TRƯỜNG ĐẠI HỌC QUY NHƠN
KHOA CÔNG NGHỆ THÔNG TIN

TS. LÊ XUÂN VINH

BÀI GIẢNG

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Quy Nhơn – 2016

Nội dung chương trình

Thời gian: 45 tiết

Nội dung: 3 phần

Phần 1: Giải quyết vấn đề bằng tìm kiếm (15t)

Phần 2: Biểu diễn tri thức và lập luận (15t)

Phần 3: Logic mờ và ứng dụng (15t)

Ngành Công nghệ thông tin: học 45 tiết, 3 phần

Ngành Sư phạm Tin học: học 30 tiết, 2 phần 1 và 2

Tài liệu tham khảo

[1] Trí tuệ nhân tạo – Đinh Mạnh Tường, NXB Đại học Quốc gia Hà Nội, 2008.

[2] Trí tuệ nhân tạo – George F. Luger, NXB Thống kê, 2000.

[3] Trí tuệ nhân tạo – Nguyễn Thanh Thủy, NXB Khoa học Kỹ thuật, 1998

[4] Artificial Intelligence, A Modern Approach - Stuart J. Russell, Prentice Hall, 2010

GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO

1. Một số ứng dụng của trí tuệ nhân tạo

Những năm gần đây chúng ta thường nghe nói nhiều về máy tính thế hệ 5, hệ chuyên gia, lập trình Prolog, logic mờ, mạng nơron nhân tạo, giải thuật di truyền,…Đây là một số thuật ngữ trong một ngành mới của khoa học máy tính. Đó là: Trí tuệ nhân tạo (TTNT). Để hình dung TTNT giải quyết những vấn đề gì, chúng ta hãy xem những ứng dụng với những đòi hỏi cụ thể của nó.

Trò chơi: cờ carô, cờ vua, các ô số,…mỗi một bước đi trên bàn cờ là một quyết định trong số rất nhiều khả năng có thể lựa chọn. Tất cả các khả năng sẽ sinh ra một không gian quá lớn và phức tạp. Sẽ rất khó khăn nếu như sòng phẳng xét hết tất cả các khả năng. Vì lý do thời gian, một người đánh cờ chỉ có thể cảm nhận khả năng tốt trong lựa chọn. Chương trình thông minh phải có khả năng như vậy. Chiến lược lựa chọn mang tính cảm nhận nhưng có cơ sở sẽ được gọi là heuristic, nó không chắc chắn mang lại kết quả nhưng nhiều khả năng mang đến thành công, tuy nhiên vẫn có thể hàm chứa sự rủi ro đẫn đến thất bại. Năm 1953, Samuel đã viết chương trình chơi cờ gây ấn tượng lớn khi nó được công chiếu trên tivi. Chương trình của Ông có khả năng học và khi được huấn luyện có khả năng chơi hay hơn người viết ra nó. Chương trình đánh cờ cho máy tính Deep-Blue (1997) cũng là một ứng dụng của TTNT vào trò chơi.

Hệ chuyên gia: Một chuyên gia phải có nhiều tri thức chuyên môn và kỹ năng sử dụng những tri thức đó để giải quyết vấn đề. Một hệ chương trình thay thế cho chuyên gia được gọi là hệ chuyên gia. Nó bao gồm cơ sở tri thức và các quy tắc suy luận. Hệ chuyên gia đang được ứng dụng rộng rãi trong các lĩnh vực y tế, giáo dục, thiết kế, kinh doanh, khoa học,…Những hệ chuyên gia nổi tiếng như DENDRAL (Stanford, 1960) dùng để phỏng đoán cấu trúc các phân tử hữu cơ từ công thức hóa học của chúng; MYCIN (Stanford, 1970) chẩn đoán và kê đơn điều trị cho bệnh viêm màng não và nhiễm trùng máu; PROSPECTOR (MIT, 1979) xác định vị trí, loại quặng mỏ dựa trên thông tin địa lý. Đòi hỏi cơ bản của mọi hệ chuyên gia là biểu diễn tri thức ngôn ngữ như thế nào và tiếp cận cách suy luận của con người ra sao. Cho đến nay, đó vẫn là những vấn đề nan giải cần giải quyết.

Lập kế hoạch và robot: Lập kế hoạch là xác định một dãy thao tác để đạt được mục đích đặt ra. Đối với con người đây đã là một yêu cầu phức tạp, tuy nhiên có thể giải quyết được do con người có khả năng phán đoán, suy luận. Khi trang bị chương trình như vậy cho robot chúng ta gặp phải những khó khăn: biểu diễn tri thức về không gian, môi trường tác động luôn biến động, số lượng các chuỗi thao tác là rất lớn, thông tin không đầy đủ, thao tác sửa chữa hành vi khi gặp bất lợi,…Các robot của Nhật Bản là những minh chứng cho sự thành công trong việc giải quyết những vấn đề trên.

Điều khiển mờ: Tích hợp các thiết bị điều khiển mờ tự động vào các sản phẩm công nghệ phục vụ đời sống bắt đầu từ những năm 1990 tại Nhật Bản. Điển hình là các sản phẩm như máy giặt, máy điều hòa nhiệt độ của Toshiba; máy ảnh, máy quay phim kỹ thuật số của Canon; hướng dẫn lùi xe tự động của Nissan, Mitshubisi và các ứng dụng trong điều khiển tàu điện không người lái, trong các dây chuyền công nghiệp, sản xuất xi măng,…Đặc trưng của kỹ thuật này dựa trên lý thuyết mờ của L. A. Zadeh (1965) với quan điểm mờ hóa đầu vào các tác động của môi trường nhằm đạt được kết quả liên tục tốt nhất, phù hợp với quan điểm sử dụng ngôn ngữ để mô tả cho dữ liệu của con người.

Hiểu và mô hình hóa ngữ nghĩa ngôn ngữ tự nhiên: Khi đọc cùng một bài viết, mỗi người hiểu một mức độ khác nhau. Điều này phụ thuộc vào tri thức, khả năng nắm bắt, suy luận, xử lý linh hoạt vấn đề, ngữ cảnh của người đọc. Những ứng dụng như dịch máy phải phát triển các kỹ thuật để cấu trúc hóa ý nghĩa ngữ nghĩa. Việc hiểu ngôn ngữ tổng quát theo cách con người vẫn nằm ngoài tầm tay của chúng ta thậm chí chỉ là mức phương pháp luận.

2. Khái niệm về trí tuệ nhân tạo

Trí tuệ nhân tạo (Artificial Intelligence – AI) là thuật ngữ do McCathy đưa ra tại hội thảo Dartmount năm 1956 dùng để chỉ cho một ngành khoa học mới trong lĩnh vực khoa học máy tính. Nghiên cứu những vấn đề liên quan đến tư duy của con người, TTNT kế thừa nhiều ý tưởng, quan điểm, kỹ thuật từ nhiều ngành khoa học khác như Triết học, Toán học, Tâm lý học,…Triết học được nhắc đến như nguồn gốc xa xưa của TTNT khi mà qui tắc suy diễn "modus ponens" (tam đoạn luận) được sử dụng trong suy luận hình thức ngày nay đã được Aristotle đưa ra từ vài nghìn năm trước và nhiều công trình nghiên cứu về tư duy của nhiều nhà triết

học khác. Descartes cũng là nhân vật trung tâm trong sự phát triển các khái niệm hiện đại về tư duy và tinh thần với câu nói nổi tiếng "tôi tư duy nghĩa là tôi tồn tại". Các ngành logic, lý thuyết đồ thị, xác suất của Toán học đóng góp rất nhiều cho TTNT. Logic kinh điển Boole, logic vị từ Frege là những cơ sở quan trọng để biểu diễn tri thức. Lý thuyết đồ thị cung cấp công cụ để mô hình một vấn đề, tìm kiếm lời giải, phân tích tính chính xác, tính hiệu quả của các chiến lược tìm kiếm lời giải.

Khác với các ngành khoa học khác nghiên cứu về trí tuệ, TTNT nghiên cứu và tạo ra những thực thể có mang tính trí tuệ và ứng dụng trong sản xuất các thiết bị phục vụ cho đời sống, một xu thế tất yếu của thời đại công nghệ tri thức. Vì vậy, có một số định nghĩa về TTNT được chấp nhận nhiều hơn như sau:

+ TTNT là sự nghiên cứu, thiết kế các chương trình máy tính ứng xử một cách thông minh.

+ TTNT là sự nghiên cứu, thiết kế các tác nhân thông minh (Intelligent Agents).

Các định nghĩa về TTNT tập trung quanh hai quan điểm: suy nghĩ, hành động như con người (think, act like human): quan tâm đến yếu tố kinh nghiệm; và suy nghĩ và hành động hợp lý (think, act rationally) quan tâm đến yếu tố logic của vấn đề.

Thế nào là máy tính hay một chương trình máy tính thông minh? Trắc nghiệm Turing đưa ra dựa trên việc so sánh với khả năng của con người, một đối tượng được coi là có hành vi thông minh và chuẩn mực nhất về trí tuệ. Trắc nghiệm này có người thẩm vấn cách ly với người trả lời thẩm vấn và máy tính. Nếu người thẩm vấn không phân biệt được câu trả lời của máy tính và người thì máy tính đó được coi là thông minh.

Trắc nghiệm như trên không dễ thực hiện và để hiểu đánh giá sự thông minh của chương trình chúng ta tìm hiểu khái niệm tác nhân thông minh (intelligent agent).

Tác nhân là bất cứ cái gì có khả năng nhận thức và tác động phản ứng lại đối với môi trường. Ví dụ robot tiếp nhận các trạng thái của môi trường thông qua các bộ cảm nhận, hành động theo quyết định điều khiển.

Tác nhân được gọi là thông minh nếu nó phản ứng lại môi trường để đạt được mục tiêu hoặc ít nhất là có cơ sở sẽ đạt được mục tiêu đề ra. Một tác nhân thông minh phải có các khả năng sau đây:

- Hiểu ngôn ngữ tự nhiên: tiếng Anh hay một ngôn ngữ nào đó.

- Có khả năng biểu diễn tri thức: thu thập, sử dụng tri thức.

- Lập luận tự động: xử lý tri thức và đưa ra kết luận.

- Học máy: thích nghi với hoàn cảnh và khả năng ngoại suy.

3. Những đặc điểm của công nghệ xử lý thông tin dựa trên TTNT

Với những yêu cầu mới đối với những ứng dụng của TTNT, bản thân công nghệ xử lý thông tin phải có những đặc điểm sau:

Phải có những công cụ hình thức hóa như các mô hình logic ngôn ngữ, logic mờ, mạng ngữ nghĩa,…để biểu diễn tri thức trên máy tính và quá trình giải quyết bài toán được tiến hành hữu hiệu hơn.

Phải có tính mềm dẻo, thích nghi với những tình huống mới nảy sinh, chẳng hạn như các hệ chuyên gia. Các cơ chế suy diễn cũng phải mềm dẻo được áp dụng tùy tình huống, chưa chắc cơ chế nào tốt hơn cơ chế nào.

Phải được trang bị tri thức heuristic do chuyên gia con người cung cấp khác với chương trình thông thường chỉ cần dựa trên thuật toán là đủ.

Việc xây dựng các chương trình TTNT phải có sự tham gia của các kỹ sư xử lý tri thức: phân tích phương pháp giải quyết bài toán theo chuyên gia con người, diễn đạt tri thức và cơ chế suy diễn để dễ mã hóa trong máy tính.

Nhìn chung, xử lý thông tin dựa trên TTNT có những đặc điểm riêng khác với công nghệ truyền thống trước đây:

Lập trình truyền thống	**Lập trình theo TTNT**
Xử lý dữ liệu	Xử lý tri thức – dữ liệu thường mang tính định tính hơn là định lượng
Dữ liệu được đánh địa chỉ	Tri thức được biểu diễn theo một số cấu

	trúc nhất định
Xử lý theo thuật toán	Xử lý theo các giải thuật heuricstics và cơ chế lập luận
Xử lý tuần tự, theo lô	Tính tương tác cao (robot)
Không cần giải thích việc thực hiện	Có thể giải thích lý do thực hiện

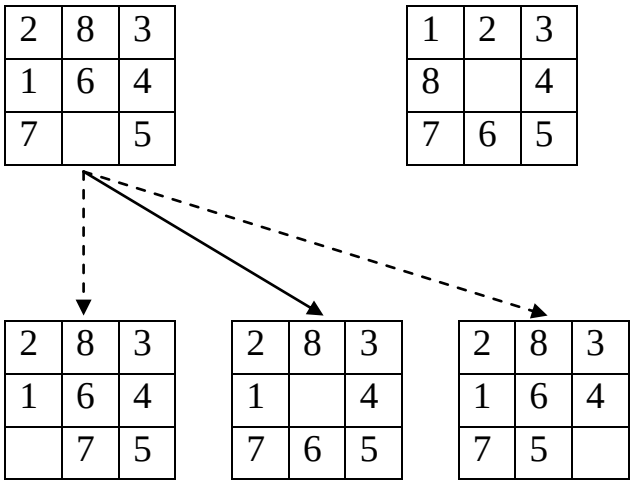
Chương 1. GIẢI QUYẾT VẤN ĐỀ BẰNG TÌM KIẾM

$1. CÁC CHIẾN LƯỢC TÌM KIẾM MÙ

1. Biểu diễn vấn đề trong không gian trạng thái.

Vấn đề là một bài toán nào đó cần giải quyết, chẳng hạn như một trò chơi, chứng minh một định lý,…Một lớp rộng các bài toán có thể giải quyết bằng cách biểu diễn bởi các trạng thái và các tóan tử (phép biến đổi trạng thái). Để dễ hình dung các khái niệm này, chúng ta tìm hiểu ví dụ sau.

Ví dụ 1. (Trò chơi 8 số). Cho hình vuông 3x3 và 8 số từ 1,..,8 xếp ngẫu nhiên vào các ô. Hãy di chuyển các số nhờ ô trống để thu được sự sắp xếp như hình bên phải



Sự sắp xếp các số tại mỗi thời điểm là một trạng thái. Hình bên trái là trạng thái ban đầu. Hình bên phải là trạng thái kết thúc hay trạng thái đích (goal). Trạng

thái đích của một bài toán có thể nhiều hơn một trạng thái. Một toán tử là một phép biến đổi hợp lệ chuyển từ trạng thái này sang trạng thái khác. Gọi u là một trạng thái tùy ý, tập hợp tất cả các trạng thái mở rộng từ trạng thái u được ký hiệu là S(u). Từ các trạng thái được mở rộng từ trạng thái ban đầu, bằng các tóan tử, ta tiếp tục mở rộng và cuối cùng thu được một không gian trạng thái (KGTT).

Như vậy, một không gian trạng thái là một bộ bốn (X, u_0, F, G), trong đó:

+ X là tập các trạng thái

+ u_0 là trạng thái bắt đầu

+ F là tập các toán tử, gồm các phép biến đổi.

+ G là tập trạng thái đích.

Không gian trạng thái có thể biểu diễn bằng đồ thị có hướng: mỗi đỉnh là một trạng thái, mỗi cung là một toán tử. Giải quyết được vấn đề hay tìm được nghiệm của bài toán nếu như ta tìm được đường đi từ trạng thái bắt đầu đến một trong các trạng thái đích.

Trong đồ thị của KGTT có thể xuất hiện chu trình gây khó khăn cho việc tìm kiếm, hạn chế các toán tử trong F có thể đưa đồ thị trở thành cây, một cấu trúc dễ tìm kiếm hơn. KGTT của trò chơi caro là một trường hợp như vậy.

Việc biểu diễn các trạng thái hợp lý là rất quan trọng, nó có thể làm cho KGTT nhỏ lại cũng như việc tìm kiếm trở nên đơn giản hơn.

Khi tìm kiếm lời giải, từ một trạng thái nào đó chưa phải là trạng thái đích, ta dựa theo toán tử sinh ra tập các trạng thái mới. Quá trình này gọi là quá trình mở rộng không gian trạng thái. Để được lời giải, ta phải liên tục chọn trạng thái mới, mở rộng, kiểm tra cho đến khi tìm được trạng thái đích hoặc không mở rộng được không gian trạng thái. Thông thường, tập các trạng thái được mở rộng sẽ có nhiều phần tử, việc chọn trạng thái nào để tiếp tục mở rộng được gọi là chiến lược tìm kiếm. Việc tìm kiếm lời giải cho bài toán thường sinh ra một cây gọi là cây tìm kiếm. Trong đó, mỗi nút trên cây cần giữ những thông tin sau đây:

+ Nội dung trạng thái mà nút hiện hành đang biểu diễn.

+ Nút cha đã sinh ra nó.

+ Toán tử đã được sử dụng để sinh ra nút hiện hành.

+ Độ sâu của nút.

+ Giá trị đường đi từ nút gốc đến nút hiện hành.

Để triển khai một chiến lược tìm kiếm nào đó, thông thường chúng ta phải lưu trữ các trạng thái chờ phát triển. Hàng đợi (queue) là một cấu trúc dữ liệu được ưu tiên lựa chọn với những thao tác sau:

+ Tạo hàng đợi với một số phần tử: Make-Queue(elements)

+ Kiểm tra hàng đợi rỗng: Empty(Queue)?

+ Lấy một phần tử ở đầu hàng đợi: Remove-Front(Queue)

+ Chèn một số phần tử vào hàng đợi:Insert(elements, Queue)

2. Các chiến lược tìm kiếm mù

Khi tìm kiếm trạng thái đích việc lựa chọn trạng thái nào để mở rộng được gọi là chiến lược tìm kiếm. Một chiến lược tìm kiếm được đánh giá dựa theo các tiêu chí sau đây:

+ Tính đầy đủ: chiến lược phải đảm bảo tìm được lời giải nếu có.

+ Độ phức tạp thời gian: mất thời gian bao lâu để tìm được lời giải.

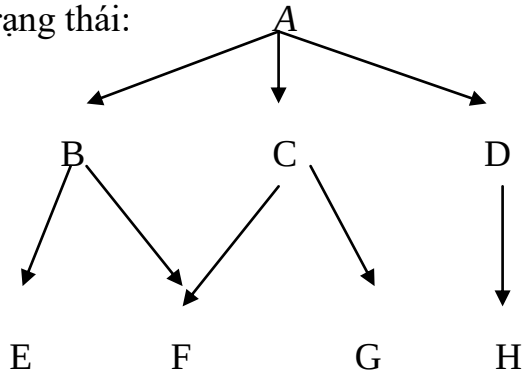
+ Độ phức tạp không gian: tốn bao nhiêu đơn vị bộ nhớ để tìm được lời giải.

+ Tính tối ưu: tốt hơn so với một số chiến lược khác hay không.

Trong tìm kiếm mù, trạng thái được chọn để phát triển chỉ đơn thuần dựa theo cấu trúc của KGTT mà không có thông tin hướng dẫn nào khác. Cho nên, nói chung tìm kiếm mù sẽ không hiệu quả. Tuy nhiên, nó được quan tâm bởi vì đây là cơ sở để chúng ta cải tiến và thu được những chiến lược hiệu quả hơn.

Tìm kiếm theo bề rộng (BFS)

Ví dụ. Không gian trạng thái:



Trạng thái được ưu tiên phát triển là trạng thái được sinh ra trước. Dùng danh sách open chứa các trạng thái sinh ra đang chờ phát triển, danh sách closed chứa các trạng thái đã được khảo sát. Thuật toán:

```
procedure bfs;
begin
    open:=[start]; closed:=[];
    while open<>[] do
        begin
            loại trạng thái ngoài cùng bên trái của open, gọi nó là u
            if (u là một đích) then thông báo kết quả, thoát
            else begin
                Đưa u vào closed
                Phát sinh các con v của u
                Loại các con vừa phát sinh nhưng đã có mặt trong open hoặc closed
                Đưa các con còn lại vào bên phải open (1)
                end
        end
    Thông báo thất bại
End
```

Các trạng thái con phát sinh nhờ các toán tử hợp lệ. Danh sách open bổ sung phần tử bên phải, lấy phần tử bên trái. Thuật toán khảo sát tất cả các nút ở độ sâu d sau đó mới đến mức d+1 nên chắc chắn tìm được nghiệm nếu có. Trong trường hợp bài tóan vô nghiệm và không gian trạng thái hữu hạn thuật toán sẽ dừng và thông báo vô nghiệm.

Để lưu được nghiệm, cần giữ nút cha của nút đã được khảo sát.

Đánh giá: Giả sử mỗi trạng thái trung bình sinh ra b trạng thái con (kề), b được gọi là nhân tố nhánh. Giả sử đường đi nghiệm có độ dài d. Trường hợp xấu nhất số nút phải khảo sát là:

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

Như vậy độ phức tạp thời gian là $O(b^d)$, độ phức tạp không gian cũng là $O(b^d)$.

Giả sử mỗi giây máy tính xử lý được 1000 trạng thái, và mỗi trạng thái cần 100 bytes để lưu trữ, thời gian và bộ nhớ cần thiết cho tìm kiếm rộng được tính trong bảng sau:

Độ sâu	Số nút	Thời gian	Bộ nhớ
2	111	0.1giây	11 kilobytes
4	11,111	11 giây	1 megabyte
6	10^6	18 phút	111 megabytes
8	10^8	31 giờ	11 gigabytes
10	10^{10}	128 ngày	1 terabyte
12	10^{12}	35 năm	111 terabyte
14	10^{14}	3500 năm	11.111 terabyte

Bảng 1. Thời gian, bộ nhớ cần thiết cho tìm kiếm theo bề rộng với b=10.

Tìm kiếm theo chiều sâu (DFS)

Chiến lược này mở rộng nút có độ sâu hơn trước các nút khác đang chờ xử lý. Chỉ khi nào không mở rộng được nữa thì mới quay lại nút ở độ sâu thấp hơn. Do đó, các nút mới được sinh ra chờ xử lý phải được bỏ bên trái của hàng đợi open (tại câu lệnh 1). DFS sử dụng bộ nhớ dùng cho open để lưu trữ ít hơn. Nếu nhân tố nhánh là b và chiến lược đang khảo sát nút ở độ sâu d thì nó chỉ lưu trữ b*d nút, trong khi đó BFS phải lưu trữ b^d nút. Ở độ sâu d=12, tính toán cho thấy DFS chỉ sử dụng 12KB trong khi BFS dùng đến 111TB. Độ phức tạp thời gian của DFS vẫn là $O(b^d)$ vì trong trường hợp xấu nhất các nút được khảo sát vẫn như BFS. Tuy nhiên, cơ hội để tìm thấy trạng thái đích cao hơn nếu nó nằm ở phần không gian trạng thái bên trái.

DFS có những hạn chế của nó. Nó bỏ qua cơ hội tìm thấy ngay trạng thái đích khi trạng thái này nằm gần gốc. Trường hợp không gian trạng thái được mở rộng vô hạn có thể nó không tìm được trạng thái đích. Hơn nữa trong trường hợp không gian trạng thái là đồ thị đường đi nghiệm nói chung không phải là đường đi ngắn nhất. Vì vậy, DFS bị đánh giá là chiến lược không đầy đủ mà cũng không tối ưu. Do đó nó không được khuyến khích sử dụng khi KGTT có độ sâu lớn hoặc vô hạn. Tuy nhiên, nó được đánh giá cao hơn tìm kiếm rộng BFS nhiều do giải quyết được

vấn đề không gian và cơ hội tiếp cận đích của bài toán là cao hơn. Một giải pháp có thể cho là dung hòa ưu thế giữa hai chiến lược tìm kiếm trên là giới hạn độ sâu theo chiến lược sau đây.

Tìm kiếm với độ sâu hạn chế

Vẫn sử dụng chiến lược tìm kiếm theo chiều sâu nhưng giới hạn độ sâu của đường đi nghiệm trên cây, tức là chúng ta sẽ không tiếp tục mở rộng KGTT nếu đã đến một độ sâu d cố định nào đó. Số d được gọi là bán kính trên KGTT. Chiến lược này sẽ tìm được nghiệm hay không phụ thuộc vào d. Nếu d được chọn thích hợp thì nó tìm được nghiệm, khi đó chiến lược là đầy đủ. Tuy nhiên nó không là chiến lược tối ưu. Tương tự như DFS, độ phức tạp thời gian là $O(b^d)$ và độ phức tạp không gian là O(bd).

Thuật toán cho chiến lược này có thêm tham số d

```
procedure DFS(d);
begin
    open:=[start]; closed:=[]; depth(start):=0;
    while open<>[] do
        begin
            loại trạng thái ngoài cùng bên trái của open, gọi nó là u
            if (u là một đích) then thông báo kết quả, thoát
            else begin
                Đưa u vào closed
                If  depth(u)<=d then begin
Phát sinh các con v của u
                Loại các con vừa phát sinh nhưng đã có mặt trong open hoặc closed
                Gán độ sâu cho các v bằng depth(u)+1
Đưa các con v còn lại vào bên trái open
            End;
End;
        End;
    Thông báo thất bại
End.
```

Vấn đề khó khăn là xác định độ sâu hạn chế d. Hầu hết các bài tóan chúng ta không biết trước d bằng bao nhiêu. Chiến lược sau đây giải quyết vấn đề này.

Tìm kiếm sâu dần

Từng bước tăng dần độ sâu hạn chế là cách làm trong chiến lược này. Lần lượt cho d bằng 1,2,…và tìm kiếm với độ sâu hạn chế ứng với d. Cách làm này tận dụng được lợi thế của hai chiến lược tìm kiếm rộng và tìm kiếm sâu ở chỗ vì nó tìm hết cây độ sâu d đến d+1 nên giống như tìm kiếm rộng chắc chắn tìm được nghiệm và do sử dụng tìm kiếm sâu trong cây hạn chế nên ít tốn kém bộ nhớ hơn. Như vậy, chiến lược này là đầy đủ và nếu KGTT là cây thì nó là chiến lược tối ưu. Tìm kiếm sâu dần sử dụng tìm kiếm sâu hạn chế như một thủ tục con

```
procedure DFS1;
begin
    for d:=0 to max do
        begin
            DFS(d);
            If thành công then exit
        End;
End;
```

Qúa trình tìm kiếm sâu sẽ lặp lại việc khảo sát các nút trong lớp có độ sâu bé hơn. Tưởng chừng điều này sẽ làm tăng độ phức tạp thời gian và không gian, tuy nhiên không như vậy. Chúng ta xét ví dụ sau đây. Giả sử nhân tố nhánh b=10 và d=max=5. Nhớ lại rằng số nút phải khảo sát trong tìm kiếm sâu với độ sâu d=5 là

$$1+b+b^2+\cdots+b^d = 1+10+100+1.000+10.000+100.000=111.111$$

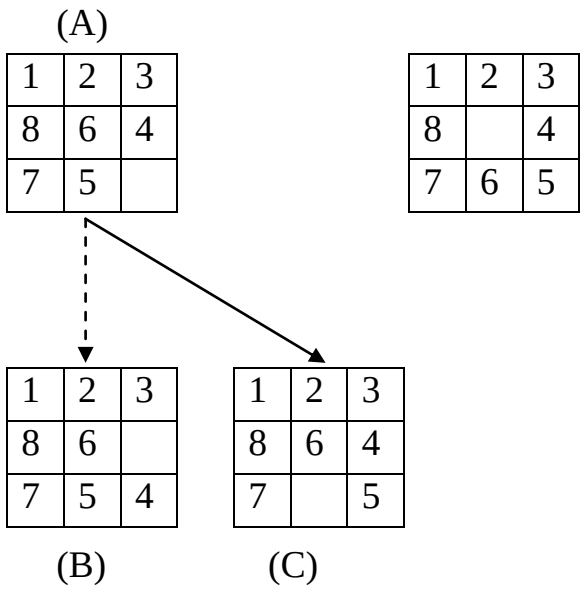
Khi tìm kiếm sâu dần, do vòng lặp For thực hiện max+1 lần nên số nút phải khảo sát là: $(d+1)1 + d*b+ (d-1)*b^2 +\ldots+2*b^{d-1}+1*b^d = 6 + 50 + 400 + 3000 + 20.000 +100.000= 123.456$. Số này phụ thuộc b nhưng nói chung không vượt quá 2 nhưng tính đầy đủ của chiến lược bảo đảm như tìm kiếm rộng. Thực tế số nút tăng lên nhưng độ phức tạp của thuật tóan vẫn là $O(b^d)$, trong khi đó độ phức tạp không gian vẫn là O(bd). Vì vậy, tìm kiếm sâu dần được đánh giá là chiến lược tìm kiếm thích hợp hơn khi KGTT lớn và không biết trước được độ sâu của trạng thái đích.

$2. CÁC CHIẾN LƯỢC TÌM KIẾM KINH NGHIỆM

Các chiến lược tìm kiếm mù kém hiệu quả và không thể áp dụng được trong nhiều trường hợp. Sử dụng thông tin của trạng thái kết hợp với nhận xét dựa theo kinh nghiệm để cải tiến cách lựa chọn để phát triển trạng thái là quan điểm chung của các chiến lược tìm kiếm kinh nghiệm.

1. Hàm đánh giá

Xét một số trạng thái của bài toán trò chơi 8 số. Nếu dùng các chiến lược tìm kiếm mù thì trạng thái (B) sẽ được lựa chọn để phát triển. Dễ thấy việc lựa chọn này là không hiệu quả, việc tìm đến đích sẽ lâu hơn là phát triển trạng thái (C). Nói cách khác, trạng thái (C) được phát triển có nhiều khả năng dẫn đến đích hơn trạng thái (B). Điều này phải được lượng hóa. Hàm đánh giá là một hàm ước lượng khả năng về đích của mỗi trạng thái. Để ý rằng ở đây chỉ là ước lượng vì nói chung chúng ta không thể tính toán chính xác khả năng này, bởi vì nếu tính được nghĩa là đã tìm được lời giải. Tuy nhiên, nhờ kinh nghiệm trong nhiều bài toán cụ thể chúng ta có thể ước lượng được giá trị này.



Ví dụ nếu đếm các số sai vị trí của một trạng thái so với trạng thái đích thì số này đối với (B) là 3 và đối với (C) là 1. Như vậy, số này càng nhỏ thì trạng thái đó càng giống với trạng thái đích, tức khả năng về đích cao. Chúng ta có thể dùng số này làm giá trị cho hàm đánh giá. Một cách tổng quát, hàm đánh giá là một hàm h như sau:

$$h: X \to R^+,$$

trong đó X là không gian trạng thái của bài toán. Dễ thấy, nếu g là trạng thái đích thì $h(g) = 0$. Mỗi $u \in X$, ta có một giá trị ước lượng là $h(u) \geq 0$. Hàm đánh giá còn được gọi là hàm heuristic.

Trong ví dụ trên, hàm h có thể xác định bởi $h(u) =$ số các số sai vị trí đối với đích.

Vì được xây dựng theo kinh nghiệm nên có nhiều cách để xây dựng hàm h.

1	5	3
8		4
7	6	2

(D)

1	4	3
8		2
7	6	5

(E)

Nếu tính như trên thì $h(D) = h(E) = 2$ vì đều có 2 số sai vị trí nhưng xét về mặt đường đi qua các ô ngang dọc để di chuyển về đúng vị trí thì (D) phải di chuyển đoạn đường xa hơn (E) nên có thể là (D) không tốt bằng (E). Theo nhận xét này, ta có thể xây dựng hàm

$h_1(u) =$ tổng khoảng cách phải di chuyển của các ô sai vị trí.

Ví dụ: $h_1(D) = 3+3 = 6$, $h_1(E) = 2+2=4$

Một ví dụ khác là lấy độ dài đường chim bay làm giá trị cho hàm đánh giá.

Giải quyết vấn đề bằng chiến lược tìm kiếm kinh nghiệm cần thực hiện các công việc sau:

+ Biểu diễn bài toán bằng một KGTT thích hợp;

+ Xây dựng hàm đánh giá;

+ Thiết kế chiến lược chọn trạng thái.

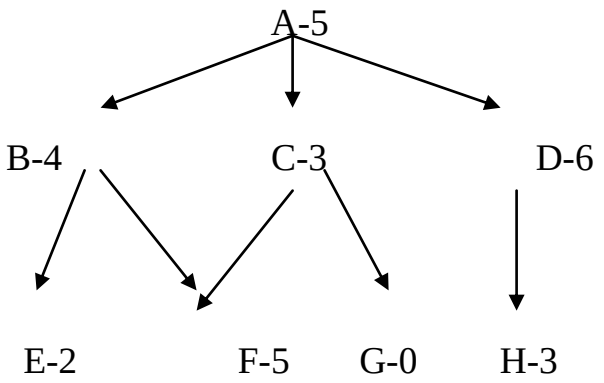
2. Các chiến lược tìm kiếm kinh nghiệm.

a) Tìm kiếm leo đồi – Hill Climbing Search (Pearl, 1984)

Chiến lược này chọn một trạng thái tốt hơn trạng thái đang khảo sát để phát triển. Trường hợp không tìm được trạng thái tốt hơn, thuật tóan phải dừng. Nếu chỉ chọn một trạng thái tốt hơn nào đó thì gọi là chiến lược leo đồi đơn giản, nếu trong số các trạng thái tốt hơn chọn trạng thái tốt nhất thì gọi là leo đồi dốc đứng. Nhắc

lại rằng, chúng ta sử dụng hàm đánh giá để biết trạng thái nào tốt hơn. Sử dụng chiến lược tìm kiếm sâu nhưng khác với tìm kiếm sâu, tìm kiếm leo đồi không lưu tất cả các trạng thái con được sinh ra mà chỉ lưu đúng một trạng thái được chọn nếu có.

Ví dụ. KGTT của bài toán như sau



Các trạng thái sẽ xuất hiện trong open và closed khi chiến lược leo đồi dốc đứng hoạt động như sau:

Bước	Open	Closed
1	A-5	Rỗng
2	C-3	A-5
3	G-0	C-3
		G-0

Chiến lược khá hiệu quả, tìm nhanh được trạng thái đích. Tuy nhiên, nó phụ thuộc nhiều vào hàm đánh giá. Giả sử, hàm đánh giá sai C-4, B-3, chiến lược này sẽ bế tắc, không tìm được đường đi đến trạng thái đích. Dễ dàng kiểm tra trường hợp này.

Tìm kiếm leo đồi chỉ hiệu quả khi có một hàm đánh giá tốt. Khi hàm đánh giá không tốt, việc tìm kiếm có thể rơi vào bế tắc trong những trường hợp sau đây:

- Phát triển đến được nút tốt nhất trong khu vực nhưng không phải là trạng thái đích. Nút như vậy được gọi là điểm cực đại địa phương. Điều này xảy ra do thuật giải leo đồi chỉ phát triển nếu tìm được trạng thái con tốt hơn. Khi không tìm được

trạng thái như vậy, việc tìm kiếm dừng lại. Một cách hình ảnh, việc tìm kiếm không vượt được qua khu vực trũng để đến khu vực chứa trạng thái đích.

- Các trạng thái con được sinh ra có độ tốt ngang bằng với trạng thái đang xét. Điều này cũng làm cho việc tìm kiếm dừng lại. Nói cách khác, khu vực bằng phẳng cũng là một trở ngại trong chiến lược tìm kiếm này.

Cũng có một vài giải pháp như xáo trộn ngẫu nghiên các trạng thái chờ đợi trong open để khắc phục các tình trạng trên. Tuy nhiên, không có một giải pháp tổng quát cho vấn đề này. Lý do xảy ra các tình huống thất bại của giải thuật leo đồi là do giải thuật này quá cực đoan, nó không lưu lại các trạng thái anh em nên không thể quay lại các trạng thái trước đó khi cần thiết. Khắc phục hạn chế này là chiến lược sau đây.

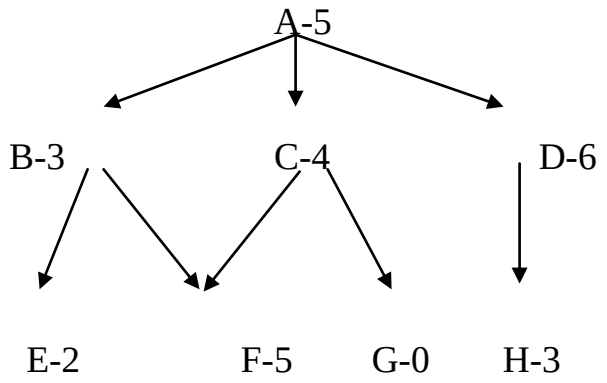
b) Tìm kiếm ưu tiên tốt nhất – Best First Search (Best-FS)

Chiến lược này chọn trạng thái tốt nhất trong các trạng thái đang chờ trong open để phát triển, kể cả trạng thái này không tốt bằng trạng thái đã sinh ra nó. Điều này giải thích vì sao nó có thể đi xuống vùng thấp trong khi giải thuật leo đồi thì không xuống được. Một điều nữa là nó lưu trữ tất cả các trạng thái anh em khi phát triển trạng thái mới vì vậy khi đi vào ngõ cụt, chiến lược này có thể lui ra được mà không bị bế tắc. So với một số chiến lược khác chúng ta có nhận xét rằng, giải thuật Best-FS kết hợp hai phương pháp tìm kiếm theo chiều sâu và chiều rộng nhằm tận dụng ưu thế của cả hai. Tìm kiếm theo chiều sâu có ưu thế là không phải quan tâm đến sự mở rộng của tất cả các nhánh, trong khi đó tìm kiếm theo chiều rộng lại không bị sa vào các nhánh bế tắc. Best-FS "cẩn thận" hơn ở chỗ khi di chuyển theo một hướng khả quan nó vẫn ghi nhớ lại các một số trạng thái không tốt hơn trước đó để còn có thể quay lại nếu như những bước tiếp theo của hướng đang đi không còn khả quan như trước.

Cụ thể hơn, Best-FS sẽ lựa chọn trạng thái tiếp theo là trạng thái có khả năng tốt nhất trong các trạng thái đã gặp, khác với tìm kiếm leo đồi dốc đứng chọn trạng thái tốt nhất trong số các trạng thái kế tiếp có thể từ trạng thái hiện tại. Như vậy, Best-FS sẽ linh hoạt hơn, nó kịp thời di chuyển sang một nhánh khác khả quan hơn, không tiếp tục đi vào nhánh cụt.

Để làm điều này Best-FS cải tiến so với leo đồi ở chỗ: thứ nhất là nó giữ lại trong open tất cả các con của trạng thái đang xét – để có thể sẽ quay lại nếu bế tắc

trên một hướng đi khác. Thứ hai là trong số các trạng thái đang xét nó sẽ chọn trạng thái tốt nhất, vì vậy phải sắp xếp open để chuẩn bị cho việc chọn phần tử này.



Ví dụ: Không gian trạng thái như hình 2.

Bước 1	Open	Close (bổ sung mỗi thời điểm)
1	A-5	
2	B-3, C-4, D-6	A-5
3	E-2, C-4, F-5,D-6	B-3
4	C-4, F-5,D-6	E-2
5	G-0 ,F-5,D-6	C-4

Giải thuật được mô tả như sau.

```
procedure Best_FS;
begin
    open:={u0}; closed:={ };
    while open<>{ } do
        begin
            loại trạng thái ngoài cùng bên trái của open, gọi nó là u
            if (u là một đích) then thông báo thắng lợi, thoát
            else begin
                Đưa u vào closed
                Phát sinh các con v của u
                Loại các con  v đã có mặt trong open + closed
                Đưa các con còn lại vào open
                Sắp xếp open sao cho phần tử tốt nhất nằm bên trái
```

```
            end
        end
    Thông báo thất bại
end
```

Một cách đầy đủ, để có được kết quả là đường đi nghiệm chúng ta phải lưu ý thêm về việc lưu giữ các trạng thái cha để truy lại vết của đường đi này.

$3. CÁC CHIẾN LƯỢC TÌM KIẾM TỐI ƯU

Trong các chiến lược tìm kiếm trên, chúng ta chỉ quan tâm đến việc làm thế nào để tìm được trạng thái đích mà không quan tâm đến độ dài hay chi phí của đường đi nghiệm. Trong thực tế, đây là một yếu tố quan trọng vì giá trị hay ý nghĩa của nghiệm tìm được phụ thuộc vào yếu tố này. Ví dụ như số bước đi của một người chơi cờ. Cũng tìm được trạng thái đích nhưng chỉ cần nhiều hơn đối thủ một bước thì đối thủ đã thắng. Chi phí một hành trình trong bài toán người du lịch cũng là một ví dụ cho yếu tố này. Vì vậy chúng ta không chỉ quan tâm đến việc tìm ra nghiệm mà còn phải quan tâm đến việc nghiệm đó có tối ưu hay không.

KGTT trong bài toán tối ưu phải có thêm trọng số của các cung. Chúng ta đã phân tích các chiến lược từ tìm kiếm mù, tìm kiếm kinh nghiệm và biết được những ưu nhược điểm của từng chiến lược. Tìm kiếm Best-FS là chiến lược tìm kiếm linh hoạt nhất cho đến lúc này. Chúng ta sẽ tiếp tục cải tiến chiến lược này để giải quyết vấn đề tối ưu.

1. Hàm đánh giá cải tiến.

Trong Best-FS, một trạng thái u được ưu tiên phát triển nếu như nó có giá trị h(u) nhỏ nhất trong các trạng thái trong open. Tuy nhiên, chỉ riêng h(u) là không đủ khi tìm kiếm đường đi tối ưu. Với quan điểm tối ưu, đồng thời với h(u) nhỏ để mau về đích thì độ dài đường đi từ trạng thái xuất phát đến u cũng phải nhỏ. Vì vậy, người ta dùng hàm đánh giá cải tiến sau:

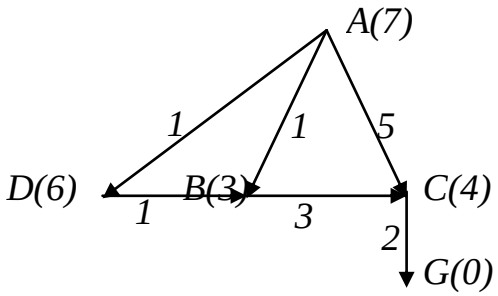
$$f(u) = g(u) + h(u)$$

trong đó g(u) là độ dài đường đi ngắn nhất từ u_0 đến u, h(u) là đánh giá độ tốt theo hàm heuristic. Hàm h(u) được gọi là chấp nhận được hay hàm đánh giá thấp nếu như h(u) nhỏ hơn độ dài đường đi ngắn nhất thực sự từ u đến đích.

2. Giải thuật A*

A* sẽ là giải thuật tổng quát tìm kiếm trên KGTT là đồ thị. Vì là đồ thị nên phát sinh nhiều vấn đề khi tìm đường đi tối ưu. Để ý rằng nghiệm là đường đi nên ta phải lưu lại vết của đường đi này. Trong các giải thuật trước, để tập trung cho tư tưởng chính của các giải thuật đó chúng ta bỏ qua chi tiết này, ở đây chi tiết này được đề cập vì nó liên quan đến nghiệm một cách trực tiếp. Trạng thái u tùy ý sẽ gồm bốn yếu tố (g(u), h(u), f(u), cha(u)). Trong đó, g(u) là độ dài đường đi ngắn nhất từ đích đến u, h(u) là giá trị ước lượng khả năng về đích, f(u) là giá trị của hàm đánh giá mở rộng, cha(u) là trạng thái sinh ra u.

Bây giờ ta sẽ mô tả cách làm việc của A*. Xét KGTT được cho trong hình vẽ sau.



Các trạng thái sẽ được lưu giữ trong open và closed trong các bước như sau

Bước	Open	closed
1	A(0,7,0,-)	
2	B(3,3,6,A), D(1,6,7,A), C(5,4,9,A)	A(0,7,0,-)
3	D(1,6,7,A), C(4,4,8,B)	~~B(3,3,6,A)~~
4	B(2,3,5,D), C(4,4,8,B)	D(1,6,7,A)
5	C(3,4,7,B)	B(2,3,5,D)
6	G(5,0,5,C)	C(3,4,7,B)

Ở bước 2, mọi việc xảy ra bình thường như với các giải thuật trước.

Ở bước 3, tìm được đường đi đến C qua B ngắn hơn nên các giá trị của C trong open phải được sửa đổi.

Ở bước 4, mặc dù B đã nằm trong closed, tức đã xét xong nhưng đường đi mới qua D đến B ngắn hơn nên B phải được lấy khỏi closed chuyển qua open chờ xét lại với giá trị mới.

Ở bước 5, lại xảy ra việc chỉnh sửa các giá trị của C như ở bước 3.

Giải thuật dừng ở bước 6 và đường đi thu được độ dài 5 như sau A-D-B-C-G.

Như vậy, khác với các chi tiết đã trình bày trong các giải thuật trước đây, trạng thái đã lưu trong open vẫn phải thay đổi giá trị, hoặc thậm chí lưu trong closed cũng phải bị xóa đi và chuyển qua open với các giá trị mới.

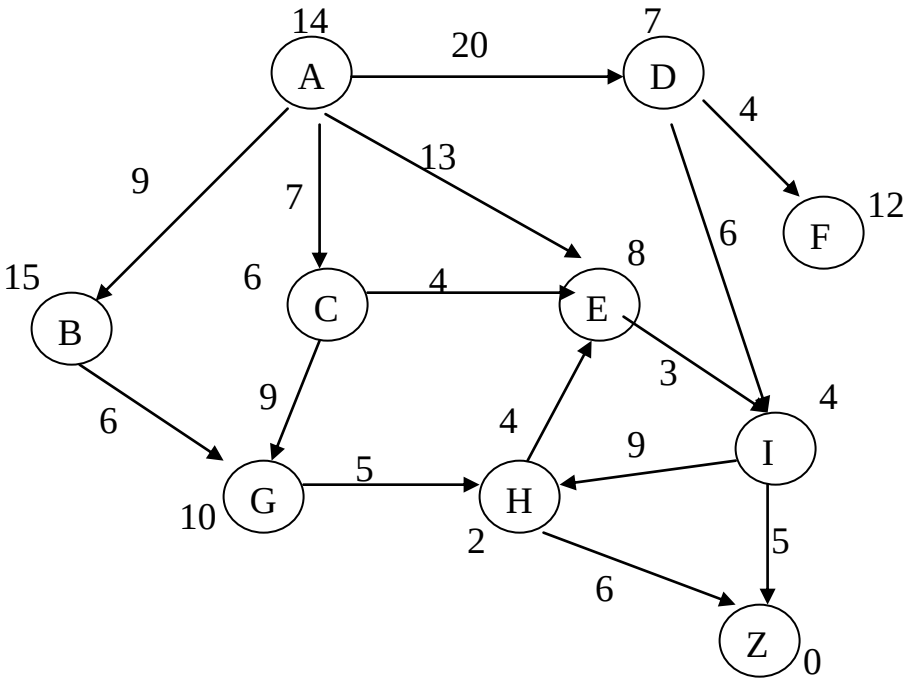
Tóm lại, giải thuật được trình bày dưới dạng giả mã sau:

```
procedure A*(uo);
begin
    g(uo)=0; f(uo)=g(uo);
    open:=[uo]; closed:=[];
    while open<>[] do
        begin
            loại trạng thái ngoài cùng bên trái của open, gọi nó là u
            Đưa u vào closed;
            if (u là một đích) then thông báo thắng lợi, thoát
            else begin
                Sinh các con v của u;
                For v thuộc con(u) do
                    begin
                        g(v):=g(u)+c[u,v];
                        If v không thuộc open hay closed
                            begin
                                f(v):=g(v)+h(v);
                                cha(v):=u;
                                Bỏ v  vào open;
                            end
                        If v thuộc open (tồn tại v' thuộc open, sao cho v=v')
                            If g(v)<g(v') then
                                Begin
                                    g(v'):=g(v);
                                    f(v'):=g(v')+h(v');
                                    Cha(v'):=u;
                                End;
```

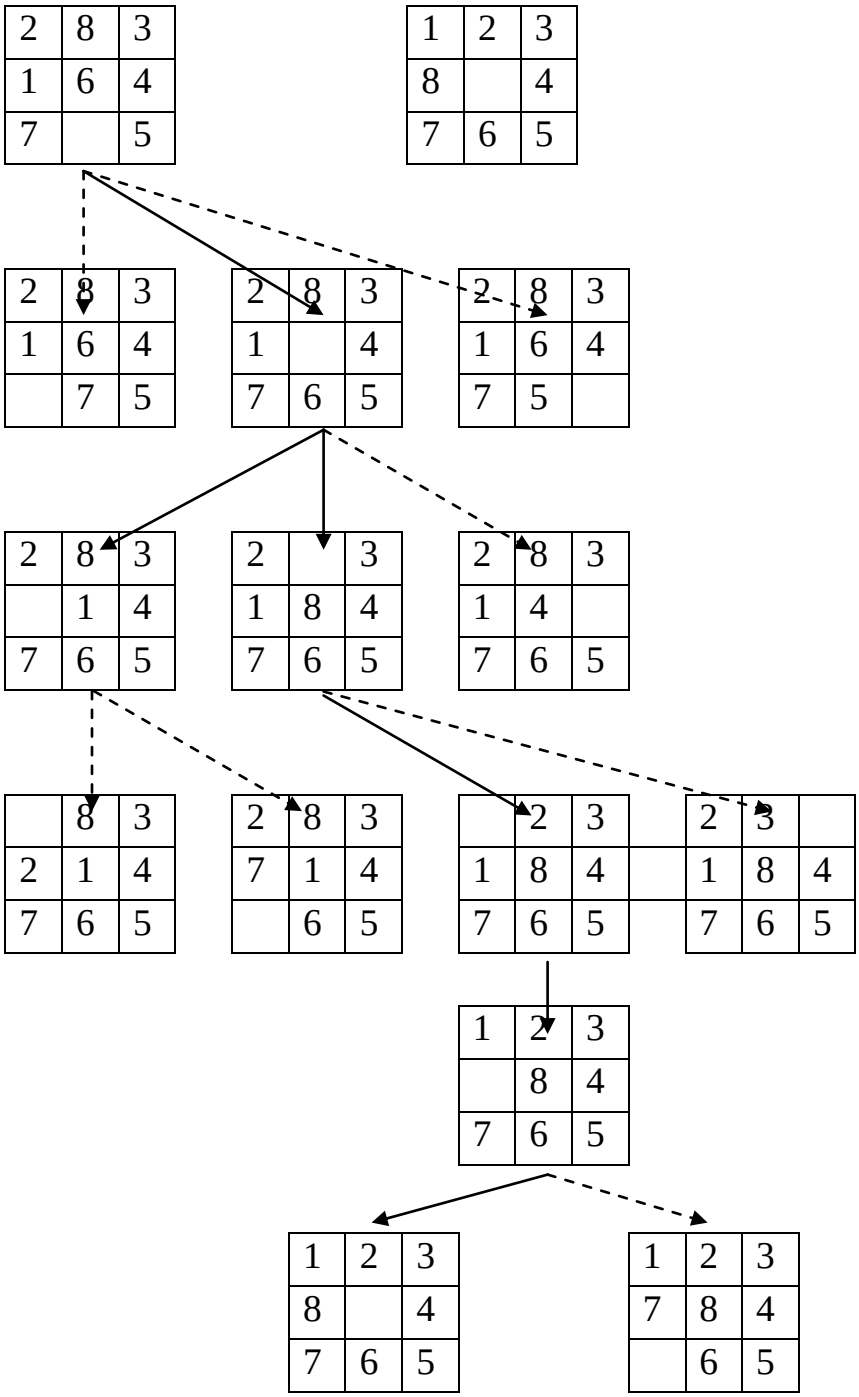
If v thuộc closed (tồn tại v' thuộc closed, sao cho v=v')
If g(v)<g(v') then
Begin
f(v):=g(v)+h(v);
cha(v):=u;
Đưa v vào open;
Loại v' khỏi closed;
End;
End;{for}
Xếp open để trạng thái tốt nhất nằm bên trái;
End{else}
End;{while}
Thông báo thất bại

End;{procedure}

Ví dụ. Hoạt động của giải thuật với KGTT là bản đồ giao thông.



Bài tập: Hãy đánh giá các trạng thái của bài toán trò chơi 8 số và mô tả hoạt động của A* đối với không gian trạng thái này.



Khái niệm Hàm đánh giá thấp

Một hàm đánh giá h(u) được gọi là chấp nhận được hay là hàm đánh giá thấp nếu như h(u)<=h*(u) với mọi u, ở đây h*(u) là đường đi ngắn nhất từ u đến đích.

Mệnh đề

Nếu hàm đánh giá h(u) là chấp nhận được thì thuật toán A* là tối ưu.

Thật vậy, giả sử G là một trạng thái đích mà thuật toán tìm được và có độ dài đường đi tử uo đến G là g(G). Do G là trạng thái đích, h(G)=0 nên f(G)=g(G).

Giả sử nghiệm tối ưu là một đường đi khác từ uo kết thúc tại G1 có độ dài l. Nếu G1 nằm trong cây thì do G được chọn trước nên f(G)<=f(G1), suy ra g(G)<=g(G1)=l. Trường hợp G1 nằm ngoài cây tìm kiếm. Giả sử I là nút lá mà đường đi tối ưu thóat ra khỏi cây tìm kiếm. Vì hàm đánh giá thấp nên f(I) = g(I)+ h(I) <=g(G1). Do G được chọn trước I nên f(G)<=f(I)<=g(G1), do đó g(G)<=g(G1). Vậy G mới là nghiệm tối ưu.

A* là thuật tóan hiệu quả nhất trong các thuật toán đầy đủ và tối ưu cho bài toán tìm đường đi ngắn nhất.

$4. CÁC CHIẾN LƯỢC TÌM KIẾM CÓ ĐỐI THỦ

1. Trò chơi có đối thủ và đặc trưng

Trong các bài trước, chúng ta giải quyết trò chơi chỉ có một người thực hiện như trò chơi 8 số. Chơi caro, cờ tướng, cờ vua,…không như vậy. Những trò chơi này có hai người thực hiện, máy tính sẽ đóng vai người thứ nhất, người còn lại chúng ta gọi là đối thủ và gọi cho gọn loại trò chơi này là trò chơi có đối thủ.

Vì hai người thay phiên thực hiện các bước đi nên máy tính chỉ chủ động một nữa, nữa còn lại do đối thủ quyết định. Chúng ta cũng không biết trước chiến lược của đối thủ. Vì vậy, phải tôn trọng đối thủ và cho rằng đối thủ sẽ sử dụng phương án tốt nhất có thể. Đặc trưng của loại trò chơi này là hai bên biết rõ thông tin về nhau.

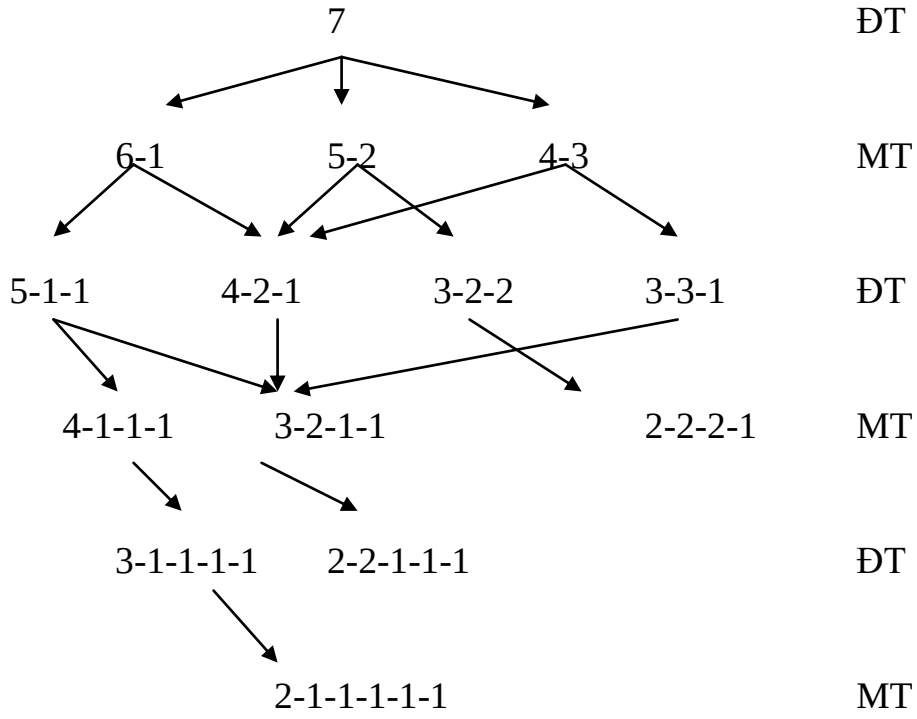
Giống như các bài toán trước đây, không gian trạng thái của trò chơi là cây, được gọi là cây trò chơi. Các lớp trên cây trò chơi luân phiên ứng với đối thủ và máy tính. Nghiệm của bài toán là đường đi từ trạng thái ban đầu đến trạng thái đích. Trong đó, một nữa số nút sẽ nằm tại lớp của đối thủ và số còn lại nằm tại lớp của máy tính. Nhiệm vụ của chúng ta là tìm được nghiệm và cụ thể là cách đi của máy tính tại các lớp tương ứng của nó, hay nói cách khác là tại mỗi lớp của máy tính, ta phải chỉ ra đi theo phương án nào trong những phương án có thể.

2.Thủ tục Minimax cơ bản

Trò chơi Nim. Có n (n>2) đồng xu. Mỗi nước đi, người chơi chia các đồng xu này thành hai đống nhỏ có số lượng mỗi đống khác nhau. Người thua sẽ là người

cuối cùng không chia được theo yêu cầu số lượng mỗi đống khác nhau của bài toán.

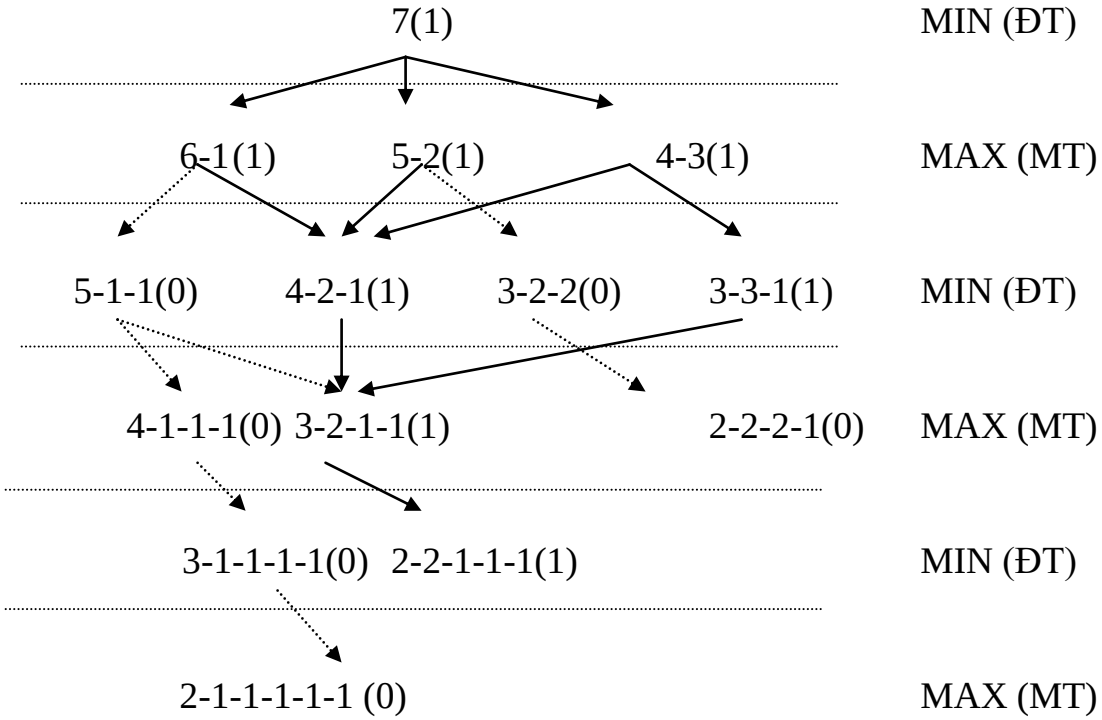
Ví dụ: n=7, một số trạng thái ứng với các cách chia. Các nút lá là các trạng thái không thể chia được nữa. Nó xuất hiện ở lược chơi của người nào thì người đó thua cuộc.



Nút lá xuất hiện ở lớp của máy tính (MT) thì máy tính thua vì máy tính không thể chia được nữa. Tương tự, nút lá xuất hiện ở lớp của (ĐT) thì đối thủ thua. Tất nhiên, máy tính thua thì đối thủ thắng và ngược lại. Các nút này được cho điểm bằng hàm đánh giá. Cho 1 nếu nút lá rơi vào lớp của máy tính, cho 0 nếu nút lá rơi vào lớp của đối thủ. Giá trị 1 với ý nghĩa máy tính thắng (đối thủ thua), giá trị 0 ứng với máy tính thua (đối thủ thắng). Giá trị càng lớn máy tính càng có khả năng thắng và ngược lại. Theo ý nghĩa đó, chúng ta sẽ đánh giá các nút còn lại.

Từ các nút lá lần ngược các nút ở lớp trên để đánh giá cho đến hết. Khi cần đánh giá nút ở lớp của máy tính, ta xem xét giá trị các nút con của nó. Nếu có ít nhất một nút con của nó có giá trị 1 thì khả năng thắng thuộc về máy tính vì chúng ta có thể điều khiển máy tính đi theo nhánh có giá trị 1. Ngược lại, nếu tất cả các nút con đều có giá trị 0 thì giá trị của nút cha ở lớp của máy tính được đánh giá là 0 vì không có phương án nào dẫn đến khả năng thắng cho máy tính. Tổng quát, nút thuộc lớp của máy tính sẽ có giá trị bằng max của các nút con. Phân tích tương

tự, nút thuộc lớp của đối thủ có giá trị bằng min của các nút con. Vì vậy, lớp của máy tính còn được kí hiệu là lớp MAX, lớp của đối thủ được kí hiệu là lớp MIN.



Như vậy, thủ tục gán nhãn cho một trạng thái u như sau:

Trường hợp u thuộc lớp MAX:

```
Function MaxVal(u);
Begin
    If  u là nút lá then Val(u):=f(u)
        Else MaxVal(u):=max{MinVal(v) | v là các con của u}
End;
```

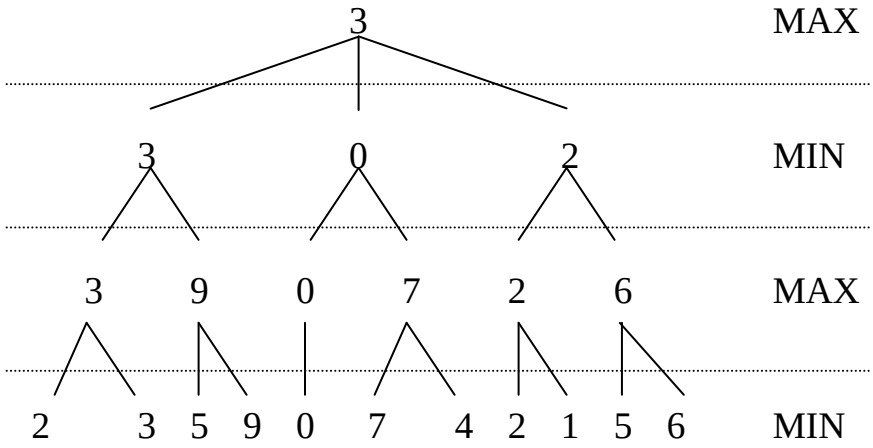
Trường hợp u thuộc lớp MIN:

```
Function MinVal(u);
Begin
    If  u là nút lá then MinVal(u):=f(u)
        Else MinVal(u):=min{MaxVal(v) | v là các con của u}
End;
```

Chúng ta đã có cách để đánh giá các nút khi cần. Việc còn lại là hướng dẫn cách đi tại nút mà máy tính được điều khiển. Để máy tính thắng phải hướng dẫn đi vào nhánh có khả năng thắng cuộc. Đó là nhánh dẫn đến nút có giá trị lớn nhất. Do đó, công việc đơn giản chỉ là tìm nút giá trị lớn nhất để di chuyển đến bằng thủ tục sau. Trong đó, u là nút thuộc lớp MAX và v là nút sẽ di chuyển đến.

```
Procedure Minimax(u, var v);
Begin
    Val:= −∞;
    For  mỗi con w của u do
        If  val<=MinVal(w) then begin
            Val:=MinVal(w);
            v:=w;
        End;
End;
```

3. Chiến lược Minimax với độ sâu nhất định



Trong thực tế, trừ những trường hợp quá đơn giản hiếm khi mở rộng được không gian trạng thái đến các nút lá , bởi vì sẽ gặp phải khó khăn về thời gian, bộ nhớ do không gian trạng thái rất lớn. Do đó ta chỉ tính được đến một mức n nào đó thôi. Vì chưa phải là nút lá nên không thể gán cho một đỉnh nhãn thắng hay thua cuộc một cách chính xác mà phải thay vào đó bằng khả năng thắng thua – tức là một heuristic nào đó. Sau đó, giống như trên, truyền ngược nhãn cho các nút đến

nút xuất phát. Như vậy các đỉnh có thể chọn lựa trong n mức kể từ đỉnh xuất phát sẽ có các giá trị ước lượng. Căn cứ vào giá trị này để quyết định bước đi.

Ví dụ. Cách thực hiện của thuật toán Minimax đối với một bộ phận không gian tìm kiếm với độ sâu cố định 3 mức (các giá trị dòng cuối cùng là đánh giá heuristic) trong hình trên.

Như vậy, thủ tục gán nhãn cho một trạng thái u, độ sâu mức d:

Trường hợp u thuộc lớp MAX:

```
Function MaxVal(u, d);
Begin
    If  d=0 hoặc u là đỉnh lá then MaxVal(u):=f(u)
        Else MaxVal(u):=max{MinVal(v, d-1) | v là các con của u}
End;
```

Trường hợp u thuộc lớp MIN:

```
Function MinVal(u,d);
Begin
    If  d=0 hoặc u là đỉnh lá then MinVal(u):=f(u)
        Else MinVal(u):=min{MaxVal(v,d-1) | v là các con của u}
End;
```

Và thủ tục chọn nước đi cho MAX khi đang ở trạng thái u, kết quả trả về là v

```
Procedure Minimax(u, var v, d);
Begin
    Val:= −∞;
    For  mỗi con w của u do
        If  MinVal(w, d-1) >=Val then begin
            Val:=MinVal(w,d-1);
            v:=w;
        End;
End;
```

Một số ví dụ để xác định heuristic kiểu này như sự chênh lệch số lượng các quân cờ hay gán cho mỗi quân cờ một hệ số chỉ mức độ quan trọng, hay kết hợp với đánh giá vị trí của chúng.

Chẳng hạn trong cờ vua, đối với quân trắng mỗi tốt cho hệ số 1; mã, tượng hệ số 3; xe hệ số 5, hậu hệ số 9. Quân đen nhận giá trị âm ngược lại đối với quân trắng. Tổng giá trị cả hai quân dùng để đánh giá trạng thái đó. Với cách đánh giá này ta chưa tính đến vị trí của chúng.

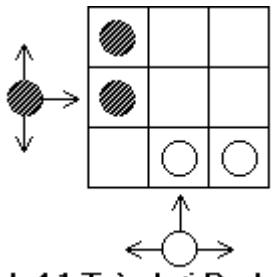
Trong chương trình chơi cờ của Samuel, heuristic được sử dụng là một tổng $\sum_i a_i x_i$ với mỗi x_i là một đặc trưng của bàn cờ như ưu thế quân, vị trí quân, khả năng kiểm soát trung tâm, cơ hội thí quân để ăn quân của đối thủ, a_i là hệ số của x_i đánh giá mức độ quan trọng của yếu tố này trong trạng thái. Thậm chí các hệ số có thể được điều chỉnh một cách linh hoạt để nâng cao hiệu năng dựa trên quá trình học, cụ thể là những yếu tố có hệ số lớn nhưng thất bại bị giảm nhỏ đi, tăng hệ số cho các yếu tố khác. Tập luyện bằng cách chơi với đối thủ cao hơn hoặc một phiên bản khác của chính nó là việc điều chỉnh để có được một bộ hệ số tốt nhất. Tuy nhiên, chương trình này không có một chiến lược toàn cục, một điều rất khó trở thành hiện thực nên một khi đối phương đi một nước không theo "sách vở" thì các hệ số của đa thức đánh giá không theo được sự điều chỉnh, chọn các giá trị ngẫu nhiên dẫn đến sự rối loạn trong chiến lược.

Ví dụ 1. Thuật giải Minimax cho trò chơi caro

Heuristic sử dụng ở đây là chênh lệch giữa số đường có khả năng thắng của MAX trừ đi số đường có khả năng thắng của MIN.

Ta sẽ xét một ví dụ về hàm heuristic trong trò chơi sau.

Ví dụ 2. Trò chơi Dodgem – một ví dụ về thủ tục minimax với cách đánh giá heuristic.

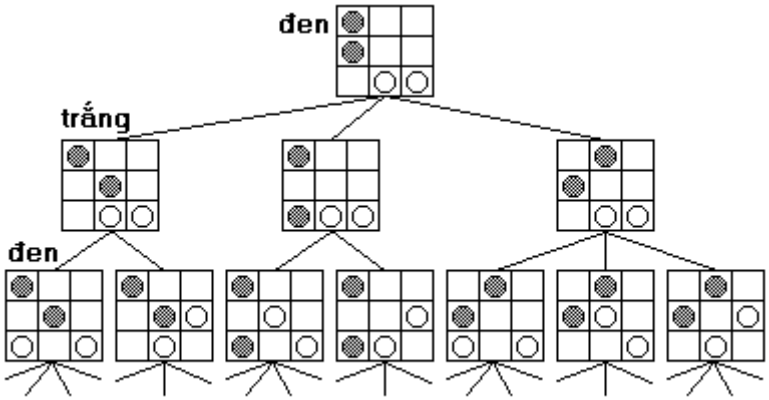


Hình 4.1 Trò chơi Dodgem.

Có hai quân trắng và hai quân đen trên bàn cờ 3x3. Quân đen không qua được bên trái mà chỉ di chuyển lên trên, xuống dưới hoặc qua phải. Quân trắng không đi xuống dưới mà chỉ lên trên qua trái hoặc qua phải.

Quân đen chỉ được ra khỏi bàn cờ nếu đang đứng ở cột cuối cùng, quân trắng chỉ được ra khỏi bàn cờ nếu đang đứng ở dòng trên cùng. Đấu thủ thắng cuộc nếu đưa hết 2 quân của mình ra khỏi bàn cờ hoặc làm cho đối phương không đi được.

Sau đây là một số trạng thái của trò chơi.



Hình 4.2 Cây trò chơi Dodgem với Đen đi trước.

Một ước lượng heuristic có thể thực hiện dựa trên một số đánh giá vị trí:

30	35	40
15	20	25
0	5	10

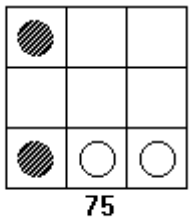
Giá trị quân Trắng.

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

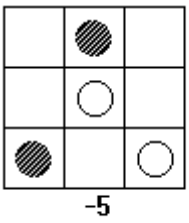
Giá trị quân Đen.

Hình 4.4 Đánh giá các quân trong trò chơi Dodgem.

Hơn nữa, có thể xét thêm thế cản. Mỗi quân trắng cản trực tiếp quân đen cộng 40 điểm, cản gián tiếp cộng 30 điểm và ngược lại cho quân đen. Ví dụ



75



-5

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

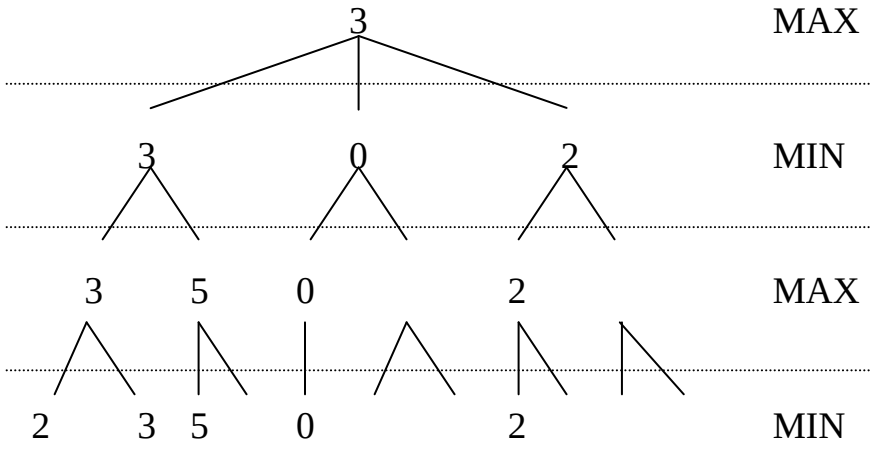
Ngoài ra, do số quân cờ càng ít thì khả năng thắng càn cao nên nếu chỉ còn một quân trắng thì được cộng thêm như 50 chẳng hạn, ngược lại chỉ còn một quân đen thì -50.

3. Thủ tục alpha-beta

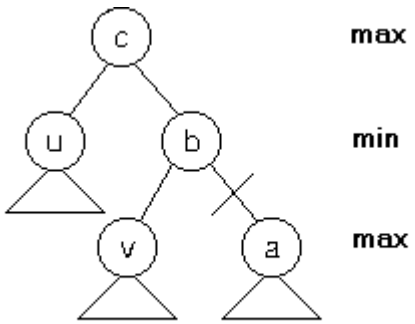
Trong thủ tục Minimax, mặc dù hạn chế với số mức d đi nữa thì số trạng thái – tức số nút cần khảo sát cũng đã rất lớn. Trong cờ vua, số con trung bình của mỗi nút khoảng 35. Khi đó ở mức thứ 3 – chưa phải là nhiều, ta phải lượng giá 35*35*35=42.785 trạng thái chưa kể ở các mức 1, 2 trong một thời gian một vài phút (vì không thể để đối phương chờ quá lâu). Với khả năng nhất định của máy tính, giải quyết vấn đề thời gian như thế nào? Liệu có cách để giảm bớt các trạng thái cần khảo sát mà vẫn không ảnh hưởng gì đến việc giải quyết bài toán. Điều này thực hiện được với thủ tục được mang tên là alpha-beta sau đây.

Trong thủ tục Minimax, để đánh giá một trạng thái u với độ sâu m ta phải đánh giá hết tất cả các nút của cây từ lớp mức m, sau đó truyền ngược các nhãn lên các mức trên.

Ví dụ.



Tư tưởng của thuật toán alpha-beta:



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

Giả sử, trong quá trình gán nhãn ta đi xuống nút a trong lớp max, đỉnh a có nút anh em là v đã được đánh giá; nút cha của b là c; u là nút anh em với b đã được

đánh giá. Khi đó nếu val(v)< val(u) thì bất kể nhãn của a bằng bao nhiêu val(c) cũng phải là val(u). Tương tự, giả sử a thuộc lớp min. Khi đó nếu val(u)<val(v) thì val(c) cũng phải là val(u). Như vậy không phải khảo sát cây con nút a vẫn có được đánh giá cho c như thuật toán minimax.

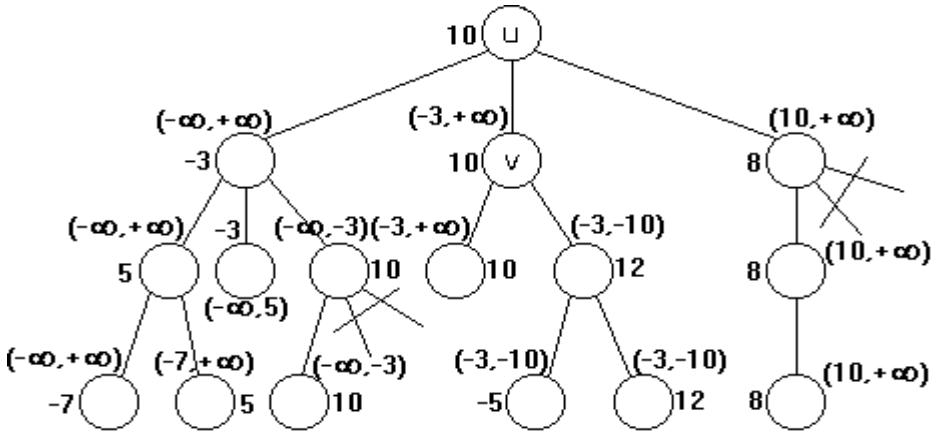
Khi cài đặt, ta dùng một cặp biến alpha, beta. Trong đó alpha là giá trị lớn nhất trong các giá trị của các các nút con đã đánh giá của một nút max. Tương tự beta là giá trị lớn nhất trong các giá trị của các nút con đã được đánh giá của một nút min. Giá trị apha, beta là biến cục bộ trong các hàm gán giá trị cho các nút. Để ý rằng quá trình gán giá trị cho các nhãn sử dụng thuật toán tìm kiếm sâu trên cây có độ sâu hạn chế nên các giá trị apha, beta sẽ được nhận khi đi xuống và giá trị các nút được gán khi quay lên. Quá trình này cũng làm thay đổi giá trị anpha, beta cho những lần gọi sau đó.

```
Function MaxVal(u, anpha, beta);
begin
    if (u là nút lá của cây hoặc cây hạn chế) then Maxval:=f(u)
    else
        for w con của u do
            begin
            alpha:=max{alpha, MinVal(w, alpha, beta)}
            if anpha>=beta then exit;
        end;
MaxVal:=alpha;
end;
Function MinVal(u, anpha, beta);
begin
    if (u là nút lá của cây hoặc cây hạn chế) then Minval:=f(u)
    else
        for w con của u do
            begin
                beta:=min{beta, MaxVal(w, alpha, beta)}
                if anpha>=beta then exit;
            end;
    MinVal:=beta;
end;
```

Thủ tục hướng dẫn nước đi cho Max

```
Procedure Alpha_beta(u, var v)
begin
    alpha:=−∞; beta:=+∞;
    for (w con u) do
        if alpha<MinVal(w, alpha, beta) then
            begin
                alpha:=MinVal(w, anpha, beta);
                v:=w;
            end;
end;
```

Sau đây là minh họa cho thủ tục cắt nhánh anpha-beta nói trên trong một không gian trạng thái



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

CHƯƠNG 2. BIỂU DIỄN TRI THỨC VÀ LẬP LUẬN

Khái niệm về biểu diễn và xử lý tri thức

Dữ liệu là tín hiệu mà máy tính có thể lưu trữ, xử lý. Bản thân dữ liệu không có ý nghĩa. Chỉ khi con người cảm nhận, tư duy thì dữ liệu mới có một ý nghĩa nhất định, đó chính là thông tin. Tri thức là kết tinh, cô đọng, chắt lọc của thông tin. Tri thức hình thành do quá trình xử lý thông tin mang lại.

Ví dụ:

Các định luật tóan học, vật lý là các tri thức mang tính khẳng định sự kiện.

Các phương pháp điều chế hóa học, thuật toán là tri thức mang tính thủ tục.

Các nhận định, kết luận về sự kiện, hiện tượng là tri thức mô tả.

Các ước lượng, suy đoán hình thành qua kinh nghiệm là tri thức heuristic

Trí tuệ, sự thông minh phải dựa trên nền tảng của tri thức. Tuy nhiên, nó chỉ là một yếu tố cấu thành và còn phụ thuộc vào việc vận dụng, xử lý tri thức. Một người có trí nhớ tốt chưa hẳn đã thông minh mà phải biết xử lý và vận dụng tri thức để đưa ra các giải pháp, quyết định một cách đúng đắn, hiệu quả.

Biểu diễn tri thức là việc đưa tri thức vào máy tính. Và chỉ có ý nghĩa nếu công việc tiếp theo: "xử lý tri thức được thực hiện".

Ví dụ. Cho 2 bình rỗng X, Y có thể tích lần lượt là Vx, Vy. Dùng 2 bình này để đong ra z lít nước.

Cụ thể với Vx=5, Vy=7 và z=4, ta làm như sau:

- Múc đầy bình 7
- Đổ qua cho đầy bình 5.
- Đổ hết nước trong bình 5
- Đổ phần còn lại trong bình 7 qua bình 5
- Múc đầy bình 7
- Đổ từ bình 7 qua cho đầy bình 5
- Phần còn lại trong bình 7 là 4 lít

Kinh nghiệm rút ra từ trường hợp cụ thể này là gì? “Khi bình 7 đầy mà bình 5 chưa đầy thì đổ nước từ bình 7 qua cho đầy bình 5”. Thử truyền kinh nghiệm này cho máy tính để nó vận dụng cho trường hợp tổng quát.

Giả sử Vx<Vy, gọi lượng nước trong bình X là x, trong bình Y là y. Điều kiện kết thúc bài toán là x=z hoặc y=z. Ban đầu x=y=0.

Kinh nghiệm trên có thể biểu diễn bằng các luật sau:

Luat 1) Nếu bình Y rỗng thì đổ nước đầy bình Y.

Luat 2) Nếu bình X đầy thì đổ hết ra.

Luat 3) Nếu bình Y không rỗng, bình X chưa đầy thì đổ nước từ bình Y sang bình X cho đến khi hết nước trong bình Y hoặc bình X đầy.

```
x:=0; y:=0;
While (x<>z) and (y<>z) do
  begin
    if y=0 then y:=Vy;
    if x=Vx then x:=0;
    if y>0 then
      begin
        k:=min(Vx-x, y);
        y:=y-k;
        x:=x+k;
      end;
  end;
```

Ví dụ: Vx=3, Vy=4, z=2

L1: x=0, y=4

L3: x=3, y=1

L2: x=0, y=1

L3: x=1, y=0

L1: x=1, y=4

L3: x=3, y=2

Tập 3 luật được cài đặt trên đây được gọi là cơ sở tri thức. Cách tìm kiếm các luật rồi áp dụng nó được gọi là động cơ suy diễn.

Như vậy, cơ sở tri thức là tập hợp các tri thức liên quan đến vấn đề cần giải quyết; động cơ suy diễn là cách vận dụng cơ sở tri thức để giải quyết vấn đề.

Một cách tự nhiên, tri thức được phát biểu bằng ngôn ngữ tự nhiên. Con người có thể hiểu và xử lý tri thức ở dạng này nhờ suy luận nhưng máy tính thì không thể. Để máy tính có thể xử lý được tri thức chúng ta phải biểu diễn chúng theo một quy tắc nào đó thuận lợi cho việc xử lý, tương tự như các ngôn ngữ lập trình. Tuy nhiên, ngôn ngữ lập trình quá gò bó và không phong phú. Vì vậy chúng ta sẽ biểu diễn bằng các ngôn ngữ hình thức, tổng quát hơn, gọi chung là ngôn ngữ biểu diễn tri thức.

$1. LOGIC MỆNH ĐỀ

Ngôn ngữ biểu diễn tri thức: Để biểu diễn sự hiểu biết, kinh nghiệm của con người cần phải có một sự đặc tả dựa vào một công cụ nào đó. Chúng ta có thể dùng các câu trong ngôn ngữ tự nhiên. Tuy nhiên để thuận tiện cho việc xử lý của máy tính người ta sẽ dùng một số ngôn ngữ hình thức. Ngôn ngữ biểu diễn tri thức là một ngôn ngữ hình thức gồm hai thành phần cơ bản: cú pháp và ngữ nghĩa.

Cú pháp của một ngôn ngữ bao gồm các ký hiệu, qui tắc liên kết (các phép toán) để tạo thành các câu (các công thức) trong ngôn ngữ.

Ngữ nghĩa chỉ xuất hiện khi các ký hiệu nhận giá trị cụ thể trong một miền nào đó.

Ngoài ra, ngôn ngữ cần được trang bị một cơ chế lập luận. Cơ chế này sử dụng các luật suy diễn. Một luật suy diễn là cách để suy ra một công thức từ một tập công thức nào đó. Như vậy,

Ngôn ngữ biểu diễn tri thức = cú pháp + ngữ nghĩa + cơ chế lập luận.

1 Cú pháp

Các ký hiệu:

- Hằng logic True, False
- Mệnh đề P, Q,...
- Phép toán logic: $\wedge, \vee, \neg, \rightarrow$
- Các dấu (,)

Quy tắc xây dựng công thức:

- Các mệnh đề là công thức
- Nếu A, B là công thức thì $\neg A$ là công thức và A*B là công thức với * là một phép toán logic 2 ngôi.

Quy ước: Các mệnh đề là các câu đơn, còn lại là các câu phức. Nếu P là một mệnh đề thì P và $\neg P$ được gọi là các literal, P là literal dương, $\neg P$ là literal âm. Câu phức có dạng $L_1 \vee L_2 \vee ... \vee L_n$ gọi là câu tuyển. Trong đó, mỗi L_i là một literal dương hoặc âm.

2 **Ngữ nghĩa**.

Công thức là một tập kí hiệu được liên kết theo quy tắc, nó chỉ có ý nghĩa tổng quát mang tính hình thức. Chẳng hạn $P \rightarrow Q$, có ý nghĩa tổng quát của một câu dạng nếu *P* thì *Q*, còn P, Q là gì thì chưa có nghĩa.

Khi mỗi thành phần cơ bản (mệnh đề) của một công thức nhận một giá trị cụ thể trên một miền nào đó, ta nói công thức có một thể hiện (interpretation) hay một minh họa.

Như vậy, một thể hiện là một phép gán mỗi mệnh đề với một giá trị chân lý. Khi đó mỗi câu sẽ có ngữ nghĩa có một giá trị chân lý. Giá trị này được tính qua giá trị chân lý của các thành phần bởi các phép toán logic bằng phương pháp bảng chân lý.

Các khái niệm

Công thức thỏa được là công thức mà tồn tại một minh họa làm cho công thức đúng.

Ví dụ. $P \rightarrow Q$ là một công thức thỏa được.

Công thức không thỏa được nếu nó sai đối với mọi minh họa.

Công thức là hằng đúng hay một tautology nếu nó đúng với mọi minh họa.

Ví dụ. $A \vee \neg A$, $A \to B$ không là công thức hằng đúng.

Mô hình của một công thức là một minh họa làm cho công thức đúng.

3 Dạng chuẩn tắc hội

Nhằm chuẩn hóa công thức, đưa về dạng thuận lợi cho việc lập luận suy diễn.

Một công thức ở dạng chuẩn hội nếu nó là hội của các câu tuyển.

Hai công thức A, B là tương đương, kí hiệu là $A \equiv B$, nếu chúng có cùng giá trị chân lý đối với mọi minh họa. Ví dụ

$A \to B \equiv \neg A \vee B$, $A \Leftrightarrow B \equiv (A \to B) \wedge (B \to A)$, $\neg(\neg A) \equiv A$

Hoặc công thức tương đương khác về tính giao hóan, kết hợp, phân phối, De Morgan.

Để đưa một công thức về dạng chuẩn hội ta thực hiện các thủ tục sau:

1) Bỏ các dấu kéo theo bằng cách dùng $A \to B \equiv \neg A \vee B$

2) Chuyển các dấu phủ định vào sát các biến mệnh đề bằng luật De Morgan hoặc $\neg(\neg A) \equiv A$

3) Áp dụng luật phân phối thay các công thức dạng
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

4 Câu Horn

Mọi công thức đều có thể đưa về dạng chuẩn hội, tức là hội của các tuyển, mỗi câu tuyển có dạng $\neg P_1 \vee ... \vee \neg P_m \vee Q_1 \vee ... \vee Q_n$, trong đó P_i, Q_i là các mệnh đề. Câu này tương đương với câu $P_1 \wedge ... \wedge P_m \Rightarrow Q_1 \vee ... \vee Q_n$.

Khi $n \le 1$, dạng câu trên được gọi là câu Horn. Với $m \ge 1, n = 1$ dạng câu này trở thành $P_1 \wedge ... \wedge P_m \Rightarrow Q$, rất thông dụng để biểu diễn các luật trong các hệ chuyên gia hoặc trong các bài toán lập luận xấp xỉ.

Khi $m = 0, n = 1$ ta có câu dạng $\to Q$ được hiểu là $1 \to Q \equiv 0 \vee Q \equiv Q$ và được gọi là câu sự kiện.

Nói chung không phải công thức nào cũng biểu diễn được thành hội các câu Horn, nhưng một cơ sở tri thức là một tập luật thì nó chính là tập các câu Horn.

5 Luật suy diễn

Công thức B được gọi là một hệ quả logic của tập công thức $\{A_1,...A_n\}$ nếu bất kỳ minh họa nào làm cho các công thức A_i đúng thì B cũng đúng.

Để suy ra một tri thức mới từ một cơ sở tri thức ta dùng các luật suy diễn. Các luật suy diễn quan trọng trong logic mệnh đề bao gồm:

Modus Ponens: $\frac{A \to B, A}{B}$

Modus Tollens: $\frac{A \to B, \neg B}{\neg A}$

Bắc cầu: $\frac{A \to B, B \to C}{A \to C}$

Loại bỏ hội: $\frac{A_1 \wedge ... \wedge A_i \wedge ... \wedge A_m}{A_i}$

Luật đưa vào hội: $\frac{A_1,...,A_n}{A_1 \wedge ... \wedge A_n}$

Luật phân giải: $\frac{A \vee B, \neg B \vee C}{A \vee C}$

Một luật suy diễn là đúng đắn (sound) nếu với bất kỳ minh họa làm cho các giả thiết đúng thì cũng làm cho kết luận đúng. Chúng ta có thể kiểm tra tính đúng đắn của các luật suy diễn nói trên bằng bảng chân lý.

Để ý rằng luật phân giải là tổng quát cho cả 3 luật MP, MT và bắc cầu.

6 Chứng minh suy diễn

Tiên đề là một tập các công thức cho trước, thường là không suy ra từ các công thức khác.

Định lý là tập các công thức suy ra từ các tiên đề bằng các luật suy diễn.

Dãy các suy diễn được sử dụng gọi là chứng minh của định lý.

Trong ứng dụng chúng ta thường có một tập công thức cho trước, tức các giả thiết, tương tự như các tiên đề và cần chứng minh một khẳng định nào đó, tức là một công thức đúng tương tự như một định lý. Khi đó, chúng ta dùng các luật suy diễn theo một trình tự hợp lý để từ các giả thiết suy ra điều cần chứng minh. Phương pháp như vậy được gọi là phương pháp chứng minh suy diễn.

Khi sử dụng phương pháp này nếu như ta chỉ ra được dãy luật suy diễn để chứng minh thì điều phải chứng minh là đúng nhưng nếu không chỉ ra được dãy luật suy diễn để chứng minh thì không khẳng định được điều phải chứng minh là sai.

7 Chứng minh phản chứng bằng luật phân giải.

Tư tưởng của chứng minh phản chứng là giả sử điều cần chứng minh sai, ta kết hợp với những căn cứ của giả thiết để dẫn tới một điều mâu thuẫn.

Dễ dàng kiểm tra được $A \to B \equiv (A \wedge \neg B) \to 0$ bằng cách biến đổi tương đương hoặc lập bảng chân lý. Để ý rằng công thức $(\neg A \wedge B) \to 0$ chỉ đúng khi $(A \wedge \neg B) = 0$ với mọi minh họa, nghĩa là $A \wedge \neg B$ là công thức không thỏa được hay tập công thức $\{A, \neg B\}$ không thỏa được. Để chứng minh một tập công thức không thỏa được ta phải chứng minh rằng chúng nhận giá trị chân lý bằng 0 với mọi minh họa. Tránh việc kiểm tra bằng bảng chân lý với độ phức tạp hàm mũ của số mệnh đề trong công thức, chúng ta sẽ sử dụng định lý phân giải. Định lý phân giải còn là cơ sở của thuật toán chứng minh có thể cài đặt được trên máy tính. Trước khi nói đến định lý này, chúng ta tìm hiểu một số khái niệm sau.

Nếu có thể áp dụng luật phân giải cho hai câu tuyển A, B thì hai câu A, B được gọi là hai câu phân giải được. Kết quả nhận được khi phân giải hai câu A, B được gọi là giải thức của chúng, kí hiệu là res(A,B).

Đặc biệt, $\text{Res}(P, \neg P)$ được định nghĩa là câu rỗng.

Giải thức của tập công thức $A_1, ..., A_m$ là tập các câu gồm chính nó và các câu được sinh ra bằng luật phân giải từ các câu A_i và các câu vừa được sinh ra. Kí hiệu là R($A_1, ..., A_m$)

Định lý: Tập câu tuyển là không thỏa được nếu và chỉ nếu giải thức của tập câu đã cho có chứa câu rỗng.

Thủ tục phân giải:

procedure Resolution;

Input: Tap G cac cau tuyen

Output: G thoa duoc hay khong thoa duoc

Begin

repeat

Chọn 2 câu A,B thuộc G

If A, B phân giải được then tính Res(A,B)

If Res(A,B) là câu mới then thêm Res(A,B) vào G

until nhận được câu rỗng hoặc không có câu mới nào xuất hiện;

If nhận được câu rỗng then G không thỏa được

else G thỏa được

end.

Chỉ sử dụng luật phân giải không thể suy ra tất cả các hệ quả logic của một tập công thức, tuy nhiên sử dụng luật phân giải có thể chứng minh một công thức bất kỳ có là hệ quả logic của một tập công thức cho trước hay không bằng phương pháp chứng minh phản chứng.

Thủ tục chứng minh phản chứng bằng luật phân giải.

procedure Refutation_Proof;

input: Tập công thức G, công thức H

output: Công thức H có là hệ quả logic của G hay không?

Begin

Thêm $\neg H$ vào G

Chuyển các công thức trong G thành dạng chuẩn hội

Từ các dạng chuẩn hội thành lập các câu tuyển G'

Áp dụng thủ tục phân giải cho tập câu G'

Nếu G' không thỏa được thì H là hệ quả logic của G

ngược lại H không là hệ quả logic của G.

end;

Ví dụ. Cho G là tập các câu tuyển sau:

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

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

$\neg E \vee C$ (3)

A (4)

E (5)

D (6)

Giả sử ta cần chứng minh P.

Thêm vào G câu

$\neg P$ (7)

Dùng luật phân giải cho (2) và (7) ta được

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

Dùng luật phân giải cho (3) và (8) ta được

$\neg E \vee \neg D$ (9)

Dùng luật phân giải cho (5) và (9) ta được

$\neg D$ (10)

Dùng luật phân giải cho (10) và (6) ta được câu rỗng.

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

$2. LOGIC VỊ TỪ

Sử dụng logic mệnh đề chúng ta đã biểu diễn được tri thức và chứng minh được một khẳng định nào đó có được suy diễn từ tri thức hay không. Tuy nhiên, các câu biểu diễn được trong logic mệnh đề không tổng quát cho một lớp đối tượng cũng như không biểu diễn được các lượng từ với mọi, tồn tại vốn thường được dùng trong thực tế. Đơn giản như câu "Mọi sinh viên đều học chăm", logic mệnh đề đã không biểu diễn được. Logic mệnh đề không có khái niệm biến nên việc thay thế trở nên khó khăn, điều đó không cho phép biểu diễn một lớp các đối tượng trong một miền nào đó mà biến có thể nhận được. Thuộc tính của lớp đối tượng; mối quan hệ, phép tương ứng giữa các lớp đối tượng không biểu diễn được bởi logic mệnh đề.

Logic vị từ mở rộng logic mệnh đề và giải quyết những hạn chế trên. Thuộc tính của lớp đối tượng, mối quan hệ giữa các đối tượng sẽ được biểu diễn bởi các vị từ. Khái niệm biến được đưa vào để biểu diễn cho lớp đối tượng. Ngoài các phép tóan logic như logic mệnh đề, logic vị từ có các lượng từ tồn tại, với mọi để tạo ra câu phong phú hơn.

1. Cú pháp

Kí hiệu: bao gồm các thành phần sau

- Hằng: a,b,c,..
- Biến: x,y,u,v,..
- Vị từ: P(x), Q(x,y),..Vị từ không biến là mệnh đề.
- Hàm: f,g
- Phép toán logic: $\wedge, \vee, \neg, \rightarrow$
- Kí hiệu lượng từ: $\forall, \exists$
- Dấu câu: (,)

Các hạng thức (term): là các biểu thức biểu diễn đối tượng.

- Các hằng là hạng thức.
- Nếu t1,...,tn là các hạng thức và f là hàm thì f(t1,...,tn) là hạng thức.

Công thức phân tử: là các câu đơn.

- Các mệnh đề là công thức phân tử
- Nếu P là vị từ n biến và t1,...,tn là các hạng thức thì P(t1,...,tn) là các công thức phân tử.

Ví dụ: Thích là vị từ hai biến, Thích(An, Java) là một công thức phân tử.

Công thức: từ các công thức phân tử, xây dựng nên bằng các phép toán logic, các lượng từ. Công thức được định nghĩa đệ quy như sau.

- Các công thức phân tử là công thức
- Nếu G và H là công thức thì phủ định của G và G*H là các công thức với * là phép toán logic.
- Nếu G là công thức và x là biến thì $\forall xG, \exists xG$ là các công thức.

2. Ngữ nghĩa

- Minh họa: hằng, biến nhận giá trị trên một miền cụ thể; các vị từ nhận các thuộc tính, quan hệ cụ thể, xác định các hàm cụ thể. Ý nghĩa của các lượng từ như tên gọi của nó.

- Ngữ nghĩa câu đơn (công thức phân tử): P(x) : x là phụ nữ, khi x=Lan với minh họa này P(Lan) biểu diễn cho câu Lan là phụ nữ. Câu đơn có thể xuất hiện các hạng thức (term). Ví dụ Me(y) chỉ đối tượng là mẹ của y nào đó. Câu đơn P(Me(y)) có nghĩa là "Mẹ của y là một phụ nữ".

- Ngữ nghĩa của lượng từ: xác định giá trị bằng hội các giá trị của công thức khi biến nhận mỗi đối tượng trong miền xác định hoặc bằng tuyển ứng với lượng từ với mọi hay tồn tại. Ví dụ nếu lấy miền xác định là sinh viên lớp CNTT thì $\forall xP(x)$ là sai, $\exists xP(x)$ là đúng.

- Ngữ nghĩa câu phức (công thức): Mọi phụ nữ đều thích trang điểm được biểu diễn là $\forall x(P(x) \Rightarrow Thich(x,trangdiem))$ xác định từ ý nghĩa của các phép toán logic và ý nghĩa của các lượng từ. Nếu có một phụ nữ x nào đó không thích trang điểm thì câu này sai, còn lại đều đúng.

Nếu đã có cách xác định được giá trị chân lý của công thức ứng với một minh họa thì ta có thể định nghĩa công thức thỏa được, công thức không thỏa được, công thức hằng đúng, mô hình như trong logic mệnh đề.

3. Chuẩn hóa công thức

- Công thức tương đương: định nghĩa như trong logic mệnh đề. Ngoài ra, liên quan đến các lượng từ ta có các công thức tương đương thông dụng sau:

+ Đổi tên biến: $\forall xG(x) \equiv \forall yG(y); \exists xG(x) \equiv \exists yG(y)$

+ Phủ định công thức chứa lượng từ:

$\neg\forall xG(x) \equiv \exists x\neg G(x)$; $\neg\exists xG(x) \equiv \forall x\neg G(x)$

+ Phân phối lượng từ với phép hội, tuyển.

$\forall x(H(x) \wedge G(x)) \equiv \forall xH(x) \wedge \forall xG(x)$

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

Để đưa công thức trong logic vị từ về dạng chuẩn tắc hội, tức hội của các câu tuyển, sử dụng trong chứng minh sau này, chúng ta thực hiện thủ tục gồm các bước:

- Bỏ các kéo theo;

- Chuyển các phủ định đến các công thức phân tử;

- Bỏ lượng từ tồn tại bằng cách dùng hàm Skolem;

- Đặt tên lại để tránh nghĩa nhập nhằng khi lấy trùng miền xác định;

- Loại bỏ lượng tử với mọi, ngầm hiểu cho toàn miền xác định;

- Chuyển các tuyển đến các công thức phân tử;

- Loại bỏ hội, tách riêng các câu tuyển;

- Đặt lại tên biến cho mỗi câu.

Kết quả, chúng ta thu được cơ sở tri thưc chỉ gồm các câu tuyển.

4. Các luật suy diễn

Phép thế: dùng để thay thế các biến trong công thức bởi các hằng hoặc các hạng thức. Một phép thế θ được kí hiệu

$$\theta = [x_1 \mid t_1, ...x_n \mid t_n]$$

Áp dụng phép thế θ cho công thức *G* nghĩa là chúng ta thay các biến trong *G* bởi các hạng thức tương ứng; kết quả thu được kí hiệu là $G\theta$.

Luật suy diễn: Các luật suy diễn trong logic mệnh đề đều đúng cho logic vị từ. Ngoài ra, ta có các luật sau:

Luật MP tổng quát:

Nếu có một phép thế θ sao cho $P_i\theta = P_i'\theta$ với mọi *i* thì

$$\frac{(P_1 \wedge ... \wedge P_n \rightarrow Q), P_1^{'}, ..., P_n^{'}}{Q'}$$

với $Q' = Q\theta$

Luật phân giải tổng quát:

Nếu có một phép thế θ sao cho $C\theta = D\theta$ với mọi *i* thì

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

Với $A_i' = A_i\theta, B_i' = B_i\theta$.

5. Chứng minh bằng luật phân giải

Trong logic mệnh đề chúng ta đã biết luật phân giải là đầy đủ cho chứng minh bác bỏ. Điều này vẫn còn đúng cho logic vị từ, nghĩa là dùng luật phân giải ta có thể chứng minh rằng H có là hệ quả logic của một tập công thức G hay không.

Mục đích của vấn đề là xét xem kết luận có là điều hợp lý, suy luận được từ các giả thiết hay không.

Nhắc lại rằng để chứng minh bác bỏ bằng luật phân giải: chúng ta chứng minh tập công thức gồm phủ định của kết luận và các giả thiết là tập công thức không thoả được. Để chứng minh điều này, ta sẽ sử dụng định lý phân giải "Một tâp công thức là không thoả được khi và chỉ khi nó suy dẫn ra tập rỗng bằng luật phân giải".

Thủ tục chứng minh bác bỏ:

Procedure P_Resolution;

Input: Tập G, H

Output: H có suy ra được từ G hay không?

Begin

1. Biến đổi các câu trong G và phủ định của H về dạng chuẩn tắc hội
2. Thành lập các câu tuyển
3. Repeat

   Chọn 2 câu A, B

   Tính giải thức Res(A,B)

   Bổ sung vào tập công thức

   Until xuất hiện câu rỗng hoặc không sinh câu mới
4. Nếu xuất hiện câu rỗng thì H đúng, ngược lại thì H sai

End;

Ví dụ: Dùng luật phân giải để chứng minh cuộc sống của Nam ổn định nếu ta có các khẳng định sau:

1. Mọi sv đậu TN và xin được việc làm đều ổn định

2. Mọi Sv học chăm hoặc may mắn đều đậu TN

3. Mọi sinh viên may mắn đều xin được việc làm

4. Nam học không chăm nhưng may mắn

6. Các chiến lược phân giải:

Cái khó nhất trong thủ tục chứng minh trên là việc chọn hai câu nào để tiến hành phân giải sao cho nhanh chóng mang lại kết quả. Việc lựa chọn này được gọi là chiến lược phân giải.

Tập câu ban đầu và các câu phân giải được lần lượt tạo thành một đồ thị gọi là đồ thị phân giải với:

+ Đỉnh là một câu

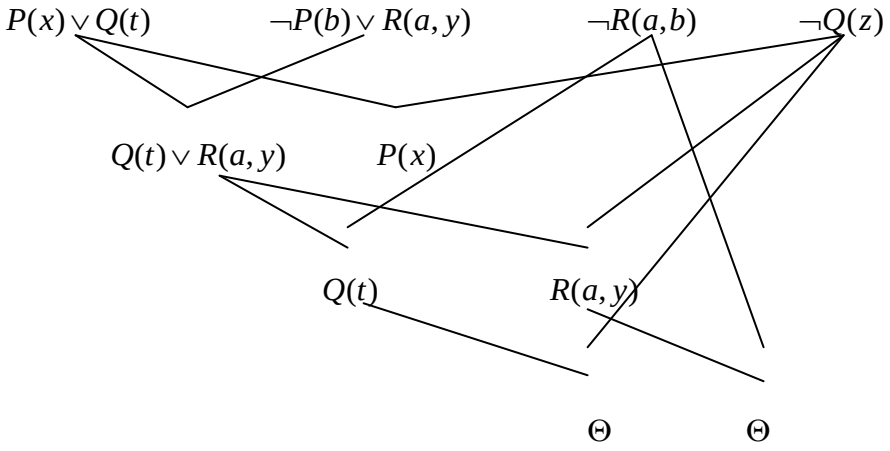
+ Cung: nối các câu đến các giải thức của chúng

+ Nếu tập các câu ban đầu là không phân giải được thì cuối cùng đồ thị là một cây có gốc là câu rỗng. Khi đó ta gọi đồ thị là cây chứng minh. Một đồ thị phân giải có thể có nhiều cây chứng minh.

Sau đây là một số chiến lược phân giải quan trọng.

a) Chiến lược phân giải theo bề rộng:

Ví dụ:



Nguyên tắc:

Mức 0 gồm các câu xuất phát.

Phân giải câu ở mức i với các câu có mức nhỏ hơn i để sinh câu có mức i+1

b) Chiến lược phân giải sử dụng tập hỗ trợ.

Sẽ hiệu quả hơn nếu giảm việc sinh các giải thức mà vẫn không ảnh hưởng việc sinh câu rỗng.

Chọn tập S gồm các câu nào đó.

Chỉ sinh các giải thức từ ít một câu là hậu duệ của S

Sử dụng điều khiển bằng chiến lược theo bề rộng

Sau đây là một số cách để chọn S đàm bảo chiến lược phân giải là đầy đủ (luôn sinh được câu rỗng nếu tập công thức ban đầu là không thoả được):

+ S chỉ gồm các câu không chứa literal âm trong tập ban đầu;

+ S chỉ gồm các câu không chứa literal dương trong tập ban đầu;

+ S chỉ gồm các câu tuyển từ phủ định của H.

$3. BIỂU DIỄN TRI THỨC BẰNG LUẬT VÀ LẬP LUẬN

1. Biểu diễn tri thức bằng luật

Khả năng biểu diễn tri thức của logic vị từ rất mạnh, phạm vi biểu diễn khá rộng. Chúng ta cũng có thủ tục để chuyển các công thức trong logic vị từ thành các câu tuyển và định lý phân giải đã cung cấp thuật toán để chứng minh tự động trong logic vị từ. Hơn nữa, luật phân giải là đầy đủ đối với phương pháp chứng minh phản chứng, nghĩa là chỉ cần sử dụng luật phân giải chúng ta có thể khẳng định một kết luận nào đó đúng hay sai từ tập các giả thiết. Tuy nhiên, thuật toán để chứng minh có độ phức tạp cao. Số các câu sinh ra khi tìm giải thức của tập công thức nói chung là rất lớn. Vì vậy, phương pháp chứng minh phản chứng bằng luật phân giải có ý nghĩa về mặt lý thuyết nhưng chưa hiệu quả khi ứng dụng. Chúng ta cần phương pháp lập luận hiệu quả hơn mặc dù sẽ phải giới hạn miền biểu diễn tri thức.

Chúng ta biết rằng mọi câu tuyển đều có thể chuyển được về câu Lowaski và giới hạn của câu Lowaski là các câu Horn. Như vậy một phần không nhỏ các câu trong logic vị từ tương đương với câu Horn qua các phép chuyển đổi. Hơn nữa, các sự kiện chính là câu Horn không có giả thiết và tri thức là thông tin chắt lọc mang tính quy luật nên đa phần có thể biểu diễn bởi các câu nếu…thì, chính là câu Horn có một hoặc nhiều giả thiết. Vì vậy, chúng ta sẽ giới hạn các câu trong logic vị từ bởi các câu Horn để biểu diễn tri thức và nghiên cứu những phương pháp lập luận hiệu quả trên các câu này.

Cơ sở tri thức bao gồm tập luật R (rules) và tập sự kiện F (facts). Tập luật gồm các câu Horn có dạng $P_1 \wedge ... \wedge P_n \rightarrow Q$.

Ví dụ. Nếu

bệnh nhân ho lâu và

bệnh nhân thường sốt vào buổi chiều

thì

bệnh nhân có khả năng bị bệnh lao.

Ví dụ. Nếu

Tam giác có một góc bằng 60 độ

Tam giác có hai cạnh bằng nhau

thì

Tam giác đó là tam giác đều.

Tập luật gồm các câu Horn có dạng $\rightarrow Q$ tương đương với Q.

Ví dụ: Bệnh nhân ho lâu, tam giác ABC có AB=AC,…

Cơ sở tri thức dùng luật có những đặc trưng sau:

- Mỗi luật mô tả một phần tương đối độc lập của tri thức;

- Dễ dàng cập nhật, sửa đổi, loại bỏ, bổ sung các luật vào cơ sở tri thức mà không ảnh hưởng đến các luật khác.

2. Các phương pháp suy diễn, lập luận

Xét một cơ sở tri thức dùng luật của một hệ tri thức gồm tập luật R và tập sự kiện F. Các sự kiện trong trong F là đầu vào của hệ, được quy ước là đúng. Chúng ta cần chứng minh một khẳng định H nào đó là đúng bằng cách suy diễn từ các sự kiện trong F và tập luật R. Hai phương pháp sau đây giải quyết vấn đề này và

chúng tỏ ra hiệu quả hơn so với chứng minh bằng luật phân giải đối với các ứng dụng.

2.1 Suy diễn tiến

Suy diễn tiến áp dụng qui tắc suy diễn Modus ponens tổng quát. Các sự kiện trong F là đúng. Mỗi bước ta xét một luật trong R. Nếu các sự kiện thỏa mãn các điều kiện của luật này thì ta có một sự kiện mới được suy ra đó chính là phần kết luận của luật, bổ sung nó vào cơ sở sự kiện F. Tập F ngày càng được mở rộng đến khi xuất hiện điều cần chứng minh H trong F, khi đó H đúng; ngược lại nếu H không xuất hiện mà không sinh ra sự kiện mới thì H sai hay H không được chứng minh.

Thủ tục sau đây dùng để chứng minh H từ tập luật R và tập sự kiện ban đầu F. Trong đó, TG là tập các sự kiện thu được tại mỗi thời điểm, ban đầu TG = F. Mỗi bước sử dụng luật, TG sẽ được bổ sung vế phải của luật đó. S=Loc(R,TG) là tập các luật trong R thỏa điều kiện, tức các giả thiết nằm trong tập TG. Sau mỗi lần sử dụng luật r, luật r không cần phải quan tâm trong những lần tiếp theo, ta bỏ r khỏi tập luật, tức R:=R-r. Dạng giả mã của thủ tục như sau:

```
Procedure SDtien;
Input: R, F, H
Output: H đúng/sai
Begin
    TG:=F;
    S:=Loc(R,TG);
    While H∉TG và S≠φ do
        Begin
            r ← S;
            TG:=TG + right(r);
            R:=R-r;
            S:=Loc(R,TG);
        End;
    If  H∈TG then H đúng
    Else H sai;
End;
```

Suy diễn tiến phụ thuộc vào tập các sự kiện, không định hướng được tới vấn đề cần chứng minh, nhiều sự kiện được sinh ra không liên quan đến vấn đề cần chứng minh. Phương pháp phù hợp khi cần thu thập, tổng hợp tri thức. Khi cần ra quyết định, khẳng định một dự đoán nào đó đúng, phương pháp sau đây hiệu quả hơn.

2.2. Suy diễn lùi

Tư tưởng chính của suy diễn lùi là bám theo khẳng định cần chứng minh. Để chứng minh khẳng định ta sẽ kiểm tra các giả thiết cho khẳng định này là đúng. Đến lượt các giả thiết này đóng vai trò của khẳng định cần chứng minh và liên tục như vậy cho đến khi tính đúng đắn của giả thiết là hiển nhiên, chẳng hạn nó chính là giả thiết ban đầu của bài toán.

Một cách hình thức, giả sử ta cần chứng minh q và q là kết luận của các luật trong cơ sở tri thức, có dạng

$$p_1 \wedge ... \wedge p_n \rightarrow q$$

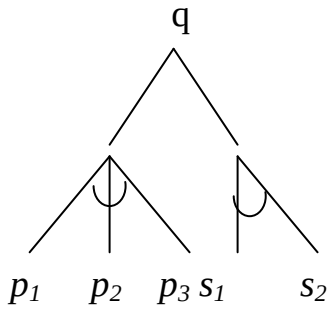
Khi đó ta đi chứng minh các $p_i (i = 1,...,n)$ là đúng. Nếu tất cả các p_i đều đúng thì q đúng. Nếu tồn tại một p_i nào đó sai thì ta chưa kết luận q sai mà phải đi tìm một luật khác nếu có trong cơ sở tri thức, chẳng hạn $s_1 \wedge ... \wedge s_m \rightarrow q$ và công việc thực hiện tương tự như đã làm đối với luật trước đó. Khi không tìm được luật nào có dạng như vậy thì theo hệ này q mới được khẳng định là sai. Tóm lại, phương pháp suy diễn lùi làm việc như kỹ thuật đệ quy.

Ví dụ: Cho cơ sở tri thức có Tập luật gồm: r1: $p_1 \wedge p_2 \wedge p_3 \rightarrow q$; r2: $s_1 \wedge s_2 \rightarrow q$

Tập sự kiện $F = \{p_1, s_1, s_2\}$.

Chứng minh q:

Biểu diễn việc chứng minh q như một đồ thị và hoặc



Thủ tục hay hàm sau đây mô tả cách chứng minh q bằng phương pháp suy diễn lùi. Trong đó, S(q) là tập các luật trong cơ sở tri thức có vế phải là q. Để ý rằng, khi một pi nào đó sai, ta không cần phải đi chứng minh các pi+1 nữa vì các điều kiện của luật kết hợp bởi phép and. Biến OK trong thủ tục để đánh dấu, ghi nhớ thành công hay thất bại trong một pha chứng minh. Rõ ràng không dùng lại r khi đã sử dụng nhưng không thành công, tức S:=S – {r} sau khi sử dụng r.

```
procedure sdlui(q): boolean;
{trả về true nếu q đúng và ngược lại trả về false nếu q sai}
begin
    if q∈F then sdlui:=true
    else if S(q)=[ ] then sdlui:=false
        else
            begin
                S:=S(q);
                while S<>[ ] do
                    begin
                        r ← S;
                        OK:=true;
                        for (l ∈ left(r)) do
                            if sdlui(l)=false then
                                begin
                                    OK:=false;
                                    break; {for}
                                end;
                        if OK then break; {while}
                        S:=S – {r};
                    end;
                if  OK then sdlui:=true else sdlui:=false;
            end;
end;
```

3. Ứng dụng

3.1. Hệ chuyên gia

Hệ chuyên gia là một chương trình máy tính mô phỏng theo vai trò của một chuyên gia trong một lĩnh vực nào đó như y tế, giáo dục, kinh doanh, xã hội,…

MYCIN, DELDRAL, PROSPECTOR là những hệ chuyên gia điển hình tương ứng trong các lĩnh vực y tế, hóa học, địa chất mà chúng ta đã biết. Hiện nay, có rất nhiều hệ chuyên gia lớn, nhỏ được xây dựng và ứng dụng đa dạng trong cuộc sống.

Lõi của một hệ chuyên gia chính là cơ sở tri thức và cơ chế lập luận (suy diễn). Để hoàn chỉnh, những chức năng phụ trợ của hệ cần xây dựng thông thường là bộ giải thích, chức năng cập nhật tri thức và hệ thống giao diện với người sử dụng.

3.2 Lập trình logic.

Ngôn ngữ Prolog là một trong những ngôn ngữ lập trình logic mà cơ sở tri thức được sử dụng là các câu Horn và thủ tục suy diễn được sử dụng là suy diễn lùi.

Cú pháp

Hằng: viết chữ thường a,b,an, nam,...

Biến: viết chữ in đầu tên: X, Y, Sinhvien, Giaovien

Các phép toán:

, chỉ phép and

; chỉ phép or

:- chỉ phép if...then....

not chỉ cho phép phủ định (không dùng trong kết luận)

Một đối tượng trong prolog là một term. Có các dạng term sau đây:

- Các hằng số dạng 3, 3.12, +6, -4
- Các atom: các hằng không phải là số:
  - Dãy một hoặc nhiều kí tự, bắt đầu bằng một chữ thường
  - Một chuỗi đặt trong một cặp nháy đơn.
  - Dấu phép toán phức >=, ++,...
- Các biến: bắt đầu bằng một chữ in hoặc dấu _

- Các term phức hợp: biểu diễn vị từ một hay nhiều biến như thich(nam, bongda), >(3,2), doc(X),...
- Danh sách: viết các phần tử giữa dấu ngoặc vuông; mỗi phần tử có thể có kiểu khác nhau thậm chí có thể là một danh sách.

Ví dụ.

sinhvien(nam).

sinhvien(lan).

sinhvien(hoa).

hocvien(hung).

cunhan(X):-sinhvien(X).

Sau khi consult chương trình trên, thực hiện truy vấn như sau:

? cunhan(nam).

Yes

? cunhan(hung).

No

Câu là thành phần đơn vị của chương trình. Một câu có thể viết trên nhiều dòng. Kết thúc câu là dấu chấm. Có hai loại câu: câu sự kiện và câu luật.

Câu luật viết dưới dạng q:-p1,p2,...,pn. Trong đó q gọi là đầu (head), p1,...,pn gọi là thân của luật.

Vị từ bao gồm tên vị từ và các biến. Một vị từ có thể có nhiều biến.

Một luật có thể bao hàm cả khai báo và thủ tục tiến hành trong luật.

Ví dụ.

Thich(X,Y):- sinhvien(X),thethao(Y),write(X),write(‘ thich ‘).

Sử dụng biến: Biến có thể sử dụng trong phần đầu, thân, hoặc trong goal.

Chú ý rằng tham số của các hàm không được dùng biểu thức, phải đặt biến để thay thế cho biểu thức.

Một số vị từ, hằng thông dụng:

write(.): in giá trị ra màn hình.

nl: xuống dòng.

fail: hằng có giá trị sai.

Một số ví dụ:

```
Chương trình tính N!
gt(0,1).
gt(N,X):- N1 is N-1,
        gt(N1,X1),
        X is X1*N.
```

```
Chương trình tính tổng các số từ 1 đến N
tong(1,1).
tong(N,S):-     N1 is N-1,
            tong(N1,S1),
            S is S1+N.
```

```
Chương trình "sinh viên"
ondinh(X):-totnghiep(X),covieclam(X).
totnghiep(X):-mayman(X).
totnghiep(X):-cham(X).
covieclam(X):-mayman(X).
cham(nam):-fail.
mayman(nam).

Chương trình "Tháp Hà Nội"
chuyen(1,A,B,C):-write(A),write('->'),write(B),nl.
chuyen(N,A,B,C):-N1 is N-1,
            chuyen(N1,A,C,B),
            chuyen(1,A,B,C),
            chuyen(N1,C,B,A).
```

```
Chương trình tính ước chung lớn nhất của 2 số nguyên
ucln(X,0,X).
ucln(0,X,X).
ucln(X,X,X).
ucln(X,Y,Z):-X>Y,T is X-Y,ucln(T,Y,Z).
ucln(X,Y,Z):-X<Y,T is Y-X,ucln(X,T,Z).
```

CHƯƠNG 3. LOGIC MỜ VÀ ỨNG DỤNG

Lý thuyết tập mờ do Lofti. A. Zadeh sáng lập năm 1965 với khái niệm đầu tiên là tập mờ (Fuzzy Set). Ngày nay, các sản phẩm sử dụng công nghệ mờ là khá phổ biến trên toàn thế giới như máy giặt logic mờ, máy ảnh, camera kỹ thuật số, xe hơi, sản xuất xi măng lò đứng, tàu điện tự động ở Nhật Bản,...

Lý thuyết tập mờ lại có những ứng dụng rộng rãi và đem lại những kết quả to lớn như vậy có lẽ do lý thuyết tập mờ đưa ra cơ sở để xử lý được các khái niệm mơ hồ, nhập nhằng của ngôn ngữ tự nhiên và cố gắng mô phỏng hoạt động suy luận của con người.

Những hệ thống mờ (Fuzzy System) hoạt động trên cơ sở logic mờ và tập mờ. Trong chương này, chúng ta trình bày những khái niệm cơ bản của tập mờ và logic mờ; sau đó là các nguyên lý cơ bản của hệ mờ và xem xét một số ứng dụng điển hình của hệ mờ.

$1. TẬP MỜ

Mỗi tập hợp $A \subseteq X$ được xác định bởi một hàm đặc trưng:

$$\lambda_A(x) = \begin{cases} 1 & \text{nếu } x \in A \\ 0 & \text{nếu } x \notin A \end{cases}$$

Đây là một ánh xạ từ X vào $\{0,1\}$. Như vậy có thể biểu diễn A bằng các cặp $(x, \lambda_A(x))$ với mỗi $x \in X$ và $\lambda_A(x) \in \{0,1\}$. Ở đây, ứng với giá trị 0 là phần tử không thuộc A, ứng với giá trị 1 là phần tử thuộc A. Mặt khác, một tập hợp A có thể xác định thông qua một vị từ $P(x)$ mô tả tính chất của các phần tử x trong A. Đơn giản, trong trường hợp trên có thể lấy $P(x) =$"x thuộc A". Khi đó, với mỗi x cụ thể mệnh đề $P(x)$ có giá trị chân lý chính là $\lambda_A(x)$. Giá trị chân lý của mệnh đề chỉ nhận 1 hoặc 0 ứng với đúng hoặc sai trong logic kinh điển.

Tập mờ là mở rộng của khái niệm tập hợp nói trên. Giả sử biên của tập A là "mờ", khi đó sẽ có phần tử hoàn toàn thuộc A, có những phần tử không hoàn toàn thuộc A, nhưng cũng có phần tử thuộc A không chắc lắm. Đánh giá mức độ này ta sẽ dùng một số thực trong đoạn $[0,1]$. Thay cho hàm đặc trưng chúng ta dùng

hàm thành viên (Membership function) - một ánh xạ μ_A từ X vào $[0,1]$. Tương tự như trên, tập mờ A được xác định bởi các cặp $(x, \mu_A(x))$ với $x \in X$ và $\mu_A(x) \in [0,1]$. Giá trị 0 ứng với phần tử không hoàn toàn thuộc A, giá trị 1 ứng với phần tử hoàn toàn thuộc A, những giá trị khác ứng với những phần tử không chắc thuộc A hay không. Mỗi giá trị $\mu_A(x)$ ứng với mỗi x gọi là mức độ thuộc của x trong A. Và giá trị chân lý của mệnh đề $P(x) =$"x thuộc A" có thể nhận không chỉ là 1 hoặc 0 mà còn có thể nhận một giá trị tùy ý trong khoảng (0,1). Đây là điểm khác nhau đầu tiên của logic mờ đối với logic kinh điển. Để tiện cho việc trình bày sau này, từ đây chúng ta sẽ ký hiệu U thay cho không gian X.

Định nghĩa 1.1 Cho U là một tập hợp khác rỗng. Tập mờ A trên U được xác định bởi một hàm $\mu_A : U \to [0,1]$.

Hàm $\mu_A(x)$ được gọi là hàm thành viên biểu diễn mức độ thuộc của x đối với tập mờ A, có thể dùng kí hiệu tương đương là $A(x)$.

Một cách khác, một tập mờ A hoàn toàn xác định bởi các cặp $(x, \mu_A(x))$, tức là

$$A = \{(x, \mu_A(x) \mid x \in U\}$$

Về kí hiệu, nếu U là tập rời rạc gồm các phần tử $\{x_1,...,x_n\}$ thì ta dùng kí hiệu như sau để chỉ cho tập mờ A trên U

$$A = \sum_{i=1}^{n} \mu_A(x_i) / x_i \ ,$$

Và nếu U liên tục thì dùng kí hiệu $A = \int_U \frac{\mu_A(x)}{x} dx$

Ví dụ. A là "số gần bằng 1" biểu diễn $A = \frac{0.1}{-1} + \frac{0.6}{0} + \frac{1}{1} + \frac{0.6}{2} + \frac{0.1}{3}$.

Hoặc A cũng có thể xác định qua hàm thành viên, chẳng hạn $\mu_A(x) = e^{-k(x-1)^2}$ với $k>0$, $k \in \Re$. Để ý rằng chúng ta đang biểu diễn một tính chất mờ "gần bằng 1", tính chất này không thể biểu diễn như những tính chất rõ "bằng 1" được. Như vậy, tập mờ có thể dùng để biểu diễn các khái niệm "mờ".

Các phép tóan trên tập mờ:

Tương tự trong đại số tập hợp, chúng ta cũng có các phép toán trên các tập mờ. Cho A và B là hai tập mờ trên không gian U.

Giao của A và B là một tập mờ trên U, ký hiệu là $A \cap B$ và có hàm thuộc $(A \cap B)(x) = \min(A(x), B(x)).$

Hợp của A và B là một tập mờ trên U, ký hiệu là $A \cup B$ và có hàm thuộc $(A \cup B)(x) = \max(A(x), B(x)).$

Phần bù của A trong U là một tập mờ trên U, ký hiệu là $\overline{A}$ và có hàm thuộc $\overline{A}(x) = 1 - A(x).$

Min, max và 1- là các phép toán cơ bản để định nghĩa các tập mờ mới nhưng nhiều khi không đáp ứng được sự mềm dẻo trong các ngữ cảnh mờ. Các phép toán mở rộng bảo toàn tính chất của các phép toán trên đã được đưa vào sử dụng khi cần thiết.

$t-norm$ là phép toán hai ngôi trên $[0,1]$ bảo toàn tính chất của phép $\min$. Bất kỳ ánh xạ $\varphi : [0,1] \times [0,1] \to [0,1]$ thỏa mãn các tính chất giao hoán, kết hợp, có phần tử đơn vị là 1, không giảm theo từng biến đều được gọi là một $t-norm$.

Ví dụ. $\varphi(a,b) = a * b$; $\varphi(a,b) = \max(a+b-1,0)$ là các $t-norm$

$s-norm$ là phép toán hai ngôi trên $[0,1]$ bảo toàn tính chất của phép max. Bất kỳ ánh xạ $\varphi : [0,1] \times [0,1] \to [0,1]$ thỏa mãn các tính chất giao hoán, kết hợp, có phần tử

đơn vị là 0, không giảm theo từng biến đều được gọi là một $s-norm$ (còn được gọi là $t-conorm$).

Ví dụ. $\varphi(a,b)=a+b-a*b$; $\varphi(a,b)=\min(a+b,1)$ là các $s-norm$.

$F-Negation$ (phép bù mờ) là phép toán một ngôi trên $[0,1]$. Bất kỳ ánh xạ $\varphi:[0,1]\to[0,1]$ thỏa mãn tính chất không tăng, $\varphi(0)=1,\varphi(1)=0$ đều được gọi là một $f-Negation$.

Ví dụ. $N(a)=\dfrac{1-a}{1+\lambda a}$ với $\lambda>-1$.

Trường hợp không gian tham chiếu của các tập mờ thành phần là khác nhau, sự kết hợp được sử dụng bởi tích Đề-các của các tập mờ.

Cho A là tập mờ trên U, B là tập mờ trên V. Tích Đề-các của A và B là tập mờ trên $U\times V$, ký hiệu là $A\times B$ và có hàm thuộc

$$(A\times B)(x_1,x_2)=A(x_1)\otimes A(x_2),$$

ở đây $(x_1,x_2)\in U\times V$ và $\otimes$ là một $t-norm$ tùy chọn.

Có thể mở rộng cho tích Đề-các của hữu hạn các tập mờ một cách tương tự.

Quan hệ mờ và phép hợp thành các quan hệ mờ:

Chúng ta biết rằng một quan hệ rõ từ U tới V là một tập con của tích Đề-các $U\times V$. Nói cách khác, đó chính là một ánh xạ $r:U\times V\to\{0,1\}$. Tương tự như cách mở rộng tập hợp rõ thành tập hợp mờ, chúng ta sẽ mở rộng quan hệ rõ thành quan hệ mờ. Khi đó, ánh xạ r được mở rộng thành $R:U\times V\to[0,1]$ và ánh xạ này xác định một quan hệ mờ từ U tới V.

Như vậy, một quan hệ mờ R từ U tới V là một tập mờ trên không gian tích Đề-các $U\times V$. Ứng với mỗi cặp $(x,y)\in U\times V$, ta có một giá trị $R(x,y)\in[0,1]$.

Quan hệ mờ đóng vai trò quan trọng trong phép lập luận xấp xỉ vì được dùng để biểu thị mối quan hệ nhân quả. Mức độ chắc chắn của một suy luận ứng với cặp nguyên nhân, kết quả được đánh giá bởi một giá trị thuộc đoạn $[0,1]$.

Tiếp theo chúng ta sẽ xét phép hợp thành giữa các quan hệ mờ.

Giả sử R là quan hệ mờ trên $U \times V$ và S là quan hệ mờ trên $V \times W$. Hợp thành giữa R và S là một quan hệ mờ trên $U \times W$, ký hiệu là $R \circ S$ và có hàm thuộc được xác định như sau: $R \circ S(x,z) = \sup_{y \in V}(\min(R(x,y), S(y,z)))$.

Tổng quát có thể thay max bằng một $s-norm$ và min bằng một $t-norm$ tùy ý. Thông dụng nhất là phép hợp thành max-min và max-product.

$2. LOGIC MỜ

Trước tiên chúng ta quan tâm đến một khái niệm cơ bản trong logic mờ, khái niệm "biến ngôn ngữ" do L. A. Zadeh đưa ra năm 1973.

Biến ngôn ngữ

Khác với biến thông thường nhận dữ liệu chính xác, biến ngôn ngữ có thể nhận một từ hay một câu trong ngôn ngữ tự nhiên biểu thị thông tin không chính xác. Chẳng hạn nhiệt độ cao, nhiệt độ trung bình, nhiệt độ rất thấp,…Nhiệt độ được xem là biến và cao, trung bình, rất thấp,…được xem như những giá trị ngôn ngữ mà biến nhiệt độ có thể nhận được (thay vì trước đây biến nhiệt độ chỉ nhận giá trị chính xác bằng số như 30, 90, 100 độ). Để ý rằng, cao, trung bình, rất thấp có thể biểu diễn bằng các tập mờ và vì vậy chính xác hơn có thể xem biến ngôn ngữ là biến nhận giá trị là các tập mờ. Một cách hình thức, Zadeh đã đưa ra định nghĩa sau đây.

Định nghĩa 1.2 Biến ngôn ngữ là một bộ năm $X = (x, T(x), U, R, M)$, trong đó x là tên biến, $T(x)$ là tập các giá trị ngôn ngữ mà biến có thể nhận được, U là không gian tham chiếu của x, với tư cách là biến cơ sở có thể nhận được, R là quy tắc

sinh ra tập $T(x)$, M là quy tắc gán ngữ nghĩa mỗi giá trị trong $T(x)$ với một tập mờ trên U.

Ví dụ. Biến nhiệt độ x với $T(x) = \{cao, TB, thap, ...\}$, $U = [30,100]$, R có thể là quy tắc sinh đại số hoặc văn phạm, M xác lập tùy ứng dụng.

Mệnh đề mờ.

Một mệnh đề mờ đơn giản có dạng " *x is A* ", trong đó x là biến ngôn ngữ và A là một tập mờ trên không gian tham chiếu U.

Ví dụ. "Nhiệt độ cao" là một mệnh đề mờ.

Giá trị chân lý của mệnh đề mờ " *x is A* " là mức độ thuộc của x đối với tập mờ A, tức là $A(x)$ trong đó x với tư cách biến cơ sở đang nhận một giá trị trong U. Giá trị chân lý của mệnh đề mờ còn gọi là mức độ đúng của mệnh đề mờ.

Các mệnh đề mờ phức hợp được tạo thành từ các mệnh đề mờ đơn giản bởi các phép toán logic $\wedge, \vee, \neg, \rightarrow$.

Ví dụ " *x is A* " and " *y is B* ", if " *x is A* " then " *y is B* "

Các phép toán logic chính là các quan hệ giữa các mệnh đề. Theo quan điểm này, giá trị chân lý của các mệnh đề sẽ được tính như sau:

" *x is A* " and " *y is B* " ứng với quan hệ mờ $A \times B$ trên $U \times V$ có giá trị là $T(A(x), B(y))$

" *x is A* " or " *y is B* " ứng với quan hệ mờ $(A \times V) \cup (U \times B)$ trên $U \times V$ có giá trị là $S(A(x), B(y))$

" *x is not A* " ứng với quan hệ mờ $\overline{A}$ trên U có giá trị là $N(A(x))$

If " *x is A* " then " *y is B* " ứng với $R = (\overline{A} \times V) \cup (U \times B)$ có giá trị là $S(N(A(x)), B(y))$ hoặc

ứng với $R = (\overline{A} \times V) \cup (A \times B)$ có giá trị là $S(N(A(x), T(A(x), B(y))$

trong đó T là một $t-norm$, S là một $s-norm$ và N là một $f-negation$ tùy ý.

Mệnh đề kéo theo If " *x is A* " then " *y is B* " là quan trọng nhất vì thường gặp trong các hệ mờ dùng luật. Sử dụng các quan hệ đã được phân tích ở trên và các $t-norm$, $s-norm$, $f-negation$ được chọn ta có các phép kéo theo thường gặp sau:

Phép kéo theo Dienes-Rescher với $R(x, y) = \max(1 - A(x), B(y))$

Phép kéo theo Zadeh với $R(x, y) = \max(1 - A(x), \min(A(x), B(y))$

Đặc biệt từ phép kéo theo Zadeh, với quan điểm ảnh hưởng của phép tóan tại các điểm có giá trị $A(x), B(y)$ đủ lớn, Mamdani đã đưa ra phép kéo theo được sử dụng rất nhiều trong các hệ mờ:

Phép kéo theo Mamdani với $R(x, y) = \min(A(x), B(y))$ hoặc $R(x, y) = A(x) * B(y).$

Lập luận xấp xỉ

Trường hợp đơn giản nhất, bài toán được phát biểu như sau:

If " *x is A* " then " *y is B* "

" *x is A'* "

" *y is B'* "

Trong đó x, y là các biến ngôn ngữ, A, A' là các tập mờ trên U, B, B' là các tập mờ trên V. Yêu cầu xác định *B'*

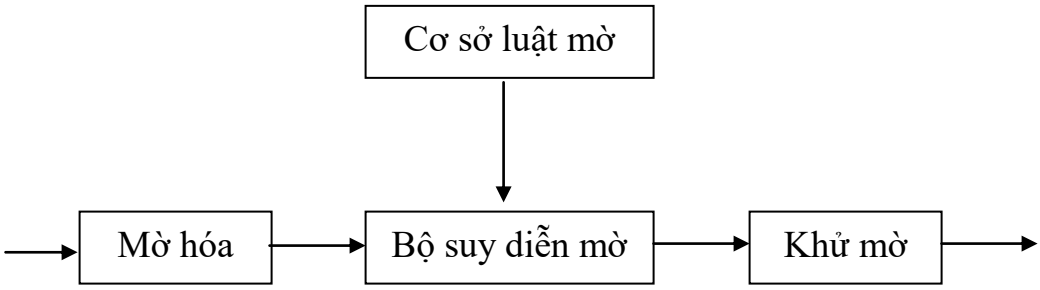
Để tìm *B'* đầu tiên chúng ta xác định quan hệ mờ R cho mệnh đề kéo theo. Mệnh đề " *x is A'* " xác định quan hệ một ngôi trên U chính là A'. Hợp thành hai quan hệ này thu được B', tức là $B' = A' \circ R$. Vì vậy, với T là một $t-norm$ tùy ý, ta có

$$B'(v) = \sup_{u \in U} T(A'(u), R(u, v)), \ v \in V.$$

$3. HỆ MỜ

Hệ mờ là hệ dựa trên tri thức, được sử dụng thành công trong rất nhiều lĩnh vực: hệ chuyên gia y học, quản lý kinh doanh và đặc biệt là trong điều khiển mờ.

Một hệ mờ có các thành phần sau:



+ Cơ sở luật mờ (fuzzy rule base) bao gồm các luật if..then mờ mô tả tri thức chuyên gia về một lĩnh vực nào đó.

+ Bộ suy diễn mờ (fuzzy inference engine) là một phép biến đổi được xây dựng từ cơ sở luật mờ để chuyển đổi một tập mờ đầu vào thành một tập mờ ở đầu ra.

+ Khử mờ (defuzzifier) là khâu biến đổi tập mờ ở đầu ra thành một giá trị trên không gian cơ sở của nó.

+ Mờ hóa (fuzzifier) là khâu biến đổi các giá trị đầu vào thành tập mờ để đưa vào bộ suy diễn.

Công việc quan trọng khi xây dựng hệ mờ là tìm bộ suy diễn mờ ứng với cơ sở luật đã cho. Dễ thấy các cơ sở luật mờ đều có thể chuyển đổi về dạng nhiều đầu vào, một đầu ra. Tổng quát, một cơ sở luật mờ có dạng (1) sau đây:

If x_1 is A_{11} and …and x_n is A_{1n} then y is B_1

……

If x_1 is A_{m1} and …and x_n is A_{mn} then y is B_m

Trong đó A_{ij} là tập mờ trên không gian U_j, $i=1,...,m$, $j=1,...,n$ và B_i là tập mờ trên V, với mọi $i=1,...,m.$

Như vậy, trường hợp tổng quát một cơ sở luật mờ gồm có m luật và luật thứ k $(k=1,...,m)$ có dạng (2):

If x_1 is A_{k1} and …and x_n is A_{kn} then y is B_k

Luật này có thể quy về dạng (3):

If x is A_k then y is B_k

trong đó $x=(x_1,...,x_n)\in U_1\times\cdots\times U_n$, $A_k=A_{k1}\times\cdots\times A_{kn}$.

Có hai phương pháp xây dựng bộ suy diễn từ cơ sở luật mờ:

Phương pháp kết hợp trước suy diễn sau:

+ Xem mỗi luật mờ dạng (3) là một phép kéo theo, tính quan hệ R_k từ A_k, B_k

+ Kết hợp các R_k $(k=1,...,m)$ thu được một quan hệ R theo một trong hai quan điểm: $R=\bigcup_{k=1}^{m}R_k$ hoặc $R=\bigcap_{k=1}^{m}R_k$.

+ Tính $B'=A'\circ R$, với A' là tập mờ đầu vào trên không gian U.

Phương pháp suy diễn trước, kết hợp sau:

+ Từ mỗi luật mờ dạng (3) là một phép kéo theo. Tính quan hệ R_k từ A_k, B_k

+ Với A' là tập mờ đầu vào trên không gian U, tính $B'_k=A'\circ R_k$

+ Kết nhập các B'_k $(k=1,...,m)$ thu được tập mờ B' theo một trong hai quan điểm: $B'=\bigcup_{k=1}^{m}B_k$ hoặc $B'=\bigcap_{k=1}^{m}B_k$.

Như vậy, từ cơ sở luật mờ ta đã xây dựng được bộ suy diễn để thu được tập mờ B' trên V từ đầu vào A' trên U. Để thu được giá trị $y_0 \in V$ đại diện cho B' ta phải tiến hành khâu giải mờ.

Giải mờ. Có nhiều phương pháp giải mờ, thông dụng là hai phương pháp sau

+ Giải mờ lấy max: Lấy y_0 tùy ý thuộc $\left\{ y \in V \mid B'(y) = \sup_{v \in V}(B'(v)) \right\}$

+ Giải mờ lấy trọng tâm: $y_0 = \dfrac{\int_V vB'(v)dv}{\int_V B'(v)dv}$

Đầu vào của bộ suy diễn mờ là tập mờ tuy nhiên đầu vào của hệ mờ lại là giá trị trên không gian cơ sở. Để có được đầu vào cho bộ suy diễn ta phải thực hiện khâu mờ hóa, chuyển đổi giá trị trên không gian cơ sở thành tập mờ.

Mờ hóa. Mỗi $x \in U$ cần được chuyển đổi thành một tập mờ A' trên U. Thông dụng là các phương pháp sau:

+ Mờ hóa đơn thể:

$$A'(u) = \begin{cases} 1 & \text{if } u = x \\ 0 & \text{if } u \neq x \end{cases}$$

+ Mờ hóa tam giác: Chọn b là một hằng số dương,

$$A'(u) = \begin{cases} 1 - \dfrac{|u - x|}{b} & \text{if } |u - x| \leq b \\ 0 & \text{if } |u - x| > b \end{cases}$$

+ Mờ hóa Gauss: Với b là một hằng số dương,

$$A'(u) = e^{-\left(\frac{u-x}{b}\right)^2}$$

Hệ mờ được ứng dụng thành công trong nhiều lĩnh vực. Không những do sử dụng được tri thức chuyên gia bằng ngôn ngữ tự nhiên, khả năng biểu diễn hợp lý các thông tin gần đúng, không chính xác mà còn do sự tồn tại của hệ mờ phù hợp với mỗi bài toán thực tế được khẳng định qua định lý sau.

Định lý Giả sử f là một hàm, $f:U \to V$, $U \subseteq R^n, V \subseteq R$. Nếu f là hàm liên tục trên tập compact U của không gian R^n thì tồn tại một hệ mờ F xấp xỉ với hàm f với độ chính xác tùy ý, tức là với mọi $\varepsilon > 0$ nhỏ tùy ý

$$\sup_{x \in U} | F(x) - f(x) | < \varepsilon$$
