điển, khi mà cấu trúc của các nhiễm sắc thể là các chuỗi nhị phân với các toán tử di truyền đã được mô tả ở trên. Song trong nhiều vấn đề thực tế, thuận tiện hơn, ta có thể biểu diễn nhiễm sắc thể bởi các cấu trúc khác, chẳng hạn vectơ thực, mảng hai chiều, cây,... Tương ứng với cấu trúc của nhiễm sắc thể, có thể có nhiều cách xác định các toán tử di truyền. Quá trình sinh ra thế hệ mới P(t) từ thế hệ cũ P(t - 1) cũng có nhiều cách chọn lựa. Người ta gọi chung các thuật toán này là thuật toán tiến hóa (evolutionary algorithms) hoặc chương trình tiến hóa (evolution program).

Thuật toán tiến hóa đã được áp dụng trong các vấn đề tối ưu và học máy. Để hiểu biết sâu sắc hơn về thuật toán tiến hoá, bạn đọc có thể tìm đọc [3], [9] và [16]. Nếu như [9] và 6] được xem là các sách hay nhất viết về ***TTDT***, [3] lại cho ta cái nhìn tổng quát về sự phát triển gần đây của ***TTDT***.

CHƯƠNG 4

TÌM KIẾM CÓ ĐỐI THỦ

Nghiên cứu máy tính chơi cờ đã xuất hiện rất sớm. Không lâu sau khi máy tính lập trình được ra đời, vào năm 1950, Claude Shannon đã viết chương trình chơi cờ đầu tiên. Các nhà nghiên cứu **Trí Tuệ Nhân Tạo** đã nghiên cứu việc chơi cờ, vì rằng máy tính chơi cờ là một bằng chứng rõ ràng về khả năng máy tính có thể làm được các công việc đòi hỏi trí thông minh của con người. Trong chương này chúng ta sẽ xét các vấn đề sau đây:

- Chơi cờ có thể xem như vấn đề tìm kiếm trong không gian trạng thái.
- Chiến lược tìm kiếm nước đi Minimax.
- Phương pháp cắt cụt α-β, một kỹ thuật để tăng hiệu quả của tìm kiếm Minimax.

4.1. CÂY TRÒ CHƠI VÀ TÌM KIẾM TRÊN CÂY TRÒ CHƠI.

Trong chương này chúng ta chỉ quan tâm nghiên cứu các trò chơi có hai người tham gia, chẳng hạn các loại cờ (cờ vua, cờ tướng, cờ ca rô...). Một người chơi được gọi là Trắng, đối thủ của anh ta được gọi là Đen. Mục tiêu của chúng ta là nghiên cứu chiến lược chọn nước đi cho Trắng (Máy tính cầm quân Trắng).

Chúng ta sẽ xét các trò chơi hai người với các đặc điểm sau. Hai người chơi thay phiên nhau đưa ra các nước đi tuân theo các luật đi nào đó, các luật này là như nhau cho cả hai người. Điển hình là cờ vua, trong cờ vua hai người chơi có thể áp dụng các luật đi con tốt, con xe,... để đưa ra nước đi. Luật đi con tốt Trắng xe Trắng,... cũng như luật đi con tốt Đen, xe Đen,... Một đặc điểm nữa là hai người chơi đều được biết thông tin đầy đủ về các tình thế trong trò chơi (không như trong chơi bài, người chơi không thể biết các người chơi khác còn những con bài gì). Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm nước đi, tại mỗi lần đến lượt mình, người chơi phải tìm trong số rất nhiều nước đi hợp lệ (tuân theo đúng luật đi), một nước đi tốt nhất sao cho qua một dãy nước đi đã thực hiện, anh ta giành phần thắng. Tuy nhiên vấn đề tìm kiếm ở đây sẽ phức tạp hơn vấn đề tìm kiếm mà chúng ta

đã xét trong các chương trước, bởi vì ở đây có đối thủ, người chơi không biết được đối thủ của mình sẽ đi nước nào trong tương lai. Sau đây chúng ta sẽ phát biểu chính xác hơn vấn đề tìm kiếm này.

Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm trong không gian trạng thái. Mỗi trạng thái là một tình thế (sự bố trí các quân của hai bên trên bàn cờ).

- Trạng thái ban đầu là sự sắp xếp các quân cờ của hai bên lúc bắt đầu cuộc chơi.
- Các toán tử là các nước đi hợp lệ.
- Các trạng thái kết thúc là các tình thế mà cuộc chơi dừng, thường được xác định bởi một số điều kiện dừng nào đó.
- Một hàm kết cuộc (payoff function) ứng mỗi trạng thái kết thúc với một giá trị nào đó. Chẳng hạn như cờ vua, mỗi trạng thái kết thúc chỉ có thể là thắng, hoặc thua (đối với Trắng) hoặc hòa. Do đó, ta có thễ xác định hàm kết cuộc là hàm nhận giá trị 1 tại các trạng thái kết thúc là thắng (đối với Trắng), -1 tại các trạng thái kết thúc là thua (đối với Trắng) và 0 tại các trạng thái kết thúc hòa. Trong một số trò chơi khác, chẳng hạn trò chơi tính điểm, hàm kết cuộc có thể nhận giá trị nguyên trong khoảng [-k, k] với k là một số nguyên dương nào đó.

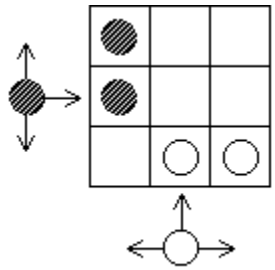
Như vậy vấn đề của Trắng là, tìm một dãy nước đi sao cho xen kẽ với các nước đi của Đen tạo thành một đường đi từ trạng thái ban đầu tới trạng thái kết thúc là thắng cho Trắng.

Để thuận lợi cho việc nghiên cứu các chiến lược chọn nước đi, ta biểu diễn không gian trạng thái trên dưới dạng cây trò chơi.

CÂY TRÒ CHƠI

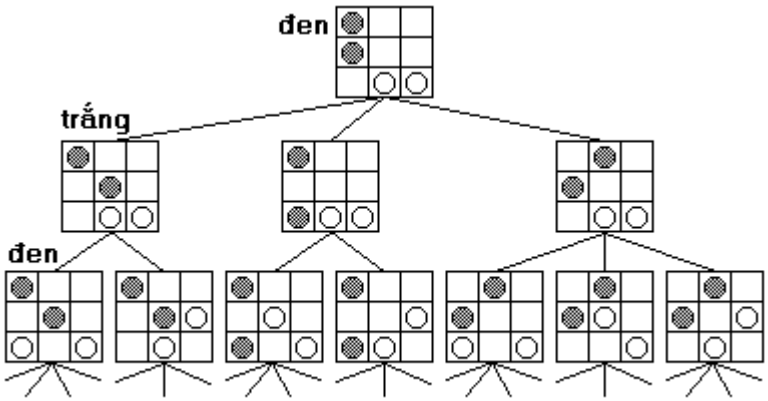
Cây trò chơi được xây dựng như sau. Gốc của cây ứng với trạng thái ban đầu. Ta sẽ gọi đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen). Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó. Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hoặc đều là Đen, các lá của cây ứng với các trnạg thái kết thúc.

Ví dụ: Xét trò chơi Dodgen (được tạo ra bởi Colin Vout). Có hai quân Trắng và hai quân Đen, ban đầu được xếp vào bàn cờ 3*3 (Hình vẽ). Quân Đen có thể đi tới ô trống ở bên phải, ở trên hoặc ở dưới. Quân Trắng có thể đi tới trống ở bên trái, bên phải, ở trên. Quân Đen nếu ở cột ngoài



Hình 4.1 Trò chơi Dodgem.

cùng bên phải có thể đi ra khỏi bàn cờ, quân Trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ. Ai đưa hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình thế mà đối phương không đi được cũng sẽ thắng.



Hình 4.2 Cây trò chơi Dodgem với Đen đi trước.

Giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như trong hình 4.2.

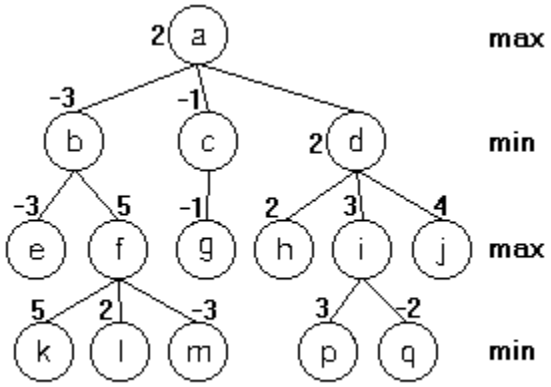
4.2. CHIẾN LƯỢC MINIMAX

Quá trình chơi cờ là quá trình Trắng và Đen thay phiên nhau đưa ra quyết định, thực hiện một trong số các nước đi hợp lệ. Trên cây trò chơi, quá trình đó sẽ tạo ra đường đi từ gốc tới lá. Giả sử tới một thời điểm nào đó, đường đi đã dẫn tới đỉnh u. Nếu u là đỉnh Trắng (Đen) thì Trắng (Đen) cần chọn đi tới một trong các đỉnh Đen (Trắng) v là con của u. Tại đỉnh Đen (Trắng) v mà Trắng (Đen) vừa chọn, Đen (Trắng) sẽ phải chọn đi tới một trong các đỉnh Trắng (Đen) w là con của v. Quá trình trên sẽ dừng lại khi đạt tới một đỉnh là lá của cây.

Giả sử Trắng cần tìm nước đi tại đỉnh u. Nước đi tối ưu cho Trắng là nước đi dẫn tới đỉnh con của v là đỉnh tốt nhất (cho Trắng) trong số các đỉnh

con của u. Ta cần giả thiết rằng, đến lượt đối thủ chọn nước đi từ v, Đen cũng sẽ chọn nước đi tốt nhất cho anh ta. Như vậy, để chọn nước đi tối ưu cho Trắng tại đỉnh u, ta cần phải xác định giá trị các đỉnh của cây trò chơi gốc u. Giá trị của các đỉnh lá (ứng với các trạng thái kết thúc) là giá trị của hàm kết cuộc. Đỉnh có giá trị càng lớn càng tốt cho Trắng, đỉnh có giá trị càng nhỏ càng tốt cho Đen. Để xác định giá trị các đỉnh của cây trò chơi gốc u, ta đi từ mức thấp nhất lên gốc u. Giả sử v là đỉnh trong của cây và giá trị các đỉnh con của nó đã được xác định. Khi đó nếu v là đỉnh Trắng thì giá trị của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con. Còn nếu v là đỉnh Đen thì giá trị của nó là giá trị nhỏ nhất trong các giá trị của các đỉnh con.

Ví dụ: Xét cây trò chơi trong hình 4.3, gốc a là đỉnh Trắng. Giá trị của các đỉnh là số ghi cạnh mỗi đỉnh. Đỉnh i là Trắng, nên giá trị của nó là max(3,-2) = 3, đỉnh d là đỉnh Đen, nên giá trị của nó là min(2, 3, 4) = 2.



Hình 4.3 Gán giá trị cho các đỉnh của cây trò chơi.

Việc gán giá trị cho các đỉnh được thực hiện bởi các hàm đệ qui MaxVal và MinVal. Hàm MaxVal xác định giá trị cho các đỉnh Trắng, hàm MinVal xác định giá trị cho các đỉnh Đen.

```
function MaxVal(u);
begin
        if u là đỉnh kết thúc then MaxVal(u) ← f(u)
        else MaxVal(u) ← max{MinVal(v) | v là đỉnh con của u}
end;

 function MinVal(u);
  begin
      if u là đỉnh kết thúc then MinVal(u) ← f(u)
```

else *MinVal(u) ← min{MaxVal(v) | v là đỉnh con của u}*

end;

Trong các hàm đệ quy trên, f(u) là giá trị của hàm kết cuộc tại đỉnh kết thúc u. Sau đây là thủ tục chọn nước đi cho trắng tại đỉnh u. Trong thủ tục Minimax(u,v), v là biến lưu lại trạng thái mà Trắng sẽ chọn đi tới từ u.

procedure *Minimax(u, v);*
begin
val ← -∞;
for *mỗi w là đỉnh con của u* **do**
if *val <= MinVal(w)* **then**
{val ← MinVal(w); v ← w}
end;

Thủ tục chọn nước đi như trên được gọi là chiến lược Minimax, bởi vì Trắng đã chọ được nước đi dẫn tới đỉnh con có giá trị là max của các giá trị các đỉnh con, và Đen đáp lại bằng nước đi tới đỉnh có giá trị là min của các giá trị các đỉnh con.

Thuật toán Minimax là thuật toán tìm kiếm theo độ sâu, ở đây ta đã cài đặt thuật toán Minimax bởi các hàm đệ quy. Bạn đọc hãy viết thủ tục không đệ quy thực hiện thuật toán này.

Về mặt lí thuyết, chiến lược Minimax cho phép ta tìm được nước đi tối ưu cho Trắng. Song nó không thực tế, chúng ta sẽ không có đủ thời gian để tính được nước đi tối ưu. Bởi vì thuật toán Minimax đòi hỏi ta phải xem xét toàn bộ các đỉnh của cây trò chơi. Trong các trò chơi hay, cây trò chơi là cực kỳ lớn. Chẳng hạn, đối với cờ vua, chỉ tính đến độ sâu 40, thì cây trò chơi đã có khoảng 10^{120} đỉnh! Nếu cây có độ cao m, và tại mỗi đỉnh có b nước đi thì độ phức tạp về thời gian của thuật toán Minimax là $O(b^m)$.

Để có thể tìm ra nhanh nước đi tốt (không phải là tối ưu) thay cho việc sử dụng hàm kết cuộc và xem xét tất cả các khả năng dẫn tới các trạng thái kết thúc, chúng ta sẽ sử dụng hàm đánh giá và chỉ xem xét một bộ phận của cây trò chơi.

HÀM ĐÁNH GIÁ

Hàm đánh giá eval ứng với mỗi trạng thái u của trò chơi với một giá trị số eval(u), giá trị này là sự đánh giá "độ lợi thế"của trạng thái u. Trạng thái u càng thuận lợi cho Trắng thì eval(u) là số dương càng lớn; u càng

thuận lợi cho Đen thì eval(u) là số âm càng nhỏ; eval(u) ≈ 0 đối với trạng thái không lợi thế cho ai cả.

Chất lượng của chương trình chơi cờ phụ thuộc rất nhiều vào hàm đánh giá. Nếu hàm đánh giá cho ta sự đánh giá không chính xác về các trạng thái, nó có thể hướng dẫn ta đi tới trạng thái được xem là tốt, nhưng thực tế lại rất bất lợi cho ta. Thiết kế một hàm đánh giá tốt là một việc khó, đòi hỏi ta phải quan tâm đến nhiều nhân tố: các quân còn lại của hai bên, sự bố trí của các quân đó,... Ở đây có sự mâu thuẫn giữa độ chính xác của hàm đánh giá và thời gian tính của nó. Hàm đánh giá chính xác sẽ đòi hỏi rất nhiều thời gian tính toán, mà người chơi lại bị giới hạn bởi thời gian phải đưa ra nước đi.

Ví dụ 1: Sau đây ta đưa ra một cách xây dựng hàm đánh giá đơn giản cho cờ vua. Mỗi loại quân được gán một giá trị số phù hợp với "sức mạnh"của nó. Chẳng hạn, mỗi tốt Trắng (Đen) được cho 1 (-1), mã hoặc tượng Trắng (Đen) được cho 3 (-3), xe Trắng (Đen) được cho 5 (-5) và hoàng hậu Trắng (Đen) được cho 9 (-9). Lấy tổng giá trị của tất cả các quân trong một trạng thái, ta sẽ được giá trị đánh giá của trạng thái đó. Hàm đánh giá như thế được gọi là hàm tuyến tính có trọng số, vì nó có thể biểu diễn dưới dạng:

$$s_1w_1 + s_2w_2 + \ldots + s_nw_n.$$

Trong đó, w_i là giá trị mỗi loại quân, còn s_i là số quân loại đó. Trong cách đánh giá này, ta đã không tính đến sự bố trí của các quân, các mối tương quan giữa chúng.

Ví dụ 2: Bây giờ ta đưa ra một cách đánh giá các trạng thái trong trò chơi Dodgem. Mỗi quân Trắng ở một vị trí trên bàn cờ được cho một giá trị tương ứng trong bảng bên trái hình 4.4. Còn mỗi quân Đen ở một vị trí sẽ

30	35	40
15	20	25
0	5	10

Giá trị quân Trắng.

-10	-25	-40
-5	-20	-35
0	-15	-30

Giá trị quân Đen.

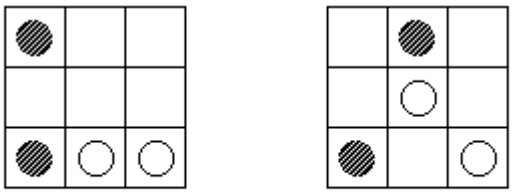
Hình 4.4 Đánh giá các quân trong trò chơi Dodgem.

được cho một giá trị tương ứng trong bảng bên phải hình 4.4:

Trắng cản trực tiếp Đen được thêm 40 điểm. Trắng cản gián tiếp Đen được thêm 30 điểm.

Hình 4.5 Đánh giá sự tương quan giữa quân Trắng và Đen.

Ngoài ra, nếu quân Trắng cản trực tiếp một quân Đen, nó được thêm 40 điểm, nếu cản gián tiếp nó được thêm 30 điểm (Xem hình 4.5). Tương tự, nếu quân Đen cản trực tiếp quân Trắng nó được thêm -40 điểm, còn cản gián tiếp nó được thêm -30 điểm.



75 -5

Hình 4.6 Giá trị của một số trạng thái trong trò chơi Dodgem.

Áp dụng các qui tắc trên, ta tính được giá trị của trạng thái ở bên trái hình 4.6 là 75, giá trị của trạng thái bên phải hình vẽ là -5.

Trong cánh đánh giá trên, ta đã xét đến vị trí của các quân và mối tương quan giữa các quân.

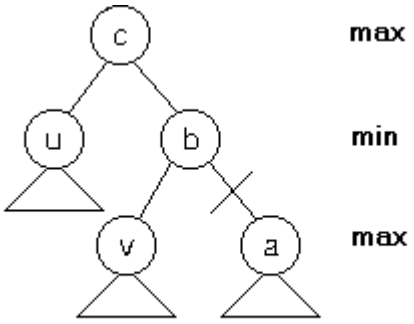
Một cách đơn giản để hạn chế không gian tìm kiếm là, khi cần xác định nước đi cho Trắng tại u, ta chỉ xem xét cây trò chơi gốc u tới độ cao h nào đó. Áp dụng thủ tục Minimax cho cây trò chơi gốc u, độ cao h và sử dụng giá trị của hàm đánh giá cho các lá của cây đó, chúng ta sẽ tìm được nước đi tốt cho Trắng tại u.

4.3. PHƯƠNG PHÁP CẮT CỤT ALPHA - BETA

Trong chiến lược tìm kiếm Minimax, để tìm kiếm nước đi tốt cho Trắng tại trạng thái u, cho dù ta hạn chế không gian tìm kiếm trong phạm vi cây trò chơi gốc u với độ cao h, thì số đỉnh của cây trò chơi này cũng còn rất lớn với $h \geq 3$. Chẳng hạn, trong cờ vua, nhân tố nhánh trong cây trò chơi trung bình khoảng 35, thời gian đòi hỏi phải đưa ra nước đi là 150 giây, với thời gian này trên máy tính thông thường chương trình của bạn chỉ có thể xem xét các đỉnh trong độ sâu 3 hoặc 4. Một người chơi cờ trình độ trung bình cũng có thể tính trước được 5, 6 nước hoặc hơn nữa, và do đó chương trình của bạn mới đạt trình độ người mới tập chơi!

Khi đánh giá đỉnh u tới độ sâu h, thuật toán Minimax đòi hỏi ta phải đánh giá tất cả các đỉnh của cây gốc u tới độ sâu h. Song ta có thể giảm bớt số đỉnh cần phải dánh giá mà vẫn không ảnh hưởng gì đến sự đánh giá đỉnh u. Phương pháp cắt cụt alpha-beta cho phép ta cắt bỏ các nhánh không cần thiết cho sự đánh giá đỉnh u.

Tư tưởng của kỹ thuật cắt cụt alpha-beta là như sau: Nhớ lại rằng, chiến lược tìm kiếm Minimax là chiến lược tìm kiếm theo độ sâu. Giả sử



Hình 4.7 Cắt bỏ cây con gốc a, nếu eval(u) > eval(v).

trong quá trình tìm kiếm ta đi xuống đỉnh a là đỉnh Trắng, đỉnh a có người anh em v đã được đánh giá. Giả sử cha của đỉnh a là b và b có người anh em u dã được đánh giá, và giả sử cha của b là c (Xem hình 4.7). Khi đó ta có giá trị đỉnh c (đỉnh Trắng) ít nhất là giá trị của u, giá trị của đỉnh b (đỉnh Đen) nhiều nhất là giá trị v. Do đó, nếu eval(u) > eval(v), ta không cần đi xuống để đánh giá đỉnh a nữa mà vẫn không ảnh hưởng gì dến đánh giá đỉnh c. Hay nói cách khác ta có thể cắt bỏ cây con gốc a. Lập luận tương tự cho trường hợp a là đỉnh Đen, trong trường hợp này nếu eval(u) < eval(v) ta cũng có thể cắt bỏ cây con gốc a.

Để cài đặt kỹ thuật cắt cụt alpha-beta, đối với các đỉnh nằm trên đường đi từ gốc tới đỉnh hiện thời, ta sử dụng tham số α để ghi lại giá trị lớn nhất trong các giá trị của các đỉnh con đã đánh giá của một đỉnh Trắng, còn tham số β ghi lại giá trị nhỏ nhất trong các đỉnh con đã đánh giá của một đỉnh Đen. Giá trị của α và β sẽ được cập nhật trong quá trình tìm kiếm. α và β được sử dụng như các biến địa phương trong các hàm MaxVal(u, α, β) (hàm xác định giá trị của đỉnh Trắng u) và Minval(u, α, β) (hàm xác định giá trị của đỉnh Đen u).

```
function MaxVal(u, α, β);
begin
        if u là lá của cây hạn chế hoặc u là đỉnh kết thúc
        then  MaxVal ← eval(u)
        else
          for mỗi đỉnh v là con của u do
        {α ← max[α, MinVal(v, α, β)];
         // Cắt bỏ các cây con từ các đỉnh v còn lại
         if α ≥ β then exit};
```

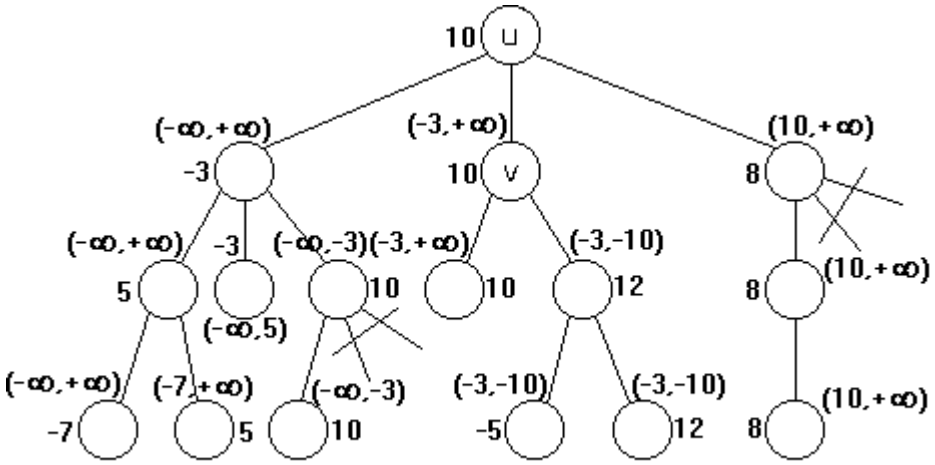
```
        MaxVal ← α;
end;

function MinVal(u, α, β);
begin
        if u là lá của cây hạn chế hoặc u là đỉnh kết thúc
        then MinVal ← eval(u)
        else
            for mỗi đỉnh v là con của u do
                {β ← min[β, MaxVal(v, α, β)];
                 // Cắt bỏ các cây con từ các đỉnh v còn lại
        if α ≥ β then exit};
        MinVal ← β;
end;
```

Thuật toán tìm nước đi cho Trắng sử dụng kỹ thuật cắt cụt alpha-beta, được cài đặt bởi thủ tục Alpha_beta(u,v), trong đó v là tham biến ghi lại đỉnh mà Trắng cần đi tới từ u.

```
procedure Alpha_beta(u,v);
begin
        α ← -∞;
        β ← ∞;
      for mỗi đỉnh w là con của u do
            if α ≤ MinVal(w, α, β) then
                {α ← MinVal(w, α, β);
                 v ← w;}
end;
```

Ví dụ. Xét cây trò chơi gốc u (đỉnh Trắng) giới hạn bởi độ cao h = 3 (hình 4.8). Số ghi cạnh các lá là giá trị của hàm đánh giá. Áp dụng chiến lược Minimax và kỹ thuật cắt cụt, ta xác định được nước đi tốt nhất cho Trắng tại u, đó là nước đi dẫn tới đỉnh v có giá trị 10. Cạnh mỗi đỉnh ta cũng cho giá trị của cặp tham số (α, β). Khi gọi các hàm MaxVal và MinVal để xác định giá trị của đỉnh đó. Các nhánh bị cắt bỏ được chỉ ra trong hình:



Hình 4.8 Xác định giá trị các đỉnh bằng kỹ thuật cắt cụt.

PHẦN II

TRI THỨC VÀ LẬP LUẬN

Hai thành phần cơ bản của một hệ dựa trên tri thức (khnowledge based system) là cơ sở tri thức và bộ suy diễn. Để tạo ra các phương pháp và kỹ thuật xây dựng cơ sở tri thức và bộ suy diễn cho một hệ dựa trên tri thức, chúng ta cần phải nghiên cứu các mô hình biểu diễn tri thức và lập luận. Biểu diễn tri thức và lập luận là lĩnh vực nghiên cứu trung tâm của trí tuệ nhân tạo. Trong phần này chúng ta sẽ xét các ngôn ngữ biểu diễn tri thức và các phương pháp lập luận trong từng ngôn ngữ đó. Ngôn ngữ biểu diễn tri thức quan trọng nhất là logic vị từ cấp một (sẽ được xét trong chương 6). Logic vị từ cấp một là cơ sở để xây dựng nhiều ngôn ngữ biểu diễn khác. Các ngôn ngữ con của logic vị từ cấp một (ngôn ngữ các luật, ngôn ngữ mô tả khái niệm) sẽ được xét trong các chương 7 và 9. Chương 8 dành cho việc nghiên cứu các lập luận không đơn điệu. Vấn đề biểu diễn tri thức không chắc chắn sẽ được trình bày trong chương 10. Chương 11 trình bày logic mờ và lập luận xấp xỉ

CHƯƠNG 5
LOGIC MỆNH ĐỀ

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

5.1. BIỂU DIỄN TRI THỨC

Con người sống trong môi trường có thể nhận thức được thế giới nhờ các giác quan (tai, mắt và các giác quan khác), sử dụng các tri thức tích luỹ được và nhờ khả năng lập luận, suy diễn, con người có thể đưa ra các hành động hợp lý cho công việc mà con người đang làm. Một mục tiêu của Trí tuệ nhân tạo ứng dụng là thiết kế các ***tác nhân thông minh*** (intelligent agent) cũng có khả năng đó như con người. Chúng ta có thể hiểu tác nhân thông minh là bất cứ cái gì có thể nhận thức được môi trường thông qua các ***bộ cảm nhận*** (sensors) và đưa ra hành động hợp lý đáp ứng lại môi trường thông qua ***bộ phận hành động*** (effectors). Các robots, các softbot (software robot), các hệ chuyên gia,... là các ví dụ về tác nhân thông minh. Các tác nhân thông minh cần phải có tri thức về thế giới hiện thực mới có thể đưa ra các quyết định đúng đắn.

Thành phần trung tâm của các ***tác nhân dựa trên tri thức*** (knowledge-based agent), còn được gọi là ***hệ dựa trên tri thức*** (knowledge-based system) hoặc đơn giản là hệ tri thức, là cơ sở tri thức. Cơ sở tri thức (CSTT) là một tập hợp các tri thức được biểu diễn dưới dạng nào đó. Mỗi khi nhận được các thông tin đưa vào, tác nhân cần có khả năng suy diễn để đưa ra các câu trả lời, các hành động hợp lý, đúng đắn. Nhiệm vụ này được thực hiện bởi bộ suy diễn. Bộ suy diễn là thành phần cơ bản khác của các hệ tri thức. Như vậy hệ tri thức bảo trì một CSTT và được trang bị một thủ tục suy diễn. Mỗi khi tiếp nhận được các sự kiện từ môi trường, thủ tục suy diễn thực hiện quá trình liên kết các sự kiện với các tri thức trong CSTT để rút ra

các câu trả lời, hoặc các hành động hợp lý mà tác nhân cần thực hiện. Đương nhiên là, khi ta thiết kế một tác nhân giải quyết một vấn đề nào đó thì CSTT sẽ chứa các tri thức về miền đối tượng cụ thể đó. Để máy tính có thể sử dụng được tri thức, có thể xử lý tri thức, chúng ta cần biểu diễn tri thức dưới dạng thuận tiện cho máy tính. Đó là mục tiêu của biểu diễn tri thức.

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

- Cú pháp của một ngôn ngữ bao gồm các ký hiệu và các quy tắc liên kết các ký hiệu (các luật cú pháp) để tạo thành các câu (công thức) trong ngôn ngữ. Các câu ở đây là biểu diễn ngoài, cần phân biệt với biểu diễn bên trong máy tính. Các câu sẽ được chuyển thành các cấu trúc dữ liệu thích hợp được cài đặt trong một vùng nhớ nào đó của máy tính, đó là biểu diễn bên trong. Bản thân các câu chưa chứa đựng một nội dung nào cả, chưa mang một ý nghĩa nào cả.
- Ngữ nghĩa của ngôn ngữ cho phép ta xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực. Chẳng hạn, trong ngôn ngữ các biểu thức số học, dãy ký hiệu (x+y)*z là một câu viết đúng cú pháp. Ngữ nghĩa của ngôn ngữ này cho phép ta hiểu rằng, nếu x, y, z, ứng với các số nguyên, ký hiệu + ứng với phép toán cộng, còn * ứng với phép chia, thì biểu thức (x+y)*z biểu diễn quá trình tính toán: lấy số nguyên x cộng với số nguyên y, kết quả được nhân với số nguyên z.
- Ngoài hai thành phần cú pháp và ngữ nghĩa, ngôn ngữ biểu diễn tri thức cần được cung cấp ***cơ chế suy diễn***. Một luật suy diễn (rule of inference) cho phép ta suy ra một công thức từ một tập nào đó các công thức. Chẳng hạn, trong logic mệnh đề, luật modus ponens cho phép từ hai công thức A và A⇒B suy ra công thức B. Chúng ta sẽ hiểu ***lập luận*** hoặc ***suy diễn*** là một quá trình áp dụng các luật suy diễn để từ các tri thức trong cơ sở tri thức và các sự kiện ta nhận được các tri thức mới. Như vậy chúng ta xác định:

Ngôn ngữ biểu diễn tri thức = Cú pháp + Ngữ nghĩa + Cơ chế suy diễn.

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

Trong sách này, chúng ta sẽ tập trung nghiên cứu logic vị từ cấp một (first-order predicate logic hoặc first-order predicate calculus) - một ngôn ngữ biểu diễn tri thức, bởi vì logic vị từ cấp một có khả năng biểu diễn tương đối tốt, và hơn nữa nó là cơ sở cho nhiều ngôn ngữ biểu diễn tri thức khác, chẳng hạn toán hoàn cảnh (situation calculus) hoặc logic thời gian khoảng cấp một (first-order interval tempral logic). Nhưng trước hết chúng ta sẽ nghiên cứu logic mệnh đề (propositional logic hoặc propositional calculus). Nó là ngôn ngữ rất đơn giản, có khả năng biểu diễn hạn chế, song thuận tiện cho ta đưa vào nhiều khái niệm quan trọng trong logic.

5.2. CÚ PHÁP VÀ NGỮ NGHĨA CỦA LOGIC MỆNH ĐỀ

5.2.1. Cú Pháp

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

1. Các ký hiệu

Hai hằng logic True và False.

Các ký hiệu mệnh đề (còn được gọi là các biến mệnh đề): P, Q,...

Các kết nối logic $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$.

Các dấu mở ngoặc (và đóng ngoặc).

2. Các quy tắc xây dựng các công thức

Các biến mệnh đề là công thức.

Nếu A và B là công thức thì:

($A \wedge B$) (đọc "A hội B"hoặc "A và B")

($A \vee B$) (đọc "A tuyển B"hoặc "A hoặc B")

($\neg A$) (đọc "phủ định A")

($A \Rightarrow B$) (đọc "A kéo theo B"hoặc "nếu A thì B")

(A⇔B) (đọc "A và B kéo theo nhau")

là các công thức.

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

Các công thức là các ký hiệu mệnh đề sẽ được gọi là các ***câu đơn*** hoặc ***câu phân tử***. Các công thức không phải là câu đơn sẽ được gọi là câu phức hợp. Nếu P là ký hiệu mệnh đề thì P và ⌉P được gọi là ***literal*,** P là ***literal dương***, còn ⌉P là ***literal âm***. Câu phức hợp có dạng $A_1 \vee ... \vee A_m$ trong đó A_i là các literal sẽ được gọi là ***câu tuyển*** (clause).

5.2.2. Ngữ nghĩa

Ngữ nghĩa của logic mệnh đề cho phép ta ***xác định*** ý nghĩa của các công thức trong thế giới hiện thực nào đó. Điều đó được thực hiện bằng cách kết hợp mỗi ký hiệu mệnh đề với sự kiện nào đó trong thế giới hiện thực. Chẳng hạn, ký hiệu mệnh đề P có thể ứng với sự kiện "Paris là thủ đô nước Pháp" hoặc bất kỳ một sự kiện nào khác. Bất kỳ một sự kết hợp các kí hiệu mệnh đề với các sự kiện trong thế giới thực được gọi là một ***minh họa*** (interpretation). Chẳng hạn minh họa của kí hiệu mệnh đề P có thể là một sự kiện (mệnh đề) "Paris là thủ đô nước Pháp ". Một sự kiện chỉ có thể đúng hoặc sai. Chẳng hạn, sự kiện "Paris là thủ đô nước Pháp "là đúng, còn sự kiện "Số Pi là số hữu tỉ "là sai.

Một cách chính xác hơn, ta hiểu một minh họa là một cách gán cho mỗi ký hiệu mệnh đề một giá trị chân lý **True** hoặc **False.** Trong một minh họa, nếu kí hiệu mệnh đề P được gán giá trị chân lý **True/False** (P σ**True**/ Pσ**False**) thì ta nói mệnh đề P **đúng/sai** trong minh họa đó. Trong một minh họa, ý nghĩa của các câu phức hợp được xác định bởi ý nghĩa của các kết nối logic. Chúng ta xác định ý nghĩa của các kết nối logic trong các bảng chân lý (xem hình 5.1)

P	Q	lP	P∧Q	P ∨ Q	P⇒Q	P⇔Q
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Hình 5.1 Bảng chân lý của các kết nối logic

Ý nghĩa của các kết nối logic ∧, ∨ và l được xác định như ý nghĩa của các từ "**và**","**hoặc là**"và "**phủ định**"trong ngôn ngữ tự nhiên. Chúng ta cần phải giải thích thêm về ý nghĩa của phép kéo theo P ⇒ Q (P kéo theo Q), P là giả thiết, còn Q là kết luận. Trực quan cho phép ta xem rằng, khi P là đúng và Q là đúng thì câu "P kéo theo Q "là đúng, còn khi P là đúng Q là sai thì câu "P kéo theo Q"là sai. Nhưng nếu P sai và Q đúng, hoặc P sai Q sai thì "P kéo theo Q"là đúng hay sai ? Nếu chúng ta xuất phát từ giả thiết sai, thì chúng ta không thể khẳng định gì về kết luận. Không có lý do gì để nói rằng, nếu P sai và Q đúng hoặc P sai và Q sai thì "P kéo theo Q"là sai. Do đó trong trường hợp P sai thì "P kéo theo Q "là đúng dù Q là đúng hay Q là sai.

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

Một công thức được gọi là ***thoả được*** (satisfiable) nếu nó đúng trong một minh họa nào đó. Chẳng hạn công thức (P∨ Q) ∧lS là thoả được, vì nó có giá trị **True** trong minh họa {P σ **True**, QσFalse, Sσ **True**}.

Một công thức được gọi là ***vững chắc*** **(**valid hoặc tautology) nếu nó đúng trong mọi minh họa chẳng hạn câu P ∨lP là vững chắc.

Một công thức được gọi là ***không thoả được*****,** nếu nó là sai trong mọi minh họa. Chẳng hạn công thức P ∧ lP.

Chúng ta sẽ gọi một ***mô hình*** (model) của một công thức là một minh họa sao cho công thức là đúng trong minh họa này. Như vậy một công thức *thoả được* là công thức có một mô hình. Chẳng hạn, minh họa {P σ **False,** Q σ **False**, Sσ**True** } là một mô hình của công thức (P =>Q) ∧ S.

Bằng cách lập bảng chân lý (***phương pháp bảng chân lý)*** ta có thể xác định được một công thức có ***thoả được*** hay không. Trong bảng này, mỗi biến mệnh đề đứng đầu một cột, công thức cần kiểm tra đứng đầu một cột, mỗi dòng tương ứng với một minh họa. Chẳng hạn hình 5.2 là bảng chân lý cho công thức (P=>Q) ∧S. Trong bảng chân lý này ta cần đưa vào các cột phụ ứng với các công thức con của các công thức cần kiểm tra để việc tính giá trị của công thức này được dễ dàng. Từ bảng chân lý ta thấy rằng công thức (P=>Q) ∧S là ***thoả được*** nhưng không ***vững chắc*****.**

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

Hình 5.2 Bảng chân lý cho công thức (P⇒Q) ∧S

Cần lưu ý rằng, một công thức chứa n biến, thì số các minh họa của nó là 2^n, tức là bảng chân lý có 2^n dòng. Như vậy việc kiểm tra một công thức có *thoả được* hay không bằng phương pháp bảng chân lý, đòi hỏi thời gian mũ. Cook (1971) đã chứng minh rằng, vấn đề kiểm tra một công thức trong logic mệnh đề có *thoả được* hay không là vấn đề NP-đầy đủ.

Chúng ta sẽ nói rằng một tập công thức G = {G_1,..,G_m } là vững chắc (***thoả được, không thoả được***) nếu hội của chúng $G_1 \wedge ..... \wedge G_m$ là ***vững chắc*** (***thoả được, không thoả được***). Một mô hình của tập công thức G là mô hình của công thức $G_1 \wedge ..... \wedge G_{m.}$.

5.3. DẠNG CHUẨN TẮC

Trong mục này chúng ta sẽ xét việc chuẩn hóa các công thức, đưa các công thức về dạng thuận lợi cho việc lập luận, suy diễn. Trước hết ta sẽ xét các phép biến đổi tương đương. Sử dụng các phép biển đổi này, ta có thể đưa một công thức bất kỳ về dạng chuẩn tắc.

5.3.1. Sự tương đương của các công thức

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

$A \Rightarrow B \equiv lA \vee B$

$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$

$l(lA) \equiv A$

1. **Luật De Morgan**

$l(A \vee B) \equiv lA \wedge lB$

$l(A \wedge B) \equiv lA \vee lB$

2. **Luật giao hoán**

$A \vee B \equiv B \vee A$

$A \wedge B \equiv B \wedge A$

3. **Luật kết hợp**

$(A \vee B) \vee C \equiv A \vee (B \vee C)$

$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$

4. **Luật phân phối**

$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$

$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

5.3.2. Dạng chuẩn tắc

Các công thức tương đương có thể xem như các biểu diễn khác nhau của cùng một sự kiện. Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta sẽ chuẩn hóa các công thức, đưa chúng về dạng biểu diễn chuẩn được gọi là ***dạng chuẩn hội***. Một công thức ở dạng chuẩn hội nếu nó là hội của các câu tuyển. Nhớ lại rằng, câu tuyển có dạng $A_1 \vee ....\vee A_m$ trong đó các A_i là ***literal***. Chúng ta có thể biến đổi một công thức bất kỳ về công thức ở dạng chuẩn hội bằng cách áp dụng thủ tục sau.

- Bỏ các dấu kéo theo ($\Rightarrow$) bằng cách thay $(A \Rightarrow B)$ bởi $(lA \vee B)$.
- Chuyển các dấu phủ định (l) vào sát các ký hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay $l(lA)$ bởi A.
- Áp dụng luật phân phối, thay các công thức có dạng $A \vee (B \wedge C)$ bởi $(A \vee B) \wedge (A \vee B)$.

Ví dụ: Ta chuẩn hóa công thức $(P \Rightarrow Q) \vee l(R \vee lS)$:

$$(P \Rightarrow Q) \vee l(R \vee lS) \equiv (lP \vee Q) \vee (lR \wedge S)$$

$$\equiv ((lP \vee Q) \vee lR) \wedge ((lP \vee Q) \vee S)$$

$$\equiv (l\,P \vee Q \vee lR) \wedge (lP \vee Q \vee S).$$

Như vậy công thức $(P \Rightarrow Q) \vee l(R \vee lS)$ được đưa về dạng chuẩn hội $(lP \vee Q \vee lR) \wedge (lP \vee Q \vee S)$.

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

5.3.3. Các câu Horn

Ở trên ta đã chỉ ra, mọi công thức đều có thể đưa về dạng chuẩn hội, tức là hội của các tuyển, mỗi câu tuyển có dạng:

$$lP_1 \vee ..... \vee lP_m \vee Q_1 \vee ..... \vee Q_n$$

trong đó P_i, Q_i là các ký hiệu mệnh đề (literal dương) câu này tương đương với câu:

$$P_1 \wedge ..... \wedge lP_m \Rightarrow Q_1 \vee ..... \vee Q_n$$

Dạng câu này được gọi là ***câu Kowalski*** (do nhà logic Kowalski đưa ra năm 1971).

Khi n <=1, tức là câu tuyển chỉ chứa nhiều nhất một literal dương, ta có một dạng câu đặc biệt quan trọng được gọi là ***câu Horn*** (mang tên nhà logic Alfred Horn, năm 1951).

Nếu m>0, n=1, câu Horn có dạng:

$$P_1 \wedge ..... \wedge P_m \Rightarrow Q$$

Trong đó P_i , Q là các literal dương. Các P_i được gọi là các điều kiện (hoặc giả thiết), còn Q được gọi là kết luận (hoặc hệ quả). Các câu Horn dạng này còn được gọi là các luật ***if-then*** và được biểu diễn như sau:

If P_1 ***and****....****and*** P_m ***then*** Q.

Khi m=0, n=1 câu Horn trở thành câu đơn Q, hay sự kiện Q. Nếu m>0, n=0 câu Horn trở thành dạng lP_1 v.....v lP_m hay tương đương $l(P_1 \wedge ... \wedge P_m)$.

Cần chú ý rằng, không phải mọi công thức đều có thể biểu diễn dưới dạng hội của các câu Horn. Tuy nhiên trong các ứng dụng, cơ sở tri thức

thường là một tập nào đó các câu Horn (tức là một tập nào đó các luật if-then).

5.4. LUẬT SUY DIỄN

Một công thức H được xem là ***hệ qủa logic*** (logical consequence) của một tập công thức $G = \{G_1,.....,G_m\}$ nếu trong bất kỳ minh họa nào mà $\{G_1,.....,G_m\}$ đúng thì H cũng đúng, hay nói cách khác bất kỳ mô hình nào của *G* cũng là mô hình của H.

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

Sau đây là một số luật suy diễn quan trọng trong logic mệnh đề. Trong các luật này α, α_i, β, γ là các công thức:

- **Luật Modus Ponens**

$$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$

Từ một kéo theo và giả thiết của kéo theo, ta suy ra kết luận của nó.

- **Luật Modus Tollens**

$$\frac{\alpha \Rightarrow \beta,\ \neg \beta}{\neg \alpha}$$

Từ một kéo theo và phủ định kết luận của nó, ta suy ra phủ định giả thiết của kéo theo.

- **Luật bắc cầu**

$$\frac{\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma}$$

Từ hai kéo theo, mà kết luận của kéo theo thứ nhất trùng với giả thiết của kéo theo thứ hai, ta suy ra kéo theo mới mà giả thiết của nó là giả thiết của kéo theo thứ nhất, còn kết luận của nó là kết luận của kéo theo thứ hai.

- **Luật loại bỏ hội**

$$\frac{\alpha_1 \wedge \ldots\ldots \wedge \alpha_i \wedge \ldots\ldots\wedge\alpha_m}{\alpha_i}$$

Từ một hội ta suy ra một nhân tử bất kỳ của hội.

- **Luật đưa vào hội**

$$\frac{\alpha_1,\ldots\ldots,\alpha_i,\ldots\ldots\alpha_m}{\alpha_1 \wedge \ldots\ldots\wedge\alpha_i \wedge \ldots\ldots \wedge\alpha_m}$$

Từ một danh sách các công thức, ta suy ra hội của chúng.

- **Luật đưa vào tuyển**

$$\frac{\alpha_i}{\alpha_1\vee\ldots\ldots\vee \alpha_i. \vee\ldots\ldots\vee \alpha_m}$$

Từ một công thức, ta suy ra một tuyển mà một trong các hạng tử của tuyển là công thức đó.

- **Luật phân giải**

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

Từ hai tuyển, một tuyển chứa một hạng tử đối lập với một hạng tử trong tuyển kia, ta suy ra tuyển của các hạng tử còn lại trong cả hai tuyển.

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

Bằng phương pháp bảng chân lý, ta có thể kiểm chứng được các luật suy diễn nêu trên đều là tin cậy. Bảng chân lý của luật phân giải được cho trong hình 5.3. Từ bảng này ta thấy rằng, trong bất kỳ một minh họa nào mà

cả hai giả thiết $\alpha \vee \beta$, $\neg \beta \vee \gamma$ đúng thì kết luận $\alpha \vee \gamma$ cũng đúng. Do đó luật phân giải là luật suy điễn tin cậy.

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

Hình 5.3 Bảng chân lý chứng minh tính tin cậy của luật phân giải.

Ta có nhận xét rằng, luật phân giải là một luật suy diễn tổng quát, nó bao gồm luật Modus Ponens, luật Modus Tollens, luật bắc cầu như các trường hợp riêng. (Bạn đọc dễ dàng chứng minh được điều đó).

Tiên đề, định lý, chứng minh.

Giả sử chúng ta có một tập nào đó các công thức. Các luật suy diễn cho phép ta từ các công thức đã có suy ra công thức mới bằng một dãy áp dụng các luật suy diễn. Các công thức đã cho được gọi là các ***tiên đề***. Các công thức được suy ra được gọi là các ***định lý***. Dãy các luật được áp dụng để dẫn tới định lý được gọi là một ***chứng minh*** của định lý. Nếu các luật suy diễn là tin cậy, thì các định lý là hệ quả logic của các tiên đề.

Ví dụ: Giả sử ta có các công thức sau:

$Q \wedge S \Rightarrow G \vee H$ (1)

$P \Rightarrow Q$ (2)

$R \Rightarrow S$ (3)

P (4)

R (5)

Giả sử ta cần chứng minh công thức $G \vee H$. Từ công thức (2) và (4), ta suy ra Q (Luật Modus Ponens). Lại áp dụng luật Modus Ponens, từ (3) và (5) ta suy ra S. Từ Q, S ta suy ra $Q \wedge S$ (luật đưa vào hội). Từ (1) và $Q \wedge S$ ta suy ra $G \vee H$. Công thức $G \vee H$ đã được chứng minh.

Trong các hệ tri thức, chẳng hạn các hệ chuyên gia, hệ lập trình logic,..., sử dụng các luật suy diễn người ta thiết kế lên các ***thủ tục suy diễn*** (còn được gọi là ***thủ tục chứng minh***) để từ các tri thức trong cơ sở tri thức ta suy ra các tri thức mới đáp ứng nhu cầu của người sử dụng.

Một ***hệ hình thức*** (formal system) bao gồm một tập các tiên đề và một tập các luật suy diễn nào đó (trong ngôn ngữ biểu diễn tri thức nào đó).

Một tập luật suy diễn được xem là ***đầy đủ***, nếu mọi hệ quả logic của một tập các tiên đề đều chứng minh được bằng cách chỉ sử dụng các luật của tập đó.

PHƯƠNG PHÁP CHỨNG MINH BÁC BỎ

Phương pháp chứng minh bác bỏ (refutation proof hoặc proof by contradiction) là một phương pháp thường xuyên được sử dụng trong các chứng minh toán học. Tư tưởng của phương pháp này là như sau: Để chứng minh P đúng, ta giả sử P sai (thêm $\neg P$ vào các giả thiết) và dẫn tới một mâu thuẫn. Sau đây ta sẽ trình bầy cơ sở của phương pháp chứng minh này.

Giả sử chúng ta có một tập các công thức $G = \{G_1,.....,G_m\}$ ta cần chứng minh công thức H là hệ quả logic của G**.** Điều đó tương đương với chứng minh công thức $G_1 \wedge .... \wedge G_m \Rightarrow H$ là vững chắc. Thay cho chứng minh $G_1 \wedge ..... \wedge G_m \Rightarrow H$ là vững chắc, ta chứng minh $G_1 \wedge .... \wedge G_m \wedge \neg H$ là không thỏa mãn được. Tức là ta chứng minh tập $G' = (G_1,.....,G_m, \neg H)$ là không thỏa được. G sẽ không thoả được nếu từ G'ta suy ra hai mệnh đề đối lập nhau. Việc chứng minh công thức H là hệ quả logic của tập các tiêu đề G bằng cách chứng minh tính không thỏa được của tập các tiêu đề được thêm vào phủ định của công thức cần chứng minh, được gọi là chứng minh bác bỏ.

5.5. LUẬT PHÂN GIẢI. CHỨNG MINH BÁC BỎ BẰNG LUẬT PHÂN GIẢI

Để thuận tiện cho việc sử dụng luật phân giải, chúng ta sẽ cụ thể hoá luật phân giải trên các dạng câu đặc biệt quan trọng.

- Luật phân giải trên các câu tuyển

$$\frac{A_1 \vee \ldots\ldots\ldots\ldots \vee A_m \vee C \qquad \neg C \vee B_1 \vee \ldots\ldots\ldots\ldots \vee B_n}{A_1 \vee \ldots\ldots\ldots \vee A_m \vee B_1 \vee \ldots \vee B_n}$$

trong đó A_i, B_j và C là các literal.

- Luật phân giải trên các câu Horn:

Giả sử P_i, R_j, Q và S là các literal. Khi đó ta có các luật sau:

$$\frac{P_1 \wedge \ldots\ldots\ldots\ldots \wedge P_m \wedge S \Rightarrow Q, \qquad R_1 \wedge \ldots\ldots\ldots\ldots \wedge R_n \Rightarrow S}{P_1 \wedge \ldots\ldots\ldots \wedge P_m \wedge R_1 \wedge \ldots\ldots \wedge R_n \Rightarrow Q}$$

Một trường hợp riêng hay được sử dụng của luật trên là:

$$\frac{P_1 \wedge \ldots\ldots \wedge P_m \wedge S \Rightarrow Q, \qquad S}{P_1 \wedge \ldots\ldots\ldots \wedge P_m \Rightarrow Q}$$

Khi ta có thể áp dụng luật phân giải cho hai câu, thì hai câu này được gọi là ***hai câu phân giải được*** và kết quả nhận được khi áp dụng luật phân giải cho hai câu đó được gọi là ***phân giải thức*** của chúng. Phân giải thức của hai câu A và B được kí hiệu là res(A,B). Chẳng hạn, hai câu tuyển phân giải được nếu một câu chứa một literal đối lập với một literal trong câu kia. Phân giải thức của hai literal đối lập nhau (P và $\neg$ P) là câu rỗng, chúng ta sẽ ký hiệu câu rỗng là [], câu rỗng không thoả được.

Giả sử G là một tập các câu tuyển (bằng cách chuẩn hoá ta có thể đưa một tập các công thức về một tập các câu tuyển). Ta sẽ ký hiệu R(G) là tập

câu bao gồm các câu thuộc G và tất cả các câu nhận được từ G bằng một dãy áp dụng luật phân giải.

Luật phân giải là luật đầy đủ để chứng minh một tập câu là không thỏa được. Điều này được suy từ định lý sau:

ĐỊNH LÝ PHÂN GIẢI

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

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

Từ định lý phân giải, ta đưa ra thủ tục sau đây để xác định một tập câu tuyển G là thỏa được hay không. Thủ tục này được gọi là thủ tục phân giải.

procedure *Resolution;*
Input: tập G các câu tuyển ;
begin
*1.**Repeat***
1.1 Chọn hai câu A và B thuộc G;
*1.2 **if** A và B phân giải được **then** tính Res(A,B);*
*1.3 **if** Res(A,B)là câu mới **then** thêm Res(A,B)vào G;*
until *nhận được [] hoặc không có câu mới xuất hiện;*
*2. **if** nhận được câu rỗng **then** thông báo G không thoả được*
else *thông báo G thoả được;*
end

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

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

Sau đây là thủ tục chứng minh bác bỏ bằng luật phân giải

Procedure *Refutation_Proof;*
input: Tập G các công thức;
Công thức cần chứng minh H;
Begin
1.Thêm ¬H vào G;
2.Chuyển các công thức trong G về dạng chuẩn hội;
3.Từ các dạng chuẩn hội ở bước hai, thành lập tập các câu tuyển G';
4. áp dụng thủ tục phân giải cho tập câu G';
5. **if** *G' không thoả được* **then** *thông báo H là hệ quả logic*
else *thông báo H không là hệ quả logic của G;*
end;

Ví dụ: Giả giử G là tập hợp các câu tuyển sau

$\neg A \vee \neg B \vee P$ (1)

$\neg C \vee \neg D \vee P$ (2)

$\neg E \vee C$ (3)

A (4)

E (5)

D (6)

Giả sử ta cần chứng minh P. Thêm vào G câu sau:

$\neg P$ (7)

áp dụng luật phân giải cho câu (2) và (7) ta được câu:

$\neg C \vee \neg D$ (8)

Từ câu (6) và (8) ta nhận được câu:

$\neg C$ (9)

Từ câu (3) và (9) ta nhận được câu:

$\neg E$ (10)

Tới đây đã xuất hiện mâu thuẫn, vì câu (5) và (10) đối lập nhau. Từ câu (5) và (10) ta nhận được câu rỗng [].

Vậy P là hệ quả logic của các câu (1) --(6).

CHƯƠNG 6

LOGIC VỊ TỪ CẤP MỘT

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

Thế giới hiện thực bao gồm các ***đối tượng***, mỗi đối tượng có những ***tính chất*** riêng để phân biệt nó với các đối tượng khác. Các đối tượng lại có ***quan hệ*** với nhau. Các mối quan hệ rất đa dạng và phong phú. Chúng ta có thể liệt kê ra rất nhiều ví dụ về đối tượng, tính chất, quan hệ.

- Đối tượng: một cái bàn, một cái nhà, một cái cây, một con người, một con số....
- Tính chất: Cái bàn có thể có tính chất: có bốn chân, làm bằng gỗ, không có ngăn kéo. Con số có thể có tính chất là số nguyên, số hữu tỉ, là số chính phương...
- Quan hệ: cha con, anh em, bè bạn (giữa con người); lớn hơn, nhỏ hơn, bằng nhau (giữa các con số); bên trong, bên ngoài nằm trên nằm dưới (giữa các đồ vật)...
- Hàm: Một trường hợp riêng của quan hệ là quan hệ hàm. Chẳng hạn, vì mỗi người có một mẹ, do đó ta có quan hệ hàm ứng mỗi người với mẹ của nó.

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

Chương này dành cho nghiên cứu logic vị từ cấp một với tư cách là một ngôn ngữ biểu diễn tri thức. Logic vị từ cấp một đóng vai trò cực kì quan trọng trong biểu diễn tri thức, vì khả năng biểu diễn của nó (nó cho phép ta biểu diễn tri thức về thế giới với các đối tượng, các thuộc tính của đối tượng và các quan hệ của đối tượng), và hơn nữa, nó là cơ sở cho nhiều ngôn ngữ logic khác.

6.1. CÚ PHÁP VÀ NGỮ NGHĨA CỦA LÔGIC VỊ TỪ CẤP MỘT

6.1.1. Cú pháp.

CÁC KÍ HIỆU

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

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

Mỗi vị từ là vị từ của n biến (n≥0). Chẳng hạn Like là vị từ của hai biến, Prime là vị từ một biến. Các ký hiệu vị từ không biến là các ký hiệu mệnh đề.

Các ký hiệu hàm: f, g, cos, sin, mother, husband, distance,...

Mỗi hàm là hàm của n biến (n≥1). Chẳng hạn, cos, sin là hàm một biến, distance là hàm của ba biến.

Các ký hiệu kết nối logic: $\wedge$ (hội), $\vee$ (tuyển), $\neg$ (phủ định), $\Rightarrow$(kéo theo), $\Leftrightarrow$ (kéo theo nhau).

Các ký hiệu lượng tử: $\forall$ (với mọi), $\exists$ (tồn tại).

Các ký hiệu ngăn cách: dấu phẩy, dấu mở ngoặc và dấu đóng ngoặc.

CÁC HẠNG THỨC

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

- Các ký hiệu hằng và các ký hiệu biến là hạng thức.

- Nếu t_1, t_2, t_3,..., t_n là n hạng thức và f là một ký hiệu hàm n biến thì f(t_1, t_2,..., t_n) là hạng thức. Một hạng thức không chứa biến được gọi là một ***hạng thức cụ thể*** (ground term).

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

CÁC CÔNG THỨC PHÂN TỬ

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

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

1. Các ký hiệu vị từ không biến (các ký hiệu mệnh đề) là công thức phân tử.
2. Nếu t_1, t_2,...,t_n là n hạng thức và P là vị từ của n biến thì P(t_1,t_2,...,t_n) là công thức phân tử.

Chẳng hạn, Hoa là một ký hiệu hằng, Love là một vị từ của hai biến, husband là hàm của một biến, thì Love(Hoa, husband(Hoa)) là một công thức phân tử.

CÁC CÔNG THỨC

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

Các công thức được xác định đệ quy như sau:

- Các công thức phân tử là công thức.
- Nếu G và H là các công thức, thì các biểu thức ($G \wedge H$), ($G \vee H$), ($\neg G$), ($G \Rightarrow H$), ($G \Leftrightarrow H$) là công thức.
- Nếu G là một công thức và x là biến thì các biểu thức ($\forall x\ G$), ($\exists x\ G$) là công thức.

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

Lượng tử phổ dụng cho phép mô tả tính chất của cả một lớp các đối tượng, chứ không phải của một đối tượng, mà không cần phải liệt kê ra tất cả

các đối tượng trong lớp. Chẳng hạn sử dụng vị từ Elephant(x) (đối tượng x là con voi) và vị từ Color(x, Gray) (đối tượng x có mầu xám) thì câu "tất cả các con voi đều có mầu xám"có thể biểu diễn bởi công thức $\forall x$ (Elephant(x) $\Rightarrow$ Color(x, Gray)).

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

Một công thức là công thức phân tử hoặc phủ định của công thức phân tử được gọi là ***literal***. Chẳng hạn, Play(x, Football), $\neg$ Like(Lan, Rose) là các literal. Một công thức là tuyển của các literal sẽ được gọi là ***câu tuyển***. Chẳng hạn, Male(x) $\vee$ $\neg$ Like(x, Foodball) là câu tuyển.

Trong công thức $\forall x$ G, hoặc $\exists x$ G trong đó G là một công thức nào đó, thì mỗi xuất hiện của biến x trong công thức G được gọi là ***xuất hiện buộc***. Một công thức mà tất cả các biến đều là xuất hiện buộc thì được gọi là ***công thức đóng.***

Ví dụ: Công thức $\forall x$ P(x, f(a, x)) $\wedge$ $\exists y$ Q(y) là công thức đóng, còn công thức $\forall x$ P(x, f(y, x)) không phải là công thức đóng, vì sự xuất hiện của biến y trong công thức này không chịu ràng buộc bởi một lượng tử nào cả (Sự xuất hiện của y gọi là ***sự xuất hiện tự do***).

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

6.1.2. Ngữ nghĩa.

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

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

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

Father(x) gắn với hàm: ứng với mỗi x là cha của nó, thì hạng thức Father(An) sẽ chỉ người cha của An.

NGỮ NGHĨA CỦA CÁC CÂU ĐƠN

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

NGỮ NGHĨA CỦA CÁC CÂU PHỨC HỢP

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

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

Câu Like(Lan, Rose) $\vee$ Like(An, Tulip) là đúng nếu câu Like(Lan, Rose) là đúng hoặc câu Like(An, Tulip) là đúng.

NGỮ NGHĨA CỦA CÁC CÂU CHỨA CÁC LƯỢNG TỬ

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

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

Ngữ nghĩa của công thức $\exists$x G được xác định như là ngữ nghĩa của công thức là tuyển của tất cả các công thức nhận được từ G bằng cách thay x

bởi một đối tượng trong miền đối tượng. Chẳng hạn, nếu ngữ nghĩa của câu Younger(x,20) là "x trẻ hơn 20 tuổi "và miền đối tượng gồm ba người {Lan, An, Hoa} thì ngữ nghĩa của câu ∃x Yourger(x,20) là ngữ nghĩa của câu Yourger(Lan,20) ∨ Yourger(An,20) ∨ Yourger(Hoa,20). Câu này nhận giá trị True nếu và chỉ nếu ít nhất một trong ba người Lan, An, Hoa trẻ hơn 20 tuổi.

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

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

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

CÁC CÔNG THỨC TƯƠNG ĐƯƠNG

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

Chúng ta có các tương đương sau đây:

1. $\forall x\, G(x) \equiv \forall y\, G(y)$

   $\exists x\, G(x) \equiv \exists y\, G(y)$

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

2. $\neg(\forall x\, G(x)) \equiv \exists x\, (\neg G(x))$

   $\neg(\exists x\, G(x)) \equiv \forall x\, (\neg G(x))$

3. $\forall x\, (G(x) \wedge H(x)) \equiv \forall x\, G(x) \wedge \forall x\, H(x)$

$\exists x\ (G(x) \vee H(x)) \equiv \exists x\ G(x) \vee \exists x\ H(x)$

Ví dụ: $\forall x$ Love(x, Husband(x)) $\equiv \forall y$ Love(y, Husband(y)).

6.2. CHUẨN HOÁ CÁC CÔNG THỨC

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

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

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

1. Loại bỏ các kéo theo

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

2. Chuyển các phủ định tới các phân tử

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

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

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

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

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

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

3. Loại bỏ các lượng tử tồn tại

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

Một cách tổng quát, giả sử $\exists y$ (G) là một công thức con của công thức đang xét và nằm trong miền tác dụng của các lượng tử $\forall x_1, ..., \forall x_n$. Khi đó

ta có thể xem y là hàm của n biến x_1,.....,x_n, chẳng hạn f(x_1.....x_n). Sau đó ta thay các xuất hiện của y trong công thức G bởi hạng thức f(x_1...x_n) và loại bỏ các lượng tử tồn tại. Các hàm f được đưa vào để loại bỏ các lượng tử tồn tại được gọi là ***hàm Skolem*.**

Ví dụ: xét công thức sau:

$$\forall x\ (\exists y\ (P(x,y) \vee \forall u\ (\exists \mathbf{v}\ (Q(a, \mathbf{v}) \wedge \exists y\ \neg R(x,y)))$$

(1)

Công thức con $\exists y\ P(x,y)$ nằm trong miền tác dụng của lượng tử $\forall x$, ta xem y là hàm của x: f(x). Các công thức con $\exists \mathbf{v}\ (Q(a, \mathbf{v}))$ và $\exists y \neg R(x,y)$ nằm trong miền tác dụng của các lượng tử $\forall x$, $\forall u$ ta xem **v** là hàm g(x,u) và y là hàm h(x,u) của hai biến x,u. Thay các xuất hiện của y và $\vee$ bởi các hàm tương ứng, sau đó loại bỏ các lượng tử tồn tại, từ công thức (1) ta nhận được công thức:

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

(2)

4. Loại bỏ các lượng tử phổ dụng

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

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

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

5. Chuyển các tuyển tới các literal

Bước này được thực hiện bằng cách thay các công thức dạng: $P \vee (Q \wedge R)$ bởi $(P \vee Q) \wedge (P \vee R)$ và thay $(P \wedge Q) \vee R$ bởi $(P \vee Q) \wedge (P \vee R)$. Sau bước này công thức trở thành hội của các câu tuyển nghĩa là ta nhận được các công thức ở dạng chuẩn tắc hội.

Chẳng hạn, câu (3) được chuyển thành công thức sau

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

6. Loại bỏ các hội

Một câu hội là đúng nếu và chỉ nếu tất cả các thành phần của nó đều đúng. Do đó công thức ở dạng chuẩn tắc hội tương đương với tập các thành phần.

Chẳng hạn, câu (4) tương đương với tập hai câu tuyển sau

$P(f(x)) \vee (Q(a,g(x,u))$

$P(f(x)) \vee \neg R(x,h(x,u))$ (5)

7. Đặt tên lại các biến

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

$P(f(x)) \vee (Q(a,g(x,u))$

$P(f(x)) \vee \neg R(x,h(x,u))$ (5')

Như vậy, khi tri thức là một tập hợp nào đó các công thức trong logic vị từ, bằng cách áp dụng thủ tục trên ta nhận được cơ sở tri thức chỉ gồm các câu tuyển (tức là ta luôn luôn có thể xem mỗi câu trong cơ sở tri thức là tuyển của các literal). Hoàn toàn tương tự như trong logic mệnh đề, mỗi câu tuyển có thể biểu diễn dưới dạng một kéo theo, vế trái của các kéo theo là hội của các câu phân tử, còn vế phải là tuyển của các câu phân tử. Dạng câu này được gọi là câu Kowlski, một trường hợp quan trọng của câu Kowlski là câu Horn (luật ***if - then***).

6.3. CÁC LUẬT SUY DIỄN

Trong chương 5 chúng ta đã đưa ra các luật suy diễn quan trọng trong logic mệnh đề: luật Modus Ponens, luật Modus Tolens, luật bắc cầu,... luật phân giải. Chúng ta đã chỉ ra rằng (mục 5.5), luật phân giải là luật đầy đủ cho bác bỏ. Điều đó có nghĩa là, bằng phương pháp chứng minh bác bỏ, chỉ sử dụng luật phân giải ta có thể chứng minh được một công thức có là hệ quả logic của một tập các công thức cho trước hay không. Kết quả quan trọng này sẽ được mở rộng sang lôgic vị từ.

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

- **Luật thay thế phổ dụng:**

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

$$\frac{\forall xG}{G[x/t]}$$

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

- **Hợp nhất**

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

Friend(x,Ba) ⇒ Good(x) (Mọi bạn của Ba đều là người tốt)

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

Ta có thể hợp nhất hai câu Friend(x,Ba) ⇒ Good(x) và Friend(Lan,y) bởi phép thay thế [x/Lan,y/Ba]. áp dụng luật thay thế phổ dụng với phép thay thế này ta nhận được hai câu:

Friend(Lan,Ba) ⇒ Good(Lan)

Friend(Lan,Ba)

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

Một cách tổng quát, một phép thế θ là một dãy các cặp x_i/t_i, $\theta = [x_1/t_1\ x_2/t_2 \ldots\ x_n/t_n]$ trong đó các x_i là các biến khác nhau, các t_i là các hạng thức và các x_i không có mặt trong t_i (i=1,...,n). Áp dụng phép thế θ vào công thức G, ta nhận được công thức G_θ, đó là công thức nhận được từ công thức G bằng cách thay mỗi sự xuất hiện của các x_i bởi t_i. Chẳng hạn, nếu G = P(x,y,f(a,x)) và θ =[x/b,y/g(z)] thì G_θ =P(b,g(z),f(a,b)).

Với hai câu phân tử G và H mà tồn tại phép thế θ sao cho G_θ và H_θ trở thành đồng nhất (G_θ=H_θ) thì G và H được gọi là ***hợp nhất được,*** phép thế θ được gọi là ***hợp nhất tử*** của G và H. Chẳng hạn, hai câu Like(An,y)và Like(x,Football) là hợp nhất được bởi hợp nhất tử [x/An,y/Football]. Vấn đề đặt ra là, với hai câu phân tử bất kì G và H, chúng có hợp nhất được không và nếu có thì làm thế nào tìm được hợp nhất tử ? Vấn đề này sẽ được nghiên cứu trong mục sau. Còn bây giờ chúng ta đưa ra các luật suy diễn quan trọng nhất, trong đó có sử dụng phép hợp nhất.

- **Luật Modus Ponens tổng quát.**

Giả sử P_i,P_i' (i= 1,..,n) và Q là các công thức phân tử sao cho tất cả các cặp câu P_i,P_i' hợp nhất được bởi phép thế θ, tức là $P_{i\theta}$=$P_{i\theta}$‘ (i =1,..,n). Khi đó ta có luật:

$$\frac{(P_i \wedge \ldots \wedge P_n \Rightarrow Q), P_i', .., P_n'}{Q'}$$

Trong đó Q' =Q_θ.

Ví dụ: Giả sử ta có các câu (Student (x) ∧ Male (x) ⇒ Like (x,Football)) và Student(Anh), Male(Anh). Với phép thế θ = [x|Anh], các cặp câu Student(x),Student(Anh) và Male(x), Male(Anh) hợp nhất được.Do đó ta suy ra câu Like(Anh,Football).

Luật phân giải tổng quát

- **Luật phân giải trên các câu tuyển**

Giả sử ta có hai câu tuyển $A_1 \vee \ldots \vee A_m \vee C$ và $B_1 \vee \ldots \vee B_n \vee \neg D$, trong đó A_i (i =1,..,m) và B_j (j=1,..,n) là các literal, còn C và D là các câu phân tử có thể hợp nhất được bởi phép thế θ, $C_\theta\theta$=D_θ. Khi đó ta có luật:

$$\frac{A_1 \vee \ldots \vee A_m \vee C, B_1 \vee \ldots \vee B_n \vee \neg D}{A_1' \vee \ldots \vee A_m' \vee B_1' \vee \ldots \vee B_n'}$$

Trong đó $A_i'=A_i\theta$ (i=1,..,m) và $B_j'=B_j\theta$ (j=1,..,n)

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

Ví dụ: Giả sử ta có hai câu A=Hear(x,Music)$\vee$ Play(x,Tennis) và B=$\neg$Play(An,y) $\vee$ Study (An). Hai câu Play(x,Tennis) và Play(An,y) hợp nhất được bởi phép thế θ=[x|An,y|Tennis]. Do đó từ hai câu đã cho, ta suy ra câu Hear(An,Music) $\vee$ Study (An). Trong ví dụ này, hai câu A=Hear(x,Music) $\vee$ Play(x,Tennis) và B=$\neg$Play(An,y) $\vee$ Study (An) là phân giải được và phân giải thức của chúng là Hear(An,Music)$\vee$ Study(An).

- **Luật phân giải trên các câu Horn:**

Câu Horn (luật If-Then) là các câu có dạng

$$P_1\wedge\ldots\wedge P_m \Rightarrow Q$$

trong đó P_i(i =1,...,m; m $\geq$ 0) và Q là các câu phần tử.

Giả sử ta có hai câu Horn $P_1\wedge\ldots\wedge P_m \wedge S \Rightarrow Q$ và $R_1\wedge\ldots\wedge R_n \Rightarrow T$, trong đó hai câu S và T hợp nhất được bởi phép thế θ, $S_\theta=T_\theta$. Khi đó ta có luật:

$P_1 \wedge\ldots\wedge P_m \wedge S \Rightarrow Q$,
$R_1 \wedge\ldots\wedge R_n \Rightarrow T$

$P_1'\wedge\ldots\wedge P_m'\wedge R_1'\wedge\ldots\wedge R_n \Rightarrow Q$

trong đó $P_i'=P_i\theta$ (i=1,..,m), $R_j'=R_j\theta$ (j=1,..,n), $Q'=Q\theta$.

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

$P_1\wedge\ldots\wedge P_m \wedge S \Rightarrow Q$,
T

$P_1'\wedge\ldots\wedge P_m' \Rightarrow Q'$

trong đó $P_i' = P_i\theta$ (i = 1,...,m) và $Q' = Q\theta$.

Ví dụ: Xét hai câu Student(x) $\wedge$ Male(x) $\Rightarrow$ Play(x,Football) và Male(Ba). Hai câu Male(Ba) và Male(x) hợp nhất được với phép thế [x|Ba], do đó từ hai câu trên ta suy ra Student (Ba) $\Rightarrow$ Play (Ba, Football).

6.4. THUẬT TOÁN HỢP NHẤT

Về mặt cú pháp, hạng thức và công thức phân tử có cấu trúc giống nhau, do đó ta gọi chung các hạng thức và các công thức phân tử là các ***biểu thức đơn.***

Trong mục này chúng ta sẽ trình bày thuật toán xác định hai biểu thức đơn cho trước có hợp nhất được không, và nếu được thuật toán sẽ cho ra hợp nhất tử tổng quát nhất.

Nhớ lại rằng, một phép thế θ là một danh sách $[x_1|t_1, x_2|t_2, \ldots x_n|t_n]$, trong đó các x_i là các biến khác nhau, t_i là các hạng thức không chứa x_i (i=1,..,n). Kết quả áp dụng phép thế θ vào biểu thức E là biểu thức Eθ nhận được từ E bằng cách thay mỗi xuất hiện của biến x_i bởi t_i. Hai biểu thức được xem là hợp nhất được nếu tồn tại phép thế θ để chúng trở thành đồng nhất, và khi đó θ được gọi là hợp nhất tử của chúng. Chẳng hạn, xét hai biểu thức sau:

Know(An, x)

Know(y, husband(z))

Với phép thế θ = [y|An, x|husband(z)], cả biểu thức trên trở thành

Know(An, husband(z))

Tuy nhiên, nếu hai biểu thức hợp nhất được thì nói chung sẽ có vô số hợp nhất tử. Chẳng hạn, ngoài hợp nhất tử đã nêu, hai câu Know(An, x) và Know(y, husband(z)) còn có các hợp nhất tử sau

[y|An, x|husband(Hoa), z|Hoa]

[y|An, x|husband(Lan), z|Lan]

…

Chúng ta sẽ gọi hợp thành của hai phép thế θ và η là phép thế θη sao cho với mọi biểu thức E ta có E(θη) = (Eθ)η. Nói cụ thể hơn, hợp thành của phép thế $\theta=[x_1|t_1, \ldots, x_1|t]$ và $\eta = [y_1|s_1, \ldots, y_n|s_n]$ là phép thế $\theta\eta = [x_1|t_1\eta, \ldots, x_m|t_m\eta, y_1|s_1, \ldots, y_n|s_n]$ trong đó ta cần loại bỏ các cặp $y_i|s_i$ mà y_i trùng với một x_k nào đó.

Ví dụ: Xét hai phép thế:

$$\theta = [x|a, v|g(y,z)]$$

$$\eta = [x|b, y|c, z|f(x), v|h(a)]$$

khi đó hợp thành của chúng là phép thế

$$\theta\eta = [x|a, v|g(c,f(x)), y|c, z|f(x)]$$

Quan sát ví dụ trên ta thấy rằng phép thế $\theta\eta$ áp đặt nhiều hạn chế cho các biến hơn là θ. Do đó ta sẽ nói rằng phép thế θ ***tổng quát hơn*** phép thế $\theta\eta$.

Chẳng hạn, phép thế đã đưa ra ở trên:

[y|An, x|husband(z)]

tổng quát hơn phép thế:

[y|An, x|husband(Hoa), z|Hoa]

Giả sử E và F là hai biểu thức đơn hợp nhất được. Ta gọi ***hợp nhất tử tổng quát nhất*** của E và F là hợp nhất tử θ sao cho θ tổng quất hơn bất kỳ hợp nhất tử δ nào của E và F. Nói một cách khác,θ là hợp nhất tử tổng quát nhất của E và F nếu với mọi hợp nhất tử δ của E và F đều tồn tại phép thế η sao cho $\delta=\theta\eta$. Chẳng hạn, với E và F là các câu Know(An,x) và Know(y,husband(z)) thì hợp nhất tử tổng quát nhất là:

$$\theta=[y|An, x|husband(z)]$$

Sau đây chúng ta sẽ trình bày thuật toán hợp nhất, đó là thuật toán đệ quy, có 3 tham biến: hai biểu thức đơn E và F,và hợp nhất tử tổng quát nhất là θ. Thuật toán này sẽ dừng và cho ra hợp nhất tử tổng quát nhất θ nếu E và F hợp nhất được. Nếu E và F không hợp nhất được, thuật toán cũng sẽ dừng và cho thông báo về điều đó.

Ta sẽ giả sử rằng E và F chứa các biến có tên khác nhau, nếu không ta chỉ cần đặt tên lại các biến.

Trong thuật toán, ta cần tìm ***sự khác biệt*** giữa hai biểu thức. Sự khác biệt được xác định như sau: Đọc hai biểu thức đồng thời từ trái sang phải cho tới khi gặp hai ký hiệu khác nhau trong biểu thức. Trích ra hai biểu thức con bắt đầu từ các ký hiệu khác nhau đó. Cặp biểu thức con đó tạo thành sự khác biệt giữa hai biểu thức đã cho.

Ví dụ: Sự khác biệt giữa hai câu Like(x,f(a,g(z))) và Like(x,y) là cặp (f(a,g(z)),y) còn sự khác biệt giữa hai câu Know(x,f(a,u)) và Know(x,f(a,g(b)) là cặp (u,g(b)).

Thuật toán hợp nhất được mô tả bởi thủ tục đệ quy sau:

Procedure *Unify(E,F,θ);*

Begin

1.Xác định sự khác biệt giữa E và F;

2.if *không có sự khác biệt* **then**

{thông báo thành công;stop};

3.if *sự khác biệt là cặp (x,t),trong đó x là biến, t là hạng thức không chứa x* **then**

{

3.1 E←E[x|t]; F←F[x|t];

// tức là áp dụng phép thế {x|t} vào các biểu thức E và F

3.2 θ←θ [x/t]; // hợp thành của phép thế q và phép thế [x/t]

3.3 Unify(E,F, θ);

}

else

{thông báo thất bại; **stop}**

end;

Nhận xét: Thuật toán hợp nhất trên luôn luôn dừng sau một số hữu hạn bước vì cứ mỗi lần bước 3 được thực hiện thì số biến còn lại trong hai biểu thức sẽ bớt đi một, mà số biến trong hai biểu thức là hữu hạn. Để biết hai biểu thức P và Q có hợp nhất được hay không ta chỉ cần gọi thủ tục Unify(P,Q,q) trong đó q là phép thế rỗng. Bạn đọc dễ dàng chuyển thủ tục đệ quy sang thủ tục không đệ quy (bài tập).

Ví dụ: Xét hai biểu thức sau:

P(a, x, f(a,g(x,y)))

P(u.h(a), f(u,v)) (1)

Sự khác biệt giữa hai biểu thức này là (a,u). Thế u bởi a ta nhận được hai biểu thức sau:

P(a, x, f(a, g(x, y)))

P(a, h(a), f(a, v)) (2)

và phép thế:

q = [u|a].

Sự khác biệt giữa hai biểu thức (2) là (x, h(a)).Thay x bởi h(a), ta có hai biểu thức sau: P(a, h(a), f(a, g(h(a), y)))

P(a, h(a), f(a,v)) (3)

và phép thế: q = [u|a, x|h(a)]

Sự khác biệt giữa hai biểu thức (3) là (g(h(a), y), v).Thế v bởi g(h(a), y), hai biểu thức (3) trở thành đồng nhất:

P(a, h(a), f(a, g(h(a), y)))

Như vậy,hai câu (1) hợp nhất được với hợp nhất tử tổng quát nhất là:

q = [u|a, x|h(a), v|g(h(a), y)]

6.5. CHỨNG MINH BẰNG LUẬT PHÂN GIẢI

Giả sử chúng ta có một cơ sở tri thức (CSTT) gồm các câu trong logic vị từ cấp một.Chúng ta luôn luôn xem CSTT là thoả được, tức là có một minh hoạ mà tất cả các câu trong CSTT đều đúng. Chẳng hạn minh họa đó là thế giới hiện thực của vấn đề mà chúng ta đang quan tâm, và CSTT gồm các câu mô tả sự hiểu biết của chúng ta về thế giới hiện thực đó.

Không mất tính tổng quát, ta có thể xem các câu trong CSTT là các câu tuyển. Chúng ta có thể sử dụng luật phân giải để suy ra các câu mới là hệ quả logic của CSTT.

Ví dụ:

Giả sử CSTT gồm các câu tuyển sau:

ự P(w)Ú Q(w) (1)

P(x)Ú R(x) (2)

ự Q(y)Ú S(y) (3)

⎤R(z) Ú S(z) (4)

Sau đây chúng ta sẽ đưa ra một chứng minh của câu S(a) từ CSTT trên bằng luật phân giải. Áp dụng luật phân giải cho câu (2) và (4) với phép thế [x⏐a, x⏐a], ta suy ra câu sau:

⎤P(a) Ú S(a) (5)

Áp dụng luật phân giải cho câu (1) và (3) với phép thế [w|a,y|a] ta nhận được câu:

$$\neg P(a) \acute{U} S(a) \qquad (6)$$

Áp dụng luật phân giải cho câu (5) và (6), ta suy ra câu S(a) Ú S(a). Câu này tương đương với câu S(a).

Chứng minh bằng cách áp dụng các luật suy diễn để dẫn tới điều cần phải chứng minh (như chứng minh trên) được gọi là ***chứng minh diễn dịch*** (deduction proof). Nhưng cần biết rằng, luật phân giải không phải là luật đầy đủ cho diễn dịch (deduction_complete), tức là từ một tập các tiên đề, chỉ sử dụng luật phân giải chúng ta không thể sinh ra tất cả các câu là hệ quả logic của các tiên đề đã cho.

Tuy nhiên định lý phân giải (xem mục 5.5) vẫn còn đúng trong logic vị từ cấp một. Điều đó có nghĩa là, chỉ sử dụng luật phân giải ta có thể xác định được một tập câu trong logic vị từ cấp một là thoả được hay không thoả được. Nếu một tập câu là không thoả được thì qua một số bước áp dụng luật phân giải ta sẽ sinh ra một câu rỗng (tức là dẫn tới mâu thuẫn).

Để chứng minh câu H là hệ quả logic của tập các câu $\{G_1,G_2,..,..G_n\}$ (các tiên đề), ta có thể áp dụng phương pháp chứng minh bác bỏ, tức là chứng minh tập câu $\{G_1,G_2,..,..G_n,\neg H\}$ không thoả được. Mặt khác, ở trên ta đã chỉ ra rằng, luật phân giải cho phép ta xác định được một tập câu là thoả được hay không thoả được. Vì vậy luật phân giải được xem là luật đầy đủ cho bác bỏ (refutation -complete).

Ví dụ:

Từ CSTT gồm các câu (1)_(4) trong ví dụ trên, ta có thể chứng minh được câu S(a) bằng phương pháp bác bỏ như sau. Thêm vào CSTT câu:

$$\neg S(a) \qquad (7)$$

(lấy phủ định của câu cần chứng minh). áp dụng luật phân giải cho câu (3) và (7) với phép thế [y|a], ta suy ra câu:

$$\neg Q(a) \qquad (8)$$

Từ câu (1) và (8) với phép thế [w|a], ta nhận được câu:

$$\neg P(a) \qquad (9)$$

Từ câu (4) và (7) với phép thế [z|a], ta suy ra câu:

$\neg R(a)$ (10)

Từ câu (2) và (10) với phép thế [x|a] ta nhận được câu:

$P(a)$ (11)

Áp dụng luật phân giải cho câu (9) và (11) ta nhận được câu rỗng (mâu thuẫn: $\neg P(a)$ và $P(a)$).

Bây giờ chúng ta trình bày thủ tục tổng quát sử dụng luật phân giải để chứng minh một công thức H là hoặc không là hệ quả logic của một tập công thức $G=\{G_1,G_2,..,..G_n\}$: thủ tục chứng minh bằng luật phân giải.

Procedure *Proof_by_Resolution*

Input: tập G={G1,G2,..,..Gn} các công thức (các tiên đề);

H_Công thức cần chứng minh;

Begin

1. Biến đổi các công thức Gi (i=1,...,n) và $\neg H$ về dạng chuẩn hội;

2. Từ các dạng chuẩn hội nhận được ở bước 1, thành lập tập các câu tuyển C;

3. **Repeat**

3.1 Chọn ra hai câu A và B trong C;

3.2 If A và B phân giải được then tính phân giải thức Res(A,B);

3.2 If ResA,B) là câu mới then thêm Res(A,B) vào tập C;

Until Nhận được câu rỗng hoặc không có câu mới nào được sinh ra;

4. **If** *câu rỗng được sinh ra* **then** *thông báo H đúng*

else *thông báo H sai;*

End

Sau đây chúng ta làm sáng tỏ thêm một số bước trong thủ tục chứng minh bằng luật phân giải:

1. Để thực hiện bước 1, ta cần áp dụng thủ tục chuẩn hoá các câu đã được đưa ra trong mục 6.2.

2. Để thực hiện bước 3.2, ta cần áp dụng thủ tục hợp nhất cho các cặp câu phân tử (A_i,B_i), trong đó A_i và $\neg B_j$ (hoặc $\neg A_i$ và B_j) là các thành phần của các câu tuyển A và B tương ứng. Với mỗi cặp như thế hợp nhất được thì áp dụng luật phân giải trên các câu tuyển để tính phân giải thức Res(A,B).

3. Bước 3.1 là bước chưa được xác định rõ ràng. Có rất nhiều phương pháp chọn ra hai câu A và B từ tập câu C. Mục sau chúng ta sẽ trình bày các

chiến lược, nó cho phép ta lấy ra hai câu ở mỗi bước 3 sao cho vòng lặp ở bước 3 được thực hiên hiệu quả. Các chiến lược này được gọi là ***chiến lược phân giải.***

Trong nhiều ứng dụng, CSTT (tập câu G) chỉ gồm các câu Horn (các luật If_then). Trong các trường hợp đó, chúng ta sẽ có các thủ tục suy diễn hiệu quả hơn. Vấn đề này sẽ được nghiên cứu kỹ trong chương sau.

Một ví dụ chứng minh

Giả sử chúng ta biết các thông tin sau đây:

1) Ông Ba nuôi một con chó.
2) Hoặc ông Ba hoặc ông Am đã giết con mèo Bibi.

Chúng ta cũng biết rằng:

3) Mọi người nuôi chó đều yêu quý động vật.
4) Ai yêu quý động vật cũng không giết động vật.

Và đương nhiên là:

5) Chó mèo đều là động vật.

Để biểu diễn các tri thức trên trong logic vị từ cấp một, chúng ta cần sử dụng các hằng D, Ba, Am, Bibi, các vị từ Dog(x) (x là chó), Cat(y), (y là mèo), Rear(u,v) (u nuôi v), AnimalLover(u) (u là người yêu quý động vật), Kill(u,v) (u giết v), Animal(x) (x là động vật). Sử dụng các hằng và các vị từ trên, chúng ta có thể chuyển các câu 1_5 thành các câu trong logic vị từ cấp một như sau:

1. $Dog(D) \wedge Rear(Ba,D)$
2. $Cat(Bibi) \wedge (Kill(Ba,Bibi)$ Ú $Kill(Am,Bibi))$
3. $\forall x\ (\forall y(Dog(y) \wedge Rear(x,y)) \Rightarrow AnimalLover(x)))$
4. $\forall u\ (AnimalLover(u) \Rightarrow (\forall v\ (Animal(v) \Rightarrow \neg Kill(u,v))\ )$
5. $\forall z\ (Dog(z) \Rightarrow Animal(z)) \wedge \forall w(Cat(w) \Rightarrow Animal(w))$.

Chuẩn hoá các câu 3_5, chúng ta nhận được các câu sau:

3'. $\neg Dog(y) \vee \neg Rear(x,y) \vee AnimalLover(x)$

4'. $\neg AnimalLover(u) \vee \neg Animal(v) \vee \neg Kill(u,v)$

5'. $(\neg Dog(z) \vee Animal(z)) \wedge (\neg Cat(w) \vee Animal(w))$

Từ các câu ở dạng chuẩn hội 1, 2, 3', 4' và 5' chúng ta tạo ra cơ sở tri thức gồm các câu tuyển sau:

(1) Dog(D)

(2) Rear(Ba, D)

(3) Cat(Bibi)

(4) Kill(Ba,Bibi) ∨ Kill(Am, Bibi)

(5) ⌉Dog(y) ∨ ⌉Rear(x,y) ∨ AnimalLover(x)

(6) ⌉AnimalLover(u) ∨ ⌉Animal(v) ∨ ⌉Kill(u,v)

(7) ⌉Dog(z) ∨ Animal(z)

(8) ⌉Cat(w) ∨ Animal(w)

Bây giờ ta muốn hỏi cơ sở tri thức gồm các câu (1)_(8): Ai đã giết Bibi?

Điều đó có nghĩa là ta cần tìm các đối tượng ứng với biến t mà câu Kill(t,Bibi) là hệ quả logic của các câu(1)_(8).

Thêm vào các câu (1)_(8) câu:

(9) ⌉Kill (t,Ba)

Từ câu (4) và câu (9) với phép thế [t|Am], áp dụng luật phân giải ta nhận được câu:

(10) Kill (Ba,Bibi)

Từ (6) và (10) với phép thế [u|Ba,v|Bibi], ta nhận được câu:

(11) ⌉AnimalLover (Ba) ∨ ⌉Animal (Bibi)

Từ (3) và (8) với phép thế [w|Bibi], ta nhận được câu

(12) Animal (Bibi)

Từ (11) và (12) ta nhận được câu:

(13) ⌉AnimalLover (Ba)

Từ (1) và (5) với phép thế [y|D] ta nhận được câu:

(14) ⌉Rear (x,D) ∨ AnimalLover(x)

Từ câu (2) và (14) với phép thế [x|Ba] ta nhận được câu

(15) AnimalLover(Ba)

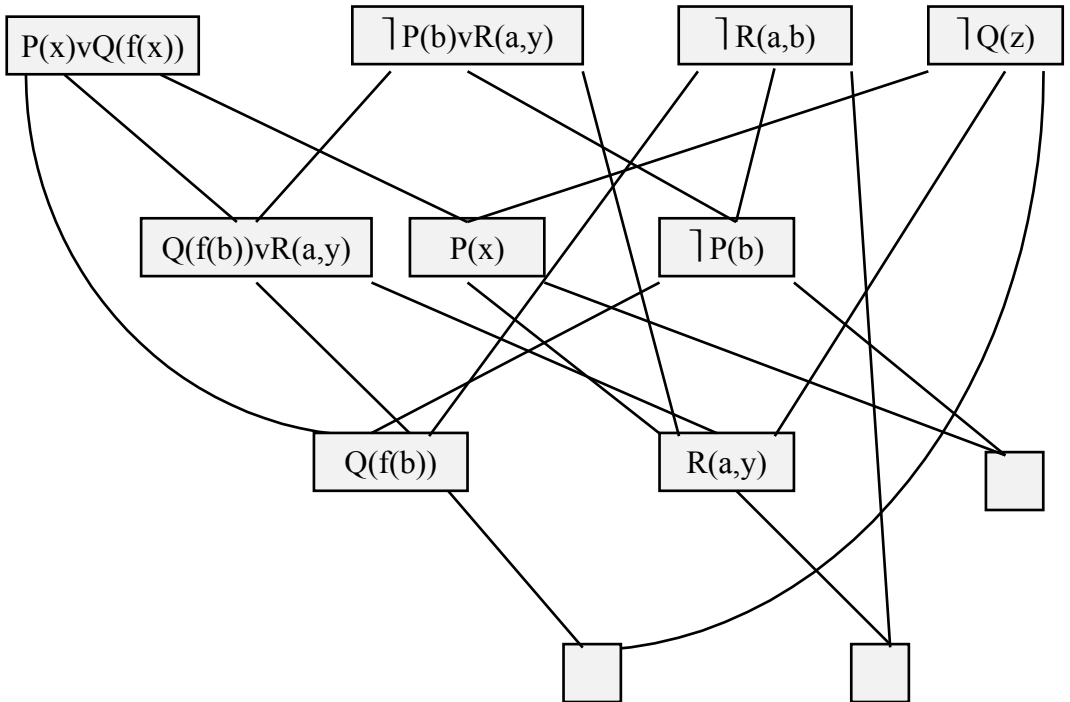
Từ câu (13) và (15) ta suy ra câu rỗng. Như vậy ông Am đã giết con mèo Bibi. Còn một khả năng nữa, câu (4) và (9) phân giải được với phép thế [t|Ba], song trường hợp này không dẫn tới câu rỗng.

6.6. CÁC CHIẾN LƯỢC PHÂN GIẢI

Thủ tục tổng quát chứng minh bằng luật phân giải trong mục 6.5 chứa bước 3.1 (chọn ra hai câu A và B để xét xem chúng có phân giải được hay không và nếu A và B phân giải được thì tính phân giải thức của chúng). Vấn đề đặt ra là, ta cần có chiến lược chọn các câu để phân giải chúng ở mỗi bước sao cho thủ tục chứng minh được thực hiện hiệu quả. Các chiến lược này được gọi là các ***chiến lược phân giải.***

Chúng ta có thể xem thủ tục chứng minh bằng luật phân giải như thủ tục tìm kiếm trên ***đồ thị phân giải***. Đỉnh của đồ thị này là các câu, các cung đi từ các câu cha tới câu là phân giải thức của các câu cha.

Ví dụ:



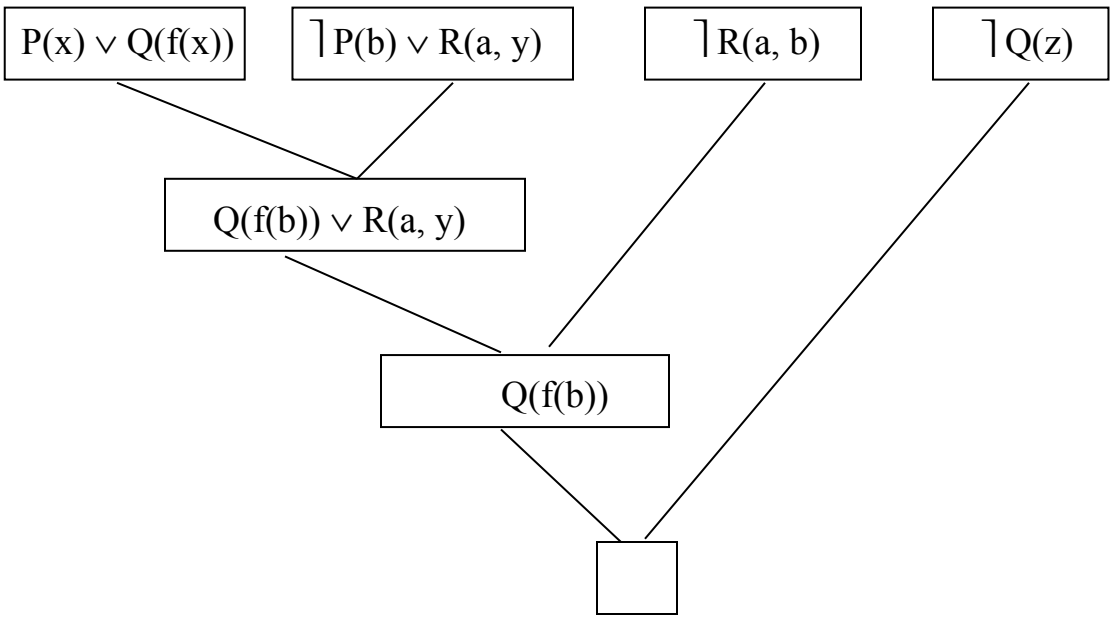
Hình 6.1 Đồ thị phân giải

Giả sử chúng ta có một tập hợp các câu

C= { P(x) ∨ Q(f(x)), ¬P(b) ∨ R(a,y), ¬R (a,b), ¬Q(z) }.

Đồ thị phân giải trên tập câu đó được biểu diễn trong hình 6.1 dưới đây, trong đó các cung đi từ trên xuống dưới.

Chúng ta đã biết rằng, từ một tập câu tuyển C nếu ta tìn được một dãy các phân giải thức dẫn tới câu rỗng [] thì ta đã chứng minh được tập không thoả mãn được. Dãy các phân giải thức kết thúc bởi câu rỗng tạo thành một cây nhị phân, gốc là câu rỗng. Cây này sẽ được gọi là ***cây chứng minh***. Chẳng hạn, hình 6.2 biểu diễn một cây chứng minh tập câu C trong ví dụ trên không thoả được. Đương nhiên là, trong một đồ thị phân giải có thể có nhiều cây chứng minh. Chẳng hạn, đồ thị phân giải trong hình 6.1 có 5 cây chứng minh, một trong các cây đó là cây trong hình 6.2



Hình6.2. Một cây chứng minh từ đồ thị phân giải trong hình 6.1

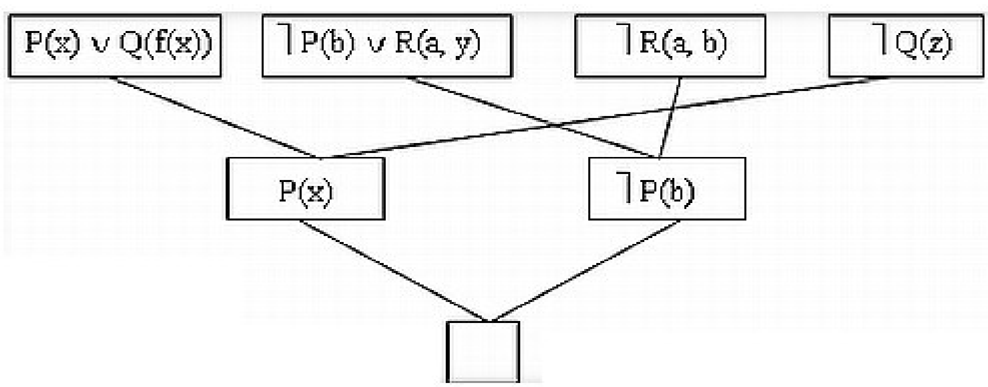
Các chiến lược phân giải hoặc là chiến lược sinh ra các phân giải thức theo một hệ thống nào đó, hoặc là chiến lược hạn chế việc sinh ra các phân giải thức nhằm nhanh chóng đạt tới câu rỗng. Một chiến lược được xem là

đầy đủ nếu nó đảm bảo sinh ra câu rỗng, nếu tập C các câu cho trước là không thoả được. Sau đây chúng ta sẽ trình bày một số chiến lược phân giải quan trọng.

6.6.1. Chiến lược phân giải theo bề rộng

Chúng ta sẽ phân chia các đỉnh của đồ thị phân giải thành các mức. Các câu thuộc tập câu C cho trước ở mức 0. Phân giải thức của các câu ở mức 0 sẽ ở mức một. Các câu ở mức i sẽ là các phân giải thức mà một trong các câu cha của nó ở mức i-1, còn cha kia ở mức ≤ i-1. Trong chiến lược phân giải theo bề rộng, các phân giải thức được sinh ra theo bề rộng, các phân giải thức ở mức i+1 chỉ được sinh ra khi tất cả các câu ở mức i đã được sinh ra. Chẳng hạn, đồ thị trong hình 6.1 gồm có bốn mức, trên cùng là mức 0, rồi đến các mức 1,2 và 3. Nếu chúng ta áp dụng chiến lược phân giải theo bề rộng, thi ta sẽ đạt tới câu rỗng ở trên mức 2, cây chứng minh được cho trong hình 6.3. Cây chứng minh tìm được bởi chiến thuật phân giải theo bề rộng sẽ là cây ngắn nhất.

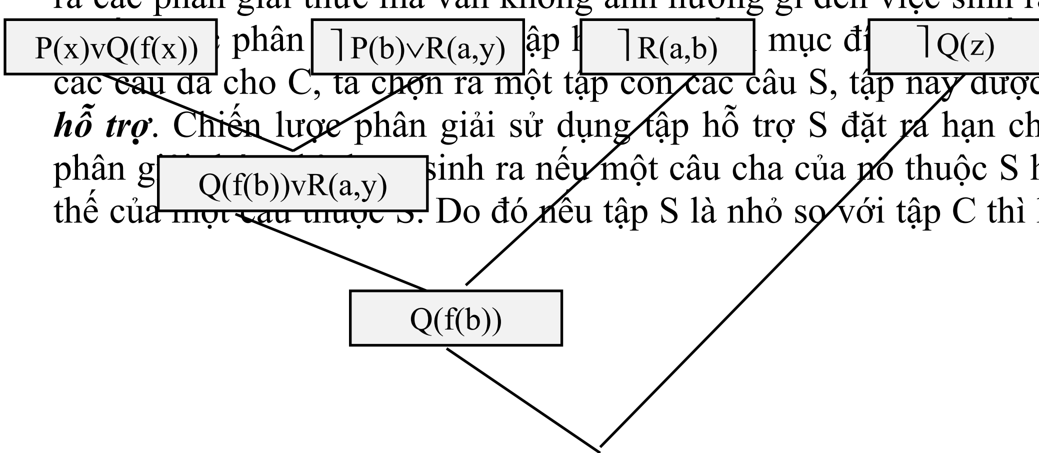
Chiến lược phân giải theo bề rộng là chiến lược đầy đủ.



Hình 6.3. Một cây chứng minh tìm được theo chiến lược bề rộng.

6.6.2. Chiến lược phân giải sử dụng tập hỗ trợ

Thủ tục chứng minh sẽ hiệu quả hơn nhiều nếu chúng ta hạn chế sinh ra các phân giải thức mà vẫn không ảnh hưởng gì đến việc sinh ra câu rỗng. [illegible] phân [illegible] ập [illegible] mục đí [illegible] tiên từ tập các câu đã cho C, ta chọn ra một tập con các câu S, tập này được gọi là ***tập hỗ trợ***. Chiến lược phân giải sử dụng tập hỗ trợ S đặt ra hạn chế sau: một phân g [illegible] sinh ra nếu một câu cha của nó thuộc S hoặc là hậu thế của một câu thuộc S. Do đó nếu tập S là nhỏ so với tập C thì không gian



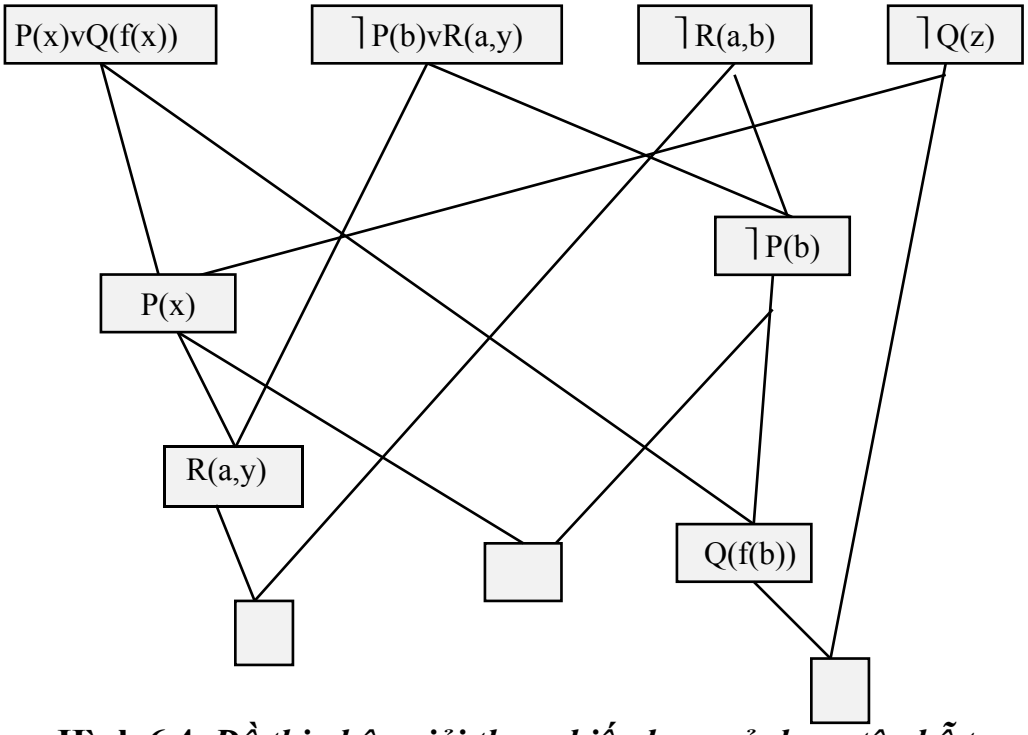
tìm kiếm sẽ được thu hẹp nhiều. Trật tự các câu được sinh ra có thể điều khiển bởi chiến lược bề rộng. Chẳng hạn, nếu C là tập các câu trong ví dụ đã nêu, và nếu ta chọn S ={¬ R(a,b), ¬Q(z)} làm tập hỗ trợ thì đồ thị phân giải của chiến lược sử dụng tập hỗ trợ S được cho trong hình 6.4.

Nếu tập hỗ trợ S được chọn sao cho tập các câu còn lại C - S thoả được thì chiến lược phân giải sử dụng tập hỗ trợ S là đầy đủ. Sau đây là một số phương pháp chọn tập hỗ trợ đảm bảo cho chiến lược phân giải đầy đủ.

- S gồm tất cả các câu thuộc C không chứa literal âm. Khi đó C-S sẽ gồm các câu chứa literal âm, và do đó C-S là thoả được, vì một minh hoạ mà tất cả các literal âm trong C-S nhận giá trị true là một mô hình của C-S.

- S gồm tất cả các câu thuộc C không chứa các literal dương. Tương tự như trên, tập các C-S là thoả được.

- Giả sử ta cần chứng minh công thức H từ tập tiên đề G. Khi đó ta có thể lấy tập hỗ trợ S gồm tất cả các câu tuyển từ dạng chuẩn của ¬H. Tập các câu tuyển từ tập tiên đề G, đương nhiên đã được giả thiết là thoả được.



Hình 6.4. ***Đồ thị phân giải theo chiến lược sử dụng tập hỗ trợ***

6.6.3. Chiến lược tuyến tính

Giả sử C là tập câu mà ta cần chứng minh là không thoả được. Chọn một câu C_0 thuộc C, C_0 được gọi là câu trung tâm xuất phát. Các phân giải thức là hậu thế của C_0 cũng được gọi là câu trung tâm. Chiến lược tuyến tính đặt ra hạn chế sau: chỉ sinh ra các phân giải thức mà một trong hai câu cha là câu trung tâm, đồng thời hai câu trung tâm chỉ được phân giải cùng nhau khi một câu là hậu thế của một câu khác. Tính chất "tuyến tính"thể hiện ở chỗ, hai câu trung tâm không được phép phân giải cùng nhau trừ khi một câu là hậu thế của câu kia.

Chiến lược phân giải tuyến tính sẽ đầy đủ, nếu câu trung tâm xuất phát C_0 được chọn từ tập C sao cho các câu còn lại thoả được. Trong thực tế, nếu câu cần chính minh H là hội của các literal, thì người ta lấy $C_0 = \rceil H$.

6.7. SỬ DỤNG LOGIC VỊ TỪ CẤP MỘT ĐỂ BIỂU DIỄN TRI THỨC

Logic vị từ cấp một cho phép chúng ta biểu diễn các đối tượng trong thế giới hiện thực với các tính chất của chúng và các mối quan hệ giữa chúng. Để biểu diễn tri thức của chúng ta về một miền đối tượng nào đó trong logic vị từ cấp một, trước hết chúng ta cần đưa ra các kí hiệu: các kí hiệu hằng (hằng đối tượng) để chỉ ra các đối tượng cụ thể; các kí hiệu biến để chỉ các đối tượng bất kỳ trong miền đối tượng; các kí hiệu hàm để biểu diễn các quan hệ hàm; các kí hiệu vị từ để biểu diễn các mối quan hệ khác nhau giữa các đối tượng. Các kí hiệu được đưa ra đó tạo thành ***hệ thống từ vựng*** về miền đối tượng mà chúng ta đang quan tâm. Sử dụng các từ vựng đã đưa ra, chúng ta sẽ tạo ra các câu trong logic vị từ cấp một để biểu diễn tri thức của chúng ta về miền đối tượng đó. Tập hợp tất cả các câu được tạo thành sẽ lập nên cơ sở tri thức trong hệ tri thức mà chúng ta mong muốn xây dựng. Vấn đề xây dựng cơ sở tri thức sẽ được đề cập đến trong mục sau.

Sử dụng các kí hiệu hằng, các kí hiệu biến và các kí hiệu hàm, chúng ta sẽ tạo ra các hạng thức (term) để biểu diễn các đối tượng (xem mục 6.1). Tuy nhiên trong rất nhiều vấn đề, để thuận lợi cho biểu diễn, chúng ta cần đến một dạng hạng thức đặc biệt, đó là danh sách. Danh sách là một cấu trúc dữ liệu được sử dụng thường xuyên nhất trong các ngôn ngữ Prolog và Lisp. Trong mục này chúng ta sẽ trình bày sự tạo thành các hạng thức danh sách và các phép toán trên danh sách. Song trước hết chúng ta cần xác định vị từ bằng, một vị từ thường xuyên được sử dụng.

6.7.1. Vị từ bằng

Vị từ bằng, kí hiệu truyền thống là =, biểu diễn mối quan hệ đồng nhất. Giả sử T_1 và T_2 là hai hạng thức bất kỳ, khi đó công thức phân tử $T_1 = T_2$ là đúng trong một minh hoạ nếu trong minh hoạ đó T_1 và T_2 ứng với cùng một đối tượng. Chẳng hạn, câu: Father (An) = Ba là đúng trong một minh hoạ, nếu người ứng với Father(An) và người ứng với Ba là một.

Sau đây là một ví dụ đơn giản về sử dụng vị từ bằng nhau. Giả sử vị từ Sister(x,y) có nghĩa là "x là chị gái của y", khi đó câu "Tam có ít nhất hai chị gái"có thể biểu diễn bởi công thức:

$\exists$ x,y (Sister(x,Tam) ^ Sister(y,Tam) ^ $\neg$(x=y))

Sau này chúng ta thường viết $T_1 \neq T_2$ thay cho $\neg(T_1=T_2)$.

6.7.2 Danh sách và các phép toán trên danh sách

Để mô tả vấn đề, trong rất nhiều trường hợp chúng ta cần sử dụng các danh sách. Danh sách là cấu trúc dữ liệu được sử dụng rộng rãi nhất trong các ngôn ngữ xử lý các thông tin không phải là số.

Danh sách là một dãy gồm n ($n \geq 0$) đối tượng bất kỳ, ở đây đối tượng có thể là đối tượng đơn hoặc đối tượng có cấu trúc, và do đó danh sách cũng là một dạng đối tượng. Chúng ta sẽ biểu diễn danh sách bởi cặp dấu ngoặc vuông, bên trong liệt kê các thành phần của danh sách, các thành phần ngăn cách nhau bởi dấu phẩy.

Sau đây là một số ví dụ về danh sách:

[ ] (danh sách rỗng)

[spring, summer, autumn, winter]

[john, data(23, may, 1964), 8354268]

[a, [a,c], b, [a,d,e]]

Một danh sách không phải là danh sách rỗng có thể phân tách làm hai phần: thanh phần đầu tiên của danh sách được gọi là ***đầu danh sách,*** phần còn lại của danh sách được gọi là ***đuôi danh sách***. Chẳng hạn trong danh sách:

[blue, red, white, yellow]

đầu của danh sách là blue, và đuôi của danh sách là danh sách [red, white, yellow].

Chúng ta có thể biểu diễn danh sách bởi cách viết:

[đầu_danh_sách |đuôi_danh_sách]

Chẳng hạn:

[blue, red, white, yellow]=[blue|[red, white, yellow]]

Chúng ta cũng có thể biểu diễn danh sách bằng cách liệt kê ra một số thành phần ở đầu danh sách, theo sau là dấu gạch đứng "|", rồi đến danh sách các thành phần còn lại. Chẳng hạn, sau đây là một số cách viết danh sách trên:

[blue, red, white, yellow]

=[blue|[red, white, yellow]]

=[blue, red|[white, yellow]]

=[blue, red, white|[yellow]]

=[blue, red, white, yellow|[ ]]

Chúng ta có thể biểu diễn danh sách bởi các hạng thức trong logic vị từ cấp một. Trong logic vị từ, một danh sách được định nghĩa như sau:

Danh sách hoặc là kí hiệu hằng [ ], hoặc là một hạng thức có dạng list(X,Y), trong đó list là kí hiệu hàm của hai biến, đối số X là một hạng thức bất kì và được gọi là đầu danh sách, đối số Y là một danh sách và được gọi là đuôi danh sách. (Trong ngôn ngữ Prolog, người ta sử dụng kí hiệu "." thay cho kí hiệu list; tức là hạng thức list(X,Y) trong Prolog được viết là.(X,Y).

Như vậy các cặp biểu diễn sau là tương đương:

Biểu diễn hạng thức	Biểu diễn dấu ngoặc vuông
list(X,Y)	[X|Y]
list(X,list(Y,Z))	[X, Y|Z]
list(X,list(Y,list(Z, [ ])))	[X,Y,Z]

Trong ngôn ngữ Prolog, ta có thể sử dụng cả hai dạng biểu diễn danh sách.

Các phép toán cơ bản trên danh sách:

1. Quan hệ thành phần

Quan hệ "đối tượng X là thành phần của danh sách L"được biểu diễn bởi vị từ:

Member(X,L)

Quan hệ này được xác định như sau:

X là thành phần của danh sách L nếu:

(1) hoặc X là đầu danh sách L

(2) hoặc X là thành phần của đuôi danh sách L

Tức là, vị từ Member(X,L) được xác định bởi công thức:

Member(X,L) $\Leftarrow$(L = [X | L1])$\vee$ (L = [Y | L2]$\wedge$ Member (X,L2))

Chẳng hạn,

Member(a,[a,b,c])

Member([b,c],[a,[b,c],d])

là các quan hệ đúng, còn

Member(b,[a,[b,c],d])

là sai.

2.Kết nối hai danh sách thành một danh sách

Phép toán kết nối 2 danh sách L_1 và L_2 thành một danh sách L được biểu diễn bởi vị từ

Conc(L_1,L_2,L)

Quan hệ này được xác định như sau:

Danh sách L là kết nối của 2 danh sách L_1 và L_2 nếu:

(1) hoặc L_1=[ ] và L= L_2

(2) hoặc $L_1 \neq$[ ] và đầu của L là đầu của L_1 và đuôi của L là kết nối của đuôi L_1 và L_2.

Tức là, quan hệ Conc(L_1,L_2,L) được xác định bởi công thức:

Conc(L_1,L_2,L) $\Leftarrow$ (L_1=[] $\wedge$ L=L_2) $\vee$ (L_1=[X|L_3] $\wedge$ L= [X|L_4] $\wedge$ Conc (L_3,L_2,L_4)

Chẳng hạn,

Conc([a,b],[1,2,3], [a,b,1,2,3])

Conc([a,[b,c]],[a,[ ],d],[a,[b,c],a,[ ],d)

là các quan hệ đúng, nhưng:

Conc([a,b],[c,d],[a,b,a,c,d])

là sai.

3.Loại bỏ một thành phần khỏi danh sách

Việc loại bỏ một thành phần X khỏi danh sách L được xác định bởi quan hệ:

Delete(X,L,L1)

Quan hệ này được xác định như sau:

Danh sách L_1 là kết quả loại thành phần X khỏi danh sách L, nếu

(1) hoặc X là đầu của L và L_1 là đuôi của L

(2) hoặc đầu của L_1 là đầu của L và đuôi của L_1 là kết quả việc loại X khỏi đuôi của L.

Tức là,

Delete $(X,L,L_1) \Leftarrow (L =[X \mid L_1]) \vee (L = [Y \mid L_2] \wedge$ Delete $(X,L_2,L_3) \wedge L_3 =[Y,L_3])$

Chẳng hạn, ta có quan hệ đúng sau:

Delete(a,[a,b,a,c],[b,a,c])

Delete(a,[a,b,a,c],[a,b,c])

4.Quan hệ danh sách con

Quan hệ Sublist(L_1,L) là đúng nếu danh sách L_1 là danh sách con của danh sách L, chẳng hạn:

Sublist([c,d,e],[a,b,c,d,e,f])

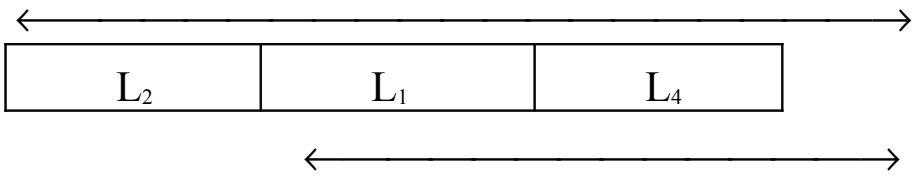
là quan hệ đúng còn

Sublist([b,d],[a,b,c,d,e,f])

là sai.

Đương nhiên, nếu L_1 là danh sách con của L, thì danh sách L có thể phân tách thành các danh sách con như trong hình sau:

L



Do đó ta có thể xác định quan hệ này như sau:

L_1 là danh sách con của danh sách L, nếu

(1) L là kết nối của L_2 và L_3, và

(2) L_3 là kết nối của L_1 và L_4

tức là, ta có: Sublist(L_1,L) <= Conc(L_2,L_3,L) ^ Conc(L_1,L_4,L_3)

Trên đây là một số phép toán cơ bản trên danh sách, để dễ dàng cho xử lý danh sách, ta cần xác định một số phép toán khác.

Tập hợp là một khái niệm cơ bản trong toán học, ta có thể mô tả rất nhiều vấn đề bằng cách sử dụng tập hợp và các phép toán trên tập hợp.Tuy nhiên ta có thể biểu diễn tập hợp bởi danh sách,và các phép toán tập hợp có thể được xác định thông qua các phép toán trên danh sách.

6.8. XÂY DỰNG CƠ SỞ TRI THỨC

Như chúng ta đã biết một hệ tri thức bao gồm hai thành phần chính là cơ sở tri thức (CSTT)và thủ tục suy diễn. Như vậy để thiết kế một hệ tri thức, chúng ta cần phải xây dựng nên CSTT. CSTT bao gồm các câu (trong một ngôn ngữ biểu diễn tri thức nào đó, ở đây là logic vị từ cấp một). Các câu này biểu diễn tri thức của chúng ta về một lĩnh vực áp dụng mà chúng ta đang quan tâm. Logic vị từ cấp một là một công cụ mạnh để biểu diễn tri thức và lập luận. Song một vấn đề đặt ra là, ta phải lựa chọn các đối tượng nào, các sự kiện nào, các quan hệ nào, các tri thức chung nào để đưa vào CSTT, sao cho với CSTT đó, thủ tục suy diễn có thể đưa ra các câu trả lời cho các câu hỏi của người sử dụng.

Quá trình xây dựng CSTT được gọi là ***công nghệ tri thức*** (knowledge engineering).Người kỹ sư tri thức (người kỹ sư làm việc trong lĩnh vực công nghệ tri thức) có thể nắm được các công nghệ làm ra CSTT, nhưng nói chung anh ta không hiểu biết về lĩnh vực áp dụng. Người kỹ sư tri thức cần phải phỏng vấn các chuyên gia trong lĩnh vực đó để khai thác các tri thức

cần thiết đưa vào CSTT,quá trình này được gọi là ***quá trình thu thập tri thức*** (knowledge acquisition).Chỉ nắm được cú pháp và ngữ nghĩa của ngôn ngữ biểu diễn tri thức, chúng ta không thể xây dựng được CSTT. Người kỹ sư cần phải hiểu biết các kỹ thuật của công nghệ tri thức. Trong mục này chúng ta sẽ trình bày vắn tắt các kỹ thuật cơ bản của công nghệ tri thức.

Phương pháp luận xây dựng một CSTT bao gồm các bước chính sau đây:

- Trước hết cần phải xác định CSTT mà ta xây dựng nói tới các loại đối tượng nào, các sự kiện nào, các thuộc tính nào, các quan hệ nào. Muốn vậy người kỹ sư tri thức cần phải tìm hiểu các chuyên gia trong lĩnh vực áp dụng để có sự hiểu biết đầy đủ về lĩnh vực đó. Cần nhớ rằng, chỉ khi nào đã hiểu biết tường tận về lĩnh vực áp dụng mới bắt đầu xây dựng CSTT.

- **Xây dựng hệ thống từ vựng.**

Hệ thống từ vựng bao gồm các hằng, các hàm và các vị từ. Bước này thực hiện việc chuyển dịch các khái niệm trong lĩnh vực áp dụng thành các tên hằng, tên hàm, tên vị từ. Các tên hằng, tên hàm, tên vị từ cần được lựa chọn sao cho người đọc CSTT có thể hiểu được dễ dàng ý nghĩa của nó. Chẳng hạn, ta có thể dùng vị từ HasColor (x,y) để biểu diễn quan hệ "đối tượng x có màu y", dùng vị từ Small(x) để biểu diễn thuộc tính "đối tượng x có cỡ nhỏ". Cùng một quan hệ, ta có thể biểu diễn bởi hàm hoặc vị từ, chẳng hạn, ta có thể sử dụng hàm HusbandOf(x) để biểu diễn đối tượng là chồng của x, hoặc có thể sử dụng vị từ IsHusband(y,x) để biểu diễn quan hệ " y là chồng của x". Một ví dụ khác, nếu chúng ta quan tâm tới các đối tượng với các màu khác nhau, ta có thể sử dụng vi từ HasColor(x,y) đã đưa ra ở trên. Song nếu chỉ quan tâm tới các đối tượng với màu đỏ hay không, ta có thể dùng vị từ Red(x) (x có màu đỏ).

Như vậy, trong giai đoạn xây dựng hệ thống từ vựng, ta cần quyết định biểu diễn một quan hệ bởi hàm hay vị từ, các hàm (vị từ) là các hàm (vị từ) của các đối số nào. Ta cần chọn các tên hằng, tên hàm, tên vị từ sao cho nó nói lên được nội dung mà nó cần mô tả.

- **Biểu diễn tri thức chung về lĩnh vực:**

Hệ thống từ vựng chỉ là danh sách các thuật ngữ. Chúng ta cần phải sử dụng các thuật ngữ này để viết ra các công thức logic mô tả các tri thức chung của chúng ta về lĩnh vực áp dụng, và cũng là để chính xác hoá các

thuật ngữ mà chúng ta đưa ra trong hệ thống từ vựng. Chẳng hạn, khi nói tới các quan hệ họ hàng, ta cần đưa vào các vị từ Sibling(x,y) biểu diễn quan hệ "x và y là anh em", vị từ Parent (u,v) biểu diễn "u là cha mẹ của v",... Sau đó ta cần đưa vào tiên đề sau đây:

$$\forall x,y(Sibling(x,y) \Leftrightarrow (x \neq y) \wedge \exists p(Parent(p,y) \wedge Parent(p,x)))$$

Câu này nói rằng, nếu x và y là anh em thì họ phải cùng cha mẹ và ngược lại.

Một số lượng lớn các tri thức của con người được mô tả bởi các câu trong ngôn ngữ tự nhiên. Do đó để xây dựng CSTT chúng ta cần biết chuyển các câu trong ngôn ngữ tự nhiên thành các câu trong ngôn ngữ vị từ. Sau đây là một vài ví dụ cho ta biết cách chuyển các câu trong ngôn ngữ tự nhiên thành công thức logic (chúng tôi khuyên độc giả hãy tự mình viết ra các công thức logic trước khi đọc tiếp). Sử dụng các vị từ Mushroom(x)(x là nấm), Purple(y) (y màu tím), Poisonnous(z) (z có độc),chúng ta hãy chuyển các câu sau đây thành các công thức logic.

Mọi cây nấm tím đều có độc

$$\forall x(Mushroom(x) \wedge Purple(x) \Rightarrow Poisonnous(x))$$

Tất cả cây nấm hoặc có màu tím hoặc có độc

$$\forall x(Mushroom(x) \Rightarrow Purple(x) \vee Poisonnous(x))$$

Mọi cây nấm hoặc màu tím hoặc có độc nhưng không là cả hai

$$\forall x(Mushroom(x) \Rightarrow (Purple(x) \wedge \neg Poisonnous(x)) \vee (\neg Purple(x) \wedge Poisonnous(x)))$$

Tất cả các cây nấm tím đều có độc trừ một cây

$$\exists x(Mushroom(x) \wedge Purple(x) \wedge \neg Poisonnous(x)) \wedge \forall y(Mushroom(y) \wedge Purple(y) \wedge (y \neq x) \Rightarrow Poisonnous(y)))$$

Chỉ có hai cây nấm tím

$$\exists x,y(Mushroom(x) \wedge Purple(x) \wedge Mushroom(y) \wedge Purple(y) \wedge (x \neq y) \wedge$$

$$\forall z,x(Mushroom(z) \wedge Purple(z) \Rightarrow (z=x) \vee (z=y)))$$

Sau khi chúng ta đã viết ra các công thức logic (các tiên đề) mô tả các tri thức chung về lĩnh vực áp dụng, có thể xem như chúng ta đã xây dựng nên một CSTT.Để thuận lợi cho việc lưu trữ CSTT trong máy tính, và thuận lợi cho thủ tục suy diễn hoạt động, chúng ta có thể chuẩn hoá các câu trong CSTT.

Trong toán học, người ta cố gắng tìm được một tập các tiên đề độc lập, tức là trong đó không có tiên đề nào có thể suy ra từ các tiên đề còn lại. Tuy nhiên, trong các hệ tri thức, CSTT có thể chứa các tiên đề "thừa", chúng không làm tăng thêm các tri thức mới được suy ra, song chúng có thể làm cho quá trình suy diễn hiệu quả hơn.

6.9. CÀI ĐẶT CƠ SỞ TRI THỨC

Trong mục trước, chúng ta đã xét một số kĩ thuật quan trọng nhất để xây dựng cơ sở tri thức(CSTT), một trong hai thành phần chính của hệ tri thức. CSTT là tập các câu (các tiên đề) mô tả tri thức của chúng ta về một lĩnh vực nào đó mà chúng ta quan tâm. Chúng ta sẽ giả thiết rằng, CSTT đã được chuẩn hoá, tức là nó gồm các câu tuyển (mỗi câu là tuyển của các literal). Thủ tục suy diễn, nói chung, là thủ tục chứng minh bác bỏ mà chúng ta đã trình bày trong mục **6.5.** Trong trường hợp CSTT là một tập hợp các câu Horn (các luật if-then), chúng ta có thể sử dụng thủ tục suy diễn tiến (forward chaining) hoặc thủ tục suy diễn lùi (forward chaining) sẽ được trình bày trong chương tiếp theo. Trong các thủ tục suy diễn, chúng ta đều phải thực hiện lặp lại công việc như sau: tìm ra các literal có thể hợp nhất được với một literal đang xét nào đó, nếu tìm được thì áp dụng luật phân giải cho hai câu chứa hai literal hợp nhất được đó. Do đó để cho thủ tục suy diễn thực hiện hiệu quả, chúng ta cần cài đặt CSTT sao cho công việc thực hiện lặp lại trên được thực hiện hiệu quả. Trước hết ta cần biểu diễn các hạng thức và các câu phân tử bởi các cấu trúc dữ liệu thích hợp.

6.9.1. Cài đặt các hạng thức và các câu phân tử:

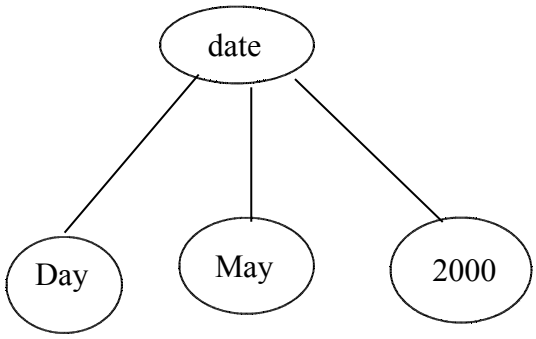
Về mặt cú pháp, các hạng thức và các câu phân tử có cấu trúc hoàn toàn giống nhau (xem lại mục 6.1.1). Do đó ta chỉ cần xét việc cài đặt các hạng thức. Các hạng thức biểu diễn các đối tượng trong thế giới hiện thực. Từ các đối tượng đơn được biểu diễn bởi các kí hiệu hằng, và từ các kí hiệu biến biểu diễn một đối tượng bất kì trong một lớp các đối tượng, ta có thể tạo ra các đối tượng có cấu trúc. Các đối tượng có cấu trúc là các đối tượng có một số thành phần, các thành phần này đến lượt mình lại có thể là các đối tượng có cấu trúc. Để kết hợp các thành phần tạo thành một đối tượng mới, chúng ta sử dụng một kí hiệu hàm (functor). Chẳng hạn, sử dụng kí hiệu hàm Date, ngày 1 tháng 5 năm 2000 được biểu diễn bởi hạng thức

date (1, May, 2000)

Một ngày bất kỳ trong tháng 5 năm 2000 được biểu diễn bởi hạng thức:

date (Day, May, 2000)

trong đó Day là kí hiệu biến. Hạng thức này gồm 3 thành phần, một là kí hiệu biến, hai thành phần khác là các kí hiệu hằng. Hạng thức này được biểu diễn bởi cấu trúc cây trong hình 6.5



Hình 6.5. ***Cây biểu diễn hạng thức date (Day, May,2000)***

Các ví dụ sau cho ta thấy cách tạo ra các hạng thức biểu diễn các đối tượng hình học trong mặt phẳng. Một điểm trong không gian hai chiều được xác định bởi hai toạ độ; một đoạn thẳng được xác định bởi hai điểm; một tam giác được xác định bởi ba điểm. Do đó, nếu ta sử dụng các kí hiệu hàm point (điểm), seg (đoạn thẳng), triangle (tam giác) thì điểm có toạ độ (2,3) được biểu diễn bởi hạng thức point (2, 3)

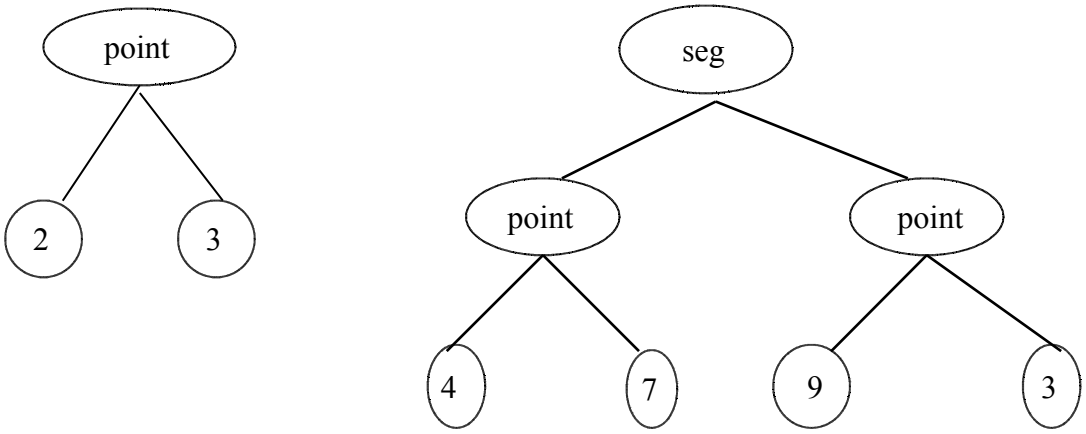
Đoạn thẳng nối hai điểm (4, 7) và (9, 3) được biểu diễn bởi hạng thức:

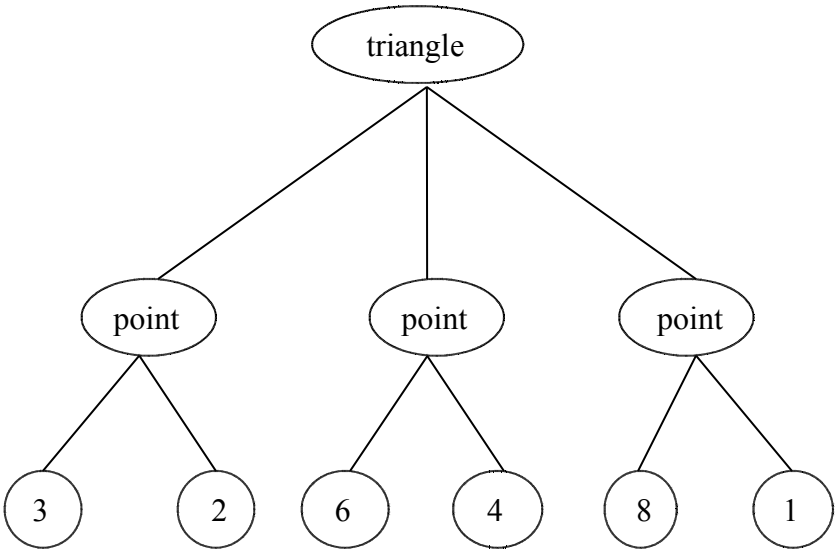
seg(point (4, 7), point (9, 3))

Tam giác có ba đỉnh là ba điểm (3, 2), (6, 4), (8, 1) được biểu diễn bởi hạng thức:

Triangle (point(3, 2), point(6, 4), point(8, 1))

Các hạng thức trên được biểu diễn bởi các cấu trúc cây trong hình 6.6.





Hình 6.6. ***Các cây biểu diễn các hạng thức.***

Một cách tổng quát, các hạng thức được biểu diễn bởi các cây (chẳng hạn, các cây trong **hình 6.6**), trong đó gốc của cây là các kí hiệu hàm. Nếu các đối số này không phải là kí hiệu hằng, hoặc kí hiệu biến thì chúng là các cây con của gốc được tạo thành theo quy tắc trên.

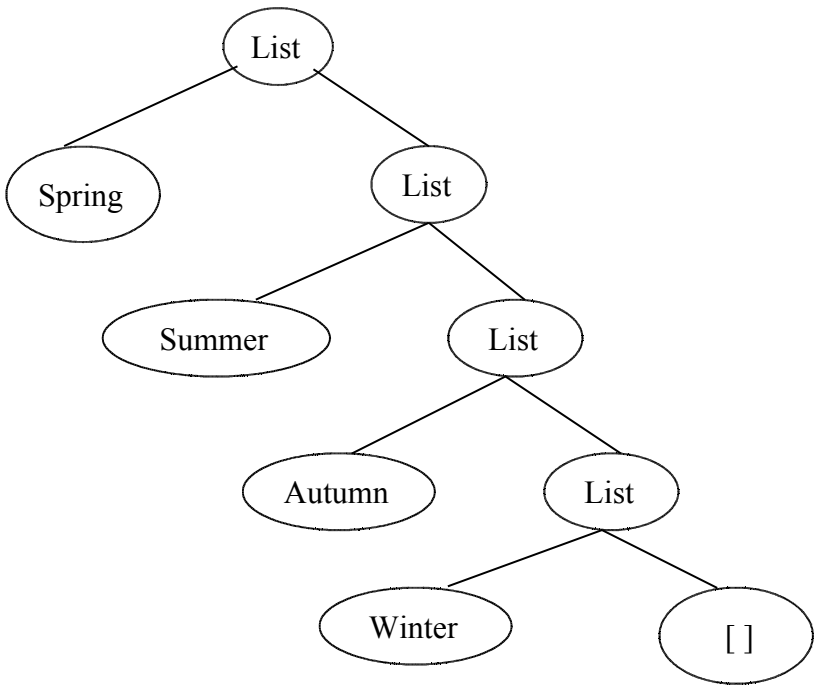
Bây giờ chúng ta xem xét các đối tượng danh sách (xem mục 6.7) được biểu diễn bởi cấu trúc cây như thế nào. Nhớ lại rằng, danh sách:

[spring, summer, autumn, winter]

được biểu diễn bởi hạng thức:

list (spring, list(summer, list (autumn, list (winter,[ ]))))

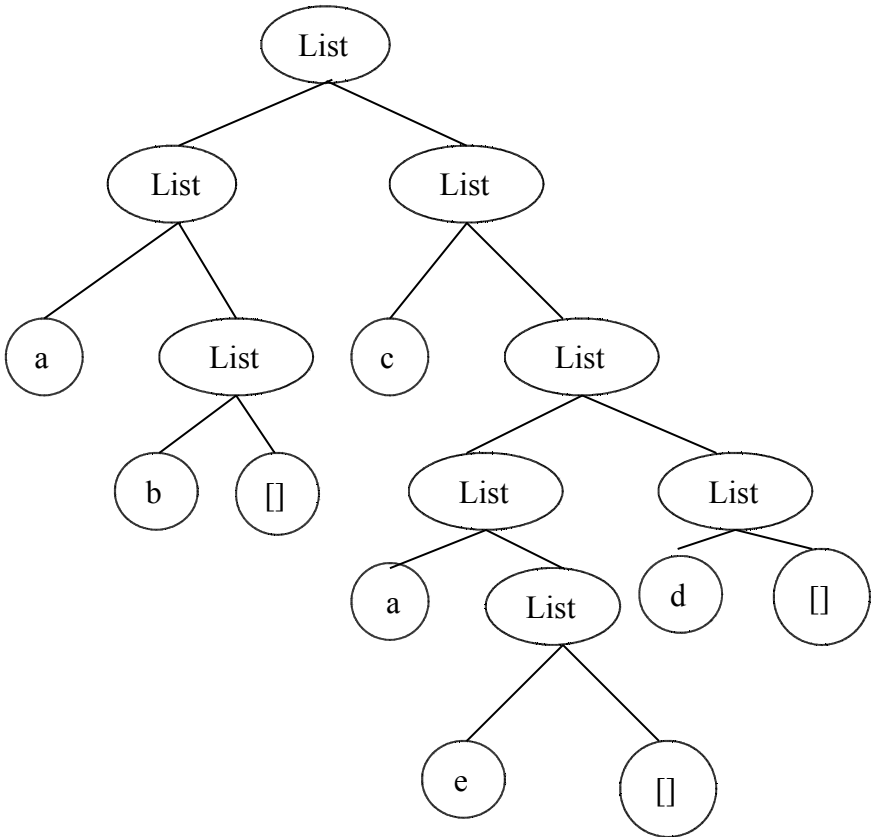
Hạng thức này được biểu diễn bởi cây trong **hình 6.7**



Hình 6.7. ***Cây biểu diễn danh sách [spring, summer, autumn, winter]***

Một cách tương tự, bằng cách chuyển sang dạng hạng thức, ta có thể biểu diễn danh sách: [a,b], c, [[a, e], d]] bởi cây trong **hình 6.8.** Trong cây **hình 6.8**, nếu a, b, c, d, e không phải là các đối tượng đơn mà là các đối tượng có cấu trúc được biểu diễn bởi các hạng thức, thì ở vị trí của các đỉnh gắn nhãn a, b, c, d, e sẽ là các cây con biểu diễn các hạng thức đó.

Trên đây chúng ta đã chỉ ra rằng, các hạng thức (và các câu phân tử) có thể biểu diễn một cách tự nhiên bởi các cấu trúc cây.



Hình 6.8. ***Cây biểu diễn danh sách [[a,b], c, [[a,e],d]]***

6.9.2. Cài đặt cơ sở tri thức

Chúng ta đã biết cách tạo ra các cấu trúc dữ liệu để biểu diễn các hạng thức và các câu phân tử. Bây giờ chúng ta nghiên cứu các kĩ thuật cài đặt cơ sở tri thức sao cho các thủ tục suy diễn có thể thực hiện có hiệu quả. Giả sử CSTT bao gồm các câu tuyển dạng

$$C = P_1 \vee \ldots \vee P_m \vee \neg Q_1 \vee \ldots \vee \neg Q_n$$

trong đó, Pi (i = 1,....,m, m ≥ 0), Q_k(k = 1,2,...,n, n ≥ 0) là các câu phân tử. Một cách tự nhiên, mỗi câu C có thể được biểu diễn bởi bản ghi gồm hai trường:

- Danh sách các Literal dương [P_1,....,P_m]
- Danh sách các Literal âm [Q_1,..., Q_n]

Một cách đơn giản nhất, ta có thể cài đặt CSTT như một danh sách các câu tuyển. Tuy nhiên với cách cài đặt này, thủ tục suy diễn kém hiệu quả, bởi mỗi lần cần xét xem một câu phân tử S có hợp nhất với một thành phần nào đó của một câu trong CSTT, ta phải đi qua danh sách, xem xét từng thành phần của các câu trong danh sách cho tới khi tìm ra hoặc đi tới hết danh sách.

Một giải pháp khác tốt hơn, ta có thể cài đặt CSTT bởi bảng băm. Các khoá cho bảng băm này là các kí hiệu vị từ, tức là bảng băm được định chỉ số theo các kí hiệu vị từ. Trong bảng băm, tại chỉ số ứng với mỗi kí hiệu vị từ ta sẽ lưu:

- Danh sách các literal dương của kí hiệu vị từ đó.
- Danh sách các literal âm.
- Danh sách các câu mà kí hiệu vị từ xuất hiện trong các literal dương của câu (Câu dương).
- Danh sách các câu mà kí hiệu vị từ xuất hiện trong các literal âm của câu (Câu âm).

Ví dụ: Giả sử CSTT chứa các câu sau:

Brother (An, Ba)

Brother (Tam, Hoa)

¬Brother (Lan, Cao)

¬Brother (x,y)∨ Male(x)

¬Brother (x,y)∨ ¬Male(y) ∨ ¬Brother (y,x)

Male(Cao)

Male(Ba)

¬ Male(Lan)

Khi đó, tại các chỉ số ứng với khoá Brother và Male, bảng băm sẽ lưu giữ các thành phần được cho trong bảng sau

Khoá	Literal dương	Literal âm	Câu dương	Câu âm
Brother	Brother(An,Ba) Brother(Tam, Hoa)	¬Brother (Lan, Cao)	¬Brother(x,y)∨ Male(y)∨Brother(y,x)	¬Brother (x,y) ∨ ¬Male(y) ∨Brother(y,x) ¬Brother (x,y)∨ Male(x)
Male	Male(Cao) Male(Ba)	¬ Male(Lan)	¬Brother(x,y)∨ Male(x)	¬Brother (x,y)∨ ¬Male(y) ∨ Brother (y,x)

Cài đặt CSTT bởi bảng băm định chỉ số theo các kí hiệu vị từ là phương pháp rất hiệu quả cho việc tìm kiếm trên CSTT, nếu như CSTT chứa nhiều kí hiệu vị từ và với mỗi kí hiệu vị từ chỉ có một số ít các câu chứa kí hiệu vị từ đó. Tuy nhiên trong một số áp dụng, có thể có rất nhiều câu chứa cùng một kí hiệu vị từ nào đó. Chẳng hạn, Cơ Sở Tri Thức có thể chứa hàng triệu câu nói về người lao động, mỗi người lao động được biểu diễn bởi các thông tin về họ tên, ngày tháng năm sinh, số thẻ bảo hiểm, công việc (Công việc được xác định bởi nơi làm việc và tiền lương). Tức là mỗi người lao động được mô tả bằng câu có dạng:

Worker (Tom, date (3, may,1965), 012-34-567, job(UNIMEX, 300))

(câu này nói rằng, có người lao động tên là Tom, sinh ngày 3 tháng 5 năm 1965, số thẻ bảo hiểm 012-34-567, làm việc tại công ty UNIMEX với mức lương 300). Trong các trường hợp như thế, để tìm kiếm có hiệu quả, ngoài việc xây dựng bảng băm định chỉ số theo các đối số của các kí hiệu vị từ, ta cần xây dựng các bảng băm định chỉ số theo các đối số của các vị từ. Chẳng hạn, ở đây các literal dương ứng với khoá Worker cần được tổ chức dưới dạng bảng băm định chỉ số theo khóa là số bảo hiểm y tế hoặc họ tên và ngày tháng năm sinh.

CHƯƠNG 7
BIỂU DIỄN TRI THỨC BỞI CÁC LUẬT VÀ LẬP LUẬN

Trong chương 6, chúng ta đã biết rằng, với một cơ sở tri thức gồm các câu trong logic vị từ cấp một, ta có thể chứng minh được một công thức có là hệ quả logic của cơ sở tri thức hay không, bằng phương pháp chứng minh bác bỏ và thủ tục giải. Tuy nhiên, thủ tục chứng minh tổng quát này có độ phức tạp lớn và đòi hỏi chiến lược giải thích hợp. Chính vì lý do này mà các nhà nghiên cứu cố gắng tìm các tập con của logic vị từ cấp một, sao cho chúng đủ khả năng biểu diễn cơ sở tri thức trong nhiều lĩnh vực áp dụng, và có thể đưa ra các thủ tục suy diễn hiệu quả. Các tập con này của logic vị từ cấp một sẽ xác định các ngôn ngữ biểu diễn tri thức đặc biệt. Trong chương này chúng ta sẽ nghiên cứu ngôn ngữ chỉ bao gồm các câu Horn (các luật *nếu - thì*). Chỉ sử dụng các luật *nếu - thì* chúng ta không thể biểu diễn được mọi điều mà chúng ta có thể biểu diễn được trong logic vị từ cấp một. Tuy nhiên với các luật *nếu - thì* ta có thể biểu diễn được một khối lượng lớn tri thức trong nhiều lĩnh vực áp dụng khác nhau, và có thể thực hiện các thủ tục suy diễn hiệu quả.

7.1. BIỂU DIỄN TRI THỨC BỞI CÁC LUẬT NẾU - THÌ

Ngôn ngữ bao gồm các luật *nếu - thì* (if - then), (còn gọi là các luật sản xuất -production rule), là ngôn ngữ phổ biến nhất để biểu diễn tri thức. Nhớ lại rằng, các câu Horn có dạng

$$P_1 \wedge \ldots \wedge P_n \Rightarrow Q$$

trong đó các P_i (i = 1, ..., n) và Q là các câu phần tử.

Các câu Horn còn được viết dưới dạng

Nếu

P_1 và P_2 ... và P_n

thì

Q

(*if P_1 and ... and P_n then Q*)

các P_i (i = 1, ..., n) được gọi là các điều kiện, Q được gọi là kết luận của luật.

Các luật *nếu - thì* có các ưu điểm sau đây

- Mỗi luật *nếu - thì* mô tả một phần nhỏ tương đối độc lập của tri thức.
- Có thể thêm và cơ sở tri thức các luật mới, hoặc loại bỏ một số luật cũ mà không ảnh hưởng nhiều tới các luật khác.
- Các hệ tri thức với cơ sở tri thức gồm các luật *nếu - thì* có khả năng đưa ra lời giải thích cho các quyết định của hệ.

Các luật *nếu - thì* là dạng biểu diễn tự nhiên của tri thức. Bằng cách sử dụng các luật *nếu - thì* chúng ta có thể biểu diễn được một số lượng lớn tri thức của con người về tự nhiên, về xã hội, kinh nghiệm của con người trong lao động, sản xuất, tri thức của các thầy thuốc, tri thức của các kỹ sư, tri thức trong các ngành khoa học: kinh tế, sinh học, hoá học, vật lý, toán học,...

Sau đây là một luật về chẩn đoán bệnh:

Nếu

1. bệnh nhân ho lâu ngày, và
2. bệnh nhân thường sốt vào buổi chiều

thì

bệnh nhân có khả năng bệnh lao

Một luật về kinh nghiệm dự báo thời tiết:

Nếu

chuồn chuồn bay thấp

thì

trời sẽ mưa

Nhiều định lý trong toán học có thể biểu diễn bởi các luật. Chẳng hạn,

Nếu

1. tam giác có một góc bằng 60^0, và
2. tam giác có hai cạnh bằng nhau

thì

tam giác đó là tam giác đều.

Các hệ tri thức mà cơ sở tri thức bao gồm các luật sẽ được gọi là các **hệ dựa trên luật** (*rule - based system*). Trong các mục còn lại của chương này chúng ta sẽ nghiên cứu các thủ tục suy diễn trong các hệ dựa trên luật.

7.2. LẬP LUẬN TIẾN VÀ LẬP LUẬN LÙI TRONG CÁC HỆ DỰA TRÊN LUẬT

Một khi chúng ta đã lưu trữ một cơ sở tri thức, chúng ta cần có thủ tục lập luận để rút ra các kết luận từ cơ sở tri thức. Trong các hệ dựa trên luật, có hai phương pháp luận lập luận cơ bản:

- Lập luận tiến, và
- Lập luận lùi

Chúng ta sẽ phân chia cơ sở tri thức thành hai bộ phận: **cơ sở luật** (rule base) và **cơ sở sự kiện** (fact base) (hoặc **bộ nhớ làm việc (working memory)**). Cơ sở luật bao gồm các luật có ít nhất một điều kiện, biểu diễn các tri thức chung về lĩnh vực áp dụng. Còn cơ sở sự kiện bao gồm các câu phần tử (các luật không điều kiện) mô tả các sự kiện mà chúng ta biết về các đối tượng trong lĩnh vực áp dụng.

7.2.1. Lập luận tiến

Tư tưởng cơ bản của lập luận tiến là áp dụng luật suy diễn Modus Ponens tổng quát (xem mục 6.3). Trong mỗi bước của thủ tục lập luận tiến, người ta xét một luật trong cơ sở luật. Đối sánh mỗi điều kiện của luật với các sự kiện trong cơ sở sự kiện, nếu tất cả các điều kiện của luật đều được thoả mãn thì sự kiện trong phần kết luận của luật được xem là sự kiện được suy ra. Nếu sự kiện này là sự kiện mới (không có trong bộ nhớ làm việc), thì nó được đặt vào bộ nhớ làm việc. Quá trình trên được lặp lại cho tới khi nào không có luật nào sinh ra các sự kiện mới.

Như vậy quá trình lập luận tiến là quá trình xem xét các luật. Với mỗi luật, ta đi từ phần điều kiện tới phần kết luận của luật, khi mà tất cả các điều kiện của luật đều được làm thoả mãn (bởi các sự kiện trong cơ sở sự kiện), thì ta suy ra sự kiện trong phần kết luận của luật. Chính vì lẽ đó mà có tên lập luận tiến (forward chaining hoặc forward reasoning).

Quá trình lập luận tiến không định hướng tới giải quyết một vấn đề nào cả, không định hướng tới tìm ra câu trả lời cho một câu hỏi nào cả. Lập luận tiến chỉ là quá trình suy ra các sự kiện mới từ các sự kiện trong bộ nhớ làm việc. Vì vậy lập luận tiến còn được gọi là **lập luận điều khiển bởi dữ liệu** (*data - driven reasioning*), hoặc **lập luận định hướng dữ liệu** (*data - directed reasioning*).

Để thấy được quá trình lập luận tiến diễn ra như thế nào, chúng ta xét ví dụ sau đây. (Ví dụ này là của P. H. Winston xem [ ]).

Giả sử cơ sở luật (cơ sở luật về các động vật trong sở thú) gồm các luật sau

Luật 1: *nếu* động vật có lông mao

thì động vật là loài có vú

Luật 2: *nếu* động vật có lông vũ

thì động vật là chim

Luật 3: *nếu* 1. động vật biết bay, và
2. động vật đẻ trứng
thì động vật là chim

Luật 4: *nếu* 1. động vật là loài có vú, và
2. động vật ăn thịt
thì động vật là thú ăn thịt

Luật 5: *nếu* 1. động vật là loài có vú, và
2. động vật có răng nhọn, và
3. động vật có móng vuốt
thì động vật là thú ăn thịt

Luật 6: *nếu* 1. động vật là thú ăn thịt, và
2. động vật có màu lông vàng hung, và
3. động vật có đốm sẫm
thì động vật là báo Châu Phi

Luật 7: *nếu* 1. động vật là thú ăn thịt, và
2. động vật có màu lông vàng hung, và
3. động vật có vằn đen
thì động vật là hổ

Luật 8: *nếu* 1. động vật là chim, và
2. động vật không biết bay, và
3. động vật có chân dài, và
4. động vật có cổ dài
thì động vật là đà điểu

Luật 9: *nếu* 1. động vật là chim, và
2. động vật không biết bay, và

3. động vật biết bơi, và

4. động vật có lông đen và trắng

thì động vật là chim cánh cụt

Giả sử một em bé quan sát một con vật có tên là Ki trong sở thú, em thấy nó có các đặc điểm sau

Ki có lông mao

Ki ăn thịt

Ki có màu lông vàng hung

Ki có đốm sẫm

Lúc này cơ sở sự kiện sẽ bao gồm các sự kiện trên.

Thủ tục lập luận tiến xem xét luật 1. Khi biến "động vật" trong luật này được thay bởi Ki, điều kiện của luật trở thành "Ki có lông mao", đây là một sự kiện có trong bộ nhớ làm việc, do đó ta suy ra "Ki là loài có vú". Đây là sự kiện mới, do đó nó được thêm vào bộ nhớ làm việc. Xét luật 4, thế biến "động vật" bởi Ki, thì hai điều kiện của luật trở thành

Ki là loài có vú, và

Ki ăn thịt

Cả hai sự kiện này đều có trong bộ nhớ làm việc, do đó từ luật 4 ta suy ra "Ki là thú ăn thịt". Sự kiện mới này lại được thêm vào bộ nhớ làm việc. Ta xét tiếp luật 6, thế biến "động vật" bởi Ki, các điều kiện của luật trở thành

Ki là loài thú ăn thịt, và

Ki có màu lông vàng hung, và

Ki có đốm sẫm

Tất cả các điều kiện này đều đúng, do đó từ luật 6, ta suy ra "Ki là báo Châu Phi". Như vậy từ các sự kiện đã biết về Ki, lập luận tiến đã suy ra các sự kiện mới sau

Ki là loài có vú.

Ki là thú ăn thịt.

Ki là báo Châu Phi.

7.2.2. Lập luận lùi

Trong các hệ dựa trên luật, chúng ta còn có thể sử dụng phương pháp **lập luận lùi** (*backward chaining* hoặc *backward reasoning*).

Trong lập luận lùi, người ta đưa ra các giả thuyết cần được đánh giá. Sử dụng lập luận lùi, giả thuyết đưa ra hoặc là được chứng minh, hoặc là bị bác bỏ (bởi các sự kiện trong bộ nhớ làm việc). Cần lưu ý rằng, chúng ta nói giả thuyết được chứng minh, hoặc bị bác bỏ là muốn nói tới nó được chứng minh, hoặc bác bỏ bởi tình trạng hiện thời của bộ nhớ làm việc. Khi mà bộ nhớ làm việc thay đổi (chúng ta thêm vào hoặc loại bỏ một số sự kiện) thì một giả thuyết đã được chứng minh có thể trở thành bị bác bỏ và ngược lại.

Quá trình lập luận lùi diễn ra như sau: Ta đối sánh giả thuyết đưa ra với các sự kiện trong bộ nhớ làm việc. Nếu có một sự kiện khớp với giả thuyết, (ở đây "khớp" được hiểu là hai câu mô tả sự kiện và giả thuyết trùng nhau qua một phép thế nào đó), thì ta xem như giả thuyết là đúng. Nếu không có sự kiện nào khớp với giả thuyết, thì ta đối sánh giả thuyết với phần kết luận của các luật. Với mỗi luật mà kết luận của luật khớp với giả thuyết, ta đi lùi lại phần điều kiện của luật. Các điều kiện này của luật được xem như các giả thuyết mới. Với giả thuyết mới, ta lặp lại quá trình trên.

Nếu tất cả các giả thuyết được sinh ra trong quá trình phát triển các giả thuyết bởi các luật được chọn thích hợp đều được thoả mãn (đều có trong bộ nhớ làm việc) thì giả thuyết đã đưa ra được xem là đúng. Ngược lại, dù ta áp dụng luật nào để phát triển các giả thuyết cũng dẫn tới các giả thuyết không có trong bộ nhớ làm việc và không thể quy giả thuyết này về các giả thuyết mới khác, thì giả thuyết đã đưa ra được xem là sai.

Để làm sáng tỏ tư tưởng của lập luận lùi, ta xét ví dụ sau. Ta sử dụng cơ sở luật đã đưa ra trong mục 7.2.1. Giả sử bộ nhớ làm việc chứa các sự kiện sau.

Bibi có lông vũ

Bibi có chân dài

Bibi có cổ dài

Bibi không biết bay

Ta đưa ra giả thuyết sau đây

Bibi là đà điểu

Đối sánh giả thuyết này với phần kết luận của các luật, ta thấy nó khớp với kết luận của luật 8 nếu thế biến "động vật" bởi Bibi. Từ luật 8, ta suy ra rằng, giả thuyết "Bibi là đà điểu" là đúng, nếu các điều kiện sau là đúng

1. Bibi là chim
2. Bibi không biết bay
3. Bibi có chân dài
4. Bibi có cổ dài

Đây là 4 giả thuyết mới, và việc đánh giá giả thuyết "Bibi là đà điểu" được quy về việc đánh giá bốn giả thuyết mới này. Các giả thuyết 2, 3 và 4 đều có trong bộ nhớ làm việc, ta chỉ cần đánh giá giả thuyết "Bibi là chim". Lại đối sánh giả thuyết này với phần kết luận của các luật. Ta thấy nó khớp với kết luận của luật 2 và luật 3. Xét luật 3, đi lùi lại phần điều kiện của luật này, ta nhận được các giả thuyết mới là

Bibi biết bay

Bibi đẻ trứng

Cả hai giả thuyết này đều không có trong bộ nhớ làm việc và cũng không khớp với phần kết luận của luật nào cả. Do đó, ta không thể phát triển

tiếp các giả thuyết này được nữa. Chuyển sang xét luật 2, để "Bibi là chim" luật này đòi hỏi điều kiện "Bibi có lông vũ". Điều kiện này có trong bộ nhớ làm việc. Vậy giả thuyết đã đưa ra "Bibi là đà điểu" là đúng.

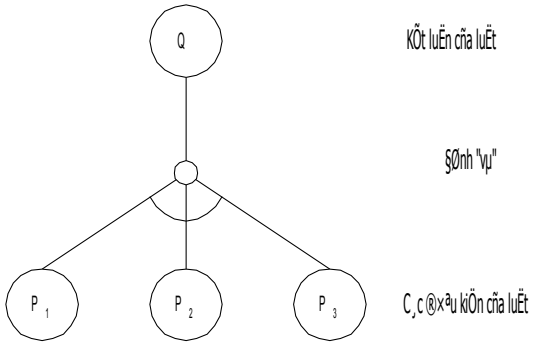
Lập luận lùi nhằm chứng minh một giả thuyết, chính vì thế mà lập luận lùi còn được gọi là **lập luận định hướng mục đích** (*goal - ariented reasoning*). Sau này chúng ta sẽ thấy rằng, chúng ta có thể sử dụng lập luận lùi để tìm ra các câu trả lời cho các câu hỏi của người sử dụng.

7.2.3. Lập luận lùi như tìm kiếm trên đồ thì và/hoặc

Mỗi luật **nếu - thì** dạng

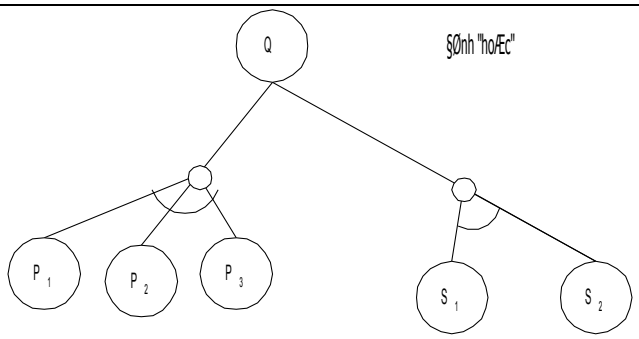
nếu P_1 và P_2 ... và P_m **thì** Q

có thể biểu diễn bởi đồ thị và/hoặc trong hình 7.1.



Hình 7.1 Biểu diễn đồ thị của luật

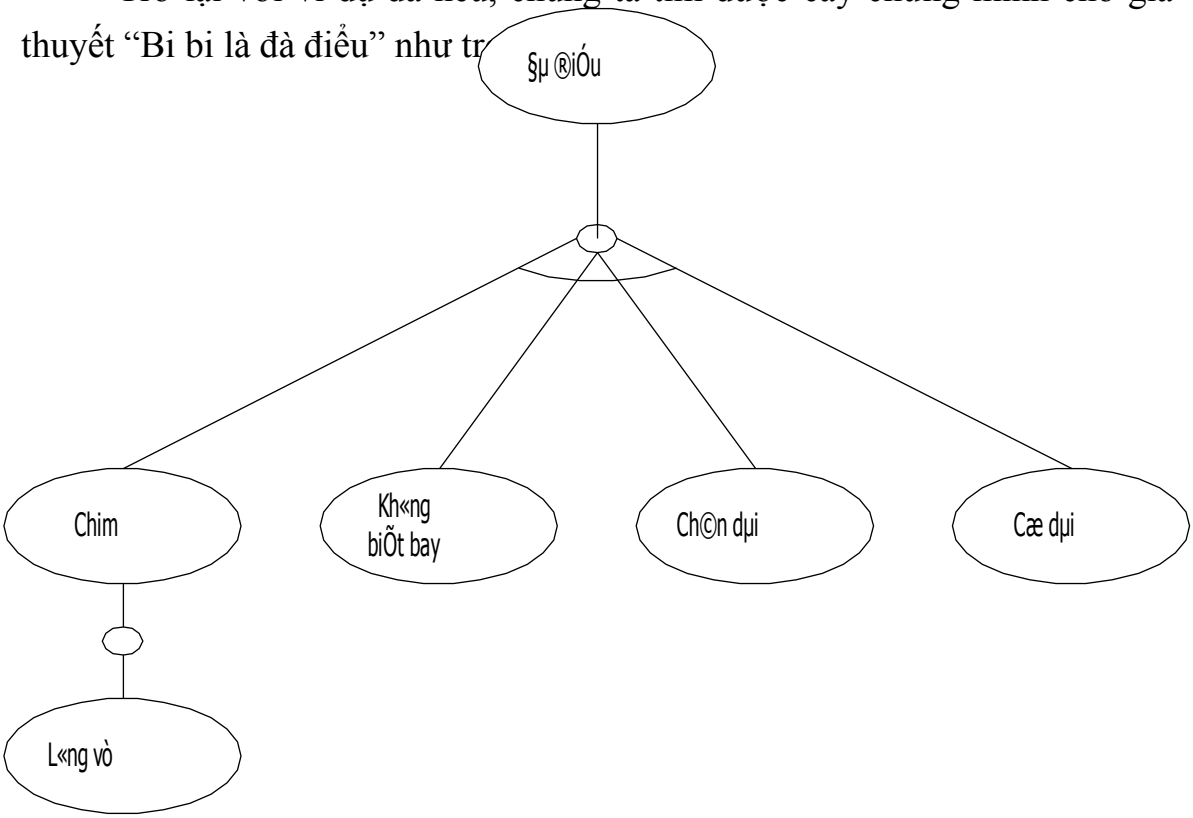
Có thể xảy ra, nhiều luật khác nhau có cùng phần kết luận. Chẳng hạn, có hai luật cho kết luận Q, một luật gồm ba điều kiện P_1, P_2, P_3, một luật gồm hai điều kiện S_1, S_2. Hoàn cảnh này được biểu diễn bởi đồ thị trong hình 7.2.



Bằng cách biểu diễn các luật bởi đồ thị như trên, từ cơ sở luật, ta xây dựng nên đồ thị và/hoặc. Khi đó lập luận lùi có thể xem như quá trình tìm kiếm trên đồ thị và/hoặc được xây dựng nên từ cơ sở luật. Quá trình tìm kiếm xuất phát từ đỉnh khớp với giả thuyết cần đánh giá.

Việc tìm kiếm để xác định một giả thuyết là đúng hay sai hoàn toàn tương tự như việc tìm kiếm trên đồ thị và/hoặc (xem mục 1.4.3) để xác định một đỉnh ứng với bài toán đã cho là giải được hay không giải được. Nếu giả thuyết được chứng minh là đúng thì chúng ta sẽ tìm được **cây chứng minh** giống như tìm cây nghiệm cho bài toán cần giải (xem mục 1.4.3).

Trở lại với ví dụ đã nêu, chúng ta tìm được cây chứng minh cho giả thuyết "Bi bi là đà điểu" như tr



H×nh 7.3 C©y chøng minh "Bibi lµ ®µ ®iÓu"

Từ cây chứng minh, hệ có thể đưa ra các câu trả lời cho các câu hỏi về các lý do mà hệ đã đưa ra các kết luận. Chẳng hạn, nếu người sử dụng hỏi "tại sao hệ khẳng định Bibi là đà điểu", hệ có thể trả lời rằng "vì Bibi là chim, Bibi không biết bay, Bibi có chân dài, và Bibi có cổ dài". Nếu người sử dụng hỏi tiếp "tại sao Bibi là chim ?", hệ có thể trả lời rằng "vì Bibi có lông vũ".

Trong mục này, chúng ta mới chỉ nêu ra các ý tưởng chính của phương pháp lập luận tiến và lập luận lùi. Sau đây chúng ta sẽ trình bày chi tiết hơn về thủ tục lập luận tiến và thủ tục lập luận lùi.

7.3. THỦ TỤC LẬP LUẬN TIẾN

Như chúng ta đã nói, trong các hệ dựa trên luật, chúng ta sẽ tách cơ sở tri thức thành hai phần

- Cở sở luật, ký hiệu là RB (*Rule Base*), và
- Cở sở sự kiện (bộ nhớ làm việc), ký hiệu là FB (*Fact Base*)

Với mỗi luật R:

Nếu P_1 và P_2 ... P_m **thì** Q

ta ký hiệu Conds là danh sách các điều kiện của luật, Conds = [P_1, P_2, ..., P_m], và ký hiệu Conc là kết luận của luật, Conc = Q. Ta sẽ xem mỗi luật R như là một cặp gồm danh sách các điều kiện và một kết luận:

$$R = (Conds(R), Conc(R))$$

Trong thủ tục lập luận tiến, chúng ta sẽ sử dụng luật suy diễn sau (xem mục 6.3).

$$\frac{P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_m \Rightarrow Q, \quad S}{P_1' \wedge \ldots \wedge P_{i-1}' \wedge P_i' \wedge \ldots \wedge P_m' \Rightarrow Q'}$$

trong đó P_i hợp nhất với S bởi phép thế θ, tức là $P_i\theta = S\theta$, và $P_k' = P_k\theta$ ($k = 1, ..., m; k \neq i$), $Q' = Q\theta$.

Luật suy diễn trên cho phép ta từ một luật có m điều kiện, một trong các điều kiện đó "khớp" với một sự kiện suy ra một luật mới có m - 1 điều kiện. Do đó nếu luật có m điều kiện, thì bằng cách áp dụng m lần luật suy diễn trên (nếu có thể) ta suy ra được một sự kiện. Sự kiện này là kết quả của việc ta áp dụng phép thế biến vào kết luận của luật. (Phép thế này là hợp thành của các phép thế trong mỗi lần áp dụng luật suy diễn trên). Việc áp dụng luật suy diễn Modus Ponens tổng quát cho luật có m điều kiện tương đương với việc áp dụng m lần luật suy diễn trên.

7.3.1. Thủ tục For_chain

Thủ tục sau đây, thủ tục For_Chair, thực hiện quá trình áp dụng luật suy diễn nêu trên để giảm bớt số điều kiện của một luật trong cơ sở luật. Khi mà ta dẫn tới một luật có phần điều kiện rỗng tức là ta đã suy ra một sự kiện. Trong thủ tục For_Chain, luật R = (Conds, Conc) là biến địa phương của thủ tục, Conds = [P_1, ..., P_i, ..., P_m]

procedure For_Chain (conds, conc);

begin

for mỗi S trong FB **do**

if S hợp nhất với điều kiện P_i trong Conds bởi phép thế θ

then {

Conds ← [$P_1\theta$, ..., $P_{i-1}\theta$, $P_{i+1}\theta$, ..., $P_m\theta$];

Conc ← Conc θ;

if Conds rỗng **then** Add(Conc, FB)

else For_Chain(Conds, Conc);

}

end;

***Chú ý*:** Trong thủ tục trên, thủ tục Add(Conc, FB) thực hiện việc kiểm tra xem kết luận conc có là sự kiện mới không (tức là không có sự kiện nào trong cơ sở sự kiện FB trùng với Conc hoặc nhận được từ Conc bằng cách đặt tên lại các biến), nếu Conc là sự kiện mới thì nó được đặt vào FB.

Quá trình lập luận tiến là quá trình áp dụng thủ tục trên cho các luật trong cơ sở luật cho tới khi nào không có sự kiện mới nào xuất hiện. Ta có thủ tục sau:

procedure Forward_Reasoning (RB, FB);

begin

repeat

for mỗi luật (conds, conc) trong FB **do** For_Chain(Conds, Conc);

until không có luật nào sinh ra sự kiện mới;

end;

Ví dụ. Giả sử cơ sở luật chứa luật sau (luật mẹ)

nếu

1. x là ngựa, và
2. x là mẹ của y, và
3. y chạy nhanh

thì x có giá

Cơ sở sự kiện gồm các sự kiện sau

Tom là ngựa

Ken là ngựa

Kit là ngựa

Bin là ngựa

Tom là mẹ của Bin

Tom là mẹ của Ken

Bin là mẹ của Kit.

Kit chạy nhanh.

Bin chạy nhanh.

Bằng cách sử dụng các vị từ House(x) (x là ngựa), Mother(x, y) (x là mẹ của y), Fast(y) (y chạy nhanh), Valuable(x) (x có giá), ta có thể viết luật trên thành câu:

House(x) Λ Mother(x, y) Λ Fast(y) ⇒ Valuable(x)

Cơ sở sự kiện gồm các câu phân tử sau

House(Tom) (1)

House(Ken) (2)

House(Kit) (3)

House(Bin) (4)

Mother(Tom, Bin) (5)

Mother(Tom, Ken)(6)

Mother(Bin, Kit) (7)

Fast(Kit) (8)

Fast(Bin) (9)

Ta xét xem quá trình sẽ diễn ra như thế nào khi ta áp dụng thủ tục For_chain cho luật mẹ và FB gồm các sự kiện (1) - (9).

Sự kiện (1) khớp với điều kiện thứ nhất của luật bởi phép thế [x/Tom], từ luật mẹ ta suy ra

$$\text{Mother(Tom, y)} \wedge \text{Fast(y)} \Rightarrow \text{Valuable(Tom)}$$

Sự kiện (5) hợp nhất với điều kiện Mother(Tom/y) bởi phép thế [y/ Bin], ta suy ra

$$\text{Fast(Bin)} \Rightarrow \text{Valuable(Tom)}$$

Từ sự kiện (9) và kéo theo trên, ta suy ra Valuable(Tom).

Sự kiện (2) cũng hợp nhất với điều kiện thứ nhất của luật, do đó ta suy ra

$$\text{Mother(Ken, y)} \wedge \text{Fast(y)} \Rightarrow \text{Valuable(Ken)}$$

Tới đây ta không suy diễn tiếp được, vì không có sự kiện nào hợp nhất được với điều kiện Mother(Ken, y). Điều tương tự cũng xảy ra, khi mà biến x trong luật mẹ được thế bởi Kit.

Từ sự kiện (4) và luật mẹ, ta suy ra

$$\text{Mother(Bin, y)} \wedge \text{Fast(y)} \Rightarrow \text{Valuable(Bin)}$$

Sự kiện (7) hợp nhất với điều kiện Mother(Bin, y), từ đó ta suy ra

$$\text{Fast(Kit)} \Rightarrow \text{Valuable(Bin)}$$

Từ kéo theo này và sự kiện (8), ta suy ra Valuable(Bin). Như vậy áp dụng thủ tục For_chain cho luật mẹ, chúng ta suy ra được hai sự kiện mới là "Tom có giá" và "Bin có giá".

Chúng ta có nhận xét rằng, nếu cài đặt thủ tục lập luận tiến một cách trực tiếp, thì thủ tục này sẽ rất không hiệu quả. Bởi vì chúng ta phải lặp lại rất nhiều lần thao tác đối sánh mỗi sự kiện trong bộ nhớ làm việc với các điều kiện trong các luật. Nếu bộ nhớ làm việc chứa f sự kiện, cơ sở luật chứa

r luật, mỗi luật gồm c điều kiện, thì chỉ trong một chu kỳ xem xét các luật ta đã phải thực hiện fcr phép đối sánh.

Sau đây chúng ta sẽ trình bày **thủ tục rete**, đó là một phương pháp cài đặt thuật toán suy diễn tiến. Thủ tục rete sẽ loại bỏ được các thao tác đối sánh lặp lại không cần thiết. (Rete là tên Latin, đồng nghĩa với từ net trong Tiếng Anh).

7.3.2. Thủ tục rete

Tư tưởng của thủ tục rete là người ta tạo ra một lưới cho mỗi luật trong cơ sở luật. Mỗi đỉnh trong lưới biểu diễn một bảng dữ liệu mà mỗi bản ghi trong bảng là một phép thế biến sao cho một hoặc một số điều kiện của luật được thoả mãn. Để thấy rõ lưới được tạo thành như thế nào, ta hãy xét một luật cụ thể sau (luật mẹ đã nêu trong mục 7.3.1)

$$House(x) \Lambda Mother(x, y) \Lambda Fast(y) \Rightarrow Valuable(x)$$

Luật này chứa ba điều kiện, người ta tạo ra ba đỉnh A, mỗi đỉnh ứng với một điều kiện.

- Đỉnh A_1 ứng với điều kiện thứ nhất House(x). Bảng A_1 chỉ gồm một cột x, vì House(x) chỉ chứa một biến x. Khi mà ta đưa các sự kiện trong bộ nhớ làm việc

  House(Tom)

  House(Ken)

  House(Kit)

  House(Bin)

  vào lưới, bảng A_1 sẽ chứa các bản ghi sau

A_1:

x
Tom
Ken

Kit
Bin

- Đỉnh A_2 ứng với điều kiện thứ hai Mother(x, y). Đỉnh này biểu diễn bảng gồm hai cột x và y. Mỗi bản ghi trong bảng này biểu diễn một phép thế biến mà công thức phân tử Mother(x, y) hợp nhất với một sự kiện trong bộ nhớ làm việc. Nếu ta cho các sự kiện

  Mother(Tom, Bin)

  Mother(Tom, Ken)

  Mother(Bin, Kit)

đi vào lưới, thì chúng ta sẽ có bảng A_2 như sau

A_2:

x	y
Tom	Bin
Tom	Ken
Bin	Kit

- Đỉnh A_3 ứng với điều kiện thứ ba Fast(y). Bảng A_3 chỉ chứa một cột y. Khi đưa các sự kiện

  Fast(Kit)

  Fast(Bin)

đi vào lưới, chúng ta sẽ có bảng A_3 như sau

A_3:

y
Kit

Bin

- Từ hai đỉnh A_1 và A_2, có các cung đi tới đỉnh B_2. Bảng B_2 là kết nối (theo nghĩa của phép toán kết nối quan hệ) của hai bảng A_1 và A_2 theo các cột cùng tên (ở đây là theo cột x). Như vậy mỗi bản ghi trong bảng B_2 là một phép thế biến mà cả hai điều kiện thứ nhất và thứ hai đều thoả mãn. Chẳng hạn, với nội dung của hai bảng A_1 và A_2 như ở trên, ta nhận được bảng B_2 như sau

B_2:

x	y
Tom	Bin
Tom	Ken
Bin	Kit

- Từ hai đỉnh B_2 và A_3, có các cung đi tới đỉnh B_3. Đỉnh B_3 là bảng nhận được bằng cách kết nối bảng B_2 với A_3. Bảng B_3 sẽ chứa các bản ghi là các phép thế biến mà cả ba điều kiện của luật đều thoả mãn. Chẳng hạn, từ hai bảng B_2 và A_3 như trên, ta có bảng B_3 như sau

B_3:

x	y
Tom	Bin
Bin	Kit

- Từ bảng B_3, bằng cách chiếu lên các cột ứng với các biến trong phần kết luận của luật, ta nhận được bảng C. Với bảng B_3 như trên, ta nhận được bảng C như sau

C:

x
Tom
Bin

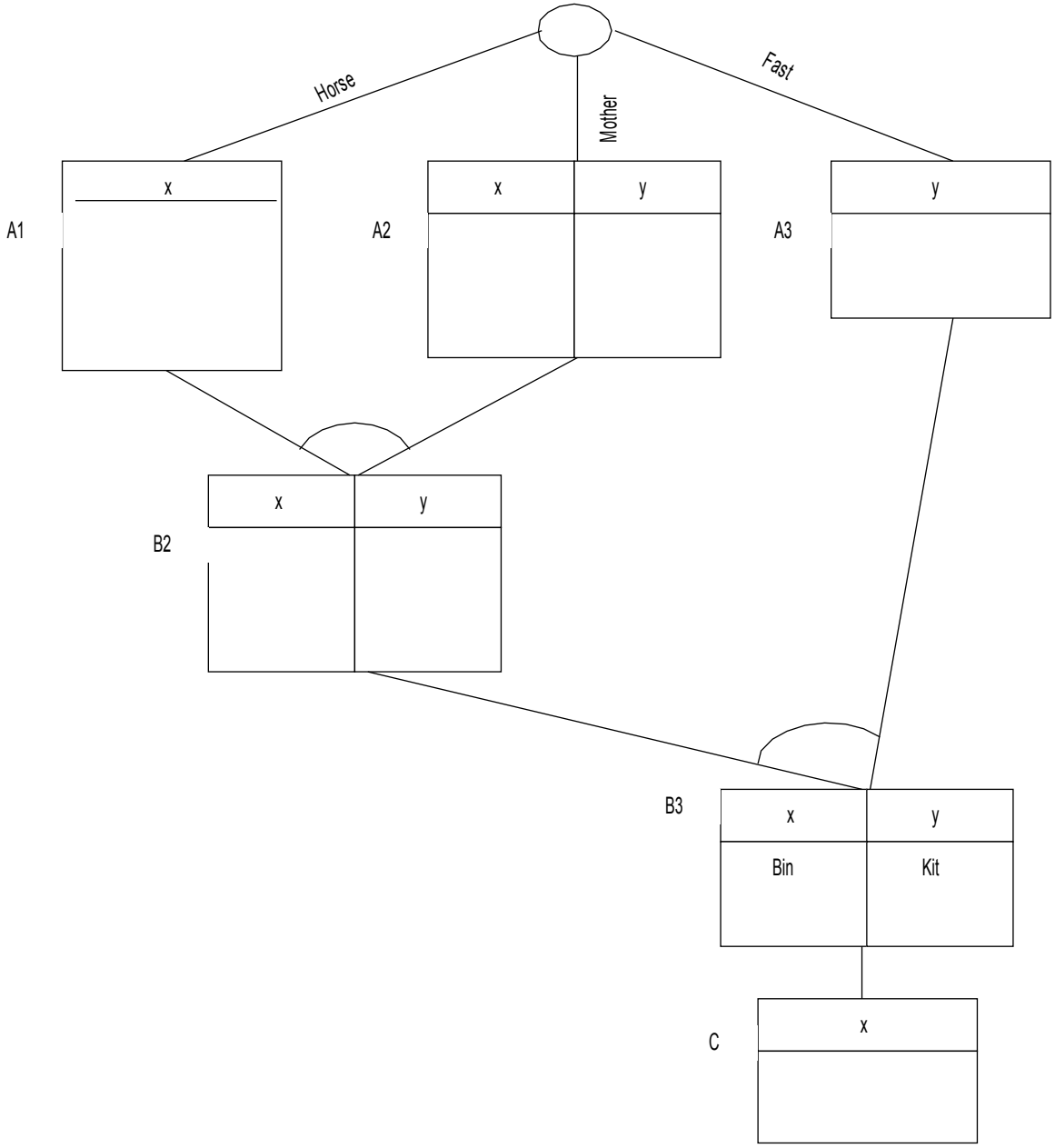
Bảng C chỉ ra rằng, từ luật mẹ và các sự kiện (1) - (9), ta suy ra các sự kiện Valuable(Tom) và Valuable(Bin).

Như vậy chúng ta đã xây dựng nên một lưới cho luật mẹ, lưới này được biểu diễn trong hình 7.4.

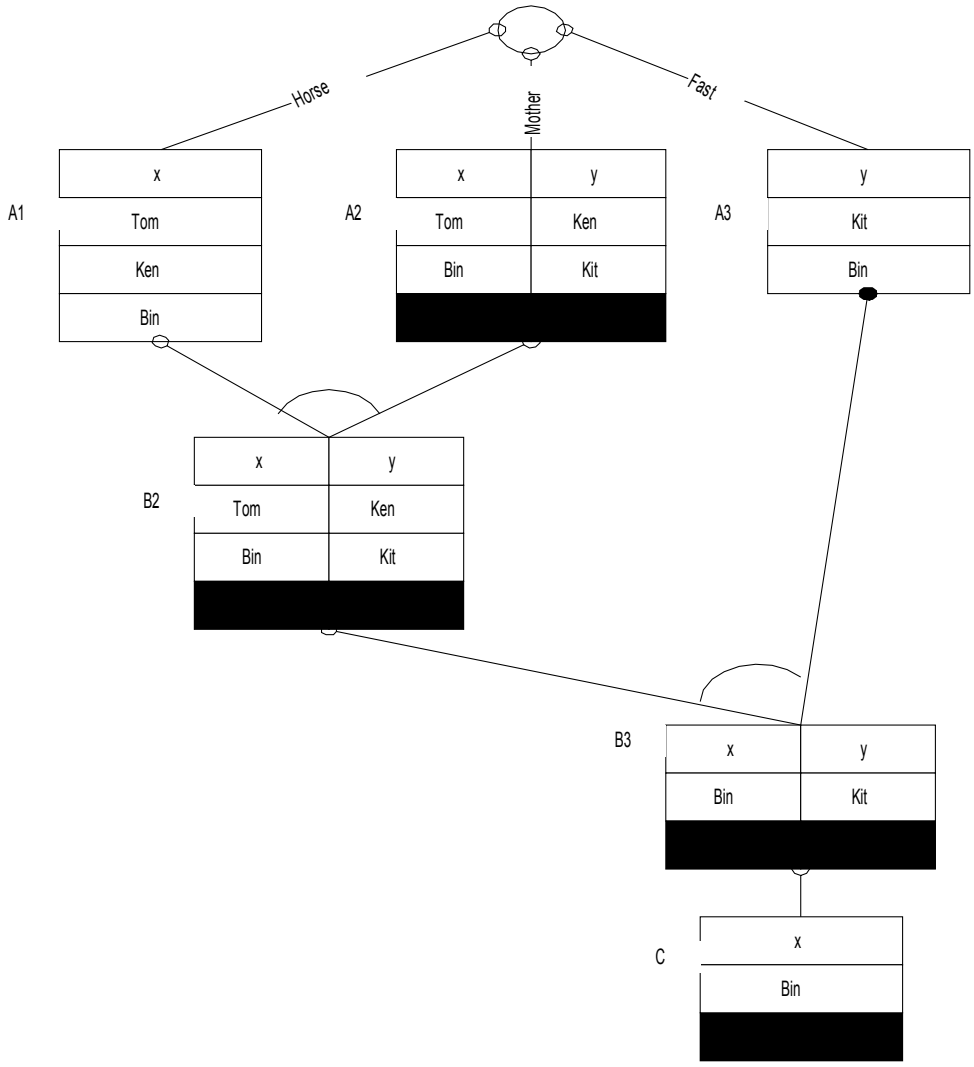
Sau khi đã tạo ra các lưới cho các luật trong cơ sở luật, quá trình lập luận tiến được thực hiện bằng cách lần lượt cho các sự kiện trong bộ nhớ làm việc đi qua lưới. Dữ liệu đi tới đỉnh nào ta sẽ thực hiện phép toán tương ứng với đỉnh đó. Phép toán ứng với các đỉnh A là phép hợp nhất, nếu hợp nhất thành công thì một bản ghi mới (ứng với phép thế biến là hợp nhất tử tìm được) được ghi vào bảng. Tại các đỉnh B ta sẽ thực hiện phép toán kết nối. Khi một bản ghi được chuyển tới đỉnh B từ một đỉnh cha của B, bản ghi này sẽ được kết nối với các bản ghi trong bảng ở đỉnh cha khác của B. Các bản ghi nhận được qua phép kết nối tại đỉnh B sẽ được ghi vào bảng B và được chuyển tới đỉnh con của B. Luồng dữ liệu cứ thế tiếp tục được chuyển qua lưới. Khi dữ liệu đi tới đỉnh C, ta thực hiện phép chiếu, các bản ghi mới nhận được qua phép chiếu sẽ được ghi vào bảng C. Từ đỉnh C sẽ đi ra các sự kiện mới được suy ra bởi luật (mỗi sự kiện ứng với một bản ghi trong bảng C). Các sự kiện mới này lại được tiếp tục cho đi qua lưới.

Ví dụ. Giả sử các bảng trong lưới hình 7.4. đã chứa các dữ liệu được cho trong hình 7.5 (các bản ghi không tô đậm). Giả sử bây giờ ta cho sự kiện Mother(Tom, Bin) đi vào lưới. Sự kiện này theo cung gắn nhãn Mother đi tới đỉnh A_2 để đối sánh với điều kiện Mother(x, y). Phép hợp nhất thành công, một bản ghi mới [x/Tom, y/Bin] (bản ghi tô đậm) được thêm vào bảng A_2. Dữ liệu này đi xuống đỉnh B_2, nó kết nối với các bản ghi đã có trong bảng A_1, tạo thành một bản ghi mới trong bảng B_2, [x/Tom, y/Bin], bản ghi

này lại được chuyển tới đỉnh B_3, nó kết nối với các bản ghi đã có trong bản A_3, tạo ra một bản ghi mới trong bảng B_3 (bản ghi tô đậm). Ta lại chuyển bản ghi này tới đỉnh C, tại đây nó được chiếu lên các cột của bảng C, tạo ra bản ghi mới [x/Tom]. Một sự kiện mới được suy ra là Valuable(Tom). (hình 7.5)



Hình 7.4. Lưới tương ứng với luật mẹ.



Hình 7.5. Kết quả của một sự kiện đi vào lưới.

Sau đây chúng ta đưa ra thủ tục xây dựng lưới cho mỗi luật trong cơ sở luật.

Giả sử luật có chứa m điều kiện. Để xây dựng một lưới cho luật này ta thực hiện các bước sau đây:

1. Lưới chứa một đỉnh xuất phát, từ đỉnh này có các cung đi tới các đỉnh A_i (i = 1, ..., m). Mỗi cung được gắn nhãn, với nhãn là công thức phân tử, công thức này là một điều kiện trong luật đó.

2. Mỗi đỉnh A_i tương ứng với một bảng, các cột trong bảng ứng với các biến trong công thức phân tử là nhãn của cung đi tới đỉnh A_i. Mỗi dòng trong bảng này là một phép thế biến mà một sự kiện trong bộ nhớ làm việc hợp nhất với công thức phân tử đó.

3. Nếu $m \geq 2$ thì từ đỉnh A_1 và A_2 có các cung đi tới đỉnh B_2. Đỉnh B_2 tương ứng với một bảng, mỗi cột trong bảng này là một cột trong bảng A_1 hoặc trong A_2. Bảng B_2 là kết nối của bảng A_1 và A_2, mỗi dòng trong bảng này là một phép thế biến sao cho hai điều kiện đầu tiên của luật trở thành đúng.

4. Từ các đỉnh B_k và A_{k+1} ($k = 2, ..., m-1$) có các cung đi tới đỉnh B_{k+1}. Đỉnh B_{k+1} tương ứng với một bảng, bảng này là kết nối của bảng B_k với bảng A_{k+1}. Mỗi dòng trong bảng này là một phép thế biến mà các điều kiện 1, 2, ..., k+1 của luật được thoả mãn.

5. Từ đỉnh B_m có cung đi tới đỉnh C. Đỉnh C tương ứng với một bảng, mỗi cột của bảng này ứng với một biến trong phần kết luận của luật. Bảng C nhận được từ bảng B_m bằng cách chiếu lên các cột của bảng C. Mỗi bản ghi trong bảng C biểu diễn một kết luận được suy ra từ luật.

Sau khi đã xây dựng được các lưới cho các luật trong cơ sở luật, ta liên kết các lưới đó để tạo ra một lưới chung cho toàn bộ cơ sở luật. Có thể có nhiều luật khác nhau có chung một số điều kiện nào đó, khi đó các đỉnh ứng với các điều kiện chung đó sẽ được đồng nhất làm một. Chẳng hạn, ngoài luật mẹ, còn có các luật khác cũng chứa điều kiện House(x), khi đó đỉnh ứng với điều kiện này sẽ là đỉnh chung cho tất cả các luật chứa điều kiện House(x). Tương tự, có thể có nhiều luật khác nhau cùng chung phần kết luận, trong trường hợp này đỉnh C ứng với kết luận chung này sẽ được đồng nhất làm một trong lưới chung. Sơ đồ của lưới chung được cho trong hình 7.6.

7.3.3. Hệ hành động dựa trên luật

Trên đây chúng ta chỉ mới xét các luật, trong đó mỗi phần kết luận của một luật xác định một khẳng định mới được suy ra khi mà tất cả các điều kiện của luật được thoả mãn. Trong nhiều áp dụng, cơ sở luật của hệ cần được đưa vào các luật mà phần kết luận của luật là một hành động hệ cần thực hiện. Chúng ta sẽ gọi các luật dạng này là luật hành động. Một luật hành động có dạng

nếu

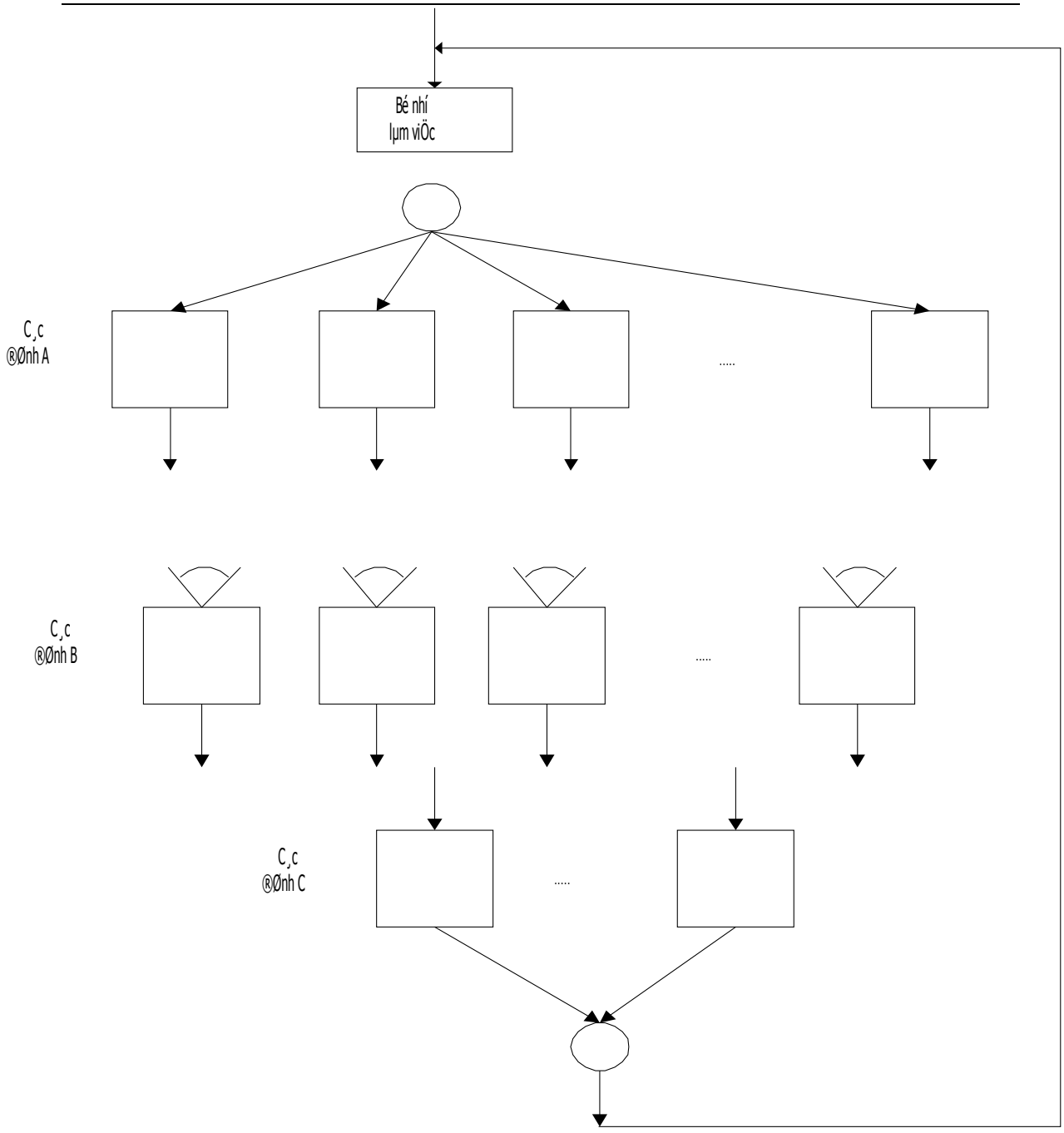
1. <điều kiện 1>, và
2. <điều kiện 2>, và

....

m. <điều kiện m>

thì <hành động>

Hành động trong các luật hành động có thể là thêm vào một sự kiện mới, có thể là loại bỏ một sự kiện đã có trong bộ nhớ làm việc; hành động cũng có thể là thực hiện một thủ tục nào đó. Trong các robot được trang bị một hệ dựa trên luật, thì phần kết luận của luật có thể là một hành động nào đó mà robot cần thực hiện.



Hình 7.6. Sơ đồ một lưới chung cho cơ sở luật

Người ta phân biệt hai dạng hệ:

- Các hệ dựa trên luật sử dụng lập luận tiến và phần kết luận của các luật xác định các khẳng định mới được gọi là các **hệ diễn dịch** (*deduction systems*).

- Các hệ dựa trên luật mà phần kết luận của các luật xác định các hành động cần thực hiện được gọi là **hệ hành động dựa trên luật** (*rule-based reaetion systems*).

Trong các hệ diễn dịch, chúng ta xem mọi luật đều sinh ra các khẳng định mới "có giá trị" như nhau; tức là, ta không xem khẳng định do luật này sinh ra là "tốt" hơn khẳng định do luật khác sinh ra. Do đó trong các hệ diễn dịch, khi mà nhiều luật có thể cháy được (một luật được gọi là **cháy được** nếu tất cả các điều kiện của luật được thoả mãn) ta có thể cho tất cả các luật đó cháy (một luật cháy để sinh ra khẳng định mới).

Trong các hệ hành động, khi có nhiều hơn một luật có thể cháy, nói chung, chúng ta chỉ muốn thực hiện một trong các hành động có thể. Do đó, trong các hệ hành động, chúng ta cần có chiến lược giải quyết va chạm để quyết định cho luật nào cháy trong số các luật có thể cháy. Sau đây là một số chiến lược giải quyết va chạm.

- Sắp xếp các luật theo thứ tự ưu tiên. Trong các luật có thể cháy, luật nào có mức ưu tiên cao nhất sẽ được thực hiện.
- Sắp xếp dữ liệu. Các sự kiện trong bộ nhớ làm việc được sắp xếp theo thứ tự ưu tiên. Luật nào mà các điều kiện của nó được làm thoả mãn bởi các điều kiện có mức ưu tiên cao sẽ được thực hiện trước.
- Các luật được phân thành các nhóm. Trong mỗi nhóm luật, được chỉ ra các điều kiện để áp dụng các luật của nhóm. Các điều kiện này liên quan đến nội dung của bộ nhớ làm việc.
- Sử dụng các siêu luật (*metarule*). Đó là các luật mà phần điều kiện của nó liên quan tới nội dung của các luật và nội dung của bộ nhớ làm việc, còn phần kết luận của nó chỉ ra các luật có thể được áp dụng hoặc có thể bi cấm áp dụng. Các siêu luật sẽ điều khiển sự cho phép các luật cháy.

Đương nhiên là, việc sử dụng chiến lược giải quyết va chạm nào phụ thuộc vào từng áp dụng. Tuỳ theo mục đích thiết kế của hệ mà ta lựa chọn chiến lược giải quyết va chạm cho thích hợp.

7.4. THỦ TỤC LẬP LUẬN LÙI

Trong các hệ dựa trên luật, chúng ta còn có thể sử dụng phương pháp lập luận lùi. Lập luận lùi cho phép ta tìm ra các phép thế biến mà giả thuyết đưa ra trở thành đúng (là hệ quả logic của cơ sở tri thức). Do đó trong hệ dựa trên luật chúng ta có thể sử dụng lập luận lùi để tìm ra các câu trả lời cho các câu hỏi được đặt ra bởi người sử dụng.

Một câu hỏi đặt ra có thể xem như một giả thuyết (ký hiệu là Hyp) cần kiểm tra. Giả thuyết có thể là một câu phân tử hoặc là hội của các câu phân tử:

$$Hyp = H_1 \Lambda \ldots \Lambda H_m$$

trong đó H_i (i = 1, ..., m) là các câu phân tử.

Mục đích của chúng ta là kiểm chứng xem giả thuyết có thể trở thành đúng không, và nếu có thì với các phép thế biến nào nó trở thành đúng.

Chúng ta sẽ xử lý Hyp như một danh sách các giả thuyết H_i:

$$Hyp = [H_1, ..., H_m]$$

Chúng ta sẽ xét mỗi luật

$$P_1 \wedge \ldots \wedge P_m \Rightarrow Q$$

như một cặp (conds, conc); trong đó Conds là danh sách các điều kiện của luật.

$$Conds = [P_1, ..., P_m]$$

và Conc là kết luận của luật, Conc = Q.

Một sự kiện S (câu phân tử) được xem như một luật không có điều kiện, tức là Conds = [ ] và Conc = S.

Tư tưởng của phương pháp lập luận lùi là như sau. Với mỗi giả thuyết trong danh sách các giả thuyết, ta tìm những luật có phần kết luận hợp nhất với giả thuyết đó. Nếu luật này là một sự kiện thì ta loại bỏ giả thuyết đang xét khỏi danh sách các giả thuyết. Nếu không thì ta xem các điều kiện của luật là các giả thuyết mới xuất hiện và giả thuyết đang xét được thay bởi các giả thuyết mới đó. Khi đó ta nhận được một danh sách các giả thuyết mới. Lặp lại quá trình trên cho danh sách các giả thuyết mới này. Trong quá trình trên ta lưu lại hợp thành của các phép thế đã sử dụng θ. Nếu tới một bước nào đó, danh sách các giả thuyết trở thành rỗng, thì ta kết luận giả thuyết ban đầu là đúng với phép thế biến θ.

Sau đây là thủ tục suy diễn lùi. Trong thủ tục này, Hyp và θ là các biến địa phương trong thủ tục. Giá trị ban đầu của Hyp là danh sách các giả thuyết ban đầu (biểu diễn câu hỏi được đặt ra), còn giá trị ban đầu của θ là phép thế rỗng.

procedure *Backward_Chaining (Hyp, θ);*
begin
H ← giả thuyết đầu tiên trong danh sách Hyp;
for *mỗi luật R = (Conds, Q)* ***do***
if *H hợp nhất với Q bởi phép thế* θ_1 **then**
1. *Loại H khỏi danh sách Hyp;*
2. *Thêm các điều kiện của luật Conds vào danh sách Hyp;*
3. *áp dụng phép thế* θ_1 *vào các giả thuyết trong danh sách Hyp;*
4. *Lấy hợp thành của các phép thế* θ *và* θ_1 *để nhận được phép thế* θ *mới, tức là* $\theta \leftarrow \theta\theta_1$*;*

if *Hyp = [ ]* **then** *cho ra θ*
else *Backward_Chaining (Hyp, θ);*
end*;*

Trong thủ tục lập luận lùi, mỗi θ được cho ra là một phép thế biến làm cho giả thuyết ban đầu trở thành đúng, tức là (Hyp) $\theta = H_1\theta \Lambda ... \Lambda H_m\theta$ là đúng (là hệ quả logic của cơ sở tri thức). Do đó mỗi phép thế biến θ được cho ra bởi thủ tục là một câu trả lời cho câu hỏi đặt ra.

Ví dụ. Giả sử cơ sở tri thức chứa các sự kiện sau

House(Tom) (Tom là ngựa) (1)

House(Ken) (2)

House(Kit) (3)

House(Bin) (4)

Mother(Tom, Bin) (Tom là mẹ Bin) (5)

Mother(Tom, Ken) (6)

Mother(Bin, Kit) (7)

Fast(Kit) (Kit chạy nhanh) (8)

Winner(Bin) (Bin thắng cuộc) (9)

Giả sử cơ sở tri thức chứa hai luật sau

House(x) Λ Mother(x, y) Λ Fast(y) ⇒ Valuable(x) (10)

(**nếu** 1. x là ngựa, và

2. x là mẹ y, và

3. y chạy nhanh

thì x có giá)

Winner(z) ⇒ Fast(z) (11)

(**nếu** z thắng cuộc **thì** x chạy nhanh)

Câu hỏi đặt ra là: Con ngựa nào có giá ?

Giả thuyết ban đầu Hyp = [Valuable(w)] và θ = [ ]. Giả thuyết Valuable(w) hợp nhất được với kết luận của luật (10) bởi phép thế θ_1 = [w/x], do đó ta nhận được danh sách các giả thuyết mới

Hyp = [House(x), Mother(x, y), Fast(y)]

và θ = $\theta\theta_1$ = [w/x]

Giả thuyết House(x) hợp nhất được với sự kiện (1) bởi phép thế θ_1 = [x/Tom], ta nhận được danh sách các giả thuyết mới

Hyp = [Mother(Tom, y), Fast(y)]

và θ = [w/x][x/Tom] = [w/Tom]

Giả thuyết Mother(Tom, y) hợp nhất được với sự kiện (5) bởi phép thế θ_1 = [y/Bin], ta nhận được danh sách các giả thuyết

Hyp = [Fast(Bin)]

và θ = [w/Tom][y/Bin] = [w/Tom, y/Bin]

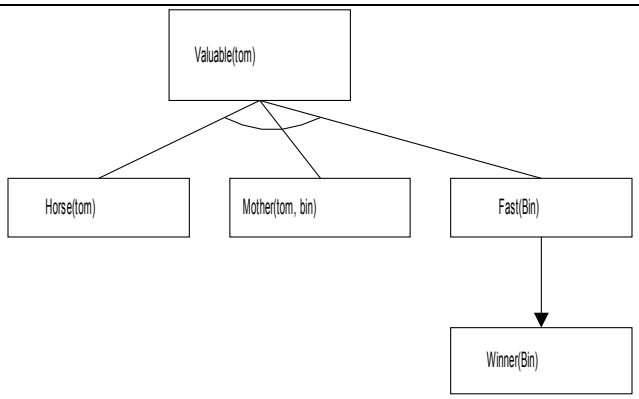
Giả thuyết Fast(Bin) hợp nhất được với kết luận của luật (11) bởi phép thế [z/Bin], do đó ta có

Hyp = [Winner(Bin)]

và θ = [w/Tom, y/Bin, z/Bin]

Giả thuyết Winner(Bin) trùng với sự kiện (9) (hợp nhất được bởi phép thế θ_1 = []). Do đó danh sách các giả thuyết trở thành rỗng với phép thế θ = [w/Tom, y/Bin, z/Bin]. Như vậy với phép thế này thì giả thuyết Valuable(w) trở thành đúng, hay nói cách khác, Tom là con ngựa có giá.

Từ các luật được sử dụng trong quá trình lập luận trên, và từ phép thế θ thu được, ta có thể xây dựng nên cây chứng minh cho Valuable(Tom) (xem hình 7.7).



Hình 7.7. Cây chứng minh cho Valuable(Tom)

Chúng ta còn tìm được một phép thế biến khác θ = [w/Bin, y/Kit] để cho Valuable(w) trở thành đúng. Do đó, ta tìm ra Tom và Bin là các con ngựa có giá.

7.5. BIỂU DIỄN TRI THỨC KHÔNG CHẮC CHẮN

Trong đời sống thực tế, có rất nhiều điều mà ngay cả các chuyên gia cũng không hoàn toàn tin tưởng chúng là đúng hay sai. Đặc biệt là các kết luận trong chẩn đoán y học, trong dự báo thời tiết, trong phỏng đoán sự hỏng hóc của máy móc, chúng ta không thể tin tưởng 100% các kết luận đưa ra là đúng. Chẳng hạn, nếu xe máy đang chạy bị chết máy và kiểm tra xăng hãy còn thì có thể tin rằng 90% là do có vấn đề ở bugi. Tuy nhiên vẫn còn 10% phỏng đoán đó là sai, xe bị chết máy do các nguyên nhân khác. Do đó trong các hệ dựa trên luật, chúng ta còn phải đưa vào **mức độ chắc chắn** của các luật và các sự kiện trong cơ sở tri thức. Chúng ta sẽ gán cho mỗi luật hoặc sự kiện một mức độ chắc chắn nào đó, mức độ chắc chắn là một số nằm giữa 0 và 1. Cách viết

$$A_1 \wedge \ldots \wedge A_n \Rightarrow B \quad : \quad C \qquad (1)$$

có nghĩa là luật $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ có độ chắn chắn là C ($0 \le C \le 1$).

Chúng ta cần phải đưa ra phương pháp xác định mức độ chắc chắn của các kết luận được suy ra.

Trước hết chúng ta đánh giá kết luận suy ra từ luật chỉ có một điều kiện. Giả sử ta có luật

$$A \Rightarrow B \quad : \quad C \qquad (2)$$

Theo lý thuyết xác suất, ta có

$$\Pr(B) = \Pr(B \mid A)\Pr(A) \qquad (3)$$

trong đó Pr(B), Pr(A) là xác suất của sự kiện B, A tương ứng (tức là mức độ chắc chắn của B, A tương ứng), còn Pr(B | A) là xác suất có điều kiện của B khi A đã xảy ra, ở đây Pr(B | A) là mức độ chắc chắn của luật A $\Rightarrow$ B, tức là bằng C.

Trong trường hợp luật có n (n > 1) điều kiện, tức là các luật dạng (1), ta xem A = $A_1 \wedge \ldots \wedge A_n$. Trong trường hợp này, mức độ chắc chắn của A, Pr(A) được tính bằng các phương pháp khác nhau, tuỳ thuộc vào các sự kiện A_i (i = 1, ..., n) là độc lập hay phụ thuộc.

Giả sử các sự kiện A_i (i = 1, ..., n) là độc lập, khi đó

$$\Pr(A) = \Pr(A_1) \ldots \Pr(A_n) \qquad (4)$$

***Ví dụ*:** Giả sử cơ sở tri thức của hệ chứa luật sau

nếu 1. X có tiền án, và

2. X có thù oán với nạn nhân Y, và

3. X đưa ra bằng chứng ngoại phạm sai

thì X là kẻ giết Y.

với mức độ chắc chắn 90%.

Giả sử ta có các sự kiện

- Hung có tiền án, với mức độ chắc chắn là 1.
- Hung có thù oán với nạn nhân Meo, với mức độ chắc chắn là 0,7.
- Hung đưa ra bằng chứng ngoại phạm sai, với mức độ chắc chắn là 0,8.

Từ các sự kiện và luật trên, ta có

Pr(A) = 1.0,7.0,8 = 0,56

Pr(B) = 0,9.0,56 = 0,504

Như vậy mức độ chắc chắn của kết luận "Hung là kẻ giết Meo" là 50,4%.

Công thức (4) chỉ áp dụng cho các sự kiện A_1, ..., A_n là độc lập (tức sự xuất hiện của sự kiện này không ảnh hưởng gì đến sự xuất hiện của các sự kiện khác). Nếu các sự kiện A_1, ..., A_n là phụ thuộc, ta có thể tính mức độ chắc chắn của điều kiện của luật, $A = A_1 \wedge \ldots \wedge A_n$, theo công thức sau:

$$Pr(A) = \min (Pr(A_1), ..., Pr(A_n)) \qquad (5)$$

Chẳng hạn, với các thông tin trong ví dụ trên, từ công thức (5) ta có

Pr(A) = min(1, 0,7, 0,8) = 0,7

Do đó Pr(B) = 0,9. 0,7 = 0,63

Ngoài công thức (5), người ta còn đưa ra các phương pháp khác để tính mức độ chắc chắn Pr(A), khi mà $A = A_1 \wedge \ldots \wedge A_n$ và các A_1, ..., A_n không độc lập.

7.6. HỆ LẬP TRÌNH LOGIC

Hiện nay đã có nhiều hệ lập trình logic ra đời mà tiêu biểu là Prolog. Prolog là viết tắt của cụm từ tiếng Pháp

Programmation en Logique

Hệ Prolog đầu tiên ra đời vào năm 1973 bởi Alain Colmerauer và nhóm trí tuệ nhân tạo thuộc Đại học tổng hợp Aix-Marseilie, Pháp. Mục đích ban đầu của hệ này là dịch các ngôn ngữ tự nhiên. Năm 1977, David Warren thuộc Đại học tổng hợp Edinburgh đã cài đặt một phiên bản của Prolog, mang tên là Prolog-10. Năm 1981, người Nhật đã thông báo sử dụng Prolog như ngôn ngữ cơ bản cho máy tính thế hệ thứ năm. Hiện nay đã có

nhiều hệ Prolog khác nhau về tốc độ, môi trường làm việc, ..., song phần lớn các hệ này tương thích với Prolog-10, Prolog-10 được công nhận như Prolog chuẩn.

Prolog đã được sử dụng như một công cụ phần mềm để phát triển các hệ thông minh. Nó đã được áp dụng trong nhiều lĩnh vực trí tuệ nhân tạo: giải quyết vấn đề, các hệ chuyên gia, biểu diễn tri thức, lập kế hoạch, xử lý ngôn ngữ tự nhiên, học máy, ...

Trong các ngôn ngữ lập trình truyền thống (chẳng hạn, Pascal, C, ...) một chương trình là một dãy các lệnh mà máy cần thực hiện. Người lập trình để viết một chương trình trong các ngôn ngữ truyền thống, phải dựa vào thuật toán đã có và cách biểu diễn dữ liệu để lập ra một dãy các lệnh chỉ dẫn cho máy cần phải thực hiện các hành động nào.

Điều khác nhau căn bản của lập trình Prolog so với lập trình truyền thống là như sau:

- Trong Prolog người lập trình mô tả vấn đề bằng các câu trong logic.
- Hệ sẽ sử dụng lập luận logic để tìm ra các câu trả lời cho vấn đề.

Do đó một chương trình Prolog là sự đặc tả của một vấn đề. Vì lý do này Prolog là ngôn ngữ lập trình khai báo (*declarative language*).

Trong Prolog, chỉ được phép sử dụng các câu Horn, tức là mỗi câu hoặc là câu phân tử, hoặc là một luật nếu - thì mà các điều kiện của luật và kết luận của luật đều là câu phân tử.

Một chương trình Prolog là một dãy các luật có dạng sau

$$A :- B_1, ..., B_m$$

trong đó $m \geq 0$, A và B_i ($i = 1, ..., m$) là các câu phân tử. Luật trên được đọc là "A nếu B_1 và ... và B_m", nó là cách viết trong Prolog câu sau

$$B_1 \wedge \ldots \wedge B_m \Rightarrow A$$

Trong luật trên, A được gọi là **đầu**, danh sách các câu (B_1, ..., B_m) được gọi là **thân** của luật. Nếu m = 0, ký hiệu ":-" sẽ được bỏ đi, khi đó ta có câu phân tử A và nó được gọi là một **sự kiện**.

Ví dụ: Giả sử chúng ta biết các thông tin sau đây về An và Ba.

An yêu thích mọi môn thể thao mà cậu chơi.

Bóng đá là môn thể thao.

Bóng bàn là môn thể thao.

An chơi bóng đá.

Ba yêu thích mọi thứ mà An yêu thích.

Các câu trên được chuyển thành một chương trình Prolog như sau

```
Likes(An, X) :-  Sport(X), Plays(An, X)

Sport(Football).

Sport(Tennis).

Plays(An, Football).

Likes(Ba, Y) :- Likes(An, Y).
```

Với chương trình Prolog trên (nó mô tả sở thích thể thao của An và Ba), ta có thể đặt ra các câu hỏi, chẳng hạn "An yêu thích cái gì ?". Câu hỏi này được viết trong Prolog như sau.

```
? - Likes(An, X).
```

Khi đưa vào một câu hỏi, hệ Prolog sẽ thực hiện quá trình suy diễn logic để tìm ra các câu trả lời cho câu hỏi. Chẳng hạn, với câu hỏi trên Prolog sẽ đưa ra câu trả lời:

```
X = football
```

điều đó có nghĩa là "An yêu thích bóng đá".

Một cách tổng quát, một câu hỏi có dạng

```
? - G1, ..., Gn.
```

trong đó mỗi G_i (i = 1, ..., n) là một công thức phân tử, danh sách (G_1, ..., G_n) được gọi là đích, các G_i (i = 1, ..., n) được gọi là đích con. Nếu n = 0, ta có đích rỗng, ký hiệu là .

Đến đây chúng ta muốn biết, làm thế nào mà Prolog tìm ra các câu trả lời cho các câu hỏi ? Một chương trình Prolog có thể xem như một cơ sở tri thức. Thủ tục tìm câu trả lời của Prolog chẳng qua là một cách cài đặt phương pháp lập luận lùi mà chúng ta đã trình bày ở mục 7.4. Trong cách cài đặt phương pháp lập luận lùi này, người ta đã sử dụng kỹ thuật tìm kiếm theo độ sâu, các câu trong chương trình Prolog được xem xét theo thứ tự từ trên xuống dưới, các đích con được xem xét để làm thoả mãn theo thứ tự từ trái sang phải. Ngoài ra, thủ tục lập luận của Prolog còn cho phép người lập trình có thể sử dụng vị từ "**cut**" khi cần thiết để đảm bảo cho chương trình đúng đắn và hiệu quả.

Một đặc diểm nữa của Prolog là, thay cho việc sử dụng các câu là phủ định của các câu phân tử. Prolog đưa vào vị từ **not** biểu diễn ***phủ định như thất bại*** (*negation as failure*). Điều đó có nghĩa là not(P) được xem là đúng nếu ta thất bại trong việc tìm ra một chứng minh P đúng, tức là với cơ sở tri thức hiện có, ta không tìm được một phép thế biến nào để cho P trở thành hệ quả logic của cơ sở tri thức. Sau đây là một ví dụ sử dụng vị từ **not**.

Likes(An, X) : - Animal(X), not(Snaker(X)).

Bạn đọc muốn tìm hiểu sâu hơn về các kỹ thuật lập trình Prolog và các ứng dụng trong Trí tuệ nhân tạo, có thể tìm đọc các tài liệu [4] và [14].

7.7. HỆ CHUYÊN GIA

Trong mục này chúng ta sẽ giới thiệu về ***hệ chuyên gia*** (***expert system***). Hệ chuyên gia là một chương trình máy tính có khả năng giải quyết vấn đề trong một lĩnh vực áp dụng nào đó (thường là hẹp), nó có thể làm việc giống như một chuyên gia con người trong lĩnh vực đó. Một bác sĩ chữa bệnh, từ các triệu chứng của bệnh nhân, từ các kết quả xét nghiệm, với vốn tri thức của mình, bác sĩ có thể đưa ra các kết luận bệnh nhân bị bệnh gì và đưa ra phương án điều trị. Một hệ chuyên gia chẩn đoán bệnh cũng có thể

làm việc như các bác sĩ. Một hệ chuyên gia cần được trang bị các tri thức của các chuyên gia trong một lĩnh vực áp dụng. Vì vậy hệ chuyên gia là một ***hệ tri thức*** (***knowledge - based system***). Cũng giống như một chuyên gia con người, hệ chuyên gia còn có khả năng giải thích được các hành vi của nó, các kết luận mà nó đã đưa ra cho người sử dụng. Hệ chuyên gia còn có khả năng cập nhật tri thức từ các chuyên gia và từ người sử dụng.

Hiện nay, nhiều hệ chuyên gia đã được ử dụng trong nhiều lĩnh vực. Chúng ta nêu ra một số hệ chuyên gia nổi tiếng:

Trong chuẩn đoán y học có các hệ MYCIN, CASNET vaf CADUCEUS.

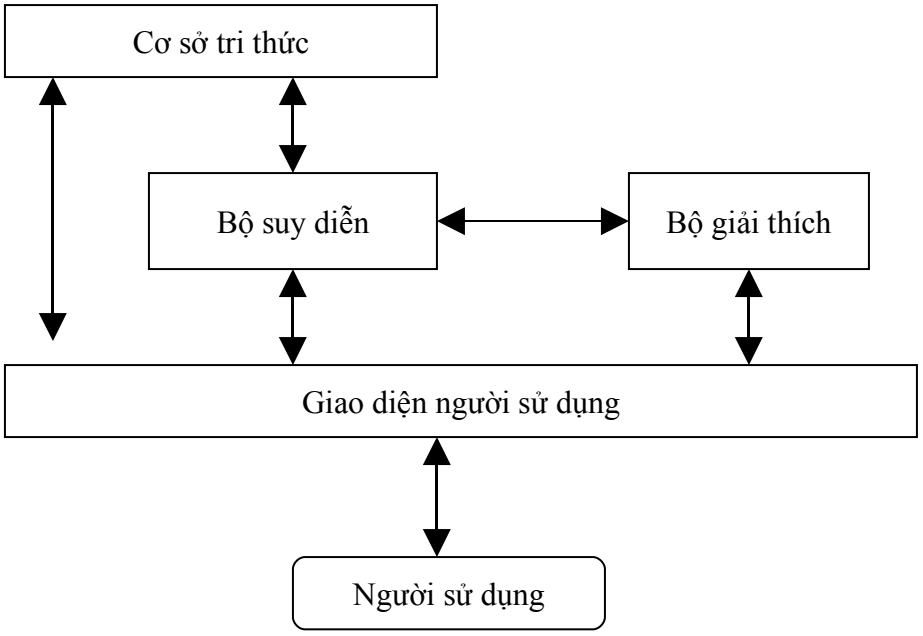
Phân tích cấu trúc phân tử: hệ DENDRAL.

Vi phân và tích phân ký hiệu: các hệ MACSYMA, SAINT và MATHLAB.

Hiểu tiếng nói: các hệ HEARSAY I và II.

Chuẩn đoán hỏng hóc của máy tính: hệ DART.

Cấu trúc cơ bản của hệ chuyên gia được mô tả trong hình 7.8.



Hình 7.8. Kiến trúc của một hệ chuyên gia.

Một hệ chuyên gia bao gồm các thành phần cơ bản sau:

- ***Cơ sở tri thức*** (***Knowledge base***) chứa các tri thức của các chuyên gia trongmột lĩnh vực áp dụng nào đó.

Tri thức của các chuyên gia có thể biểu hiện trong một ngôn ngữ biểu diễn tri thức thích hợp nào đó. Hiện nay trongđa số các hệ chuyên gia, tri thức được biểu hiện dưới dang luật if then. Cơ sở tri thức của cáchệ chuyên gia có thể chứa các tri thức không chắc chắn, hoặc không đầy đủ.

- ***Bộ suy diễn*** (***inference engine***) có chức năng thực hiện quá trình suy diễn dựa trên tri thức trongcơ sở tri thức và các thôngtin mà người sử dụng đưa vào. để tìm ra câu trả lời cho các vấn đề được đặt ra.

Trong các hệ chuyên gia dựa trên luật (***rule-based expert system***), thủ tục suy diễn có thể là suy diễn tiến hoặc suy diễn lùi như chúng ta đã trình bày trong các mục 7.3 và 7.4. Sử dụng cơ chế suy diễn nào(tiến hoặc lùi) là tuỳ thuộc vào mục đích xây dựng hệ chuyên gia.

- ***Bộ giải thích*** (***explanation generator***) có chức năng cung cấp cho người sử dụng những lời giải thích về các kết luận mà hệ đưa ra, tại sao hệ đã dẫn đến những kết luận như thế.
- ***Giao diện người sử dụng*** (***user-interface***) giúp cho hệ giao tiếp với người sử dụng một cách thuận tiện. Nó chuyển đổi các thông tin mà người sử dụng đưa vào thành dạng mà hệ có thể xử lý được, và ngược lại, nó chuyển đổi các câu trả lời của hệ và các lời giải thích sang ngôn ngữ mà người sử dụng có thể hiểu.

Người ta tách cơ sở tri thức khỏi các bộ phận còn lại của hệ chuyên gia. Các bộ phận còn lai gồm bộ suy diễn, bộ giải thích và giao diện người sử dụng tạo thành ***khung hệ chuyên gia*** (***expert system shell***). Sở dĩ có thể tách hệ chuyên gia thành hai phần: cơ sở tri thức và khung, là vì các lý do sau. Cơ sở tri thức phụ thuộc hoàn toàn vào lĩnh vực áp dụng. Mặt khác, khung là tương đối độc lập với lĩnh vực áp dụng. Do đó để phát triển các hệ chuyên gia, cho một số lĩnh vực, người ta có thể xây dựng khung để sử dụng chung cho nhiều hệ. Sau đó chỉ cần "nạp" cơ sở tri thức về một lĩnh vực áp dụng cụ thể, ta sẽ tạo ra một hệ chuyên gia cho lĩnh vực đó. Đương nhiên là, cơ sở tri thức phải được xây dưng theo đúng "khuôn mẫu" mà khung có thể hiểu được.

Trên đây chúng ta mới chỉ giới thiệu khái quát về hệ chuyên gia. Để nắm được các kỹ thuật phát triển một hệ chuyên gia, bạn đọc có thể tham khảo các tài liệu chuyên khảo, chẳng hạn [11].

CHƯƠNG 8
LOGIC KHÔNG ĐƠN ĐIỆU

Lập luận mà chúng ta đã xét trong logic vị từ cấp một là lập luận đơn điệu theo nghĩa rằng, một kết luận được suy ra từ một CSTT vẫn vẫn còn đúng khi ta bổ sung vào CSTT các tiền đề mới. Trong chương này chúng ta sẽ tiếp tục nghiên cứu các kỹ thuật lập luận dựa các trên giả thiết mà chúng ta đưa ra. Các lập luận này mang tính không đơn điệu theo nghĩa rằng, khi chúng ta bổ sung vào CSTT các thông tin mới, thì các kết luận mà chúng ta đã suy ra trước đó cần phải xem xét lại, chúng có thể không còn phù hợp nữa. Chúng ta sẽ lần lượt xét các kỹ thuật sau:

Lập luận mặc định.

Giả thiết thế giới đóng.

Kỹ thuật bổ sung vị từ.

Kỹ thuật hạn chế phạm vi.

8. 1. LẬP LUẬN CÓ THỂ XEM XÉT LẠI VÀ LOGIC KHÔNG ĐƠN ĐIỆU

Con người có khả năng tiến hành suy diễn một cách sáng suốt, đưa ra các kết luận hợp lý ngay cả trong trường hợp thiếu thông tin hoặc có những thông tin không đầy đủ. Các suy diễn này thường mang tính chất có thể chấp nhận được, nó cần phải được xem xét lại khi chúng ta biết thêm các thông tin mới. Chúng ta không thể cho rằng một kết luận được suy ra trong hoàn cảnh thiếu thông tin là không có sai lầm. Khi chúng ta bổ sung những thông tin mới vào cơ sở tri thức thì một kết luận được suy ra từ cơ sở tri thức cũ có thể không còn phù hợp nữa.

Xét ví dụ đơn giản sau. Giả sử chúng ta biết rằng, "phần lớn chim biết bay" và "Kiti là chim". Khi đó ta có thể suy ra rằng "Kiti biết bay". Suy diễn kiểu này xem ra có thể chấp nhận được. Song ta không thể coi kết luận "Kiti biết bay" là tuyệt đối đúng, bởi vì khi đưa ra kết luận này ta đã không tính đến các trường hợp ngoại lệ. Do đó nó cần phải được xem xét lại khi ta biết thêm các thông tin mới. Chẳng hạn, sau đó ta biết thêm rằng, "Kiti là đà điểu" và "đà điểu là loài chim không biết bay". Rõ ràng lúc này ta không thể suy ra kết luận "Kiti biết bay".

Để phục vụ cho việc xây dựng các hệ thống thông minh có khả năng suy diễn trong môi trường thiếu thông tin hoặc có các thông tin không đầy đủ, các thông tin có tính tiến triển, chúng ta cần nghiên cứu các kỹ thuật lập luận dựa trên các giả thiết. Các lập luận này mang tính không đơn điệu. Trong chương này chúng ta sẽ nghiên cứu các hệ logic được cung cấp các kỹ thuật lập luận không đơn điệu. Các logic này được gọi chung là các ***logic không đơn điệu*** (nonmonotonic logic). Sở dĩ có tên gọi là logic không đơn điệu là vì, số các kết luận mà các hệ này cho phép suy ra có thể giảm khi số các tiên đề tăng (cơ sở tri thức được bổ sung thêm các tiên đề mới).

Trong chương này chúng ta sẽ đề cập đến các vấn đề sau đây:

- ***Logic mặc định*** (default logic hoặc logic for default resonning). Trong logic này, trên nền tảng logic vị từ cấp một, người ta đưa vào các ***luật suy diễn mặc định*** (default rule of inference). Các luật suy diễn này cho phép ta biểu diễn các quy luật có ngoại lệ mà không cần phải liệt kê ra các trường hợp ngoại lệ.
- ***Giả thiết thế giới đóng*** (closed-world assumption). Giả thiết thế giới đóng là quy ước cho rằng, với cơ sở tri thức đã cho tất cả khẳng định sơ cấp không thể chứng minh đực bởi các thông tin trong cơ sở tri thức được xem là sai. Do đó các kết luận được suy ra dưới giả thiết thế giới dóng cần phải được xem xét lại khi có những thông tin mới được bổ sung vào cơ sở tri thức. Giả thiết thế giới đóng được đưa ra để làm "cơ sở pháp lý" cho những kết luận được suy ra trong môi trường thiếu thông tin.
- ***Bổ sung vị từ*** (completion of predicat). Kỹ thuật bổ sung vị từ cho phép ta xem các điều kiện cần và đủ cho vị từ được thoả mãn như là các điều kiện cần và đủ cho vị từ đó được thoả mãn. Chúng ta biểu diễn các điều kiện cần và đủ cho một vị từ dưới dạng một tiên đề. Đương nhiên là tiên đề cần phải được viết lại khi có những điều kiện đủ mới được bổ sung vào cơ sở tri thức và do đó các kết luận được suy ra cũng cần phải xem xét lại.
- ***Hạn chế phạm vi*** (circumscruption). Hạn chế phạm vi là kỹ thuật đưa thêm vào cơ sở tri thức một số tiên đề đặc biệt. Các tiên đề này cho phape ta hạn chế phạm vi kiểm tra sự đúng đắn của một số vị từ trong phạm vi các thông tin chứa trong cơ sở tri thức. Vì các tiên đề hạn chế phạm vi kiểm tra sự đúng đắn của các vị từ phụ thuộc vào nội dung của cơ sở tri thức, do đó nó cần những thay đổi thích hợp khi có những thông tin mới được bổ sung vào cơ sở tri thức.

Khi chúng ta xây dựng một hệ thông minh trong một lĩnh vực áp dụng, việc thu thập đầy đủ thông tin để đưa vào cơ sở tri thức là hết sức khó khăn và trong nhiều trường hợp là không thể thực hiện được. Nói cách khác, các hệ thường xuyên phải làm việc trong môi trường thiếu thông tin. Những thông tin mà ta biết được nhiều khi chỉ là các thông tin không đầy đủ. Vì vậy, việc nghiên cứu các kỹ thuật lập luận trong môi trường thiếu thông tin hoặc chứa các thông tin không đầy đủ là một lĩnh vực nghiên cứu quan trọng trong trí tuệ nhân tạo.

8. 2. ĐẶC ĐIỂM CỦA LOGIC KHÔNG ĐƠN ĐIỆU

Trong mục này chúng ta sẽ phân tích, so sánh logic cổ điển (logic vị từ cấp một) và logic không đơn điệu để đưa ra một số đặc điểm của logic không đơn điệu. Trước hết chúng ta hãy xét một số đặc điểm của lập luận trong logic cổ điển.

LẬP LUẬN VỮNG CHẮC TRONG LOGIC CỔ ĐIỂN

Một hệ hình thức (một lý thuyết hình thức) trong logic cổ điển (logic vị từ cấp một) gồm một tập tiên đề và một tập các luật suy diễn. Một hệ hình thức như thế cho phép ta suy ra các kết luận từ các tiên đề (bằng cách áp dụng một dãy các luật suy diễn). Chúng ta sử dụng ký hiệu $A \models q$ để nói rằng kết luận q được suy ra từ tập tiên đề A. Trong lôgic cổ điển, quan hệ $\models$ có các tính chất sau đây:

- Tính phản xạ:

$\{p_1, ..., p_n, q\} \models q;$

- Tính đơn điệu:

Nếu $\{p_1, ..., p_n\} \models q$ *thì* $\{p_1, ..., p_n, r\} \models q;$

- Tính bắc cầu:

Nếu $\{p_1, ..., p_n\} \models r$ *và* $\{p_1, ..., p_n, r\} \models q$ *thì*$\{p_1, ..., p_n\} \models q;$

Trong các biểu thức trên, $p_1, ..., p_n, q, r$ là các công thức trong logic vị từ.

Chúng ta cần quan tâm tới tính chất đơn điệu. tính chất này nói rằng, nếu một kết luận q được suy ra từ một tập tiên đề, thì khi ta bổ sung vào tập tiên đề đó các tiên đề mới, ta vẫn suy ra được kết luận q. Sau này ta sẽ thấy rằng tính chất này của logic cổ điển không còn đúng trong logic đơn điệu.

Chúng ta nêu ra một đặc điểm quan trọng khác của logic cổ điển. Một hệ hình thức trong logic cổ điển cho phép ta suy ra các kết luận là hệ quả logic của các tiên đề:

Nếu A|= q thì q là hệ quả logic của tập tiên đề A.

Nhớ lại khái niệm hệ quả logic đã đưa ra trong chương 7, khẳng trên có nghĩa là, trong logic cổ điển, nếu A |= q thì bất kỳ mô hình nào của tập tiên đề A cũng là mô hình của kết luận q. Do đó lập luận trong logic cổ điển là lập luận vững chắc, nó cho phép ta suy ra các kết luận đúng trong bất kỳ minh họa nào mà các tiên đè được thoả mãn.

ĐẶC ĐIỂM CỦA LOGIC KHÔNG ĐƠN ĐIỆU

Mục tiêu của logic không đơn điệu là xây dựng các hệ hình thức mô hình hoá các lập luận có thể xem xét lại, tức là các lập luận không vững chắc như trong logic cổ điển. Chúng ta sẽ nêu ra một số đặc điểm quan trọng của logic không đơn điệu.

- Logic không đơn điệu chỉ cho phép ta suy ra các kết luận chấp nhận được. Về mặt ngữ nghĩa, điều đó có nghĩa là, trong logic không đơn điệu, nếu ta suy ra kết luận p từ tập tiên đề A, thì p là phù hợp với tập tiên đề A theo nghĩa rằng, p đúng trong ít nhất một mô hình của tập tiên đề A, chứ không phải p là hệ quả logic của tập tiên đề A như trong logic cổ điển. Chẳng hạn từ hai tiên đề "phần lớn chim biết bay" và "Kiti là chim", trong logic không đơn điệu, ta có thể suy ra "Kiti biết bay". Kết luận này là đúng trong các minh hoạ mà Kiti là chim bồ câu, chim sẻ,…Song kết luận "Kiti biết bay" không đúng trong mọi mô hình của hai tiên đề đã nêu. Chẳng hạn trong các minh hoạ mà Kiti là chim cánh cụt hoặc Kiti là đà điểu.
- Một câu hỏi đương nhiên là, các kết luận được suy ra phải phù hợp với nhau. Khi ta đã suy ra một kết luận thì sau đó ta không được phép suy ra một kết luận khác không phù hợp với kết luận đã được suy ra. Chẳng hạn, nếu ta suy ra "Kiti biết bay" thì sau đó ta bị cấm suy ra kết luận "Kiti không biết bay". Do đó trình tự tiến hành suy luận sẽ xác định một tập các kết luận nào đó.
- Logic không đơn điệu nhằm xây dựng các hệ suy diễn trong môi trường thiếu thông tin hoặc chứa các thông tin không đầy đủ, các thông tin có tính tiến triển. Các kết luận được suy ra trong logic không đơn điệu chỉ có tính chấp nhận được, chỉ phù hợp với các

tiên đề. Vì vậy khi ta bổ sung vào tập tiên đề các thông tin mới, một kết luận được suy ra từ tập tiên đề được bổ sung, do đó nó có thể không được suy ra từ tập tiên đề được bổ sung. Nói cách khác, khi ta tăng tập tiên đề thì tập các kết luận được suy ra có thể giảm. Tính chất đơn điệu trong logic cổ điển không còn đúng nữa. Chẳng hạn, từ hai tiên đề "phần lớn chim biết bay"và "Kiti là chim" ta có thể suy ra "Kiti biết bay". Song nếu ta bổ sung vào các tiên đề mới "Kiti là đà điểu"và "đà điểu không biết bay", ta không thể kết luận "Kiti biết bay".

8.3.LOGIC MẶC ĐỊNH

Trong hoàn cảnh cơ sở tri thức có chứa những thông tin không đầy đủ, chúng ta có thể đưa ra các kết luận chấp nhận được, các kết luận này phù hợp với những thông tin mà chúng ta đã biết. chẳng hạn, nếu biết "Kiti là chim" và "phần lớn chim biết bay", ta có thể suy ra "Kiti biết bay", nếu với sự hiểu biết hiện tại của chúng ta , không có gì ngăn cấm ta suy ra kết luận đó. Loại lập luận này gọi là lập luận mặc định (default reasoning).

Logic mặc định đã được đưa ra và phát triển bởi Reiter (1980). Logic mặc định cho phép ta hình thức hoá lập luận mặc định nhờ đưa vào các luật suy diễn đặc biệt: luật mặc định (default rule of inference). Dạng tổng quát của luật mặc định như sau:

$$\frac{\alpha : M\beta}{\gamma}$$

Trong đó α. β.γ là các công thức trong logic vị từ cấp một. Công thức α được gọi là tiền điều kiện của mặc định, β là minh chứng của mặc định, còn γ là kết luận cuả mặc định. M là một ký hiệu đặc biệt. Mβ có nghĩa là công thức β phù hợp với các tri thức khác đã biết. Nói một cách khác, ta xem Mβ là đúng nếu là các tri thức đã biết không thể suy ra được $\neg\beta$.

Ý nghĩa trực quan của luật mặc định như sau: Nếu α đúng và β phù hợp với tất cả các tri thức mà chúng ta đã biết, khi đó ta có thể suy ra γ.

Các luật mặc định cho phép ta biểu diễn các quy luật tổng quát, nó đúng trong phần lớn các trường hợp, song có những trường hợp ngoại lệ. Đó là các quy luật mà khi phát biểu trong ngôn ngữ tự nhiên ta vẫn hay dùng cụm từ "nói chung", "phần lớn",…Chẳng hạn, phát biểu "nói chung chim biết bay" được biểu diễn dưới dạng luật mặc định sau:

$$\frac{\text{bird(x)} : \text{M flies(x)}}{\text{Flies(x)}}$$

Luật này có ý nghĩa trực quan như sau: Nếu "x là chim" và nếu "x biết bay" không mâu thuẫn với những thông tin mà chúng ta đã biết, ta có thể rút ra kết luận "x biết bay".

Một luật mặc định mà kết luận của luật trùng với minh chứng của luật được gọi là ***luật mặc định chuẩn tắc***. Như vậy một luật mặc định chuẩn tắc có dạng:

$$\frac{\alpha : M\beta}{\gamma}$$

Sau đây chúng ta sẽ biểu diễn một số thông tin không dầy đủ dưới dạng các luật mặc định chuẩn tắc. Đầu tiên chúng ta sẽ đưa ra các phát biểu trong ngôn ngữ tự nhiên, sau đó đưa luật mặc định biểu diễn phát biểu đó.

- " Đa số các giáo sư là tiến sĩ"

$$\frac{\text{professor(x)} : \text{M doctor(x)}}{\text{doctor(x)}}$$

- "Nói chung các bà vợ sống với chồng"

$$\frac{\text{husbandof(x, y) A livein(y,z)} : \text{M livein (y,z)}}{\text{livein(y,z)}}$$

Các bác sĩ, kỹ sư trong việc chẩn đoán bệnh hoặc chẩn đoán hỏng hóc của máy móc thường sử dụng một dạng luật mặc định chuẩn tắc đặc biệt: **luật chẩn đoán** (abductive ru le of inference). Đó là luật mặc định chẩn tắc có dạng:

$$\frac{(\alpha \Rightarrow \beta) A\beta : M\alpha}{\alpha}$$

Luật này nói rằng, nếu $\alpha \quad \beta$ là đúng và β là đúng, ta có thể suy ra α nếu như α phù hợp với các tri thức đã biết. Xét ví dụ sau: chúng ta biết rằng, nếu xe hết xăng thì xe không khởi động được, tức là ta có:

$$\neg\text{havegas(x)} \Rightarrow \quad \neg\text{start(x)}$$

Bây giờ xe máy của bạn không khởi động được, bạn có thể kết luận rằng xe máy hết xăng nếu như không có lý do gì ngăn cấm bạn rút ra kết

luận đó (chẳng hạn, xe bạn là xe mới mua của hãng sản xuất có uy tín). Suy luận của bạn đã được dựa trên luật định chẩn đoán sau:

$$\frac{(\neg \text{havegas}(x) \Rightarrow \neg \text{start}(x))\ A\ \neg \text{start}(x) : M(\neg \text{havegas}(x))}{\neg \text{havegas}(x)}$$

Giả sử D là một tập nào đó các luật mặc định và W là một tập nào đó các công thức (các tiên đề)trong logic vị từ cấp một. Khi đó (D, W) được gọi là **hệ hình thức với các luật mặc định**. Tập tất cả các công thức có thể suy ra từ tập tiên đề W (bằng cách áp dụng các luật suy diễn cổ điển và cácluật mặc định trong D) sẽ được gọi là một **mở rộng** của (D,W). và sẽ được ký hiệu là E (D,W). Một điều đặc biệt quan trọng là mở rộng của hệ với các luật mặc định không phải là duy nhất. Một hệ với các luật mặc định (D,W) có thể có nhiều mở rộng. Chẳng hạn, xét hệ (D,W) trong đó tập tiên đề W rỗng và Dgồm hai luật mặc định:

$$\frac{:Mp}{\neg q} \quad \text{và} \quad \frac{:Mq}{\neg p}$$

Hệ này có một mở rộng chứa $\neg q$ nhưng không chứa $\neg p$ và một mở rộng khác chứa p nhưng không chứa $\neg q$.

Sau này chúng ta chỉ quan tâm tới các hệ với các luật mặc định chuẩn tắc.

Một vấn đề quan trọng là, có thể xây dựng được một thủ tục chứng minh trong các hệ mặc định chuẩn tắc? một cách chính xác hơn, cho trước một hệ mặc định chuẩn tắc (D,W) và một công thức A, có tồn tại không một kỹ thuật minh chứng cho phép ta xác định A thuộc hay không thuộc một mở rộng của (D,W).

Sau đây chúng ta sẽ trình bày thủ tục chứng minh được đưa ra bởi Reiter.

Chúng ta ký hiệu Precond(D_i) (i=0, 1, …, k) là tập hợp công thức mà chúng là tiền điều kiện của các luật mặc định trong D_i; Conc (D_i) là tập các công thức mà chúng là các kết luận của các luật mặc định trong D_i.

Thủ tục chứng minh được thực hiện qua các bước sau:

- Trước hết ta chứng minh (bằng bác bỏ) rằng A được suy ra từ tập tiên đề W và từ các kết luận của các luật mặc định trong một tập con Do nào đó của D. Tức là chứng minh rằng:

$$.1.1 \quad W \cup Conc\ (D_0)\ |= A$$

- Sau đó chúng ta chứng minh (bằng bác bỏ) rằng tập các tiền điều kiện của các luật mặc định trong Do được suy ra từ tập tiên đề W và từ các kết luận của các luật mặc định trong một tập con D1 nào đó của D.

$$.1.2 \quad W \cup Conc\ (D_1)\ |= Precond(D_0)$$

Tiếp tục quá trình trên cho tới khi ta nhận được tập rỗng và lúc này ta chứng minh tập các tiền điều kiện của các luật mặc định trong Dk-1 được suy ra từ tập tiên đề W.

$$W |= Precond\ (D_{k-1})$$

- Cuối cùng ta chứng minh rằng, tập công thức bao gồm các tiên đề trong W và các công thức trong Conc (Di) (i= 0,1…, k-1) là htoả mãn được (không chứa mâu thuẫn).

Để thấy rõ thủ tục được thực hiện như thế nào, chúng ta xét ví dụ sau: Giả sử (D,W) là một hệ với các luật mặc định chuẩn tắc trong logic mệnh đề, tập D gồm các luật mặc định sau:

$$\frac{A \ : MB}{B} \qquad (d_1)$$

$$\frac{F \ : M \neg C}{\neg C} \qquad (d_2)$$

$$\frac{B \wedge F \ : MG}{G} \qquad (d_3)$$

Tập tiên đề W gồm các công thức sau:

$$A \qquad (1)$$

$$G \vee C \qquad (2)$$

$$F \qquad (3)$$

Chúng ta cần chứng minh rằng, G thuộc một mở rộng của hệ (D,W), nói cách khác, G suy ra được từ tập tiên đề (1) – (3) và các luật mặc định (d_1) – (d_3).

Lấy phủ định của công thức cần chứng minh, ta có ⌉G. Từ ⌉G và tiên đề (2), áp dụng luật giải ta suy ra công thức C. Từ C và ⌉C (kết luận của luật mặc định d_2)) ta suy ra công thức rỗng. Như vậy, D_0 chỉ gồm một luật mặc định (d_2). Tiên đề điều kiện của luật mặc định (d_2) là F (tiên đề (3)). Do đó, tập $D_1 = \varnothing$. Chúng ta cần phải chứng minh tập tiên đề (1) – (3) được bổ sung thêm kết luận của mặc định (d_2) là thoả mãn đực. Hiển nhiên là nó thoả mãn được.

Chúng ta đưa ra một chứng minh khác. Từ ⌉G và G (kết luận của mặc định (d_3)) ta suy ra công thức rỗng. Như vậy D_0 chỉ gồm một mặc định (d3). Tiền điều kiện của mặc định (d_3) là $B \wedge F$. Lấy phủ định của công thức này ta có:

$$⌉(B \wedge F) = ⌉B \vee ⌉F.$$

Từ $⌉B \vee ⌉F$ và tiên đề (3), ta suy ra ⌉B. Từ ⌉B và kết luận của mặc định (d_1), ta suy ra công thức rỗng. Như vậy, D_1 chỉ gồm một mặc định (d_1). Tiền điều kiện của (d_1) là A (tiên đề (1)). Do đó $D_2 = \varnothing$.Cuối cùng chúng ta thấy rằng, nếu bổ sung các kết luận của mặc định (d_3) và (d_1) vào tập tiên đề ta vẫn nhận được một tập công thức thoả mãn được.

8. 4. GIẢ THIẾT THẾ GIỚI ĐÓNG

Việc mô tả đầy đủ thế giới của một vấn đề là cực kỳ khó khăn và trong nhiều trường hợp là không thể thực hiện được. Chẳng hạn, chúng ta có thể liệt kê tất cả các đồ vật trong một căn phòng, song không thể liệt kê tất cả các đồ vật không có trong căn phòng vì các đồ vật đó là vô hạn. Tương tự, ta có thể liệt kê ra tất cả các tri thức đúng (trong thế giới của một vấn đề nào đó), song liệt kê ra tất cả các tri thức không đúng là không thể thực hiện được.

Xét một cơ sở dữ liệu chứa danh sách các nhân viên làm việc trong một công ty. Danh sách chứa họ tên, tuổi và công việc của mỗi người. Giả sử ta muốn biết (Lê Kim Anh, 24 tuổi, thư ký) có làm việc trong công ty đó không? Giả sử cơ sở dữ liệu đó không chứa bản ghi (Lê Kim Anh, 24 tuổi, thư ký). Các thông tin trong cơ sở dữ liệu không cho phép ta xác nhận hoặc

bác bỏ câu "Lê Kim Anh, 24 tuổi, thư ký là nhân viên của công ty". Song thực tế ta vẫn thường đưa ra câu trả lời phủ định: Lê Kim Anh, 24 tuổi, thư ký không làm việc trong công ty đó. Tức là ta bác bỏ mọi sự việc không được ghi nhớ trong cơ sở dữ liệu.

Giả thiết thế giới đóng (closed world assumption) là quy ước cho rằng tất cả các tri thức chưa xác định được là đúng hay sai (với cơ sở tri thức đã cho) đều được xem là sai. Giả thiết thế giới đóng được sử dụng để chứng minh chứng cho các kết luận được suy ra trong các trường hợp thiếu thông tin (cũng cần nhấn mạnh rằng, các hệ thường xuyên phải làm viêc trong môi trường thiếu thông tin). Khi chúng ta sử dụng giả thiết thế giới đóng thì lập luận là không đơn điệu, bởi vì khi ta bổ sung các thông tin mới vào cơ sở tri thức thì có thể một khẳng định trước kia được xem là đúng bây giờ lại trở thành sai.

Sau đây chúng ta đưa ra định nghĩa chính xác hơn về giả thiết thế đóng. Giả sử Δ là một tập các công thức trong logic vị từ (tập tiên đề). Chúng ta gọi tập tất cả các hệ quả logic của Δ là một ***lý thuyết với tập tiên đề Δ*** và được ký hiệu là $T(\Delta)$.

Chúng ta ký hiệu Δ' là tập tất cả các literal âm cụ thể $\neg P(t_1,\ldots,t_n)$ mà $P(t_1,\ldots,t_n)$ không thuộc $T(\Delta)$ (trong đó P là vị từ của n biến và $t_1, \ldots, t_n$ là các hạng thức không chứa biến). Dưới giả thiết thế giới không đóng ta xây dựng lý thuyết mới $T(\Delta \cup \Delta')$. Ta ký hiệu lý thuyết mới này là $CWT(\Delta)$. Như vậy lý thuyết mở rộng $CWT(\Delta)$ bao gồm tất cả các hệ quả logic của các tiên đề trong Δ và tiên đề trong Δ. Do đó muốn biết công thức A có suy ra được từ cơ sở tri thức Δ dưới giả thiết thế giới đóng hay không,chúng ta cần kiểm tra xem A có thuộc $CWT(\Delta)$ hay không.

Cần chú ý rằng, việc áp dụng giả thiết thế giới đóng cho một cơ sở tri thức Δ có thể dẫn tới lý thuyết $CWT(\Delta)$ không phù hợp (chứa các thông tin mâu thuẫn).

Ví dụ. Giả sử Δ gồm các câu:

$$\text{Male}(x) \vee \text{Female}(x)$$
$$\text{Teachear}(An)$$

Từ cơ sở tri thức này ta không suy ra được Male (An) và Female (An). Do đó Δ' gồm các literal âm cụ thể

$$\neg \text{Male}(An)$$

⌉Female(An)

Nhưng khi đó lý thuyết CWT (Δ) sẽ không phù hợp vì từ các công thức Male(x) ∨ Female(x) và ⌉Male (An), ta suy ra Female (An). Điều này mâu thuẫn với ⌉Female(An).

Người ta đã tìm ra các điều kiện cho cơ sở tri thức Δ để áp dụng giả thiết thế giới đóng ta nhận được lý thuyết mở rộng CWT (Δ) phù hợp. Cụ thể là, nếu cơ sở tri thức Δ chỉ gồm cáccau Horn (và thoả được) thì lý thuyết mở rộng dưới giả thiết thế giới đóng CWT (Δ) sẽ phù hợp.

Giả thế giới đóng thường được sử dụng kết hợp với các giả thiết khác; giả thiết miền đóng (domain-closure assumption) và giả thiết tên duy nhất (unique-names assumption).

GIẢ THIẾT MIỀN ĐÓNG

Giả thiết miền đónglà sự hạn chế rằng, miền đối tượng chỉ bao gồm các đối tượng được xác định bởi các ký hiệu hằng đối tượng và csc ký hiệu hàm có mặt trong cơ sở tri thức. Trong trường hợp đặc biệt, khi mà - không chứa các ký hiệu hàm thì giả thiết thế giới đóng có thể viết dưới dạng một tiên đề sau:

$$\forall x ((x = c_1) \vee \ldots \vee (x = c_n))$$

trong đó $c_1,\ldots, c_n$ là tất cả các ký hiệu hằng đối tượng có mặt trong Δ. Chẳng hạn, với cơ sở tri thức Δ trong ví dụ trên, áp dụng giả thiết miền đóng, chúng ta hạn chế miền minh hoạ của Δ chỉ gồm một đối tượng duy nhất {An}, bởi vì trong các công thức của Δ chỉ có một hằng đối tượng An và không có ký hiệu hàm nào.

GIẢ THIẾT TÊN DUY NHẤT

Giả thiết tên duy nhất là quy ước cho rằng, các hạng thức cụ thể (hạng thức không chứa biến) không thể chứng minh được chúng bằng nhau sẽ được xem là khác nhau. Tất cả các tên đối tượng khác nhau sẽ chỉ định các đối tượng khác nhau trong minh hoạ bất kỳ.

ÁP DỤNG GIẢ THIẾT THẾ GIỚI ĐÓNG VÀO HỆ LẬP TRÌNH LOGIC PROLOG

Như chúng ta đã nói trong mục 7.6. để tăng khả năng biểu diễn của Prolog, người ta đưa vào Prolog một vị từ xác định trước, vị từ not. Vị từ not

trong Prolog là sự cài đặt một dạng phủ định: phủ định như thất bại (negation-as-failure). Vị từ này được xác định như sau. Nếu G là một literal dương cụ thể, thì not(G) là đúng nếu chúng ta không tìm được một chứng minh G đúng. Còn nếu G là một literal dương chứa biến thì not(G) là đúng nếu với mọi giá trị của các biến ta đều không chứng minh được G là đúng. Như vậy, sự cài đặt vị từ not trong Prolog là dựa trên giả thiết thế giới đóng.

8.5. BỔ SUNG VỊ TỪ

Trong mục 8.4. chúng ta đã thảo luận về giả thiết thế giới đóng. Khi áp dụng giả thiết thế giới đóng cho cơ sở tri thức Δ, ta bổ sung vào cơ sở tri thức Δ tất cả các phủ định của mỗi công thức phân tử cụ thể mà nó không phải là hệ quả logic của Δ. Trong mục này chúng ta sẽ xét kỹ thuật bổ sung vị từ.

Giả sử chúng ta chỉ biết đối tượng obj_1 có màu đỏ, $Red(obj_1)$. Trong thực tế, chúng ta thường quan niệm rằng, tất cả các đối tượng khác với đối tượng obj_1 đều không có màu đỏ. Vấn đề đặt ra là, làm thế nào biểu diễn được điều đó. Điều kiện "x là obj_1" là điều kiện dủ để "x màu đỏ".

$$(x = obj_1) \Rightarrow Red(x)$$

Chúng ta sẽ xem điều kiện "x là obj_1" là điều kiện cần và đủ để "x màu đỏ". Với quan niệm này, chúng ta bổ sung vị từ Red bằng cách đưa vào công thức bổ sung sau:

$$\forall x\ (Red(x) \Leftrightarrow (x = obj_1))$$

Bằng cách sử dụng công thức bổ sung trên cho vị từ Red, ta có thể chứng minh được đối tượng bất kỳ khác, chẳng hạn obj_2, không có màu đỏ. Sau này khi ta biết thêm obj_3 cũng có màu đỏ, ta cần xác định lại công thức bổ sung cho vị từ Red.

$$\forall x\ (Red(x) \Leftrightarrow (x = obj_1) \vee (x = obj_3))$$

Chúng ta xét trường hợp phức tạp hơn. Giả sử ta biết đối tượng obj_1 màu đỏ và tất cả các đối tượng nằm trên bàn đều màu đỏ. Cơ sở tri thức của chúng ta chứa các câu:

$$Red(obj_1)$$
$$\forall x\ (On(x, table) \Rightarrow Red(x))$$

Như vậy, "x là obj_1" hoặc "x nằm trên bàn" là điều kiện đủ để "x màu đỏ". Chúng ta xem điều kiện trên là điều kiện cần và đủ để "x màu đỏ" và do đó chúng ta có công thức bổ sung cho vị từ Red như sau:

$$\forall x\ (Red(x) \Leftrightarrow (x = obj_1) \vee On(x, table))$$

Sau đây chúng ta sẽ trình bày chính xác hơn về kỹ thuật bổ sung vị từ. Giả sử cơ sở tri thức Δ chứa một số công thức là điều kiện đủ để xác định vị từ P. Chúng ta quy ước các công thức đó là điều kiện cần và đủ để xác định vị từ P. Chúng ta biểu diễn thoả thuận này dưới dạng một tiên đề. Tiên đề này phát biểu rằng, vị từ P được thoả mãn khi và chỉ khi một trong các điều kiện đủ cho P được thoả mãn. Chúng ta sẽ gọi tiên đề này là công thức bổ sung cho vị từ P.

Khi chúng ta bổ sung vị từ P trong cơ sở tri thức Δ, chúng ta sẽ nhận được cơ sở tri thức mới: cơ sở tri thức bổ sung C(Δ,P). C(Δ,P) nhận được từ Δ bằng cách thay các công thức là điều kiện đủ cho P bởi công thức bổ sung cho P.

Ví dụ. Xét cơ sở tri thức Δ gồm các công thức

$$\text{Bird}(x) \Rightarrow \text{Flies}(x)$$

$$\text{Flies}(\text{Mic})$$

$$\neg \text{Bird}(\text{Bin})$$

Chúng ta bổ sung vị từ Flies. Trong cơ sở tri thức trên, điều kiện đủ để một đối tượng biết bay là đối tượng đó là chim hoặc đối tượng đó là Mic.Do đó công thức bổ sung cho vị từ Flies là:

$$\forall x\ (\text{Flies}(x) \Leftrightarrow \text{Bird}(x) \vee (x = \text{Mic}))$$

Như vậy cơ sở tri thức bổ sung C (Δ,Flies) gồm các công thức:

$$(\text{Flies}(x) \Leftrightarrow \text{Bird}(x) \vee (x = \text{Mic})$$

$$\neg \text{Bird}(\text{Bin})$$

Từ cơ sở tri thức bổ sung này, ta có thể suy ra rằng ¬ Flies(Bin) (Bin không biết bay). Kết luận này cần phải được xem xét lại khi ta bổ sung các thông tin mới vào cơ sở tri thức Δ. Chẳng hạn, nếu ta thêm thông tin Flies(Bin) vào công thức bổ sung cho vị từ Flies là:

$$\forall x\ (\text{Flies}(x) \Leftrightarrow \text{Bird}(x) \vee (x = \text{Mic}) \vee (x = \text{Bin}))$$

Bây giờ ¬ Flies(Bin) không thể suy ra được từ cơ sử tri thức bổ sung ***C***(Δ, Flies).

Cần lưu ý rằng, khi ta tiến hành bổ sung các vị từ trong một cơ cở tri thức có thể dẫn tới cơ sở tri thức bổ sung không còn phù hợp nữa (tức là chứa các thông tin mâu thuẫn). Ta cần hạn chế sự áp dụng kỹ thuật bổ sung vị từ, chỉ bổ sung các vị từ sao cho cơ sở tri thức bổ sung không chứa mâu thuẫn. Chẳng hạn, ta có thể bổ sung các vị từ P mà P là **đơn độc** trong cơ sở tri thức Δ. (Vị từ P được xem là đơn độc trong cơ sở tri thức Δ nếu một câu trong Δ chứa một xuất hiện dương của P thì đó là sự xuất hiện duy nhất của P trong câu đó).

8.6. HẠN CHẾ PHẠM VI

Chúng ta đã nghiên cứu kỹ thuật dựa trên giả thiêt thế giới đóng và kỹ thuật bổ sung vị từ. Sử dụng các kỹ thuật này chúng ta sẽ bổ sung vào cơ sở tri thức đã cho các tiên đề mới nhằm "bù đắp" cho sự thiếu thông tin. Trong mục này chúng ta sẽ trình bày một kỹ thuật khác: kỹ thuật hạn chế phạm vi (circumscription). Kỹ thuật hạn chế phạm vi là sự tổng quát hoá của kỹ thuật bổ sung vị từ và kỹ thuật dựa trên quy ước về thế giới đóng.

Chúng ta xét luật sau đây:

$$\text{Bird}(x) \wedge \neg \text{Anormal}(x) \Rightarrow \text{Flies}(x)$$

Luật này nói rằng, nếu x là một con chim bình thường thì x biết bay. Bây giờ, giả sử Bird (Bin), khi đó ta có thể chứng minh được Flies (Bin) nếu chúng ta chứng minh được Bin là con chim bình thường, tức là chứng minh được ¬Anormal(Bin).Chúng ta xem một con chim là bình thường trừ khi ta chứng minh được nó là không bình thường. Để xác định một con chim có là không bình thường (Anormal) hay không, chúng ta hạn chế phạm vi kiểm tra vị từ Anormal trong giới hạn các thông tin mà chúng ta đã biết. Chẳng hạn, nếu chúng ta biết chim cánh cụt và chim đà điểu không biết bay, ta có thể hạn chế kiểm tra vị từ Anormal trong phạm vi các con chim cánh cụt và chim đà điểu. Tức là ta xem rằng, một con chim là không bình thường nếu và chỉ nếu nó là chim cánh cụt hoặc chim đà điểu.

$$\forall x\ (\text{Anormal}(x) \Leftrightarrow \text{Pengium}(x) \vee \text{Ostrich}(x))$$

Hạn chế một phạm vi một vị từ trong một cơ sở tri thức Δ là hạn chế phạm vi kiểm tra sự đúng đắn của vị từ đó. Kỹ thuật hạn chế phạm vi cho phép ta tối thiểu hoá miền kiểm tra một số vị từ trong một cơ sở tri thức.

Giả sử P là một tập nào đó các vị từ có mặt trong cơ sở tri thức Δ, các vị từ trong P là các vị từ được hạn chế phạm vi. Khi chúng ta hạn chế phạm vi kiểm tra sự đúng đắn của các vị từ trong P, chúng ta sẽ chỉ quan tâm tới các ***mô hình tối thiểu*** (minimal model) đối với P. Để xác định khái niệm mô hình tối thiểu đối với P, chúng ta dựa vào quan hệ "hẹp hơn" giữa các mô hình của cơ sở tri thức Δ. Giả sử M_1và M_2 là hai mô hình của cơ sở tri thức Δ, ta nói rằng mô hình M_1 hẹp hơn mô hình M_2 nếu:

- M_1 và M_2 cho cùng một minh hoạ đối với tất cả các ký hiệu hàm và tất cả các ký hiệu vị từ khác với ký hiệu vị từ trong p.
- Với mỗi vị từ P trong P và với mọi đối tượng *x* mà P(x) đúng trong M_1 thì P(x) cũng đúng trong M_2.

Với sự hạn chế phạm vi các vị từ trong P thuộc cơ sở tri thức Δ, chúng ta chỉ xét đến các mô hình tối thiểu đối với P của cơ sở tri thức Δ. Chúng ta xem rằng, một công thức suy ra được từ cơ sở tri thức Δ nếu nó thoả được trong tất cả các mô hình tối thiểu của Δ.

Trong thực tế, khi phát triển một cơ sở tri thức dựa trên kỹ thuật hạn chế phạm vi các vị từ, chúng ta cần giải quyết các vấn đề sau:

- Chọn các vị từ cần được hạn chế phạm vi?
- Biểu diễn sự hạn chế thông qua các vị từ nào?

Chúng ta có thể giải quyết vấn đề chọn vị từ để hạn chế phạm vi bằng giải pháp đơn giản sau. Chúng ta biểu diễn các quy luật chung nhưng có ngoại lệ bởi các công thức. các trường hợp ngoại lệ được biểu diễn bởi vị từ một biến Anormal(x). Các vị từ Anormal sẽ là các vị từ cần được hạn chế phạm vi.

Ví dụ. Giả sử chúng ta có các tri thức sau:

- Một đối tượng bình thường không biết bay
- Một con chim là một đối tượng không bình thường
- một con chim bình thường biết bay
- Đà điểu là chim không biết bay
- Chúng ta biểu diễn các tri thức trên bởi các công thức sau:

$$\text{Object}(x) \wedge \neg \text{Anormal}_1(x) \Rightarrow \neg \text{Flies}(x)$$

$$\text{Bird}(x) \Rightarrow \text{Object}(x) \wedge \text{anormal}_2(x)$$

$$\text{Bird}(x) \wedge \neg \text{Anormal}_1(x) \Rightarrow \neg \text{Flies}(x)$$

$$\text{Ostrich}(x) \Rightarrow \text{Bird}(x) \wedge \neg \text{Flies}(x)$$

Trong các công thức trên, vị từ $Anormal_1(x)$ biểu diễn tính chất "x là đối tượng không bình thường", vị từ $Anormal_2(x)$ biểu diễn "x là con chim không bình thường". các câu trên tạo thành cơ sở tri thức Δ. Chúng ta cần hạn chế phạm vi cho hai vị từ $anormal_1$ và $Anormal_2$. Với cơ sở tri thức như trên, trực quan cho ta thấy rằng, cần hạn chế phạm vi cho hai vị từ như sau:

$$Anormal_1(x) \equiv Bird(x)$$

$$Anormal_2(x) \equiv Ostrich(x)$$

CHƯƠNG 9

LƯỚI NGỮ NGHĨA VÀ HỆ KHUNG

Lưới ngữ nghĩa (semantic network) là một trong các mô hình cơ bản biểu diễn tri thức, lưới ngữ nghĩa cho phép ta mô tả các đối tượng, các khái niệm (một lớp đối tượng) và các mối quan hệ giữa chúng. Trong chương này chúng ta sẽ xét khái niệm lưới ngữ nghĩa, vấn đề biểu diễn tri thức bởi lưới ngữ nghĩa và ván đề suy diễn bằng thừa kế trong lưới ngữ nghĩa.

Trong phương pháp biểu diễn tri thức bởi các **khung** (frame), các sự kiện liên quan đến một đối tượng (hoặc một khái niệm) được tập hợp lại để tạo thành một khung. Khung là một đơn vị có cấu trúc để biểu diễn tri thức. Chúng ta sẽ nghiên cứu các thành phần cấu thành các khung và vấn đề suy diễn trong các hệ khung.

9.1. NGÔN NGỮ MÔ TẢ KHÁI NIỆM

Thủ tục chứng minh định lý tổng quát (thủ tục chứng minh bác bỏ bằng luật phân giải) trong logic vị từ là thủ tục có độ phức tạp tính toán lớn. Vì vậy các nhà nghiên cứu cố gắng tìm kiếm các ngôn ngữ con đặc biệt của logic vị từ, chúng có đủ khả năng biểu diễn trong nhiều lĩnh vực áp dụng, và đặc biệt có thể thực hiện các thủ tục suy diễn hiệu quả. Trong chương 7 chúng ta đã nghiên cứu ngôn ngữ đặc biệt chỉ bao gồm các luật. Sau đây chúng ta sẽ đưa ra một ngôn ngữ đặc biệt khác: Ngôn ngữ mô tả khái niệm (concept diseription language).

Trong các ngành khoa học, khái niệm là đơn vị tri thức cơ sở. Người ta phát biểu các quy luật (quy luật về tự nhiên và quy luật về xã hội) thông qua các khái niệm đã được đưa ra. Chúng ta đưa ra một số ví dụ về khái niệm: các khái niệm về số nguyên, số nguyên tố,…trong toán học: cáckhái niệm chim, bò sát, động vật có vú,… trong động vật học.

Khi đưa ra một khái niệm mới, chúng ta thường xác định các mối quan hệ của nó với các khái niệm đã biết và nêu ra các đặc tính của nó. chẳng hạn ,người ta định nghĩa số nguyết tố là số dương chỉ chia hết cho 1 và chính nó.

Các khái niệm, chẳng hạn như “xe đạp”, “tam giác”, “chim”,… có thể xem như sư mô tả các lớp đối tượng nào đó. Chúng ta có thể xem khái niệm “xe đạp” như một lớp đối tượng dùng để đi lại, các đối tượng này có đặc tính như có hai bánh, chuyển động được nhờ sức đạp của người,…Lớp các đối

tượng được xác định bởi một khái niệm là một lớp con của lớp các đối tượng được xác định bởi khái niệm tổng quát hơn. Chẳng hạn, lớp các đối tượng hình học được xác định bởi khái niệm tổng quát hơn là "đa giác".

Ngôn ngữ mô tả khái niệm là ngôn ngữ đặc biệt cho phép mô tả các đối tượng, các lớp đối tượng, các mối quan hệ giữa chúng, các tính chất của các đối tượng và các tính chất của các lớp.

Ngôn ngữ mô tả khái niệm chỉ sử dụng ba vị từ sau: isa(x,y) (isa là viết tắt của cụm từ is a member of), ako (y,z) (ako là viết tắt của cụm từ a kind of) và feature (x,p,v). Ngữ nghĩa của các vị từ này là như sau:

- isa (x,y) có nghĩa rằng, đối tượng x là thành viên của lớp y.
- ako (y,z) có nghĩa rằng, lớp y là lớp con của lớp mẹ z.
- feature (x,p,v) có nghĩa rằng, đối tượng x (hoặc lớp x) có thuộc tính p với giá trị v.

Các cá thể và các lớp tuân theo một số quy luật tổng quát sau:

- Nếu một đối tượng là thành viên của một lớp thì nó là thành viên của tất cả các lớp mẹ của lớp này.

$$isa(x,y) \wedge ako(y,z) \Rightarrow isa(x,z)$$

- Quan hệ "là lớp con" có tính chất bắc cầu

$$ako(x,y) \wedge ako(y,z) \Rightarrow ako(x,z)$$

- Nếu một lớp có đặc tính thì các thành viên của lớp cũng có đặc tính đó.

$$isa(x,y) \wedge feature(y,p,v) \Rightarrow feature(x,p,v)$$

Cần lưu ý rằng quy luật này có thể có các ngoại lệ.

Ví dụ. Giả sử chúng ta có các câu sau đây mmo tả mmột số khái niệm và một số cá thể .

- Động vật có vú (mammal) là động vật (animal) có lông mao, có 4 chân cho sữa.
- Chim là động vật có lông vũ , có hai chân, biết bay, đẻ trứng.
- Chó là động vật có vú, có đức tính trung thành.
- Bin là chó cái, màu xám.
- Mic là chó đực, màu đen.
- Dơi (bát) là động vật có vú, có hai chân, biết bay
- Chim cánh cụt (penguin) là loài chim biết bơi, không biết bay.

Chúng ta có thể biểu diễn các mô tả về một số loài vật và con vật trên bởi các câu trên ngôn ngữ mô tả khái niệm như sau:

Ako (mammal, animal)
Feature (mammal, cover, hair)
Feature (mammal, legs, 4)
Feature (mammal, give, milk)
Ako (bird, animal)
Feature (bird, cover, feathes)
Feature (bird, legs, 2)
Feature (bird, flys, T)
Feature(bird, lays, eggs)
Ako (dog, mammal)
Feature (dog, loyal, T)
Isa (Bin, color, green)
Feature (Bin, sex, female)
Isa (Bin, dog)
Isa (Mic, dog)
Feature (Mic, color, black)
Feature (Mic, sex, male)
Ako (bat, mammal)
Feature (bat,fly, T)
Feature (bat, legs, 2)
Ako (penguin, swim,T)
Feature (penguin, fly, F)

Khi dã cho một cơ sở tri thức bao gồm các câu trong ngôn ngữ mô tả khái niệm, chúng ta cần có thủ tục suy diễn để suy ra các đặc tính mới của cá thể (hoặc của khái niệm) được hỏi. Chẳng hạn, từ các tri thức trên, ta có thể suy ra rằng, chim cánh cụt là loài chim có lông vũ, có hai chân và đẻ trứng.

Để thuận tiện cho quá trình suy diễn, chúng ta sẽ chuyển các tri thức được mô tả trong ngôn ngữ mô tả khái niệm sang dạng đồ thị đặc biệt: lưới ngữ nghĩa.

9.2. LƯỚI NGỮ NGHĨA

Trong mục này chúng ta sẽ xét lưới ngữ nghĩa như một mô hình biểu diễn tri thức. Sau đó chúng ta sẽ nghiên cứu phương pháp suy diễn bằng thừa kế trong lưới ngữ nghĩa.

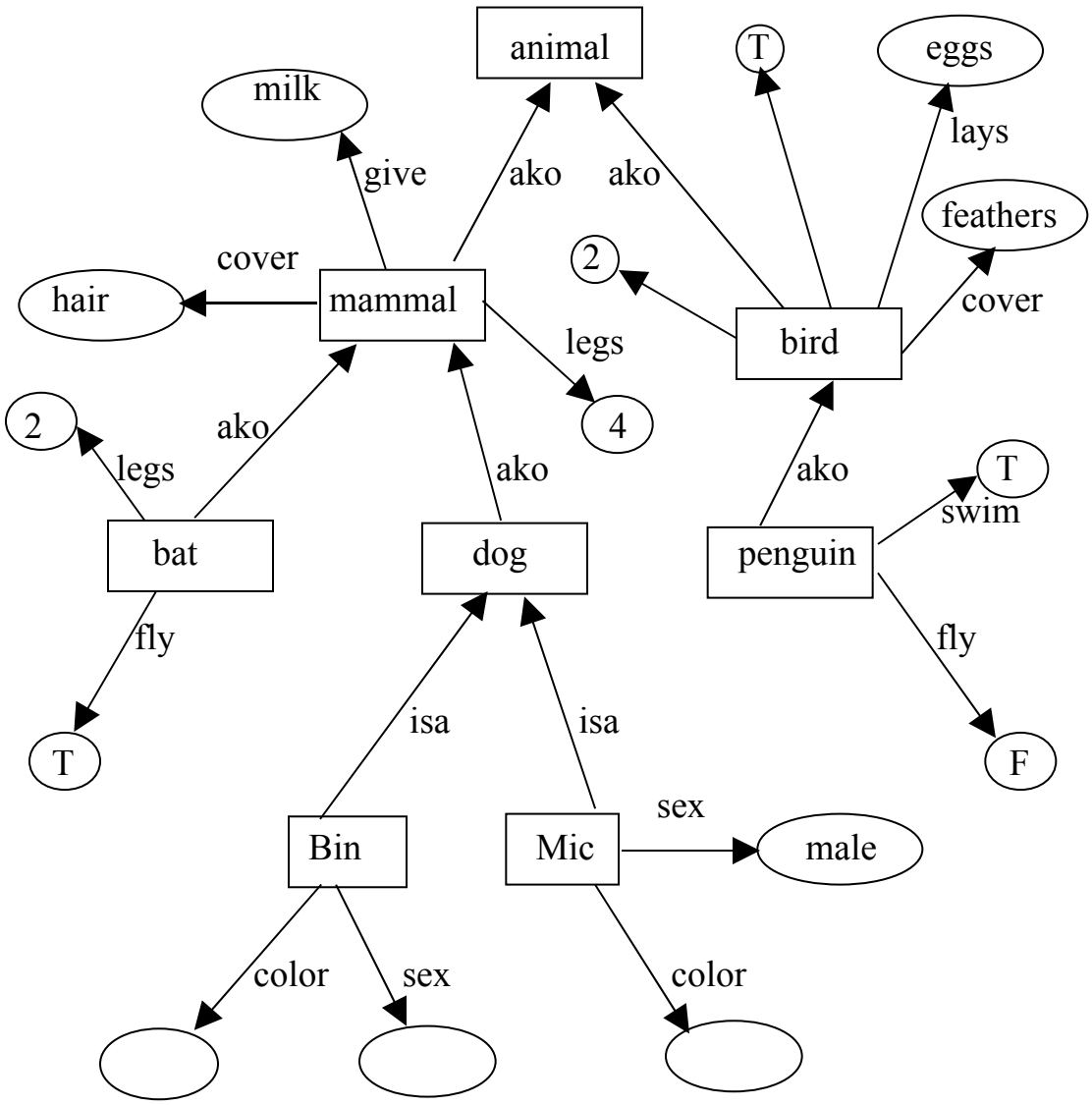
Lưới ngữ nghĩa là một đồ thị định hướng không có chu trình với các đỉnh và các cung được gắn nhãn:

Các đỉnh biểu diễn các đối tượng hoặc các lớp đối tượng (các khái niệm), hoặc các giá trị của các thuộc tính. Chúng ta quy ước các đỉnh hình

chữ nhật biểu diễn các đối tượng hoặc các lớp, còn các đỉnh elip biểu diễn các giá trị.

Các cung biểu diễn các quan hệ. Cung gắn nhãn isa đi từ đỉnh biểu diễn các đối tượng tới đỉnh biểu diễn lớp mà đối tượng là thành viên của lớp. Cung gắn nhãn ako đi từ đỉnh biểu diễn một lớp này tới đỉnh biểu diễn một lớp khác nếu lớp ứng với đỉnh nguồn của cung là lớp con của lớp ứng với đỉnh đích của cung. Các cung khác được gắn nhãn bởi tên của các thuộc tính, các cung này đi từ các đỉnh biểu diễn các đối tượng (hoặc các lớp), tới các đỉnh biểu diễn giá trị của thuộc tính.

Từ định nghĩa lưới ngữ nghĩa trên, chúng ta dễ dàng chuyển các câu trong ngôn ngữ mô tả khái niệm thành một lưới ngữ nghĩa. Hình 9.1 là lưới ngữ nghĩa biểu diễn các tri thức được mô tả bởi các câu trong ngôn ngữ mô tả khái niệm đã được đưa ra trong ví dụ ở mục trước.



green female black

***Hinh 9.1*. Một lưới ngữ nghĩa.**

Khi chúng ta biểu diễn tri thức bởi lưới ngữ nghĩa, vấn đề quan trọng được đặt ra bây giờ là làm thế nào để xác định được giá trị của các thuộc tính của các đối tượng được hỏi.

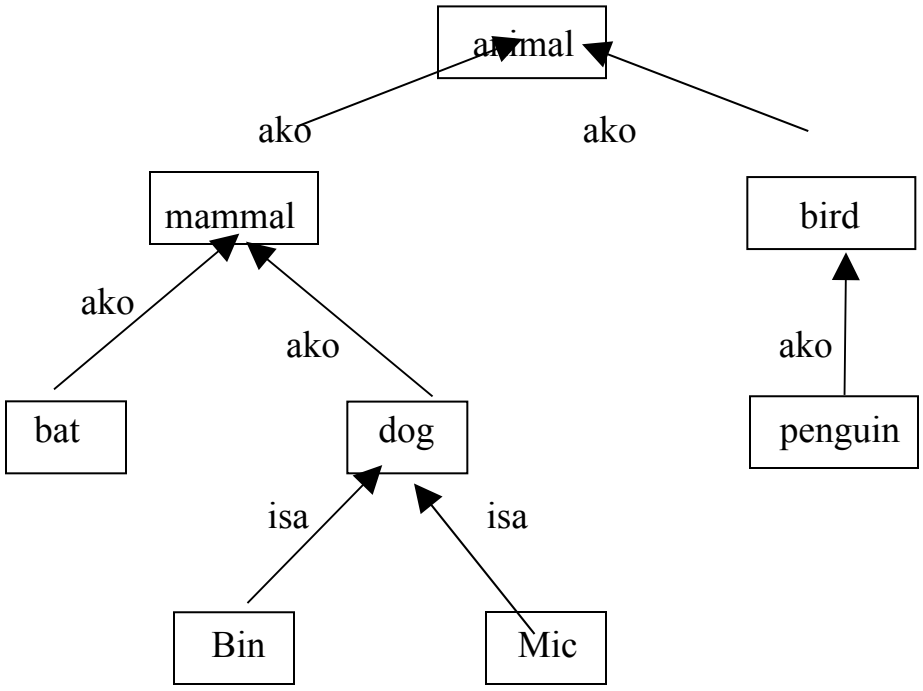
Cơ chế suy diễn trong lưới ngữ nghĩa dựa trên ***nguyên lí thừa kế.*** Chẳng hạn, sự kiện "*chó có 4 chân*" được thừa kế từ sự kiện "*động vật có vú có 4 chân*". Tương tự, nhừ nguyên lí thừa kế, ta có thể suy ra rằng "*chim cánh cụt đẻ trứng*" và "*chim cánh cụt có 2 chân*". Một cách tổng quát, ta có thể phát biểu nguyên lý thừa kế như sau : ***Nếu một cá thể là thành viên của một lớp (hoặc một lớp là lớp con của lớp khác) thì nó có thể thừa kế các giá trị của các thuộc tính của lớp đó.***

Các giá trị của các thuộc tính của một lớp được gọi là các *giá trị ngầm định* (default value). Khi nột đối tượng là thành viên của một lớp, nó có thể nhận giá trị ngầm định trong lớp đó cho các thuộc tính chưa được xác định. Chẳng hạn, giá trị ngầm định trong lớp chim của thuộc tính "*số chân*" là 2, chim cánh cụt thuộc lớp chim, do đó chim cánh cụt có 2 chân. Chúng ta có thể xác định các cá thể có các thuộc tính "*biết bay*" trong lớp chim là ***T***, nhưng chim cánh cụt là lớp con của lớp chim lại có giá trị ***F*** cho thuộc tính "*biết bay*". Trong trường hợp này, giá trị ngầm định của thuộc tính trong lớp mẹ không được lớp con (hoặc đối tượng con) thừa kế, giá trị ngầm định bị "*che lấp*" bởi giá trị thực tế đã được xác định trong lớp con (hoặc đối tượng con).

Bây giờ chúng ta sẽ xét cơ chế thừa kế trong lưới ngữ nghĩa. Trong lưới ngữ nghĩa, nếu chúng ta bỏ đi tất cả các cung biểu diễn các quan hệ thuộc tính – giá trị, chỉ giữ lại các các quan hệ *isa* và các quan hệ *ako*, chúng ta sẽ nhận được một đồ thị định hướng biểu diễn ***cấu trúc phân cấp thừa kế***. Chẳng hạn, từ lưới ngữ nghĩa trong hình 9.1, ta có cấu trúc phân cấp thừa kế trong hình 9.2.

Mục đích của suy diễn trong lưới ngữ nghĩa là: khi được hỏi về giá trị của một thuộc tính của một đối tượng (hoặc một lớp), ta cần xác định được giá trị của thuộc tính đó. Trước hết ta xét trường hợp đơn giản nhất, khi mà trong cấu trúc phân cấp thừa kế, từ một đỉnh chỉ có nhiều nhất một cung đi ra, tức là một đối tượng chỉ có thể là thành viên của một lớp duy nhất, một lớp chỉ có thể là lớp con của một lớp mẹ duy nhất.Chẳng hạn, lưới ngữ nghĩa trong hình 9.1 có các tính chất đó. Trong trường hợp này, để xác định giá trị của một thuộc tính cho một đối tượng, ta đi lên từ đỉnh ứng với đối tượng đó

theo các cung được gắn nhãn isa hoặc ako cho tới lớp mà từ đó cung đi ra được gắn nhãn bởi thuộc tính được hỏi. Chẳng hạn, từ lưới ngữ nghĩa trong hình 9.1, để tìm câu trả lời cho câu hỏi "Mic có mấy chân?", ta đi từ đỉnh Mic đến đỉnh dog rồi tới đỉnh mammal, từ đỉnh này có cung đi ra nhãn legs (thuộc tính số chân) tới đỉnh biểu diễn giá trị 4. Do đo câu trả lời là Mic có 4 chân.



***Hình 9.2*. Cấu trúc phân cấp thừa kế.**

THỪA KẾ NHIỀU LỚP

Bây giờ chúng ta xét sự thừa kế trong trường hợp tổng quát: thừa kế từ nhiều lớp (multiple inheritance).

Việc xác định giá trị của một thuộc tính cho một cá thể trong lưới ngữ nghĩa sẽ trở nên rất phức tạp khi một cá thể có thể thừa kế các giá trị ngầm định khác nhau trong các lớp khác nhau. Trường hợp này xảy ra khi trong lưới ngữ nghĩa, một cá thể có thể là thành viên của các lớp khác nhau, một lớp có thể là lớp con của các lớp mẹ khác nhau: tức là trong cấu trúc phân cấp từ một đỉnh có nhiều cung đi ra.

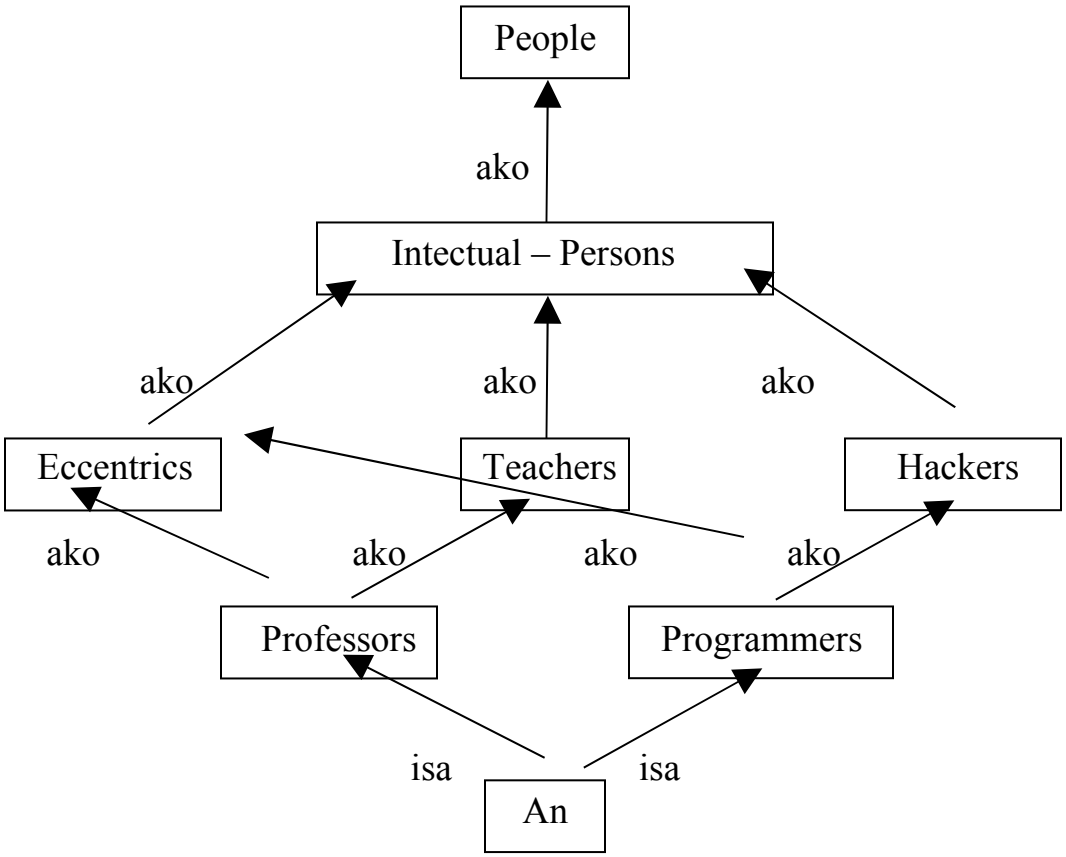
Ví dụ. Giả sử chúng ta có các lớp sau

- Lớp giáo sư (Professors)
- Lớp giáo viên (Teachers)
- Lớp người lập dị (Eccentrics)

- Lớp Hackers
- Lớp lập trình viên (Programmers)
- Lớp trí thức (Intellectual-Persons)
- Lớp mọi người (People)

Ông An vừa là giáo sư, vừa là lập trình viên.

Giả sử chúng ta có cấu trúc phân cấp thừa kế các lớp đã liệt kê như trong hình 9.3.



***Hình 9.3*. Cấu trúc phân cấp thừa kế bội.**

Giả sử cho biết người lập dị có đặc tính đãng trí, còn người Hacker có đặc tính không đãng trí. Giả sử người ta chưa biết ông An có đãng trí hay không. Chúng ta cần tìm câu trả lời. Ở đây chúng ta có hai lớp có thể cung cấp giá trị ngầm định cho thuộc tính "đãng trí" của ông An. Đó là lớp người lập dị và lớp người Hacker vì ông An là thành viên của cả hai lớp này. Chúng ta phải quyết định lớp nào cung cấp giá trị ngầm định cho thuộc tính "đãng trí" của ông An. Để giải quyết vấn đề này chúng ta cần phải có chiến lược cho phép lớp nào có quyền ưu tiên cho thừa kế.

Sau đây chúng ta trình bày một chiến lược quyết định lớp có quyền cho thừa kế. Từ đối tượng được hỏi, chúng ta thành lập một danh sách ưu tiên các lớp có thể cung cấp giá trị ngầm định cho các thuộc tính của đối tượng đó. Chẳng hạn, danh sách ưu tiên các lớp cho đối tượng An thừa kế các gái trị ngầm định có thể là danh sách sau:

An
Professors
Programers
Eccentrics
Hackers
Teachers
Intellectual-Persons
People

Trong danh sách này, hai lớp Eccentrics và Hackers đều có thể cung cấp giá trị ngầm định cho thuộc tính "đãng trí". Song lớp Eccentrics đứng trước lớp Hackers trong danh sách ưu tiên. Eccentrics là lớp có quyền cho thừa kế. Do đó đối tượng An thừa kế giá trị ngầm định của thuộc tính "đãng trí" từ lớp Ecccentrics, tức là ta suy ra "ông An là người đãng trí".

Người ta đưa ra một số quy luật để xác định quyền ưu tiên cho thừa kế của các lớp:

Mỗi lớp cần đứng trước bất kỳ lớp mẹ nào của nó trong danh sách ưu tiên.

Luật này có nghĩa rằng, trong bất kỳ đường thừa kế nào (đường thừa kế là đường đi từ đối tượng được hỏi tới các lớp theo các cung isa và ako), lớp gần đối tượng có quyền cho thừa kế hơn là lớp đứng xa đối tượng.

Trong danh sách ưu tiên một lớp mẹ trực tiếp cần đứng trước lớp mẹ trực tiếp khác ở bên phải nó.

Chẳng hạn, Eccentrics và Teachers là hai lớp mẹ trực tiếp của Professors, nhưng Teachers đứng ở bên phải Eccentrics trong lưới, do đó Eccentrics phải đứng trước Teachers trong danh sách ưu tiên đã đưa ra ở trên đều thoả mãn hai luật này.

Chúng ta có thể sử dụng kỹ thuật tìm kiếm theo độ sâu từ trái sang phải hoặc sử dụng thuật toán sắp xếp topo để xác định danh sách ưu tiên các lớp cho thừa kế.

9.3. KHUNG

Khái niệm khung (frame) lần đầu tiên được đưa ra và nghiên cứu bởi Marvin Minsky (1975). Các khung được sử dụng để biểu diễn tri thức về một đối tượng cụ thể, một khái niệm (một lớp đói tượng), hoặc một hoàn cảnh nào đó. Trong mục này chúng ta sẽ xét khái niệm khung , việc biểu diễn tri thức trong một lĩnh vực bởi hệ khung và cơ chế suy diễn trong một hệ khung.

Khung là đơn vị có cấu trúc để biểu diễn tri thức. Có thể hiểu khung như là một cấu trúc dữ liệu. Mỗi khung đều có tên, tên của khung là tên của đối tượng hoặc của khái niệm hoặc của hoàn cảnh mà khung biểu diễn. Mỗi khung gồm một số thành phần, các thành phần của khung được xem như là các lố (slot) chứa các thông tin về đối tượng (đối tượng ở đây cần được hiểu là một đối tượng cụ thể, một khái niệm, một hoàn cảnh điển hình hoặc một hoàn cảnh cá biệt nào đó). Mỗi lỗ có tên và giá trị. Các lỗ biểu diễn các quan hệ với các đối tượng khác hoặc biểu diễn các thuộc tính của đối tượng. Chúng ta sẽ quan tâm tới hai lỗ đặc biệt. Một lỗ đặc biệt có tên là isa (hoặc instance-of) biểu diễn đối tượng được mô tả là thành viên của một lớp đối tượng. Một lỗ đặc biệt khác có tên là ako biểu diễn một lớp là lớp con của một lớp khác (lớp mẹ trực tiếp). Các khung mô tả các đối tượng cụ thể sẽ được gọi là các cá thể (instance frame hoặc instance), các khung mô tả các khái niệm (chẳng hạn, khái niệm chim, bò sát,…), hoặc các hoàn cảnh điển hình (chẳng hạn, một phòng hội thảo điển hình) sẽ được gọi là các lớp (class).

Thông tin chứa đựng trong các lỗ của một khung có thể là một chỉ dẫn tới một khung khác, chẳng hạn thông tin chứa trong các lỗ có tên là isa hoặc ako. Thong tin chứa trong các lỗ biểu diễn các thuộc tính của đối tượng có thể là giá trị của thuộc tính đó, hoặc là một thủ tục cho phép ta tính giá trị của thuộc tính đó thông qua các thông tin khác.

Trong mô hình biểu diễn tri thức bởi các khung, các thông tin về một đối tượng, được bó lại để tạo thành một dơn vị khung biểu diễn đối tượng đó. Từ lưới ngữ nghĩa đã cho, mỗi đỉnh (biểu diễn đối tượng) và các cung xuất phát từ nó có thể được gom lại thành một khung. Ví dụ, từ lưới ngữ nghĩa trong hình 9.1, ta có thể xây dựng lên một hệ khung. Chẳng hạn, từ đỉnh mammal và đỉnh mic ta tạo ra hai khung sau (trang sau).

Bây giờ chúng ta nghiên cứu cơ chế suy diễn trong các hệ khung. Cũng như đối với lưới ngữ nghĩa, vấn đè suy diễn trong các hệ khung là, khi được hỏi về giá trị của một thuộc tính đó. Cơ ché suy diễn trong các hệ khung dựa trên nguyên lý thừa kế.

Giả sử chúng ta có một số tri thức về một số loài chim như sau:

- Chim là đọng vật, biết bay, có hai chân.

Frame: Mammal	
Ako	animal
Cover	hair
Legs	4
Give	milk

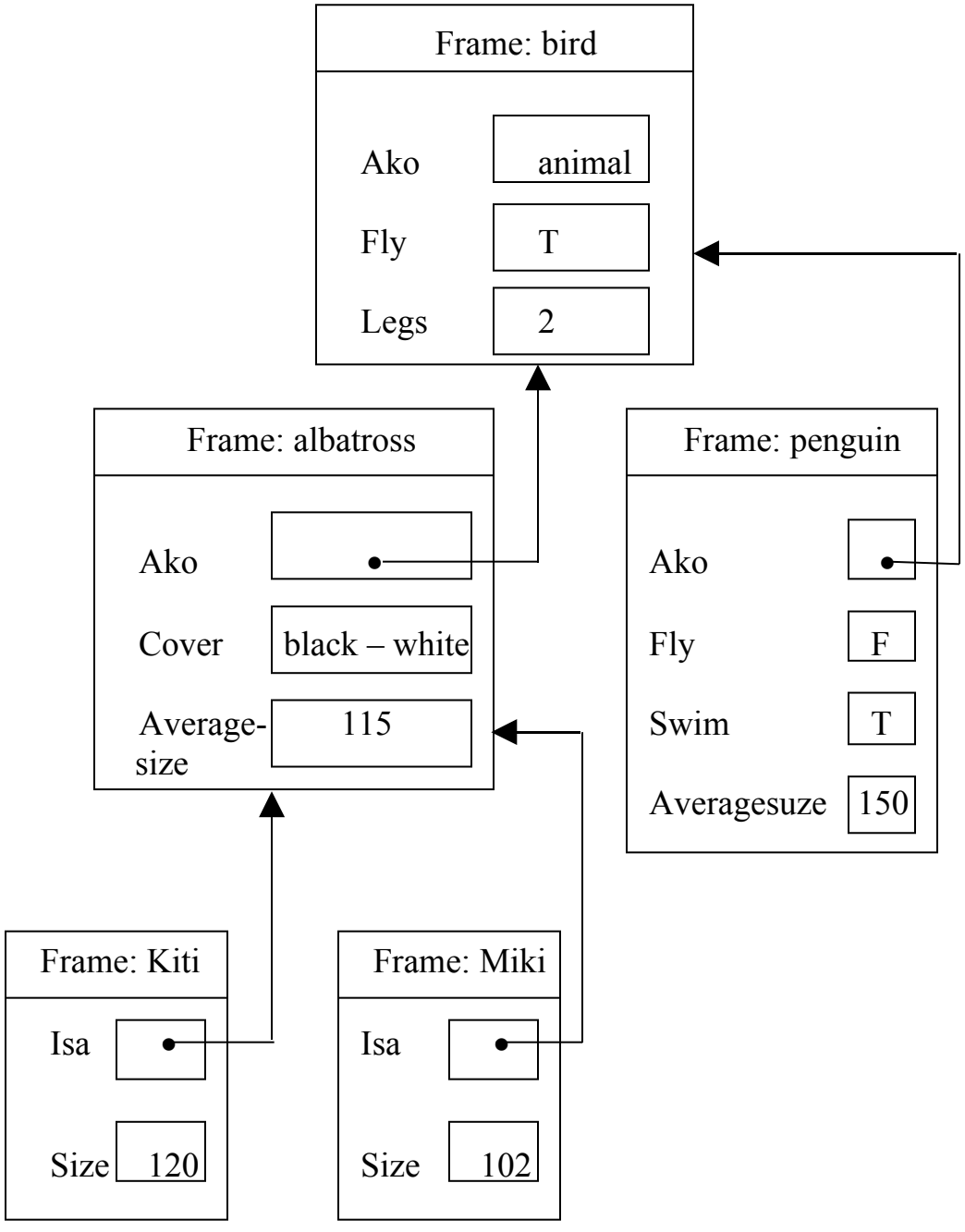
Frame: Mic	
Isa	dog
Color	black
Sex	male

- Hải âu là một loài chim, có lông màu đen - trắng, có cỡ trung bình là 115
- Chim cánh cụt là một loài chim, không biết bay, biết bơi, có cỡ trung bình là 150
- Kiti là một con hải âu, có cỡ 120
- Miki là một con hải âu, có cỡ 102

Từ các tri thức trên, ta xây dựng được một hệ khung được cho trong hình 9.4.

Giả sử chúng ta được hỏi "Kiti có biết bay không?". Trong khung Kiti không có chỗ fly (điều đó có nghĩa là csc thông tin trong khung Kiti không cho phép ta xác định được giá trị của thuộc tính được hỏi). Theo chỉ dẫn trong lỗ isa của khung Kiti,ta tìm đến khung albatross. Trong khung này

cũng không có lỗ fly. Theo chỉ dẫn trong lỗ ako của khung này, ta tìm tới khung bird. Trong khung này, giá trị chưa trong lỗ fly là T. Khung Kiti thừa kế giá trị ngầm định T của lỗ fly từ khung mẹ bird. Như vậy ta đã tìm ra câu trả lời là "Kiti biết bay".



***Hình 9.4*. Một hệ khung.**

Tổng quát, để xác định giá trị của một thuộc tính của một đối tượng, ta có thể thực hiện thủ tục sau. Xuất phát từ khung biểu diễn đối tượng được hỏi. Nếu trong khung có lỗ biểu diễn thuộc tính được hỏi thì giá trị chứa trong lỗ đó là giá trị cần tìm.. Nếu không, ta xác định giá trị của thuộc tính được hỏi bằng thừa kế. Ta đi từ khung hiện thời tới khung tổng quát hơn theo chỉ dẫn được chứa trong các lỗ isa hoặc ako cho tới khung mẹ mà tại đó có lỗ biểu diễn thuộc tính được hỏi. Giá trị ngầm định trong lỗ đó được lấy là giá trị của thuộc tính được hỏi.

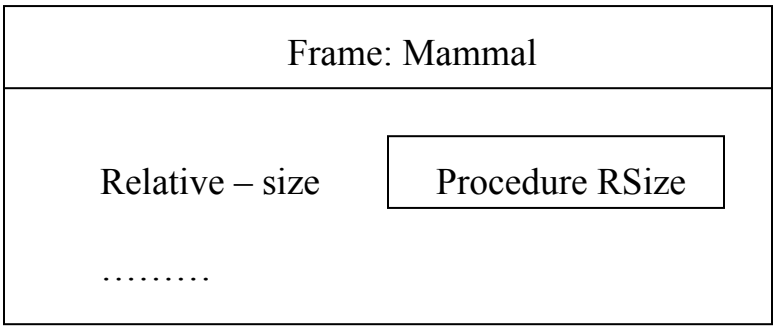
Như chúng ta đã nói, thông tin chứa trong các lỗ biểu diễn thuộc tính có thể là một thủ tục tính giá trị của thuộc tính. Giả sử chúng ta cần xác định cỡ tương đối của các con vật cụ thể: cỡ tương đối là tỷ lệ giữa cỡ của con vật đó với cỡ trung bình của loài vật của con vật đó. chẳng hạn, Kiti có cỡ 120 thuộc loìa chim hải âu có cỡ trung bình 115, do đó cỡ tương đối của Kiti là:

120/115 * 100% = 104%

Tương tự, cỡ tương đối của Miki là:

.1.1 102/115 * 100% = 88%

Như vậy chúng ta cần thủ tục cho phép ta xác định được cỡ tương đối (relative – size) của các con vật, ta gọi thủ tục này là Rsize. Vì thủ tục này được dùng chungcho tất cả các con vật nên ta đưa vào khung animal (khung này biểu diễn lớp tất cả các con vật) một lỗ có tên là relative – size, lỗ này chứa thủ tục Rsize.



Để tính cỡ tương đối của một đối tượng, chẳng hạn Kiti, ta đi từ khung Kiti tới khung albatross rồi tới khung bird, sau đó tới khung animal. Trong khung animal, ta tìm thấy thủ tục Rsize tính cỡ tương đối. Thủ tục này cần tới giá trị của lỗ size trong khung Kiti và giá trị của lỗ average – size trong khung albatross (size và average – size là các tham số của thủ tục). Gọi thủ tục Rsize với các tham số đã được xác định ta tính được cỡ tương đối của Kiti.

Trên đây chúng ta mới chỉ đề cập đến sự thừa kế đơn, tức là mỗi khung chỉ có thể có một khung mẹ trực tiếp duy nhất, hay nói cách khác mỗi khung chỉ có một đường thừa kế duy nhất. trong trường hợp thừa kế bội, khi mỗi khung có nhiều hơn một khung mẹ, giá trị một thuộc tính của khung có thể thừa kế từ nhiều khung mẹ khác nhau. Trong trường hợp này, cũng như trong lưới ngữ nghĩa, chúng ta cần có chiến lược quyết định khung mẹ nào có quyền cho thừa kế.

Các hệ khung đã được sử dụng trong nhiều hệ chuyên gia, trong các hệ nhận dạng, trong các hệ sử lý ngôn ngữ tự nhiên,…

Lý thuyết các hệ khung là cơ sở của phương pháp luận lập trình định hướng đối tượng.

CHƯƠNG 10

TRI THỨC KHÔNG CHẮC CHẮN

Con người hầu như không hiểu biết một cách đầy dủ và chính xác về môi trường, trong nhiều trường hợp các thông tin mà chúng ta biết được hoặc không chính xác, hoặc không chác chắn. Trong hoàn cảnh các thông tin biết được không chắc chắn, các tác nhân thông minh (intelligent agents) vẫn cần phải đưa ra các quyết định, các hành động. Một vấn đề đặt ra là, trên cơ sở nào để các quyết định, các hành động mà các tác nhân đưa ra được xem là hợp lý, đúng đắn. Vì vậy, việc nghiên cứu các mô hình biểu diễn tri thức không chắc chắn và các phương thức suy luận liên quan tới tính không chắc chắn (uncertainty) đóng vai trò quan trọngtrong các hệ thông minh. Trong chương này chúng ta sẽ trình bày lý thuyết xác suất như là cơ sở toán học cho sự biểu diễn tri thức không chắc chắn. sau đó chúng ta sẽ nghiên cứu các mạng xác suất, một mô hình biểu diễn tri thức không chắc chắn và các phương pháp suy luận trong csc mạng xác suất. Cuối cùng chúng ta sẽ xét việc sử dụng các phương pháp lập luận trong mạng xác suất để trợ giúp cho việc đưa ra quyết định.

10.1. KHÔNG CHẮC CHẮN VÀ BIỂU DIỄN

Giả sử ngày mai bạn cần đáp chuyến bay 10 giờ từ sân bay Nội Bài đi thành phố Hồ Chí Minh. Bạn cần lập kế hoạch để đi được chuyến bay tới đó sao cho tốn ít thời gian và tiền bạc. các thông tin chính xác mà bạn biết được là bạn cần đến trước giờ bay một giờ để làm thủ tục; nhà bạn cách sân bay Nội Bài 50 km và xe có thể chạy tới sân bay Nội Bài với vận tốc 50km/h. Bạn có thể lên kế hoạch: gọi taxi đến nhà đón bạn lúc 8h. Tuy nhiên trên đường đi taxi của bạn có thể bị hỏng hóc phải dừng lại sửa chữa, có thể có đám tang làm tắc nghẽn giao thông, có thể có tai nạn trên cầu Thăng Long và công an không cho qua lại cầu để giải quyết hậu quả,… như vậy, có thể có rất nhiều sự kiện không chắc chắn, mang tính ngẫu nhiên xảy ra mà bạn chưa tính đến khi đưa ra kế hoạch trên. Kế hoạch xuất phát lúc 8h bằng xe taxi xem ra chưa đủ sức thuyết phục, nó có thể làm lỡ chuyến bay của bạn.

Với ví dụ trên, chúng ta muốn nói rằng, chúng ta phải thường xuyên đưa ra các quyết định, hành động, rút ra các kết luận trong hoàn cảnh thông tin chúng ta biết được là không chắc chắn.

Có các sự kiện mà chúng ta không biết trước được nó xảy ra hay không, sự xuất hiện của nó mang tính ngẫu nhiên, như các sự kiện trong ví dụ trên. Có các mệnh đề mà tính chân lý của nó ta không thể biết trước được, chẳng hạn mệnh đề "ngày tết trung thu là ngày nắng". chúng ta sẽ biểu diễn khả năng một sự kiện có thể xảy ra, khả năng có một mệnh đề có thể đúng bởi có một số p, $0 \leq p \leq 1$. Số này sẽ được gọi là xác suất của một sự kiện hoặc xác suất của một mệnh đề. Chẳng hạn, chúng ta có thể nói rằng, xác suất "ngày tết trung thu là nắng" là 0,63. Lý thuyết xác suất là cơ sở toán học để biểu diễn tính không chắc chắn của các sự kiện. Trong các mục sau chúng ta sẽ trình bày kỹ hơn về lý thuyết xác suất. Còn bây giờ chúng ta sẽ bàn luận về bản chất và nguồn gốc của tính không chắc chắn.

Tính không chắc chắn có thể xuất hiện từ nhiều nguồn. Trước hết, có những quá trình mà bản chất của nó chúng ta không thể mô tả chính xác bởi các mô hình đơn định. Chẳng hạn, chúng ta xét quá trình một cầu thủ sút phạt 11m. trước khi cầu thủ đá phạt, chúng ta không thể đoán biết được quỹ đạo chuyển động của quả bóng, tốc độ chuyển động của nó, quả bóng có vào gôn, vọt xà ngang hay trúng cột bên trái,…

Tính không chắc chắn có thể xuất hiện do sự hiểu biết không đầy đủ của chúng ta về vấn đề đang xét. Có thể thấy rõ điều này khi các bác sĩ chẩn đoán bệnh, khi các kỹ sư phỏng đoán sự hỏng hóc của máy móc. Do sự hiểu biết không đầy đủ nguyên nhân, cơ chế gây bệnh, người thầy thuốc chỉ có thể nói, chẳng hạn với các triệu chứng và các kết quả xét nghiệm ,à họ biết thì có 80% khả năng bệnh nhân bị sốt xuất huyết.

Ngay cả khi chúng ta có thể mô tả chính xác, đơn định một quá trình, nhưng nếu mô tả đầy đủ và chính xác thì sẽ rất phức tạp,độ phức tạp của tính toán, lập luận sẽ rất cao. Trong các trường hợp đó, Trong các trường hợp đó, người ta có thể mô tả xấp xỉ bằng cách sử dụng tính không chắc chắn để đơn giản cho việc tính toán, suy diễn.

Để làm sáng tỏ các thảo luận đã dẫn ra ở trên, chúng ta xét ví dụ sau. Giả sử bệnh nhân hút thuốc (smoking). Đương nhiên, không phải bệnh nhân nào hút thuốc cũng ung thư (cancer). Trong y học, người ta chưa biết được chính xác những nguyên nhân nào thì một người hút thuốc sẽ bị ung thư và những nguyên nhân nào thì một người hút thuốc sẽ không ung thư. Bây giờ, giả sử rằng người ta biết được đầy đủ các điều kiện mà một bệnh nhân hút thuốc sẽ bị ung thư. Vị từ smoking(p) có nghĩa là bệnh nhân p hút thuốc; vị

từ symtom(p, S_k): bệnh nhân p có triệu chứng S_k và vị từ cancer(p): bệnh nhân p bị ung thư. Sử dụng các vị từ này có:

$$\text{Smoking}(p) \wedge \text{Symtom}(p, S_1) \wedge \ldots \wedge \text{Symtom}(p, S_n) \Rightarrow \text{cancer}(p)$$

Nếu như số các điều kiện Symtom(p, S_k) rất lớn thì trong trường hợp này, thay vì sự mô tả chính xác bởi luật trên người ta sử dụng tính không chắc chắn để mô tả xấp xỉ mối quan hệ nhân-quả. Chẳng hạn, người ta nói:

$$\text{Smoking}(p) \Rightarrow \text{Cancer}(p)$$ với độ tin cậy là 8 %.

10.2. MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA LÝ THUYẾT XÁC SUẤT:

Trong mục này chúng ta sẽ trình bày một số khái niệm cơ bản của lý thuyết xác suất và các công thức tính xác suất được sử dụng đến trong các mục sau. Phương pháp trình bày và các ký hiệu mà chúng ta sử dụng là để thuận lợi cho việc biểu diễn tri thức không chắc chắn và các tính toán cần thiết sau này.

Chúng ta sẽ gọi các mệnh đề mà giá trị chân lý của nó (true/ false) chúng ta không thể biết trước được, nó nhận giá trị true hoặc false một cách ngẫu nhiên, may rủi là mệnh đề không chắc chắn. Chẳng hạn, mệnh đề "ngày 2 tháng 9 sắp tới. Hà Nội trời nhiều mây không mưa" là mệnh đề không chắc chắn. Cần lưu ý rằng, các sự kiện không chắc chắn đều có thể mô tả bởi mệnh đề không chắc chắn.

XÁC SUẤT VÀ CÁC MINH HỌA CỦA XÁC SUẤT

Giả sử bạn đến nhà An nhiều lần và có tới 90% số lần bạn đến gặp An ở nhà. Hôm nay bạn cùng Ba đến nhà An. Ba hỏi bạn, liệu An có nhà không. Bạn trả lời Ba rằng 90% khả năng An ở nhà. Nói cách khác, mệnh đề "An ở nhà" có xác suất là 0,9. Xác suất là một số nằm giữa 0 và 1, biểu diễn khả năng một mệnh để là đúng.

Có nhiều cách minh hoạ khái niệm xác suất. Trong ví dụ trên, qua thống kê các lần bạn đến nhà An, bạn thấy có 90% số lần bạn đến gặp An ở nhà. Người ta xem xác suất "An có nhà" là 0,9. Như vậy có thể xem con số 0,9 như số đo sự thường xuyên của sự kiện "An có nhà".

Giả sử để chuẩn bị cho kỳ thi sắp tới, Cao học tập rất chăm chỉ, Cao lại là học sinh giỏi của lớp. Các thầy dạy Cao nói rằng, xác suất "Cao đỗ trong kỳ thi sắp tới" là 0,98. Trong trường hợp này ta có thể hiểu xác suất là số đo mức độ tin tưởng vào sự đúng đắn của một mệnh đề hoặc sự xuất hiện của một sự kiện. Xác suất trong trường hợp này được gọi là xác suất chủ quan (subjective probability). Một ví dụ khác về xác suất chủ quan là, khi so

sánh đánh giá về điểm mạnh và yếu của hai đội Real Madrid và Valencia, các chuyên gia bóng đá nói rằng, xác suất "Real Madrid thắng Valencia trong trận tới" là 0,58.

Chúng ta sẽ sử dụng các chữ số in hoa A, B, C, … để chỉ các mệnh đề A và B không chắc chắn. Xác suất của mệnh đề A sẽ được ký hiệu là Pr(A). Chúng ta cũng sẽ quan tâm đến xác suất của các mệnh đề hợp thành, đó là các mệnh đề được tạo thành từ các mệnh đề khác bằng cách sử dụng các kết nối logic: và ($\wedge$), hoặc ($\vee$), phủ định ($\neg$).

- Pr($\neg$A) là xác suất để mệnh đề A sai.
- Pr(A $\wedge$ B) là xác suất để cả hai mệnh đề A và B đều đúng. Sau này chúng ta sẽ ký hiệu xác suất này là Pr(A, B).
- Pr(A $\vee$ B) là xác suất để mệnh đề A hoặc mệnh đề B là đúng.

CÁC TÍNH CHẤT CƠ BẢN CỦA XÁC SUẤT

Xác suất của một mệnh đề thoả mãn các tính chất sau đây:

1. Với mệnh đề Q bất kỳ, xác suất là một số nằm giữa 0 và 1.

$$0 \leq Pr(Q) \leq 1.$$

2. Một mệnh đề chắc chắn (một mệnh đề đúng trong mọi minh hoạ) có xác suất là 1.Một mệnh đề không thoả được (mệnh đề sai trong mọi minh hoạ) có xác suất là 0.

3. $Pr(A \vee B) = Pr(A) + Pr(B) - Pr(A \wedge B)$

4. $Pr(\neg A) = 1 - Pr(A)$

Tính chất 4 là hệ quả của 3 tính chất đầu. Thật vậy, từ tính chất 3 thay B bởi $\neg$A ta có:

$$Pr(A \vee \neg A) = Pr(A) + Pr(\neg A) - Pr(A \wedge \neg A)$$

Nhưng theo tính chất 2, vì A $\vee$ $\neg$A là mệnh đề chắc chắn, A $\wedge$ $\neg$A là mệnh đề không thoả được nên $Pr(A \vee \neg A) = 1$ và $Pr(A \wedge \neg A) = 0$. Do đó ta có $1 = Pr(A) + Pr(\neg A)$.

Người ta thường lấy ba tính chát 1,2,3 làm các tiên đề của lý thuyết xác suất.

XÁC SUẤT CÓ ĐIỀU KIỆN

Chúng ta thường quan tâm đến xác suất của một mệnh đề khi chúng ta được biết trước một số thông tin nào đó (thông tin mà ta biết trước này còn được gọi là một bằng chứng (evidence)). Chẳng hạn, chúng ta nói xác suất "một bệnh nhân hút thuốc lá bị ung thư" là 0,28, chúng ta ký hiệu xác suất này là Pr(cancer| smoke) = 0,28. Thông tin mà chúng ta biết trước là bệnh nhân hút thuốc. Xác suất trên có nghĩa là, trong số các bệnh nhân hút thuốc, khả năng một bệnh nhân bị ung thư là 0,28.

Xác suất có điều kiện (cônditinal probability hoặc posterior probability) của mệnh đề A khi cho trước mệnh đề B được ký hiệu là Pr(A| B) và được xác lập bởi công thức.

$$Pr(A|B) = \frac{Pr(A,B)}{Pr(B)}$$

Từ công thức này ta có:

$$Pr(A,B) = Pr(A|B)\ Pr(B)$$

Nói chung, nếu chúng ta quan tâm đến xác suất của một mệnh đề A thì khi biết thêm thông tin mới là mệnh đề B, chúng ta cần tính được xác suất có điều kiện Pr(A|B) bằng suy diễn. Vấn đề này sẽ được nghiên cứu trong các mục sau.

BIẾN NGẪU NHIÊN VÀ PHÂN PHỐI XÁC SUẤT

Chúng ta sẽ quan tâm tới các biến mà nó có thể nhận giá trị này hoặc giá trị khác một cách ngẫu nhiên, không thể biết trước được. Chẳng hạn, khi ta gieo con xúc xắc (khối lập phương), kết quả của các lần gieo có thể biểu diễn bởi một biến, biến này cí thể nhận giá trị bất kỳ từ 1 đến 6. Nhưng nó nhận giá trị nào chúng ta không thể biết trước được khi gieo xúc xắc.

Biến ngẫu nhiên (random variable) là một biến nhận giá trị này hay giá trị khác một cách ngẫu nhiên từ một tập các giá trị nào đó. Tập giá trị này được gọi là miền giá trị (hay còn được gọi là không gian mẫu) của biến ngẫu nhiên. Chẳng hạn, biến Weather (thời tiết khu vực ở Hà Nội lúc 12h) có thể nhận các giá trị sunny, rainy, cloudy. Chúng ta sẽ ký hiệu các biến ngẫu nhiên bổ các chữ cái in X, Y, Z, … hoặc các từ, chẳng hạn, Weather, color, … Sau này chúng ta chỉ quan tâm đến các ***biến ngẫu nhiên rời rạc*****,** tức là miền giá trị của nó là một tập các giá trị rời rạc.

Giả sử X là biến ngẫu nhiên với miền giá trị $\mathbf{\Omega}$ và $a \in \mathbf{\Omega}$. Mệnh đề "X = a" là viết tắt của mệnh đề "biến X nhận giá trị a". Chúng ta sẽ gọi các mệnh đề "X = a" là mệnh đề phân tử (hoặc sự kiện phân tử) cần lưu ý rằng,

các sự kiện phân tử không thể xảy ra đồng thời và trong mỗi lần thử nghiệm, một trong các sự kiện phân tử sẽ xảy ra.

Chúng ta có thể xem một mệnh đề như một biến ngẫu nhiên chỉ nhận hai giá trị (true, false): biến ngẫu nhiên boolean. Giả sử A là một mệnh đề, khi ta xem A như một biến ngẫu nhiên boolean thì Pr(A) là viết tắt của Pr(A = true), còn Pr(¬A) là viết tắt của Pr(A=false).

Hàm Pr(X) xác định trên miền giá trị Ω của biến ngẫu nhiên X, ứng với mỗi $x \in \Omega$ với xác suất Pr(X = x), được gọi là phân phối xác suất của biến ngẫu nhiên X là tập hợp tất cả các xác suất Pr(X = x) với $x \in \Omega$. Các xác suất này cần thoả mãn điều kiện sau:

$$\sum_{x \in \Omega} \Pr(X = x) = 1$$

Ví dụ.Giả sử biến ngẫu nhiên Wather với miền giá trị là (sunny, Rain, Cloudy). Phân phối xác suất của biến ngẫu nhiên này được cho trong bảng sau:

Weather	Pr(Weather)
Sunny	0,6
Rain	0,3
Cloudy	0,1

PHÂN PHỐI XÁC SUẤT CÓ ĐIỀU KIỆN

Giả sử X và Y là hai biến ngẫu nhiên với miền giá trị tương ứng là Ω_X và Ω_Y.Chúng ta ký hiệu Pr(X|Y) là hàm ứng mỗi cặp giá trị $x \in \Omega_X$ và $y \in \Omega_Y$ với xác suất có điều kiện Pr(X = x|Y =y). Hàm Pr(X|Y) được gọi là phân phối xác suất có điều kiện của biến ngẫu nhiên X khi đã cho giá trị của biến ngẫu nhiên Y. các xác suất Pr(X =x|Y = y) cần thoả mãn điều kiện sau.

$$\sum_{x \in \Omega} \Pr(X = x \mid Y = y) = 1, \text{ với mọi } y \in \Omega_y.$$

Ví dụ.Giả sử A và B là hai mệnh đề. Phân phối xác suất Pr (A|B) được cho trong bảng hai chiều sau:

	Pr(A|B)

	A = true	A = false
B = true	0,8	0,2
B = false	0,3	0,7

Cần lưu ý rằng, tổng các số trong mỗi dòng của bảng trên bằng 1.

PHÂN PHỐI XÁC SUẤT KẾT HỢP

Khi sử dụng mô hình xác suất để biểu diễn tri thức không chắc chắn trong một lĩnh vực áp dụng, chúng ta cần đưa vào một tập biến ngẫu nhiên $\{X_1, X_2, \ldots, X_n\}$. Mỗi X_i là một biến ngẫu nhiên rời rạc nhận giá trị trong một miền giá trị $\mathbf{\Omega}_i$ (i = 1, …, n) tương ứng. Giả sử x_i là một giá trị nào đó của X_i (i = 1,…, n). Mệnh đề $(X_1 = x_1) \wedge (X_1 = x_1) \wedge \ldots \wedge (X_1 = x_1)$ biểu diễn một trạng thái của thé giới hiện thực mà chúng ta đang quan tâm. Xác suất của mệnh đề trên được ký hiệu là

$$\Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

Hàm ứng mỗi trạng thái $(X_1 = x_1) \wedge (X_2 = x_2) \wedge \ldots \wedge (X_n = x_n)$ với xác suất $\Pr(X_1 = x_1, X_2 = x_2,\ldots, X_n = x_n)$ được ký hiệu là $\Pr(X_1, X_2, \ldots, X_n)$. Hàm này được gọi là ***phân phối xác suất kết hợp*** (joint probability distribution) của tập biến ngẫu nhiên $\{X_1, \ldots, X_n\}$. Các xác suất $\Pr(X_1 = x_1,\ldots, X_n = x_n)$ cần thoả mãn điều kiện sau:

$$\sum_{x_1 \in \Omega_1} \cdots \sum_{x_n \in \Omega_n} \Pr(X_1 = x_1,\ldots, X_n = x_n) = 1$$

Ví dụ. Giả sử A và B là hai biến ngẫu nhiên boolean. Phân phối xác suất kết hợp Pr(A,B) được cho trong bảng hai chiều sau:

	Pr(A,B)	
	B = true	B = false
A = true	0,1	0,2
A = false	0,3	0,4

Cần lưu ý rằng, tổng của tất cả các số trong bảng phân phối xác suất kết hợp cần bằng 1.

Nếu chúng ta đưa vào biến ngẫu nhiên $X = (X_1, \ldots, X_n)$, mỗi giá trị của X là vectơ $(x_1, \ldots, x_n)$, trong đó $x_i \in \mathbf{\Omega}_i$ (i = 1, …, n), tức là miền giá trị

$\mathbf{\Omega}$ của biến X là tích Đề-các của các miền giá trị $\mathbf{\Omega}_i$ của X_i, $\mathbf{\Omega} = \mathbf{\Omega}_i \times \ldots \times \mathbf{\Omega}_n$, thì phân phối xác suất kết hợp $Pr(X_n, \ldots, X_n)$ chính là phân phối xác suất của biến ngẫu nhiên X.

CÁC CÔNG THỨC TÍNH XÁC SUẤT

Khi sử dụng mô hình xác suất để biểu diễn tri thức không chắc chắn, thường là bằng thực nghiệm, bằng sự hiểu biết và bằng các kinh nghiệm tích luỹ được chúng ta đã xác định trước được một số xác suất nào đó. Sau này khi được biết các thông tin mới (các bằng chứng), chúng ta cần tính xác suất của các mệnh đề được hỏi. Sau đây chúng ta sẽ trình bày một số công thức quan trọng. Các công thức này cho phép ta biểu diễn một xác suất qua các xác suất khác và được sử dụng thường xuyên trong các mục sau.

Trước hết chúng ta đưa ra một số ký hiệu. Giả sử A, B, C, D, E là các mệnh đề.Trong các biểu thức sau, vế trái là một cách viết khác của vế phải.

$$Pr(A, B, C) = Pr(A \wedge B \wedge C)$$

$$Pr(A, B| C) = Pr(A \wedge B |C)$$

$$Pr(A |B, C) = Pr(A |B \wedge C)$$

$$Pr(A, B, C |D, E) = Pr(A \wedge B \wedge C| D \wedge E)$$

Giả sử chúng ta nhận được phân phối xác suất kết hợp $Pr(X_1 \ldots\ldots X_n)$ của tập biến ngẫu nhiên $\{X_1, \ldots, X_n\}$. Khi đó chúng ta có thể tính được phân phối xác suất kết hợp của một tổ hợp bất kỳ của các biến $\{X_1, \ldots, X_n\}$. Để đơn giản, ta xét trường hợp ba biến. Giả sử chúng ta đã biết phân phối xác suất Pr (X,Y,Z), trong đó X, Y, Z là các biến ngẫu nhiên với các miền giá trị là $\mathbf{\Omega}_X$, $\mathbf{\Omega}_Y$, $\mathbf{\Omega}_Z$ tương ứng. Khi đó ta có thể tính được phân phối xác suất của hai biến, chẳng hạn Pr(X,Y) bằng công thức sau:

$$Pr(X,Y) = \sum_{z \in \Omega_Z} Pr(X,Y,Z = z)$$

Cần nhớ rằng, công thức này là viết tắt của công thức sau:

$$Pr(X = x.Y = y) = Pr(X = x,Y = y,Z = z)$$

trong đó x, y là các giá trị bất kỳ trong miền $\mathbf{\Omega}_X$, $\mathbf{\Omega}_Y$ tương ứng.

Ta cũng có thể tính được phân phối xác suất của một biến, chẳng hạn Pr(X) theo công thức sau:

$$\Pr(X) = \sum_{y \in \Omega_Y} \sum_{z \in \Omega_Z} \Pr(X, Y = y, Z = z)$$

LUẬT TỔNG

Từ các công thức

$$\Pr(X) = \sum_{y \in \Omega_Y} \Pr(X, Y = y)$$

$$\Pr(X, Y = y) = \Pr(X|Y = y) \Pr(Y = y)$$

Chúng ta suy ra một công thức quan trọng sau:

$$\Pr(X) = \sum_{y \in \Omega_Y} \Pr(X, Y = y) \Pr(Y = y)$$

Công thức này được gọi là ***luật tổng***.

LUẬT TÍCH

Chúng ta biết rằng

$$\Pr(X,Y) = \Pr(X|Y) \Pr(Y)$$

Bây giờ chúng ta xét phân phối xác suất của ba biến X, Y, Z. Theo công thức trên ta có:

$$\Pr(X,Y,Z) = \Pr(X|Y,Z) \Pr(Y,Z)$$

$$\Pr(Y,Z) = \Pr(Y|Z) \Pr(Z)$$

Từ hai công thức trên ta có công thức sau:

$$\Pr(X,Y,Z) = \Pr(X|Y,Z) \Pr(Y|Z) \Pr(Z)$$

Tổng quát ta có:

$$\Pr(X_1, X_2, \ldots, X_n) = \Pr(X_1|X_2, \ldots, X_n) \Pr(X_2|X_3, \ldots, X_n) \ldots$$
$$\Pr(X_{n-1}|X_n) \Pr(X_n)$$

Công thức này gọi là ***luật tích***. Luật tích cho phép ta phân tích một xác suất kết hợp thành tích của các xác suất có điều kiện. Cần lưu ý rằng, thứ tự của các biến trong phân phối xác suất kết hợp là không quan trọng.

Do đó, vì chúng ta có n! hoán vị của n biến $X_1, \ldots, X_n$ nên chúng ta sẽ có n! cách phân tích một xác suất kết hợp thành tích của các xác suất có điều kiện.

CÔNG THỨC BAYES

Theo công thức tính xác suất có điều kiện ta có:

$$\Pr(X|Y) = \frac{\Pr(X, Y)}{\Pr(Y)}$$

Mặt khác, theo luật tích $\Pr(X,Y) = \Pr(Y|X)\Pr(X)$ ta có công thức:

$$\Pr(X|Y) = \frac{\Pr(Y \mid X)\Pr(X)}{\Pr(Y)}$$

Công thức này được gọi là công thức Bayes, đó là công thức được sử dụng thường xuyên nhất để tính xác suất có điều kiện.

Một nhận xét quan trọng là, để tính phân phối xác suất $\Pr(X|Y)$ theo công thức Bayes, chúng ta chỉ cần biết các phân phối xác suất $\Pr(Y|X)$ và $\Pr(X)$, bởi vì nếu biết các xác suất này thì theo luật tổng chúng ta sẽ tính dược các xác suất $\Pr(Y)$.

Bây giờ chúng ta đưa ra một áp dụng đơn giản của công thức Bayes. Trong y học, người ta phải tính các xác suất có điều kiện (điều kiện ở đây thường là một triệu chứng một số các triệu chứng) để đưa ra các kết luận chẩn đoán. Giả sử bác sĩ biết xác suất một bệnh nhân sâu răng (cavity) bị đau răng (toothache) là 0,65, tức là $\Pr(\text{Toothache}|\text{Cavity}) = 0,65$. Bác sĩ cũng biết rằng, xác suất của một người sâu răng 0,05, $\Pr(\text{Cavity}) = 0,05$ và xác suất của một người đau răng là 0,04, $\Pr(\text{Toothache}) = 0,04$. Khi đó, theo công thức Bayes ta tính được:

$$\Pr(\text{Cavity}|\text{Toothache}) = \frac{\Pr(\text{Toothache} \mid \text{Cavity})\Pr(\text{Cavity})}{\Pr(\text{Toothache})}$$

$$= \frac{0,65 * 0,05}{0,04} = 0,81$$

Như vậy, xác suất để một người đau răng bị sâu răng là 0,81.

SỰ ĐỘC LẬP CÓ ĐIỀU KIỆN CỦA CÁC BIẾN NGẪU NHIÊN

Sau đây chúng ta sẽ bghiên cứu các mối quan hệ độc lập và phụ thuộc giữa các biến ngẫu nhiên.

Giả sử X, Y và Z là các biến ngẫu nhiên với các miền giá trị là $\mathbf{\Omega}_X$, $\mathbf{\Omega}_Y$, và $\mathbf{\Omega}_Z$ tương ứng. Chúng ta nói rằng, biến ngẫu nhiên ***X độc lập*** với biến ngẫu nhiên Y nếu:

$$Pr(X|Y) = Pr(X)$$

Công thức này có nghĩa là, với mọi $x \in \mathbf{\Omega}_X$ và với mọi $y \in \mathbf{\Omega}_Y$

$$Pr(X = x|Y = y) = Pr(X = y)$$

Như vậy, biến ngẫu nhiên X độc lập với biến ngẫu nhiên Y có nghĩa là Y nhận giá trị nào không ảnh hưởng gì đến X.

Biến ngẫu nhiên X được gọi là ***độc lập có điều kiện*** với biến ngẫu nhiên Y khi cho trước Z, nếu:

$$Pr(X|Y,Z) = Pr(X|Z)$$

Nếu X không độc lập có điều kiện với Y khi cho trước Z thì ta nói X phụ thuộc có điều kiện vào Y khi cho trước Z. Độc giả có thể dễ dàng chứng minh được rằng, nếu X độc lập (có điều kiện) với Y (khi cho trước Z) thì Y cũng độc lập (có điều kiện) với X (khi cho trước Z). Hay nói cách khác, quan hệ độc lập có tính đối xứng.

Giả sử X độc lập có điều kiện với Y khi cho trước Z. Khi đó chúng ta có thể tính được phân phối xác Pr(X,Y|Z) theo công thức sau:

$$Pr(X,Y|Z) = Pr(X|Z)\ Pr(Y|Z)$$

Công thức này dễ dàng được suy ra từ định nghĩa tính độc lập và từ luật tích.

10.3. MẠNG XÁC SUẤT

Như chúng ta đã nói, để biểu diễn chi thức không chắc chắn trong một lĩnh vực áp dụng, chúng ta có thể sử dụng một tập các biến ngẫu nhiên $\{X_1, \ldots, X_n\}$ và phân phối xác suất $Pr(X_1, \ldots, X_n)$. Khi đó chúng ta có thể tính được xác suất kết hợp của một tổ hợp bất kỳ các biến ngẫu nhiên và do đó có thể tính được xác suất có điều kiện bất kỳ của các biến ngẫu nhiên

$X_1,\ldots,X_n$.Điều đó có nghĩa là nếu biết được phân phối xác suất $Pr(X_1,\ldots,X_n)$ thì ta có thể tìm được câu trả lời cho các câu hỏi về các biến $X_1, \ldots, X_n$. Tuy nhiên phương pháp này có nhiều nhược điểm. Trước hết chúng ta rất khó xác định được các xác suất $Pr(X_1, \ldots, X_n)$. Mặt khác, để lưu phân phối xác suất $Pr(X_1, \ldots, X_n)$ chúng ta cần bảng n chiều, chỉ với $X_i(i =1, \ldots, n)$ là các biến ngẫu nhiên boolean thì bảng cũng đã chứa 2^n số! Do đó chúng ta cần xây dựng một mô hình thích hợp để biểu diễn tri thức không chắc chắn. Mô hình này phải thoả mãn hai điều kiện sau:

- Giảm bớt số các xác suất ban đầu cần biết trước
- Đơn giản sự tính toán để tìm ra câu trả lời cho các câu hỏi

Mô hình mà chúng ta sẽ nghiên cứu là mạng xác suất (probabilistic network). Người ta còn sử dụng các thuật ngữ khác; mạng tin tưởng (belief network), mạng nguyên nhân (causal network).

10.3.1. Định nghĩa mạng xác suất

mạng xác suất là một đồ thị định hướng không có chu trình và thoả mãn các điều kiện sau:

- Các đỉnh của đồ thị là các biến ngẫu nhiên
- Mỗi cung từ đỉnh X đến đỉnh Y biểu diễn sự ảnh hưởng trực tiếp của biến ngẫu nhiên X đến biến ngẫu nhiên Y (hay Y phụ thuộc trực tiếp vào X). Đỉnh X được gọi là đỉnh cha của Y.
- Tại một đỉnh được cho phân phối xác suất có điều kiện của đỉnh đó khi cho trước các cha của nó. Các xác suất này biểu diễn hiệu quả mà các cha tác dụng vào nó.

Chẳng hạn, nếu $X_1, \ldots, X_n$ là tất cả các đỉnh cha của đỉnh Y trong mạng thì tại đỉnh Y được cho phân phối xác suất có điều kiện $Pr(Y| X_1 , \ldots, X_n)$.

Ví dụ. Để làm ví dụ về mạng xác suất, chúng ta xét ví dụ sau, ví dụ này là của Judea Pearl. Xét hoàn cảnh sau: nhà bạn có lắp đặt hệ thống báo động trộm. Nó sẽ rung chuông khi phát hiện ra trộm, nó cũng kêu khi có động đất nhẹ. Bạn có hai người hàng xóm là Lan và Mai. Khi bạn ở cơ quan, họ hứa sẽ gọi điện cho bạn nếu nghe thấy chuông báo trộm kêu. Lan đôi khi nhầm lẫn chuông điện thoại với chuông báo trộm và cũng gọi cho bạn. mai thì hay nghe âm nhạc to nên đôi khi không nghe thấy chuông báo trộm. Chúng ta đưa ra các ký hiệu sau:

B: sự kiện "có trộm"

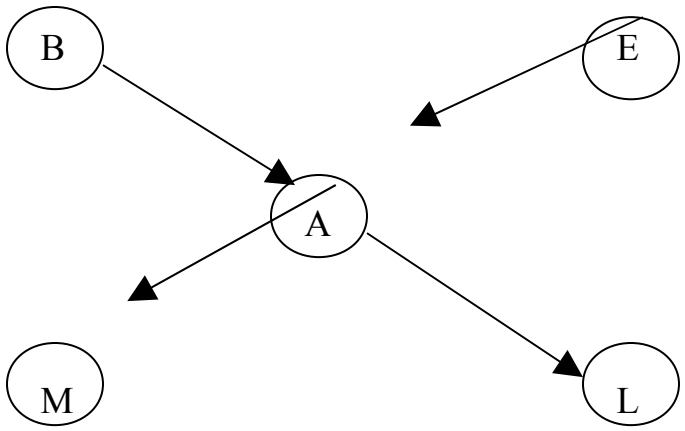
E: Sự kiện "có động đất nhẹ"

A: Sự kiện "chuông báo trộm kêu"

L: Sự kiện "Lan gọi cho bạn"

M: Sự kiện "Mai gọi cho bạn"

Mạng xác suất cho phép chúng ta biểu diễn các mối quan hệ nhân quả trực tiếp giữa các sự kiện. Sự kiện "có trộm" và sự kiện "có động đất nhẹ" ảnh hưởng trực tiếp tới sự kiện "chuông báo trộm kêu". Chúng ta xem rằng cả Lan và Mai đều không nhận biết được nhà bạn có trộm, họ cũng không cảm nhận được động đất nhẹ. Chỉ có chuông báo trộm là tác động đến các cuộc gọi của họ cho bạn. Từ các mối quan hệ phụ thuộc trực tiếp giữa các sự kiện đã nêu, chúng ta xây dựng được mạng xác suất trong hình 10.1. chúng ta sẽ gọi mạng xác suất này là mạng báo động trộm.



***Hình 10.1*. Một mạng xác suất : Mạng báo động trộm**

Trong mạng báo động trộm, chúng ta mới chỉ đưa vào các mối quan hệ nhân quả trực tiếp, cơ bản giữa các sự kiện. còn các thông tin lan đôi khi nhầm lẫn chuông điện thoại với chuông báo trộm và Mai vì nghe nhạc to mà không nghe thấy chuông báo trộm, chúng ta sẽ sử lý những thông tin này như thế nào? Thực ra còn vô số nguyên nhân làm cho chuông báo trộm kêu hoặc không kêu (chẳng hạn, các nguyên nhân về thời tiết, về điện …). Cũng như vậy, còn có rất nhiều nguyên nhân khác làm cho Lan và Mai gọi hoặc không gọi cho bạn (chẳng hạn, Lan tình cờ nghe thấy tiếng kêu gì đó giống như tiếng chuông báo trộm và gọi cho bạn, hoặc có lúc chuông báo trộm thì mai vừa ra khỏi nhà và không gọi cho bạn được, …). Chúng ta tích hợp tất cả các nguyên nhân chủ yếu và các nguyên nhân khác của một kết quả và biểu diễn xấp xỉ bởi csc bảng xác suất có điều kiện. như vậy, trong mạng xác

suất, chúng ta cần xác định các bảng phân phối xác suất có điều kiện tại các đỉnh có cha. Trong ví dụ đang xét, đó là các phân phối xác suất Pr(A|B,E), Pr(L|A) và Pr(M|A). Chúng ta cũng cần xác định các phân phối xác suất của các đỉnh không có cha. Trong mạng báo động trộm, chúng ta cần có các phân phối xác suất Pr(B) và Pr(E). Do đó, với mạng báo động trộm có năm đỉnh, chúng ta cần được cho năm bảng phân phối xác suất, chẳng hạn năm bảng sau:

B	Pr(B)
True	0,01
False	0,99

E	Pr(E)
True	0,02
False	0,98

B	E	Pr(A|B,E)	
		A = True	A = False
True	True	0,95	0,05
True	False	0,93	0,07
False	True	0,21	0,71
False	False	0,01	0,99

A	Pr(L|A)	
	L = True	L = False
True	0,92	0,08
False	0,05	0,95

A	Pr(M|A)	
	M = True	M = False
True	0,8	0,2
False	0,01	0,99

Cần lưu ý rằng, trong các bảng phân phối xác suất không điều kiện tổng của tất cả các số trong một cột phải bằng 1, còn trong các bảng phân phối xác suất có điều kiện, tổng của tất cả các số trong mỗi dòng phải bằng 1.

10.3.2. Vấn đề lập luận trong mạng xác suất

Khi chúng ta biểu diễn tri thức không chắc chắn bởi mạng xác suất, chúng ta cần có các phương pháp suy ra các kết luận từ các thông tin mà chúng ta đã biết trong mạng và từ các bằng chứng mà chúng ta thu thập được. Chúng ta trở lại mạng báo động trộm. Giả sử bạn nhận được điện thoại do Lan gọi, bạn cần biết khả năng có trộm, tức là bạn cần tính được xác suất

Pr(B|L). Hoặc có thể bạn nhận được điện thoại của cả Lan và Mai. bạn cần đánh giá khả năng có trộm khi cả Lan và Mai đều gọi cho bạn, tức là bạn cần tính được xác suất có điều kiện Pr(B|L,M).

Giả sử chúng ta đã biết giá trị của một biến ngẫu nhiên trong mạng. các biến ngẫu nhiên mà chúng ta đã biết giá trị của chúng được gọi là các ***biến bằng chứng*** (evidence variables). Và sự gán giá trị cho một biến bằng chứng được gọi là một bằng chứng. Vấn đề suy diễn trong mạng xác suất, một cách tổng quát, được phát biểu như sau: Cho trước tập các bằng chứng E và một biến mà ta quan tâm X (X được gọi là ***biến hỏi***), chúng ta cần tính được phân phối xác suất có điều kiện Pr(X|E). Chẳng hạn, trong mạng báo động trộm, giả sử chúng ta biết Lan và Mai gọi. Như vậy, tập các bằng chứng E là {L = true, M = true}. Giả sử chúng ta muốn biết khả năng có trộm, khi đó biến hỏi ở đây là B. Chúng ta cần tính Pr(B = true|, M = true).

Trong mạng xác suất người ta thường phân biệt hai dạng lập luận sau.

- ***Lập luận chẩn đoán*** (diagnostic reasoning). Đó là lập luận đi từ các hậu quả tới các nguyên nhân. Trong trường hợp này, chúng ta được cho các biến bằng chứng là các hậu thế của biến hỏi. chẳng hạn, trong mạng báo động trộm, chúng ta muốn biết khả năng có trộm khi Lan gọi. Biến chứng là L, Biến hỏi là B. trong trường hợp này chúng ta có thể suy ra Pr(B|L) = 0,016. trong chẩn đoán y học, các bác sĩ cần tính xác suất của một bệnh khi biết các bằng trứng dưới dạng triệu chứng và các kết quả xét nghiệm của bệnh nhân.
- ***Lập luận tiên đoán*** (predictive reasoning). Đó là lập luận đi từ hậu quả tới các nguyên nhân. Chúng ta được cho các biến bằng chứng là tiền thân của biến hỏi. Chẳng hạn, biết trức là có trộm, biến bằng chứng là B. Chúng ta cần biết khả năng Lan và khả Mai gọi điện cho bạn, tức là các biến hỏi là L và M. chúng ta có thể tính được Pr(L|B) = 0,86 và Pr(M|B) = 0,67.

Trong trường hợp tổng quát, các biến bằng chứng có thể không phải là tiền thân hoặc không phải là hậu thế của biến hỏi. Chẳng hạn, chúng ta biết trước rằng không có động đất và Lan gọi. Chúng ta cần đánh giá khả năng có trộm khi được biết các thông tin đó. Có thể tính được:

$$Pr(B = true|E = false, L = true) = 0,03.$$

Vấn đề tính chính xác phân phối xác suất Pr(X|E) của biến hỏi X khi được cho tập bằng chứng E là vấn đề NP khó. Tuy nhiên, trong một số trường hợp đặc biệt người ta có thể tính được Pr(X|E) trong thời gian đa thức. Trong các mục sau chúng ta sẽ trình bày các thuật toán cho phép ta

tính chính xác Pr(X|E) với thời gian tỷ lệ với số đỉnh trong mạng, trong trường hợp mạng xác suất có kết nối đơn. Sau đó chúng ta sẽ trình bày các thuật toán cho phép ta tính xấp xỉ Pr(X|E) trong trường hợp tổng quát.

10.3.3. Khả năng biểu diễn của mạng xác suất

Mô hình xác suất để biểu diễn tri thức không chắc chắn trong một lĩnh vực bao gồm một tập các biến ngẫu nhiên $X_1, \ldots, X_n$. Nếu chúng ta biết được phân phối xác suất kết hợp $Pr(X_1, \ldots, X_n)$, chúng ta có thể tính được xác suất của một tổ hợp biến bất kỳ. Do đó nếu E là một tập các bằng chứng và X là biến hỏi thì ta có thể tính được xác suất Pr(X|E) theo công thức định nghĩa xác suất có điều kiện. Như vậy phân phối xác suất $Pr(X_1, \ldots, X_n)$ mô tả đầy đủ tri thức không chắc chắn về một lĩnh vực áp dụng.

Sau đây chúng ta sẽ chứng tỏ rằng, mạng xác suất cũng cho phép ta mô tả đầy đủ một lĩnh vực. Muốn vậy, chúng ta sẽ chứng minh rằng, từ các thông tin trong mạng chứa n biến $X_1, \ldots, X_n$, chúng ta có thể tính được phân phối xác suất $Pr(X_1, \ldots, X_n)$.

Trước hết, chúng ta giả thiết rằng, trong mạng xác suất mỗi đỉnh là độc lập có điều kiện với tất cả các đỉnh không phải là hậu thế của nó khi cho trước các đỉnh cha của nó. Lưu ý rằng, đỉnh Y là hậu thế (con cháu) của đỉnh X nếu tồn tại một đường đi trong mạng từ X đến Y. Chẳng hạn, với mạng trong hình 10.2 thì các đỉnh H, G, S là các hậu thế của đỉnh C, còn các đỉnh B, D, R, S là hậu thế của đỉnh A. Nếu Y là hậu thế của X thì ta nói X là tiền thân của Y. Trong hình 10.2, ta có đỉnh S độc lập với các đỉnh A, B, C, G, R khi cho trước D là H, tức là:

$$Pr(S|A, B, C, D, H, R, G) = Pr(S|D, H)$$

Giả thiết về sự độc lập có điều kiện của một đỉnh với các đỉnh không phải là hậu thế của đỉnh đó khi cho trước các cha của nó là một giả thiết hợp lý, bởi vì tiền thân của một đỉnh X chỉ ảnh hưởng tới X thông qua các cha của X, còn các đỉnh khác không phải là hậu thế của X thì không ảnh hưởng gì tới X.

Chúng ta có thể sử dụng phương pháp sắp xếp topo (xem trong [8]) để sắp xếp các đỉnh của mạng xác suất thành một danh sách sao cho nếu X là tiền thân của Y thì X phải đứng sau Y trong một danh sách. Giả sử $X_1X_2\ldots X_n$ là danh sách các biến trong mạng xác suất theo thứ tự topo như trên. Do giả thiết về sự độc lập của các biến trong mạng xác suất, chúng ta có:

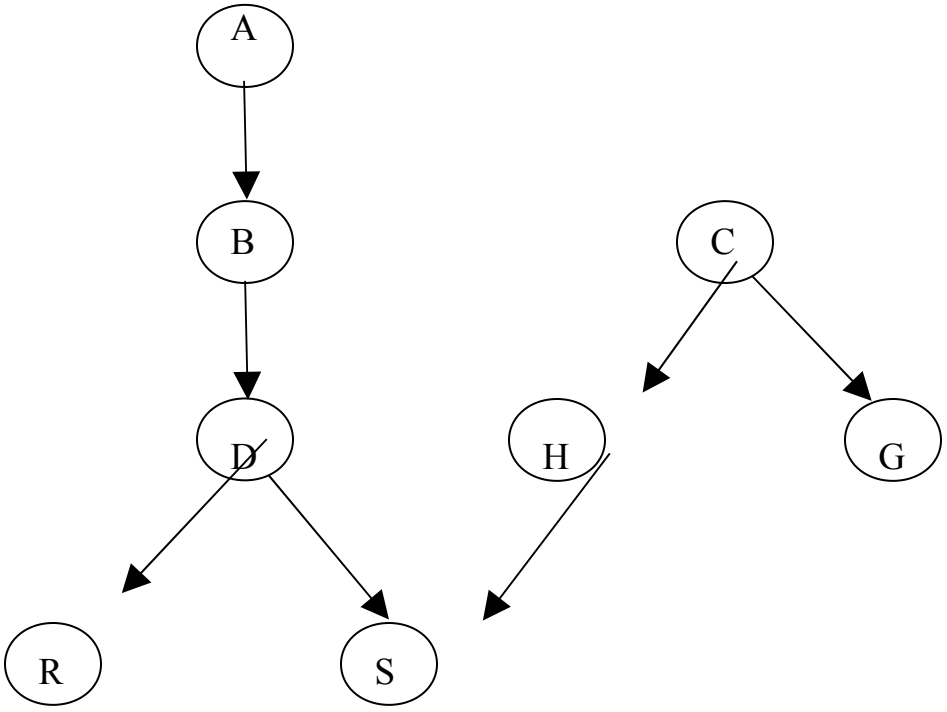
$$Pr(X_i|X_{i+1}, \ldots, X_n) = Pr(X_i|parent(X_i))$$

Trong đó, $parent(X_i)$ là tập các đỉnh cha của X_i.

Sử dụng luật tích ta có:

$$\Pr(X_1X_2\ldots X_n) = \prod_{i=1}^{n} \Pr(X_i \mid \text{parent}(X_i))$$

Công thức này cho phép ta tính được phân phối xác suất $\Pr(X_1, \ldots, X_n)$ từ các bảng xác suất tại mỗi đỉnh của mạng xác suất. Như vậy mạng xác suất có khả năng mô tả đầy đủ tri thức không chắc chắn trong một lĩnh vực.



***Hình 10.2*. Một mạng xác suất**

Sự mô tả tri thức không chắc chắn bởi tập biến ngẫu nhiên $\{X_1, \ldots, X_n\}$ và phân phối xác suất $\Pr(X_1, \ldots, X_n)$ là sự mô tả có tính toàn cục. Sự mô tả bởi mạng xác suất với bảng phân phối xác suất có điều kiện tại mỗi đỉnh là sự mô tả mang tính địa phương. Song sự mô tả toàn cục rất khó thực hiện bởi vì chúng ta gặp rất nhiều khó khăn trong việc xác định phân phối xác suất $\Pr(X_1,\ldots,X_n)$. còn với cách mô tả địa phương bởi mạng xác suất thì bằng cách tham khảo các chuyên gia trong lĩnh vực áp dụng, chúng ta có thể xác định được các bảng xác suất có điều kiện tại mỗi đỉnh. Mặt khác, việc lưu trữ các bảng xác suất tiết kiệm được rất nhiều không gian nhớ so với việc lưu trữ bảng xác suất $\Pr(X_1,\ldots,X_n)$. Chẳng hạn, với mạng gồm n biến ngẫu nhiên boolean và mỗi đỉnh chỉ có nhiều nhất hai cha, tại mỗi đỉnh ta

chỉ cần lưu tối đa 2^3=8 số và do đó trong toàn bộ mạng ta chỉ cần lưu 8n số. trong khi đó trong bảng xác suất $Pr(X_1,\ldots,X_n)$ ta phải lưu 2^n số.

10.3.4. Sự độc lập của các biến trong mạng xác suất

Chúng ta đã giả thiết rằng, trong các mạng xác suất mỗi đỉnh độc lập có điều kiện với các đỉnh không phải là hậu thế của nó, khi cho trước các cha của nó. Một vấn đề mà bây giờ chúng ta quan tâm là liệu chúng ta có thể biết được tập đỉnh U là độc lập có điều kiện với tập đỉnh V khi cho trước tập đỉnh bằng chứng E? Câu trả lời là có. Trong mạng xác suất, dựa vào các đương đi nối các đỉnh trong mạng, chúng ta có thể xác định được sự đọc lập của các tập đỉnh.

Chúng ta gọi một đường đi vô hướng trong mạng là đường đi mà chúng ta không quan tâm tới hướng của các cung trên đường đi đó. Chẳng hạn, RDSHCG là một đường đi vô hướng nối đỉnh R với đỉnh G trong mạng hình 10.2.

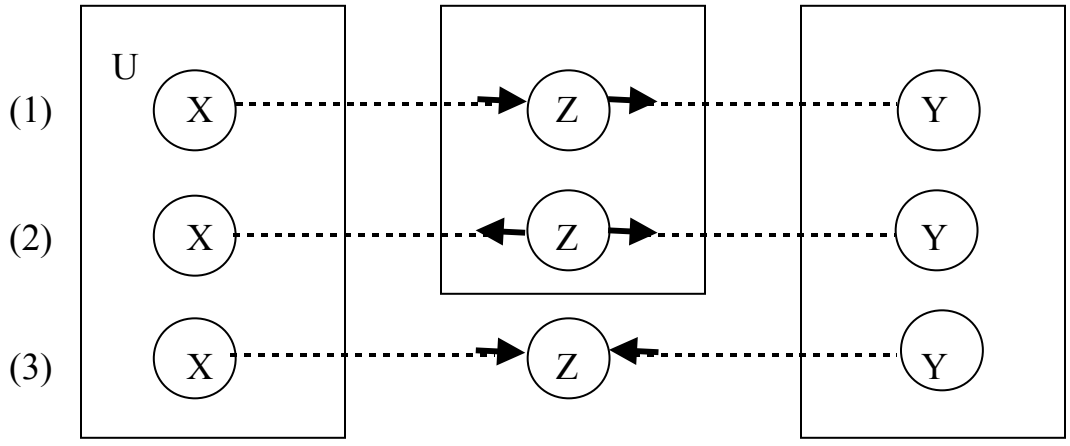
Chúng ta nói tập đỉnh E tách hai tập đỉnh U và V trong mạng nếu mọi đường đi vô hướng nối một đỉnh X trong U với một đỉnh Y trong V đều tồn tại một đỉnh Z trên đường đi sao cho một trong ba điều kiện sau được thoả mãn.

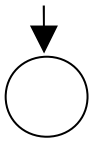
1. Z nằm trong E và có một cung trên đường đi tới Z, một cung trên đường đi xuất phát từ Z.

2. Z nằm trong E và có hai đường trên cung trên đường đi xuất phát từ z.

3. Z và các hậu thế của Z đều không thuộc E và có hai cung trên đường đi tới Z.

Hình 10.3 minh hoạ một tập đỉnh E tách hai tập đỉnh U và V. Trong hình này đã chỉ ra các đường đi trong mạng nối một đỉnh trong U với một đỉnh trong V thoả mãn một trong ba điều kiện trên.





***Hình 10.3*. Tập đỉnh E tách các tập đỉnh U và V.**

Người ta chứng minh được rằng, trong mạng xác suất, nếu các tập đỉnh U và V tách được bởi tập đỉnh E thì U độc lập có điều kiện với V khi cho trước E. tức là pr(U|V,E) = Pr(U|E).

Ví dụ. Trong mạng hình 10.2, B độc lập với R khi cho trước D, H độc lập với G, khi cho trước C và D độc lập với H khi cho trước a.

Như vậy tính tách được của các tập đỉnh đặc trưng cho sự độc lập có điều kiện của các tập đỉnh đó.

Khi xây dựng các thủ tục suy diễn trong mạng xác suất, chúng ta thường sử dụng đến đặc trưng trên để tính các xác suất có điều kiện.

Trong mục sau, chúng ta sẽ nghiên cứu vấn đề suy diễn trong mạng xác suất. vấn đề suy diễn trong mạng xác suất là vấn đề NP- khó. Đầu tiên, để làm quen với các phương pháp tính các xác suất có điều kiện trong mạng, chúng ta xét mạng đơn giản nhất: mạng có cấu trúc cây. Sau đó chúng ta xét vấn đề suy diễn trong mạng tổng quát hơn: mạng kết nối đơn (còn gọi là đa cây). Thuật toán suy diễn trong mạng kết nối đơn có độ phức tạp thời gian tỷ lệ với số đỉnh của mạng. cuối cùng chúng ta xét vấn đề suy diễn trong mạng tổng quát: mạng đa kết nối, ở đó chúng ta sẽ trình bày các phương pháp tính xấp xỉ, đó là các phương pháp mô phỏng ngẫu nhiên, các phương pháp này được ứng dụng rộng rãi trong các ứng dụng.

10.4. SUY DIỄN TRONG MẠNG CÓ CẤU TRÚC CÂY

Trong mục này chúng ta sẽ xét mạng có cấu trúc cây. Đó là mạng mà mỗi đỉnh của nó chỉ có nhiều nhất một cha, tức là có dạng cây.

Đối với mạng có cấu trúc cây, cho trước tập các bằng chứng E, X là một biến ngẫu nhiên trong mạng, chúng ta có thể tính được chính xác phân phối xác suất Pr(X|E) trong thời gian tuyến tính với số đỉnh trong mạng.

Chúng ta giả thiết rằng, biến hỏi X không nằm trong tập các biến bằng chứng đó là $X = x_0$ (x_0 là giá trị nào đó trong miền X), việc tính Pr(X|E) là tầm thường. Cụ thể, ta có

$$Pr(X = x_0 \mid E) = 1$$

$$Pr(X = x \mid E) = 0, \text{ với mọi } x \neq x_0$$

Khi loại đỉnh X ra khỏi cây, chúng ta sẽ phân hoạch cây đã cho thành n + 1 cây con: cây con của X với gốc là $Y_1,\ldots,Y_n$ và một cây còn lại (xem hình 10.4).

Gọi E_i (i = 1, …, n) là các tập bằng chứng nằm trên cây con gốc Y_i, và E^- là tập các bằng chứng nằm dưới X. Ta có:

$$E^- = E_1^- \cup \ldots \cup E_n^-$$

Các E_1^- (i = 1, …, n) rời nhau.

Gọi E^+ là tập các bằng chứng nằm trong cây con ở trên X, tức là E^+ gồm các bằng chứng được nói với X bởi đường vô hướng đi qua đỉnh cha Z của X (xem hình 10.4). Ta có:

$$E = E^+ \cup E^-$$

Các E^+ và E^- rời nhau.

Áp dụng công thức Bayes và luật tích, ta có :

$$Pr(X|E) = Pr(X|E^+, E^-) = \frac{Pr(E^-, X, E^+)}{Pr(E)}$$

$$= \frac{Pr(E^- \mid X, E^+)\, Pr(X \mid E^+)\, Pr(E^+)}{Pr(E)}$$

Khi đã cho X thì E^- độc lập với E^+, do đó $Pr(E^-|X, E^+) = Pr(E^-|X)$. Đặt tỷ số $Pr(E^+)/Pr(E)$ là hằng số α (vì nó không phụ thuộc vào X), chúng ta có:

$$Pr(X|E) = \alpha\, Pr(E^-|X)\, Pr(X|E^+) \qquad (1)$$

Trong công thức trên, chúng ta sẽ gọi $Pr(X|E^+)$ là hỗ trợ nguyên nhân cho X, còn $Pr(E^-|X)$ là hỗ trợ chẩn đoán cho X. Như vậy, để tính Pr(X|E) ta cần tính các xác suất $Pr(E^-|X)$ và $Pr(X|E^+)$.

- ***Tính $Pr(X|E^+)$***

Theo luật tổng, ta có :

$$Pr(X|E^+) = \sum_z Pr(X|Z = z, E^+)\, Pr(Z = z|E^+)$$

Tổng này lấy trên tất cả các giá trị z của biến Z. Khi đã cho Z thì X độc lập với E^+, $Pr(X|Z = z, E^+) = Pr(X|Z = z)$. Do đó ta có :

$$Pr(X|E^+) = \sum_z Pr(X|Z = z)\, Pr(Z = z|E^+) \qquad (2)$$

Trong công thức này, các xác suất $Pr(X|Z = z)$ đã được cho biết khi cho mang xác suất. Do đó việc tính các xác suất $Pr(X|E^+)$ được quy về việc tính xác suất Pr(Z|E). Các xác suất này lại được tính bằng cách áp dụng đệ quy công thức (1). Các xác suất $Pr(Z|E^+)$ được gọi là các hỗ trợ nhuyên nhân từ cha Z truyền xuống con X.

- ***Tính $Pr(E^-|X)$***

Ta có : $Pr(E^-|X) = Pr(E_1^-, \ldots, E_n^-|X)$. Khi đã cho X thì các E_i^- (i = 1, …, n) độc lập với nhau, do đó :

$$Pr(E^-|X) = Pr(E_1^-|X) \ldots Pr(E_n^-|X) \qquad (3)$$

Các xác suất $Pr(E_i^-|X)$ được gọi là các hỗ trợ chẩn đoán từ con của X truyền lên X. Như vậy, để tính hỗ trợ chẩn đoán cho X : $Pr(E^-|X)$, theo công thức (3) chúng ta cần tính các hỗ trợ chẩn đoán từ cây con của X truyền lên X : $Pr(E_i^-|X)$.

- ***Tính hỗ trợ chẩn đoán từ con truyền lên cha***

Để tránh phải đưa vào các ký hiệu mới, chúng ta tính hỗ trợ chẩn đoán từ con X truyền lên cha Z. chúng ta ký hiệu E_X^- là tập các bằng chứng nằm trên cây con gốc X. Khi đó hỗ trợ chẩn đoán từ X gửi lên cha Z là xác suất $Pr(E_X^-|\ Z)$. Rõ ràng nếu X không phải là biến bằng chứng thì $E_X^- = E^-$ (xem hình vẽ 10.4), còn nếu X là biến bằng chứng và $X = x_0$ thì $E_X^- = (E^-, X = x_0)$.

Như vậy, nếu X không phải là biến bằng chứng thì

$$Pr(E_X^-|Z) = Pr(E^-|\ Z)$$

Ta có:

$$Pr(E^-|X) = \sum_z Pr(E^-, X = x|Z)$$

$$Pr(E^-, X = x|Z) = Pr(E^-|X = x, Z)\, Pr(X = x|Z)$$

Khi đã cho X thì E^- độc lập với Z, do đó:

$$Pr(E^-, X = x|Z) = Pr(E^-|X = x) = Pr(E_1^-, \ldots, E_n^-|X = x)$$

Nhưng các E_i^- cũng độc lập với nhau, khi cho trước X. Do đó :

$$\Pr(E_1^-, \ldots, E_n^-|X = x) = \prod_{i=1}^{n} \Pr(E_i^-|X = x)$$

Như vậy ta có :

$$\Pr(E_X^-|Z) = \Pr(E^-|\ Z) = \sum_{z} \Pr(X = x|Z) \prod_{i=1}^{n} \Pr(E_i^-|X = x) \qquad (4)$$

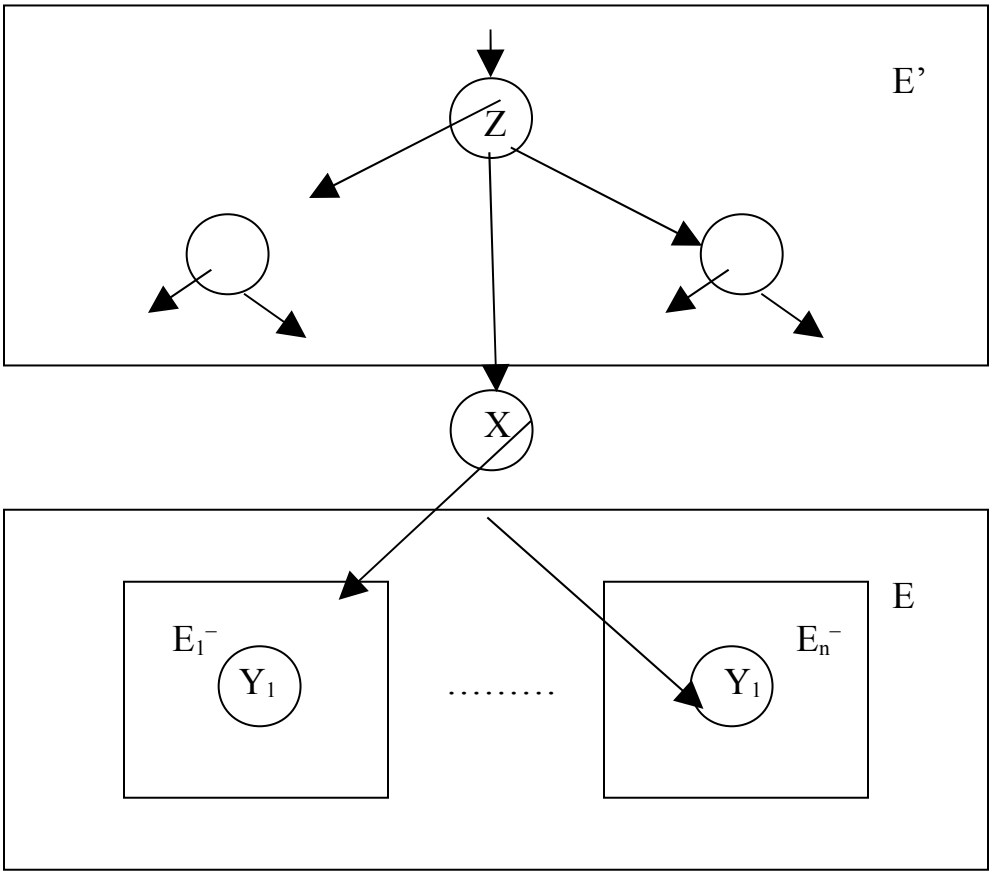
Như vậy, nếu X không phải là biến bằng chứng thì hỗ trợ chẩn đoán từ X gửi lên cha nó Z là xác suất $\Pr(E^-|\ Z)$, và được quy về việc tính các hỗ trợ chẩn đoán từ các con của X gửi lên X, tức là các xác suất $\Pr(E_i^-|X)$ $(i = 1, \ldots, n)$.

Nếu X là biến bằng chứng và $X = x_0$ thì

$$\Pr(E_X^-|Z) = \Pr(E^-, X = x_0|Z)$$

Do đó, nếu $X = x_0$ là bằng chứng thì

$$\Pr(E_X^-|Z) = \Pr(E^-|\ Z) = \Pr(X = x_0|Z) \prod_{i=1}^{n} \Pr(E_i^-|X = x_0) \qquad (5)$$



Hình 10.4. **Mạng có cấu trúc cây.**

SỰ TRUYỀN BẰNG CHỨNG TRONG MẠNG CÂY

Từ các lập luận trên, chúng ta suy ra rằng, các bằng chứng được truyền trong mạng cây như sau.

- Các hỗ trợ nguyên nhân được truyên từ các cha xuống các con, bắt đầu từ gốc cây và dừng lại tại biến hỏi.

  Cần lưu ý rằng, nếu A là gốc cây thì tập các bằng chứng ở trên A là tập rỗng. Do đó hỗ trợ nguyên nhân cho A là Pr(A| ∅) = Pr(A). Phân phối xác suất Pr(A) đã được cho.

- Các hỗ trợ chẩn đoán được truyền từ các con lên các cha, bắt đầu từ các lá và truyền lên cho tới gốc cây.

Nếu L là lá thì tập các bằng chứng nằm dưới L là tập rỗng. Do đó hỗ trợ chẩn đoán cho L là Pr(∅ | L) = 1 với mọi giá trị của biến ngẫu nhiên L.

Sau đây là hàm đề quy tính hỗ trợ chẩn đoán từ con X gửi lên cha Z hàm Pr(E_X^-|Z).

Function *Pr(E_X |Z)*;

begin

1. Tính hỗ trợ chẩn đoán từ các con của X truyền lên X:

Tính Pr (E_x^-|X) với I = 1, …, n;

2. Nếu X = x_0 là bằng chứng thì

$$Pr(E_x^- \mid Z) = Pr(X = x_0 \mid Z) \prod_{i=1}^{n} Pr(E_i^- \mid X = x_0)$$

3. Nếu X không phải là biến bằng chứng thì

$$\Pr(E_x^- \mid Z) = \sum_x \Pr(X = x \mid Z) \prod_{i=1}^{n} \Pr(E_i^- \mid X = x)$$

end;

Và sau đây là hàm đệ quy xác định phân phối xác suất Pr(X|E)

Function *Pr(X|E)*;

begin

1. Nếu $X = x_0$ là bằng chứng thì:

$\Pr(X = x_0 \mid E) = 1$

$\Pr(X = x \mid E) = 0$ với $x \neq x_0$;

2. Nếu X không phải là biến bằng chứng thì

2.1. Tính hỗ trợ nguyên nhân từ cha Z truyền xuống X:
Tính $\Pr(Z|E^+)$, xác suất này được tính bằng cách gọi đệ quy hàm này;

2.2. Tính các hỗ trợ chẩn đoán từ các con của X truyền lên X:
Tính $\Pr(E_i^- \mid X)$ với i = 1, …, n;

2.3. Tính hỗ trợ nguyên nhân cho X

$$\Pr(X \mid E^+) = \sum_z \Pr(X \mid Z = z) \Pr(Z = z \mid E^+);$$

2.4. Tính hỗ trợ chẩn đoán cho X

$$\Pr(E^- \mid X) = \prod_{i=1}^{n} \Pr(E_i^- \mid X);$$

2.5. Tính $\Pr(X \mid E) = \alpha \Pr(E^- \mid X) \Pr(X \mid E^+)$;

end;

Chú ý rằng, khi tính phân phối xác suất Pr(X|E) bởi hàm đệ quy trên, chúng ta kết hợp tất cả các nhân tử hằng thành nhân tử λ, nhân tử còn lại là p(X):

$$\Pr(X|E) = \lambda\, p(X)$$

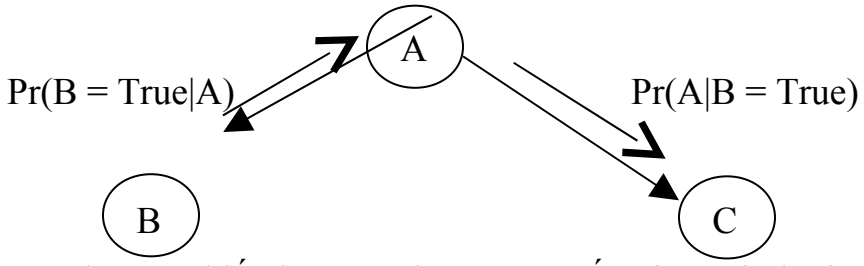
Lưu ý rằng:

$$\sum_x \Pr(X = x \mid E) = 1$$

Nên khi đã tính được p(X), chúng ta sẽ xác định được

$$\lambda = \frac{1}{\sum_{x} p(x)}$$

Ví dụ. Giả sử phòng làm việc của An trong cơ quan ở trên tầng 3. Khi An ở trong phòng làm việc của mình, An có thể để đèn sang hoặc không, An có thể nhập dữ liệu vào máy tính hoặc An ngồi làm việc ở bàn viết. Khi không ở trong phòng của mình, đôi khi cần thiết An vẫn có thể nhập dữ liệu vào máy tính của mình thong qua đường truyền thông. Chúng ta ký hiệu A là sự kiện "An ở trong phòng làm việc của mình", B là sự kiện "đèn trong phòng An sáng" và C là sự kiện "An nhập dữ liệu vào máy tính". Chúng ta có mạng xác suất dạng cây như sau:



Giả sử chúng ta biết được các bảng xác suất tại các đỉnh như sau:

Pr(A)	Pr(¬A)
0,4	0,6

	Pr(B|A)	
	B = True	B = False
A = True	0,6	0,4
A = False	0,1	0,9

A = False	0,3	0,7

	Pr(C|A)	
	C = True	C = False
A = True	0,8	0,2

Giả sử chúng ta nhìn thấy đèn trong phòng An sáng, chúng ta muốn biết xác suất "An nhập dữ liệu vào máy tính". Tức là chúng ta cần tính Pr(C=True|B=True). Trong trường hợp này tập bằng chứng là E= {B=True}.

Vì C là lá, nên hỗ trợ chẩn đoán cho C là 1. Chúng ta cần tính hỗ trợ nguyên nhân từ cha A truyền xuống C: Pr(A|B=True).

Hỗ trợ nguyên nhân cho A là Pr(A) vì A là gốc. Hỗ trợ chẩn đoán từ B truyền lên A là Pr(B=True|A). Do đó hỗ trợ nguyên nhân từ A truyền xuống C là:

$$Pr(A|B=True) = \alpha\ Pr(A)\ Pr(B=True|A)$$

Từ công thức này, thay A = True và A = False, chúng ta tính được

$$Pr(A=True|B=True) = \alpha\ .0{,}4.0{,}6 = \alpha\ .0{,}24$$

$$Pr(A = False|B = True) = \alpha\ .0{,}6.0{,}1 = \alpha\ .0{,}06$$

Do đó chúng ta có

$$Pr(C|B=True) = \beta\ (Pr(C|\ A=True)\ Pr(A=True|\ B=True) + Pr(C|\ A=False)\ Pr(A=False|\ B=True))$$

Thay C = True và C = False vào công thức trên và đặt $\lambda = \alpha\beta$, chúng ta có

$$Pr(C = True|B = True) = \lambda\ (0{,}8\ .\ 0{,}24 + 0{,}3\ .\ 0{,}06) = \lambda\ .\ 0{,}21$$

$$Pr(C = False|B =True) = \lambda\ (0{,}2\ .\ 0{,}24 + 0{,}7\ .\ 0{,}06) = \lambda\ .\ 0{,}09$$

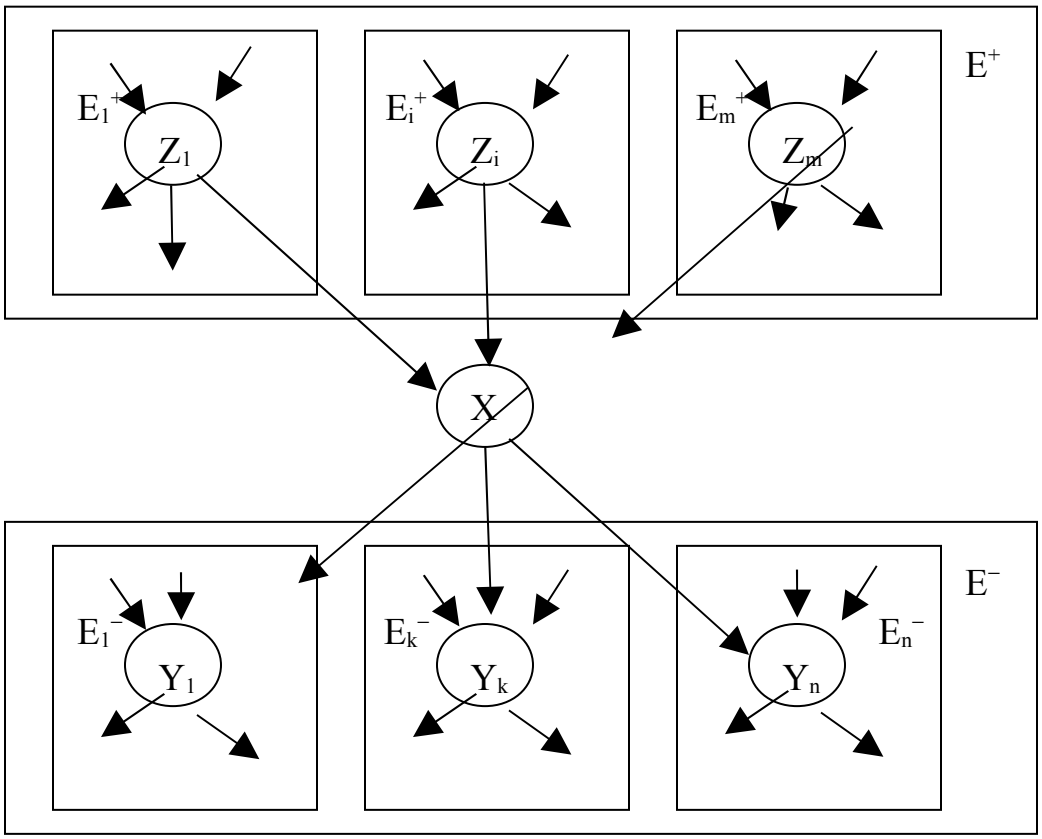
Nhưng $\lambda\ .\ 0{,}21 + \lambda\ .\ 0{,}09 = 1$, do đó $\lambda = 10/3$.

Vậy $Pr(C=True|\ B=True) = \lambda\ .\ 0{,}21 = 10/3\ .\ 0{,}21 = 0{,}7$

Mạng cây là trường hợp riêng của mạng kết nối đơn được trình bày ở mục sau. Chúng tôi trình bày suy diễn trong mạng cây để độc giả làm quen với phương pháp suy diễn. Bạn đọc có thể bỏ qua mục này, đọc ngay vào mục sau.

10.5. MẠNG KẾT NỐI ĐƠN

Trong mục này chúng ta sẽ xét vấn đề suy diễn trong *mạng kết nối đơn* (simply connected network). Đó là mạng mà giữa hai đỉnh bất kỳ tồn tại nhiều nhất một đường vô hướng nối chúng. Hình 10.5 biểu diễn cấu trúc mạng kết nối đơn.



***Hình 10.5*. Mạng kết nối đơn.**

Giả sử bằng chứng là E và biến hỏi là X. Mục đích của chúng ta là tính phân phối xác suất Pr(X|E).

Chúng ta gọi E^+ là tập các bằng chứng "nằm trên X", tức là các bằng chứng được nối với X qua các cha Z_i (i = 1, …, m) của X (xem hình 10.5). Trong E^+, chúng ta gọi E_i^+ (i = 1, …, m) là tập các bằng chứng nối với X qua cha Z_i. Do tính kết nối đơn, chúng ta có các E_i^+ rời nhau và $E^+ = E_1^+ \cup \ldots \cup E_m^+$.

Tương tự, chúng ta gọi tập các bằng chứng "nằm dưới X" là E^-. Ký hiệu E_k^- là tập các bằng chứng được nối với X qua con Y_k (k = 1, .., n) của X. Cũng do tính kết nối đơn, các E_k^- rời nhau và $E^- = E_1^- \cup \ldots \cup E_n^-$.

Tương tự như trong trường hợp mạng cây, chúng ta có

$$Pr(X|E) = \alpha\, Pr(E^-|X)\, Pr(X|E^+)$$

- ***Tính hỗ trợ nguyên nhân cho X: $Pr(X|E^+)$***

Ký hiệu Z là vectơ các biến ngẫu nhiên $Z = (Z_1, \ldots, Z_m)$ và z là vectơ các giá trị $z = (z_1, \ldots, z_m)$, trong đó z_i (i = 1, …, m) là một giá trị bất kỳ của Z_i. Ta sẽ viết Z = z thay cho $(Z_1 = z_1, \ldots, Z_i = z_i, \ldots, Z_m = z_m)$.

Chúng ta có

$$Pr(X|E^+) = Pr(X|Z = z, E^+)\, Pr(Z = z|E^+) \qquad (1)$$

Cần nhớ rằng, sau này khi tính các xác suất có điều kiện, chúng ta sẽ dựa vào tính tách được của các tập biến ngẫu nhiên để suy ra sự độc lập của chúng. Chẳng hạn, vì các biến $Z_1, \ldots, Z_m$ tách X với E^+, nên khi đã cho $Z = (Z_1, \ldots, Z_m)$ thì X độc lập với E^+. Do đó

$$Pr(X|Z = z, E^+) = Pr(X \mid Z = z) \qquad (2)$$

E^+ tách với Z_i với Z_j ($j \neq i$), nên cho trước E^+ thì các Z_i (i=1,…,m) độc lập với nhau và do đó chúng ta có

$$Pr(X|Z = z, E^+) = Pr(Z_1 = z_1, \ldots, Z_i = z_i, \ldots, Z_m = z_m|E^+)$$

$$= \prod_{i=1}^{m} Pr(Z_i = z_i|E^-)$$

Chúng ta tính $Pr(Z_i = z_i| E^+)$. Ta có

$$Pr(Z_i = z_i| E^+) = Pr(Z_i = z_i| E_1^+, \ldots, E_i^+, \ldots, E_m^+)$$

E_i^+ tách Z_i với các E_j^+ ($i \neq j$), do đó Z_i độc lập với E_j^+ ($i \neq j$) khi cho trước E_i^+. Từ công thức trên ta có

$$Pr(Z_i = z_i| E^+) = Pr(Z_i = z_i| E_i^+)$$

Như vậy ta nhận được

$$Pr(Z = z| E^+) = \prod_{i=1}^{m} Pr(Z_i = z_i| E_i^+) \qquad (3)$$

Thay công thức (2) và (3) vào (1), ta có

$$Pr(Z = z| E^+) = \sum_{z} Pr(X \mid Z = z) \prod_{i=1}^{m} Pr(Z_i = z_i| E_i^+) \qquad (4)$$

Trong thức (4), các xác suất Pr(X|Z = z) đã được cho, các xác suất $Pr(Z_i| E_i^+)$ được gọi hỗ trợ nguyên nhân từ cha $Z_i$truyền xuống con X. Như

vậy để tính hỗ trợ nguyên nhân cho X : $Pr(X|E^+)$, chúng ta cần tính các hỗ trợ nguyên nhân từ các cha của X truyền xuống X : $Pr(Z_1| E_1^+)$.

- ***Tính hỗ trợ chẩn đoán cho X: $Pr(E^- |X)$***

Hoàn toàn giống như trường hợp mạng cây, chúng ta có:

$$Pr(E^- | X) = \prod_{k=1}^{n} Pr(E_k^- | X)$$

Chúng ta cũng gọi các xác suất $Pr(E_k^-|X)$ là hỗ trợ chẩn đoán từ đỉnh Y_k gửi lên cha X. Như vậy việc tính hỗ trợ chẩn đoán cho X được quy về việc tính các hỗ trợ chẩn đoán từ các con của X gửi lên X.

- ***Tính hỗ trợ chẩn đoán từ một đỉnh gửi lên một đỉnh cha nó***

Để khỏi phải đưa vào các ký hiệu mới, thay vì đi tính hỗ trợ chẩn đoán từ Y_k gửi lên cha X, chúng ta tính hỗ trợ chẩn đoán từ X gửi lên cha Z_1, tức là các xác suất $Pr(E_X^-|Z_i)$, trong đó E_X^- nhận từ tập bằng chứng E bằng cách loại bỏ tất cả các bằng chứng được nối với X bởi đường vô hướng qua Z_i.

Nếu X không phải là biến bằng chứng thì

$$E_X^- = (E_1^+,\ldots, E_{i-1}^+, E_{i+1}^+,\ldots, E_m^+, E_1^-,\ldots, E_n^-)$$

Nếu X là một biến bằng chứng và $X = x_o$ thì

$$E_X^- = (E_1^+, \ldots, E_{i-1}^+, E_{i+1}^+,\ldots E_m^+ , E_1^-, \ldots,E_n^-, X = x_o)$$

1. Tính $Pr(E_X^-|Z_i)$ Khi X không phải là biến bằng chứng.

Chúng ta có

$$Pr(E_X^-|Z_i) = \sum_{x,\, z_j\ (j \neq i)} Pr(E_X^-,X = x,Z_1 = z_1,\ldots,Z_{i-1} = z_{i-1},Z_{i+1} = z_{i+1},\ldots,Z_m = z_m,Z_j)$$

Tổng này lấy tren tát cả các giá trị x của X và các giá trị z_j của Z_j ($j \neq i$).

Theo luật tích, ta có

$$Pr(E_X^-, X = x, Z_1 = z_1, \ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, \ldots, Z_m = z_m|Z_j)$$

$$= Pr(E_X^-|X = x, Z_1 = z_1, \ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, \ldots, Z_m = z_m)*$$

$$Pr(X=x , Z_1 = z_1, \ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, \ldots, Z_m = z_m| Z_i)(6)$$

Chúng ta đi tính nhân tử đầu tiên của tích ở vế phải của công thức trên.

Khi đã cho X và $Z_1, \ldots, Z_m$ thì các E_j^+ (j = 1, …, m) độc lập với các E_k^- (k = 1, …, n). Do đó

$$\Pr(E_X^-|X = x, Z_1 = z_1, \ldots, Z_i = z_i, \ldots, Z_m = z_m)$$

$$= \Pr(E_1^+, \ldots, E_{i-1}^+, E_{i+1}^+, \ldots, E_m^+|X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)*$$

$$\Pr(E_1^-, \ldots, E_n^-|X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m) \quad (7)$$

Khi đã cho $Z_1, \ldots, Z_m$ thì X độc lập với các E_j^+ (j = 1, …, m) và các E_j^+ độc lập với nhau. Do đó chúng ta có

$$\Pr(E_1^+, \ldots, E_{i-1}^+, E_{i+1}^+, \ldots, E_m^+|X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

$$= \prod_{j \neq i} \Pr(E_j^+|Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

Mặt khác, E_j^+ độc lập với Z_i ($i \neq j$) khi đã cho Z_j, do đó

$$\Pr(E_j^+|Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

Áp dụng công thức Bayes, ta có

$$\Pr(E_j^+|Z_j = z_j) = \frac{\Pr(Z_j = z_j \mid E_j^+)\Pr(E_j^+)}{\Pr(Z_j = z_j)}$$

Từ các công thức trên, ta nhận được

$$\Pr(E_1^+, \ldots, E_{i-1}^+, E_{i+1}^+, \ldots, E_m^+|X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

$$= \beta \frac{\prod_{j \neq i} \Pr(Z_j = z_j \mid E_j^+)}{\prod_{j \neq i} \Pr(Z_j = z_j)} \quad (8)$$

trong đó hằng số β thay cho

$$\prod_{j \neq i} \Pr(E_j^+)$$

Chúng ta tính nhân tử thứ hai của tích ở vế phải của công thức (7).

Chúng ta có nhận xét rằng, khii đã cho X thì các E_k^- (k = 1, …, n) độc lập với nhau và độc lập với các Z_j (j = 1, …, m). Do đó

$$\Pr(E_1^-, \ldots, E_n^-|X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

$$= \prod_{k=1}^{n} \Pr(E_k^- | X = x) \qquad (9)$$

Thay công thức (8) và (9) vào (7), ta có

$$\Pr(E_X^- | X = x, Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

$$= \beta \frac{\prod_{j \neq i} \Pr(Z_j = z_j \mid E_j^+) \prod_{k=1}^{n} \Pr(E_k^- \mid X = x)}{\prod_{j \neq i} \Pr(Z_j = z_j)} \qquad (10)$$

Bây giờ chúng ta tính nhân tử thứ hai của tích ở vế phải của công thức (6). Theo luật tích, ta có

$$\Pr(X=x , Z_1 = z_1,\ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, ..., Z_m = z_m| Z_i)$$

$$= \Pr(X=x|Z_1 = z_1,\ldots, Z_i, ..., Z_m = z_m)*$$

$$\Pr(Z_1 = z_1,\ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, ..., Z_m = z_m| Z_i)$$

Nhưng các Z_j ($j \neq i$) là độc lập với nhau, khi đã cho Z_i. Do đó

$$\Pr(X = x , Z_1 = z_1,\ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, ..., Z_m = z_m)$$

$$= \Pr(X = x|Z_1 = z_1,\ldots, Z_i, ..., Z_m = z_m) \prod_{j \neq i} \Pr(Z_j = z_j) \qquad (11)$$

Thay công thức (10) và (11) vào (6), ta có

$$\Pr(E_X^- | X=x, Z_1 = z_1, \ldots, Z_{i-1} = z_{i-1}, Z_{i+1} = z_{i+1}, \ldots, Z_m = z_m | Z_i)$$

$$= \beta \Pr(X = x|Z_1 = z_1,\ldots, Z_i, ..., Z_m = z_m)*$$

$$\prod_{j \neq i} \Pr(Z_j = z_j|E_j^+) \prod_{k=1}^{n} \Pr(E_k^-|X = x)$$

Từ công thức này chúng ta nhận được công thức tính hỗ trợ chẩn đoán từ X gửi lên cha Z_i.

$$\Pr(E_x^-|Z_i) = \beta \sum_{x,\, z_j\, (j \neq i)} \Pr(X = x|Z_1 = z_1, \ldots, Z_i, ..., Z_m = z_m)*$$

$$\prod_{j \neq i} \Pr(Z_j = z_j) \prod_{k=1}^{n} \Pr(E_k^-|X = x) \qquad (12)$$

2. Tính $Pr(E_X|Z_i)$ khi $X = x_0$ là một bằng chứng

Bằng các tính toán như trong trường hợp X không phải là biến bằng chứng và lưu ý rằng trong trường hợp này E_X^- giống như trường hợp cho trước được bổ sung thêm bằng chứng $X = x_0$, chúng ta suy ra

$$Pr(E_x^-|Z_i) = \beta \sum_{z_j\,(j \neq i)} Pr(X = x|Z_1 = z_1, \ldots, Z_i, \ldots, Z_m = z_m)$$

$$\prod_{k=1}^{n} Pr(Z_j = z_j) \prod_{k=1}^{n} Pr(E_k^-|X = x) \qquad (13)$$

Trong các công thức (12) và (13), các xác suất Pr(X|Z) đã được cho. Như vậy để tính hỗ trợ chẩn đoán từ đỉnh X gửi lên cha Z_i, chúng ta cần tính các hỗ trợ nguyên nhân từ các cha Z_j ($j \neq i$) gửi xuống X: $Pr(Z_j|E_j^+)$, và cần phải tính các hỗ trợ chẩn đoán từ các con Y_k ($k = 1, \ldots, n$) gửi lên cha Z : $Pr(E_k^-|X)$.

Từ các công thức đã tìm ra, chúng ta suy ra các hàm đệ quy sau đây.

Hàm đệ quy tính hỗ trợ chẩn đoán từ con X gửi lên cha Z_i.

Function *$Pr(E_X^-|Z_i)$*:

begin

1. Tính các hỗ trợ nguyên nhân từ các cha Z_i truyền xuống con X:

   Tính $Pr(Z_i|E_j^+)$ với $j = 1, \ldots, m$ và $j \neq i$;

   Các xác suất này được tính bởi hàm đệ quy Pr(X|E)

2. Tính các hỗ trợ chẩn đoán từ các con Y_k truyền lên X:

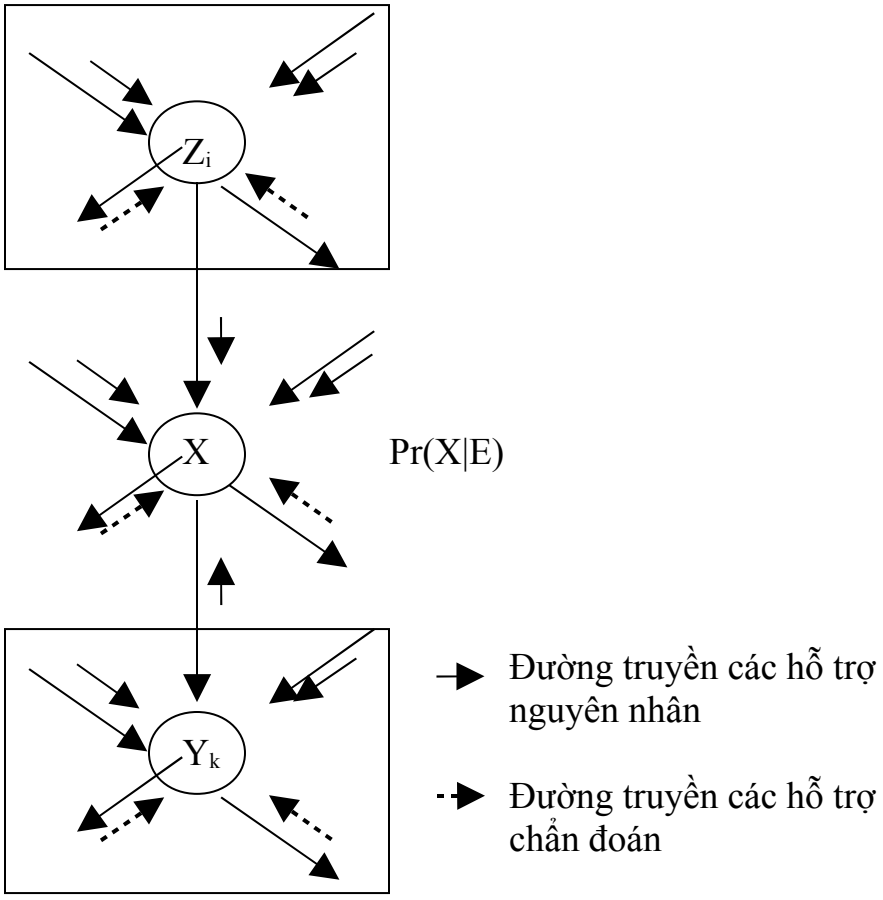
   Tính $Pr(E_k^-|X)$ với $k = 1, \ldots, n$;

   Các xác suất này được tính bằng gọi đệ quy hàm này;

3. Nếu X không là biến bằng chứng thì tính $Pr(E_X^-|Z_i)$ theo công thức (12).

4. Nếu $X = x_0$ là một bằng chứng thì tính $Pr(E_X^-|Z_i)$ theo công thức (13).

end;



***Hình 10.6*. Sự truyền các hỗ trợ nguyên nhân và hỗ trợ chẩn đoán**

Và sau đây là hàm đệ quy tính phân phối xác suất của biến hỏi X, khi cho tập bằng chứng E, hàm Pr(X|E).

Function *Pr(X|E):*

begin

1. Nếu $X = x_0$ là một bằng chứng thì

   $Pr(X = x_0|E) = 1$;

   $Pr(X = x|E) = 0$ với mọi $x \neq x_0$;

2. Nếu X không phải là biến bằng chứng thì

   2.1. Tính các hỗ trợ nguyên nhân từ các cha Z_i truyền xuống X:

   Tính $Pr(Z_i|E_i^+)$ với i = 1, …, m;

   Các xác suất này được tính bằng gọi đệ quy hàm này;

   2.2. Tính hỗ trợ nguyên nhân cho X:

$$\Pr(X|E^+) = \sum_z \Pr(X|Z = z) \prod_{i=1}^{m} \Pr(Z_i = z_i|E_i^+)$$

2.3. Tính các hỗ trợ chẩn đoán từ các con Y_k truyền lên X:

Tính $\Pr(E_k^-|X)$ với k = 1, …, n;

Các xác suất này được tính theo hàm đệ quy $\Pr(E_x^-|Z_i)$;

2.4. Tính hỗ trợ chẩn đoán cho X:

$$\Pr(E^-|X) = \prod_{k=1}^{n} \Pr(E_k^-|x)$$

2.5. Tính $\Pr(X|E) = \alpha \Pr(E^-|X) \Pr(X|E^+)$

end;

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN SUY DIỄN TRONG MẠNG KẾT NỐI ĐƠN

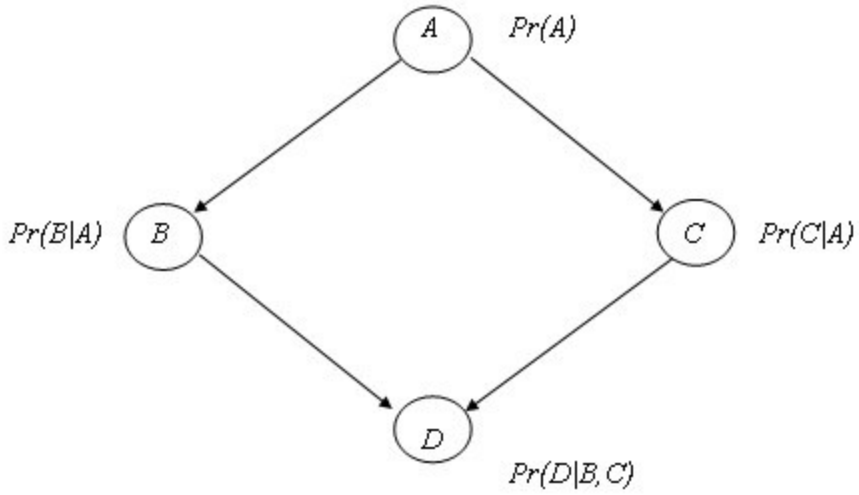
Thuật toán tính phân phối xác suất của biến hỏi X khi được cho tập các bằng chứng E, Pr(X|E), bao gồm các lời gọi đệ quy tính các hỗ trợ nguyên nhân và các hỗ trợ chẩn đoán. Các lời đệ quy này lan rộng ra từ đỉnh X theo tất cả các đường trong mạng. hình 10.6 mô tả sự truyền các hỗ trợ nguyên nhân và các hỗ trợ chẩn đoán. Các lời gọi các tính hỗ trợ nguyên nhân sẽ dừng lại tại các đỉnh gốc (các đỉnh không có cha). Các lời gọi đệ quy tính hỗ trợ chẩn đoán sẽ dừng lại tại các đỉnh lá (các đỉnh không có con). Do tính kết nối đơn, mỗi đỉnh trong mạng có mặt trong một lời gọi đệ quy chỉ một lần. Do đó thuật toán tính Pr(X|E) là tuyến tính với đỉnh số trong mạng.

10.6. SUY DIỄN TRONG MẠNG ĐA KẾT NỐI

10.6.1. Suy diễn trong mạng đa kết nối

Trong mục này chúng ta sẽ xé vấn đề suy diễn trong mạng đa kết nối. Mạng đa kết nối là mạng mà giữa hai đỉnh có thể được nói với nhau bởi nhiều hơn một đường vô hướng. trường hợp này có thể xảy ra khi có hai hoặc nhiều đỉnh ảnh hưởng đến nhiều đỉnh khác, và các đỉnh nguyên nhân đó có chung một đỉnh nguyên nhân chung.

Ví dụ. chúng ta có mạng đa kết nối được cho trong hình 10.7.



Hình 10.7. Một mạng đa kết nối.

Sau đây chúng ta sẽ trình bày một số suy diễn quan trọng trong mạng đa kết nối.

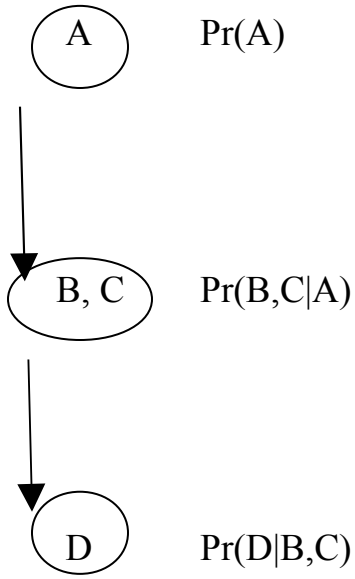
10.6.2. Biến đổi mạng đa kết nối thành mạng kết nói đơn

Tư tưởng của phương pháp này là kết nối một số đỉnh của mạng đa kết nối thành một đỉnh mới (đỉnh này được gọi là đỉnh kết hợp) sao cho mạng đã cho được chuyển thành mạng kết nối đơn. chẳng hạn, trong mạng hình 10.7, nếu ta kết hợp hai đỉnh B và C thành đỉnh mới ký hiệu là (B,C) thì mạng được chuyển thành mạng kết nối đơn trong hình 10.8. Cần lưu ý rằng, mỗi đỉnh kết hợp là một vectơ các biến ngẫu nhiên. Dựa vào các bảng xác suất trong mạng đa kết nối, chúng ta có thể dễ dàng tính được phân phối xác suất tại các đỉnh kết hợp. chẳng hạn, trong mạng hình 10.8, ta có thể tính được phân phối xác suất Pr(B,C|A) theo Pr(B|A) và Pr(C|A) đã được cho trong mạng hình 10.7. thật vậy,

$$Pr(B,C|A) = Pr(B|C,A)\ Pr(C|A)$$

Nhưng B và C độc lập khi đã cho A, do đó

$$Pr(B,C|A) = Pr(B|A)\ Pr(C|A)$$

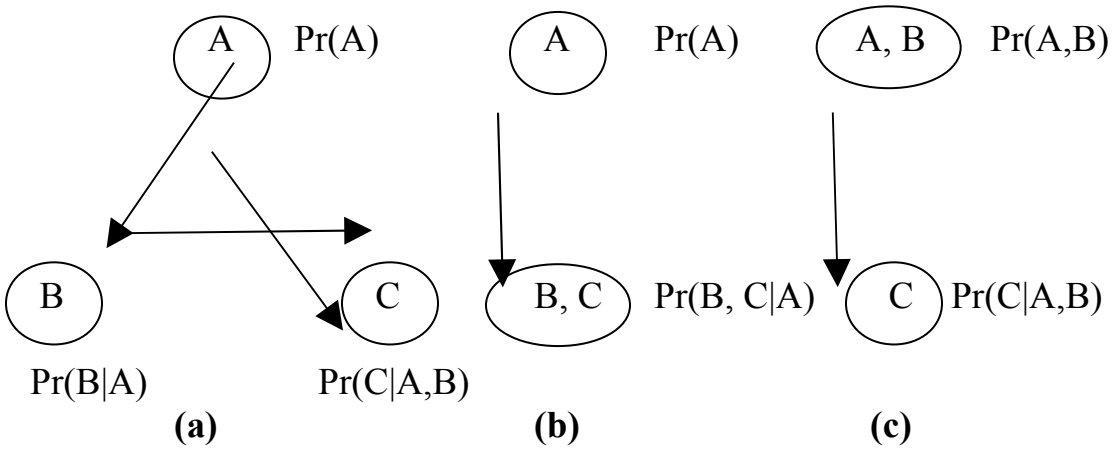


***Hình 10.8*. Mạng kết nối đơn ứng với mạng trong hình 10.7.**

Khi đã chuyển một mạng đa kết nối thành mạng kết nối đơn bằng cách tạo ra các đỉnh kết hợp, chúng ta có thể sử dụng thuật toán suy diễn với thời gian tuyến tính trong mạng kết nối đơn đã được trình bày trong mục 10.5. Tuy nhiên khi gộp một số đỉnh thành đỉnh mới, chúng ta phải tính bảng phân phối xác suất của đỉnh kết hợp. Nếu đỉnh kết hợp chứa n đỉnh thành phần $A_1, \ldots, A_n$ thì bảng xác suất có điều kiện của đỉnh $(A_1, \ldots, A_n)$ cần chứa 2^n số (giả sử các A_i là các biến ngẫu nhiên boolean). Điều đó có nghĩa là, để chuyển mạng đa kết nối thành mạng kết nối đơn chúng ta cần khối lượng tính toán tăng theo hàm mũ với số đỉnh trong mạng. Do đó phương pháp này vẫn đòi hỏi thời gian mũ.

Đương nhiên là từ một mạng đa kết nối, chúng ta có nhiều cách chọn các đỉnh để gộp lại thành đỉnh kết hợp. Chẳng hạn, từ mạng đa kết nối trong hình 10.9a, chúng ta có thể biến đổi thành các mạng kết nối đơn trong hình 10.9b và 10.9c. Trong mạng 10.9b, bảng xác suất Pr(B, C|A) được tính theo công thức Pr(B, C|A) = Pr(C|B, A) Pr(B|A). Còn trong mạng 10.9c, Pr(A, B) = Pr(B|A) Pr(A).

Phương pháp kết hợp các đỉnh để chuyển mạng đa kết nối thành mạng kết nối đơn, hiện nay vẫn là một phương pháp hiệu quả nhất để tính chính xác phân phối xác suất có điều kiện trong mạng đa kết nối.



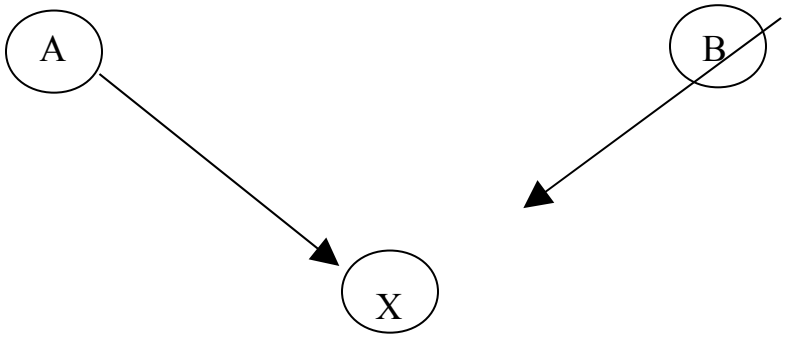
***Hình 10.9*. Biến đổi mạng đa kết nối thành các mạng kết nối đơn.**

10.6.3. Phương pháp mô phỏng ngẫu nhiên

Vấn đề tính chính xác xác suất của một biến khi cho trước các bằng chứng nào đó là vấn đề NP –khó. Tuy nhiên trong các ứng dụng, trong nhiều trường hợp chúng ta chỉ cần biết các giá trị xấp xỉ là đủ.

Trong mục này chúng ta sẽ trình bày các phương pháp cho phép ta tính xấp xỉ xác suất của một biến khi cho trước các bằng chứng trong mạng đa kết nối. Các phương pháp này được gọi là các phương pháp mô phỏng ngẫu nhiên (stochastic simulation methods). Các phương pháp này đóng vai trò quan trọng trong các ứng dụng.

Tư tưởng của phương pháp mô phỏng ngẫu nhiên là như là như sau. Chúng ta sẽ gắn giá trị cho các biến trong mạng tương ứng với phân phối xác suất gán với mỗi đỉnh trong mạng. Một đỉnh sẽ được gán giá trị. Chẳng hạn, giả sử X có hai đỉnh cha là A và B đã được gán giá trị.



Giả sử X, A,B đều là các biến ngẫu nhiên Boolean và A đã được gán True. B đã được gán False và Pr(X = true| A = true, B = false) = p. Việc gán

giá trị cho X được thực hiện như sau. Chúng ta sinh ra một số ngẫu nhiên q, $0 \le q \le 1$. nếu $q < p$ thì X được gán true , còn nếu $q \ge p$ thì X được gán false.

Trong trường hợp X, A và B không phải là các biến ngẫu nhiên Boolean, chúng ta có thể tiến hành như sau. Giả sử A đã được gán giá trị a, B đã được gán giá trị b . Giả sử X là biến ngẫu nhiên có thể nhận các giá trị $x_1, \ldots, x_n$. Giả sử

$$Pr(X=x_i|A = a, B = b) = p_i \qquad (i = 1,\ldots,n)$$

Cần lưu ý rằng $p_1 + \ldots + p_n = 1$. Chúng ta sử dụng phương pháp quay bánh xe (phương pháp này đã được sử dụng trong các giải thuật di truyền) để gán giá trị cho biến X. Sinh ra số ngẫu nhiên q, $0 \le q \le 1$. giả sử k là số nhỏ nhất sao cho $p_1 + \ldots + p_k \ge q$. Khi đó biến X được gán giá trị x_k.

Sử dụng thủ tục đã trình bày trên, chúng ta sẽ gán được giá trị cho tất cả các đỉnh trong mạng. lập lại thủ tục mô phỏng đó n lần, và giả sử trong n lần đó, biến X được gán giá trị x_k là n_k lần. Khi đó chúng ta có thể đánh giá $Pr(X = x_k) = n_k / n$.

Vấn đề của chúng ta bây giờ là: tính xấp xỉ phân phối xác suất Pr(X| E), trong đó E là tập các bằng chứng và X là một biến ngẫu nhiên bất kỳ trong mạng.

Trước hết, chúng ta xét các trường hợp biến bằng chứng là các biến không có cha. Trong trường hợp này mỗi lần thực hiện thủ tục mô phỏng ngẫu nhiên, các biến bằng chứng được gán giá trị là các bằng chứng. sau n lần lặp lại thủ tục mô phỏng ngẫu nhiên, nếu biến X nhận gía trị x_k với số lần là n_k thì chúng ta đánh giá xấp xỉ $Pr(X = x_k|E)$ là $n_k|n$.

Trong các thuật toán dưới đây, để đơn giản cho trình bày chúng ta giả sử rằng các biến trong mạng là các biến ngẫu nhiên Boolean. Chúng ta ký hiệu $p(X = x)$ là số lần mà biến X được gán giá trị x trong các lần lặp lại thủ tục mo phỏng ngẫu nhiên.

Procedure ***Stochastic – Simulation:***

begin

1. Đặt L là danh sách các đỉnh trong mạng sao cho nếu đỉnh A là cha của đỉnh B thì A phải đứng trước B trong danh sách;

2. Lặp lại các bước sau đây cho tới khi danh sách L rỗng.

   2.1. Loại biến X ở đầu danh sách L;

2.2. Nếu X là biến bằng chứng thì gán cho X giá trị là bằng chứng của X và quay lại 2.1;

2.3. Nếu X không có cha và Pr(X = true) = p thì

- Sinh ra số ngẫu nhiên $q \in [0,1]$;
- Nếu $q<p$ thì $X \leftarrow$ true và $\rho(X = true) \leftarrow \rho(X = true) + 1$;

  Nếu $q \geq p$ thì $X \leftarrow$ false và $\rho(X = false) \leftarrow \rho(X = false) + 1$;
- Quay lại 2.1;

2.4. Nếu X có các cha, chẳng hạn là A và B, giá trị đã gán cho A là a, giá trị đã gán cho B là b và Pr(X = true | A = a,B = b) = p, thì

- Sinh ra số ngẫu nhiên $q \in [0,1]$;
- Nếu $q<p$ thì $X \leftarrow$ true và $\rho(X = true) \leftarrow \rho(X = true) + 1$;

  Nếu $q \geq p$ thì $X \leftarrow$ false và $\rho(X = false) \leftarrow \rho(X = false) + 1$;
- Quay lại 2.1;

end;

Sau đây là thuật tính toán xấp xỉ Pr(X|E).

Thuật toán mô phỏng cho trường hợp các biến bằng chứng là các đỉnh không có cha:

1. Với mỗi biến ngẫu nhiên X trong mạng và mỗi giá trị x của X : $\rho(X = x) \leftarrow 0$;
2. Thực hiện thủ tục Stochastic – Simulation n lần đủ lớn:
3. Với mỗi biến X trong mạng và X không phải là biến bằng chứng và với mỗi giá trị x của X, lấy giá trị xấp xỉ của Pr(X = x|E) là $\rho(X =x) / n$;

Như vậy trong thuật toán mô phỏng trên, xác suất Pr(X=x|E) được tính gần đúng bằng tỷ số $\rho(X=x)|n$: tỷ số giữa lần biến X được gán giá trị x và số lần thực hiện mô phỏng. Tỷ số này sẽ hội tụ tới xác suất thực tế Pr(X=x|E) khi $n \rightarrow \infty$.

Mỗi lần thủ tục mô phỏng ngẫu nhiên được thực hiện, nó sẽ gán giá trị cho tất cả các ngẫu nhiên trong mạng. Một cách gán giá trị cho tất cả các

biến ngẫu nhiên trong mạng được gọi là một *mẫu* (sample). Trong trường hợp các biến bằng chứng đều không có cha, các mẫu được sinh ra bởi thủ tục mô phỏng ngẫu nhiên đều phù hợp với các bằng chứng.

Bây giờ chúng ta xét trường hợp các biến bằng chứng có thể có cha, tức là các biến bằng chứng có thể là đỉnh bất kỳ trong mạng. trong trường hợp này, mẫu sinh ra bởi thủ tục mô phỏng có thể không phù hợp với các bằng chứng, chẳng hạn, bằng chứng là X = a nhưng giá trị được gán cho X trong mẫu lại là X = b với b ≠ a. Có một cách đơn giản để khắc phục tình trạng này là , chúng ta sử dụng thủ tục mô phỏng ngẫu nhiên để sinh ra một số đủ lớn các mẫu, trong số các mẫu được sinh ra này, chúng ta sẽ loại bỏ đi tất cả các mẫu không phù hợp với các bằng chứng. Cách này có nhược điểm cơ bản là rất nhiều mẫu bị loại bỏ trong trường hợp các bằng chứng ít khi xảy ra. Do đó chúng ta phải thực hiện thủ tục mô phỏng ngẫu nhiên một số rất lớn lần.

THUẬT TOÁN TRỌNG SỐ KHẢ NĂNG

Trong thuật toán mô phỏng ở trên, vì là tất cả các mẫu được sinh ra bởi thủ tục mô phỏng ngẫu nhiên đều phù hợp với các bằng chứng, do đó tất cả các mẫu đều có cùng một trọng số là 1, và chúng ta đã lấy tỷ số giữa số mẫu trong đó X = x và tổng số mẫu làm giá trị xấp xỉ của Pr(X = x|E).

Trong thuật toán dưới đây, mỗi mẫu được cho một trọng số. Giá trị của X trong một mẫu được ký hiệu là Sample(X). Nếu A là một biến bằng chứng, thì chúng ta ký hiệu *e*(A) là bằng chứng của A (giá trị được gán cho A trong tập bằng chứng E).trọng số của mọt mẫu được xác định là

$$\prod_{X \in E} \Pr(X = e(X) | \text{parents}(X) = \text{Sample}(\text{Parents}(X)))$$

Mỗi khi biến X nhận giá trị x trong một mẫu thì X = x sẽ được cho một số điểm là trọng số của mẫu đó. Chúng ta sẽ tích luỹ một số điểm mà X = x nhận được trong các lần lặp lại thủ tục mô phỏng, số điểm đó được ký hiệu là δ (X =x). Khác với thuật toán mô phỏng trước đây, trong thuật toán trọng số khả năng (Likelihood – Weighting Algorithm), thay cho việc đếm số lần biến X nhận giá trị x, chúng ta tính số điểm tích luỹ δ (X = x).

Thủ tục được mô phỏng được trình bày dưới đây là hoàn toàn tương tự thủ tục mô phỏng trước đây, chỉ khác một điều là chúng ta tính số điểm tích luỹ δ(X = x) thay cho đếm số lần biến X nhận giá trị x.

Procedure ***Stochastic – Simulation;***

begin

1. Đặt L là danh sách các đỉnh trong mạng sao cho nếu đỉnh A là cha của đỉnh B thì A phải đứng trước B trong danh sách.
2. Lặp lại các bước sau đây cho tới khi danh sách L rỗng:
   2.1. Loại biến X ở đầu danh sách L;
   2.2. Nếu X là biến bằng chứng và X không có cha thì X được gán gía trị bằng chứng của X và quay lại bước 2. 1;
   2.3. Gán giá trị cho X như sau:
   - Giả sử p là xấc suất để X = true khi đã cho các giá trị đã gán cho cha của X:
   - Sinh ra một số ngẫu nhiên $q \in [0,1]$:
   - Nếu q< p thì X ← true :
   - Nếu q ≥ p thì ← false:
3. Tính số điểm tích luỹ:
   3.1. Tính trọng số của mẫu

$$w = \prod_{z \in E} \Pr(Z = e(Z) | \text{parents}(Z) = \text{Sample}(\text{Parents}(Z)))$$

   3.2. Với mỗi biến X trong mạng

$$\delta(X = \text{Sample}((X)) \leftarrow \delta(X = \text{Sample}(X)) + w;$$

end;

Thuật toán trọng số khả năng sẽ sử dụng lặp lại thủ tục mô phỏng trên để sinh ra một số đủ lớn các mẫu và tính số điểm cho X = x trong các mẫu đã được sinh ra. Sau đây là một số thuật toán trọng số khả năng.

1. Với mỗi biến X trong mạng và mỗi giá trị x của X, $\delta(X = x) \leftarrow 0$;
2. Thực hiện thủ tục mô phỏng trên một số lần đủ lớn;
3. Với mỗi biến X trong mạng và X không phải là biến bằng chứng, và với mỗi giá trị x của X, lấy giá trị gần đúng của Pr(X = x|E) là tỷ số $\delta(X = x) / \Sigma \delta(X = x')$, trong đó tổng lấy trên tất cả các giá trị x᾽ của biến X;

Các áp dụng thực tế đã chứng tỏ rằng thuật toán trọng số khả năng hội tụ nhanh hơn phương pháp lấy mẫu đơn giản. Thuật toán trọng số khả năng có thể làm việc rất tốt cho các mạng đa kết nối lớn và các mạng với xác suất của các bằng chứng gần 0 hoặc 1.

10.7. LÝ THUYẾT QUYẾT ĐỊNH

Trong mục này chúng ta sẽ xét vấn đề đưa ra quyết định trong môi trường với các thong tin không chắc chắn. Chúng ta xét ví dụ sau đây.

Ví dụ. Giả sử bạn cần đến cơ quan. Có hai phương án.

- *Phương án 1*: Bạn đi xe buýt mất 15 phút
- *Phương án 2*: Bạn dắt xe máy ra và khởi động. Nếu xe máy khởi động được thì đi xe máy và mất 10 phút. Nếu xe máy không khởi động được thì đi xe buýt. Giả sử thời gian dắt xe máy ra và khởi động mất 2 phút. Xã suất mà xe máy của bạn khởi động được là 0,8.

Bạn nên chọn phương án nào trong hai phương án trên? Muốn biết cần chọn phương án nào, chúng ta phải đánh giá các phương án (các kế hoạch). Trong phương án 2, khó khăn cho việc đánh giá là ở chỗ kết quả của hành động khởi động xe là không chắc chắn, xe có thể khởi động được hoặc không.

VẤN ĐỀ QUYẾT ĐỊNH

Chúng ta sẽ hiểu trạng thái (hoặc hoàn cảnh) là hình ảnh của thế giới mà ta quan tâm tại một thời điểm nào đó. Chúng ta sẽ mô tả trạng thái bởi các đặc trưng nào đó cần thiết cho vấn đè mà ta cần giải quyết. Hành động sẽ biến đổi thế giới từ trạng thái này thành trạng thái khác. Trong các vấn đề cần quyết định chúng ta cần quan tâm tới các hành động không đơn định. Đó là các hành động mà chúng ta chỉ biết rằng khi được thực hiện, nó sẽ dẫn đến mọt số trạng thái khác nhau, song trạng thái nào trong số các trạng thái đó được sinh ra, thì trước khi thực hiện hành động xhúng ta không thể biết được. Chẳng hạn, hành động "gọi tới số phone 8370496". Các trạng thái có thể có mà hành động này sinh ra là "đường dây bận", hoặc "không có ai ở đầu máy kia", hoặc "anh Hai cầm máy", hoặc "cô Ba cầm máy", … Hành động "gieo con xúc xắc" hoặc hành động "khởi động máy vi tính" đều là các hành động không đơn định.

Giả sử chúng ta đang ở trạng thái nào đó của thế giới (trạng thái ban đầu). Chúng ta cần thực hiện một dãy hành động để đạt được trạng thái

mong muốn nào đó (trạng thái đích). Dãy hành động như thế được gọi là một kế hoạch. Tại mỗi trạng thái của thế giới, giả sử chúng ta có thể thực hiện nhiều hành động khác nhau. Chúng ta phải quyết định chọn hành động nào trong số các hành động đó để sao cho dãy hành động đưa ta tới trạng thái đích với lợi ích mong đợi lớn nhất. Đó là vấn đề quyết định mà chúng ta xét trong mục này.

HÀM LỢI ÍCH

Giả sử A là hành động (không đơn định), nó có thể sinh ra các trạng thái $S_1,\ldots S_n$. Giả sử chúng ta biết được xác suất sinh ra trạng thái S_i ($i = 1, \ldots n$) khi chúng ta thực hiện hành động A trong các điều kiện E, xác suất này được ký hiệu là $Pr(S_i|A,E)$. Trong số các trạng thái $S_1,\ldots S_n$ đương nhiên là có trạng thái "có giá trị hơn", "tốt hơn" trạng thái khác. Hay nói cách khác, chúng ta ưa thích trạng thái này hơn trạng thái khác. Chúng ta sẽ gán cho mỗi trạng thái một số thực, số này được gọi là lợi ích của trạng thái (số đo lợi ích của trạng thái). Lợi ích của trạng thái S được ký hiệu là U(S). Hàm U được gọi là ***hàm lợi ích*** (utility function), nó ánh xạ tập các trạng thái vào tập các số thực.

Bây giờ chúng ta xác định lợi ích mong đợi (expected utility) của hành động A khi được thực hiện trong hoàn cảnh E là số thực EU(A|E). Số này được xác định bởi công thức

$$EU(A|E) = \sum_{i=1}^{n} Pr(S_i|A, E)\, U(S_i)$$

Trong vấn đế quyết định, chúng ta cần lựa chọn một dãy hành động để thực hiện sao cho dãy hành động đó sẽ dẫn tới một tập các trạng thái kết cục (outcomes) với lợi ích mong đợi lớn nhất có thể được. Đó chính là ***nguyên lý lợi ích mong đợi cực đại*** (principle of maximum expected utility).

Nguyên lý lợi ích mong đợi cực đại nói rằng, hành động hợp lý mà tác nhân cần phải lựa chọn là hành động làm cực đại lợi ích mong đợi của tác nhân.

Giả sử một trạng thái nào đó của thế giới, tác nhân cần đưa ra quyết định: chọn một trong các hành động A để thực hiện sao cho nó thu được lợi ích mong đợi lớn nhất. Muốn vậy chúng ta cần tính được các xác suất $Pr(S_i|A, E)$. Các xác suất này có thể tính dựa trên mô hình biểu diễn các mối quan hệ nhân-quả (mạng xác suất) mà chúng ta đã nghiên cứu trong các mục trước. Theo công thức tính lợi ích mong đợi EU(A|E), chúng ta còn phải tính các lợi ích $U(S_i)$ của các trạng thái S_i được sinh ra bởi hành động A. Nói

chung, chúng ta chỉ có thể biết được lợi ích của các trạng thái kết cục, còn lợi ích của trạng thái trung gian ta chưa xác định được, bởi vì khi chưa biết nó dẫn tới các trạng thái kết cục nào, ta chưa thể xác định được nó "tốt" ra sao.

Sau đây chúng ta sẽ trình bày một mô hình biểu diễn các vấn đề quyết định. Trong mô hình này, khi chúng ta thực hiện một hành động A, chúng ta biết được các xác suất $PR(S_i|A)$ (xác suất trạng thái S_i được sinh ra khi thực hiện hành động A). Chúng ta cuĩng biết lợi ích của các trạng thái kết cục.

CÂY QUYẾT ĐỊNH

Sau đây chúng ta sẽ trình bày phương pháp biểu diễn vấn đề quyết định bởi cây quyết định. Chúng ta sẽ đưa ra thuật toán tìm ra kế hoạch (tức là một dãy hành động) làm cực đại lợi ích mong đợi.

Cây quyết định là cây có hai loại đỉnh: đỉnh quyết định (decision node) vả đỉnh ngẫu nhiên (chance node). Đỉnh quyết định biểu diễn sự lựa chọn một trong nhiều hành động. Đỉnh ngẫu nhiên biểu diễn sự không chắc chắn của hậu quả của một hành động. Các lá của cây là các đỉnh ứng với một kết cục nào đó.

Trong cây quyết định, chúng ta sẽ biểu diễn các đỉnh quyết định bởi các hình vuông, còn các đỉnh ngẫu nhiên được biểu diễn bởi các hình tròn. Mỗi nhánh đi ra từ các đỉnh quyết định được gắn nhãn bởi một hành động. Mỗi nhánh đi ra từ các đỉnh ngẫu nhiên được gắn nhãn bởi một mệnh đề (mô tả trạng thái) và xác suất để mệnh đề đó đúng (xác suất để trạng thái tương ứng sinh ra). Các kết cục ở các lá được hoàn toàn xác định bởi các hành động và các mệnh đề nằm trên đường đi từ gốc tới lá đó. Các lá được gắn nhãn bởi lợi ích của kết cục tương ứng.

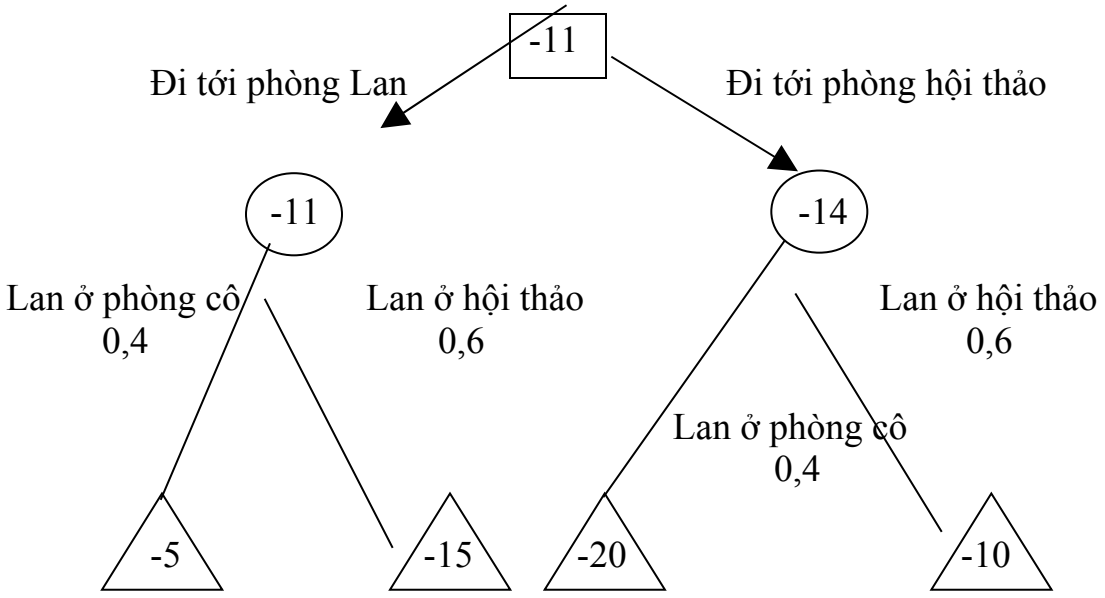
Cây quyết định chứa đựng tất cả các thong tin cần thiết cho phép ta tính được kế hoạch với lợi ích mong đợi lớn nhất.

Ví dụ. Một robot đưa thư làm việc ở khoa công nghệ thông tin chiếm một toà nhà 5 tầng. Nó đang ở trong phòng An trên tầng 5 và được lệnh đi tìm Lan để trao cho cô một bức thư. Robot biết rằng, hoặc Lan ở phòng cô trên tầng 5, hoặc Lan ở trong phòng hội thảo trên tầng 4. Robot có hai kế hoạch để tìm Lan.

1. Đi tới phòng Lan và nếu Lan không ở đó thì đi tới phòng hội thảo.
2. Đi tới phòng hội thảo và nếu Lan không ở đó thì đi tới phòng Lan.

Từ phòng An tới phòng Lan, robot phải đi theo hành lang mất 5 phút. Từ phòng An tới phòng hội thảo robot phải qua cầu thang và mất 10 phút.

Từ phòng Lan tới phòng hội thảo và ngược lại, robot cũng phải qua cầu thang và mất 10 phút. Giả sử robot biết khả năng Lan ở trong phòng cô là 0,4. Mỗi kế hoạch trên có hai kết cục. Chẳng hạn, trong kế hoạch 1, nếu Lan ở phòng cô thì robot mất 5 phút để tìm thấy Lan, còn nếu Lan ở phòng hội thảo thì robot phải mất 15 phút. Mục tiêu của robot là tìm thấy Lan nhanh nhất, do đó chúng ta phải xem lợi ích của kết cục là số đối của thời gian mà robot bỏ ra để tìm thấy Lan. Vì vậy, một kết cục lợi ích là -5. một kết cục lợi ích là -15. Tương tự, trong kế hoạch 2, một kết cục có lợi ích là -20, một kết cục có lợi ích là -10. Cây quyết định cho vấn đề của robot được thể hiện trong hình 10.10. Trong đó, các đỉnh được gắn nhãn bởi lợi ích mong đợi của nó.



***Hình 10.10*. Cây quyết định cho vấn đề robot đưa thư.**

Khi chúng ta đã biểu diễn một vấn đề dưới dạng cây quyết định, chúng ta có thể tính được lợi ích của các đỉnh trong cây. Sau đây là thuật toán tính lợi ích mong đợi của cây quyết định. Tư tưởng của thuật toán là đi từ lá lên gốc, và một đỉnh sẽ được gán giá trị khi mà tất cả các đỉnh con của nó đã được gán giá trị.

Thủ tục tính giá trị pong đợi trong cây quyết định gồm các bước sau:

1. Nếu u là lá của cây quyết định thì nó được gán giá trị là lợi ích của kết cục tương ứng với lá đó.

2. Nếu u là đỉnh trong của cây và u là đỉnh ngẫu nhiên thì u được gán giá trị bằng tổng các giá trị đã gán cho các đỉnh con v của u với hệ số là xác suất trên nhánh từ u tới v.
3. Nếu u là đỉnh trong và u là đỉnh quyết định thì giá trị gán cho u là số lớn nhất trong các giá trị của các đỉnh con của u.

Trong cây quyết định ở hình 10.10, một đỉnh ngẫu nhiên có lợi ích mong đợi là 0,4*(-5)+0,6*(-15) = -11, đỉnh ngẫu nhiên kia có lợi ích mong đợi là -14. Do đó đỉnh quyết định có lợi ích mong đợi là max(-11,-14) = -11.

Theo nguyên lý lợi ích mong đợi cực đại, tại mỗi đỉnh quyết định, tác nhân cần chọn hành động dẫn tới đỉnh có giá trị lớn nhất trong các đỉnh con của nó. Cây quyết định trong hình 10.10 cho ta thấy rằng, robot cần đi tới phòng Lan trước.

Mô hình tổng quát hơn để biểu diễn vấn đề quyết định là mạng quyết định (decision network hay còn gọi là influence diagram). Mạng quyết định là mở rộng của mạng xác suất bằng cách đưa thêm vào các đỉnh quyết định và các đỉnh lợi ích.

GIÁ TRỊ CỦA THÔNG TIN

Trong vấn đề đưa ra quyết định mà chúng ta đã xét ở trên, chúng ta xem rằng, các thong tin liên quan đã có sẵn khi chúng ta đưa ra quyết định. Trong thực tế, việc biết được các thông tin cho một vấn đề quyết định nhiều khi khó có thể thực hiện được hoặc đòi hỏi nhiều thời gian và phương tiện. Một vấn đề được đặt ra là chúng ta có cần thu thập một thông tin nào đó không? Muốn vậy chúng ta cần đưa ra cách đánh giá giá trị của thông tin, hiệu quả mà thông tin này mang lại cho vấn đề quyết định.

Giả sử tri thức hiện thời mà tác nhân biết là E. Như chúng ta đã biết giá trị của hành động tốt nhất là α (trong số các hành động A mà tác nhân có thể thực hiện trong hoàn cảnh E) là

$$\mathbf{EU}(\alpha|E) = \max_A \sum_i \Pr(S_i|A, E)\, \mathbf{U}(S_i)$$

Giả sử thông tin mà chúng ta muốn biết thêm là biến ngẫu nhiên D có thể nhận các giá trị $d_1, \ldots, d_k$. Giá trị của hành động tốt nhất β_j khi ta biết thêm $D = d_j$ là

$$\mathbf{EU}(\beta_j| E, D = d_j)$$

Vì D là biến ngẫu nhiên nên trước khi kiểm tra chúng ta chúng ta không biết được giá trị của nó. Giả sử ta biết được xác suất D nhận giá trị d_j là

$Pr(D= d_j)$. Do đó chúng ta xác định gía trị của hành động tốt nhất trong hoàn cảnh khi E biết thêm thông về D là

$$\mathbf{EU}(\alpha'|E, D) = \sum_{i=1}^{k} Pr(D = d_j)\ \mathbf{EU}(\beta_j|\ E, D = d_j)$$

Giá trị của thông tin D (trong hoàn cảnh E) được xác định là

$$\mathbf{VI}\ (D) = \mathbf{EU}(\alpha'|E, D) - \mathbf{EU}(\alpha|E)$$

Như vậy, giá trị của thông tin được xác định là hiệu số của các lợi ích mong đợi của hành động tốt nhất trước và sau khi nhận được thông tin.

Ví dụ. Trở lại với ví dụ robot đưa thư, giả sử rằng mạng máy tính được nối với máy tính trong phòng làm việc của Lan và robot có thể vào mạng để xác định Lan có làm việc với máy tính không. Nếu Lan làm việc với máy tính thì có nghĩa là Lan đang ở trong phòng cô. Nếu không thì cô đang ở trong phòng hội thảo. như vậy nếu vào mạng thì robot được biết Lan ở trong phòng cô hay ở trong phòng hội thảo. Nếu biết Lan ở trong phòng cô thì robot mất 5 phút, còn nếu Lan ở trong phòng hội thảo thì robot mất 10 phút để tìm thấy Lan. Giả sử robot biết rằng, xác suất Lan làm việc với máy tính là 0,3.

Chúng ta tính giá trị của thông tin "Lan làm việc với máy tính". Chúng ta đã biết giá trị của hành động tốt nhất khi không biết thông tin này là -11 (xem cây quyết định trong hình 10.10). Bây giờ ta tính giá trị của hành động tốt nhất khi biết thông tin "Lan làm việc với máy tính". Giá trị đó là 0,3* (-5)+0,7* (-10)= -8,5. Như vậy, theo định nghĩa, giá trị của thông tin "Lan làm việc với máy tính" sẽ là - 8,5- (-11) = 2,5. do đó nếu thời gian robot vào mạng để biết Lan có làm việc với máy tính hay khônglà 1 phút thì robot cần sử dụng mạng để kiểm tra điều đó.

Người ta chứng minh được rằng, giá trị của thông tin là một số không âm, nó bằng không khi hành động tối ưu là như nhau dù biết hay không biết thông tin đó.

CHƯƠNG 11
LOGIC MỜ VÀ LẬP LUẬN XẤP XỈ

Con người truyền tin cho nhau bằng phương tiện truyền tin riêng của mình: ngôn ngữ tự nhiên. Song bản chất của ngôn ngữ tự nhiên (tiếng Việt, tiếng Anh, tiếng Pháp,…) là không chính xác nhập nhằng. Nhưng ngôn ngữ tự nhiên vẫn là dạng truyền thông mạnh mẽ nhất, thông dụng nhất giữa con người với nhau, và mặc dầu mang yếu tố nhập nhằng, không rõ rang, không chính xác, con người hầu như ít khi hiểu khi những điều mà người khác muốn nói với mình. Tuy nhiên, một vấn đề đặt ra là làm thế nào để máy tính hiểu được các mệnh đề, chẳng hạn, "An là người cao", câu này có nghĩa là An cao 1,65m hay 1,72m?

Mặt khác, thông tin được sử dụng trong các hệ đến từ hai nguồn quan trọng. Một nguồn là các thiết bị cảm nhận môi trường (các thiết bị đo đạc, các phương tiện kỹ thuật hiện đại). Một nguồn là các chuyên gia trong các lĩnh vực. Các chuyên gia mô tả tri thức của họ bằng ngôn ngữ tự nhiên. Để máy tính có thể hiểu được tri thức của con người phát biểu bằng ngôn ngữ tự nhiên, chúng ta cần có một lý thuyết toán học cho phép mô tả chính xác ý nghĩa các khái niệm, chẳng hạn cao, thấp, già, trẻ, … của ngôn ngữ tự nhiên. Lý thuyết toán học đó là lý thuyết tập mờ.

Lý thuyết tập mờ cho chúng ta một công cụ toán học chính xác để mô tả các thông tin không chính xác, mang tính nhập nhằng, mờ (vagueness, ambiguity).

Từ bài báo khởi đầu về lý thuyết tập mờ của Lofti A. Zadeh "Fuzzy sets", công bố năm 1965, lý thuyết tập mờ và logic mờ đã phát triển mạnh mẽ ở Mỹ, Tây Âu và Nhật Bản. Từ giữa 1970 tới nay, với sự nhạy bén với các kỹ thuật mới, các nhà nghiên cứu Nhật Bản là những người đi tiên phong trong việc ứng dụng các kỹ thuật mờ. Họ đã cấp hàng nghìn bằng sáng chế về các ứng dụng của tập mờ và lơgic mờ. Họ đã đưa ra nhiều sản phẩm công nghiệp được bán khắp thế giới.Ví dụ, máy giặt mờ đã sử dụng các bộ cảm nhận tinh xảo để dò ra khối lượng, màu sắc và độ bẩn của quần áo và sử dụng bộ vi sử lý mờ để tự động điều khiển quá trình giặt.

Trong chương này chúng ta sẽ trình bày một số khái niệm cơ bản của lý thuyết tập mờ và logic mờ. Sau đó chúng ta sẽ trình bày các nguyên lý cơ bản của hệ mờ. Các hệ mờ đã được áp dụng trong rất nhiều lĩnh vực: điều khiển, xử lý tín hiệu,truyền thông, các hệ chuyên gia trong y học,…

11.1. TẬP MỜ

Trong mục này chúng ta trình bày khái niệm tập mờ và một số khái niệm cơ bản liên quan tới tập mờ.

11.1.1. Khái niệm tập mờ

Giả sử chúng ta có một vũ trụ bao gồm tất cả các đối tượng mà chúng ta quan tâm. Nhớ lại rằng, một tập cổ điển (tập rõ (crisp set)) A trong một vũ trụ nào đó có thể được xác định bằng cách liệt kê ra tất cả các phần tử của nó, chẳng hạn A = {3,5,6,9}. Trong trường hợp không liệt kê ra hết được các phần tử của tập A, chúng ta có thể chỉ ra những tính chất **chính xác** mà các phần tử của tập A phải thoả mãn, chẳng hạn A= {x|x là số nguyên tố}. Chúng ta nhấn mạnh rằng, một tập rõ được hoàn toàn xác định bởi hàm đặc trưng, hay còn gọi là hàm thuộc (membership function) của nó. Hàm thuộc tập rõ A, được ký hiệu là$\lambda_A(x)$, đó là hàm hai trị (1/0), nó nhận giá trị 1 trên các đối tượng x thuộc tập A và giấ trị 0 trên các đối tượng x không thuộc A. Các tập rõ có một ranh giới rõ ràng giữa các giữa các phần tử thuộc và các phần tử không thuộc nó.

Bây giờ chúng ta quan tâm đến những người trẻ tuổi. Ai là những người được xem là trẻ? Chúng ta có thể xem những người dưới 30 tuổi là trẻ, những người trên 60 tuổi là không trẻ. thế còn những người 35, 40, 45, 50, … thì sao? Trước cách mạng tháng 8, 50 tuổi đã được xem là già, bây giờ 50 tuổi không phải là già, nhưng cũng không thể xem là trẻ. Tính chất "người trẻ" không phải là một tính chất chính xác để xác định một tập rõ, cũng như tính chất "số gần 7", hoặc "tốc độ nhanh",… Đối với tập rõ được xác định bởi các tính chất chính xác cho phép ta biết một đối tượng là thuộc hay không thuộc tập đã cho, các tập mờ được xác định bởi các tính chất không chính xác, không rõ rang (mờ), chẳng hạn các tính chất "người trẻ', "người già", "người đẹp', "áp suất cao", "số gần 7', "tốc độ nhanh",… Các tập mờ được xác định bởi hàm thuộc mà các giá trị của nó là các số thực từ 0 đến 1. Chẳng hạn, tập mờ những người thoả mãn tính chất "người trẻ" (chúng ta sẽ gọi là tập mờ "người trẻ") được xác định bởi hàm thuộc nhận giá trị 1 trên tất cả những người dưới 30 tuổi, nhận giá trị 0 trên tất cả những người trên 60 tuổi và nhận giá trị giảm dần từ 1 tới không trên các tuổi từ 30 đến 60.

Một tập mờ A trong vũ trụ U được xác định là là một hàm μ_A : U $\rightarrow$ [0,1].

Hàm μ_A được gọi là ***hàm thuộc*** (hoặc ***hàm đặc trưng***) của tập mờ A còn $\mu_A(x)$ được gọi là ***mức độ thuộc*** của x vào tập mờ A.

Như vậy tập mờ là sự tổng quát hoá tập rõ bằng cách cho phép hàm thuộc lấy giá trị bất kỳ trong khoảng [0,1], trong khi hàm thuộc của tập rõ chỉ lấy hai giá trị 0 hoặc 1.

Người ta biểu diễn tập mờ A trong vũ trụ U bởi tập tất cả các cặp phần tử và mức độ thuộc của nó:

$$A = \{(x, \mu_A(x))| \ x \in U\}$$

Ví dụ 1. Giả sử các điểm thi được cho từ 0 đến 10, U = {0,1,2,…,10}. Chúng ta xác định ba tập mờ: A = "điểm khá", B = "điểm trung bình" và C="điểm kém" bằng cách cho mức độ thuộc của các điểm vào mỗi tập mờ trong bảng sau:

Điểm	A	B	C
0	0	0	1
1	0	0	1
2	0	0	1
3	0	0,2	0,9
4	0	0,8	0,7
5	0,1	1	0,5
6	0,5	0,8	0,1
7	0.8	0,3	0
8	1	0	0
9	1	0	0
10	1	0	0

Sau đây là các ký hiệu truyền thống biểu diễn tập mờ. Nếu vũ trụ U là rời rạc và hữu hạn thì tập mờ A trong vũ trụ U được biểu diễn như sau:

$$A = \sum_{x\in U} \frac{\mu_A(x)}{x}$$

Ví dụ 2. Giả sử U = {a, b, c, d, e}, ta có thể xác định một tập mờ A như sau:

$$A = \frac{0,7}{a} + \frac{0}{b} + \frac{0,3}{c} + \frac{1}{d} + \frac{0,5}{e}$$

Ví dụ 3. Giả sử tuổi của người từ 0 đến 100. Tập mờ A = "Tuổi trẻ" có thể được xác định như sau:

$$\sum_{y=0}^{25} \frac{1}{y} + \sum_{y=25}^{100} \frac{\left(1+\left(\frac{y-25}{5}\right)^2\right)^{-1}}{y}$$

Đó là một cách biểu diễn của tập mờ có hàm thuộc là:

$$\mu_A(y) = \begin{cases} 1 & 0 \le y \le 25 \\ \left(1+\left(\frac{y-25}{5}\right)^2\right)^{-1} & 25 < y \le 100 \end{cases}$$

Khi vũ trụ U là liên tục, người ta sử dụng cách viết sau để biểu diễn tập mờ A.

$$A = \int_U \mu_A(x) / x$$

trong đó, dấu tích phân (cũng như dấu tổng ở trên) không có nghĩa là tích phân mà để chỉ tập hợp tất cả các phần tử x được gắn với mức độ thuộc của nó.

Ví dụ 4. Tập mờ A = "số gần 2" có thể được xác định bởi hàm thuộc như sau:

$$\mu_A(x) = e^{-(x-2)^2}$$

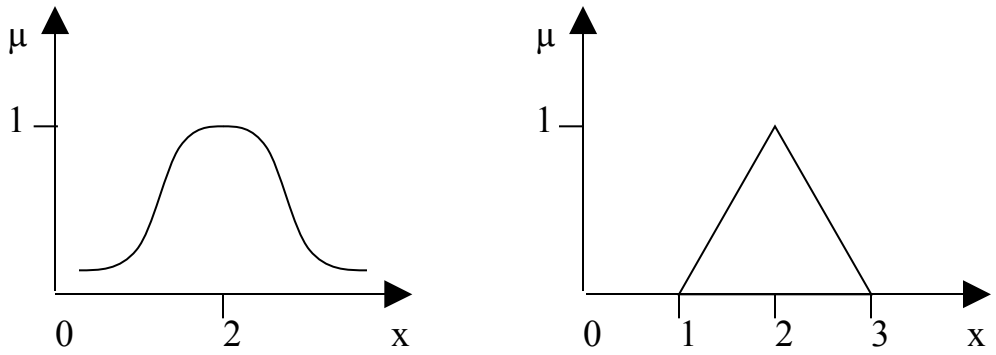
Chúng ta viết

$$A = \int_U e^{-(x-2)^2} / x$$

Cần chú ý rằng, hàm thuộc đặc trưng cho tập mờ "số gần 2" có thể được xác định một cách khác. chẳng hạn:

$$\mu_A = \begin{cases} 0 & x < 1 \\ x - 1 & 1 \le x < 2 \\ 1 & x = 2 \\ -x + 3 & 2 < x \le 3 \\ 0 & x > 3 \end{cases}$$

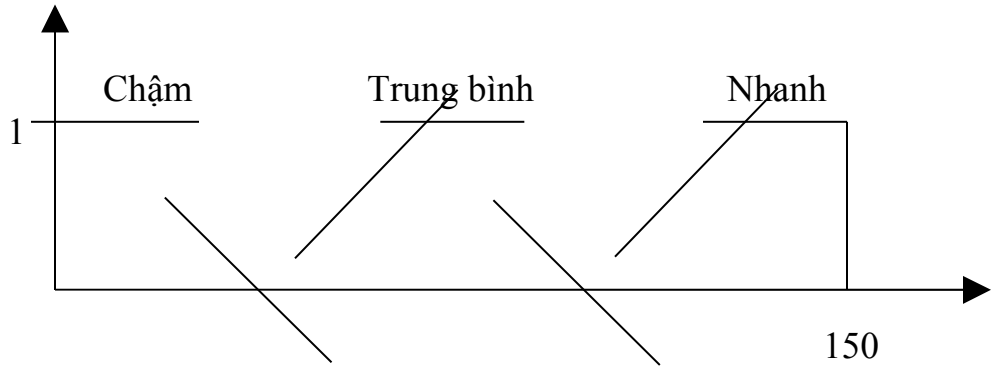
Đồ thị của các hàm thuộc trên được cho trong hình 11.1



***Hình 11.1*. Các hàm thuộc khác nhau cho tập mờ "số gần 2".**

Các tập mờ được sử dụng rât rộng rãi nhất trong các ứng dụng là các tập mờ trên đường thẳng thực $\boldsymbol{R}$ và các tập mờ trong không gian Ơclit n chiều $\boldsymbol{R}^n$ $(n \ge 2)$.

Ví dụ 5. Giả sử tốc độ của một chuyển động có thể lấy giá trị từ 0 tới $v_{max} = 150$ (km/h). Chúng ta có thể xác định ba tập mờ "tốc độ chậm", "tốc độ trung bình" và "tốc độ nhanh" như trong hình 11.2. Các tập mờ này được gọi là các tập mờ hình thang, vì hàm thuộc của chúng có dạng hình thang.



***Hình 11.2*. Các tập mờ "tốc độ chậm", "tốc độ trung bình"**

và"tốc độ nhanh".

Từ những điều đã trình bày và các ví dụ đã đưa ra, chúng ta có một số nhận xét quan trong sau đây.

- Các tập mờ được đưa ra để biểu diễn các tính chất không chính xác, không rõ rang, mờ, chẳng hạn các tính chất "người già", "số gần 2", "nhiệt độ thấp", "áp suất cao", "tốc độ nhanh",…
- Khái niệm tập mờ là một khái niệm toán học hoàn toàn chính xác: một tập mờ trong vũ trụ U là một hàm xác định trên U và nhận giá trị trong đoạn [0,1]. Các tập rõ là tập mờ, hàm thuộc của tập rõ chỉ nhận giá trị 1 hoặc 0. Khái niệm tập mờ là sự tổng quát hoá khái niệm tập rõ
- Một tính chất mờ, chẳng hạn tính chất "số gần 2" có thể mô tả bởi các tập mờ khác nhau như trong ví dụ 4. Trong các ứng dụng chúng ta cần xác định các tập mờ biểu diễn các tính chất mờ sao cho phù hợp với thực tế, với các số liệu thực nghiệm. Trong mục 11.1.4 chúng ta sẽ giới thiệu các phương pháp xác định tập mờ.

11.1.2. Một số khái niệm cơ bản liên quan đến tập mờ

Sau đây chúng ta sẽ xét một số khái niệm cơ bản liên quan tới tập mờ.

Giả sử A là tập mờ trên vũ trụ U. ***Giá đỡ*** của tập mờ A, ký hiệu là supp(A), là một tập rõ bao gồm tất cả các phần tử $x \in U$ có mức độ thuộc vào tập mờ A lớn hơn 0, tức là

$$Supp(A) = \{ x \in A \mid \mu_A(x) > 0\}$$

Nhân của tập mờ A là một tập rõ bao gồm tất cả các phần tử $x \in U$ sao cho $\mu_A(x) = 1$. Còn ***biên*** của tập mờ A sẽ gồm tất cả các $x \in U$ sao cho $0< \mu A(x) < 1$. Hình 11.3 minh hoạ giá đỡ , nhân và biên của một tập mờ.

Độ cao của một tập mờ A, ký hiệu là height(A), được xác định là cận trên đúng của các $\mu_A(x)$ với x chạy trên vũ trụ U, tức là

$$Height(A) = \sup_{x \in U} \mu_A(x)$$

Các tập mờ có độ cao bằng 1 được gọi là các ***tập mờ chuẩn tắc*** (normal fuzzy set). Chẳng hạn, các tập mờ A, B, C trong ví dụ 1 đều là các tập mờ chuẩn tắc.

Lát cắt α (α- cut) của tập mờ A, ký hiệu là A_α, là một tập rõ bao gồm tất cả các phần tử của vũ trụ U có mức độ thuộc vào A lớn hơn hoặc bằng α. Tức là

$$A_\alpha = \{x \in U \mid \mu_A(x) \geq \alpha\}$$

Ví dụ 7. Giả sử U = {a, b, c, d, e, m, n} và A là tập mờ được xác định như sau:

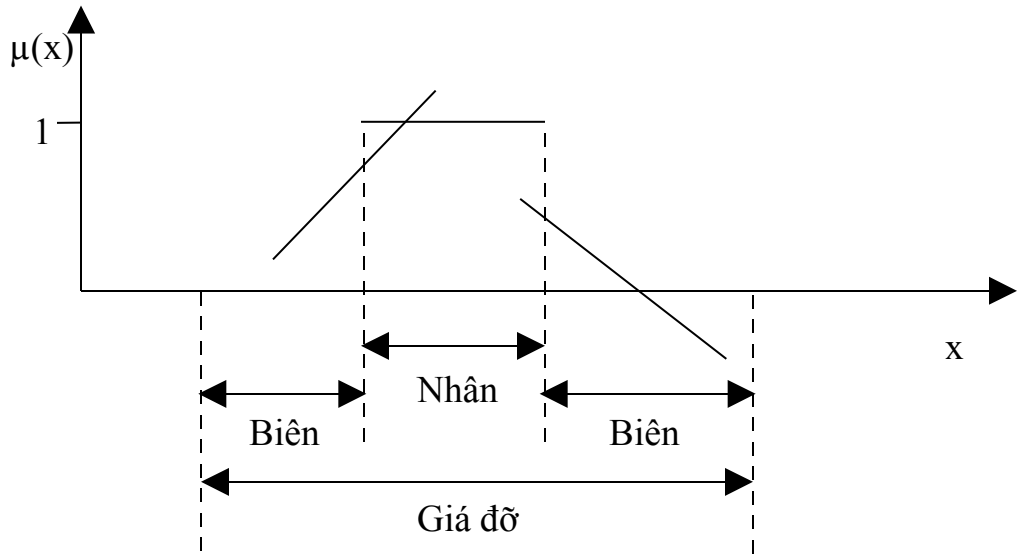
$$A = \frac{0,1}{a} + \frac{0,7}{b} + \frac{0,5}{c} + \frac{0}{d} + \frac{1}{e} + \frac{0,8}{m} + \frac{0}{n}$$

Khi đó ta có:

$$A_{0,1} = \{a, b, c, e, m\}$$

$$A_{0,3} = \{b, c, e, m\}$$

$$A_{0,8} = \{e, m\}$$



***Hình 11.3*. Giá đỡ, nhân và biên của tập mờ**

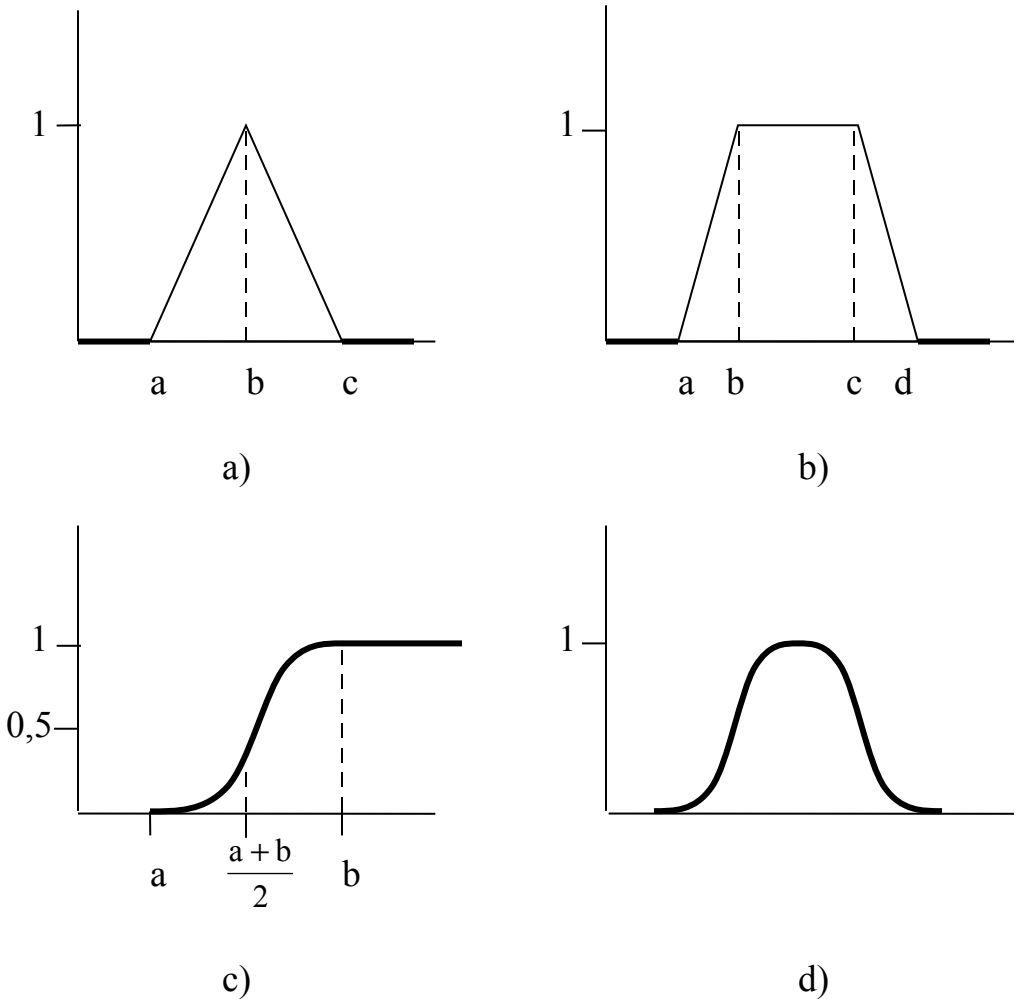
Một khái niệm quan trọng nữa là khái niệm ***tập mờ lồi***. Khi tập vũ trụ U là không gian Ơclit n chiều, $U = R^n$, khái niệm tập mờ lồi có thể tổng quát hoá cho các tập mờ. Một tập mờ A trong không gian R^n được gọi là tập mờ lồi nếu mọi lát cắt A_α đều lồi , với $\alpha \in (0,1)$. Sử dụng khái niệm lát cắt và tập lồi trong không gian R^n, chúng ta sẽ dễ dàng chứng minh được khẳng định sau đây:

$$\mu_A[\lambda x - (1 - \lambda)y] \geq [\mu_A(x), \mu_A(y)]$$

Với mọi x, y $\in R^n$ và với mọi $\lambda \in [0;1]$. Người ta thường sử dụng tính chất này làm định nghĩa tập mờ lồi. Các tập mờ lồi đóng vai trò quan trọng trong các ứng dụng là các tập mờ lồi trên đường thẳng thực.

Chúng ta sẽ biểu diễn các định lượng không chính xác, chẳng hạn "số gần 5", bởi các số mờ.

Một tập mờ lồi chuẩn tắc trên đường thẳng thực mà mỗi lát cắt α là một khoảng đóng, được gọi là ***số mờ*** (fuzzy number). (Lưu ý rằng, diều kiện các lát cắt là các khoảng đóng tương đương với điều kiện hàm thuộc liên tục từng khúc). Việc nghiên cứu các phép toán số học +, –, *, / và các phép so sánh trên các số mờ là nội dung lĩnh vực ***số học mờ***. Số học mờ là một nhánh nghiên cứu của lý thuyết mờ. Các số mờ đóng vai trò quan trọng trong các ứng dụng, đặc biệt trong các hệ mờ.



***Hình 11.4*. Các dạng số mờ đặc biệt.**

a) số mờ hình tam giác; b) số mờ hình thang;

c) số mờ hình chữ S; d) số mờ hình chuông.

Các số mờ đặc biệt, được sử dụng nhiều trong các ứng dụng là các số mờ hình tam giác, các số mờ hình thang, các số mờ hình chữ S và các số mờ hình chuông. Các dạng số mờ này được minh hoạ trong hình 11.4. Chúng ta có thể đưa ra biểu thức giải tích của các hàm thuộc của các số mờ này.

Chẳng hạn, số mờ hình tam giác A (hình 11.4a) có hàm thuộc được xác định như sau:

$$\mu_A(x) = \begin{cases} (x - a)/(b - a) & a \le x < b \\ (c - b)/(c - b) & b \le x \le c \\ 0 & x > c \text{ or } x < a \end{cases}$$

Hàm thuộc của số mờ S hình (11.4c) được xác định như sau:

$$\mu_S(x) = \begin{cases} 0 & x < a \\ 2\left(\dfrac{x - a}{b - a}\right)^2 & a \le x < \dfrac{a + b}{2} \\ 1 - 2\left(\dfrac{x - b}{b - a}\right)^2 & \dfrac{a + b}{2} \le x < b \\ 1 & x \ge b \end{cases}$$

11.1.3. Tính mờ và tính ngẫu nhiên

Chúng ta đã đưa ra khái niệm tập mờ để biểu diễn các tính chất mờ. Khi biểu diễn mọt tính chất mờ bởi một tập mờ A, và x là một đối tượng bất kỳ thì mức độ thuộc của x vào tập mờ A là một số $\mu_A(x) \in [0,1]$. Mặt khác, chúng ta biết rằng, xác suất của một sự kiện ngẫu nhiên e, Pr(e) cũng là một số nằm giữa 0 và 1. Một câu hỏi được đặt ra: Có gì khác nhau giữa tính mờ (fuzziness) và tính ngẫu nhiên (randomness)? Câu trả lời là: Có. Tính mờ mô tả sự không rõ rang của một sự kiện, còn tính ngẫu nhiên mô tả sự không chắc chắn xuất hiện của một sự kiện. Một sự kiện ngẫu nhiên có thể xuất hiện hoặc không, xác suất biểu diễn mức độ thường xuyên xuất hiện của sự kiện ngẫu nhiên. Để thấy được sự khác nhau giữa tập mờ và xác suất, chúng ta xét ví dụ sau đây (ví dụ này là của Bezdek, 1993).

Ví dụ 8. Bạn đang trên sa mạc và khát nước. Người ta mang đến cho bạn hai chai nước gắn nhãn A và B. Trên nhãn A ghi rằng nước trong chai

này có mức độ thuộc vào tập mờ "nước uống được" là 0,95. Trên nhãn B ghi rằng, xác suất để chai nước này uống được là 0,95. Bạn sẽ chọn chai nào trong hai chai?

Mức độ thuộc 0,95 có nghĩa là nước trong chai A khá gần với nước tinh khiết, chẳng hạn nước máy của ta, nó có thể lẫn tạp chất nhưng không ảnh hưởng lớn tới sức khoẻ. Mặt khác, xác suất để chai nước gắn nhãn B uống được là 0,95 có nghĩa là trong 100 lần lấy chai nước nhãn B có khoảng 95 lần nước trong chai là uống được (nước tinh khiết), song có 5 lần nước trong chai không phải là nước uống được, có thể là nước độc hại cho sức khoẻ, chẳng hạn axit. Rõ ràng là bạn nên chọn chai nước nhãn A.

Bây giờ giả sử bạn được quyền mở chai và lấy mẫu xét nghiệm. sau khi xét nghiệm bạn sẽ biết chính xác nước trong chai B là nước uống được hay không. Xác suất 0,95 sẽ là xác suất có điều kiện và bằng 1 trong trường hợp nước trong chai là uống được và 0 nếu nước trong chai là nước không uống được. Nhưng sau khi xét nghiệm thì mức độ thuộc 0,95 của chai nước nhãn A vẫn là 0,95.

11.1.4. Xác định các hàm thuộc

Một tập mờ trong vũ trụ U được định nghĩa là một hàm thuộc (membership function) xác định trên vũ trụ U và nhận giá trị trong đoạn [0,1]. Như vậy xác định một tập mờ có nghĩa là xác định hàm thuộc của nó. Một câu hỏi đặt ra là: làm thế nào để xác định được các hàm thuộc? các nhà nghiên cứu đã đưa ra rất nhiều phương pháp để tính mức độ thuộc của đối tượng x vào một tập mờ A, tức là tính $\mu_A(x)$. Giá trị $\mu_A(x)$ có thể xác định được bằng trực quan, hoặc có thể được tính bằng một thuật toán nào đó, hoặc nhận được thông qua lập luận logic. Sau đây là một số phương pháp xác định hàm thuộc.

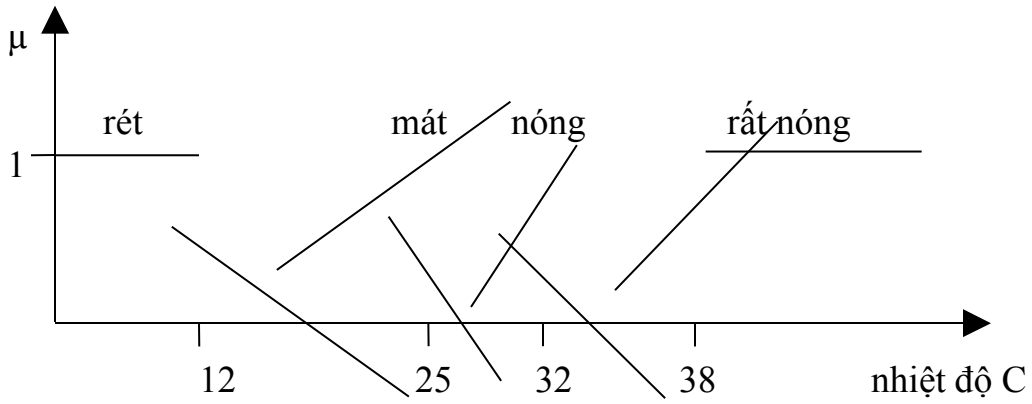
- Trực quan
- Suy diễn dựa trên tri thức về lĩnh vực
- Sử dụng mạng nơron
- Sử dụng thuật toán di truyền

Cần lưu ý rằng, danh sách các phương pháp xác định hàm thuộc còn dài. Lĩnh vực này hiện nay vẫn đang tiếp tục được nghiên cứu. Trong mục này chúng ta chỉ giới thiệu giản lược ý tưởng của một vài phương pháp.

PHƯƠNG PHÁP TRỰC QUAN

Phương pháp này được áp dụng để xác định các tập mờ trên đường thẳng thực, chẳng hạn các tập mờ "nhiệt độ cao", "áp suất thấp", "tốc đọ nhanh", … Trong các trường hợp này, người ta dựa vào sự hiểu biết trực quan, dựa vào ngữ nghĩa của các từ để đưa ra các hàm thuộc. Chẳng hạn, chúng ta muốn xác định các tập mờ "trời rét", "trời mát", "trời nóng", "trời rất nóng". Giả sử nhiệt độ không khí được đo bởi độ C. Với thời tiết ở Hà Nội, chúng ta có thể xác định các hàm thuộc của các tập mờ trên như trong hình vẽ 11.5.

Cần lưu ý rằng, ở đây chúng ta nói tới thời tiết ở Hà Nội. Với người ở xứ lạnh, chẳng hạn ở Henxinki, nhiệt độ 12^0C chưa phải là rét. Do đó nếu nói tới thời tiết ở Henxinki thì hàm thuộc của các tập mờ "trời rét", "trời mát",… cần phải xác định khác đi.



***Hình 11.5*. Các hàm thuộc của một số tập mờ.**

PHƯƠNG PHÁP SUY DIỄN DỰA TRÊN TRI THỨC VỀ LĨNH VỰC

Trong phương pháp này, người ta dựa trên các tri thức về lĩnh vực đang quan tâm để suy diễn ra cách xác định các hàm thuộc cần xây dựng. Chúng ta minh hoạ phương pháp này bằng ví dụ sau đây. Chúng ta xét các tam giác trong hình học phẳng. Chúng ta muốn xấc định các tập mờ sau:

I = "tam giác gần cân"

R = "tam giác gần vuông"

E = "tam giác gần đều"

Giả sử A,B,C là các góc trong của tam giác và $A \geq B \geq C$. Vũ trụ U các tam giác được xác định là tập $U = \{(A,B,C) \mid A \geq B \geq C,$

$A + B + C = 180^0\}$. Sở dĩ chúng ta biểu diễn mỗi tam giác bởi bộ ba góc (A,B,C) vì chúng ta chỉ quan tâm đến hình dạng của tam giác: nó cân, đều hay vuông.

Một tam giác (A,B,C) với $A \geq B \geq C$ và $A + B + C = 180^0$ sẽ cân nếu A = B hoặc B = C. Nó càng không cân nếu A càng khác B và B càng khác C. Do đó chúng ta có thể đưa ra công thức sau cho hàm thuộc của tập mờ I = "tam giác gần cân"

$$\mu_I (A,B,C) = 1- 1/60 \min (A – B, B – C)$$

Từ tri thức về tam giác, chúng ta có thể xác định hàm thuộccủa các tập mờ "tam giác gần vuông" và "tam giác gần đều" như sau:

$$\mu_R (A,B,C) = 1 – 1/90 \mid A – 90\mid$$

$$\mu_E (A,B,C) = 1 – 1/180 (A – C)$$

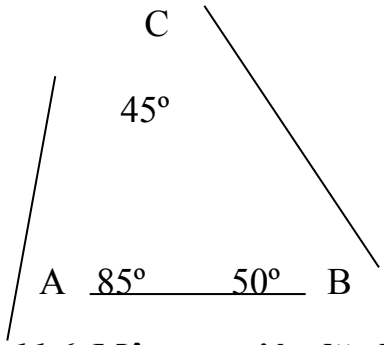
Ví dụ 9. Chúng ta xét tam giác x = (A, B, C) với các góc như sau : $A = 85^0 \geq B = 50^0 \geq C = 45^0$, ở đây $A + B + C = 85^0 + 50^0 + 45^0 = 180^0$. Tính mức độ thuộc của x vào các tập mờ I, R, E theo các công thức trên ta có:

$$\mu_I (x) = 0,916$$

$$\mu_R (x) = 0,94$$

$$\mu_E (x) = 0,7$$

Như vậy tam giác x có mức độ thuộc cao nhất vào tập mờ "tam giác gần vuông", nhưng nó cũng có mức độ thuộc khá cao vào tập mờ "tam giác gần cân". Từ hình 11.6 chúng ta thấy rằng, các hàm thuộc mà chúng ta đã xác định rất phù hợp với thực tế.



***Hình 11.6*. Một tam giác đặc biệt.**

SỬ DỤNG MẠNG NƠRON

Sau đây chúng ta sẽ giới thiệu về mạng nơron và phương pháp sử dụng mạng nơron để xác định hàm thuộc. Để có thể áp dụng được phương

pháp này, bạn cần có hiểu biết đầy đủ hơn về mạng nơron và phương pháp hình học bằng huấn luyện mạng nơron.

Một nơron là một đơn vị xử lý, cấu tạo và sự hoạt động của nó là sự mô phỏng nơron trong não người. Hình 11.7 biểu diễn một nơron. Mỗi nơron có n đầu vào x_1, …, x_i, …, x_n và một đầu ra y. Mỗi đường vào được gắn với một trọng số w_i (i = 1, …, n). Trọng số w_i dương biểu diễn tins hiệu x_i được kích thích, trọng số w_i âm biểu diễn tín hiệu x_i bị kìm chế. Các tín hiệu vào được tích hợp bởi tổng trọng số

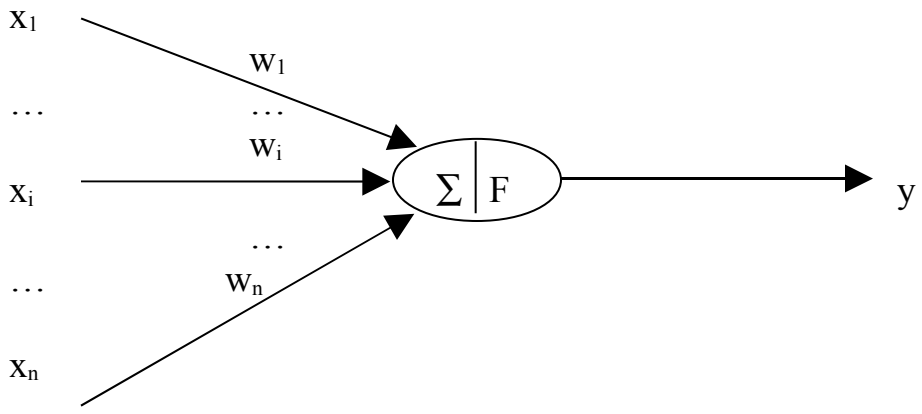
$$\sum_{i=1}^{n} w_i x_i$$

và được biến đổi thành tín hiệu ra y thông qua hàm kích hoạt F như sau:

$$y = F\left(\sum_{i=1}^{n} w_i x_i - \theta \right)$$

trong đó, là θ giá trị ngưỡng và F(s) là một hàm không tuyến tính; chẳng hạn hàm sigmoid:

$$F(s) = \frac{1}{1+e^{-s}}$$

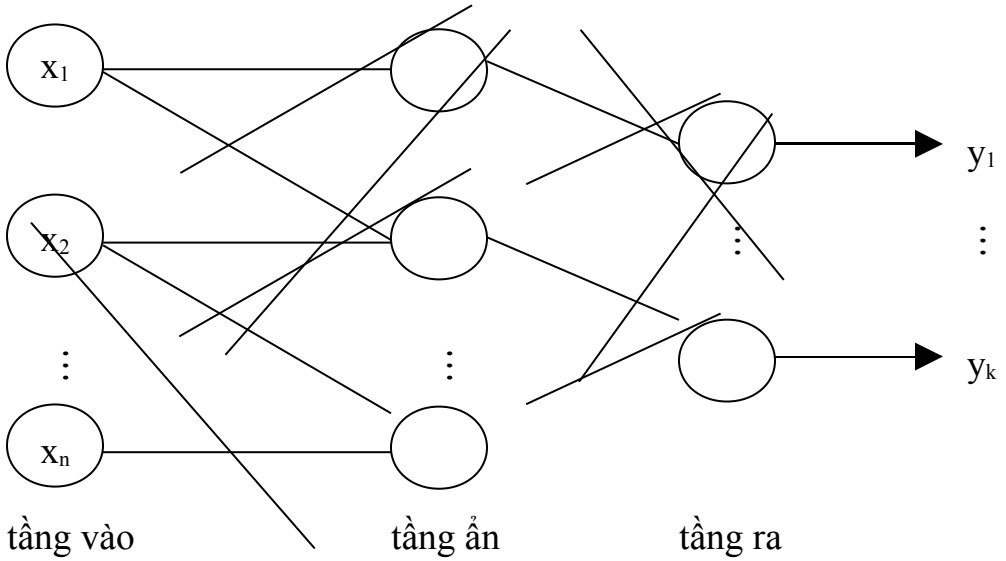


***Hình 11.7*. Một nơron.**

Các nơron được kết nối với nhau để tạo thành các mạng nơron (neural network). Hình 11.8 biểu diễn một mạng nơron truyền thẳng (feedforward network). Trong mạng truyền thẳng, đầu ra của các nơron ở một tầng (trừ tầng ra) được nối với các đầu vào của các nơron ở tầng tiếp theo. Các đơn vị ở tầng vào không phải là các nơron mà chỉ là các vùng đệm để lưu các giá trị đưa vào mạng. Ngoài tầng vào và tầng ra, trong mạng truyền thẳng có thể có

một hoặc nhiều tầng ẩn. Mạng nơron truyền thẳng là một hệ tính. Nếu mạng có n đầu vào $x_1, x_2, \ldots, x_n$ và k đầu ra $y_1, y_2, \ldots, y_k$ tương ứng với các giá trị đầu vào. Nói một cách khác, mạng sẽ xác định các hàm $y_i = f_i(x_1, \ldots, x_n)$ $(i = 1,\ldots, k)$.

Bây giờ chúng ta sẽ giới thiệu phương pháp sử dụng mạng nơron truyền thẳng để xác định các tập mờ từ một tập dữ liệu. Phương pháp này được đề xuất bởi Takagi và Hyashi, 1991. Tập dữ liệu được phân thành tập dữ liệu huấn luyện và tập dữ liệu kiểm tra. Tập dữ liệu huấn luyện được sử dụng để huấn luyện mạng nơron. Giả sử mỗi dữ liệu x là một vectơ không gian n chiều, $x = (x_1, \ldots, x_n)$. Sử dụng các kỹ thuật phân bó (clustering) chúng ta phân tập dữ liệu thành k lớp $R_1, \ldots, R_k$. Chúng ta sẽ xác định k tập mờ $A_1, \ldots, A_k$ bằng cách sử dụng mạng truyền thẳng có n đầu vào và k đầu ra như trong hình 11.8. Đầu ra y_i sẽ được xem là mức độ thuộc của điểm dữ liệu x vào tập mờ A_i, $y_i = \mu_{Ai}(x)$.



***Hình 11.8*. Mạng nơron truyền thẳng.**

Nếu điểm dữ liệu $x = (x_1, \ldots, x_n)$ thuộc lớp R_i thì giá trị đầu ra tương ứng là $y = (y_1, \ldots, y_k)$, trong đó $y_i = 1$ và $y_j = 0$ với mọi $j \neq i$. Khởi tạo ra một mạng nơron truyền thẳng với n đầu vào và k đầu ra. Tập dữ liệu huấn luyện mạng bao gồm các cặp (x, y), trong đó x được lấy từ tập dữ liệu huấn luyện. Chúng ta áp dụng thuật toán truyền ngược (backpropagation algorithm) để huấn luyện mạng với các dữ liệu huấn luyện là các cặp (x, y) đã nêu trên. Mạng nơron thu được sau quá trình huấn luyện sẽ được sử dụng để xác định các tập mờ $A_1, \ldots, A_k$. Chẳng hạn, nếu các dữ liệu là các điểm trong không gian 2 chiều và các tập mờ cần xác định là A_1, A_2, A_3. Trong trường hợp này

mạng nơron có 2 đầu vào x_1, x_2 và 3 đầu ra y_1, y_2, y_3. Khi đưa vào mạng (x, y) = (0,5 , 0,8), giả sử mạng cho ra $y_1 = 0,1$; $y_2 = 0,8$; $y_3 = 0,7$ khi đó $\mu_{A1}(0,5 , 0,8) = 0,1$; $\mu_{A2}(0,5 , 0,8) = 0,8$; $\mu_{A3}(0,5 , 0,8) = 0,7$.

11.2. CÁC PHÉP TOÁN TRÊN TẬP MỜ

Trong mục này chúng ta sẽ xác định các phép toán trên tập mờ. Các phép toán này là sự tổng quát hoá của các phép toán trên tập rõ.

11.2.1. Các phép toán chuẩn trên tập mở

Giả sử A và B là các tập mờ trên vũ trụ U. Ta nói tập mờ A *bằng* tập mờ B, nếu với mọi $x \in U$

$$\mu_A(x) = \mu_B(x)$$

Tập mờ A được gọi là tập con của tập mờ B, $A \subseteq B$, nếu với mọi $x \in U$

$$\mu_A(x) \geq \mu_B(x)$$

1. Phần bù

Phần bù của tập mờ A là tập mờ $\overline{A}$ với hàm thuộc được xác định như sau:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \qquad (1)$$

2. Hợp

Hợp của tập mờ A và tập mờ B là tập mờ $A \cup B$ được xác định như sau:

$$\mu_{A \cup B}(x) = \max (\mu_A(x), \mu_B(x)) \qquad (2)$$

3. Giao

Giao của A và B là tập mờ $A \cap B$ với hàm thuộc

$$\mu_{A \cap B}(x) = \min (\mu_A(x), \mu_B(x)) \qquad (3)$$

Ví dụ 1. Giả sử $U = \{a, b, c, d, e\}$ và A, B là các tập mờ như sau

$$A = \frac{0.3}{a} + \frac{0.7}{b} + \frac{0}{c} + \frac{1}{d} + \frac{0.5}{e}$$

$$B = \frac{0.1}{a} + \frac{0.9}{b} + \frac{0.6}{c} + \frac{1}{d} + \frac{0.5}{e}$$

Khi đó chúng ta có các tập mờ sau

$$\overline{A} = \frac{0.7}{\mathrm{a}} + \frac{0.3}{\mathrm{b}} + \frac{1}{\mathrm{c}} + \frac{0}{\mathrm{d}} + \frac{0.5}{\mathrm{e}}$$

$$A \cup B = \frac{0.3}{\mathrm{a}} + \frac{0.9}{\mathrm{b}} + \frac{0.6}{\mathrm{c}} + \frac{1}{\mathrm{d}} + \frac{0.5}{\mathrm{e}}$$

$$A \cap B = \frac{0.1}{\mathrm{a}} + \frac{0.7}{\mathrm{b}} + \frac{0}{\mathrm{c}} + \frac{1}{\mathrm{d}} + \frac{0.5}{\mathrm{e}}$$

Chúng ta có thể dễ dàng chứng minh được rằng các phép toán lấy phần bù, hợp, giao được xác định bởi (1), (2), (3) thoả mãn tất cả các tính chất của các phép toán trên tập rõ, kể cả luật De Morgan, trừ luật loại trừ trung gian (law of excluded middle), tức là với tập mờ A ta có

$$\mathrm{A} \cup \overline{\mathrm{A}} \neq \mathrm{U} \text{ và } \mathrm{A} \cap \overline{\mathrm{A}} \neq \varnothing$$

4. Tích Đề-các

Giả sử A_1, A_2, …, A_n là các tập mờ trên các vũ trụ U_1, U_2, …, U_n tương ứng. Tích Đề-các của các A_1, A_2, …, A_n là tập mờ $A = A_1 \times A_2 \times \ldots \times A_n$ trên không gian tích $U = U_1 \times U_2 \times \ldots \times U_n$ với hàm thuộc được xác định như sau

$$\mu_A(x_1, x_2, \ldots, x_n) = \min\left(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)\right)$$

$$x_1 \in U_1, x_1 \in U_1, \ldots, x_1 \in U_1 \qquad (4)$$

5. Phép chiếu

Giả sử A là tập mờ trong không gian tích $U_1 \times U_2$.Hình chiếu của trên U_1 là tập mờ A_1 trên U_1 với hàm thuộc

$$\mu_{A_1}(x_1) = \max_{x_2 \in U_2} \mu_A(x_1, x_2) \qquad (5)$$

Định nghĩa này có thể mở rộng cho trường hợp A là tập mờ trên không gian $U_{i_1} \times U_{i_2} \times \ldots \times U_{i_k}$. Ta có thể chiếu A lên không gian tích $U_{i_1} \times U_{i_2} \times \ldots \times U_{i_k}$, trong đó ($i_1$, i_2, …, i_k) là dãy con của dãy (1, 2, …, n), để nhận được tập mờ trên không gian $U_{i_1} \times U_{i_2} \times \ldots \times U_{i_k}$.

6. Mở rộng hình trụ

Giả sử A_1 là tập mờ trên vũ trụ U_1. Mở rộng hình trụ của A_1 trên không gian tích $U_1 \times U_2$ là tập mờ A trên vũ trụ $U_1 \times U_2$ với hàm thuộc được xác định bởi:

$$\mu_A(x_1, x_2) = \mu_{A_1}(x_1) \qquad (6)$$

Đương nhiên là ta có thể mở rộng một tập mờ trong không gian $U_{i_1} \times U_{i_2} \times \ldots \times U_{i_k}$ thành một tập mờ hình trụ trong không gian $U_1 \times U_2 \times \ldots \times U_n$, trong đó ($i_1$, i_2, …, i_k) là một dãy con của dãy (1, 2, …, n).

Ví dụ 2. Giả sử $U_1 = \{a, b, c\}$ và $U_2 = \{d, e\}$. Giả sử A_1, A_2 là các tập mờ trên U_1, U_2 tương ứng:

$$A_1 = \frac{1}{a} + \frac{0}{b} + \frac{0,5}{c}$$

$$A_2 = \frac{0,3}{d} + \frac{0,7}{e}$$

Khi đó ta có

$$A_1 \times A_2 = \frac{0,3}{(a, d)} + \frac{0,7}{(a, e)} + \frac{0}{(b, d)} + \frac{0}{(b, e)} + \frac{0,3}{(c, d)} + \frac{0,5}{(c, e)}$$

Nếu tập chiếu này mờ trên U_1, ta nhận được tập mờ sau:

$$\frac{0,7}{a} + \frac{0}{b} + \frac{0,5}{c}$$

Mở rộng hình trụ của tập mờ A_1 trên không gian $U_1 \times U_2$ là tập mờ sau

$$\frac{1}{(a, d)} + \frac{1}{(a, e)} + \frac{0}{(b, d)} + \frac{0}{(b, e)} + \frac{0,5}{(c, d)} + \frac{0,5}{(c, e)}$$

11.2.2. Các phép toán khác trên tập mờ

Các phép toán chuẩn: phần bù, hợp, giao được xác định bởi các công thức (1), (2) và (3) không phải là sự tổng quát hoá duy nhất của phép toán phần bù, hợp, giao trên tập rõ. Có thể thấy rằng. tập mờ $A \cup B$ được xác định bởi (2) là tập mờ nhỏ nhất chứa cả A và B, còn tập mờ $A \cap B$ được xác định bởi (3) là tập mờ lớn nhất nằm trong cả A và B. Còn có những cách khác để xác định các phép toán phần bù, hợp, giao trên các tập mờ. Chẳng hạn, ta có thể xác định hợp của A và B là tập mờ bất kỳ chứa cả A và B. Sau đây chúng ta sẽ đưa vào các phép toán mà chúng là tổng quát hoá của các phép toán chuẩn được xác định bởi (1), (2) và (3).

1. Phần bù mờ

Giả sử chúng ta xác định hàm $C : [0,1] \rightarrow [0,1]$ bởi công thức $C(a) = 1 - a, \forall a \in [0,1]$. Khi đó từ công thức (1) xác định phần bù chuẩn, ta có:

$$\mu_A(x) = C[(\mu_A(x)] \qquad (7)$$

Điều này gợi ý rằng, nếu chúng ta có một hàm C thoả mãn một số điều kiện nào đó thì chúng ta có thể xấc định phần bù $\overline{A}$ của tập mờ A bởi công thức (7). Tổng quát hoá các tính chất của hàm C . C(a) = 1- a, chúng ta đưa ra định nghĩa sau:

Phần bù của tập mờ A là tập mờ $\overline{A}$ với hàm thuộc được xác định bởi (7), trong đó C là hàm thoả mãn các điều kiện sau:

- Tiên đề C_1(điều kiện biên). C(0) = 1, C(1) = 0
- Tiên đề C_2 (đơn điệu không tăng). Nếu a < b thì $C(a) \geq C(b)$ với mọi

  $a, b, \in [0,1]$

Hàm C thoả mãn các điều kiện C_1, C_2 sẽ được gọi là hàm phần bù . Chẳng hạn, hàm C(a) = 1 – a thoả mãn cả hai điều kiện trên.

Sau đây là một số lớp phần bù mờ quan trọng.

Ví dụ 3. Các phần bù mờ lớp Sugeno được xác định bởi hàm C như sau;

$$C(a) = \frac{1 - a}{(1 + \lambda a)} \qquad \forall a \in [0,1]$$

trong đó λ là tham số, λ > -1, ứng với mỗi giá trị của λ chúng ta nhận được một phần bù. Khi λ = 0 phần bù Sugeno trở thành phần bù chuẩn (1).

Ví dụ 4. Các phần bù mờ lớp Yager được xác định bởi hàm C.

$$C(a) = (1 - a^w)^{\frac{1}{w}}$$

Trong đó w là tham số, w > 0, ứng với mỗi giá trị của tham số w chúng ta sẽ có một phần bù và với w = 1 phần bù Yager trở thànhphafn bù chuẩn (1).

2. Hợp mờ - các phép toán S – norm

Phép toán được xác định bởi (2), tức là nó được xác định nhờ hàm max (a, b): [0,1] → [0,1]. từ các tính chất của hàm max này, chúng ta đưa ra một lớp các hàm được gọi S – norm.

Một hàm S : [0,1] × [0,1] →[0,1] được gọi là S – norm nếu nó thoả mãn các tính chất sau:

- Tiên đề S_1 (điều kiện biên): S (1,1) = 1; S(0,a) = S(a,0) = a.
- Tiên đề S_2 (tính giao hoán): S(a, b) = S(b,a).
- Tiên đề S_3 (tính kết hợp) : S (S, (a, b), c) = S(a, S(b,c)).
- Tiên đề S_4 (đơn điệu tăng): Nếu a ≤a′ , b ≤b′ thì S(a, b)≤S(a′, b′).

Ứng với mỗi S – norm, chúng ta xác định một phép hợp mờ như sau. Hợp của A và B là là tập mờ A ∪ B với hàm thuộc được xác định bởi biểu thức:

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) \qquad (8)$$

Các phép hợp được xác định bởi (8) được gọi là các phép toán S – norm. Chẳng hạn, hàm max (a, b) thoả mãn các điều kiện (S) – (S), do đó hợp chuẩn (2) là phép toán S – norm. Người ta thường ký hiệu max (a, b) = a ∨ b. Sau đây là một số phép toán S – norm quan trọng khác.

Ví dụ 5. Tổng Drastic:

$$a \vee b = \begin{cases} a & \text{if} \quad b = 0 \\ b & \text{if} \quad a = 0 \\ 1 & \text{if} \quad a > 0, b > 0 \end{cases}$$

Tổng chặn: $a \oplus b = \min(1, a + b)$

Tổng đại số: $a \uplus b = a + b - ab$

Ví dụ 6. Các phép hợp Yager

$$S_w(a, b) = \min\left[1, (a^w + b^w)^{\frac{1}{w}}\right]$$

trong đó, w là tham số, w > 0, ứng với mỗi giá trị của w, chúng ta có một S – norm cụ thể. Khi w = 1, hợp Yager trở thành tổng chặn. Có thể thấy rằng:

$$\lim_{w \to \infty} S_w(a, b) = \max(a, b)$$

$$\lim_{w \to 0} S_w(a, b) = a \vee b$$

Như vậy, khi $w \to \infty$ thì hợp Yager trở thành hợp chuẩn.

3. ***Giao mờ – các phép toán T – norm.***

Chúng ta đã xác định giao chuẩn bởi hàm min(a,b): $[0,1] \times [0,1] \to [0,1]$. Tổng quát hoá từ các tính chất của hàm min(a,b), chúng ta xác định một lớp các hàm được gọi là T – norm.

Một hàm T: $[0,1] \times [0,1] \to [0,1]$ được gọi là T – norm nếu nó thoả mãn các tính chất sau:

- Tiên đề T_1 (điều kiện biên): T(0,0) = 0; T(a, 1) = T(1, a) = a
- Tiên đề T_2 (tính giao hoán): T(a, b) = T(b, a)
- Tiên đề T_3 (tính kết hợp): T(T(a, b), c) = T(a, T(b, c))
- Tiên đề T_4 (đơn điệu tăng): Nếu $a \le a'$, $b \le b'$ thì $T(a, b) \le T(a', b')$

Giao mờ của các tập A và B là tập mờ $A \cap B$ với hàm thuộc được xác định bởi:

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) \qquad (9)$$

trong đó, T là một T – norm. Các phép giao mờ được xác định bởi (9) được gọi là các phép toán T – norm. Chẳng hạn, hàm min(a, b) là T – norm. Chúng ta sẽ ký hiệu min(a, b) = a $\wedge$ b.

Chúng ta đưa ra một số T – norm quan trọng.

Ví dụ 7. Tích đại số

a . b = ab

Tích Drastic

$$a \mathbin{\dot{\wedge}} b = \begin{cases} a & \text{if} \quad b = 1 \\ b & \text{if} \quad a = 1 \\ 0 & \text{if} \quad a, b < 1 \end{cases}$$

Tích chặn

$$a \odot b = \max(0, a + b - 1)$$

Ví dụ 8. Các phép giao Yager

$$(T_n\, a, b) = 1 - \min\left[1, ((1-a)^w + (1-b)^w)^{\frac{1}{w}}\right]$$

trong đó w là tham số, w > 0. Khi w = 1, giao Yager trở thành tích chặn. Có thể chỉ ra rằng:

$$\lim_{w \to \infty} T_w(a, b) = \min(a, b)$$

$$\lim_{w \to 0} T_w(a, b) = a \mathbin{\dot{\wedge}} b$$

Như vậy w $\to \infty$, giao Yager trở thành giao chuẩn.

Các mối quan hệ giữa các S – norm và T – norm được phát biểu trong định lý sau:

$$a \mathbin{\dot{\wedge}} b \le T(a, b) \le \min(a, b)$$

$$\max(a, b) \le S(a, b) \le a \mathbin{\dot{\vee}} b$$

Từ định lý trên chúng ta thấy rằng các phép toán min và max là cận trên và cận dưới của các phép toán T – norm và S – norm tương ứng. Như vậy các phép toán hợp và giao không thể nhận giá trị trong khoảng giữa min và max.

Người ta đưa vào các phép toán V(a,b): $[0,1] \times [0,1] \rightarrow [0,1]$, mà các giá trị của nó nằm giữa min và max: $\min(a,b) \le V(a,b) \le \max(a,b)$. Các phép toan này được gọi là các phép toán lấy trung bình (averaging operators). Sau đây là một số phép toán lấy trung bình.

- Trung bình tổng quát

$$V_a(a,b) = \left(\frac{a^\alpha + b^\alpha}{2} \right)^{\frac{1}{\alpha}}$$

trong đó, α là tham số và $\alpha \neq 0$

- Trung bình max – min

$$V_\lambda(a,b) = \lambda \max(a,b) + (1 - \lambda) \min(a,b)$$

trong đó, tham số λ [0,1].

4. ***Tích Đề-các mờ***

Chúng ta đã xác định tích đề củ các tập nờ A_1, A_2, …, A_n bởi biểu thức (4). Chúng ta gọi tích đề các được xác định bởi (4) (sử dụng phép toán min) là tích đề các chuẩn. thay cho phép toán min, chúng ta có thể sử dụng phép toán T – norm bất kỳ để xác định tích đề các.

Tích đề các của các tập mờ A_1, A_2, …A_n trên các vũ trụ U_1, U_2, …, U_n tương ứng là tập mờ $A = A_1 \times A_2 \times \ldots \times A_n$ trên $U_1 \times U_2 \times \ldots \times U_n$ với hàm thuộc được xác định như sau:

$$\mu_A(x_1, x_2, \ldots, x_n) = \mu_{A1}(x_1) \bigstar \mu_{A2}(x_2) \bigstar \ldots \bigstar \mu_{An}(x) \quad (10)$$

trong đó, phép $\bigstar$ là phép toán T – norm.

11.3. QUAN HỆ MỜ VÀ NGUYÊN LÝ MỞ RỘNG

11.3.1. Quan hệ mờ

Trong mục này chúng ta sẽ trình bày khái niệm quan hệ mờ. Quan hệ mờ đóng vai trò quan trọng trong logic mờ và lập luận xấp xỉ. Khái niệm quan hệ mờ là sự tổng quát hoá trực tiếp của các khái niệm quan hệ (quan hệ rõ). Trước hết chúng ta nhắc lại khái niệm quan hệ.

Giả sử U và V là hai tập. Một quan hệ R từ U đến V (sẽ được gọi là quan hệ hai ngôi, hoặc quan hệ nhị nguyên) là một tập con của tích đề các U

x V. trong trường hợp V = U, Ta nói R là quan hệ trên U. Chẳng hạn, tập R bao gồm tát cả các cặp người (a, b), trong đó a là chồng của b , xác định quan hệ "vợ - chồng" trên một tập người nào đó.

Tổng quát, chúng ta xác định một quan hệ n – ngôi R trên các tập U_1, U_2, …, U_n là một tập con của tích đề các $U_1 \times U_2 \times \ldots \times U_n$.

Khi U và V là các tập hữu hạn, chúng ta sẽ biểu diễn quan hệ R từ U đến V bởi ma trận, trong đó cấc dòng được đánh dấu bởi các phần tử $x \in U$ và các cột được đánh dấu bởi các phân tử $y \in V$. Phần tử của ma trận nằm ở dòng x , cột y là $\lambda_R(x,y)$.

$$\lambda_B(x,y) = \begin{cases} 1 & \text{if} \quad (x,y) \in R \\ 0 & \text{if} \quad (x,y) \notin R \end{cases}$$

Ví dụ 1. Giả sử U = {1, 2, 3} và V = {a, b, c, d}. Giả sử R là quan hệ từ U đến V như sau:

R = {(1,a), (1,d), (2,a), (2,b), (3,c), (3,d)}

Chúng ta có thể biểu diễn quan hệ R bởi ma trận sau

$$R = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$$

$$\begin{matrix} a & b & c & d \end{matrix}$$

Bây giờ chúng ta xét quan hệ "anh em họ gần" trên một tập người U nào đó. Quan hệ này không thể đặc trưng bởi một tập con rõ của tích $U \times U$. Một cáh hợp lý nhất là xác định quan hệ này bởi một tập mờ R trên $U \times U$. Chẳng hạn, $\mu_R(a,b) = 1$ nếu a là anh em ruột của b; $\mu_R(a,b) = 0,9$ nếu a là anh em con chú con bác của b; $\mu_R(a,b) = 0,75$ nếu a là anh em cháu cô cháu cậu của b; …

Một ***quan hệ mờ*** từ U đến V là một tập mờ trên tích các $U \times V$.

Tổng quát, một quan hệ mờ giữa các tập U_1, U_2, …, U_n là một tập mờ trên tích đề các $U_1 \times U_2 \times \ldots \times U_n$

Tương tự như trong trường hợp quan hệ rõ, khi cả U và V là các tập hữu hạn, chúng ta sẽ biểu diễn quan hệ mờ R bởi ma trận, trong đó phần tử nằm ở dòng $x \in U$ cột $y \in V$ là mức độ thuộc của (x,y) vào tập mờ R, tức là $\mu_R(x,y)$.

Ví dụ 2. Chúng ta giả sử U = {1, 2, 3}, V = {a, b, c} và R là quan hệ từ mờ U đến V như sau:

$$R = \frac{0,5}{(1,a)} + \frac{1}{(1,b)} + \frac{0}{(1,c)} + \frac{0,3}{(2,a)} + \frac{0,75}{(2,b)} + \frac{0,8}{(2,c)} + \frac{0,9}{(3,a)} + \frac{0}{(3,b)} + \frac{0,42}{(3,c)}$$

Quan hệ mờ trên được biểu diễn bởi ma trận:

$$R = \begin{matrix} \begin{bmatrix} 0,5 & 1 & 0 \\ 0,3 & 0,75 & 0,8 \\ 0,9 & 0 & 0,42 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \mathbf{a} \quad \mathbf{b} \quad \mathbf{c} & \end{matrix}$$

11.3.2. Hợp thành của các quan hệ mờ

Nhắc lại rằng, hợp thành của quan hệ R từ U đến V với quan hệ S từ V tới W là quan hệ R ° S từ U đến W bao gồm tất cả các cặp $(u, w) \in U \times W$ sao cho có ít nhất một $v \in V$ mà $(u, v) \in R$ và $(v, w) \in S$. Từ định nghĩa này, chúng ta suy ra rằng, nếu xác định R, S và R ° S bởi các hàm đặc trưng λ_R, λ_S và $\lambda_{R\circ S}$ tương ứng thì hàm đặc trưng $\lambda_{R\circ S}$ được xác định bởi công thức:

$$\lambda_{R\circ S}(u, w) = \max_{v\in V} \min[\lambda_R(u, v), \lambda_S(v, w)] \qquad (1)$$

hoặc

$$\lambda_{R\circ S}(u, w) = \max_{v\in V} [\lambda_R(u, v) \cdot \lambda_S(v, w)] \qquad (2)$$

Ví dụ 3. Giả sử $U = \{u_1, u_2\}$, $V = \{v_1, v_2, v_3\}$, $W = \{w_1, w_2, w_3\}$ và

$$R = \begin{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} & \begin{matrix} u_1 \\ u_2 \end{matrix} \\ v_1 \quad v_2 \quad v_3 & \end{matrix} \qquad S = \begin{matrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \\ w_1 \quad w_2 \quad w_3 & \end{matrix}$$

Khi đó ta có:

$$R \circ S = \begin{matrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{matrix} u_1 \\ u_2 \end{matrix} \\ w_1 \quad w_2 \quad w_3 & \end{matrix}$$

Bây giờ, giả sử rằng R là quan hệ mờ từ U đến V và S là quan hệ mờ từ V đến W. Tổng quát hoá các biểu thức (1) và (2) cho các quan hệ mờ, chúng ta có định nghĩa sau:

Hợp thành của các quan hệ mờ R và quan hệ mờ S là quan hệ mờ R°S từ U đến W với hàm thuộc được xác định như sau:

$$\mu_{R\circ S}(u, w) = \max_{v\in V} \min[\mu_R(u, v), \mu_S(v, w)] \qquad (3)$$

hoặc

$$\mu_{R\circ S}(u, w) = \max_{v\in V} [\mu_R(u, v) \cdot \mu_S(v, w)] \qquad (4)$$

Hợp thành được xác định bởi (3) gọi là ***hợp thành max – min***. Hợp thành được bởi xác định (4) được gọi là hợp thành ***max – tích***. Ngoài hai dạng hợp thành trên, chúng ta còn có thể sử dụng một toán tử T – norm bất kỳ để xác định hợp thành của hai quan hệ mờ. Cụ thể là:

$$\mu_{R\circ S}(u, w) = \max_{v\in V} T [\mu_R(u, v), \mu_S(v, w)] \qquad (5)$$

Trong đó, T là toán tử T – norm. Trong (5) khi thay đổi T bởi một toán tử T – norm, chúng ta lại nhận được một dạng hợp thành. Trong các ứng dụng, tuỳ từng trường hợp mà chúng ta lựa chọn toán tử T – norm trong (5). Tuy nhiên hợp thành max – min và hợp thành max – tích là hai hợp thành được sử dụng rộng rãi nhất trong các ứng dụng.

Ví dụ 4. Giả sử R và S là hai quan hệ mờ như sau:

$$R = \begin{matrix} \\ u_1 \\ u_2 \\ u_3 \end{matrix} \begin{matrix} \begin{bmatrix} 0,3 & 1 & 0 & 0,5 \\ 0,7 & 0,1 & 1 & 0 \\ 0 & 0,6 & 1 & 0,3 \end{bmatrix} \\ v_1 \quad v_2 \quad v_3 \quad v_4 \end{matrix} \qquad S = \begin{matrix} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{matrix} \begin{bmatrix} 0,6 & 0 & 1 \\ 0 & 1 & 0,5 \\ 0,4 & 0,3 & 0 \\ 1 & 0,7 & 0,2 \end{bmatrix} \\ w_1 \quad w_2 \quad w_3 \end{matrix}$$

Khi đó hợp thành max – tích của chúng là quan hệ mờ

$$R \circ S = \begin{matrix} \\ u_1 \\ u_2 \\ u_3 \end{matrix} \begin{matrix} \begin{bmatrix} 0,5 & 1 & 0,5 \\ 0,42 & 0,3 & 0,7 \\ 0,4 & 0,6 & 0,3 \end{bmatrix} \\ w_1 \quad w_2 \quad w_3 \end{matrix}$$

Hợp thành max – min của R và S là quan hệ mờ

$$R \circ S = \begin{bmatrix} 0,5 & 1 & 0,5 \\ 0,6 & 0,3 & 0,7 \\ 0,4 & 0,6 & 0,5 \end{bmatrix} \begin{matrix} u_1 \\ u_2 \\ u_3 \end{matrix}$$
$$\begin{matrix} w_1 & w_2 & w_3 \end{matrix}$$

11.3.3. Nguyên lý mở rộng

Nguyên lý mở rộng (được đưa ra bởi Zadeh, 1978) là một trong các công cụ quan trọng nhất của lý thuyết tập mờ. nguyên lý mở rộng cho phép ta xxác định ảnh của một tập mờ qua một hàm.

Giả sử $f : X \to Y$ là một hàm từ không gian X vào không gian Y và A là tập mờ trên X . Vấn đề đặt ra là chúng ta muốn xác định ảnh của tập mờ A qua hàm f . Nguyên lý mở rộng (extention princile) nói rằng, ảnh của tập mờ A qua hàm f là tập mờ B trên Y (ký hiệu $B = f(A)$) với hàm thuộc như sau:

$$\mu_B(y) = \max_{x \in f^{-1}(y)} \mu_A(y)$$

Ví dụ 5. Giả sử $U = \{0, 1, \ldots, 10\}$ và $f : U \to U$ là hàm

$$f(x) = \begin{cases} 2x & \text{if} \quad x \le 5 \\ x & \text{if} \quad x > 5 \end{cases}$$

Giả sử A là tập mờ trên U:

$$A = \frac{1}{0} + \frac{1}{1} + \frac{1}{2} + \frac{0,9}{3} + \frac{0,7}{4} + \frac{0,5}{5} + \frac{0,1}{6} + \frac{0}{7} + \frac{0}{8} + \frac{0}{9} + \frac{0}{10}$$

Khi đó ta có ảnh của A là tập mờ sau:

$$B = f(A) = \frac{1}{0} + \frac{0}{1} + \frac{1}{2} + \frac{0}{3} + \frac{1}{4} + \frac{0}{5} + \frac{0,9}{6} + \frac{0}{7} + \frac{0,7}{8} + \frac{0}{9} + \frac{0,5}{10}$$

11.4. LOGIC MỜ

11.4.1. Biến ngôn ngữ và mệnh đề mờ

BIẾN NGÔN NGỮ

Chúng ta xét một biến, chẳng hạn biến "nhiệt độ" , biến này có thể nhận các giá trị số: 13_0C, 25_0C, 37_0C,… Song trong đời sống hàng ngày, chúng ta vẫn thường nói "nhiệt độ cao", "nhiệt độ trung bình", nhiệt độ

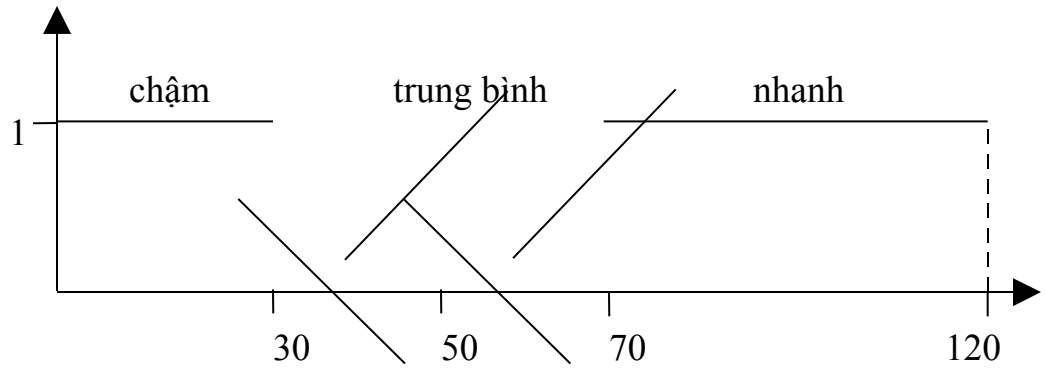
thấp". Chúng ta có thể xem biến "nhiệt độ", lấy các từ "cao", "trung bình", "thấp" làm giá trị của nó. Khi một biến nhận cấc từ trong ngôn ngữ tự nhiên làm các giá trị thì các biến đó được gọi là ***biến ngôn ngữ*** (linguistic variable).

Khái niệm biến ngôn ngữ đã được Zadeh đưa ra năm 1973. chúng ta có thể đưa ra định nghĩa hình thức về biến ngôn ngữ như sau:

Một ***biến ngôn ngữ*** được xác định bởi bộ bốn (x, T, U, M), trong đó

- x là tên biến. Chẳng hạn x có thể là "nhiệt độ", "tốc độ", "độ ẩm", "áp suất", …
- T là một tập nào đó các từ (các giá trị ngôn ngữ) mà biến x có thể nhận. Chẳng han, nếu x là "tốc độ", thì T có thể là T = {chậm, trung bình, nhanh}.
- U là miền các giá trị vật lý mà x , với tư cách là biến số, có thể nhận. Chẳng hạn, nếu x là "tốc độ" của một xe máy và tốc độ tối đa là 120 (km\h) thì U = [0..120].
- M là luật ngữ nghĩa, ứng với mỗi từ $t \in T$ với một tập mờ A_t trên vũ trụ U.

Ví dụ 1. x là "tốc độ**",** T = {chậm, trung bình, nhanh} và các từ "châm", "trung bình", "nhanh" được xác định bởi các tập mờ trong hình 11.9.



Hình 11.9. Các tập mờ biểu diễn các giá trị ngôn ngữ "chậm", "nhanh", "trung bình".

Từ định nghĩa trên, chúng ta có thể nói rằng biến ngôn ngữ là biến có thể nhận giá trị các tập mờ trên một miền nào đó.

MỆNH ĐỀ MỜ

Trong logic cổ điển (logic vị từ cấp một), một mệnh đề phân tử p(x) là một phát biểu có dạng:

x là P (1)

trong đó x là ký hiệu một đối tượng nằm trong một tập các đối tượng U nào đó (hay nói cách khác,x là một biến nhận giá trị trên miền U), còn P là một tính chất nào đó của các đối tượng trong U. Chẳng hạn, các mệnh đề

n là số nguyên tố

x là người Ấn Độ

Trong các mệnh đề (1) của logic cổ điển, tính chất P cho phép ta xác định một tập con rõ A của U sao cho $x \in A$ và nếu chỉ x thoả mãn tính chất P. Chẳng hạn, tính chất "là số nguyên tố" xác định một tập con rõ của tập tất cả các số nguyên tố, đó là tập tất cả các số nguyên tố.

Nếu chúng ta ký hiệu Truth(P(x)) là giá trị chân lý của mệnh đề rõ (1) thì

$$\text{Truth}(P(x)) = \lambda_A(x) \quad (2)$$

Trong đó, $\lambda_A(x)$ là hàm đặc trưng của tập rõ A, tập rõ A được xác định bởi tính chất P.

Một ***mệnh đề mờ*** phân tử cũng có dạng cũng có dạng tương tự như (1), chỉ có điều ở đây P không phải là một tính chất chính xác, mà là một tính chất không rõ ràng, mờ. Chẳng hạn, các mệnh đề "tốc độ là nhanh", "áp suất là cao", "nhiệt độ là thấp", … là các mệnh đề mờ. Chúng ta có định nghĩa sau:

Một ***mệnh đề mờ*** phân tử có dạng

x là t (3)

trong đó, x là biến ngôn ngữ, còn t là giá trị ngôn ngữ của x.

Theo định nghĩa biến ngôn ngữ, từ t trong (3) được xác định bởi một tập mờ A trên vũ trụ U. Do đó chúng ta còn có thể định nghĩa: mệnh đề mờ phân tử là phát biểu có dạng

x là A (4)

trong đó, x là biến ngôn ngữ, còn A là một tập mờ trên miền U các giá trị vật lý của x.

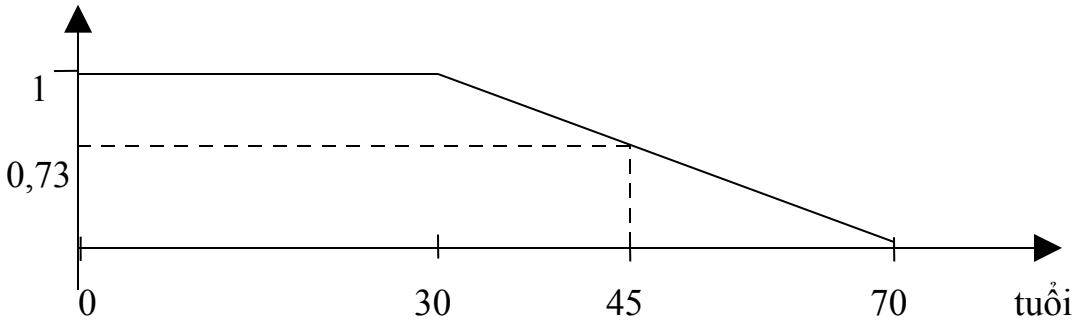
Logic cổ điển là logic hai trị, một mệnh đề chỉ có thể là đúng (giá trị chân lý là 1) hoặc sai (giá trị chân lý là 0). Logic mờ là mở rộng của logic cổ điển. Trong logic mờ, giá trị chân lý của một mệnh đề mờ là một số nằm trong [0,1].

Chúng ta ký hiệu P(x) là mệnh đề mờ (3), hoặc (4). Giá trị chân lý Truth(P(x)) của nó được xác định như sau:

$$\text{Truth}(P(x)) = \mu_A(x) \qquad (5)$$

Điều đó có nghĩa là giá trị chân lý của mệnh đề mờ P(x) = "x là A" là mức độ thuộc của x vào tập mờ A.

Ví dụ 2. Giả sử P(x) là mệnh đề mờ "tuổi là trẻ". Giả sử tập mờ A = "tuổi trẻ" được cho trong hình 11.10. và $\mu_A(45) = 0{,}73$. Khi đó mệnh đề mờ "tuổi 45 là trẻ" có giá trị chân lý là 0,73.



***Hình 11.10*. Tập mờ "tuổi trẻ".**

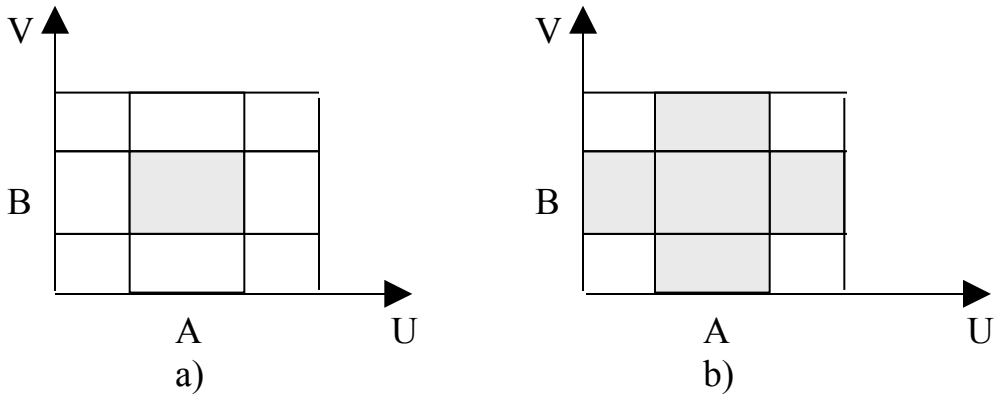
11.4.2. Các mệnh đề mờ hợp thành

Cũng như trong logic cổ điển, từ các mệnh đề phân tử, bằng cách sử dụng các kết nối logic: $\wedge$ (and), $\vee$ (or), $\neg$ (not) chúng ta sẽ tạo ra các mệnh đề mờ hợp thành.

Giả sử mệnh đề rõ P(x) được minh hoạ như tập con rõ A trong vũ trụ U, (cần lưu ý rằng, điều đó có nghĩa là $\text{Truth}(P(x)) = 1 \Leftrightarrow x \in A$), và mệnh

đề rõ Q(y) được minh hoạ như tập con rõ B trong V. Từ bảng chân lý của các phép toán $\neg$, $\wedge$, $\vee$ trong logic cổ điển chúng ta suy ra:

- Mệnh đề $\neg$P(x) được minh hoạ như tập rõ $\overline{A}$.
- Mệnh đề P(x) $\wedge$ Q(y) được minh hoạ như quan hệ rõ A $\times$ B trên U $\times$ V (xem hình 11.11.a).
- Mệnh đề P(x) $\vee$ Q(y) được minh hoạ như quan hệ rõ (A $\times$ V) $\cup$ (U $\times$ B) (xem hình 11.11.b).



Hình 11.11.

a) **Quan hệ rõ minh hoạ mệnh đề P(x) $\wedge$ Q(y)**

b) **Quan hệ rõ minh hoạ mệnh đề P(x) $\vee$ Q(y)**

Chuyển sang logic mờ, giả sử rằng P(x) là mệnh đề mờ được minh hoạ như tập mờ A trên U và Q(y) là mệnh đề mờ được minh hoạ như tập mờ B trên V. Tổng quát hoá từ các mệnh đề rõ, chúng ta xác định như sau:

- Mệnh đề mờ ¬P(x) (not P(x)) được minh hoạ như phủ định mờ $\overline{A}$ của tập mờ A:

  $\mu_A(x) = C(\mu_A(x))$ (6)

  Trong đó C là hàm phần bù đã được định nghĩa trong 11.2.2. Khi C là hàm phần bù chuẩn, ta có:

  $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$

- Mệnh đề mờ $P(x) \wedge Q(y)$ (P(x) and Q(y)) được minh hoạ như quan hệ mờ $A \wedge B$, trong đó $A \wedge B$ được xác định là tích Đề-các mờ của A và B. Từ định nghĩa tích Đề-các mờ, ta có:

  $\mu_{A \wedge B}(x, y) = T(\mu_A(x), \mu_B(y))$ (8)

  trong đó, T là một T – norm nào đó. Với T là phép lấy min, ta có:

  $\mu_{A \wedge B}(x, y) = \min(\mu_A(x), \mu_B(y))$ (9)

- Mệnh đề mờ $P(x) \vee Q(y)$ (P(x) or Q(y)) được minh hoạ như quan hệ mờ $A \vee B$, trong đó $A \vee B$ được xác định là:

  $A \vee B = (A \times V) \cup (U \times B)$

  Từ định nghĩa tích Đề-các mờ và hợp mờ, ta có:

  $\mu_{A \vee B}(x, y) = S(\mu_A(x), \mu_B(y))$ (10)

  Trong đó, S là phép toán S – norm nào đó. Khi S là phép lấy max, ta có:

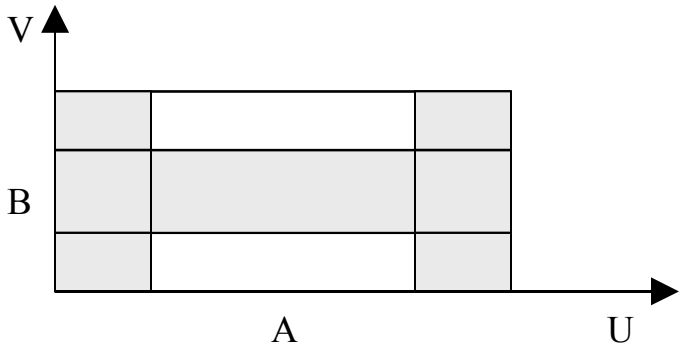
  $\mu_{A \vee B}(x, y) = \max(\mu_A(x), \mu_B(y))$ (11)

11.4.3. Kéo theo mờ - Luật if-then mờ

Trước hết, chúng ta xét phép kéo theo trong logic cổ điển. Gi P(x) và Q(y) là các mệnh đề rõ được minh hoạ như các tậpp rõ A và B trên U và V tương ứng. Từ bảng chân lý của phép kéo theo trong logic cổ điển, chúng ta suy ra rằng, mệnh đề $P(x) \Rightarrow Q(y)$ được minh hoạ như quan hệ rõ R trên U x V (xem hình 11.12):

$$R = (\overline{A} \times V) \cup (U \times B) \quad (12)$$

hoặc $$R = (A \times V) \cup (A \times B) \quad (13)$$



***Hình 11.12*. Quan hệ rõ minh hoạ mệnh đề P(x)) ⇒ Q(y) .**

Trong logic mờ, một kéo theo mờ có dạng :

< Mệnh đề mờ> ⇒ <mệnh đề mờ> (14)

Hay một cách viết khác:

If <mệnh đề mờ> ***then*** <mệnh đề mờ> (15)

Dạng này được gọi là luật if - then mờ. Chẳng hạn các phát biểu:

If "nhiệt độ cao" ***then*** "áp suất lớn"

If "tốc độ nhanh" ***then*** "ma sát lớn"

là các luật if – then mờ. Một vấn đề đặt ra là, chúng ta cần phải hiểu ngữ nghĩa của (14) như thế nào? Chúng ta xét kéo theo mờ sau đây

$$P(x) \Rightarrow Q(y) \qquad (16)$$

Trong đó P(x) là mệnh đề mờ được minh hoạ như tập mờ A trên U và Q(y) là mệnh đề mờ được minh họa như tập mờ B trên V.

Tổng quát hoá từ (12) và (13), chúng ta có thể hiểu kéo theo mờ (16) như một quan hệ mờ R trên U × V được xác định bởi (12) hoặc (13) nhưng trong đó các phép toán là các phép toán trên tập mờ.

Từ (12) và (13) và định nghĩa của các phép toán lấy phần bù mờ, tích đề và các tập mờ, chúng ta có:

$$\mu_R(x,y) = S\,(C(\mu_A(x)),\ \mu_B(y)) \qquad (17)$$

hoặc $\mu_R(x,y) = S(C(\mu_A(x)),\ T(\mu_A(x),\mu_B(y)))$ (18)

trong đó, C là hàm phần bù, S là toán tử S – norrm, T là toán tử T – norm.

Như vậy, kéo theo mờ (16) được minh hoạ như mờ Rvới hàm thuộc được xác định bởi (17) hoặc (18). Trong (17) và (18) ứng với mỗi cách lựa chọn các hàm C, S, T chúng ta nhận được một quan hệ mờ R minh hoạ cho kéo theo mờ (16). Như vậy kéo theo mờ (16) có thể được minh hoạ bởi rất nhiều quan hệ mờ khác nhau. Câu hỏi mới suất hiện là dựa vào tiêu chuẩn nào để lựa chọn quan hệ mờ R cho kéo theo mờ (16)? Chúng ta sẽ trả lời câu hỏi này ở mục 11.5. Coàn bây giờ chúng ta sẽ đưa ra một số kéo theo mờ quan trọng.

- **Kéo theo Dienes – Rescher**

Trong (17), nếu thay S bởi phép toán lấy max và C bởi phần hàm bù chuẩn, chúng ta sẽ nhận được quan hệ mờ R với hàm thuộc

$$\mu_R(x,y) = \max(1 - \mu_A(x), \mu_B(y)) \quad (19)$$

- **Kéo theo Lukasiewicz**

Nếu chúng ta sử dụng phép hợp Yager với w = 1 cho thay S và C là phần bù chuẩn thì từ (17) chúng ta nhận được quan hệ Mờ R với hàm thuộc:

$$\mu_R(x,y) = \min(1, 1 - \mu_A(x) + \mu_B(y)) \quad (20)$$

- **Kéo theo Zadeh**

Trong (18), nếu sử dụng s là max, T là min và C là hàm phần bù chuẩn, chúng ta nhận được quan hệ mờ R với hàm thuộc:

$$\mu_R(x,y) = \max(1 - \mu_A(x), \min(\mu_A(x), \mu_B(x))) \quad (21)$$

Trên đây chúng ta đã hiểu kéo theo mờ $P(x) \Rightarrow Q(y)$ như quan hệ mờ R được xác định bởi (17) hoặc (18). Cách hiểu như thế là sự tổng quát hoá trực tiếp ngữ nghĩa của kéo theo cổ điển. Tuy nhiên, chúng ta cũng có thể hiểu: kéo theo mờ $P(x) \Rightarrow Q(y)$ chỉ có giá trị chân lý lớn khi cả P(x) và Q(y) đều có giá trị chân lý lớn, tức là chúng ta có thể minh hoạ kéo theo mờ (16) như quan hệ mờ R được xácđịnh là tích các đề mờ của A và B.

$$R = A \times B \quad (22)$$

Từ (22) chúng ta xác định được hàm thuộc của quan hệ mờ R:

$$\mu_R(x,y) = T(\mu_A(x), \mu_B(y) \quad (23)$$

với T là toán tử T – norm.

- **Kéo theo Mamdani**

Trong (23) nếu sử dụng T là phép toán lấy min hoặc tích đại số, chúng ta có:

$$\mu_R(x,y) = \min(\mu_A(x), \mu_B(y)) \quad (24)$$

hoặc $$\mu_R(x,y) = (\mu_A(x), \mu_B(y) \quad (25)$$

Kéo theo mờ (16) được hiểu như quan hệ mờ R với hàm thuộc được xác định bởi (24) và (25) được gọi là kéo theo mamdani. Kéo theo mamdani được sử dụng rộng rãi nhất trong các quan hệ mờ.

Kéo theo mờ được xác định bởi (23), chẳng hạn, kéo theo Mamdani, là kéo theo có tính địa phương. Còn các kéo theo mờ được xác định bởi (17) hoặc (18) , chẳng hạn các kéo theo Rienes – Rescher, Lukasiewiez, zadeh, là các kéo theo có tính toàn cục. Phần lớn các luật if – then mờ có tính địa phương. Điều này lý giải cho câu hỏi: tại sao kéo theo mamdani được sử dụng rộng rãi trong các ứng dụng.

Ví dụ 3. Xét luật if – then mờ sau:

if "x là A" ***then*** "y là B"

trong đó, A và B là các tập mờ sau:

$$A = \frac{1}{1} + \frac{0,7}{2} + \frac{0,1}{3}$$

$$B = \frac{0}{a} + \frac{0,3}{b} + \frac{1}{c} + \frac{1}{d}$$

Áp dụng các công thức (19), (20), (21) và (22) chúng ta xác định được các quan hệ mờ sau đây:

- Quan hệ Dienes – Rescher

$$R = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{bmatrix} 0 & 0,3 & 1 & 1 \\ 0,3 & 0,3 & 1 & 1 \\ 0,9 & 0,9 & 1 & 1 \end{bmatrix} \\ a \quad b \quad c \quad d \end{matrix}$$

- Quan hệ Lukasiewicz

$$R = \begin{matrix} \begin{bmatrix} 0 & 0,3 & 1 & 1 \\ 0,3 & 0,6 & 1 & 1 \\ 0,9 & 1 & 1 & 1 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ a \quad b \quad c \quad d \end{matrix}$$

- Quan hệ Zadeh

$$R = \begin{matrix} \begin{bmatrix} 0 & 0,3 & 1 & 1 \\ 0,3 & 0,3 & 0,7 & 0,7 \\ 0,9 & 0,9 & 0,9 & 0,9 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ a \quad b \quad c \quad d \end{matrix}$$

- Quan hệ Mamdami

$$R = \begin{matrix} \begin{bmatrix} 0 & 0,3 & 1 & 1 \\ 0 & 0,3 & 0,7 & 0,7 \\ 0 & 0,1 & 0,1 & 0,1 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ a \quad b \quad c \quad d \end{matrix}$$

11.4.4. Luật Modus – Ponens tổng quát

Trong logic cổ điển, luật Modus – Pones phát biểu rằng, từ hai mệnh đề if P(x) then Q(y) và P(x), chúng ta có thể suy ra mệnh đề mới Q(y). Luật Modus – Pones là một trong cac luật suy diễn được sử dụng rộng rãi nhất trong các lập luận. chúng ta có thể tổng quát hoá luật này cho logic mờ.

Luật Modus – Ponens tổng quát trong logic mờ phát biểu rằng, từ hai mệnh đề if "x là A" then "y là B" và "x là A' ", chúng ta có thể suy ra mệnh đề mờ mới "y là B' " sao cho nếu A' càng gần với A thì B' càng gần với B, trong đó A và A' là các tập mờ trên U, còn B và B' là các tập mờ trên V.

Chúng ta viết luật Modus – Ponens tổng quát dưới dạng:

Giả thiết 1 : ***if*** "x là A" ***then*** "y là B"

Giả thiết 2 : "x là A'" (26)

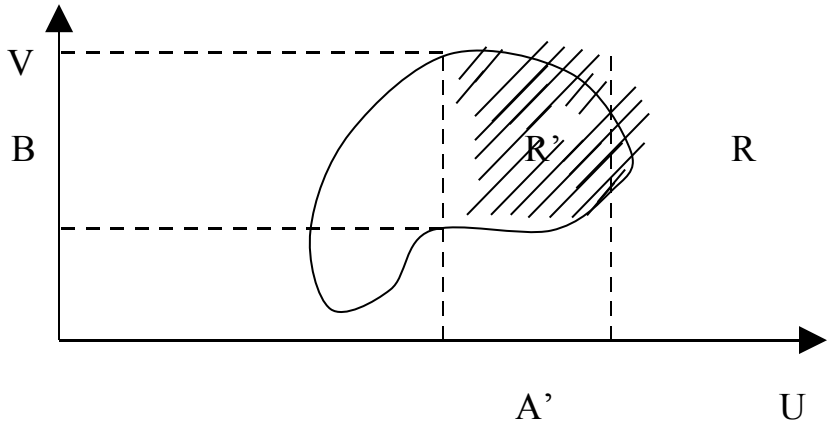
Kết luận : "y là B' "

Cần lưu ý rằng, khác với luật Modus – pones cổ điển, trong luật Modus – Ponens tổng quát giả thiết 1 là luật if – then mờ với điều kiện là

mệnh đề "x là A", trong khi đó giả thiết 2 là mệnh đề "x là A' " (dữ liệu thu được từ quan sát),mệnh đề này không đòi hỏi chính xác phải trùng với với điều kiện của luật if – then.

Vấn đề đặt ra là, làm thế nào để đánh giá được tập mờ B' trong kết luận được suy ra "y là B' " ?

Như chúng ta đã biết (xem mục 14.4.3), luật if – then mờ if "x là A" then "y là B" được minh hoạ như quan hệ mờ R trên không gian tích U x V. Từ tập mờ A' chúng ta xây dựng mở rộng hình trụ của nó A' x V trên U x V. Gọi giao của A'x V với quan hệ R là R'. chiếu quan hệ mờ R' lên U, chúng ta nhận được tập mờ B' (xem hình 11.13).



Hình 11.13. Phương pháp xác định tập mờ B'

Vì R' = R ∩ (A' x V), chúng ta có :

$$\mu_{R'}(x,y) = T\,(\mu_R(x,y), \mu_{A'}(x))$$

Mặt khác, vì B' là hình chiếu của R' trên U, do đó:

$$\mu_{B'}(y) = \sup_{x \in U} \mu_{R'}(x, y)$$

Từ hai hệ thức trên, chúng ta nhận được:

$$\mu_{B'}(y) = \sup_{x \in U} T(\mu_{R'}(x, y), \mu_{A'}(x)) \qquad (27)$$

trong đó, T là phép T – norm.

Trong luật Modus – Ponens tổng quát

Như vậy, trong luật Modus – Ponens tổng quát (26), từ các giả thiết của luật chúng ta suy ra kết luận "y là B", trong đó B' là tập mờ được xác định bởi (27). Trong (27), với T là phép lấy min, chúng ta có:

$$\mu_{B'}(y) = \sup_{x \in U} \min(\mu_R(x, y), \mu_{A'}(x)) \qquad (28)$$

Chú ý rằng, trong (27) R là quan hệ được sinh ra bởi luật if – then mờ if "x là A" then "y là B". Chúng ta có thể sử dụng R là một trong các quan hệ mờ (19), (20),(21), (23) hoặc bất kỳ quan hệ mờ nào khác được xác định bởi (17) hoặc (18).

Ví dụ 4. Xét luật if-then mờ trong ví dụ 3. Giả sử chúng ta có "x là A' " với A' là tập mờ sau:

$$A' = \frac{0,5}{1} + \frac{1}{2} + \frac{0,3}{3}$$

Khi đó chúng ta suy ra "y là B' " với B' là tập mờ được xác định như sau. Chúng ta đánh giá B'theo công thức (28). Nếu R là quan hệ lukasiwiez thì:

$$B' = \frac{0,3}{a} + \frac{0,6}{b} + \frac{1}{c} + \frac{1}{d}$$

Nếu R là quan hệ Zadeh thì:

$$B' = \frac{0,3}{a} + \frac{0,6}{b} + \frac{0,7}{c} + \frac{0,7}{d}$$

Nếu sử dụng R là quan hệ Mamdani thì:

$$B' = \frac{0}{a} + \frac{0,3}{b} + \frac{0,7}{c} + \frac{0,7}{d}$$

Ví dụ 5. Giả sử quan hệ giữa nhiệt độ và áp suất trong một thiết bị được biểu diễn bởi luật sau:

If "nhiệt độ là cao" ***then*** "áp suất là lớn".

Giả sử nhiệt độ (tính bằng độ C) nhận giá trị trong miền U = [30,35, 40, 45] và áp suất (tính bằng atmotphe) nhận giá trị trong miền V = [50, 55, 60, 65]. Giả sử

$$A = \text{"nhiệt độ cao"} = \frac{0}{30} + \frac{0,3}{35} + \frac{0,9}{40} + \frac{1}{45}$$

$$B = \text{“áp suất lớn”} = \frac{0}{50} + \frac{0,5}{55} + \frac{1}{60} + \frac{1}{65}$$

Xem luật if – then như kéo theo Mamdani (25) chúng ta nhận được quan hệ mờ sau:

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0,15 & 0,3 & 0,3 \\ 0 & 0,45 & 0,9 & 0,9 \\ 0 & 0,5 & 1 & 1 \end{bmatrix} \begin{matrix} 30 \\ 35 \\ 40 \\ 45 \end{matrix}$$

$$50 \quad 55 \quad 60 \quad 65$$

Bây giờ, giả sử chúng ta biết “nhiệt độ là trung bình” và

$$A' = \text{“nhiệt độ trung bình”} = \frac{0,6}{30} + \frac{1}{35} + \frac{0,8}{40} + \frac{0,1}{45}$$

Áp dụng công thức (28) chúng ta suy ra B’ như sau:

$$B' = \frac{0}{50} + \frac{0,45}{55} + \frac{0,8}{60} + \frac{0,8}{65}$$

11.5. HỆ MỜ

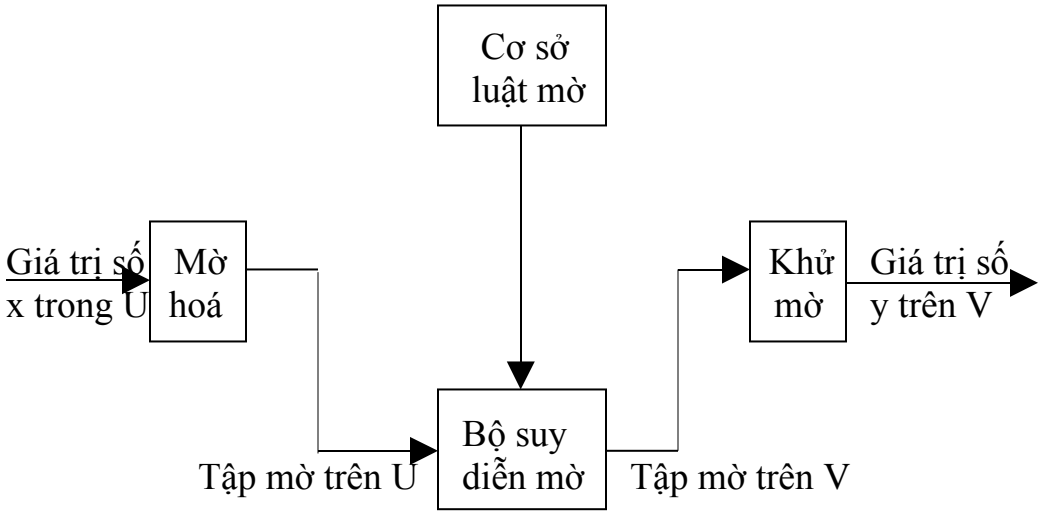
Trong mục này chúng ta sẽ trình bày các nguyên lý tổng quát để thiết kế một hệ mờ. Hệ mờ là hệ dựa trên tri thức, tri thức trong hệ mờ được biểu diễn dưới dạng các luật if – then mờ. hệ mờ đã được áp dụng thành công trong rất nhiều lĩnh vực: điều khiển tự động, sử lý tin hiệu, truyền thông các hệ chuyên gia trong y học, các hoạt động quản lý kinh doanh,…Tuy nhiên, những áp dụng thành công nhất của hệ mờ vẫn là các áp dụng trong vấn đề điều khiển: hệ điều khiển mờ (fuzzy control systems)

11.5.1.Kiến trúc của hệ mờ

Thành phần trung tâm của hệ mờ là ***cơ sở luật mờ*** (fuzzy rule base). Cơ sở luật mờ bao gồm các luật if – then mờ mô tả tri thức của các chuyên gia về một lĩnh vực áp dụng nào đó.

Thành phần thứ hai trong hệ mờ là ***bộ suy diễn mờ*** (fuzzy infernce engine), nhiệm vụ của nó là kết hợp các luật if – then mờ trong cơ sở luật mờ để tạo thành một phép biến đổi : chuyển mỗi tập mờ đầu vào trên không gian U thành một tập mờ đầu ra trên không gian V.

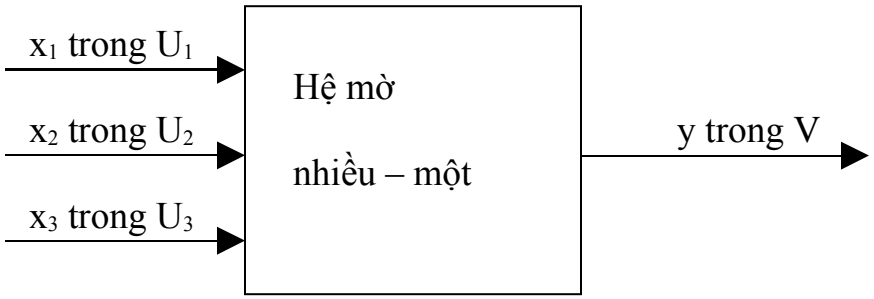
Tuy nhiên, trong thực tế dữ liệu mà hệ thu nhận được từ môi trường là các giá trị số và giá trị mà hệ cho ra cũng cần phải là giá trị số chứ không thể là các tập mờ. Vì vậy cần phải có các giao diện để hệ tương tác với môi trường, một giao diện biến đổi các giá trị số thành một tập mờ trên U. Giao diện này được gọi là ***mờ hoá*** (fuzzifier). Một giao diện khác biến đổi một tập mờ trên V thành một giá trị số. Giao diện này được gọi là ***khử mờ*** (defuzzifier). Như vậy, chúng ta có kiến trúc cơ bản của một hệ mờ như trong hình 11.14.



Hình 11.14. Kiến trúc cơ bản của hệ mờ.

Khi đầu vào x của hệ mờ trong hình 11. 14 là một vectơ trong không gian tích $U = U_1 x\ U_2\ x \ldots x\ U_n$, tức là $x = (x_1, x_2, \ldots, x_n)$ với $x_i \in U_i \subset R(I = 1, \ldots, n)$, chúng ta có hệ mờ nhiều đầu vào - một đầu ra, xem hình 11.15

Một hệ mờ nhiều đầu vào - nhiều đầu ra, chẳng hạn 3 đầu ra có thể phân tích thành 3 hệ đầu vào - một đầu ra. Vì vậy sau này chúng ta chỉ xét các hệ nhiều đầu vào - một đầu ra 9gọi tắt là hệ nhiều - một).



Hình 11.15. **Hệ mờ n đầu vào - một đầu ra.**

Sau đây chúng ta sẽ nghiên cứu kỹ hơn từng thành phần của hệ mờ nhiều - một.

11.5.2 Cơ sở luật mờ

Cơ sở luật mờ của hệ mờ n đầu vào - một đầu ra bao gồm m luật if-then mờ, luật mờ thứ k có dạng;

If "x_1 là A_{k1} và x_2 là A_{k2}" và … và "x_n là A_{kn}" ***then*** "y là B_k".

$$(k = 1 , 2,\ldots m) \qquad (1)$$

Trong đó, x_i là các biến đầu vào. A_{ki} là tập mờ trên $U_1 \subset R$ (i = 1, 2, …, n); y là biến đầu ra và B_k là tập mờ trên $V \subset R$.

Các luật mờ dạng (1) được gọi là các ***luật if – then mờ chuẩn tắc***. Cần lưu ý rằng, luật if – then mờ không chuẩn tắc có thể được biến đổi để đưa về dạng if – then mờ chuẩn tắc.

Một vấn đề quan trọng được đặt ra là, khi xây dựng một hệ mờ trong một lĩnh vực áp dụng, chúng ta làm thế nào để áp dụng được các luật if – then mờ để đưa vào cơ sở luật mờ?

Các luật if – then mờ được xây dựng nên từ tri thức của các chuyên gia trong lĩnh vực áp dụng, hoặc được tìm ra từ một tập các cặp dữ liệu (x,y), trong đó $x = (x_1, x_2, \ldots x_n)$ là các giá trị đầu vào và y là giá trị đầu ra tương ứng với x. Từ quan sát, phỏng vấn, hoặc thực nghiệm chúng ta thu được tập các cặp dữ liệu (x,y) này. Có rất nhiều phương pháp thiết kế cơ sở luật mờ từ tập các cặp dữ liệu vào – ra (x,y). Bạn đọc có thể tham khảo các phương pháp này trong các tài liệu về hệ mờ, chẳng hạn trong [24].

11.5.3. Bộ suy diễn mờ

Chúng ta sẽ nghiên cứu phương pháp thiết kế bộ suy diễn mờ trong trường hợp cơ sở luật mờ gồm m luật if – then mờ chuẩn tắc dạng (1).

Như chúng ta đã nói, bộ suy diễn mờ thực hiện nhiệm vụ kết hợp các luật if – then mờ trong cơ sở luật mờ tạo thành một phép biến đổi: chuyển một tập mờ A' trên U thành một tập mờ B' trên V. tập mờ B' phụ thuộc vào cách chúng ta quan niệm về các luật trong cơ sở luật . có hai cách nhìn:

- Cách nhìn thứ nhất: Người ta xem các luật if – then mờ trong cơ sở luật là các luật độc lập. Điều này có nghĩa là, với mỗi luật khi điều kiện của luật được thoả mãn thì luật sẽ cho ra hệ quả.
- Cách nhìn thứ hai: Đối lập với cách nhìn trên, chúng ta xem rằng, chỉ khi nào tất cả các điều kiện của các luật trong cơ sở luật được thoả mãn thì hệ luật mới cho ra hệ quả.

1. Phương pháp suy diễn kết hợp

Tư tưởng của phương pháp này là: chúng ta kết hợp m luật trong cơ sở luật thành một luật if – then mờ, sau đó chúng ta áp dụng luật suy diễn Modus – Ponens tổng quát.

Chúng ta xem mỗi luật (1) như luật

If "x là A_k" ***then*** "y là B_k" (2)

Trong đó, $x = (x_1, x_2, \ldots, x_n)$ và A_k là tập mờ trên không gian tích $U = U_1 \times U_2 \times \ldots \times U_n$, A_k là tích đề các mờ của $A_{k_1}, A_{k_2}, \ldots, A_{k_n}$.

$$A_k = A_{k_1} \times A_{k_2} \times \ldots \times A_{k_n}$$

Do đó, chúng ta có:

$$\mu_A(x) = \mu_{A_{k_1}}(x_1) \bigstar \mu_{A_{k_2}}(x_2) \bigstar \ldots \bigstar \mu_{A_{k_n}}(x_n) \qquad (3)$$

trong đó, $\bigstar$ là một phép toán T – norm.

Mỗi luật mờ (2) được minh hoạ như quan hệ mờ R_k. R_k được xác định từ A_k và B_k theo các phương pháp đã trình bày trong mục 11.4. Chẳng hạn, chúng ta có thể sử dụng các công thức (19), (20), (21) hoặc (24) trong mục 11.4.

Sau đó m luật dạng (2) được kết hợp thành luật:

If "x là A" ***then*** "y là B"

Luật này được minh hoạ như quan hệ R, với R được xác định như sau:

- Với cách nhìn thứ nhất

$$R = \bigcup_{k=1}^{m} R_k \qquad (4)$$

$$\mu_R(x,y) = \mu_{R1}(x,y) + \ldots + \mu_{Rm}(x,y)$$

trong đó, + là phép toán S – norm.

- Với cách nhìn thứ hai

$$R = \bigcap_{k=1}^{m} R_k \qquad (5)$$

$$\mu_R(x,y) = \mu_{R1}(x,y) \bigstar \ldots \bigstar \mu_{Rm}(x,y)$$

Trong đó, ★ là phép toán T – norm.

Như vậy, toàn bộ cơ sở luật mờ được lập thành một luật if – then mờ được minh họa bởi quan hệ R với hàm thuộc được xác định bởi (4) hoặc (5).

Bây giờ, nếu chúng ta đưa vào tập mờ A' trên U thì áp dụng luật suy diễn Modus – Ponens tổng quát chúng ta suy ra tập mờ B' trên V với hàm thuộc

$$\mu_{B'}(y) = \sup_{x \in U} [\mu_R(x,y) \bigstar \mu_{A'}(x,y)] \qquad (6)$$

2. Phương pháp suy diễn cá thể

Trong phương pháp này, mỗi luật dạng (1) được xem như luật dạng (2) trong đó A_k được xác định bởi (3).

Bây giờ chúng ta áp dụng luật suy diễn Modus – Ponens tổng quát cho mỗi dạng luật (2), Khi đưa vào tập mờ A' trên U, chúng ta tính tập mờ B'_k trên V.

$$\mu_{B'_k}(y) = \sup_{x \in U} [\mu_{R_k}(x,y) \bigstar \mu_{A'}(x,y)] \qquad (7)$$

Sau đó chúng ta kết hợp các tập mờ $B'_1,\ldots,B'_m$ để nhận được tập mờ đầu ra B'. Tuỳ theo cách nhìn các luật trong cơ sở luật mà chúng ta lấy B' là hợp mờ hoặc giao mờ của các tập $B'_1,\ldots,B'_m$. Tức là

$$\mu_{B'}(y) = \mu_{B'1}(y) + \ldots + \mu_{B'm}(y) \qquad (7')$$

hoặc

$$\mu_{B'}(y) = \mu_{B'1}(y) \bigstar \ldots \bigstar \mu_{B'm}(y) \qquad (8)$$

Nhận xét:

Chúng ta có rất nhiều cách lựa chọn để tính được tập mờ đầu ra B' tương ứng với tậpp mờ đầu vào A':

- Chọn phương pháp suy diễn kết hợp hay cá thể ?
- Chọn các quan hệ R_k là quan hệ nào? Chúng ta có thể chọn R_k là một quan hệ kéo theo Zadeh, lukasieview hoặc mamdani…
- Trong các công thức (3), (4), (5), (8) và (9) chúng ta cũng có rất nhiều cách lựa chọn các phép toán S – norm và T – norm.

Câu hỏi được đặt ra là, dựa vào tiêu chuẩn nào để lựa chọn? Tiêu chuẩn chung là: sự lựa chọn sao cho bộ suy diễn cho ra kết quả phù hợp với những mong muốn mà các chuyên gia kỳ vọng vào hệ và đảm bảo hiệu quả tính toán, tức là với tập mờ đầu vào A', bộ suy diễn tính nhanh được tập mờ đầu ra B'.

11.5.4. Mờ hoá

Mờ hoá (fuzzifier) là quá trình biến đổi vectơ $x = (x_1, x_2,\ldots,x_n)$ các giá trị số, $x \in U \subset R^n$, thành một tập mờ A' trên U.(A' sẽ là tập mờ đầu vào cho bộ suy diễn mờ). Mờ hoá phải thoả mãn các tiêu chuẩn sau :

- Điểm dữ liệu x phải có mức độ thuộc cao vào các tập mờ A'.
- Vectơ $x = (x_1,\ldots,x_n)$ thu được từ môi trường bằng quan sát có thể sai lệch do nhiễu. tập mờ A' phải biểu diễn được tính gần đúng của dữ liệu x.
- Hiệu quả tính toán đơn giản cho các tính tôảntng bộ suy diễn.

Sau đây chúng ta đưa ra một số phương pháp mờ hoá được sử dụng rộng rãi.

1. Mờ hoá đơn thể

Mỗi điểm dữ liệu x được xem như một ***đơn thể mờ*** tức là tập mờ A' có hàm thuộc

$$\mu_{A'}(u) = \begin{cases} 1 & \text{if} \quad u = x \\ 0 & \text{if} \quad u \in U \text{ and } u \neq x \end{cases} \tag{10}$$

2. Mờ hoá Gauss

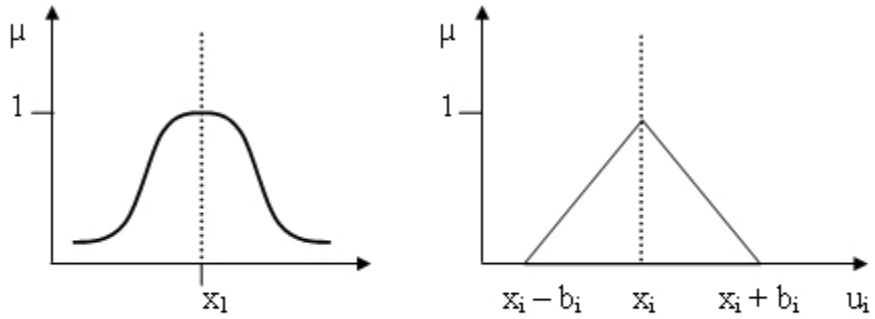
Mỗi giá trị số x_1 (i = 1,…n) trong vectơ $x = (x_1, \ldots, x_n)$ được biểu diễn thành một số mờ A'_i

$$\mu_{A'_i}(u_i) = e^{-\left(\frac{u_i - x_i}{a_i}\right)} \tag{11}$$

trong đó, $u_i \in U_i$ và a_i là tham số dương. Số mờ A'_i được cho trong hình 11.16a. tập mờ A' được xác định là tích Đề-các của các tập mờ A'_i (k = 1, …,n)

$$A' = A'_1 \times \ldots \times A'_n$$

3. *Mờ hoá tam giác*



Hình 11.16. **Các dạng số mờ A'_i.**

Mỗi giá trị số x_1(I =1,…,n) được biến đổi thành một số mờ hình tam giác A'_i.

$$\mu_{A'_i}(u_i) = \begin{cases} 1 - \dfrac{|u_i - x_i|}{b_i} & \text{if} \quad |u_i - x_i| \le b_i \\ 0 & \text{if} \quad |u_i - x_i| > b_i \end{cases} \qquad (12)$$

trong đó, $u_1 \in U_1$ và b_1 là tham số dương. Số A'_1 được cho trong hình 11.16b. sau đó chúng ta cũng lấy A'là tích đề các mờ của các tập mờ a'_1 (I = 1,…,n). trong tích đề các mờ, chúng ta có thể sử dụng T – norm là phép lấy min hoặc tích đại số.

11.5.5. Khử mờ

Khử mờ (defuzzifier) là quá trình xác định một điểm $y \in V$ từ một tập mờ B' trên V. (Tập mờ B' là đầu ra của bộ suy diễn mờ ứng với tập mờ đầu vào A') Khử mờ phải thoả mãn các tính chất sau:

- Điểm y là "đại diện tốt nhất" cho tập mờ B'. Trực quan, điều này có nghĩa là y phải là điểm có mức độ thuộc cao nhất vào tập mờ B' và y nằm ở trung tâm của tập giá đỡ của tập mờ B'.
- Hiệu quả tính toán. Tính chất này là rất quan trọng đối với các hệ điều khiển mờ, vì hành động được đưa ra bởi hệ điều khiển cần đáp ứng kịp thời các thay đổi của môi trường.
- Tính liên tục. Khi tập mờ B' thay đổi ít thì y cũng thay đổi ít.

Sau đây là một số phương pháp khử mờ quan trọng.

1. Khử mờ lấy max

Tư tưởng của phương pháp này là, chúng ta chọn y là điểm có mức độ thuộc cao nhất vào tập mờ B'.

Chúng ta xác định tập rõ H

$$H = \left\{ y \in V \mid \mu_{B'}(y) = \sup_{v \in V} \mu_{B'}(v) \right\}$$

Sau đó chúng ta có thể lấy y là :

- Một diểm bất kỳ trong H
- Điểm nhỏ nhất hoặc lớn nhất trong H
- Trung điểm của H

2. Khử mờ lấy trọng tâm

Trong phương pháp này, chúng ta lấy y là trọng tâm của tập mờ B', tức là

$$y = \frac{\int_V v\mu_{B'}(v)dv}{\int_V \mu_{B'}(v)dv} \qquad (13)$$

trong đó, $\int$ là tích phân thông thường.

11.5.6. Hệ mờ là hệ tính xấp xỉ vạn năng

Như chúng ta đã nói, các hệ mờ đã được ứng dụng thành công trong rất nhiều lĩnh vực, đặc biệt trong điều khiển các quá trình công nghiệp. Một vấn đề đặt ra là, tại sao hệ mờ lại có phạm vi ứng dụng rộng lớn và hiệu quả như thế? một cách trực quan, chúng ta có thể giải thích rằng, đó là vì các hệ mờ có thể sử dụng tri thức của các chuyên gia được phát biểu trong ngôn ngữ tự nhiên. thứ hai là vì, cấc số liệu thu nhận được từ môi trường bằng quan sát mang tính xấp xỉ, không chính xác; mà các hệ mờ cho phép biểu diễn và sử lý một cách hợp lý các dữ liệu đó.

Bây giờ chúng ta trả lời cho vấn đề đặt ra bằng một kết quả lý thuyết quan trọng.

Chúng ta xét khả năng của hệ mờ dưới quan điểm xấp xỉ hàm. một hệ mờ n đầu vào - một đầu ra (hình 11.15) xác định cho chúng ta một hàm thực của n biến $y = F(x)$, ứng với mỗi vectơ đầu vào $x = (x_1, \ldots, x_n) \in R^n$ với giá trị đầu ra $y \in R$.

Kết quả quan trọng sau đây đã được chứng minh.

Giả sử f là hàm, $f: U \to V$, $U \subset R^n$, $V \subset R$. Nếu f là hàm liên tục trên tập compact U của không gian R^n thì sẽ tồn tại một hệ mờ sao cho hàm F được xác định bởi nó là xấp xỉ của hàm f với độ chính xác tuỳ ý, tức là

$$\sup_{x \in U} |F(x) - f(x)| < \varepsilon$$

trong đó, ε là số dương nhỏ tuỳ ý cho trước.

Kết quả trên đã được chứng minh dựa trên định lý Stone – Weierstrass nổi tiếng trong giải tích. Bạn đọc có thể tham khảo cách chứng minh kết quả trên trong [15] hoặc trong [24].

TÀI LIỆU THAM KHẢO

1. Adeli H., Hung S. L. Machine Learning: Neural Network, Ginetic Algorithms and Fuzzy Systems. John Wiley & Sons, 1995.
2. Allen J. F. Natural Langrage Understanding. Benjamin/ Cummings, 1994.
3. Back T., Hammel U. and Schwefel H. P. Evolutionary Computation: Comments of the History and Current State. IEEE Transaction on Evolutionary Computation, vol 1, N1, April, 1997.
4. Bratko I. Prolog Programming for Artifical Intelligence. Addison – Wesley, 1990.
5. Charniak E., McDermott D. V. Introduction to Artifical Intelligence. Addison – Wesley, 1985.
6. Chen C. H. Fuzzy Logic and Neural Network Hanbook. Mc Graw – Hill, 1996.
7. Dean T., Allen J. and Aloimonos Y. Artifical Intelligence: Theory and Practice. Benjamin/ Cummings, 1995.
8. Đinh Mạnh Tường. Cấu Trúc Dữ Liệu và Thuật Toán. Nhà xuất bản KH&KT, 2000.
9. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison – Wesley, 1989.
10. Hagan M. T., Demuth H. B. and Beal M. Nearal Network Design. PWS Publishing Co., 1996.
11. Jackson P. Introduction to Expert Systems. Addison – Wesley, 1990.
12. Jurafsky D. and Martin J. Speech and Language Processing: An Introdutional to Natural Language Processing, Computational Linguistics and Speech Recognition. Prentice – Hall, 2000.
13. Klir G. J. and B. Yuan. Fuzzy Sets and Fuzzy Logic: Threory and Aplications. Prentice – Hall, 1995.
14. Le T. V. Techniques of Prolog Programming. John Wiley & Sons, 1993.

15. Lee C. S. G. and Link C. T. Neural Fuzzy Systems. A neuro – Fuzzy Synergism to Intelligent Systems. Prentice – Hall, 1996.
16. Michalewicz Z. Genetics Algorithms + Data Structures = Evolution Programs. Springer – Verlag, 1992.
17. Mitchell T. M. Machine Learning. McGraw – Hill, 1997.
18. Nilsson N. Problem – Solving Methods in Artifical Intelligence. McGraw – Hill, 1971.
19. Nilsson L. Principles of Artifical Intelligence. Morgan – Kaufmann, 1980.
20. Poole D., Mackworth A. and Goebel R. Computational Intelligence: A Logical Approach. Oxford University Press, 1998.
21. Rich E. Artifical Intelligence. McGraw – Hill, 1983.
22. Ross T. J. Fuzzy Logic with Engeneering Applications. McGraw – Hill, 1995
23. Russel S. and Norvig P. Artifical Intelligence: Modern Approach. Prentice – Hall, 1995.
24. Wang L. X. A Course in Fuzzy Systems and Cntrol. Prentice – Hall, 1997.
25. Winston P. H. Artifical Intelligence. Addison – Wesley, 1992.
26. Yang Q. Intelligent Planning: A Decomposition and Abstaction – Based Approach. Springer – Verlag, 1997.
27. Zimmermann H. J. Fuzzy Set Theory and Its Applications. Kluwer Academic Publishers, 1991.
28. W. Kreutzer, B. J. McKenzie. Programming for Artifical Intelligence: Methods, Tools and Applications. Addison – Wesley, 1989.