TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN
-----***-----

BM.02-QT.PDT.01
26/11/13-REV: 0

BÀI GIẢNG

TÊN HỌC PHẦN : **TRÍ TUỆ NHÂN TẠO VÀ HỆ CHUYÊN GIA**

MÃ HỌC PHẦN : **17220**

TRÌNH ĐỘ ĐÀO TẠO : **ĐẠI HỌC CHÍNH QUY**

Hải Phòng, ngày ……	*Hải Phòng, ngày ……*	*Hải Phòng, ngày ……*
TRƯỞNG KHOA	TRƯỞNG BỘ MÔN	NGƯỜI BIÊN SOẠN
TS. LÊ QUỐC ĐỊNH	**TS. NG DUY TRƯỜNG GIANG**	**Th.S NG. HẠNH PHÚC**

HẢI PHÒNG – 2015

MỤC LỤC

Đề cương chi tiết các học phần:

Tên học phần: Trí tuệ nhân tạo và Hệ chuyên gia **Mã HP: 17220**

a. Số tín chỉ: 4 TC **BTL** [x] **ĐAMH** [ ]

b. Đơn vị giảng dạy: Bộ môn Khoa học máy tính.

c. Phân bổ thời gian:

- Tổng số (TS): 60 tiết.
- Lý thuyết (LT): 43 tiết.
- Thực hành (TH): 0 tiết.
- Bài tập (BT): 0 tiết.
- Hướng dẫn BTL/ĐAMH (HD): 15 tiết.
- Kiểm tra (KT): 2 tiết.

d. Điều kiện đăng ký học phần:

Học phần này được bố trí sau các học phần: Kỹ thuật lập trình C; Cấu trúc dữ liệu và giải thuật.

e. Mục đích, yêu cầu của học phần::

Kiến thức:

- Nắm bắt các kiến thức cơ bản, các thuật toán tìm kiếm, các phương pháp biểu diễn tri thức trong trí tuệ nhân tạo cùng các vấn đề liên quan.

- Tìm hiểu cấu trúc, nguyên lý, phương pháp biểu diễn của máy học và các hệ chuyên gia.

Kỹ năng:

- Thành thạo trong việc cài đặt các thuật toán tìm kiếm trong trí tuệ nhân tạo.

- Nắm vững các phương pháp biểu diễn cơ sở dữ liệu tri thức

- Nắm vững nguyên lý hoạt động của máy học, các hệ chuyên gia và ứng dụng trong thực tế

Thái độ nghề nghiệp:

- Có thái độ ứng xử đúng trong vận hành, khai tác hiệu quả các hệ chuyên gia và các chương trình trí tuệ nhân tạo.

- Hình thành nhận thức về phân tích, giải quyết các bài toán xây dựng hệ chuyên gia trong thực tế.

f. Mô tả nội dung học phần:

Học phần này trình bày các nội dung sau: Phương pháp biểu diễn và giải quyết vấn đề; Các chiến lược tìm kiếm kinh nghiệm, các chiến lược tìm kiếm tói ưu và tìm kiếm có đối thủ, các kiến thức về logic vị từ cấp 1; Các kỹ thuật biểu diễn, xử lý tri thức và suy diễn; Tổng quan về hệ chuyên gia; Các vấn đề về hỗ trợ quyết định; Biểu diễn tri thức và lập luận bằng logic mờ và lập luận xấp xỉ; Máy học.

g. Người biên soạn: **Nguyễn Hạnh Phúc – Bộ môn Khoa học máy tính**

h. Nội dung chi tiết học phần:

TÊN CHƯƠNG MỤC	**PHÂN PHỐI SỐ TIẾT**					
	TS	LT	BT	TH	HD	KT
Chương 1. Các chiến lược tìm kiếm mù.	**4**	**4**	**0**	**0**	**5**	**0**
1.1 Biểu diễn vấn đề trong không gian trạng thái		*0,5*				
1.2 Tìm kiếm theo bề rộng		*1*				
1.3 Tìm kiếm theo độ sâu		*1*				
1.4 Các trạng thái lặp		*0,5*				
1.5 Tìm kiếm sâu lặp		*0,5*				

1.6 Quy vấn đề về vấn đề con. Tìm kiếm trên đồ thị và/hoặc		*0,5*				
Tự học: - Sinh viên tự cài đặt các thuật toán tìm kiếm theo chiều rộng, chiều sâu.						
Chương 2. Các chiến lược tìm kiếm kinh nghiệm.	**4**	**4**	**0**	**0**	**0**	**0**
2.1 Hàm đánh giá và tìm kiếm kinh nghiệm		*1*				
2.2 Tìm kiếm tốt nhất đầu tiên		*1*				
2.3 Tìm kiếm leo đồi		*1*				
2.4 Tìm kiếm beam		*1*				
Tự học (5 t): Sinh viên tự cài đặt các thuật toán tìm kiếm theo chiều rộng, chiều sâu.						
Chương 3. Các chiến lược tìm kiếm tối ưu và tìm kiếm có đối thủ	**5**	**4**	**0**	**0**	**10**	**1**
3.1 Tìm đường đi ngắn nhất		*1*				
3.2 Tìm đối tượng tốt nhất		*1*				
3.3 Cây trò chơi và tìm kiếm trên cây trò chơi		*0,5*				
3.4 Chiến lược Minimax		*1*				
3.5 Phương pháp cắt cụt alpha – beta		*0,5*				
Tự học (5 t): Sinh viên tự cài đặt các thuật toán tìm kiếm tối ưu, các thuật toán tìm kiếm theo kinh nghiệm.						
Chương 4. Logic vị từ cấp I.	**9**	**9**	**0**	**0**	**0**	**0**
4.1 Cú pháp và ngữ nghĩa của logic vị từ cấp I		*1*				
4.2 Chuẩn hóa các công thức		*1*				
4.3 Các luật suy diễn		*2*				
4.4 Thuật toán hợp nhất		*1*				
4.5 Chứng minh bằng luật phân giải		*2*				
4.6 Sử dụng logic vị từ cấp I để biểu diễn tri thức		*1*				
4.7 Xây dựng cơ sở tri thức		*1*				
Tự học (5 t): Sinh viên biểu diễn các bài tập được giao dưới dạng logic vị từ cấp 1, thực hiện các thuật toán suy diễn, hợp nhất.						
Chương 5. Hệ chuyên gia, chương trình ứng dụng	**2**	**2**	**0**	**0**	**0**	**0**
5.1 Hệ chuyên gia - chương trình ứng dụng		*1*				
5.2 Cấu trúc hệ chuyên gia		*0,5*				
5.3 Ứng dụng hệ chuyên gia		*0,5*				
Tự học (5 t): Sinh viên tìm hiểu các hệ chuyên gia đang được phát triển và các ứng dụng của chúng.						
Chương 6. Biểu diễn tri thức	**6**	**6**	**0**	**0**	**0**	**0**
6.1 Mở đầu		*0,5*				
6.2 Dư thừa (Redundancy)		*1*				

6.3 Mâu thuẫn (consistency - inconsistency)		*1,5*				
6.4 Lưu trữ CSTT		*1*				
6.5 Soạn thảo tri thức		*1*				
6.6 Cập nhật sửa đổi		*1*				
Tự học (5 t): Sinh viên tìm hiểu các phương pháp biểu diễn tri thức, áp dụng trong việc biểu diễn tri thức của các bài toán được giao.						
Chương 7. Các kỹ thuật suy diễn và lập luận	**6**	**5**	**0**	**0**	**0**	**1**
7.1 Nhập môn		*0,5*				
7.2 Phân rã CSTT		*1*				
7.3 Mô tơ suy diễn		*1,5*				
7.4 Biểu diễn tri thức bằng Logic vị từ và suy diễn		*1*				
7.5 Ứng dụng các kỹ thuật suy diễn		*1*				
Tự học (5 t): Sinh viên tìm hiểu các kỹ thuật suy diễn và lập luận, ứng dụng trong biểu diễn một số cơ sở tri thức.						
Chương 8. Hệ hỗ trợ quyết định	**3**	**3**	**0**	**0**	**0**	**0**
8.1 Khái niệm về hệ hỗ trợ ra quyết định		*1*				
8.2 Cấu trúc của một hệ hỗ trợ ra quyết định		*2*				
Tự học (5 t): Sinh viên tìm hiểu các hệ gia quyết định và các ứng dụng của chúng.						
Chương 9. Máy học	**3**	**3**	**0**	**0**	**0**	**0**
9.1 Khái niệm máy học		*1*				
9.2 Học bằng cách xây dựng cây định danh		*2*				
Tự học (5 t): Sinh viên tìm hiểu về máy học và các phương pháp học máy						
Chương 10. Logic mờ và lập luận xấp xỉ	**2**	**2**	**0**	**0**	**0**	**0**
10.1 Khái niệm		*0.5*				
10.2 Các phép toán mờ		*0.5*				
10.3 Biểu diễn tri thức bằng logic mờ		*1*				
Tự học (5 t): Sinh viên tìm hiểu về logic mờ và hệ chuyên gia logic mờ, các ứng dụng.						
Ôn tập	1	1				
Tổng số tiết	**60**	**43**	**0**	**0**	**15**	**2**

i. Mô tả cách đánh giá học phần:

- Sinh viên phải tham dự tối thiểu 75% số giờ lên lớp và phải đạt các điểm thành phần X1, X2, X3 từ 4,0 trở lên (X1 là điểm đánh giá ý thức, thái độ tham gia học tập, X2 là điểm trung bình của ít nhất 02 bài kiểm tra trên lớp trong đó có điểm kiểm tra việc ghi nhớ kiến thức, tự học và điểm vận dụng kiến thức, X3 là điểm đánh giá bài tập lớn).

- Điểm học phần (Z) được tính theo công thức: Z = 0.5X + 0.5Y.

Trong đó:

- X: điểm quá trình, X=0.3X1+0.3 X2+0.4X3.
- Y: điểm bài kiểm tra kết thúc học phần

Thang điểm đánh giá: A^+, A, B^+, B, C^+, C, D^+, D và F

k. Giáo trình:

1. Đinh Mạnh Tường, *Trí tuệ nhân tạo*, Nhà xuất bản Khoa học kỹ thuật, 2002
2. Nguyễn Thanh Thủy, *Hệ chuyên gia*, Trường Đại học Bách khoa Hà Nội, 2002

l. Tài liệu tham khảo:

1. Nguyễn Thanh Thủy, *Trí tuệ nhân tạo*, Nhà xuất bản Giáo dục, 1997
2. Ngô Trung Việt, *Trí tuệ nhân tạo*, Nhà xuất bản Giáo dục, 1995
3. Hoàng Kiếm, *Các hệ cơ sở tri thức*, Nhà xuất bản ĐHQG TPHCM, 2002

m. Ngày phê duyệt: 30/06/2014

n. Cấp phê duyệt: Khoa CNTT

Trưởng Khoa	**P. Trưởng Bộ môn**	**Người biên soạn**
TS Lê Quốc Định	***ThS. Nguyễn Văn Thủy***	***ThS. Nguyễn Hạnh Phúc***

o. Tiến trình cập nhật Đề cương:

Cập nhật lần 1: *ngày 25/06/2014* **Nội dung**: Rà soát theo kế hoạch Nhà trường gồm: - Chỉnh sửa, làm rõ các Mục **e, i** theo các mục tiêu đổi mới căn bản. - Mục **h**: bổ sung Nội dung tự học cuối mỗi chương mục, chuyển một số nội dung giảng dạy sang phần tự học. - Bổ sung các mục **m, n, o**	Người cập nhật *Nguyễn Hạnh Phúc* P. Trưởng Bộ môn *Nguyễn Văn Thủy*
Cập nhật lần 2: *ngày 11 / 05 /2015* **Nội dung:** Rà soát theo kế hoạch Chỉnh sửa lại mục i theo tiêu chí mới về cách đánh giá kết quả học tập.	Người cập nhật *Ng. Duy Trường Giang* Trưởng Bộ môn *Ng. Duy Trường Giang*

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 được 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 mộ t 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 I. Các chiến lược tìm kiếm mù

Trong chương này, chúng tôi 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 các 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 nối 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.

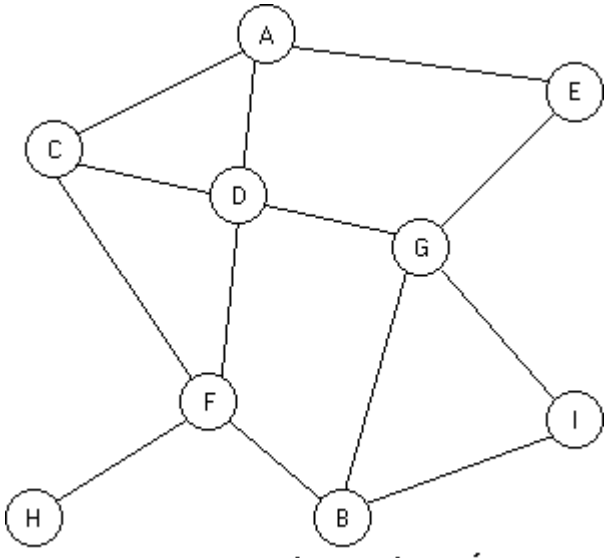
Tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu bằng cách áp dụng một dãy toán tử, lập thành không gian trạng thái của vấn đề.

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: $U \to U$. 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. N hưng

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).



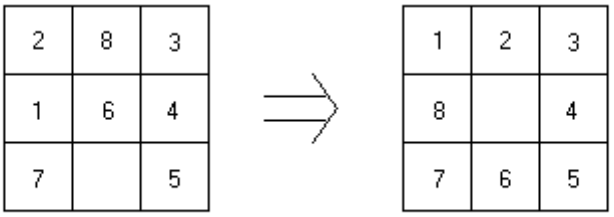
Hình 1.1 Tìm đường đi từ A đến B

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.

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

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 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***



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

(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 hiểu được biểu diễn thích hợp cho các trạng thái của vấn đề là hoàn toàn không đơn giản. Việc tìm ra dạng biể u 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. N hưng 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).

Các chiến lược tìm kiếm

Như ta đã thấy, 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 cảu 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).

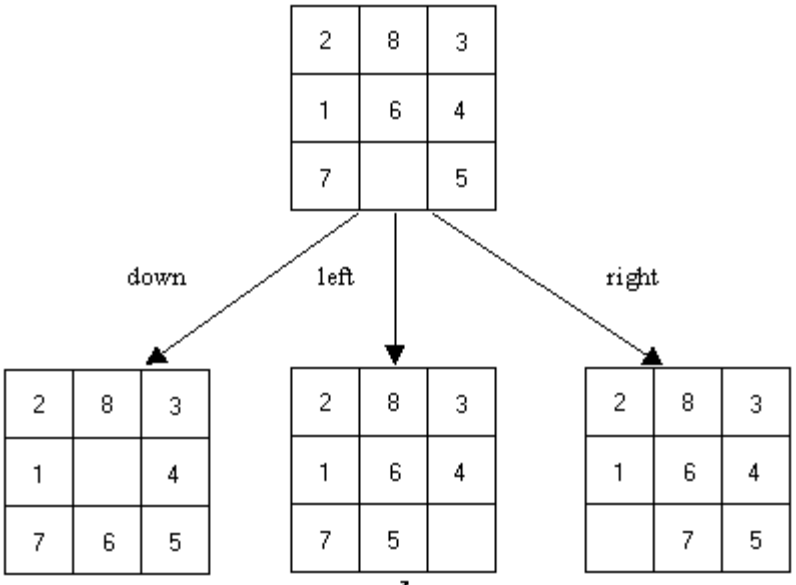
Giả sử u là một trạng thái nào đó và R là một toán tử biến đổi u thành v. Ta sẽ gọi v là trạng thái kề u, hoặc v được sinh ra từ trạng thái u bởi toán tử R. Quá trình áp dụng các toán tử để sinh ra các trạng thái kề u được gọi là phát triển trạng thái u. Chẳng hạn, trong bài toán toán số, phát triển trạng thái ban đầu (hình 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.



Hình 1.3 Phát triển một trạng thái

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.

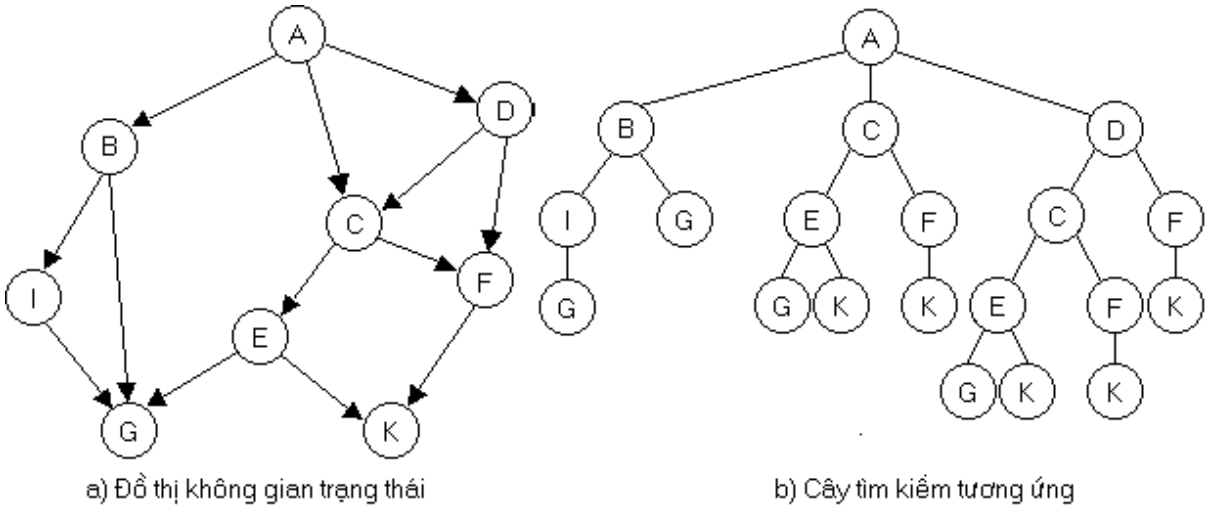
Tìm kiếm kinh nghiệm (tìm kiếm heuristic). Trong rất nhiều vấn đề, chúng ta có thể dựa vào sự hiểu biết của chúng ta về vấn đề, dựa vào kinh nghiệm, trực giác, để đánh giá các trạng thái. Sử dụng sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm: trong quá trình phát triển các trạng thái, ta sẽ chọn 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 đúng 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. K ỹ 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).

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.

1.2 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. Lặp lại các việc sau

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.

1.3 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".

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 t ìm kiếm chứa các nhánh vô hạn.

1.4 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 1.4b, 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 lập 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 Qbởi bảng băm.

1.5 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, … đến 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(u_0)←*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 mã 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:

$1 + b + b^2 + \ldots + b^d$

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:

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

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(biểu diễn) (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.

1.6 Quy vấn đề về các vấn đề con. Tìm kiếm trên đồ thị và/hoặc.

1.6.1 Quy vấn đề về các vấn đề con:

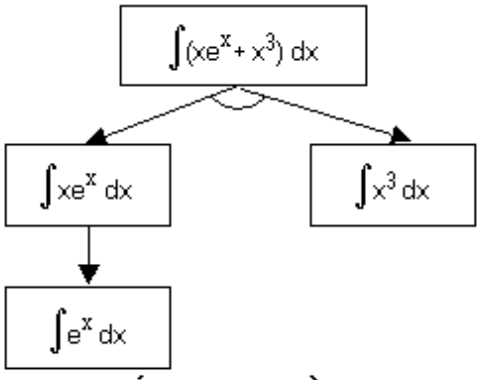
Trong mục 1.1, chúng ta đã nghiên cứu việc biểu diễn vấn đề thông qua các trạng thái và các toán tử. Khi đó việc tìm nghiệm của vấn đề được quy về việc tìm đường trong không gian trạng thái. Trong mục này chúng ta sẽ nghiên cứu một phương pháp luận khác để giải quyết vấn đề, dựa trên việc quy vấn đề 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.

1.6.2 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.

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.



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

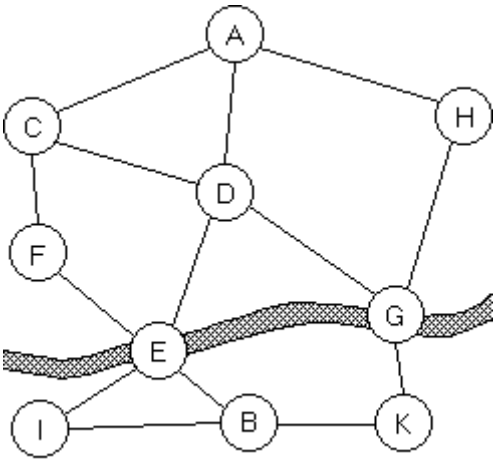
Chúng ta có thể biểu diễn việc quy một vấn đề về các vấn đề con cơ bởi các trạng thái và các toán tử. ở đây, bài toán cần giải là trạng thái ban đầu. Mỗi cách quy bài toán về các bài toán con được biểu diễn bởi một toán tử, toán tử $A \rightarrow B, C$ biểu diễn việc quy bài toán A về hai bài toán B 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 vấ n 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



Hình 1.6

Bài toán này đã được phát triển 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, 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.

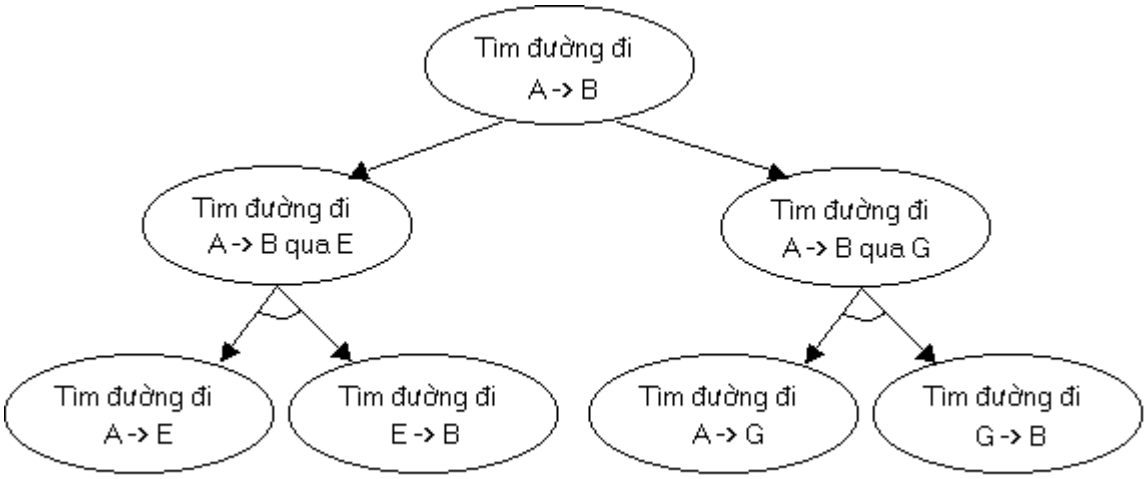
2) Bài toán tìm đường đi từ A đến B qua G được quy về:

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.

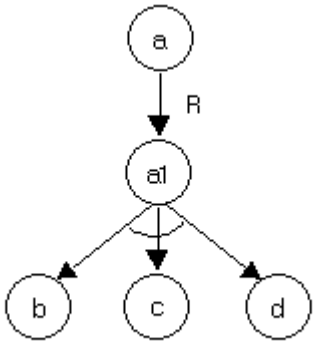
1.1.1 Đồ thị và/hoặc



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

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.
- Tập các toán tử bao 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, g

R_7 : k → e, l

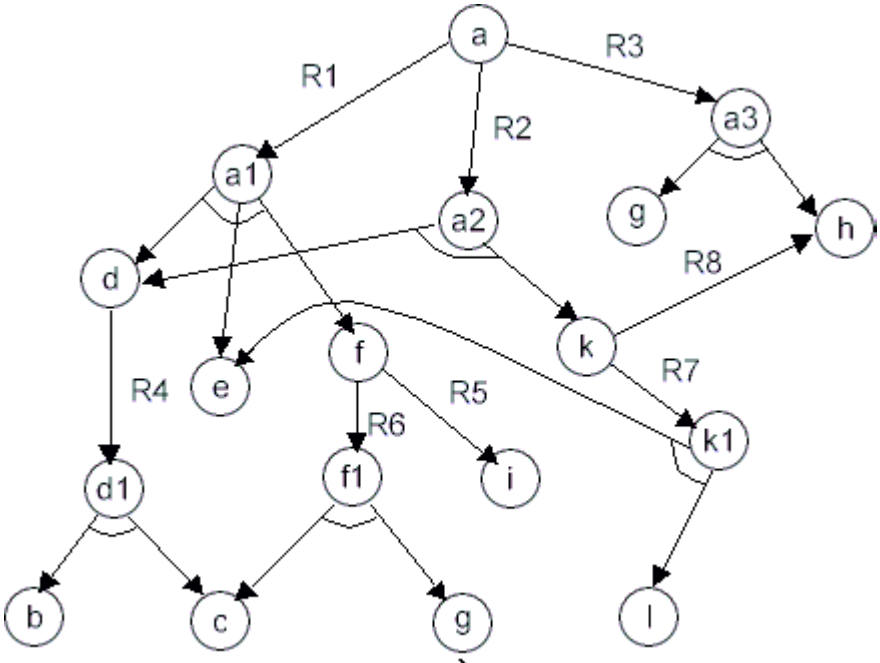
$R_8 : k \rightarrow h$

– 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 a1 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 hạnd, 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.

T = {b, c, e, j, l}.



Hình 1.9 Một đồ thị và/hoặc

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:

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 các đỉnh kề u theo một toán tử nào đó.

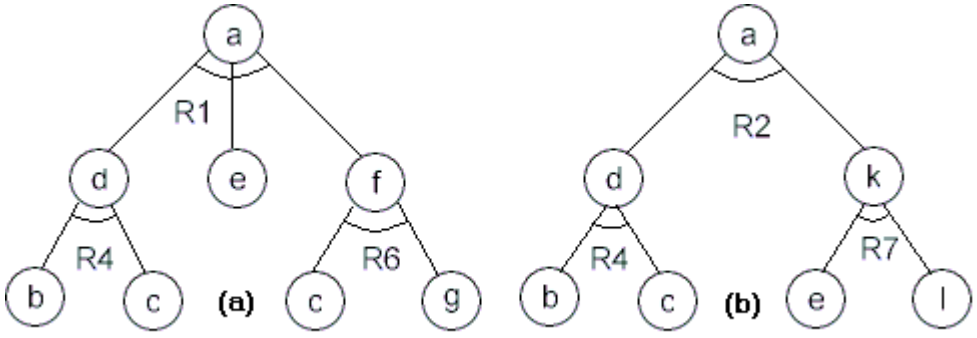
Các đỉnh của đồ thị và/hoặc sẽ được gắn nhãn giải được hoặc không giải được. Các

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.



Hình 1.11 Các cây nghiệm

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 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.

1.4.2 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. K hi 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(u)* ← **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 II 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 hướng 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

u =

3	2	8
	6	4
7	1	5

1	2	3
8		4
7	6	5

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

Hình 2.1 Đánh giá trạng thái u.

$h_1(u) = 4$, vì các quân không 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:

$h_2(u) = 2 + 3 + 1 + 3 = 9$

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:

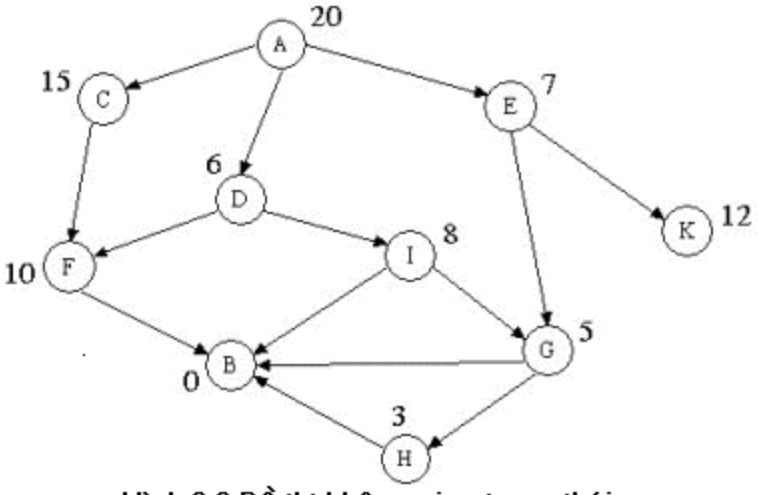
Tìm kiếm tốt nhất đầu tiên = Tìm kiếm theo bề rộng + Hàm đánh giá

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.

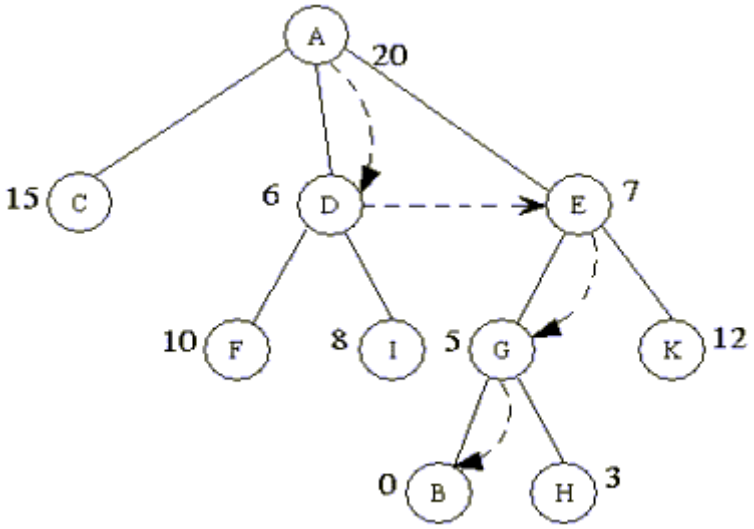
2.2 Tìm kiếm tốt nhất – đầu tiên:

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.



Hình 2.2 Đồ thị không gian trạng thái

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,



Hình 2.3 Cây tìm kiếm tốt nhất - đầu tiên

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.

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 triể n, danh sách được sắp theo thứ tự tăng dần của hàm đánh giá sao cho trạng thái có giá trị 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***;**

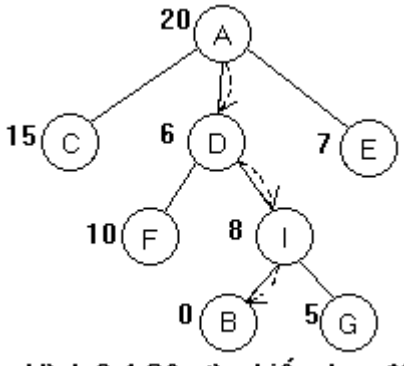
end;

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, 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.



Hình 2.4 Cây tìm kiếm leo đồi

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 L1 để lưu giữ tạm thời các trạng thái kề trạng thái u, khi ta phát triển u. Danh sách L1 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};**

2.4 **for** *mỗi trạng thái v kề u do đặt v vào L1***;**

*2.5 Sắp xếp L1 theo thứ tự tăng dần của hàm đánh giá***;**

*2.6 Chuyển danh sách L1 vào đầu danh sách L***;**

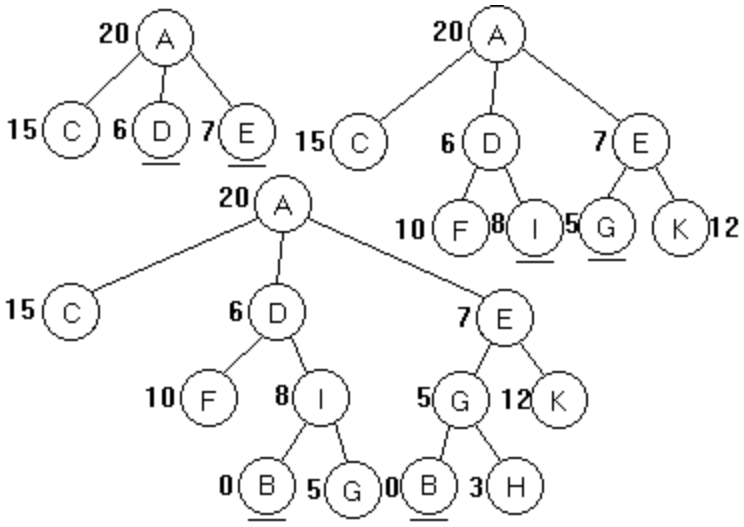
end;

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).

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.



Hình 2.5 Cây tìm kiếm beam.

Chương III. Các chiến lược tìm kiếm tối ưu và tìm kiếm có đối thủ

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;
- Các kỹ thuật tìm kiếm có đối thủ.

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) ≥ 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 hạ n, 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). (Độ dài thực tế)
- 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 thô ng, 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.

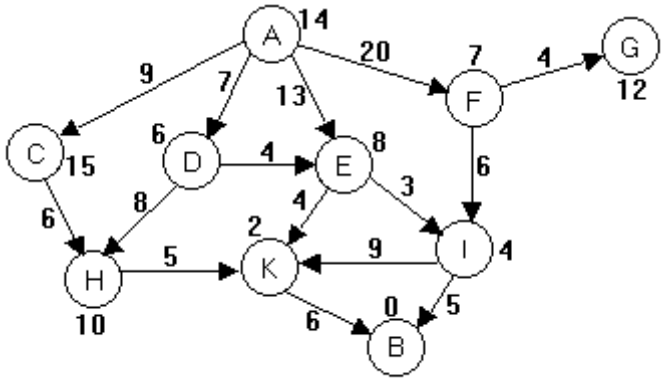
Ta cũng có thể sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(u). Phương pháp này sẽ tìm ra đường đi ngắn nhất, tuy nhiên nó có thể kém hiệu quả.

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.

Thuật toán A*



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

Bo:

DSL: A, cost=99

B1: U=A không trùng đích

V: $C^{24}{}_{A}$, $D^{13}{}_{A}$, $E^{21}{}_{A}$, $F^{27}{}_{A}$

L1: $D^{13}{}_{A}$, $E^{21}{}_{A}$, $C^{24}{}_{A}$, $F^{27}{}_{A}$

L: $D^{13}{}_{A}$, $E^{21}{}_{A}$, $C^{24}{}_{A}$, $F^{27}{}_{A}$

B2: U=$D^{13}{}_{A}$ không trùng đích

L1: $H^{25}{}_{DA}$, $E^{19}{}_{DA}$

L1: $E^{19}{}_{DA}$, $H^{25}{}_{DA}$

L: $E^{19}{}_{DA}$, $H^{25}{}_{DA}$, $E^{21}{}_{A}$, $C^{24}{}_{A}$, $F^{27}{}_{A}$

B3: U= $E^{19}{}_{DA}$ không trùng đích

L1: $K^{17}{}_{EDA}$, $I^{18}{}_{EDA}$

L1: $K^{17}{}_{EDA}$, $I^{18}{}_{EDA}$

L: $K^{17}{}_{EDA}$, $I^{18}{}_{EDA}$, $H^{25}{}_{DA}$, $E^{21}{}_{A}$, $C^{24}{}_{A}$, $F^{27}{}_{A}$

B4: U= $K^{17}{}_{EDA}$ không trùng đích

L1: $B^{21}{}_{KEDA}$

L1: B^{21}_{KEDA}

L: B^{21}_{KEDA}, I^{18}_{EDA}, H^{25}_{DA}, E^{21}_{A}, C^{24}_{A}, F^{27}_{A}

B5: U= B^{21}_{KEDA} trùng đích

Do (g(B)=21)<(cost=99) nên: cost=21 (đường đi tốt nhất tạm thời có độ dài =21)

B6: U= I^{18}_{EDA} không trùng đích

L1: K^{25}_{IEDA}, B^{19}_{IEDA}

L1: B^{19}_{IEDA}, K^{25}_{IEDA}

L: B^{19}_{IEDA}, K^{25}_{IEDA}, H^{25}_{DA}, E^{21}_{A}, C^{24}_{A}, F^{27}_{A}

B7: U= B^{19}_{IEDA} trùng đích

Do (g(B)=19)<(cost=21) nên: cost=19 (đường đi tốt nhất tạm thời có độ dài =19)

B8: U=K^{25}_{IEDA} không trùng đích

Do f(K)>cost => Chuyển B9

B9: U=H^{25}_{DA} không trùng đích

Do f(H)>cost => Chuyển B10

B10: U=E^{21}_{A} không trùng đích

Do f(E)>cost => Chuyển B11

B11: U=C^{24}_{A} không trùng đích

Do f(C)>cost => Chuyển B12

B12: U=F^{27}_{A} không trùng đích

Do f(F)>cost => Chuyển B13

B13: L= rỗng => kết thúc

Đường đi tốt nhất là: A->D->E->I->B

Thuật toán A* là thuật toán sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh

giá f(u). Trong đó, trạng thái ban đầu là trạng thái A, trạng thái đích là B, các số ghi cạnh các cung là độ dài đường đi, 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,

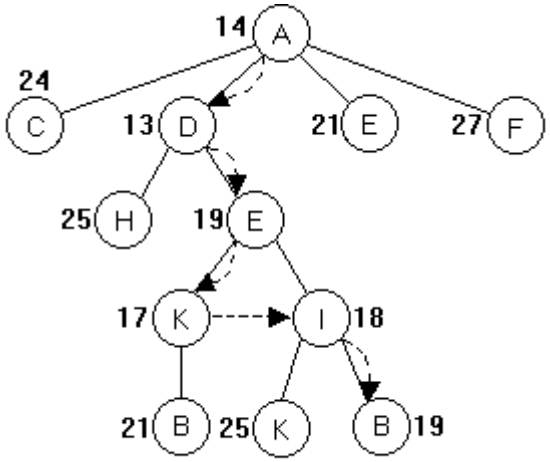
g(E) = 13, f(E) = 13 + 8 = 21, g(F) = 20, f(F) = 20 +7 = 27

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.



Hình 3.2 Cây tìm kiếm theo thuật toán A*

Vậy đỉnh E mới có đánh giá là f(E) = g(E) + h(E) = 11 + 8 = 19. Tro ng số các đỉnh cho 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 đ i ngắn nhất tới B là g(B) = 19. Q uá 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).

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;

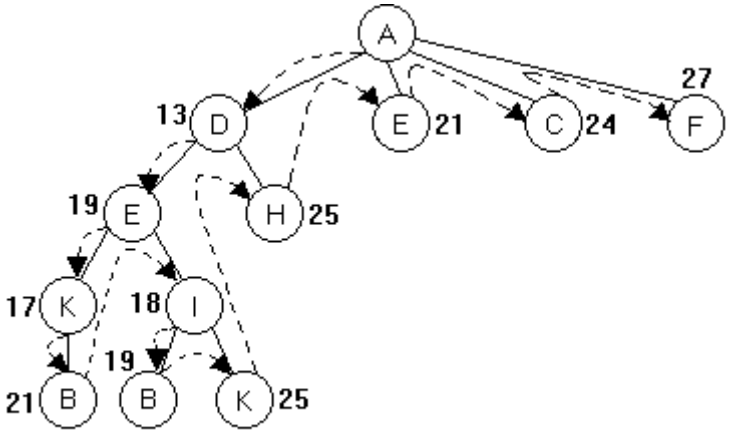
Chúng ta đưa ra một số nhận xét về thuật toán A*: Người ta chứng minh được rằng, nếu hàm đánh giá h(u) là đánh giá thấp nhất (trường hợp đặc biệt, h(u) = 0 với mọi trạng thái u) thì thuật toán A* là thuật toán ***tối ưu,*** tức là nghiệm mà nó tìm ra là nghiệm tối ưu. Ngoài ra, nếu độ dài của các cung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

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 đề tiếp tục đi xuống trạng thái tốt nhất trong các trạng thái còn lại chưa được phát triển.

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.



Hình 3.4 Cây tìm kiếm nhánh_và_cận.

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 $g(u) \leq$ 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 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 = (x1, x2, ... xn) 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 là hàm khả vi liên tục. Cuối cùng ta sẽ nghiên cứu kỹ thuật tìm kiếm mô phỏng luyện kim(simulated annealing).

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 k hác 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 trong đối tượng tìm được.

procedure *Hill_Climbing*;

begin

u ← một đối tượng ban đầu nào đó;

if *cost(v) > cost(u)* **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 rấ t 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.

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 = (x1,...,xn), thì hàm 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. $\nabla f = \left(\frac{\partial f}{\partial x1}, \frac{\partial f}{\partial x2}, ..., \frac{\partial f}{\partial xn} \right)$

procedure *Gradient_Search*;

begin

x ← *điểm xuất phát nào đó*;

repeat

$x \leftarrow x + \alpha \Delta f(x)$;

until $|\Delta f| < \varepsilon$;

end;

Trong thủ tục trên, α là hằng số dương nhỏ nhất 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 Δf = 0, hoặc tìm được điểm rất gần với cực đại địa phương.

3.3 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 Trắ ng, 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.

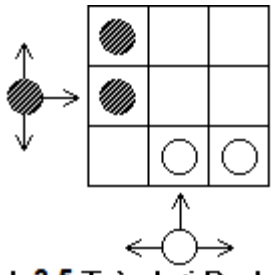
- Một hàm kết cuộc (payoff function) ứng mỗi trạng thái kết thúc với một giá trị nào đó. Chẳng hạn như cờ vua, mỗi trạng thái kết thúc chỉ có thể là thắng, hoặc thua (đối với Trắng) hoặc hòa. Do đó, ta có thễ xác định hàm kết cuộc là hàm nhận giá trị 1 tại các trạng thái kết thúc là thắng (đối với Trắng), -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 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 hặc đều là Đen, các lá của cây ứng với các trnạg thái kết thúc.

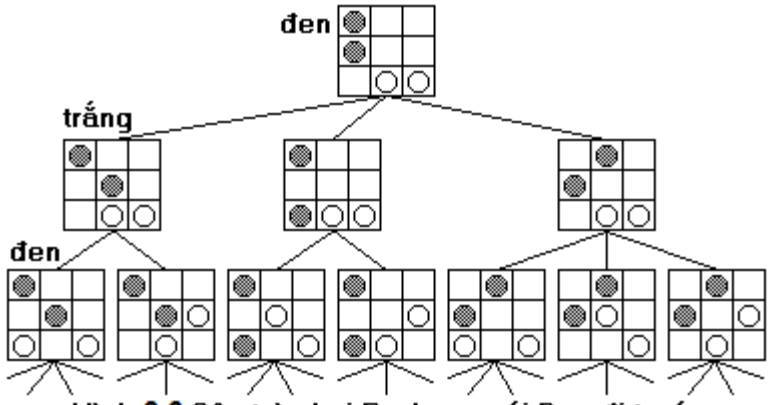
Ví dụ: Xét trò chơi **Dodgem** (được tạo ra bởi Colin Vout). Có hai quân Trắng và hai quân Đen, ban đầu được xếp vào bàn cờ 3*3 (Hình vẽ). Quân Đen có thể đi tới ô trống ở bên phải, ở trên hoặc ở dưới. Q uân Trắng có thể đi tới trống ở bên trái, bên phải, ở trên. Q uân Đen nếu ở cột ngoài cùng bên phải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



Hình 3.5 Trò chơi Dodgem.

Giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như trong hình 3.5.

3.5 Chiến lược Minimax

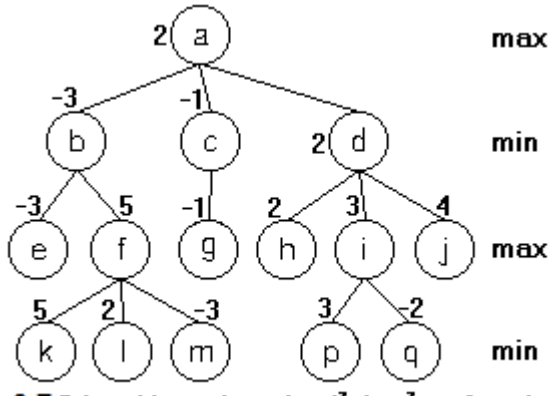


Hình 3.6 Cây trò chơi Dodgem với Đen đi trước.

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. Q uá 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. K hi 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 3.7, 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 3.7 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 đã 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 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. 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:

s1w1 +s2w2+. . . +snwn.

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 3.8. Còn mỗi quân Đen ở một vị trí sẽ được cho một giá trị tương ứng trong bảng bên phải hình 3.8:

30	35	40
15	20	25
0	5	10

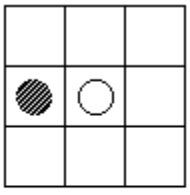
Giá trị quân Trắng.

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

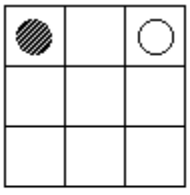
Giá trị quân Đen.

Hình 3.8 Đánh giá các quân trong trò chơi Dodgem.

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 3.9). Tương tự, nếu quân Đen cản trực tiếp quân Trắng nó được thêm -40 điểm, còn cản gián tiếp nó được thêm -30 điểm.



Trắng cản trực tiếp Đen được thêm 40 điểm.



Trắng cản gián tiếp Đen được thêm 30 điểm.

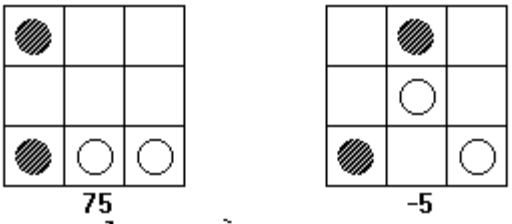
Hình 3.9 Đánh giá sự tương quan giữa quân Trắng và Đen.

Ap 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.

3.5 Phương pháp cắt cụt alpha - beta



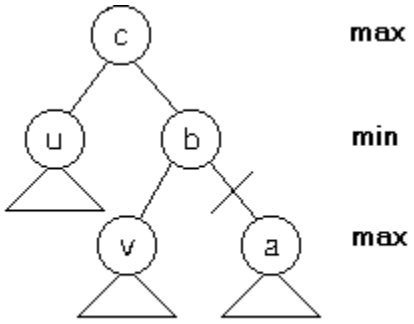
Hình 3.10 Giá trị của một sổ trạng thái trong trò chơi Dodgem.

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, một 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 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 3.11). 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).



Hình 3.11 Cắt bỏ cây con gốc a, nếu eval(u) > eval(v).

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**

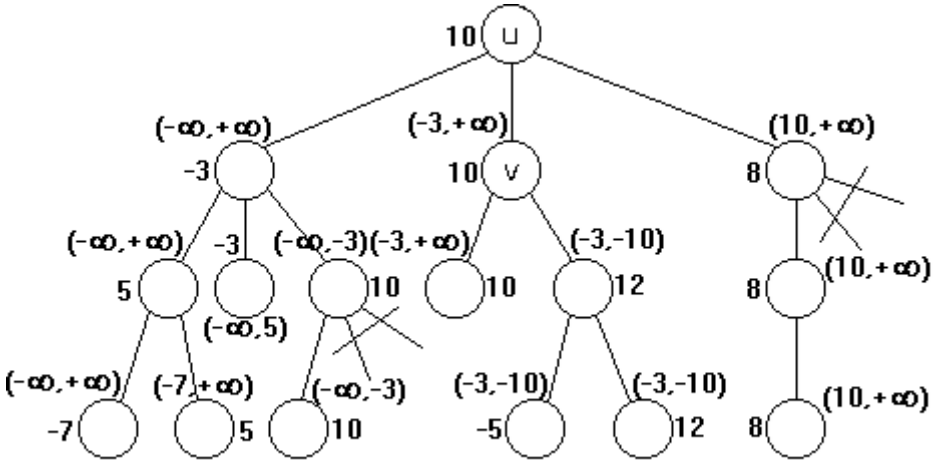
if $\alpha \le MinVal(w, \alpha, \beta)$ **then**

{$\alpha \leftarrow MinVal(w, \alpha, \beta)$**;**

$v \leftarrow w$**;}**

end;

Ví dụ. Xét cây trò chơi gốc u (đỉnh Trắng) giới hạn bởi độ cao h = 3 (hình 3.12). 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 t rị 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 3.12 Xác định giá trị các đỉnh bằng kỹ thuật cắt cụt.

Chương IV : Logic vị từ cấp 1

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 một đối tượng trong một miền đối tượng nào đó. Để mô tả các thuộc tính của đối tượng, các quan hệ giữa các đối tượng, trong logic vị từ, người ta dựa vào các *vị từ* (predicate). Ngoài các kết nối logic như trong logic mệnh đề, logic vị từ cấp một còn sử dụng các *lượng tử*. Chẳng hạn, lượng tử ∀ (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.

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

Cú pháp.

Các ký hiệu .

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

- Các ký hiệu **hằng**: a, b, c, An, Ba, John,...
- Các ký hiệu **biến**: x, y, z, u, v, w,...
- Các ký hiệu **vị từ**: P, Q, R, S, Like, Havecolor, Prime... Mỗi vị từ là vị từ của n biến (n≥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 hà m 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, A_n là ký hiệu hằng, mother là ký hiệu hàm một biến, thì mother (A_n) là một hạng thức cụ thể.

Các công thức phân tử

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

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

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

- Nếu $t_1, t_2,...,t_n$ là n hạng thức và p là vị từ của n biến thì $p(t_1,t_2,...,t_n)$ là câu đơn.

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

Các công thức

Từ công thức 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

Lượng tử phổ dụng ($\forall$) 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 ($\exists$) 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.

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

Ngữ nghĩa của các câu phức hợp .

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

Ví dụ: Câu Student(Lan) $\wedge$ 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 $\exists$x Yourger(x, 20) là ngữ nghĩa của câu Yourger(Lan, 20) $\vee$ Yourger(An, 20) $\vee$ Yourger(Hoa, 20). Câu này nhận giá trị True nếu và chỉ nếu ít nhất một trong ba người Lan, An, Hoa trẻ hơn 20.

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

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

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

Các công thức tương đương

Cũng như trong logic mệnh đề, ta nói hai công thức G và H tương đương (viết là G $\equiv$ H) nếu chúng cùng đúng hoặc cùng sai trong một minh hoạ. Ngoài các tương đương đã biết trong logic mệnh đề, trong logic vị từ cấp một còn có các tương đương khác liên quan tới các lượng tử. Giả sử G là một công thức, cách viết G(x) nói rằng công thức G có chứa các xuất hiện của biến x. K hi 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)).

4.2 Chuẩn hóa 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 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:

- Loại bỏ kéo theo

Để loại bỏ các kéo theo, ta chỉ c ần thay thế công thức $P \Rightarrow Q$ bởi công thức tương đương $\neg PvQ$ thay $P \Leftrightarrow Q$ bởi $(\neg PvQ) \wedge (\neg PvQ)$.

- 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

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

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

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

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

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

- 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 y lớn hơn “. 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 lượng tử $\forall x1,...,\forall xn$. Khi đó ta có thể xem y là hàm của n biến x1,…, xn, chẳng hạn f(x1,…, xn). Sau đó ta thay các xuất hiện của y trong công thức G bởi hạng thức f(x1,…, xn) và loại bỏ các lượng 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:

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

Công thức con $\exists yP(x,y)$ nằm trong miền tác dụng của lượng tử $\forall x$, ta xem y là hàm của x: F(x). Các công thức con $\exists b(Q(a,b)$ và $\exists y \neg R(x,y)$ nằm trong miền tác dụng của các lượng tử $\forall x$, $\forall u$ nên ta xem a là hàm g(x,u) và y là hàm h(x,u) của 2 biến x,u. Thay các xuất hiện của y và v bởi các hàm tương ứng, sau đó loại bỏ các lượng tử tồn tại, từ công thức (1) ta nhận được công thức:

$\forall x(\exists y(P(x,f(x)) \text{ v } \forall u\ (Q(a,g(x,u)) \wedge \neg R(x,h(x,u))))$ (2)

- Loại bỏ các lượng tử phổ dụng
- Chuyển các tuyển tới các Literal
- Loại bỏ các hội
- Đặt tên lại các biến

4.3 Các luật suy diễn

Luật thay thế phổ dụng:

Giả sử G là một câu, câu $\forall x$ G là đúng trong một minh hoạ nào đ ó nế u và chỉ nế u G đúng đ ối với tất cả các đối tượng nằm trong miền đối tượng c ủa minh hoạ đó. Mỗi hạng thức t ứng với một đ ối tượng vì thế nếu câu $\forall x$ G đúng thì khi thay tất cả các xuất hiệ n của biế n x bởi hạ ng thức t ta nhận được câu đúng. Công thức nhận được từ công thức G bằng cách thay tất cả các xuất hiệ n c ủa x bởi t được kí hiệu là G[x/t]. Luật thay thế phổ dụng (***universal instatiation***) phát biể u rằ ng, từ c ông thức $\forall$xG suy ra công thức G[x/t].

$\forall$xG

G[x/t]

Chẳng hạn, từ câu $\forall$x Like(x, Footba ll) (mọi người đều thích bóng đá), bằ ng cách thay x bởi An ta suy ra câu Like(An,Footba ll) (An thích bóng đá)

Hợp nhất

Trong luật tha y thế phổ dụng, ta cần sử dụng phé p thế các biế n bởi các hạng thức để nhậ n được các công thức mới từ c ông thức c hứa các lượng tử phổ dụng. Ta có thể sử dụng phép thế để hợp nhất các câu phâ n tử (tức là để các câu trở thành đồng nhất). Chẳ ng hạn xét hai câu phâ n tử Like(An, y), Like(x, Footba ll). Cần lưu ý rằng hai câu này là hai câu $\forall$y Like(An,y) và $\forall$x Like(x,Footba ll) mà để cho đơn giản ta bỏ đ i các lượng tử phổ dụng. Sử dụng phép thế [x/An, y/Footba ll] hai câu trên trở thành đồng nhất Like(An,Football). Trong các suy diễn, ta cần sử dụng phép hợp nhấ t các câu bởi các phép thế. Chẳng hạn, cho trước hai câu Friend(x,Ba) $\wedge$ 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) $\wedge$ Good(x) và Friend(Lan,y) bởi phép thay thế [x/Lan,y/Ba]. áp dụng luật thay thế phổ dụng với phé p thay thế này ta nhậ n được ha i 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 dã y các cặp xi/ti , θ = [x1/t1 x2/t2.... xn/tn] trong đó các xi là các biế n khác nhau, các ti là các hạ ng thức và các xi không c ó mặt trong ti (i=1,...,n). Áp dụng phé p thế θ vào công thức G, ta nhận được c ông thức G θ , đó là công thức nhậ n được từ công thức G bằng cách thay mỗi sự xuất hiệ n c ủa các xi bởi ti. Chẳng hạ n, nế u 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 ha i câu phâ n tử G và H mà tồn tạ i phép thế θ sao cho G θ và H θ trở thà nh đ ồng nhất (G θ =H θ) thì G và H được gọi là ***hợp nhất được***, phép thế θ được gọi là ***hợp nhất tử*** c ủa G và H. Chẳ ng hạ n, ha i câu Like(An,y)và Like(x,Footba ll) là hợp nhất được bởi hợp nhất tử [x/An,y/Footba ll]. Vấ n đề đặt ra là , với ha i câ u phân tử bất kì G và H, chúng có hợp nhất được không và nế u có thì là m thế nào tìm được hợp nhất tử ? Vấn đề này sẽ được nghiê n cứu trong mục sau. Còn bây giờ c húng ta đưa ra các luật suy diễn quan trọng nhất, trong đó c ó sử dụng phép hợp nhất.

Luật Modus Ponens tổng quát.

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

(Pi ∧⋯∧ Pn ∧ Q), Pi',..,Pn'

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 ha i câu tuyển A1 ∨⋯∨ Am ∨ C và B1 ∨⋯∨ Bn ∨ D, trong đó Ai (i =1,..,m) và Bj (j=1,..,n) là các litera l, c òn C và D là các câu phâ n tử có thể hợp nhất được bởi phé p thế θ , C θ = D θ . Khi đó ta có luật:

A1 ∨⋯∨ Am ∨ C,B1 ∨⋯∨ Bn ∨¬D (giả thiết)

A1'∨⋯∨ Am' ∨ B1'∨⋯∨ Bn' (kết luận)

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

Trong luật phâ n giả i nà y (và trong các luật phân giả i sẽ trình bày sau này), ha i câu ở tử số (giả thiết) của luật được gọi là ha i câu ***phân giải được***, còn câu ở mẫu số (kết luận) của luật được gọi là ***phân giải thức*** của ha i câu ở tử số. Ta sẽ ký hiệ u phâ n giả i thức của ha i câu A và B là Res(A,B).

Ví dụ: Giả sử ta có ha i câu A=Hear(x,Music) ∨ Play(x,Tennis) và B=¬Play(An,y) ∨ Study (An). Hai câu P la y(x,Tennis) và Pla y(An,y) hợp nhất được bởi phép thế

θ =[x|An,y|Tennis]. Do đó từ ha i câu đã cho, ta suy ra câu Hear(An,Music) $\vee$ Study (An). Trong ví dụ nà y, hai câu

A=Hear(x,Music) $\vee$ Play(x,Te nnis) và B=$\neg$Play(An,y) $\vee$ Study (An) là phân giả i được và phân giả i 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

P1^P2^⋯^Pm $\Rightarrow$ *Q*

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

Giả sử ta có hai câu Horn *P1^P2^⋯^Pm^S* $\Rightarrow$ Q và *R1^R2^⋯^Rn*$\Rightarrow$T, trong đó ha i câ u S và T hợp nhất được bởi phé p thế θ , S θ =T θ . Khi đó ta có luật:

P1 $\wedge\cdots\wedge$Pm $\wedge$S $\wedge$ Q,

R1 $\wedge\cdots\wedge$Rn $\wedge$ T

P1'$\wedge\cdots\wedge$ Pm'$\wedge$ R1'$\wedge\cdots\wedge$ Rn' $\wedge$ Q

trong đ ó Pi'=Pi θ (i=1,..,m), Rj‟‟=Rj θ (j=1,..,n), Q' =Q θ .

Trong thực tế,c húng ta thường sử dụng trường hợp riê ng sa u đây. Giả sử S và T là ha i câu phâ n tử, hợp nhất được bởi phép thế θ . Khi đó ta có luật:

P1$\wedge\cdots\wedge$ Pm $\wedge$ S $\wedge$ Q,

T

P1'$\wedge\cdots\wedge$ Pm' $\wedge$ Q'

trong đ ó Pi' = Pi θ (i = 1,...,m) và Q' = Q θ .

4.4 Thuật toán hợp nhất

Giả sử có 2 công thức E và F. Hợp nhất 2 công thức E và F thực chất là việc đọc lần lượt từ đầu đến cuối 2 công thức này và thực hiện các phép thế biến bởi hạng thức cho đến khi không còn sự khác biệt ta có các công thức E’ và F’ mới.

4.5 Chứng minh bằng luật phân giải

Tương tự thuật toán chứng minh bằng luật phân giải trong logic mệnh đề. Trong logic vị từ cấp I ta sử dụng thêm phép thế.

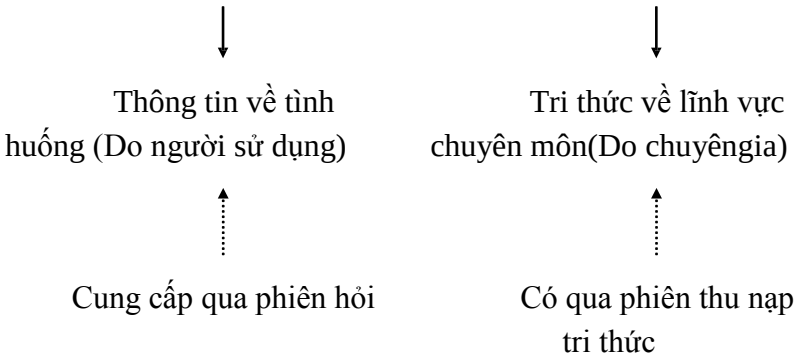
4.6 Biểu diễn tri thức bởi luật

Cơ sở tri thức bao gồm 2 bộ phận: Cơ sở luật và cơ sở sự kiện (hoặc bộ nhớ làm việc):

– Cơ sở luật bao gồm các luật có ít nhất một điều kiện, biểu diễn các tri thức chung về lĩnh vực áp dụng.

– 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 các lĩnh vực áp dụng.

CSTT (knowlegde) **Cơ sở sự kiện** **Cơ sở luật**



- Các sự kiện (Fact) được mô tả bởi Vị từ (Predicate). Mỗi vị từ là một phát biểu, quan sát về đối tượng mà ta đang xét.

F={p(t1,t2….tn)/p vị từ} p: Tên vị từ

ti: hạng thức (term) có thể là một biến, một hằng, hoặc một hàm (rất quan trọng)

- Luật(Rule): Mọi tri thức chuyên môn đều được biểu diễn bằng mệnh đề: Nếu…..thì…. p1(t1….tk)……..pn(u1…..un) suy ra q(v1….vm)

Trong đó: pi, q: Tên vị từ

ti, u, v: các hạng thức

Ví dụ:

Vị từ: Nguoi(Lan) (Lan là người), ChaMe(Lan, Hoa) (Lan là mẹ Hoa), ChaMe(Lan, Ngọc) Luật: Nếu ChaMe(Lan, Hoa),ChaMe(Lan, Ngọc) thì ChiEm(Hoa,Ngọc) (Hoa là chị em với Ngọc)

4.7. Các bước chính để xây dựng một CSTT:

– Cần phải xác định CSTT mà ta xây dựng có các đối tượng nào, các sự kiện nào, các thuộc tính nào, các quan hệ nào. Khi nào hiểu tường tận về lĩnh vực áp dụng mới bắt đầu xây dựng CSTT.

– Xây dựng hệ thống từ vựng: Các hằng, các hàm và các vị từ. Cần biểu diễn quan hệ hàm hay vị từ, các hàm (vị từ) là các hàm (vị từ) của các đối số nào. Chọn các tên hằng, tên hàm, tên vị từ sao cho nói lên được nội dung mà nó cần mô tả.

– Biểu diễn tri thức chung về lĩnh vực.

Các thủ tục suy diễn:

Các hệ tri thức mà CSTT bao gồm các luật được gọi là các hệ dựa trên luật (rule – based system). Và các thủ tục suy diễn trong các hệ dựa trên luật. Khi chúng ta đã lưu trữ một CSTT thì cần có thủ tục suy diễn cơ bản.

Thủ tục suy diễn tiến

a) Suy diễn tiến (Forward chaining)

Tư tưởng cơ bản của suy diễn tiến là áp dụng các luật duy diễn Modus Pones tổng quát. Trong mỗi bước của thủ tục suy diễn tiến thì ta xét tất các điều kiện của luật đều được thỏa 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 suy diễn 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 làm thỏa mãn (bởi các sự kiện trong cơ sở sự kiện), thì ta suy ra sự kiện trong phần kết của luật. Chính vì thế mà có tên là suy diễn tiến.

Quá trình lập luận tiến không hướng tới giải quyết một vấn đề nào cả. Nó 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 gọi là lập luận điều khiển dữ liệu hoặc lập luận định hướng dữ liệu.

b) Thủ tục suy diễn tiến

Trong hệ dựa trên luật, ta tách CSTT thành hai phần: Cơ sở luật, kí hiệu RB(Rule Base) và Cơ sở sự kiện (bộ nhớ làm việc), kí hiệu FB (Fact Base)

Với mỗi luật R:

Nếu P1 và P2 ... và Pm thì Q

Ký hiệu Conds là danh sách cá điều kiện của luật, Conds = [P1,...,P2,...,Pm], và kí hiệu Conc là kết luận của luật, Conc = Q. Xem xét 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 suy diễn tiến ta sử dụng luật suy diễn sau:

$$\frac{P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_m \Rightarrow Q, \quad S}{P_1' \wedge \ldots \wedge P_{i-1}' \wedge P_i' \wedge \ldots \wedge P_m' \Rightarrow Q'}$$

Trong đó: Pi hợp nhất với S bởi phép thế θ , tức:

$$P_i\theta = S\theta, \text{ và } P_k' = P_k\theta \ (k = 1, \ldots, m; k \neq i), \ Q' = Q\theta.$$

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 suy diễn tiến m lần (nếu có thể) ta suy ra đư ợc một sự kiện. Sự kiện này là kết quả của việc áp dụng phép thế biến vào kết luận của luật.

❖ Thủ tục For_chain: Thủ tục này 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 ta dẫn tới một luật có phần điều kiện rỗng tức là ra đã suy ra một sự kiện. Trong thủ tục For_chain, luật R = (Conds, Conc) là biến địa phương của thủ tục, Conds = [P1,...,P2,...,Pm]

```
procedure For_Chain (conds, conc);
begin
for mỗi S trong FB do
if S hợp nhất với điều kiện Pi trong Conds bởi phép thế θ
then {
Conds ← [P1θ, .... Pi-1θ, Pi-1θ, .... Pmθ];
Conc  ← Conc θ;
if Conds rỗng then Add(Conc, FB)
else For_Chain(Conds, Conc);
}
end;
```

Trong thủ tục trên, thủ tục Add(Conc,FB) thực hiện kiểm tra 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 suy diễn 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:

```
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 có các sự kiện sau :

Tom là ngựa

Ken là ngựa

Kit là ngựa

Bin là ngựa

Tom là me 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 N gua(x) (x là ngựa), Me(x,y) (x là mẹ của y), N hanh(y) (y chạy nhanh), Giá(x) (x có giá), thì luật trên có thể viết:

Ngua(x) ^ Me(x,y) ^ Nhanh(y) => Gia(x)

Cơ sở sự kiện gồm các câu phần tử sau;

Ngua(Tom) (1)

Ngua(Ken) (2)

Ngua(Kit) (3)

Ngua(Bin) (4)

Me(Tom,Bin) (5)

Me(Tom,Ken) (6)

Me(Bin,Kit) (7)

Nhanh(Kit) (8)

Nhanh(Bin) (9)

Áp dụng thủ tuc For_Chain cho các luật mẹ và FB gồm các sự kiện (1) đến (9).

Sự kiện (1) khớp với điều kiện thứ nhất của luật bởi phép thế [x/Tom], ta có:

Me(Tom,y) ^ Nhanh(y) => Gia(Tom)

Sự kiện (5) hợp nhất với điều kiện Me(Tom/y) bởi phép thế [y/Bin], suy ra :

Nhanh(Bin) => Gia(Tom) Từ (9) và kéo theo trên, suy ra Gia(Tom)

Sự kiện (2) cũng hợp nhất với điều kiện thứ nhất của luật, do đó suy ra:

Me(Ken,y) ^ Nhanh(y) => Gia(Ken)

Tới đây ta không suy diễn tiếp tục được vì không có sự kiện nào hợp nhất được với điều kiện Me(Ken,y). Điều tương tự cũng xảy ra khi mà biến x trong luật mẹ được thế bởi Kit

Từ (4) và luật mẹ, suy ra:

Me(Bin,y) ^ Nhanh(y) => Gia(Bin)

Sự kiện (7) hợp nhất với điều kiện Me(Bin,y), suy ra Nhanh(Kit) => Gia(Bin)

Từ kéo theo này và sự kiện (8), suy ra Gia(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".

b) Thủ tục Suy diễn lùi (Backward chaining)

Suy diễn lùi (Backward chaining)

Trong suy diễn 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 mình, hoặc là bị bác bỏ (bởi các sự kiện trong bộ nhớ làm việc).

Quá trình suy diễn 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 hau 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, 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 được sinh ra trong quá trình phát triển các giả thuyết bởi các luật được chọn thích hợp đều được thỏa mãn (đều có trong bộ nhớ làm việc) thì giả thuyết đã đưa ra được xem là đúng. Ngư ợc 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.

Một câu hỏi đặt ra có thể xem như một giả thuyết (kí hiệu Hyp) cần để kiểm ta. Giả thuyết có thể là một câu phân tử hoặc hội của các câu phân tử:

Hyp = H1 $\wedge$ ... $\wedge$ Hm

Trong đó: Hi (i = 1,…,m) là các câu phân tử.

Mục đích của chúng ta là kiểm chứng xem giả thuyết có thể trở thành không đúng và nếu có thì với các phép thế biến nào nó trở thành đúng.

Xét danh sách các giả thuyết Hi:

Hyp = [H1,…,Hm]

Chúng ta sẽ xét mỗi luật: P1 $\wedge \cdots \wedge$ Pm => Q như một cặp (Conds, Conc) với Conds là danh sách các điều kiện của luật.

Conds = [P1,...,Pm]

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. 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 m ới 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 chuyển thế biến θ.

Thủ tục suy diễn lùi

Tư tưởng 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 để nhận được danh sách Hyp;

2. Thêm các điều kiện cảu luật Conds vào danh sách Hyp1;

3. Áp dụng phép thế θ_1 vào các giả thuyết trong danh sách Hyp1;

4. Lấy hợp thành của các phép thế θ và θ_1để nhận được phép thế θ' mới;

if Hyp1= [] then cho ra θ'

else Bachward_Chaining (Hyp1, θ');

end;

Trong thủ tục lập luận lùi, mỗi θ được cho ra một phép thế biến làm cho giả thuyết ban đầu trở thành đúng, tức là (Hyp) θ = H1θ ^ .... ^ Hmθ là đúng (hệ quả logic cuả cơ sở tri thức). Do đó, mỗi phép thế biến θ được chọn ra bởi thủ tục là các câu trả lời cho câu hỏi đặt ra.

Ví dụ:

Giả sử CSTT chứa các sự kiện sau:

Ngua(Tom)	(Tom là ngựa)	(1)
Ngua(Ken)		(2)
Ngua(Kit)		(3)
Ngua(Bin)		(4)
Me(Tom,Bin)	(Tom là mẹ Bin)	(5)
Me(Tom,Ken)		(6)
Me(Bin,Kit)		(7)
Nhanh(Kit)		(8)
Thangcuoc(Bin)	(Bin thắng cuộc)	(9)

và chứa hai luật sau:

- Ngua(x) ^ Me(x,y) ^ Nhanh(y) => Gia(x) (10)

(Nếu 1.x là ngựa và 2. x mẹ của y, và 3. y chạy nhanh thì x có giá)

- Thangcuoc(z) $\Rightarrow$ Nhanh(z) (11)

(Nếu z thắng cuộc thì z chạy nhanh)

Câu hỏi đặt ra là: Con ngựa nào có giá? Giả thuyết ban đầu Hyp = [Gia(w)] và θ = [].

Giả thuyết Gia(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 = [Ngua(x), Me(x,y), N hanh(y)]

Và $\theta = \theta\theta$ 1 = [w/x]

Giả thuyết Ngua(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 = [Me(Tom,y), N hanh(y)] Và θ = [w/x][x/Tom] = [w/Tom]

Giả thuyết Me(Tom,y) hợp nhất với sự kiện (5) bởi phép thế θ_1 = [y/Bin], ta thừa nhận danh sách các giả thuyết

Hyp = [Nhanh(Bin)]

Và θ = [w/Tom][y/Bin] = [w/Tom,y/Bin]

Giả thuyết N hanh(Bin) hợp nhất được với kết luận của luật (11) bởi phép thế θ_1 = [z/Bin], do đó ta có

Hyp = [Thangcuoc(Bin)] Và θ = [w/Tom, y/Bin, z/Bin]

Giả thuyết Thangcuoc(Bin) trùng với sự kiện (9) (hợp nhất với phép thế θ_1= []). Do đó, danh sách cá giả thuyết trở thành rỗng với phép thế θ = [w/Tom, y/Bin, z/Bin]. N hư vậy với phép thế này thì giả thuyết Gia(w) trỏ thành đúng, hay nói cách khác, Tom là con ngựa có giá.

Phần 2. Hệ chuyên gia

Chương V: Tổng quan về hệ chuyên gia

5.1 Hệ chuyên gia – chương trình ứng dụng(HCG- CTƯD)

Theo E. Feigenbaum : «*Hệ chuyên gia (Expert System) là một chương trình máy tính thông minh sử dụng tri thức (knowledge) và các thủ tục suy luận (inference procedures) để giải những bài toán tương đối khó khăn đòi hỏi những chuyên gia mới giải được*».

Hệ chuyên gia là một hệ thống tin học có thể mô phỏng (emulates) năng lực quyết đoán (decision) và hành động (making abilily) của một chuyên gia (con người). Hệ chuyên gia là một trong những lĩnh vực ứng dụng của *trí tuệ nhân tạo* (Artificial Intelligence).

Hệ chuyên gia sử dụng các tri thức của những chuyên gia để giải quyết các vấn đề (bài toán) khác nhau thuộc mọi lĩnh vực.

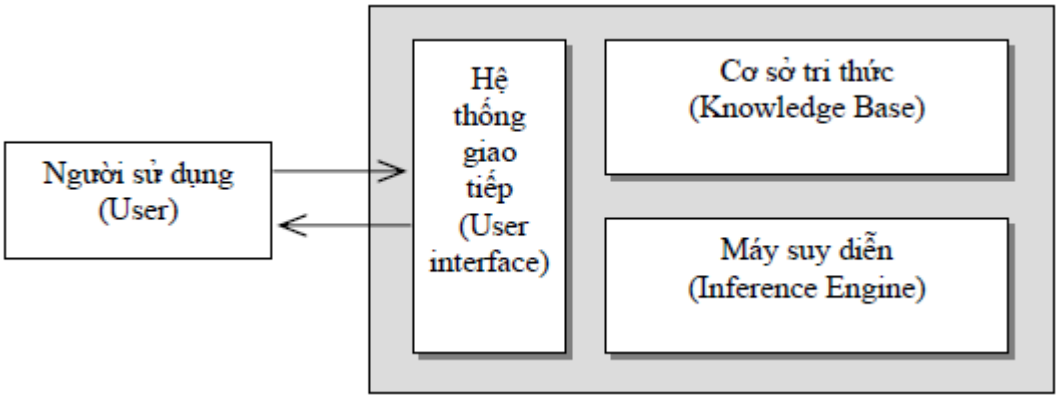
Tri thức (knowledge) trong hệ chuyên gia phản ánh sự tinh thông được tích tụ từ sách vở, tạp chí, từ các chuyên gia hay các nhà bác học. Các thuật ngữ hệ chuyên gia, *hệ thống dựa trên tri thức* (knowledge−based system) hay *hệ chuyên gia dựa trên tri thức* (knowledge−based expert system) thường có cùng nghĩa.

Một hệ chuyên gia gồm ba thành phần chính là *cơ sở tri thức* (knowledge base), *máy suy diễn* hay *môtơ suy diễn* (inference engine), và h*ệ thống giao tiếp với người sử dụng* (user interface). Cơ sở tri thức chứa các tri thức để từ đó, máy suy diễn tạo ra câu trả lời cho người sử dụng qua hệ thống giao tiếp.

Người sử dụng (user) cung cấp *sự kiện* (facts) là những gì đã biết, đã có thật hay những thông tin có ích cho hệ chuyên gia, và nhận được những câu trả lời là những lời khuyên hay những gợi ý đúng đắn (expertise).

Hoạt động của một hệ chuyên gia dựa trên tri thức được minh họa 5

như sau:



Mỗi hệ chuyên gia chỉ đặc trưng cho một *lĩnh vực vấn đề* (problem domain) nào đó, như y học, tài chính, khoa học hay công nghệ, v.v..., mà không phải cho bất cứ một lĩnh vực vấn đề nào.

Tri thức chuyên gia để giải quyết một vấn đề đặc trưng được gọi là *lĩnh vực tri thức (knowledge domain)*. Ví dụ : hệ chuyên gia về lĩnh vực y học để phát hiện các căn bệnh lây nhiễm sẽ có nhiều tri thức về một số triệu chứng lây bệnh, lĩnh vực tri thức y học bao gồm các

căn bệnh, triệu chứng và chữa trị.

Chú ý rằng lĩnh vực tri thức hoàn toàn nằm trong lĩnh vực vấn đề. Phần bên ngoài lĩnh vực tri thức nói lên rằng không phải là tri thức cho tất cả mọi vấn đề.

Tùy theo yêu cầu người sử dụng mà có nhiều cách nhìn nhận khác nhau về một hệ chuyên gia.

Loại người sử	*Vấn đề đặt ra*
Người quản trị	Tôi có thể dùng nó để làm gì ?
Kỹ thuật viên	Làm cách nào để tôi vận hành nó tốt nhất ?
Nhà nghiên cứu	Làm sao để tôi có thể mở rộng nó ?
Người sử dụng cuối	Nó sẽ giúp tôi cái gì đây ? Nó có rắc rối và tốn kém không ? Nó có đáng tin cậy không ?

Có bốn đặc trưng cơ bản của một hệ chuyên gia :

• *Hiệu quả cao* (high performance). Khả năng trả lời với mức độ tinh thông bằng hoặc cao hơn so với chuyên gia (người) trong cùng lĩnh vực.

• *Thời gian trả lời thoả đáng* (adequate response time). Thời gian trả lời hợp lý, bằng hoặc nhanh hơn so với chuyên gia (người) để đi đến cùng một quyết định. Hệ chuyên gia là một hệ thống thời gian thực (real time system).

• *Độ tin cậy cao* (good reliability). Không thể xảy ra sự cố hoặc giảm sút độ tin cậy khi sử dụng.

• *Dễ hiểu* (understandable). Hệ chuyên gia giải thích các bước suy luận một cách dễ hiểu và nhất quán, không giống như cách trả lời bí ẩn của các hộp đen (black box).

Những ưu điểm của hệ chuyên gia:

• *Phổ cập* (increased availability). Là sản phẩm chuyên gia, được phát triển không ngừng với hiệu quả sử dụng không thể phủ nhận.

• *Giảm giá thành* (reduced cost).

• *Giảm rủi ro* (reduced dangers). Giúp con người tránh được trong các môi trường rủi ro, nguy hiểm.

• *Tính thường trực* (Permanance). Bất kể lúc nào cũng có thể khai thác sử dụng, trong khi con người có thể mệt mỏi, nghỉ ngơi hay vắng mặt.

• *Đa lĩnh vực* (multiple expertise). chuyên gia về nhiều lĩnh vực khác nhau và được khai thác đồng thời bất kể thời gian sử dụng.

• *Độ tin cậy* (increased relialility). Luôn đảm bảo độ tin cậy khi khai thác.

• *Khả năng giảng giải* (explanation). Câu trả lời với mức độ tinh thông được giảng giải rõ ràng chi tiết, dễ hiểu.

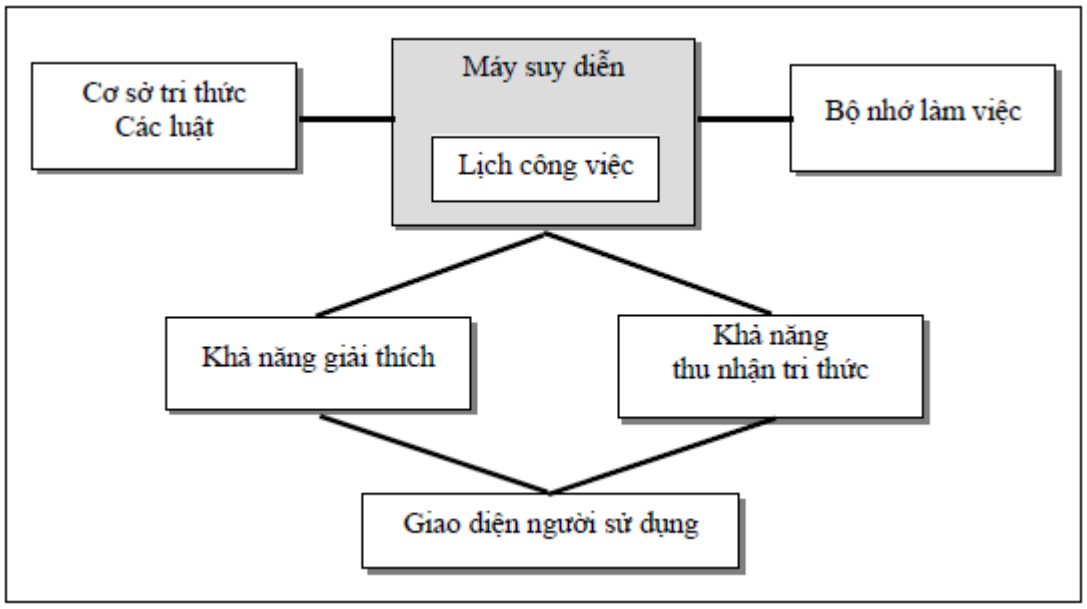
• *Khả năng trả lời* (fast reponse). Trả lời theo thời gian thực, khách quan.

• *Tính ổn định, suy luận có lý và đầy đủ mọi lúc mọi nơi* (steady, une motional, and complete response at all times).

- *Trợ giúp thông minh như một người hướng dẫn* (intelligent -tutor).
- *Có thể truy cập như là một cơ sở dữ liệu thông minh* (intelligent database).

5.2 Cấu trúc hệ chuyên gia

Minh họa cấu trúc



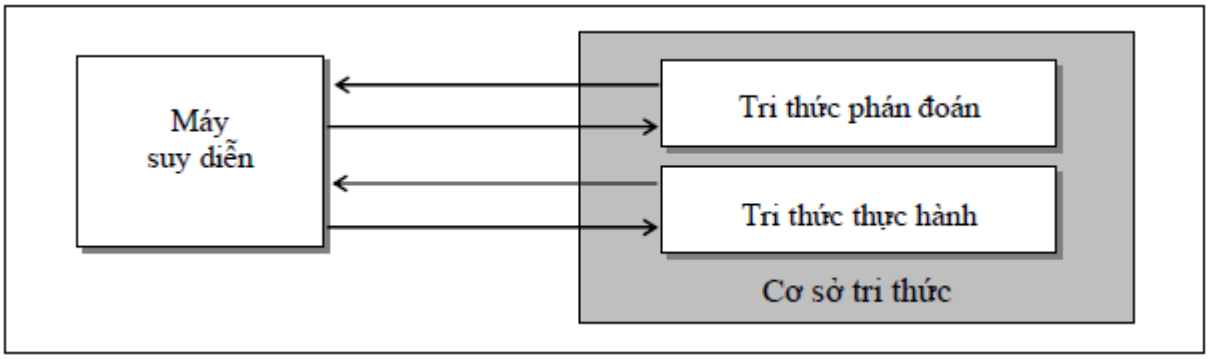
HCG:

- *Cơ sở tri thức* (knowledge base). Gồm các phần tử (hay đơn vị) tri thức, thông thường được gọi là *luật* (rule), được tổ chức như một cơ sở dữ liệu.
- *Máy duy diễn* (inference engine). Công cụ (chương trình, hay bộ xử lý) tạo ra sự suy luận bằng cách quyết định xem những luật nào sẽ làm thỏa mãn các sự kiện, các đối tượng. , chọn ưu tiên các luật thỏa mãn, thực hiện các luật có tính ưu tiên cao nhất.
- *Lịch công việc* (agenda). Danh sách các luật ưu tiên do máy suy diễn tạo ra thoả mãn các sự kiện, các đối tượng có mặt trong bộ nhớ làm việc.
- *Bộ nhớ làm việc* (working memory). Cơ sở dữ liệu toàn cục chứa các sự kiện phục vụ cho các luật.
- *Khả năng giải thích* (explanation facility). Giải nghĩa cách lập luận của hệ thống cho người sử dụng.
- *Khả năng thu nhận tri thức* (explanation facility). Cho phép người sử dụng bổ sung các tri thức vào hệ thống một cách tự động thay vì tiếp nhận tri thức bằng cách mã hoá tri thức một cách tường minh. Khả năng thu nhận tri thức là yếu tố mặc nhiên của nhiều hệ chuyên gia.
- *Giao diện người sử dụng* (user interface). Là nơi người sử dụng và hệ chuyên gia trao đổi với nhau.

Cơ sở tri thức còn được gọi là *bộ nhớ sản xuất* (production memeory) trong hệ chuyên gia. Trong một cơ sở tri thức, người ta thường phân biệt hai loại tri thức là *tri thức phán đoán* (assertion knowledge) và *tri thức thực hành* (operating knowledge).

Các tri thức phán đoán mô tả các tình huống đã được thiết lập hoặc sẽ được thiết lập. Các tri thức thực hành thể hiện những hậu quả rút ra hay những thao tác cần phải hoàn thiện

khi một tình huống đã được thiết lập hoặc sẽ được thiết lập trong lĩnh vực đang xét. Các tri thức thực hành thường được thể hiện bởi các biểu thức dễ hiểu và dễ triển khai thao tác đối với người sử dụng.



Từ việc phân biệt hai loại tri thức, người ta nói máy suy diễn là công cụ triển khai các cơ chế (hay kỹ thuật) tổng quát để tổ hợp các tri thức phán đoán và các tri thức thực hành. Hình trên đây mô tả quan hệ hữu cơ giữa máy suy diễn và cơ sở tri thức.

5.3 Ứng dụng hệ chuyên gia

Cho đến nay, hàng trăm hệ chuyên gia đã được xây dựng và đã được báo cáo thường xuyên trong các tạp chí, sách, báo và hội thảo khoa học. Ngoài ra còn các hệ chuyên gia được sử dụng trong các công ty, các tổ chức quân sự mà không được công bố vì lý do bảo mật. Bảng dưới đây liệt kê một số lĩnh vực ứng dụng diện rộng của các hệ chuyên gia.

Lĩnh vực	*Ứng dụng diện rộng*
Cấu hình (Configuration)	Tập hợp thích đáng những thành phần của một hệ thống theo cách riêng
Chẩn đoán (Diagnosis)	Lập luận dựa trên những chứng cứ quan sát được
Truyền đạt	Dạy học kiểu thông minh sao cho sinh viên có thể hỏi
(Instruction)	vì sao (why?), như thế nào (how?) và cái gì nếu (what if?) giống như hỏi một người thầy giáo
Giải thích (Interpretation)	Giải thích những dữ liệu thu nhận được
Kiểm tra (Monitoring)	So sánh dữ liệu thu lượm được với dữ liệu chuyên môn để đánh giá hiệu quả
Lập kế hoạch	Lập kế hoạch sản xuất theo yêu cầu
Dự đoán (Prognosis)	Dự đoán hậu quả từ một tình huống xảy ra
Chữa trị (Remedy)	Chỉ định cách thụ lý một vấn đề
Điều khiển (Control)	Điều khiển một quá trình, đòi hỏi diễn giải, chẩn đoán, kiểm tra, lập kế hoạch, dự đoán và chữa trị

Chương VI: Biểu diễn tri thức

6.1 Mở đầu

Ở chương trước chúng ta đã có khái niệm đơn giản như thế nào là một hệ chuyên gia. Một thành phần vô cùng quan trọng của hệ chuyên gia đó là cơ sở tri thức. Thông qua các phiên thu nạp tri thức (trực tiếp hay gián tiếp) chúng ta đã xây dựng được một cơ sở tri thức cho hệ chuyên gia. Vậy làm thế nào đẻ quản lí và thao tác xử lí để hệ chuyên gia có thể hoạt động được. Trong chương này chúng ta sẽ đề cập đến vấn đề đó và giải quyết vấn đề đó như thế nào.

Tri thức của một hệ chuyên gia có thể được biểu diễn theo nhiều cách khác nhau. Thông thường người ta sử dụng các cách sau đây :

- Biểu diễn tri thức bởi các luật sản xuất
- Biểu diễn tri thức nhờ mệnh đề logic
- Biểu diễn tri thức nhờ mạng ngữ nghĩa
- Biểu diễn tri thức nhờ ngôn ngữ nhân tạo

Ngoài ra, người ta còn sử dụng cách biểu diễn tri thức nhờ các sự kiện không chắc chắn, nhờ bộ ba : đối tượng, thuộc tính và giá trị (O-A-V: Object-Attribute-Value), nhờ khung (frame), v.v... Tuỳ theo từng hệ chuyên gia, người ta có thể sử dụng một cách hoặc đồng thời cả nhiều cách.

6.2 Dư thừa (Redundancy)

CSTT = (CS luật, CS sự kiện)

CS luật: Rule Base – tri thức chuyên về l.vực

CS sự kiện: Fact base – thông tin về một bài toán (cụ thể)

6.2.1 Dư thừa luật:

Định nghĩa: cho CSTT:

$B_1 = (R_1, F_1)$

$B_2 = (R_2, F_2)$

Ta nói R1 ≡ R2 (Sức mạnh suy diễn của R1 bằng sức mạnh suy diễn của R2)

Bao đóng suy diễn cho R. Xét $A \subseteq F$

$A_R^+ = \{f \in F \,/\, A \to f\}$

VD:

1) $a \wedge b \to c$

2) $b \wedge c \wedge d \to e$

3) $a \wedge d \to f$

4) $c \to g$

5) a $\rightarrow$ h

6) d $\wedge$ c $\rightarrow$ h

7) b $\rightarrow$ u

$\{a\}^+ = \{a,h\}$

$\{a,b\}^+ = \{a,b,c,g,u,h\}$

Nhận xét: Một luật trong logic mệnh đề ≡ PTH và bao đóng = bao đóng của PTH

6.2.2 Dư thừa sự kiện:

Giả sử cơ sở luật không chứa luật dư thừa

Định nghĩa: Xét r: left $\rightarrow$ q

f $\rightarrow$ left

f được coi là dư thừa trong r $\Leftrightarrow$ thay r bởi r': left \ $\{f\}$ $\rightarrow$ q vẫn có tập luật tương đương (R|{r})$\vee${r}≡R

CHÚ Ý:

- Dư thừa không có ý nghĩa là vô ích

- Duy trì dư thừa kéo theo nâng cao chất lượng suy diễn

6.3 **Mâu thuẫn** (consistency – inconsistency, ký hiệu ><)

6.3.1 Mâu thuẫn tường minh

Khi duyệt CSTT, chỉ qua ht bên ngoài của các luật đã phát hiện ra ><

Định nghĩa:

Ta nói: r: left $\rightarrow$ q >< r': left' $\rightarrow$ q'

$\Leftrightarrow$ left $\subseteq$ left' hoặc left' $\subseteq$ left

q >< q'

VD:

Trong logic mệnh đề: p >< $\neg$p

Trong logic vị từ: p(a) >< $\neg p$ (a)

$\neg$p (a) >< $\forall$ x p(x)

Xử lý mâu thuẫn

- Xử lý cục bộ : r >< r'

+ theo trọng số

+ theo lĩnh vực chuyên môn

r thuộc lĩnh vực chuyên môn A

r’ thuộc lĩnh vực chuyên môn B

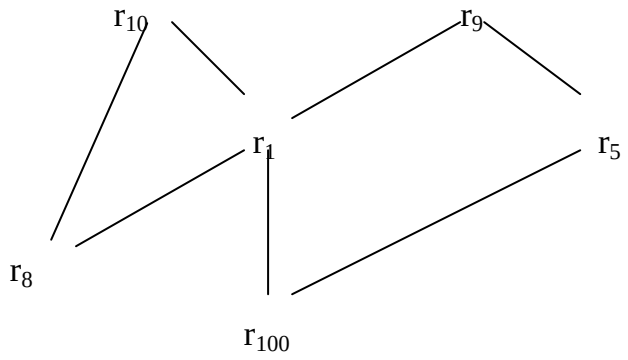
+ theo xử lý ngoại lệ

r: chung

r’: ngoại lệ

left left’

- Xử lý tổng thể: thể hiện trong đồ thị mâu thuẫn với đỉnh là các luật và cạnh thể hiện các mâu thuẫn. Như vậy:



Biện pháp: Vứt bỏ tập luật (tập con các luật) $R_0 \subseteq R$ sao cho $R \setminus R_0$ không còn ><. Theo những đồ thị: bỏ luật cùng các cạnh liên thuộc để có một đồ thị con $R \setminus R_0$ chỉ còn các đỉnh cô lập (tức là $(R, R_0,))$

6.3.2. Mâu thuẫn không tường minh (KTM)

Định nghĩa: CSTT (R,F) chứa mâu thuẫn không tường minh nếu:

1) F không chứa cặp sự kiện đụng độ, và

2) F_R^+ chứa cặp sự kiện đụng độ

6.4 Lưu trữ CSTT

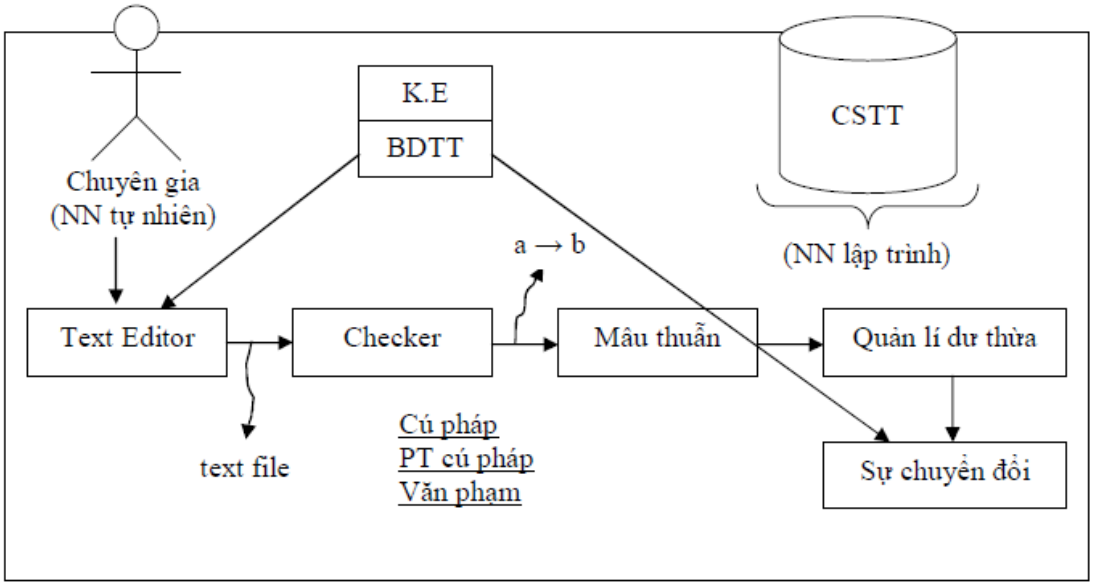
a. Cấu trúc tĩnh: Với bảng luật lưu trữ bằng mảng

VD:

1) $a \wedge b \rightarrow c$	a	b		c
2) $b \wedge c \wedge d \rightarrow e$	b	c	d	e
3) $a \wedge d \rightarrow f$	a	d		f
4) $c \rightarrow g$	c			g
5) $a \rightarrow h$	a			h
6) $d \wedge c \rightarrow h$	d	c		h
7) $b \rightarrow u$	b			u

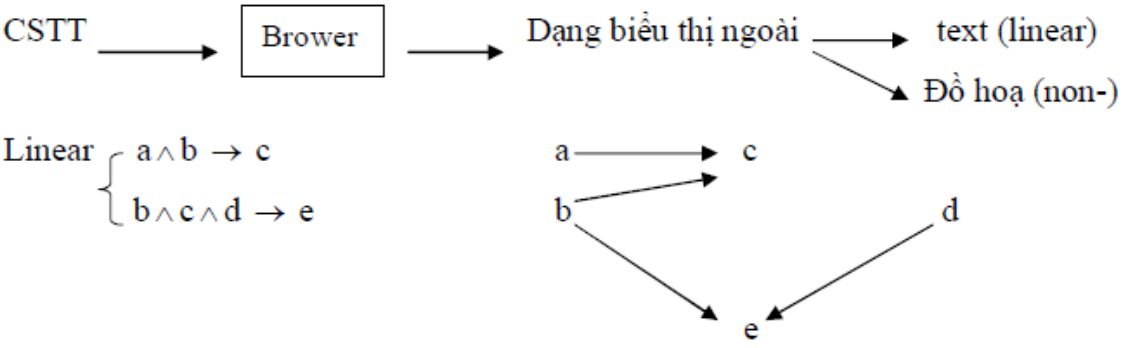
b. Cấu trúc động: Sử dụng DSLK động thay vì mảng tĩnh.

6.5 Soạn thảo tri thức

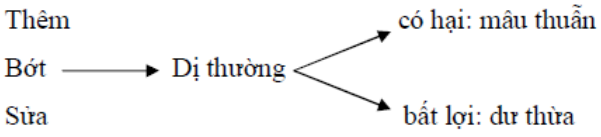


6.6 Cập nhật sửa đổi

a. Hiển thị



b. Cập nhật



Chương VII. Các kỹ thuật suy diễn và lập luận

7.1 Nhập môn

Động cơ, mô tơ hay máy suy diễn gồm 2 bộ phận chính:

Cơ chế suy diễn (Processor) gồm:

+ Suy diễn tiến

+ Suy diễn lùi

Cơ chế cổ điển (control unit):

+ chọn hướng suy diễn (MACRO)

+ chọn luật

thường có MẸO (heuristc → metaknowledge)

+ phân rã CSTT → SD phân tán và SD song song

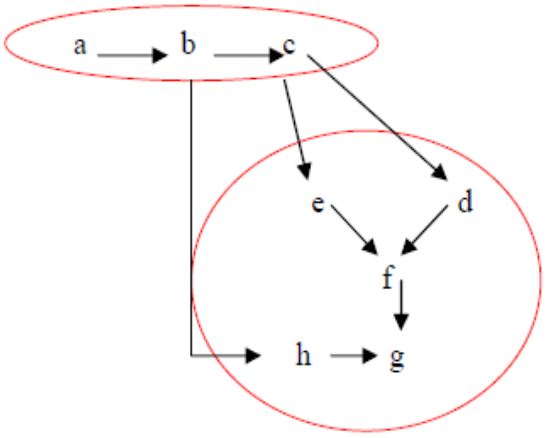
+ Lọc (tinh) (nhìn thấy cái nào không cần thiết thì loại, xác định cái nào được chọn trước)

7.2 Phân rã CSTT

Fact Precedence Graph (FPG) = (F, A)

+ Đỉnh : tập các sự kiện

+ Cung: $(a,b) \in A \Leftrightarrow \exists$ r: left $\rightarrow b \in R$; $a \in$ left



1) $a \rightarrow b$

2) $b \rightarrow c$

3) $c \rightarrow e$

4) $c \rightarrow d$

5) $d \wedge e \rightarrow f$

6) $b \rightarrow h$

7) $f \wedge h \rightarrow g$

Tập sự kiện:

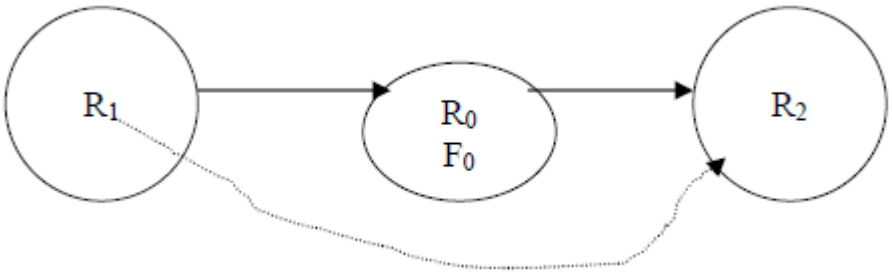
F = { a, b, c, d, e, f, g, h} tách ra hai sự kiện:

F_1 = { a, b, c} R_1 = { $a \rightarrow b$, $b \rightarrow c$}

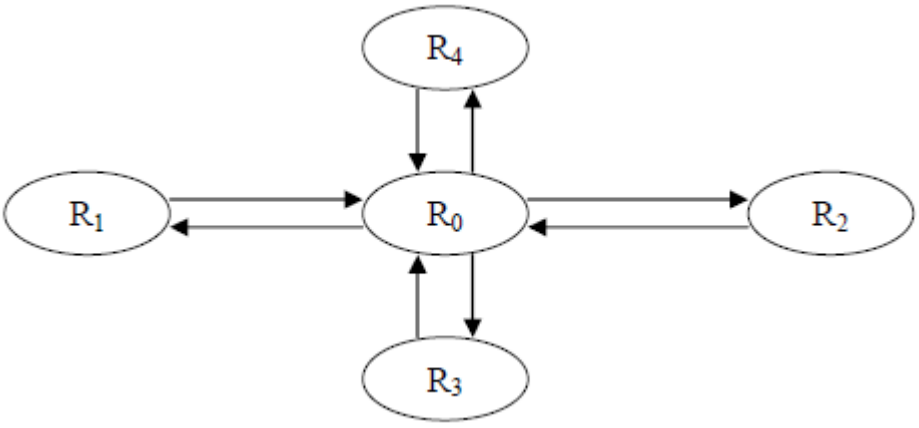
F_2 = { e, d, f, g, h} R_2 = { $d \wedge e \rightarrow f$, $f \wedge h \rightarrow g$}

F_0 = {b, c, d, b, h} R_0 = {$c \rightarrow e$, $c \rightarrow d$, $b \rightarrow h$}

Một cách phân rã CSTT: eval({F_1,F_2}) → min



Mô hình star:



Nếu phân rã dựa trên tập luật làm gốc thì dẫn đến full condition;

Phân rã theo tập sự kiện → hình sao

7.3 Mô tơ suy diễn:

Suy diễn tiến, lùi (xem lại các chương trước)

7.4 Biểu diễn tri thức bằng Logic vị từ và suy diễn:

Luật $r_{i:}$ $P_1(\ldots) \wedge P_2(\ldots) \wedge \ldots\ldots \wedge P_n(\ldots) \rightarrow q(\ldots)$

với $P_i(\ldots)$ và $q(\ldots)$ là các vị từ

7.5 Ứng dụng các kỹ thuật suy diễn

Chương VIII. Hệ hỗ trợ quyết định

8.1 Khái niệm về hệ hỗ trợ ra quyết định

Các khái niệm căn bản về quyết định

Thí dụ về hệ hỗ trợ quyết định (HHTQĐ)

- Nghiên cứu và hoạch định tiếp thị: chính sách giá cho khách hàng, dự báo sản phẩm tiêu thụ ..
- Hoạch định chiến lược và vận hành: theo dõi, phân tích và báo cáo về xu hướng thị trường ..
- Hỗ trợ bán hàng: chi tiết và tổng hợp tình hình bán hàng, so sánh và phân tích xu hướng bán hàng …

Quyết định là gì ?

Đó là một lựa chọn về "đường lối hành động" (Simon 1960; Costello & Zalkind 1963; Churchman 1968), hay "chiến lược hành động" (Fishburn 1964) dẫn đến "một mục tiêu mong muốn" (Churchman 1968)

Ra quyết định là gì ?

"Một quá trình lựa chọn có ý thức giữa hai hay nhiều phương án để chọn ra một phương án tạo ra được một kết quả mong muốn trong các điều kiện ràng buộc đã biết". Quyết định có thể là nhận thức ở dạng sự kiện, "Chi $10,000 cho quảng cáo vào quý 3" . Quyết định có thể là nhận thức ở dạng quá trình, "Trước tiên thực hiện A, sau đó B hai lần và nếu có đáp ứng tốt hãy thực thi C". Quyết định có thể là một hoạt động giàu kiến thức, Quyết định có kết luận nào thì hợp lý/hợp lệ trong hoàn cảnh nào ? Quyết định có thể là những thay đổi trạng thái kiến thức. Quyết định có chấp nhận một kiến thức mới không ?

Tại sao phải hỗ trợ ra quyết định ?

Nhu cầu hỗ trợ ra quyết định:

+ Ra quyết định luôn cần xử lý kiến thức

+ Kiến thức là nguyên liệu và thành phẩm của ra quyết định, cần được sở hữu hoặc tích lũy bởi người ra quyết định

Giới hạn về nhận thức (trí nhớ có hạn ..)

Giới hạn về kinh tế (chi phí nhân lực ..)

Giới hạn về thời gian

Áp lực cạnh tranh

Bản chất của hỗ trợ ra quyết định

cung cấp thông tin, tri thức

có thể thể hiện qua tương tác người - máy, qua mô phỏng

Các yếu tố ảnh hưởng đến ra quyết định

Công nghệ - thông tin - máy tính

Tính cạnh tranh - sự phức tạp về cấu trúc

Thị trường quốc tế - ổn định chính trị - chủ nghĩa tiêu thụ

Các thay đổi biến động

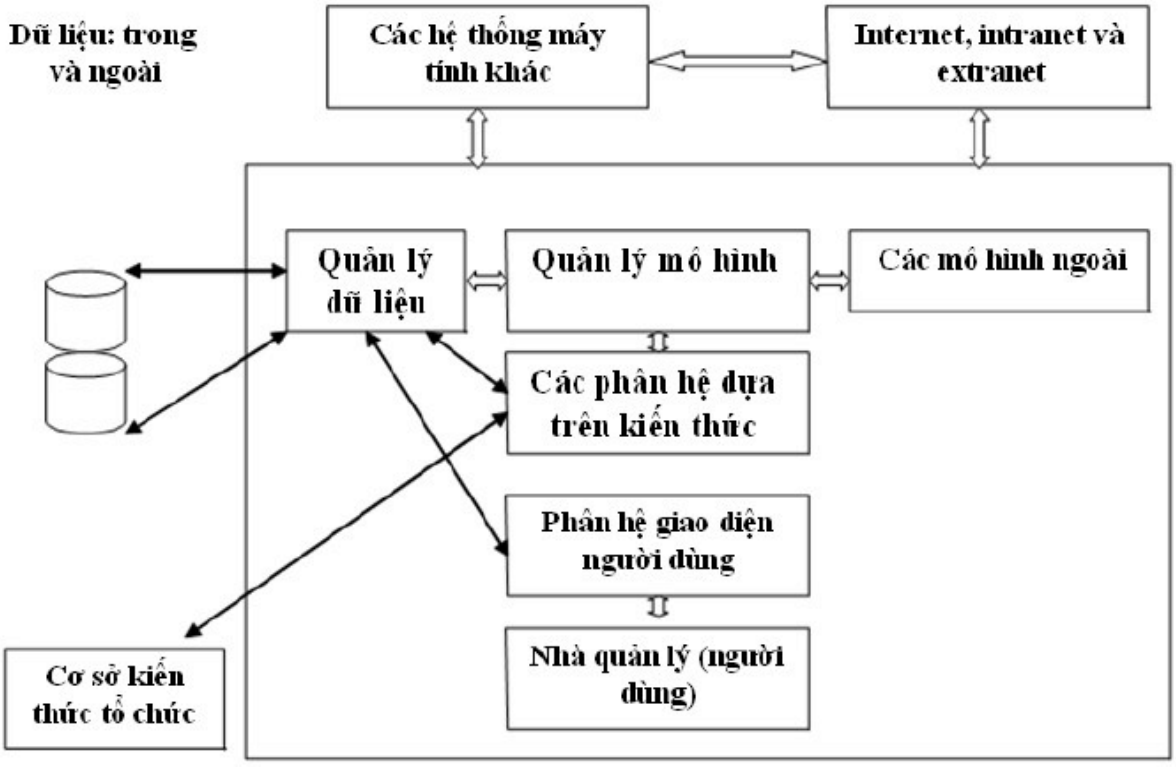
8.2 Cấu trúc của một hệ hỗ trợ ra quyết định

Các định nghĩa trước đây của HHTQĐ nhấn mạnh vào khả năng hỗ trợ các nhà ra quyết định quản lý trong các tình huống nửa cấu trúc. Như vậy, HHTQĐ có ý nghĩa là một bổ trợ cho các nhà quản lý nhằm mở rộng năng lực nhưng không thay thế khả năng phân xử của họ. Tình huống ở đây là cần đến các phân xử của các nhà quản lý hay các quyết định không hoàn toàn được giải quyết thông qua các giải thuật chặt chẽ.

Thông thường các HHTQĐ sẽ là các hệ thông tin máy tính hóa, có giao tiếp đồ họa và làm việc ở chế độ tương tác trên các mạng máy tính.

Cơ sở của các định nghĩa về HHTQĐ thay đổi từ nhận thức HHTQĐ làm gì (thí dụ, hỗ trợ ra quyết định trong các bài toán phi cấu trúc) cho đến cách thức đạt được các mục tiêu của HHTQĐ (các thành phần yêu cầu, khuôn mẫu sử dụng, quá trình phát triển ..)

Cấu trúc cơ bản của một hệ hỗ trợ ra quyết định:



Phân hệ quản lý dữ liệu gồm một cơ sở dữ liệu (database) chứa các dữ liệu cần thiết của tình huống và được quản lý bởi một hệ quản trị cơ sở dữ liệu (DBMS - data base management system). Phân hệ này có thể được kết nối với nhà kho dữ liệu của tổ chức (data warehouse) - là kho chứa dữ liệu của tổ chức có liên đới đến vấn đề ra quyết định.

Phân hệ quản lý mô hình còn được gọi là hệ quản trị cơ sở mô hình (MBMS - model base management system) là gói phần mềm gồm các thành phần về thống kê, tài chánh, khoa học quản lý hay các phương pháp định lượng nhằm trang bị cho hệ thống năng lực phân tích; cũng có thể có các ngôn ngữ mô hình hóa ở đây. Thành phần này có thể kết nối với các kho chứa mô hình của tổ chức hay ở bên ngoài nào khác.

Phân hệ quản lý dựa vào kiến thức có thể hỗ trợ các phân hệ khác hay hoạt động độc lập nhằm đưa ra tính thông minh của quyết định đưa ra. Nó cũng có thể được kết nối với các kho kiến thức khác của tổ chức.

Phân hệ giao diện người dùng giúp người sử dụng giao tiếp với và ra lệnh cho hệ thống. Các thành phần vừa kể trên tạo nên HHTQĐ, có thể kết nối với intranet/extranet của tổ chức hay kết nối trực tiếp với Internet.

Chương IX. Máy học

9.1 Khái niệm máy học

Thuật ngữ "học" theo nghĩa thông thường là tiếp thu tri thức để biết cách vận dụng. Ở ngoài đời, quá trì học diễn ra dưới nhiều hình thức khác nhau như học thuộc lòng (học vẹt), học theo kinh nghiệm (học dựa theo trường hợp), học theo kiểu nghe nhìn,... Trên máy tính cũng có nhiều thuật toán học khác nhau. Tuy nhiên, trong phạm vi của giáo trình này, chúng ta chỉ khảo sát phương pháp học dựa theo trường hợp. Theo phương pháp này, hệ thống sẽ được cung cấp một số các trường hợp "mẫu", dựa trên tập mẫu này, hệ thống sẽ tiến hành phân tích và rút ra các quy luật (biểu diễn bằng luật sinh). Sau đó, hệ thống sẽ dựa trên các luật này để "đánh giá" các trường hợp khác (thường không giống như các trường hợp "mẫu"). Ngay cả chỉ với kiểu học này, chúng ta cũng đã có nhiều thuật toán học khác nhau. Một lần nữa, với mục đích giới thiệu, chúng ta chỉ khảo sát một trường hợp đơn giản.

Có thể khái quát quá trình học theo trường hợp dưới dạng hình thức như sau :

Dữ liệu cung cấp cho hệ thống là một ánh xạ f trong đó ứng một trường hợp p trong

tập hợp P với một "lớp" r trong tập R.

$f : P \rightarrow R$

$p \rightarrow r$

Tuy nhiên, tập P thường nhỏ (và hữu hạn) so với tập tất cả các trường hợp cần quan tâm P' ($P \subset P'$). Mục tiêu của chúng ta là xây dựng ánh xạ f ' sao cho có thể ứng mọi trường hợp p' trong tập P' với một "lớp" r trong tập R. Hơn nữa, f ' phải bảo toàn f, nghĩa là :

Với mọi $p \in P$ thì $f(p) \rightarrow f'(p)$

Phương pháp học theo trường hợp là một phương pháp phổ biến trong cả nghiên cứu khoa học và mê tín dị đoan. Cả hai đều dựa trên các dữ liệu quan sát, thống kê để từ đó rút ra các quy luật. Tuy nhiên, khác với khoa học, mê tín dị đoan thường dựa trên tập mẫu không đặc trưng, cục bộ, thiếu cơ sở khoa học.

9.2 Học bằng cách xây dựng cây định danh

Phát biểu hình thức có thể khó hình dung. Để cụ thể hợn, ta hãy cùng nhau quan sát một ví dụ cụ. Nhiệm vụ của chúng ta trong ví dụ này là xây dựng các quy luật để có thể kết luận một người như thế nào khi đi tắm biển thì bị cháy nắng. Ta gọi tính chất cháy nắng hay không cháy nắng là thuộc tính quan tâm (thuộc tính mục tiêu). Như vậy, trong trường hợp này, tập R của chúng ta chỉ gồm có hai phần tử {"cháy nắng", "bình thường"}. Còn tập P là tất cả những người được liệt kê trong bảng dưới (8 người) Chúng ta quan sát hiện tượng cháy nắng dựa trên 4 thuộc tính sau : chiều cao (cao, trung bình, thấp), màu tóc (vàng, nâu, đỏ) cân nặng (nhẹ, TB, nặng), dùng kem (có, không),. Ta gọi các thuộc tính này gọi là thuộc tính dẫn xuất.

Dĩ nhiên là trong thực tế để có thể đưa ra được một kết luận như vậy, chúng ta cần nhiều dữ liệu hơn và đồng thời cũng cần nhiều thuộc tính dẫn xuất trên. Ví dụ đơn giản này chỉ nhằm để minh họa ý tưởng của thuật toán máy học mà chúng ta sắp trình bày.

Tên	Tóc	Ch.Cao	Cân Nặng	Dùng kem?	Kết quả
Sarah	Vàng	T.Bình	Nhẹ	Không	Cháy
Dana	Vàng	Cao	T.Bình	Có	Không
Alex	Nâu	Thấp	T.Bình	Có	Không
Annie	Vàng	Thấp	T.Bình	Không	Cháy
Emilie	Đỏ	T.Bình	Nặng	Không	Cháy
Peter	Nâu	Cao	Nặng	Không	Không
John	Nâu	T.Bình	Nặng	Không	Không
Kartie	Vàng	Thấp	Nhẹ	Có	Không

Ý tưởng đầu tiên của phương pháp này là tìm cách phân hoạch tập P ban đầu thành các tập Pi sao cho tất cả các phần tử trong tất cả các tập Pi đều có chung thuộc tính mục tiêu.

$P = P_1 \cup P_2 \cup ... \cup P_n$ và $\forall$ (i,j) $i \neq j$: thì ($P_i \cap P_j = \varnothing$) và $\forall$ i, $\forall$ k,l : $p_k \in P_i$ và $p_l \in P_j$ thì $f(p_k) = f(p_l)$

Sau khi đã phân hoạch xong tập P thành tập các phân hoạch Pi được đặc trưng bởi thuộc tính đích r_i ($r_i \in R$), bước tiếp theo là ứng với mỗi phân hoạch Pi ta xây dựng luật Li : $GT_i \rightarrow r_i$ trong đó các GT_i là mệnh đề được hình thành bằng cách kết hợp các thuộc tính dẫn xuất.

Một lần nữa, vấn đề hình thức có thể làm bạn cảm thấy khó khăn. Chúng ta hãy thử ý tưởng trên với bảng số liệu mà ta đã có.

Có hai cách phân hoạch hiển nhiên nhất mà ai cũng có thể nghĩ ra. Cách đầu tiên là cho mỗi người vào một phân hoạch riêng (P_1 = {Sarah}, P_2 = {Dana}, … tổng cộng sẽ có 8 phân hoạch cho 8 người). Cách thứ hai là phân hoạch thành hai tập, một tập gồm tất cả những người cháy nắng và tập còn lại bao gồm tất cả những người không cháy nắng. Tuy đơn giản nhưng phân hoạch theo kiểu này thì chúng ta chẳng giải quyết được gì !!

Đâm chồi

Chúng ta hãy thử một phương pháp khác. Bây giờ bạn hãy quan sát thuộc tính đầu tiên – màu tóc. Nếu dựa theo màu tóc để phân chia ta sẽ có được 3 phân hoạch khác nhau ứng với mỗi giá trị của thuộc tính màu tóc. Cụ thể là :

$P_{vàng}$ = { Sarah, Dana, Annie, Kartie }

$P_{nâu}$ = { Alex, Peter, John }

$P_{đỏ}$ = { Emmile }

Ta thấy rằng phân hoạch Pnâu và Pđỏ thỏa mãn được điều kiện "có chung thuộc tính

mục tiêu" ($P_{nâu}$ chứa toàn người không cháy nắng, $P_{đỏ}$ chứa toàn người cháy nắng).

Còn lại tập $P_{vàng}$ là còn lẫn lộn người cháy năng và không cháy nắng. Ta sẽ tiếp tục phân hoạch tập này thành các tập con. Bây giờ ta hãy quan sát thuộc tính chiều cao. Thuộc tính này giúp phân hoạch tập Pvàng thành 3 tập con : $P_{Vàng,\ Thấp}$ = {Annie, Kartie}, $_{PVàng,\ T.Bình}$= {Sarah} và $P_{Vàng,Cao}$= { Dana }

Quá trình này cứ thế tiếp tục cho đến khi tất cả các nút lá của cây không còn lẫn lộn giữa cháy nắng và không cháy nắng nữa. Bạn cũng thấy rằng, qua mỗi bước phân hoạch cây phân hoạch ngày càng "phình" ra. Chính vì vậy mà quá trình này được gọi là quá trình "đâm chồi". Cây mà chúng ta đang xây dựng được gọi là cây định danh.

Đến đây, chúng ta lại gặp một vấn đề mới. Nếu như ban đầu ta không chọn thuộc tính màu tóc để phân hoạch mà chọn thuộc tính khác như chiều cao chẳng hạn để phân hoạch thì sao? Cuối cùng thì cách phân hoạch nào sẽ tốt hơn?

Phương án chọn thuộc tính phân hoạch

Vấn đề mà chúng ta gặp phải cũng tương tự như bài toán tìm kiếm : "Đứng trước một ngã rẽ, ta cần phải đi vào hướng nào?". Hai phương pháp đánh giá dưới đây sẽ giúp ta chọn được thuộc tính phân hoạch tại mỗi bước xây dựng cây định danh.

Quinlan

Quinlan quyết định thuộc tính phân hoạch bằng cách xây dựng các vector đặc trưng cho mỗi giá trị của từng thuộc tính dẫn xuất và thuộc tính mục tiêu. Cách tính cụ thể như sau :

Với mỗi thuộc tính dẫn xuất A còn có thể sử dụng để phân hoạch, tính :

$V(A_j)$= ($T(j , r_1)$, $T(j , r_2)$, …, $T(j , r_n)$)

T(j, ri) = (tổng số phần tử trong phân hoạch có giá trị thuộc tính dẫn xuất A là j và có giá trị thuộc tính mục tiêu là ri) / (tổng số phần tử trong phân hoạch có giá trị thuộc tính dẫn xuất A là j)

* trong đó r_1, r_2, … , r_n là các giá trị của thuộc tính mục tiêu

Như vậy nếu một thuộc tính A có thể nhận một trong 5 giá trị khác nhau thì nó sẽ có 5 vector đặc trưng.

Một vector V(Aj) được gọi là vector đơn vị nếu nó chỉ có duy nhất một thành phần có giá trị 1 và những thành phần khác có giá trị 0.

Thuộc tính được chọn để phân hoạch là thuộc tính có nhiều vector đơn vị nhất.

Trở lại ví dụ của chúng ta, ở trạng thái ban đầu (chưa phân hoạch) chúng ta sẽ tính vector đặc trưng cho từng thuộc tính dẫn xuất để tìm ra thuộc tính dùng để phân hoạch. Đầu tiên là thuộc tính màu tóc. Thuộc tính màu tóc có 3 giá trị khác nhau (vàng, đỏ, nâu) nên sẽ có 3 vector đặc trưng tương ứng là :

VTóc (vàng) = (T(vàng, cháy nắng), T(vàng, không cháy nắng))

Số người tóc vàng là : 4

Số người tóc vàng và cháy nắng là : 2

Số người tóc vàng và không cháy nắng là : 2

Do đó

VTóc(vàng) = (2/4 , 2/4) = (0.5, 0.5)

Tương tự

VTóc(nâu) = (0/3, 3/3) = (0,1) (vector đơn vị)

Số người tóc nâu là : 3

Số người tóc nâu và cháy nắng là : 0

Số người tóc nâu và không cháy nắng là : 3

VTóc(đỏ) = (1/1, 0/1) = (1,0) (vector đơn vị)

Tổng số vector đơn vị của thuộc tính tóc vàng là 2

Các thuộc tính khác được tính tương tự, kết quả như sau :

VC.Cao(Cao) = (0/2,2/2) = (0,1)

VC.Cao(T.B) = (2/3,1/3)

VC.Cao(Thấp) = (1/3,2/3) VC.Nặng (Nhẹ) = (1/2,1/2) VC.Nặng (T.B) = (1/3,2/3)

VC.Nặng (Nặng) = (1/3,2/3)

VKem (Có) = (3/3,0/3) = (1,0)

VKem (Không) = (3/5,2/5)

Như vậy thuộc tính màu tóc có số vector đơn vị nhiều nhất nên sẽ được chọn để phân hoạch.

Sau khi phân hoạch theo màu tóc xong, chỉ có phân hoạch theo tóc vàng (Pvàng) là còn chứa những người cháy nắng và không cháy nắng nên ta sẽ tiếp tục phân hoạch tập này. Ta sẽ thực hiện thao tác tính vector đặc trưng tương tự đối với các thuộc tính còn lại (chiều cao, cân nặng, dùng kem). Trong phân hoạch Pvàng, tập dữ liệu của chúng ta còn lại là :

Tên	Ch.Cao	Cân Nặng	Dùng kem?	Kết quả
Sarah	T.Bình	Nhẹ	Không	Cháy
Dana	Cao	T.Bình	Có	Không
Annie	Thấp	T.Bình	Không	Cháy
Kartie	Thấp	Nhẹ	Có	Không

VC.Cao(Cao) = (0/1,1/1) = (0,1)

VC.Cao(T.B) = (1/1,0/1) = (1,0)

VC.Cao(Thấp) = (1/2,1/2)

VC.Nặng (Nhẹ) = (1/2,1/2)

VC.Nặng (T.B) = (1/2,1/2)

VC.Nặng (Nặng) = (0,0)

VKem (Có) = (0/2,2/2) = (0,1)

VKem (Không) = (2/2,0/2) = (1,0)

2 thuộc tính dùmg kem và chiều cao đều có 2 vector đơn vị. Tuy nhiên, số phân hoạch của thuộc tính dùng kem là ít hơn nên ta chọn phân hoạch theo thuộc tính dùng kem.

Chương X. Logic mờ và lập luận xấp xỉ

10.1 Khái niệm

Một tập hợp trong một không gian nào đó, theo khái niệm cổ điển sẽ chia không gian thành 2 phần rõ ràng. Một phần tử bất kỳ trong không gian sẽ thuộc hoặc không thuộc vào tập đã cho. Tập hợp như vậy còn được gọi là tập rõ. Lý thuyết tập hợp cổ điển là nền tảng cho nhiều ngành khoa học, chứng tỏ vai trò quan trọng của mình. Nhưng những yêu cầu phát sinh trong khoa học cũng như cuộc sống đã cho thấy rằng lý thuyết tập hợp cổ điển cần phải được mở rộng.

Ta xét tập hợp những người trẻ. Ta thấy rằng người dưới 26 tuổi thì rõ ràng là trẻ và người trên 60 tuổi thì rõ ràng là không trẻ. Nhưng những người có tuổi từ 26 đến 60 thì có thuộc tập hợp những người trẻ hay không? Nếu áp dụng khái niệm tập hợp cổ điển thì ta phải định ra một ranh giới rõ ràng và mang tính chất áp đặt chẳng hạn là 45 để xác định tập hợp những người trẻ. Và trong thực tế thì có một ranh giới mờ để ngăn cách những người trẻ và những người không trẻ đó là những người trung niên. Như vậy, những người trung niên là những người có một "độ trẻ" nào đó. Nếu coi "độ trẻ" của người dưới 26 tuổi là hoàn toàn đúng tức là có giá trị là 1 và coi "độ trẻ" của người trên 60 tuổi là hoàn toàn sai tức là có giá trị là 0, thì "độ trẻ" của người trung niên sẽ có giá trị p nào đó thoả $0 < p < 1$.

Như vậy nhu cầu mở rộng khái niệm tập hợp và lý thuyết tập hợp là hoàn toàn tự nhiên. Các công trình nghiên cứu về lý thuyết tập mờ và logic mờ đã được L.Zadeh công bố đầu tiên năm 1965, và sau đó liên tục phát triển mạnh mẽ.

Định nghĩa: Cho không gian nền U, tập $A \subset U$ được gọi là tập mờ nếu A được xác định bởi hàm μ_A:X->[0,1].

μ_A được gọi là hàm thuộc, hàm liên thuộc hay hàm thành viên (membership function)

Với $x \in X$ thì $\mu_A(x)$ được gọi là mức độ thuộc của x vào A.

Như vậy ta có thể coi tập rõ là một trường hợp đặc biệt của tập mờ, trong đó hàm thuộc chỉ nhận 2 giá trị 0 và 1.

Ký hiệu tập mờ, ta có các dạng ký hiệu sau:

- Liệt kê phần tử: giả sử U={a,b,c,d} ta co thể xác định một tập mờ $A=\frac{0.1}{a}+\frac{0.3}{b}+\frac{0.2}{c}+\frac{0}{d}$
- $A = \{(x, \mu_A(x)) \mid x \in U\}$ {(a,0.1),(b,0.3),(c,0.2),(d,0)}
- $A = \sum_{x \in U} \frac{\mu_A(x)}{x}$ trong trường hợp U là không gian rời rạc
- $A = \int_U \mu_A(x)/x$ trong trường hợp U là không gian liên tục

Lưu ý là các ký hiệu $\sum$ và $\int$ không phải là các phép tính tổng hay tích phân, mà chỉ là ký hiệu biểu thị tập hợp mờ.

Ví dụ. Tập mờ A là tập "số gần 2" xác định bởi hàm thuộc $\mu_A = e^{-(x-2)^2}$ ta có thể ký hiệu: A = $\left\{\left(x, e^{-(x-2)^2}\right) | \ x \in U\right\}$ hoặc A = $\int_{-\infty}^{+\infty} e^{-(x-2)^2} / x$

Các dạng hàm thuộc tiêu biểu

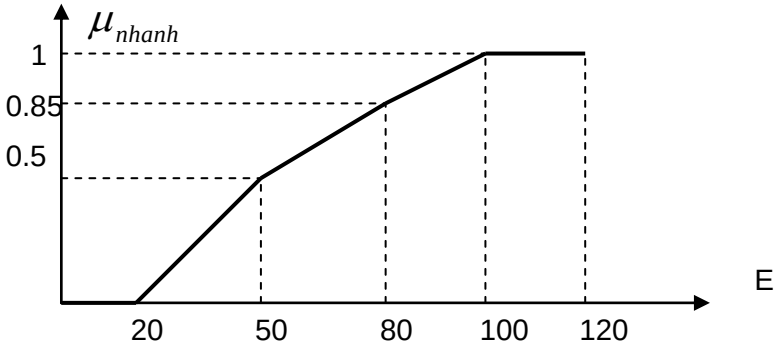
Theo lý thuyết thì hàm thuộc có thể là một hàm bất kỳ thoả μ_A:X->[0,1]. Nhưng trong thực tế thì có các dạng hàm thuộc sau đây là quan trọng và có tính ứng dụng cao hơn cả.

U={10,20,30,40,50,60,70,80,90}

Nhóm hàm đơn điệu

Nhóm này gồm đơn điệu tăng và đơn điệu giảm. Ví dụ tập hợp người già có hàm thuộc đơn điệu tăng theo tuổi trong khi đó tập hợp người trẻ có hàm thuộc đơn điệu giảm theo tuổi. Ta xét thêm ví dụ minh hoạ sau: Cho tập vũ trụ E = Tốc độ = $\{20,50,80,100,120\}$ đơn vị là km/h. Xét tập mờ F=Tốc độ nhanh xác định bởi hàm thuộc μ_{nhanh} như đồ thị

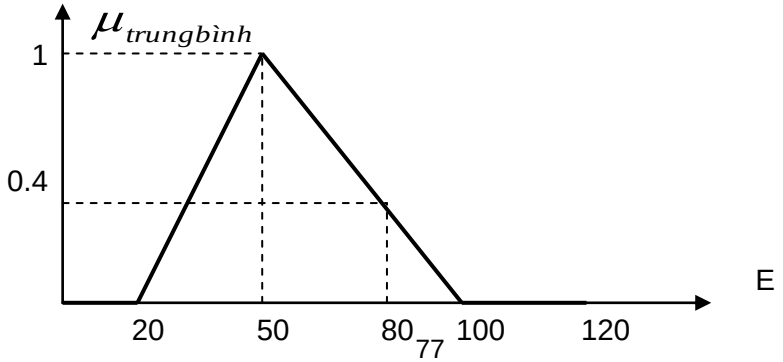
Như vậy tốc độ dưới 20km/h được coi là không nhanh. Tốc độ càng cao thì độ thuộc của nó vào tập F càng cao. Khi tốc độ là 100km/h trở lên thì độ thuộc là 1.



Nhóm hàm hình chuông

Nhóm hàm này có đồ thị dạng hình chuông, bao gồm dạng hàm tam giác, hàm hình thang, gauss.

Xét ví dụ cũng với tập vũ trụ E ở trên, xét tập mờ F=Tốc độ trung bình xác định bởi hàm thuộc $\mu_{trungbình} = \begin{cases} 0 & khi \quad x \le 20 \vee x \ge 100 \\ (x-20)/30 & khi \quad 20 \le x \le 50 \\ (100-x)/50 & khi \quad 50 \le x \le 100 \end{cases}$



Các khái niệm liên quan

Giả sử A là tập mờ trên vũ trụ U, có hàm thuộc μ_A thì ta có các khái niệm sau:

- ***Giá đỡ*** của A, ký hiệu supp(A) là một tập rõ bao gồm tất cả các phần tử $x \in U$ sao cho $\mu_A(x) > 0$
- ***Nhân*** của A là một tập rõ bao gồm tất cả các phần tử $x \in U$ sao cho $\mu_A(x) = 1$
- ***Biên*** của A là một tập rõ bao gồm tất cả các phần tử $x \in U$ sao cho $0 < \mu_A(x) < 1$
- ***Độ cao*** của A, ký hiệu height(A) là cận trên đúng của $\mu_A(x)$. $\text{height}(A) = \sup_{x \in U} \mu_A(x)$
- Tập mờ A được gọi là ***tập mờ chuẩn tắc*** (normal fuzzy set) nếu height(A)=1. Tức là tập mờ chuẩn tắc có nhân khác rỗng.

10.2 Các phép toán trên tập mờ

Giả sử A và B là các tập mờ trên vũ trụ U thì ta có các định nghĩa sau:

<u>Quan hệ bao hàm</u>

A được gọi là bằng B khi và chỉ khi $\forall x \in U,\ \mu_A(x) = \mu_B(x)$.

A được gọi là tập con của B, ký hiệu $A \subseteq B$ khi và chỉ khi $\forall x \in U,\ \mu_A(x) \le \mu_B(x)$

<u>Phần bù</u>

Phần bù mờ của tập mờ A là tập mờ $\overline{A}$ với hàm thuộc được xác định bởi:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \qquad (1)$$

<u>Hợp</u>

Hợp của tập mờ A và tập mờ B là tập mờ $A \cup B$ với hàm thuộc được xác định bởi:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \qquad (2)$$

<u>Giao</u>

Giao của tập mờ A và tập mờ B là tập mờ $A \cap B$ với hàm thuộc được xác định bởi:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \qquad (3)$$

<u>Tích đề các</u>

Giả sử $A_1, A_2, \ldots, A_n$ là các tập mờ trên các vũ trụ $U_1, U_2, \ldots, U_n$ tương ứng. Tích đề-các của $A_1, A_2, \ldots, A_n$ là tập mờ $A = A_1 \times A_2 \times \ldots \times A_n$ trên không gian tích $U_1 \times U_2 \times \ldots \times U_n$ với hàm thuộc được xác định bởi:

$$\mu_A(x_1, x_2, \ldots, x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n))$$

$$x_1 \in U_1,\ x_2 \in U_2, \ldots, x_n \in U_n \qquad (4)$$

Phép chiếu

Giả sử A là tập mờ trên không gian tích $U_1 \times U_2$. Hình chiếu của A trên U_1 là tập mờ A_1 với hàm thuộc được xác định bởi:

$$\mu_{A_1}(x) = \max_{y \in U_2} \mu_A(x, y) \qquad (5)$$

Định nghĩa trên có thể mở rộng cho trường hợp không gian tích n chiều

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 với hàm thuộc được xác định bởi:

$$\mu_A(x, y) = \mu_{A_1}(x) \qquad (6)$$

Các phép toán mở rộng

Ngoài các phép toán chuẩn: phần bù, hợp, giao được đề cập ở trên còn có nhiều cách mở rộng phép toán trên tập mờ khác có tính tổng quát hóa cao hơn.

Phần bù mờ

Giả sử xét hàm C:[0,1] -> [0,1] cho bởi công thức C(a) = 1 – a, $\forall$ a $\in$ [0,1]. Khi đó hàm thuộc của phần bù chuẩn trở thành $\mu_{\bar{A}}(x) = C(\mu_A(x))$. Nếu tổng quát hoá tính chất của hàm C thì ta sẽ có tổng quát hoá định nghĩa của phần bù mờ. Từ đó ta có định nghĩa:

Phần bù mờ của tập mờ A là tập mờ $\bar{A}$ với hàm thuộc được xác định bởi $\mu_{\bar{A}}(x) = C(\mu_A(x))$, trong đó C là một hàm số thoả các điều kiện sau:

i. Tiên đề C1 (điều kiện biên): C(0) = 1, C(1) = 0

ii. Tiên đề C2 (đơn điệu giảm): $\forall$ a, b $\in$ [0,1]. Nếu a < b thì C(a) $\geq$ C(b)

Hàm C thoả các điều kiện trên được gọi là **hàm phần bù**.

Ta thấy rằng hàm thuộc của phần bù chuẩn là một hàm đặc biệt trong họ các hàm phần bù.

Ví dụ:

Hàm phần bù Sugeno $C(a) = \frac{1-a}{1+\lambda a}$ trong đó λ là tham số thoả λ > -1. Hàm bù chuẩn là trường hợp đặc biệt của hàm Sugeno khi λ = 0.

Hàm phần bù Yager $C(a) = (1-a^w)^{\frac{1}{w}}$ trong đó w là tham số thoả w > 0. Hàm bù chuẩn là trường hợp đặc biệt của hàm Yager khi w = 1.

Hợp mờ – các phép toán S-norm

Phép toán max trong công thức hàm hợp mờ chuẩn có thể được tổng quát hoá thành các hàm S-norm:

Một hàm số S: [0,1]x[0,1] -> [0,1] được gọi là một S-norm nếu thoả các điều kiện sau:

i. Tiên đề S1 (điều kiện biên): S(0,a) = a, $\forall$ a$\in$ [0,1]

ii. Tiên đề S2 (giao hoán): S(a,b) = S(b,a), $\forall$ a,b$\in$ [0,1]

iii. Tiên đề S3 (kết hợp): S(S(a,b),c) = S(a,S(b,c)), $\forall$ a,b,c$\in$ [0,1]

iv. Tiên đề S4 (đơn điệu tăng): Nếu a$\leq$ b và c$\leq$ d thì S(a,c)$\leq$ S(b,d), $\forall$ a,b,c,d$\in$ [0,1]

S-norm còn được gọi là co-norm hoặc T-đối chuẩn.

Hợp của tập mờ A và tập mờ B là tập mờ A$\cup$B với hàm thuộc được xác định bởi:

$$\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x))$$

trong đó S là một S-norm

Ngoài hàm max, ta có một số hàm S-norm quan trọng sau đây:

- Tổng Drastic :

$$a\bar{\vee}b = \begin{cases} a & if \quad b = 0 \\ b & if \quad a = 0 \\ 1 & if \quad a > 0, b > 0 \end{cases}$$

- Tổng chặn: $a \oplus b = \min(1, a + b)$

- Tổng đại số: $a \hat{+} b = a + b - ab$

- Phép hợp Yager:

$$S_w(a,b) = \min\left[1, (a^w + b^w)^{\frac{1}{w}}\right]$$

Trong đó w là tham số thoả w > 0

<u>Giao mờ – các phép toán T-norm</u>

Ta có định nghĩa hàm T-norm là tổng quát hoá của hàm min:

Một hàm số T: [0,1]x[0,1] -> [0,1] được gọi là một T-norm nếu thoả các điều kiện:

i. Tiên đề T1 (điều kiện biên): T(1,a) = a, $\forall$ a$\in$ [0,1]

ii. Tiên đề T2 (giao hoán): T(a,b) = T(b,a), $\forall$ a,b$\in$ [0,1]

iii. Tiên đề T3 (kết hợp): T(T(a,b),c) = T(a,T(b,c)), $\forall$ a,b,c$\in$ [0,1]

iv. Tiên đề T4 (đơn điệu tăng): Nếu a$\leq$ b và c$\leq$ d thì T(a,c)$\leq$ T(b,d), $\forall$ a,b,c,d$\in$ [0,1]

T-norm còn được gọi là T-chuẩn hoặc chuẩn tam giác.

Giao của tập mờ A và tập mờ B là tập mờ A$\cap$B với hàm thuộc được xác định như sau:

$$\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x))$$

Trong đó T là một T-norm.

Ngoài hàm min, ta có một số hàm T-norm quan trọng sau đây:

- Tích Drastic:

$$a \underline{\wedge} b = \begin{cases} a & if \quad b = 1 \\ b & if \quad a = 1 \\ 0 & if \quad a < 1, b < 1 \end{cases}$$

- Tích chặn: $a \otimes b = \max(0, a + b - 1)$
- Tích đại số: $a.b = ab$
- Phép giao Yager:

$$T_w(a,b) = 1 - \min\left[1, ((1-a)^w + (1-b)^w)^{\frac{1}{w}}\right]$$

Trong đó w là tham số thoả w>0

Định lý: Với mọi T-norm bất kỳ T và S-norm bất kỳ S ta có:

$$a \underline{\wedge} b \le T(a,b) \le \min(a,b) \le \max(a,b) \le S(a,b) \le a \overline{\vee} b$$

Tích đề-các mờ

Tích đề-các của 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 không gian tích $U_1 \times U_2 \times \ldots \times U_n$ với hàm thuộc được xác định như sau:

$$\mu_A(x_1, x_2, \ldots, x_n) = \mu_{A_1}(x) \mathbf{T} \mu_{A_2}(x) \mathbf{T} \ldots \mathbf{T} \mu_{A_n}(x)$$

$$x_1 \in U_1, x_2 \in U_2, \ldots, x_n \in U_n$$

Trong đó **T** là một T-norm bất kỳ.

Ta thấy đây là định nghĩa mở rộng cho tích đề-các chuẩn khi thay thế hàm min bằng một T-norm bất kỳ.

Quan hệ mờ

Cho U và V là các vũ trụ. Khi đó một quan hệ mờ hai ngôi R giữa U và V là một tập mờ trong tích đề-các UxV. Như vậy ta có thể xác định hàm thuộc cho quan hệ mờ theo cách tính hàm thuộc cho tích đề-các mờ.

Khi U = V ta nói R là quan hệ trên U.

Tổng quát một quan hệ mờ R giữa các tập U_1, U_2, …, U_n là tập mờ $A = A_1 \times A_2 \times \ldots \times A_n$ trên không gian tích $U_1 \times U_2 \times \ldots \times U_n$. Trong đó $A_i \subseteq U_i$, i = 1..n

Hợp của các quan hệ mờ

Hợp của quan hệ mờ R từ U đến V và quan hệ mờ Z từ V đến W là quan hệ mờ RoZ từ U đến W có hàm thuộc xác định bởi

$$\mu_{RoS}(u,w) = \max_{v \in V} \{ \mathbf{T}(\mu_R(u,v), \mu_Z(v,w)) \}$$

Trong T là một T-norm bất kỳ.

Các hàm thuộc quan trọng sau được dùng rộng rãi để xác định hợp của các quan hệ mờ :

- Hàm hợp max-min:

$$\mu_{RoS}(u,w) = \max_{v \in V} \{ \mathbf{min}(\mu_R(u,v), \mu_Z(v,w)) \}$$

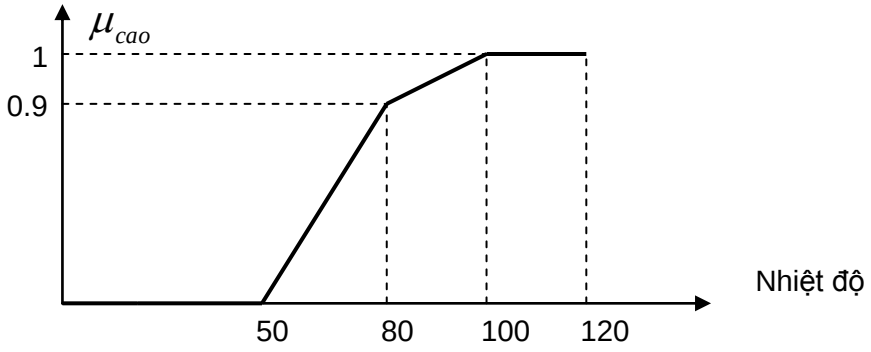
- Hàm hợp max-tích (hay max-prod):

$$\mu_{RoS}(u,w) = \max_{v \in V} \{ \mu_R(u,v) \cdot \mu_Z(v,w) \}$$

10.3 Biểu diễn tri thức bằng logic mờ

Biến ngôn ngữ

Ta xét một biến nhận giá trị trong một miền giá trị nào đó , chẳng hạn "nhiệt độ" có thể nhận giá trị số là $1^\circ C$, $2^\circ C$,… là các giá trị chính xác. Khi đó, với một giá trị cụ thể gán vào biến sẽ giúp chúng ta xác định được tính chất, quy mô của biến. Ngoài ra chúng ta còn biết được những thông tin khác liên quan đến biến đó. Ví dụ chúng ta hiểu là không nên chạm tay trần vào vật có "nhiệt độ" là $80^\circ C$ trở lên. Nhưng trong thực tế thì chúng ta thường nói "không nên chạm vào vật có nhiệt độ cao" chứ ít khi nói "không nên chạm vào vật có nhiệt độ là $80^\circ C$ trở lên". Thực tế là lời khuyên đầu thì có ích hơn bởi vì nếu nhận được lời khuyên sau thì ta dễ bị ngộ nhận là có thể chạm tay vào vật có nhiệt độ là $79^\circ C$ trong khi đó vật có nhiệt độ $80^\circ C$ trở lên thì không. Nhưng vấn đề đặt ra là nếu nghe theo lời khuyên đầu thì ta có thể xác định rõ là nhiệt độ bằng bao nhiêu thì có thể chạm tay vào? Câu trả lời là tuỳ vào ý kiến của từng người. Với nhiệt độ là $60^\circ C$ thì có người cho là cao trong khi người khác thì không. Tuy các ý kiến là khác nhau nhưng có một điều chắc chắn là khi giá trị của biến nhiệt độ càng tăng thì càng dễ dàng được chấp nhận là "cao". Như vậy nếu xét hàm μ_{cao} nhận biến nhiệt độ và trả về tỷ lệ ý kiến đồng ý là "cao" thì μ_{cao} sẽ là hàm thuộc của tập mờ "nhiệt độ cao" trên vũ trụ "nhiệt độ"



Biến nhiệt độ có thể nhận giá trị "cao" là một giá trị của ngôn ngữ tự nhiên nên nó được gọi là một biến ngôn ngữ (linguistic variable)

Khái niệm biến ngôn ngữ đã được Zadeh đưa ra năm 1973 như sau:

- Một biến ngôn ngữ được xác định bởi bộ (x, T, U, M) trong đó:
- x là tên biến. Ví dụ "nhiệt độ", "tốc độ", "độ ẩm",…
- T là tập các từ là các giá trị ngôn ngữ tự nhiên mà x có thể nhận. Ví dụ x là "tốc độ" thì T có thể là {"chậm", "trung bình", "nhanh"}
- U là miền các giá trị vật lý mà x có thể nhận Ví dụ x là "tốc độ" thì U có thể là {0km/h,1km/h, …150km/h}
- M là luật ngữ nghĩa, ứng mỗi từ trong T với một tập mờ At trong U

Từ định nghĩa trên chúng ta có thể nói rằng biến ngôn ngữ là biến có thể nhận giá trị là các tập mờ trên một vũ trụ 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" trong đó x là một đối tượng trong một vũ trụ U nào đó thoả tính chất P. Ví dụ "x là số chẵn" thì U là tập các số nguyên và P là tính chất chia hết cho 2. Như vậy ta có thể đồng nhất một mệnh đề phân tử "x là P" với một tập (rõ) A = $\{ x \in U \mid P(x) \}$.

Từ đó ta có: $P(x) = \lambda(x)$

Trong đó λ là hàm đặc trưng của tập A ($x \in A \Leftrightarrow \lambda(x) = 1$). Giá trị chân lý của P(x) chỉ nhận một trong hai giá trị 1 và 0 (true và false) tương ứng với sự kiện x thuộc A hoặc không

Trong trường hợp P là một tính chất mờ chẳng hạn như "số lớn" thì ta sẽ có một mệnh đề logic mờ phân tử. Khi đó tập hợp các phần tử trong vũ trụ U thoả P là một tập mờ B có hàm thuộc μ_B sao cho: $P(x) = \mu_B(x)$

Lúc này P(x) có thể nhận các giá trị tuỳ ý trong [0,1]. Và ta thấy có thể đồng nhất các hàm thuộc với các mệnh đề logic mờ.

Các phép toán mệnh đề mờ

Trong logic cổ điển, từ các mệnh đề phân tử và các phép toán $\wedge$(AND), $\vee$(OR), $\neg$(NOT) ta có thể lập nên các mệnh đề phức. Ta có:

$\neg P(x) = 1 - P(x)$

$P(x) \wedge Q(y) = \min(P(x), Q(y))$

$P(x) \vee Q(y) = \max(P(x), Q(y))$

$P(x) \Rightarrow Q(y) = \neg P(x) \vee Q(y) = \max(1-P(x), Q(y))$

$P(x) \Rightarrow Q(y) = \neg P(x) \vee (P(x) \wedge Q(y)) = \max(1-P(x), \min(P(x), Q(y)))$

Như vậy, ta sẽ có mở rộng một cách tự nhiên từ logic cổ điển sang logic mờ với quy tắc tổng quát hoá dùng hàm bù mờ cho phép phủ định, hàm T-norm cho phép giao và S-norm cho phép hợp. Sự mở rộng này dựa trên sự tương quan giữa mệnh đề logic mờ với hàm mờ và các phép toán trên tập mờ. Ta có:

$\neg \mu_A(x) = \mathbf{C}(\mu_A(x))$

$\mu_A(x) \wedge \mu_B(y) = \mathbf{T}(\mu_A(x), \mu_B(y))$

$\mu_A(x) \vee \mu_B(y) = \mathbf{S}(\mu_A(x), \mu_B(y))$

$\mu_A(x) => \mu_B(y) = \mathbf{S}(\mathbf{C}(\mu_A(x)), \mu_B(y))$ **(1)**

$\mu_A(x) => \mu_B(y) = \mathbf{S}(\mathbf{C}(\mu_A(x)), \mathbf{T}(\mu_A(x), \mu_B(y)))$ **(2)**

Trong đó C là hàm bù mờ (hay phủ định mờ), T là hàm T-norm, S là hàm S-norm. Các hàm này đã trình bày trong phần phép toán trên tập mờ.

Phép toán kéo theo mờ – luật if-then mờ thông dụng

Các phép toán kéo theo có vai trò quan trọng trong logic mờ. Chúng tạo nên các luật mờ để thực hiện các phép suy diễn trong tất cả các hệ mờ. Do một mệnh đề mờ tương ứng với một tập mờ nên ta có thể dùng hàm thuộc thay cho các mệnh đề.

Sau đây là một số phép kéo theo quan trọng được sử dụng rộng rãi:

Phép kéo theo Dienes – Rescher

Nếu áp dụng công thức (1) với S-norm max và C là hàm bù chuẩn cho ta có phép kéo theo Dienes – Rescher

$$\mu_A(x) => \mu_B(y) = \max(1-\mu_A(x), \mu_B(y))$$

Phép kéo theo Lukasiewicz

Nếu áp dụng công thức (1) với S-norm là hàm hợp Yager với w=1 và C là hàm bù chuẩn cho ta có phép kéo theo Lukasiewicz:

$$\mu_A(x) => \mu_B(y) = \min(1, 1-\mu_A(x)+\mu_B(y))$$

Phép kéo theo Zadeh

Nếu áp dụng công thức (2) với S-norm là max, T-norm min hoặc tích và C là hàm bù chuẩn cho ta có phép kéo theo Zadeh:

$$\mu_A(x) => \mu_B(y) = \max(1-\mu_A(x), \min(\mu_A(x), \mu_B(y)))$$ (a)

$$\mu_A(x) => \mu_B(y) = \max(1-\mu_A(x), \mu_A(x).\mu_B(y))$$ (b)

Kéo theo Mamdani

Ta có thể coi mệnh đề $\mu_A(x) => \mu_B(y)$ xác định một quan hệ 2 ngôi $R \subseteq UxV$. Trong đó U là không gian nền của x (vũ trụ chứa x), V là không gian nền của y (vũ trụ chứa y). Khi đó giá trị chân lý của mệnh đề $\mu_A(x) => \mu_B(y)$ là giá trị hàm thuộc của cặp (x,y) vào R. Theo công thức xác định hàm thuộc của quan hệ mờ ta có

$$\mu_A(x) => \mu_B(y) = T(\mu_A(x), \mu_B(y))$$

Trong đó T là một T-norm. Khi chọn T là min hoặc tích ta có các phép kéo theo Mamdani:

$$\mu_A(x) => \mu_B(y) = \min(\mu_A(x), \mu_B(y))$$ (a)

$\mu_A(x) => \mu_B(y) = \mu_A(x).\mu_B(y)$ (b)

Luật modus-ponens tổng quát

Tương tự logic cổ điển, trong logic mờ cũng có luật modus-ponens như sau:

GT1 (luật) : if "x là A" then "y là B"

GT2 (sự kiện) : "x là A'"

KL : "y là B'"

Trong đó A, B, A', B' là các biến ngôn ngữ (có nghĩa là các tập mờ).

Công thức tính kết luận của luật modus-ponens như sau:

$$\mu_{B'}(y) = \sup_x T(\mu_R(x,y), \mu_{A'}(x)) \quad (*)$$

Trong đó T là một hàm T-norm và R là quan hệ hai ngôi xác định bởi phép kéo theo. Cách tính $\mu_R(x,y)$, chính là cách tính giá trị chân lý của phép kéo theo trình bày ở phần trước. Như vậy tuỳ theo cách chọn cách tính luật kéo theo khác nhau mà ta có cách tính kết quả của luật modus-ponens khác nhau.

Ví dụ: Giả sử quan hệ giữa nhiệt độ và áp suất cho bởi luật sau:

Nếu nhiệt độ là cao thì áp suất là lớn.

Nhiệt độ nhận các giá trị trong U = {30, 35, 40, 45}

Ap suất nhận các giá trị trong V = {50,55,60,65}

Ta có các tập mờ xác định bởi các biến ngôn ngữ nhiệt độ và áp suất như sau:

$$A = \text{"nhiệt độ cao"} = \frac{0}{30} + \frac{0.3}{35} + \frac{0.9}{40} + \frac{1}{45}$$

$$B = \text{"áp suất lớn"} = \frac{0}{50} + \frac{0.5}{55} + \frac{1}{60} + \frac{1}{65}$$

Áp dụng luật kéo theo Mamdani tích ta có quan hệ mờ sau (giá trị dòng i, cột j là giá trị hàm thuộc của cặp nhiệt độ i và áp suất j vào quan hệ)

$$R = \begin{matrix} \left[\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0.15 & 0.3 & 0.3 \\ 0 & 0.45 & 0.9 & 0.9 \\ 0 & 0.5 & 1 & 1 \end{matrix}\right] & \begin{matrix} 30 \\ 35 \\ 40 \\ 45 \end{matrix} \\ \begin{matrix} 50 & 55 & 60 & 65 \end{matrix} & \end{matrix}$$

Bây giờ, giả sử ta biết sự kiện "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 (*) ta suy ra $B' = \frac{0}{50} + \frac{0.45}{55} + \frac{0.8}{60} + \frac{0.8}{65}$