TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

Ts. Nguyễn Thị Thanh Nhàn

Ths. Nguyễn Thu Hương

Ths. Phạm Thị Liên

Ths. Mai Văn Hoàn

Ths. Dương Thị Mai Thương

Ths. Trịnh Văn Hà

Ths. Nguyễn Quang Hiệp

BÀI GIẢNG

TRÍ TUỆ NHÂN TẠO

Tài liệu lưu hành nội bộ

Thái Nguyên, tháng 7 năm 2023

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

TT	Từ viết tắt	Ý nghĩa của từ
1	TTNT	Trí tuệ nhân tạo
2	BFS	Breadth-First Search – Tìm kiếm theo chiều rộng
3	DFS	Depth-First Search - Tìm kiếm theo chiều sâu
4	TNTM	Tác nhân thông minh
5	AI	Artificial Intelligence – Trí tuệ nhân tạo
6	NLP	Natural Language Processing – Xử lý ngôn ngữ tự nhiên
7	CNTT	Công nghệ thông tin
8	ASI	Artificial Super Intelligence – Siêu trí tuệ nhân tạo
9	OCR	Optical Character Recognition – Nhận dạng ký tự quang học

DANH MỤC THUẬT NGỮ

TT	Thuật ngữ	Diễn giải ý nghĩa
1	TTNT	Trí tuệ nhân tạo
2	Tác nhân thông minh	Tác nhân thông minh là bất cứ cái gì tồn tại trong môi trường và hành động một cách thông minh (Một câu hỏi được đặt ra: hành động như thế nào thì được xem là thông minh?)
3	Tác nhân	Tác nhân (agent) là bất cứ cái gì hành động trong môi trường
4	Hệ chuyên gia	**Hệ chuyên gia** (Expert System) là một chương trình máy tính thông minh sử dụng tri thức (knowledge) và các thủ tục suy luận (inference procedures) để giải những bài toán tương đối khó khăn đòi hỏi những chuyên gia mới giải được.
5	Siêu trí tuệ	ASI là một thuật ngữ được sử dụng để biểu thị một thuật toán mạnh hơn con người về tất cả bản chất của trí thông minh
6	Thuật toán	Thuật toán là một tập hợp hữu hạn các lệnh được xác định rõ ràng giúp giải quyết các vấn đề được đề ra một cách rõ ràng.
7	Chiến lược	Chiến lược là chương trình hành động, kế hoạch hành động được thiết kế để đạt được một mục tiêu cụ thể, là tổ hợp các mục tiêu dài hạn và các biện pháp, các cách thức, con đường đạt đến các mục tiêu đó.

MỞ ĐẦU

Trí tuệ nhân tạo (TTNT) là một lĩnh vực của khoa học máy tính với mục tiêu nghiên cứu xây dựng các hệ thống thông minh nhân tạo. Đây là một trong những lĩnh vực được quan tâm nghiên cứu nhiều nhất của khoa học máy tính hiện nay với nhiều kết quả ứng dụng rộng rãi.

Môn học Trí tuệ nhân tạo là môn học trong chương trình đào tạo công nghệ thông tin hệ đại học. Mục tiêu của môn học nhằm giúp sinh viên làm quen với các khái niệm của trí tuệ nhân tạo thông qua việc giới thiệu một số kỹ thuật và ứng dụng cụ thể. Từ đó sinh viên có thể ứng dụng các kỹ thuật trí tuệ nhân tạo trong nhiều bài toán thực tế.

Trong bài giảng này, các nội dung được lựa chọn là những nội dung có tính tiêu biểu, kinh điển của trí tuệ nhân tạo như các phương pháp tìm kiếm, biểu diễn tri thức bằng logic, lập luận.

Nội dung cơ bản gồm 3 phần:

Phần 1 gồm chương 1: là phần giới thiệu tổng quan về trí tuệ nhân tạo bao gồm khái niệm, lịch sử hình thành, sơ lược về những kỹ thuật và ứng dụng tiêu biểu. Nội dung chương không đi quá sâu vào việc định nghĩa chính xác trí tuệ nhân tạo là gì, thay vào đó, người đọc được giới thiệu về những lĩnh vực nghiên cứu chuyên sâu và lịch sử phát triển, trước khi làm quen với nội dung cụ thể trong các chương sau.

Phần 2 bao gồm các chương 2, 3, 4, 5: Nội dung trình bày về *"*Giải quyết vấn đề bằng phương pháp tìm kiếm". Trong phần này, chúng tôi trình bày các phương pháp biểu diễn các vấn đề và các kỹ thuật tìm kiếm, đặc biệt là tìm kiếm kinh nghiệm (heuristic serch), được sử dụng thường xuyên trong nhiều lĩnh vực nghiên cứu của TTNT.

Phần 3 bao gồm các chương 6, 7, 8: Nội dung trình bày về *"*Biểu diễn tri thức và lập luận". Phần này đề cập đến các ngôn ngữ biểu diễn tri thức, đặc biệt là các logic và các phương pháp luận trong mỗi ngôn ngữ biểu diễn tri thức. Các kỹ thuật biểu diễn tri thức và lập luận đóng vai trò quan trọng trong việc thiết kế các hệ thông minh.

Bài giảng được biên soạn từ kinh nghiệm giảng dạy học phần Trí tuệ nhân tạo của các giảng viên tại Trường Đại học Công nghệ thông tin và truyền thông, trên cơ sở tiếp thu phản hồi từ sinh viên và đồng nghiệp. Bài giảng có thể sử dụng làm tài liệu học tập cho sinh viên đại học ngành Công nghệ thông tin và các ngành liên quan.

Trong quá trình biên soạn bài giảng, mặc dù nhóm tác giả đã có nhiều cố gắng song không thể tránh khỏi những thiếu sót. Nhóm tác giả rất mong muốn nhận được ý kiến phản hồi, góp ý cho các thiếu sót để hoàn thiện nội dung bài giảng.

Nhóm tác giả

Chương 1. GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO

Nội dung chính của chương

Trí tuệ nhân tạo (TTNT) là một lĩnh vực nghiên cứu của khoa học máy tính và khoa học tính toán nói chung. Có nhiều quan điểm khác nhau về trí tuệ nhân tạo và do vậy có nhiều định nghĩa khác nhau về lĩnh vực khoa học này. Mục đích của trí tuệ nhân tạo là xây dựng các *thực thể thông minh*.

Nội dung chính của chương này bao gồm:

- Khái niệm về TTNT.
- Vai trò của TTNT.
- Các ứng dụng của TTNT.
- Lịch sử phát triển của TTNT.

Ngoài ra chúng tôi cũng đề cập đến lợi ích, ứng dụng thực tiễn của TTNT đối với sự phát triển của xã hội

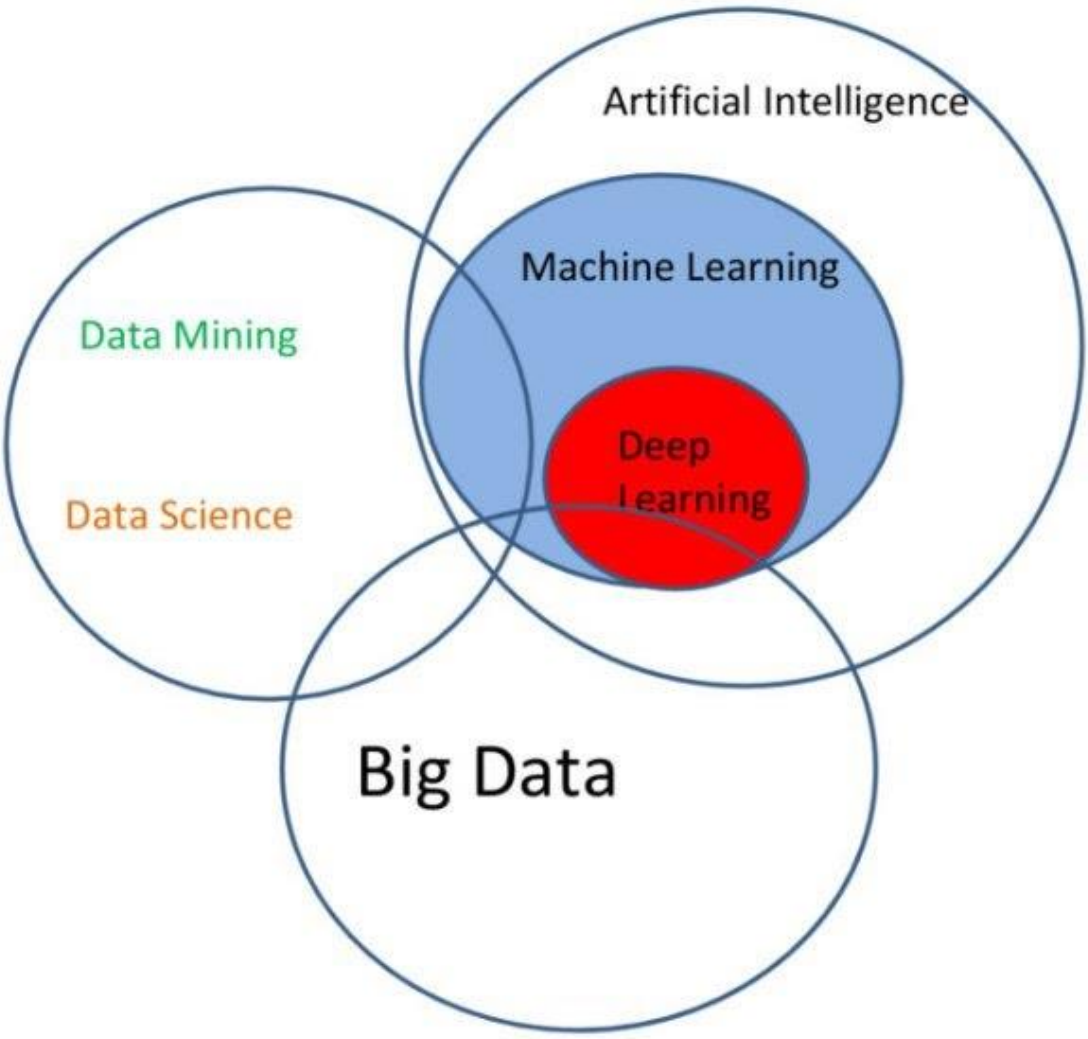
Mục tiêu cần đạt được của chương

Sinh viên hiểu được các kiến thức cơ bản về: Trí tuệ nhân tạo, vai trò của TTNT đối với ngành công nghệ thông tin hiện nay, các ứng dụng của TTNT trong thực tế và lịch sử phát triển của TTNT.

Bài 1: GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO (Số tiết: 3 tiết)

1.1. Trí tuệ nhân tạo

Trí tuệ nhân tạo là một lĩnh vực nghiên cứu của khoa học máy tính và khoa học tính toán nói chung. Có nhiều quan điểm khác nhau về trí tuệ nhân tạo và do vậy có nhiều định nghĩa khác nhau về lĩnh vực khoa học này. Có thể nhìn nhận vị trí và vai trò của TTNT như sau:



Mục đích của trí tuệ nhân tạo là xây dựng các *thực thể thông minh*. Tuy nhiên, do rất khó định nghĩa thế nào là thực thể thông minh nên cũng khó thống nhất định nghĩa trí tuệ nhân tạo. Theo một tài liệu được sử dụng rộng rãi trong giảng dạy trí tuệ nhân tạo hiện nay, các định nghĩa có thể nhóm thành bốn nhóm khác nhau, theo đó, trí tuệ nhân tạo là lĩnh vực nghiên cứu việc xây dựng các hệ thống máy tính có đặc điểm sau:

1) Hệ thống hành động như người

2) Hệ thống có thể suy nghĩ như người

3) Hệ thống có thể suy nghĩ hợp lý

4) Hệ thống hành động hợp lý

Trong số các định nghĩa trên, nhóm thứ hai và ba quan tâm tới quá trình suy nghĩ và tư duy, trong khi nhóm thứ nhất và thứ tư quan tâm chủ yếu tới hành vi. Ngoài ra, hai nhóm định nghĩa đầu xác định mức độ thông minh hay mức độ trí tuệ bằng cách so sánh với khả năng suy nghĩ và hành động của con người, trong khi hai nhóm định nghĩa sau

dựa trên khái niệm suy nghĩ hợp lý và hành động hợp lý. Trong phần phân tích bên dưới ta sẽ thấy suy nghĩ và hành động hợp lý khác với suy nghĩ và hành động như người thế nào.

Sau đây ta sẽ xem xét cụ thể các nhóm định nghĩa trên.

1) Hành động như người

Một số định nghĩa:

"*The art of creating machines that perform functions that require intelligence when performed by people*." (Kurzweil, 1990). Tạm dịch " *Nghệ thuật tạo ra những cỗ máy thực hiện những chức năng đòi hỏi trí thông minh khi thực hiện bởi con người*"

"*The study of how to make computers do things at which, at the moment, people are better*." (Rich and Knight, 1991). Tạm dịch "*Nghiên cứu về cách làm cho máy tính làm những việc mà tại thời điểm đó, con người làm tốt hơn*"

Do con người được coi là động vật có trí tuệ, nên một cách rất tự nhiên là lấy con người làm thước đo khi đánh giá mức độ thông minh của máy tính.

Theo cách định nghĩa này, trí tuệ nhân tạo nhằm tạo ra các hệ thống có hành vi hay hành động tương tự con người, đặc biệt trong những hoạt động có liên quan tới trí tuệ. Để xác định thế nào là hành động như người, có thể sử dụng phép thử Turing.

Phép thử Turing (Turing test): Vào năm 1950, Alan Turing – nhà toán học người Anh có nhiều đóng góp cho khoa học máy tính – đã xây dựng thủ tục cho phép định nghĩa trí tuệ. Thủ tục này sau đó được gọi là phép thử Turing (Turing test), và được thực hiện như sau. Hệ thống được gọi là thông minh, hay có trí tuệ nếu hệ thống có thể hành động tương tự con người trong các công việc đòi hỏi trí tuệ. Trong quá trình thử, một người kiểm tra sẽ đặt các câu hỏi (dưới dạng văn bản) và nhận câu trả lời cũng dưới dạng văn bản từ hệ thống, tương tự khi ta chat hay nhắn tin. Nếu người kiểm tra không phân biệt được câu trả lời là do người thật trả lời hay do máy sinh ra thì hệ thống qua được phép thử và được gọi là có trí tuệ.

Cần lưu ý rằng, phép thử Turing nguyên bản không đòi hỏi có sự tiếp xúc vật lý trực tiếp giữa người kiểm tra và hệ thống bị kiểm tra, do việc tạo ra hệ thống người nhân tạo một cách vật lý được coi là không liên quan tới trí tuệ.

Để qua được phép thử Turing, hệ thống cần có những khả năng:

- *Xử lý ngôn ngữ tự nhiên*: để có thể phân tích, hiểu câu hỏi và tổng hợp câu trả lời trên một ngôn ngữ giao tiếp thông thường như tiếng Việt hay tiếng Anh.
- *Biểu diễn tri thức*: phục vụ việc lưu tri thức và thông tin trong hệ thống.
- *Suy diễn*: sử dụng tri thức để trả lời câu hỏi.

- *Học máy*: để có thể thích nghi với hoàn cảnh và học những mẫu trả lời.

Trong lịch sử trí tuệ nhân tạo đã có những hệ thống như ELIZA được xây dựng nhằm mục đích vượt qua phép thử Turing mà không cần đầy đủ tới cả bốn khả năng trên.

Mặc dù không nhiều người coi mục đích chính của trí tuệ nhân tạo là vượt qua phép thử Turing, một số hệ thống đã xây dựng chuyên cho mục đích này. Gần đây nhất, vào tháng 6 năm 2014, một hệ thống chat tự động có tên là Eugene Goostman do một nhóm nghiên cứu người Nga xây dựng đã giành giải nhất trong cuộc thi phép thử Turing. Sau khi thực hiện một đoạn hội thoại dài 5 phút với hệ thống, 33% giám khảo cho rằng đó là người thực. Một số ý kiến cho rằng Eugene Goostman là hệ thống máy tính đầu tiên vượt qua phép thử Turing.

2) Suy nghĩ như người

Một số định nghĩa:

"*The exciting new effort to make computers think ... machines with minds, in the full and literal sense*." (Haugeland, 1985). Tạm dịch: "*Nỗ lực mới thú vị để làm cho máy tính biết suy nghĩ ... máy móc có trí óc, theo nghĩa đen và đầy đủ*"

"*[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning* ..." (Bellman 1978). Tạm dịch, "*[Việc tự động hóa] các hoạt động mà chúng ta liên kết với suy nghĩ của con người, các hoạt động như ra quyết định, giải quyết vấn đề, học tập ...*"

Theo nhóm định nghĩa này, hành động thông minh chỉ đạt được nếu được dẫn dắt bởi quá trình suy nghĩ tương tự quá trình suy nghĩ của con người.

Những nghiên cứu theo hướng này dựa trên việc nghiên cứu quá trình nhận thức và tư duy của con người, từ đây mô hình hóa và tạo ra những hệ thống có mô hình nhận thức, tư duy tương tự. Việc tìm hiểu quá trình nhận thức, tư duy của người có thể thực hiện theo một số phương pháp như: 1) thực nghiệm về hành vi con người khi suy nghĩ hoặc giải quyết vấn đề; 2) chụp ảnh sóng não, đo tín hiệu điện từ hoặc các tín hiệu khác của não trong quá trình thực hiện các công việc khác nhau; 3) sử dụng các phương pháp nơ ron sinh học khác như kích thích não, giải phẫu não v.v.

Một hệ thống trí tuệ nhân tạo dạng này là hệ thống GPS, viết tắt của General Problem Solver do Newell và Simon trình diễn năm 1961. GPS là chương trình máy tính cho phép giải quyết các bài toán bằng cách mô phỏng chuỗi suy nghĩ của con người khi giải quyết những bài toán như vậy.

Hiện nay, hướng nghiên cứu này được thực hiện trong khuôn khổ *khoa học nhận thức* (cognitive science). Đây là lĩnh vực khoa học liên ngành, kết hợp các mô hình máy tính với phương pháp thực nghiệm tâm lý. Nhiều kết quả nghiên cứu về nhận thức đã được áp dụng trong các mô hình tính toán. Ví dụ, nhiều nghiên cứu về quá trình tiếp nhận tín hiệu ảnh và nhận dạng đối tượng đã được áp dụng trong lĩnh vực thị giác máy. Hay, gần đây, một số nghiên cứu về việc thiết kế các vi mạch có cấu trúc dựa trên hệ thần kinh của người (neuromorphic chips) đã cho kết quả tốt trong các bài toán học máy hoặc xử lý lượng khối lượng dữ liệu lớn.

3) Suy nghĩ hợp lý

Một số định nghĩa:

"*The study of mental faculties through the use of computational models*." (Charniak and McDermott, 1985). Tạm dịch "*Nghiên cứu về các năng lực trí tuệ thông qua việc sử dụng các mô hình tính toán*"

"*The study of the computations that make it possible to perceive, reason, and act*." (Winston, 1992). Tạm dịch: "*Nghiên cứu về các tính toán giúp nhận thức, suy luận và hành động*."

Thực tế cho thấy con người bị chi phối bởi tâm lý, cảm xúc. Do vậy, không phải lúc nào con người cũng suy nghĩ và hành động theo hướng đạt tới kết quả tốt. Từ đây xuất hiện cách tiếp cận theo hướng xây dựng các hệ thống cho phép đạt tới kết quả tốt mà không cần học theo con người. Cách tiếp cận này được gọi là suy nghĩ hợp lý và hành động hợp lý. Trước hết là suy nghĩ hợp lý.

Một cách tiếp cận tiêu biểu của suy nghĩ hợp lý là xây dựng những hệ thống có khả năng lập luận dựa trên việc sử dụng các hệ thống hình thức như logic. Tiền thân của cách tiếp cận này có gốc rễ từ triết học Hy lạp do Aristot khởi xướng. Cơ sở chủ yếu là sử dụng logic để biểu diễn bài toán và giải quyết bằng suy diễn logic. Một số hệ thống logic cho phép biểu diễn mọi loại đối tượng và quan hệ giữa các đối tượng đó. Sau khi đã biểu diễn dưới dạng logic, có thể xây dựng chương trình để giải quyết các bài toán về suy diễn và lập luận.

Khó khăn chủ yếu của cách tiếp cận này là việc mô tả hay biểu diễn bài toán dưới dạng các cấu trúc logic để có thể giải quyết được. Trên thực tế, tri thức và thông tin về bài toán thường có yếu tố không đầy đủ, không chính xác. Ngoài ra, việc suy diễn logic đòi hỏi khối lượng tính toán lớn khi sử dụng trong thực tế và rất khó để triển khai cho các bài toán thực.

4) Hành động hợp lý

Một số định nghĩa:

"*Computational Intelligence is the study of the design of intelligent agents*." (Poole et al., 1998). Tạm dịch "*Trí tuệ tính toán là nghiên cứu về thiết kế của các tác nhân thông minh"*

"*AI . . .is concerned with intelligent behavior in artifacts*." (Nilsson, 1998). Tạm dịch "*AI . . .là quan tâm đến hành vi thông minh trong hiện vật*"

Cách tiếp cận này tập trung vào việc xây dựng các tác tử (agent) có khả năng hành động hợp lý, tức là hành động để đem lại kết quả tốt nhất hoặc kết quả kỳ vọng tốt nhất khi có yếu tố không chắc chắn. Cần lưu ý rằng, hành động hợp lý có thể khác với hành động giống con người: con người không phải lúc nào cũng hành động hợp lý do bị chi phối bởi các yếu tố chủ quan.

Một đặc điểm quan trọng của hành động hợp lý là hành động kiểu này có thể dựa trên việc suy nghĩ (suy luận) hợp lý hoặc không. Trong một số trường hợp, để quyết định hành động thế nào, cần dựa trên việc suy luận hợp lý. Tuy nhiên, trong nhiều tình huống, việc hành động theo phản xạ, chẳng hạn khi gặp nguy hiểm, không đòi hỏi suy diễn phức tạp, nhưng lại cho kết quả tốt hơn. Các hệ thống hành động hợp lý có thể sử dụng cả hai cách tiếp cận dựa trên suy diễn và dựa trên phản xạ để đạt được kết quả tốt.

Hệ thống có khả năng hành động hợp lý có thể bao gồm suy diễn hoặc không, có thể dựa trên cách suy nghĩ giống người hoặc không, có thể bao gồm cả các kỹ thuật dùng để vượt qua phép thử Turing. Do vậy, cách tiếp cận này được coi là tổng quát và bao gồm các cách tiếp cận khác. *Hiện có nhiều ý kiến coi hệ thống trí tuệ nhân tạo là các hệ thống dạng này*.

Tổng kết:

Các phân tích ở trên cho thấy một số cách tiếp cận chính trong định nghĩa trí tuệ nhân tạo:

- Lấy con người làm tiêu chuẩn, nghiên cứu tâm lý và thần kinh học để mô phỏng nhận thức con người, dựa trên đó xây dựng hệ thống trí tuệ nhân tạo.
- Lấy kết quả làm tiêu chuẩn, không nhất thiết phải xây dựng hệ thống mô phỏng người.
- Lấy hành vi và hành động làm mục đích, có thể có quá trình lập luận để hướng dẫn hành động hoặc không.

TTNT là môn khoa học nghiên cứu và mô phỏng các quá trình sáng tạo của con người trên máy tính điện tử, nhằm tạo ra các sản phẩm thông minh có khả năng suy nghĩ, ra quyết định, hoặc hỗ trợ ra quyết định như con người.

1.2. Vai trò của trí tuệ nhân tạo

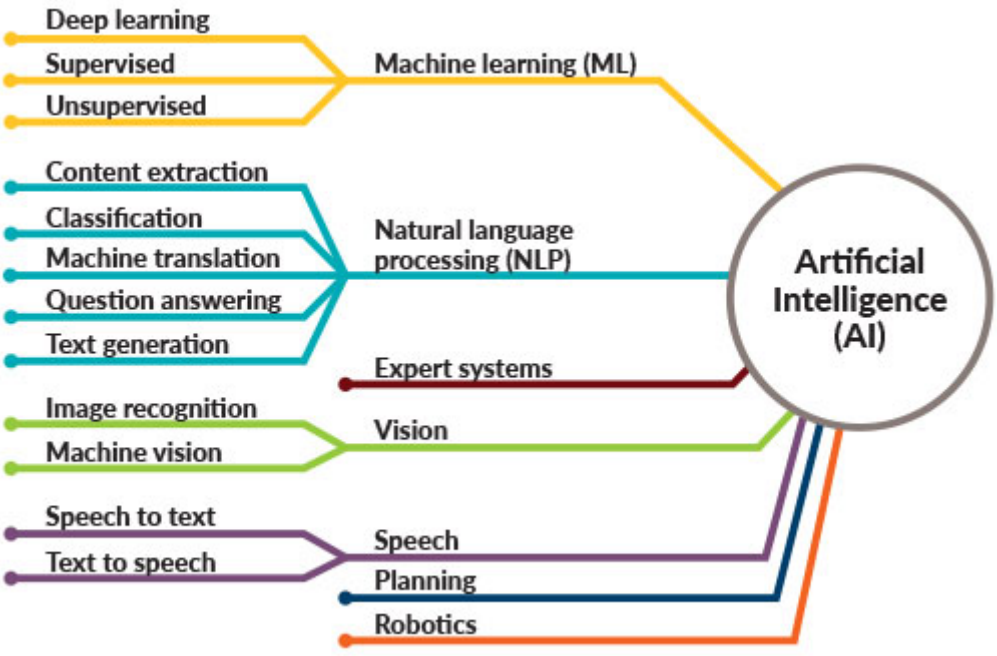
Vai trò của AI là vô cùng quan trọng đối với cuộc sống của chúng ta. AI có thể tiếp cận với con người thông qua nhiều lĩnh vực, ngành nghề khác nhau. Ưu điểm của trí tuệ nhân tạo là khả năng xử lý dữ liệu khoa học hơn, nhanh hơn, hệ thống hơn so với con người. Việc phát triển và đưa các sản phẩm AI tới tay người dùng đúng cách sẽ thúc đẩy mạnh mẽ sự phát triển của toàn nhân loại. Mở ra một thế giới hoàn toàn mới cùng các giải pháp bù đắp cho những vấn đề mà con người không thể giải quyết.

- Theo 1 nghĩa nào đó TTNT tạo nên 1 cách đơn giản để xây dựng lên cấu trúc các chương trình ra quyết định phức tạp đòi hỏi phải dựa trên những tri thức nhất định.
- Các chương trình TTNT hoạt động giống như bộ não của con người tức là nó có thể tích hợp những tri thức mới mà không cần thay đổi lại cách làm việc. Vì vậy những chương trình TTNT có thể dễ dàng cải tiến hơn so với các chương trình truyền thống.
- Khi máy tính được trang bị những phần mềm TTNT kết hợp với môi trường làm việc thì có thể cho phép giải quyết được các bài toán cỡ lớn và phân tán.
- Một số phần mềm TTNT thể hiện tính thích nghi và mềm dẻo đối với các lớp bài toán thuộc nhiều lĩnh vực khác nhau.

1.3. Các ứng dụng của trí tuệ nhân tạo

1.3.1 Các lĩnh vực nghiên cứu

Trí tuệ nhân tạo được chia thành một số lĩnh vực nghiên cứu nhỏ hơn và chuyên sâu nhằm giải quyết những vấn đề khác nhau khi xây dựng một hệ thống trí tuệ nhân tạo. Một số lĩnh vực chuyên sâu được hình thành để giải quyết một lớp bài toán. Một số lĩnh vực chuyên sâu khác tập trung vào các hướng tiếp cận hay các kỹ thuật. Một số lĩnh vực nghiên cứu lại xoay quanh các ứng dụng cụ thể. Trong khi nhiều lĩnh vực nghiên cứu nhỏ có liên quan mật thiết đến nhau thì có nhiều lĩnh vực khác rất xa nhau, cả về mục tiêu, phương pháp và cộng đồng nghiên cứu. Các lĩnh vực phổ biến có thể kể đến như sau:



Thông thường, một hệ thống trí tuệ nhân tạo hoàn chỉnh, làm việc trong việc một môi trường nào đó cần có khả năng: cảm nhận (perception), lập luận (reasoning), và hành động (action). Dưới đây là một số lĩnh vực nghiên cứu của trí tuệ nhân tạo được phân chia theo ba thành phần này.

a) Cảm nhận

Hệ thống cần có cơ chế thu nhận thông tin liên quan tới hoạt động từ môi trường bên ngoài. Đó có thể là camera, cảm biến âm thanh (microphone), cảm biến siêu âm, radar, cảm biến gia tốc, các cảm biến khác. Đó cũng có thể đơn giản hơn là thông tin do người dùng nhập vào chương trình bằng tay. Để biến đổi thông tin nhận được về dạng có thể hiểu được, thông tin cần được xử lý nhờ những kỹ thuật được nghiên cứu và trong khuôn khổ các lĩnh vực sau:

Thị giác máy (computer vision)

Nhận dạng mẫu

Xử lý ngôn ngữ tự nhiên (natural language processing)

b) Lập luận và suy diễn

Sau khi cảm nhận được thông tin về môi trường xung quanh, hệ thống cần có cơ chế để đưa ra được quyết định phù hợp. Quá trình ra quyết định thường dựa trên việc kết hợp thông tin cảm nhận được với tri thức có sẵn về thế giới xung quanh. Việc ra quyết định dựa trên tri thức được thực hiện nhờ lập luận hay suy diễn. Cũng có những trường hợp hệ thống không thực hiện suy diễn mà dựa trên những kỹ thuật khác như tìm kiếm hay tập hợp các phản xạ hoặc hành vi đơn giản.

Thành phần lập luận và ra quyết định được xây dựng dựa trên kỹ thuật từ những lĩnh vực nghiên cứu sau:

Biểu diễn tri thức (knowledge representation)

Tìm kiếm (search)

Lập luận, suy diễn (reasoning hay inference)

Học máy (machine learning)

Deep Learning (Học Sâu)

Lập kế hoạch (planning)

c) Hành động

Cho phép hệ thống tác động vào môi trường xung quanh hoặc đơn giản là đưa ra thông tin về kết luận của mình. Thành phần này được xây dựng dựa trên những kỹ thuật sau. Tổng hợp ngôn ngữ tự nhiên và tiếng nói và Kỹ thuật rô bốt (robotics).

Là kỹ thuật xây dựng các cơ quan chấp hành như cánh tay người máy, tổng hợp tiếng nói, tổng hợp ngôn ngữ tự nhiên. Đây là lĩnh vực nghiên cứu giao thoa giữa cơ khí, điện tử, và trí tuệ nhân tạo. Bên cạnh kỹ thuật cơ khí để tạo ra các cơ chế vật lý, chuyển động, cần có thuật toán và chương trình điều khiển hoạt động và chuyển động cho các cơ chế đó. Chẳng hạn, với cánh tay máy, cần tính toán quỹ đạo và điều khiển cụ thể các khớp nối cơ khí khi muốn di chuyển tay tới vị trí xác định và thực hiện hành động nào đó. Đây là những thành phần của kỹ thuật rô bốt mà trí tuệ nhân tạo có đóng góp chính. Ngoài ra, việc xây dựng những rô bốt thông minh chính là xây dựng các hệ thống trí tuệ nhân tạo hoàn chỉnh.

1.3.2 Một số ứng dụng và thành tựu

a. Các chương trình trò chơi

Xây dựng chương trình có khả năng chơi những trò chơi trí tuệ là lĩnh vực có nhiều thành tựu của trí tuệ nhân tạo. Với những trò chơi tương đối đơn giản như cờ carô hay cờ thỏ cáo, máy tính đã thắng người từ cách đây vài thập kỷ.

Đối với những trò chơi phức tạp hơn, các hệ thống trí tuệ nhân tạo cũng dần đuổi kịp và vượt qua con người. Sự kiện quan trọng thường được nhắc tới là vào tháng 5 năm 1997 chương trình cờ vua *Deep Blue* của IBM đã thắng vô địch cờ vua thế giới lúc đó là Gary Kasparov. Đây là lần đầu tiên máy tính thắng đương kim vô địch cờ vua thế giới. Năm 2015 máy chơi cờ vây AlphaGo của Google DeepMind đạt tới đẳng cấp con người nhờ sử dụng học sâu kết hợp với lấy mẫu cây Monte Carlo. Thách thức trong Poker là không gian của trạng thái lớn và nó không được quan sát đầy đủ (ta không biết

các quân bài của đối thủ). Libratus vượt chất lượng con người trong môn Poker sử dụng các chiến thuật có cấu trúc một cách hiệu quả [Brown & Sandholm, 2017]. Những điều này thể hiện một sự tiến triển ấn tượng trong các trò chơi và tầm quan trọng của các thuật toán nâng cao trong đó.

Một trường hợp tiêu biểu khác là hệ thống *trả lời tự động Watson* cũng của IBM đã chiến thắng hai quán quân của Jeopardy trong trò chơi này vào năm 2011. Jeopardy là trò chơi hỏi đáp trên truyền hình Mỹ, tương tự "Ai là triệu phú" trên truyền hình Việt Nam nhưng trong đó ba người chơi phải thi với nhau không những trả lời đúng mà còn phải nhanh. Watson là hệ thống hỏi đáp do IBM xây dựng dựa trên việc thu thập và phân tích thông tin từ khoảng 200 triệu trang Web, trong đó có toàn bộ Wikipedia. Trong một cuộc đấu với hai cựu quán quân Jeopardy, Watson đã giành thắng lợi và phần thưởng 1 triệu USD. Các kỹ thuật sử dụng trong Watson như thu thập thông tin, phát hiện tri thức, hiểu ngôn ngữ tự nhiên, tìm kiếm, đã được IBM thương mại hóa và có thể sử dụng trong nhiều ứng dụng.

b. Nhận dạng tiếng nói

Nhận dạng tiếng nói là biến đổi từ âm thanh tiếng nói thành các văn bản. Hiện người dùng công cụ tìm kiếm Google có thể đọc vào câu truy vấn thay cho việc gõ từ khóa như trước. Các điện thoại di động thông minh cũng có khả năng nhận dạng giọng nói và trả lời các câu hỏi. Ví dụ điển hình là chương trình trợ giúp Siri trên điện thoại thông minh của Apple (sử dụng công nghệ nhận dạng tiếng nói của hãng Nuance) hay hệ thống Google Now.

Chất lượng nhận dạng giọng nói đang được cải thiện và tiến bộ rất nhanh trong vài năm gần đây. Các hệ thống nhận dạng tiếng nói hiện tại cho phép nhận dạng tới vài chục ngôn ngữ khác nhau và không phụ thuộc vào người nói (ở một mức độ nhất định).

c. Thị giác máy tính

Mặc dù nhiều ứng dụng của thị giác máy tính vẫn chưa đạt tới độ chính xác như người, nhưng trong một số bài toán, thị giác máy tính cho độ chính xác tương đương hoặc gần với khả năng của người. Tiêu biểu phải kể đến các hệ thống nhận dạng chữ in với độ chính xác gần như tuyệt đối, hệ thống nhận dạng tròng mắt, vân tay, mặt người. Những hệ thống dạng này được sử dụng rộng rãi trong sản xuất để kiểm tra sản phẩm, trong hệ thống camera an ninh. Ứng dụng nhận dạng mặt người trên Facebook được dùng để xác định những người quen xuất hiện trong ảnh và gán nhãn tên cho người đó.

Các ứng dụng nhận dạng hiện nay đang được cải thiện nhiều nhờ sử dụng kỹ thuật học sâu (deep learning), trong đó các mạng nơ ron có nhiều lớp được kết nối với

nhau được sử dụng để phát hiện các đặc trưng của đối tượng ở mức từ đơn giản tới phức tạp.

Nhận dạng đối tượng cũng đã tiến một bước dài. Nhận dạng đối tượng trong ảnh là một tác vụ khó trong năm 2010. Trong bảng xếp hạng ImageNet, Lin và các cộng sự năm 2010 đạt được tỷ lệ lỗi top-5 là 28%. Tới 2017. Hu và các cộng sự giảm tỉ lệ lỗi này xuống còn 2,25%. Học sâu đã giúp phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, tự sinh ảnh.

d. Các thiết bị tự lái

Các thiết bị tự lái bao gồm máy bay, ô tô, tầu thủy, thiết bị thám hiểm vũ trụ có thể tự di chuyển mà không có sự điều khiển của người (cả điều khiển trực tiếp và điều khiển từ xa). Hiện ô tô tự lái đang được một số hãng công nghệ và các tổ chức khác nghiên cứu và phát triển, trong đó có những dự án nổi tiếng như xe tự lái của Google và Tesla. Mặc dù tại thời điểm viết sách này mới chỉ có một mẫu xe duy nhất được thương mại hóa dùng cho các khu đi bộ và chỉ có thể chạy với tốc độ khoảng 20 km/giờ nhưng các dự báo cho thấy xe tự lái sẽ được thương mại hóa thành công trong vòng vài năm tới. Các thiết bị tự lái khác bao gồm cả các xem thám hiểm vũ trụ và hành tinh khác như xe thám hiểm sao Hỏa của NASA.

e. Hệ chuyên gia

Là các hệ thống làm việc dựa trên kinh nghiệm và tri thức của chuyên gia trong một lĩnh vực tương đối hẹp nào đó để đưa ra khuyến cáo, kết luận, chuẩn đoán một cách tự động. Các ví dụ gồm:

- MYCIN (1984, Stanford): hệ chuyên gian đầu tiên chẩn đoán bệnh về nhiễm trùng máu và cách điều trị với khả năng tương đương một bác sĩ giỏi trong lĩnh vực này.

- XCON của DEC: hỗ trợ chọn cấu hình máy tính tự động.

f. Xử lý, hiểu ngôn ngữ tự nhiên

Tiêu biểu là các hệ thống dịch tự động như hệ thống dịch của Google, các hệ thống tóm tắt nội dung văn bản tự động. Hệ thống dịch tự động của Google sử dụng các mô hình thống kê xây dựng từ các văn bản song ngữ và các văn bản đơn ngữ. Hệ thống này có khả năng dịch qua lại giữa vài chục ngôn ngữ.

Các hệ thống hỏi đáp được đề cập tới trong phần về trò chơi và nhận dạng tiếng nói cũng thuộc loại ứng dụng xử lý ngôn ngữ tự nhiên. Những hệ thống này sử dụng những thành phần đơn giản hơn như các phân hệ phân tích hình thái, cú pháp, ngữ nghĩa. Các ứng dụng khác như trợ lý ảo, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix.

Các trợ lý thông minh như Apple Siri, Amazon Alexa, hay Google Assistant có khả năng trả lời các câu hỏi thoại với độ chính xác chấp nhận được. Việc này cũng bao gồm các tác vụ đơn giản như bật đèn (hữu ích đối với người tàn tật) tới đặt lịch hẹn cắt tóc và đưa ra các đoạn hội thoại để hỗ trợ các tổng đài chăm sóc khách hàng.

Nhiều kỹ thuật xử lý ngôn ngữ tự nhiên đã được ứng dụng trong các ứng dụng rất thiết thực như các bộ lọc thư rác. Dịch vụ thư điện tử của Google, Microsoft, Yahoo đều có các bộ lọc thư rác với cơ chế học tự động và thích nghi với thay đổi của người phát tán. Khả năng phát hiện thư rác của các hệ thống này là rất cao, gần như tuyệt đối trong một số trường hợp.

ChatGPT là một chatbot do công ty OpenAI của Mỹ phát triển và ra mắt vào tháng 11 năm 2022. ChatGPT được xây dựng dựa trên GPT-3.5- một dòng mô hình ngôn ngữ lớn của OpenAI đồng thời được tinh chỉnh bằng cả hai kỹ thuật học tăng cường lẫn học có giám sát. ChatGPT nhanh chóng thu hút sự chú ý nhờ việc nó có thể hồi đáp chi tiết và trả lời lưu loát trên nhiều lĩnh vực kiến thức khác nhau.

g. Lập kế hoạch, lập thời khóa biểu

Kỹ thuật trí tuệ nhân tạo được sử dụng nhiều trong bài toán lập thời khóa biểu cho trường học, xí nghiệp, các bài toán lập kế hoạch khác. Một ví dụ lập kế hoạch thành công với quy mô lớn là kế hoạch đảm bảo hậu cần cho quân đội Mỹ trong chiến dịch Cơn bão sa mạc tại Iraq đã được thực hiện gần như hoàn toàn dựa trên kỹ thuật trí tuệ nhân tạo. Đây là một kế hoạch lớn, liên quan tới khoảng 50000 thiết bị vận tải và người tại cùng một thời điểm. Kế hoạch bao gồm điểm xuất phát, điểm tới, thời gian, phương tiện và người tham gia sao cho không mâu thuẫn và tối ưu theo các tiêu chí.

Chương trình lập lịch và điều khiển thông minh trên xe tự hành và Robot tự hành của NASA.

h. Rô bốt

Một số rô bốt được xây dựng sao cho có hình dạng tương tự con người và khả năng toàn diện như thị giác máy, giao tiếp bằng ngôn ngữ tự nhiên, khả năng lập luận nhất định, khả năng di chuyển và thực hiện các hành động như nhẩy múa. Các rô bốt này chủ yếu được tạo ra để chứng minh khả năng của kỹ thuật rô bốt thay vì hướng vào ứng dụng cụ thể. Trong số này có thể kể tới rô bốt Asimo, rô bốt Nao.

Bên cạnh đó, một số rô bốt không mô phỏng người nhưng được sử dụng trong đời sống hàng ngày hoặc các ứng dụng thực tế. Ví dụ, rô bốt Roomba của hãng iRobot có khả năng tự động di chuyển trong phòng, tránh vật cản, chui vào các ngóc ngách để lau sạch toàn bộ sàn. Số lượng rô bốt Roomba đã bán lên tới vài triệu bản.

1.3.3. Những vấn đề chưa được giải quyết

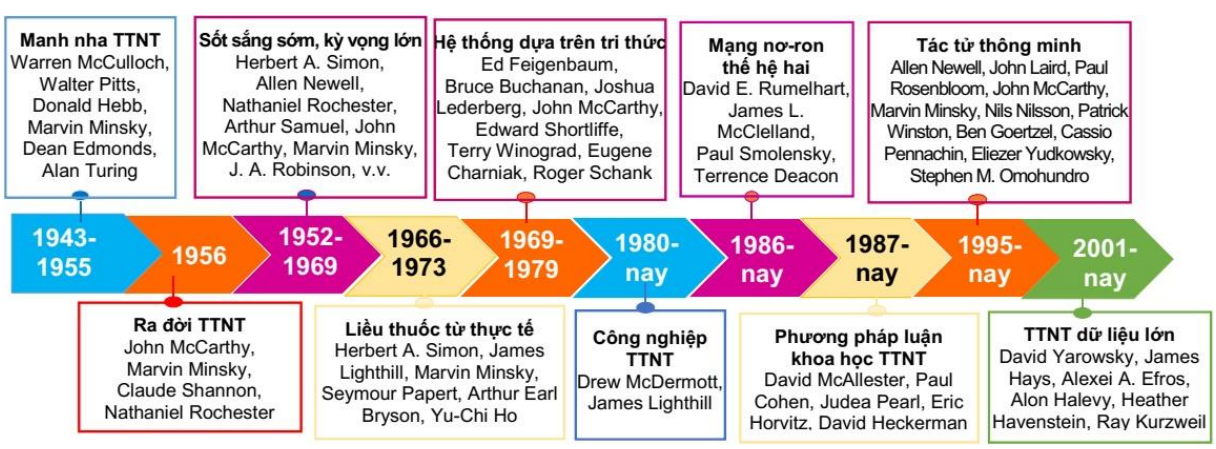
Mặc dù đạt được nhiều thành tựu và có nhiều ứng dụng đáng kể, các hệ thống trí tuệ nhân tạo hiện nay chưa đạt được mức độ *trí tuệ nhân tạo mạnh* (strong AI) hay trí tuệ nhân tạo tổng quát (Artificial General Intelligence). Đây cũng được coi là vấn đề khó nhất và chưa được giải quyết. Trí tuệ nhân tạo mạnh là khái niệm để chỉ khả năng của máy tính thực hiện bất cứ công việc trí tuệ nào mà con người có thể thực hiện. Khái niệm trí tuệ mạnh được sử dụng để phân biệt với *trí tuệ nhân tạo yếu* (weak AI) hay *trí tuệ nhân tạo ứng dụng* (applied AI), tức là dùng máy tính để giải quyết từng bài toán ra quyết định hay lập luận đơn lẻ. Như vậy, trí tuệ nhân tạo mạnh đòi hỏi giải quyết đầy đủ các công việc trí tuệ như người trong khi trí tuệ nhân tạo yếu giải quyết bài toán cụ thể.

Các khó khăn để đạt được trí tuệ nhân tạo tổng quát bao gồm khả năng thị giác máy, xử lý ngôn ngữ tự nhiên, khả năng xử lý các tình hướng mới, tình huống không ngờ tới khi giải quyết các bài toán thực tế. Đây là những lĩnh vực mà máy tính còn thua kém con người. Các hệ thống trí tuệ nhân tạo hiện nay có thể giải quyết tốt bài toán đặt ra trong một phạm vi hẹp.

Tuy nhiên, khi gặp vấn đề thực tế ở phạm vi rộng hơn, hệ thống trí tuệ nhân tạo thường không thể xử lý được các tình huống mới, vượt ra ngoài ngữ cảnh ban đầu của bài toán. Ngược lại, con người có khả năng xử lý tốt hơn nhiều những trường hợp như vậy do có hiểu biết rộng về thế giới xung quanh. Việc trang bị cho máy tính lượng tri thức như con người hiện vẫn là vấn đề chưa được giải quyết.

1.4. Lịch sử phát triển của TTNT

Lịch sử hình thành và phát triển trí tuệ nhân tạo có thể chia thành một số giai đoạn sau (các giai đoạn được chia theo mức độ phát triển và có thể giao nhau về thời gian):



a. Giai đoạn manh nha, tiền khởi đầu (1943-1955)

Mặc dù chưa có khái niệm chính thức về trí tuệ nhân tạo, giai đoạn này ghi nhận một số kết quả có liên quan trực tiếp tới nghiên cứu về trí tuệ nhân tạo sau này:

- Năm 1943, Warren McCulloch và Walter Pitts mô tả mô hình mạng nơ ron nhân tạo đầu tiên, và cho thấy mạng nơ ron nhân tạo có khả năng biểu diễn nhiều hàm số toán học.

- Năm 1950, Alan Turing công bố bài báo nhắc tới trí tuệ máy, trong đó lần đầu tiên mô tả khái niệm phép thử Turing, học máy, thuật toán di truyền, và học tăng cường.

- Năm 1956 được coi là năm chính thức ra đời của khái niệm trí tuệ nhân tạo. Mười nhà nghiên cứu trẻ đã tổ chức một cuộc hội thảo kéo dài hai tháng tại trường đại học Dartmouth với mục đích đặt nền móng đầu tiên cùng với tên gọi chính thức của trí tuệ nhân tạo: "artificial intelligence". Đa số những người tham gia hội thảo này, bao gồm John McCarthy, Marvin Minsky, Allen Newell, và Herbert Simon, sau đó đã trở thành những chuyên gia tiên phong trong các nghiên cứu về trí tuệ nhân tạo. Điểm quan trọng nhất của hội thảo này là đưa ra một số đề xuất và hình dung về trí tuệ nhân tạo. Các đề xuất này vượt ra ngoài khuôn khổ các lĩnh vực nghiên cứu đã hình thành trước đó như lý thuyết điều khiển, vận trù học, lý thuyết ra quyết định. Chính các đề xuất mới này đã đưa trí tuệ nhân tạo thành một lĩnh vực khoa học mới với đối tượng và phương pháp nghiên cứu riêng của mình.

b. Giai đoạn ra đời (1952-1969)

Đây là giai đoạn với nhiều thành tích nhất định của các nghiên cứu trí tuệ nhân tạo, được thể hiện qua một số ví dụ sau:

- Các chương trình Logic Theorist và sau đó là General Problem Solver (GPS) của Newell và Simon, có khả năng chứng minh định lý toán học theo cách tương tự tư duy của con người. Chẳng hạn, trong lớp bài toán mà GPS có thể giải quyết, việc chia bài toán thành các bài toán con và thứ tự các bước giải được tiến hành tương tự với con người khi giải quyết cùng bài toán. Chương trình Logic Theorist đã chứng minh được 38 trong số 52 định lý từ một sách giáo khoa toán, trong đó có định lý về tam giác cân được chứng minh theo cách ngắn hơn cách truyền thống.

- Năm 1952, Arthur Samuel xây dựng một số chương trình chơi cờ đam (checkers). Chương trình có khả năng học và đánh thắng các đối thủ là người chơi cờ đam nghiệp dư. Điểm đặc biệt của chương trình này là khả năng tự học từ kinh nghiệm. Nhờ khả năng học, chương trình có thể thắng cả người đã tạo ra nó.

- Năm 1958, John McCarthy đề xuất ngôn ngữ Lisp, sau này trở thành một trong hai ngôn ngữ thông dụng nhất của trí tuệ nhân tạo.

- Cũng trong những năm này, Minsky khởi xướng việc giải quyết những vấn đề có miền giới hạn hẹp và cần tới tri thức khi giải quyết. Các bài toán có miền hẹp như vậy được gọi là *thế giới nhỏ* (microworld). Chẳng hạn, trong lĩnh vực hẹp về giải tích,

chương trình SAINT do James Slagle viết năm 1963 có thể giải các bài toán tích phân ở mức độ sinh viên năm thứ nhất đại học.

- Mạng nơ ron nhân tạo tiếp tục tiếp tục được phát triển với một số phát minh mới như mạng adalines của Bernie Widrow (1962), Perceptron của Rosenblatt (1962), cho phép giải quyết nhiều bài toán học máy. Trong năm 1962, các nghiên cứu đã chứng minh khả năng học của mạng nơ ron, theo đó có thể thay đổi trọng số kết nối của các nơ ron để phù hợp với bất cứ thong tin đầu vào nào.

- Năm 1965, John Alan Robinson phát minh ra cách chứng minh và suy diễn bằng cách sử dụng phép giải cho logic vị từ. Đây là phương pháp quan trọng, cho phép chương trình máy tính thực hiện lập luận và suy diễn tự động một cách hiệu quả trong trường hợp tri thức được biểu diễn bằng logic.

c. Giai đoạn lạc quan, kỳ vọng lớn

Sau một số thành công ban đầu, đã có những dự báo lạc quan về khả năng xây dựng các hệ thống thông minh trong tương lai gần. Một số nhà khoa học dự báo về khả năng tạo ra các hệ thống thông minh trong vòng 10 tới vài chục năm. Tuy nhiên, sau giai đoạn phát triển mạnh lúc đầu, vào đầu những năm bẩy mươi, các nhà khoa học bắt đầu nhận ra một số khó khăn, đòi hỏi cách tiếp cận thực tế hơn khi nghiên cứu trí tuệ nhân tạo.

Thứ nhất, cách tiếp cận thời kỳ đầu thường sử dụng các biến đổi cú pháp đơn giản và *không quan tâm tới tri thức* về bài toán cần giải quyết. Ví dụ, khi xây dựng các hệ thống dịch máy, nhiều người kỳ vọng có nếu có từ điển và biết cách lắp ghép các từ để tạo thành câu là có thể dịch tự động. Trong khi đó trên thực tế, để dịch được, người dịch cần có hiểu biết nhất định về lĩnh vực được đề cập đến để thể hiện lại nội dung cần dịch trên ngôn ngữ đích. Các dự án dịch tự động từ tiếng Nga sang tiếng Anh do Bộ quốc phòng Mỹ tài trợ đã thất bại do cách tiếp cận đơn giản lúc đầu.

Thứ hai, nhiều hệ thống trí tuệ nhân tạo thời kỳ đầu sử dụng việc tìm kiếm các hành động dẫn tới lời giải. Với bài toán kích thước nhỏ, các kỹ thuật tìm kiếm đơn giản cho kết quả tốt. Tuy nhiên, khi kích thước bài toán tăng lên, số tổ hợp cần xem xét tăng nhanh, vượt khả năng xử lý của máy tính. Hiệu ứng này được gọi là sự "*bùng nổ tổ hợp*" và chỉ được quan tâm đúng mức sau khi lý thuyết về độ phức tạp tính toán ra đời.

Do các khó khăn nói trên, một số báo cáo bi quan về triển vọng trí tuệ nhân tạo đã được trình lên chính phủ các nước như Mỹ, Anh, dẫn tới việc các chính phủ ngừng cấp kinh phí nghiên cứu cho lĩnh vực này. Đây cũng là giai đoạn khó khăn trong lịch sử phát triển trí tuệ nhân tạo, kéo dài trong khoảng 1974 – 1980. Giai đoạn này được gọi là

mùa đông trí tuệ nhân tạo (AI winter), lấy nguyên mẫu từ khái niệm mùa đông hạt nhân, là kết quả mô phỏng khí hậu trái đất lạnh lẽo sau khi xẩy ra chiến tranh hạt nhân.

Một giai đoạn thứ hai cũng được gọi là mùa đông trí tuệ nhân tạo là giai đoạn 1987- 1993. Giai đoạn trì trệ này gắn với sự đi xuống của thị trường các hệ chuyên gia và thất bại của dự án máy tính thế hệ năm do chính phủ Nhật Bản tài trợ.

d. Hệ thống dựa trên tri thức (1969-1979)

Các chương trình trí tuệ nhân tạo xây dựng trong giai đoạn trước có một số hạn chế do không có tri thức về lĩnh vực liên quan, và do vậy không thể giải quyết những bài toán khó, đòi hỏi khối lượng tính toán lớn hoặc nhiều tri thức chuyên sâu. Để khắc phục, giai đoạn này chú trọng tới việc sử dụng nhiều tri thức, thông tin đặc thù cho lĩnh vực hẹp của vấn đề cần giải quyết. Điển hình của hệ thống dựa trên tri thức là các **hệ chuyên gia** (expert systems).

Hệ chuyên gia là các hệ thống có khả năng ra quyết định tương tự chuyên gia trong lĩnh vực hẹp của mình. Hệ thống loại này được xây dựng để giải quyết những vấn đề phức tạp bằng cách lập luận trên tri thức nhận được từ các chuyên gia. Chẳng hạn, một bác sĩ chuyên khoa giỏi có thể mô tả các quy tắc chẩn đoán bệnh trong chuyên khoa của mình. Các quy tắc đó chính là tri thức cần thiết khi chẩn đoán bệnh. Thông thường, tri thức được biểu diễn dưới dạng các luật "*Nếu…Thì…*". Hệ chuyên gia thường gồm *cơ sở tri thức* chứa các luật như vậy và *mô tơ suy diễn* giúp tìm ra lời giải từ tri thức và thông tin về trường hợp đang có. Sau đây là ví dụ một số hệ thống như vậy:

- DENDRAL (năm 1967) là chương trình hệ chuyên gia xây dựng tại trường Stanford, cho phép dự đoán cấu trúc phân tử hữu cơ. Chương trình làm việc dựa trên các luật do chuyên gia trong lĩnh vực hóa học và vật lý cung cấp.

- Một trong các tác giả của DENDRAL, sau đó đã cùng với cộng sự xây dựng MYCIN (1974), hệ chuyên gia cho phép chẩn đoán bệnh nhiễm trùng máu. Với khoảng 450 luật do chuyên gia cung cấp, hệ thống có chất lượng chẩn đoán tương đương bác sĩ giỏi trong lĩnh vực này.

- Việc sử dụng tri thức cũng được sử dụng để giải quyết vấn đề hiểu ngôn ngữ tự nhiên, ví dụ trong hệ thống dịch tự động.

Hệ chuyên gia cho phép giải quyết một phần hạn chế của các hệ thống đơn giản không dựa trên tri thức trước đó. Một số hệ chuyên gia đã được thương mại hoá và đem lại doanh thu cho lĩnh vực trí tuệ nhân tạo.

Cũng trong giai đoạn này, vào năm 1972 Alain Colmerauer đã phát triển ngôn ngữ **Prolog** (viết tắt của logic programming tức là lập trình logic) phục vụ việc biểu diễn

tri thức dưới dạng tương tự logic vị từ và lập luận trên tri thức đó. Prolog cùng với Lisp trở thành hai ngôn ngữ được dùng nhiều nhất trong trí tuệ nhân tạo.

e. Công nghiệp trí tuệ nhân tạo (1980 đến nay)

Sau thành công của những hệ chuyên gia đầu tiên, việc xây dựng hệ chuyên gia được thương mại hóa từ năm 1980 và đặc biệt phát triển cho tới 1988. Sau giai đoạn này, do một số hạn chế của hệ chuyên gia, trí tuệ nhân tạo rơi vào một giai đoạn trì trệ, không có những bước tiến đáng kể (mùa đông trí tuệ nhân tạo thứ hai).

Giai đoạn này cũng đánh dấu sự trở lại của mạng nơ ron nhân tạo sau một thời gian không có các phát minh và ứng dụng đáng kể. Cho đến hiện nay, mạng nơ ron nhân tạo vẫn được sử dụng tương đối nhiều cho học máy và như các chương trình nhận dạng, phân loại tự động. Đặc biệt, trong khoảng gần 10 năm gần đây, các mạng nơ ron nhân tạo nhiều lớp được gọi là *mạng sâu* (deep network) đang được đặc biệt quan tâm do có độ chính xác rất tốt trong các ứng dụng nhận dạng âm thanh, hình ảnh, xử lý ngôn ngữ tự nhiên.

Vào năm 1981, chính phủ Nhật Bản khởi động chương trình xây dựng máy tính thế hệ 5. Mục đích của chương trình là xây dựng các máy tính thông minh chạy trên ngôn ngữ Prolog. Chính phủ Mỹ và Anh cũng có những dự án tương tự nhưng quy mô nhỏ hơn. Mặc dù các chương trình này đều không đạt được mục tiêu đề ra những đã giúp chấm dứt mùa đông trí tuệ nhân tạo lần thứ nhất.

f. Chính thức có phương pháp luận khoa học cho TTNT (1987 đến nay)

Trong giai đoạn trước, chủ đề nghiên cứu chủ yếu có tính thăm dò và tìm kiếm định hướng. Các phương pháp nghiên cứu về trí tuệ nhân tạo cũng không bị giới hạn nhiều trong các lý thuyết có sẵn. Nhiều chủ đề nghiên cứu được đề xuất hoàn toàn mới và phương pháp giải quyết cũng tương đối tự do, không dựa trên các lý thuyết hay kết quả khoa học đã có. Mục tiêu của trí tuệ nhân tạo giai đoạn đầu là vượt ra ngoài khuôn khổ các lĩnh vực nghiên cứu đã có như lý thuyết điều khiển hay thống kê, do vậy nhiều nghiên cứu không đòi hỏi phải dựa trên cơ sở lý thuyết đã phát triển trong các lĩnh vực này.

Bắt đầu từ giai đoạn này (1987), trí tuệ nhân tạo đã có phương pháp nghiên cứu riêng của mình, tuân theo các yêu cầu chung đối với phương pháp nghiên cứu khoa học. Chẳng hạn, kết quả cần chứng minh bằng thực nghiệm, và được phân tích kỹ lưỡng bằng khoa học thống kê. Các vấn đề nghiên cứu cũng gắn nhiều với ứng dụng và đời sống, thay vì những ví dụ mang tính minh hoạ như trong các giai đoạn trước.

Nhiều phát minh trước đây của trí tuệ nhân tạo như mạng nơ ron nhân tạo được phân tích và so sánh kỹ càng với những kỹ thuật khác của thống kê, nhận dạng, và học

máy. Nhờ vậy, các phương pháp không còn mang tính kinh nghiệm thuần túy mà đều dựa trên các cơ sở lý thuyết rõ ràng hơn. Tương tự như vậy, một ví dụ khác là cách tiếp cận với bài toán nhận dạng tiếng nói. Trong giai đoạn đầu, bài toán này được tiếp cận theo một số cách tương đối tự do, hướng vào việc giải quyết một số trường hợp riêng với phạm vi hạn chế, và không dựa trên các lý thuyết đã có. Hiện nay, đa số giải pháp nhận dạng tiếng nói được xây dựng trên các mô hình thống kê như mô hình Markov ẩn (Hidden Markov Models), hay các mạng nơ ron nhiều lớp. Hiệu quả của các giải pháp này có thể giải thích dựa trên lý thuyết thống kê, học máy và các phương pháp tối ưu đã tồn tại và được kiểm chứng từ lâu.

g. Cách tiếp cận dựa trên dữ liệu, sử dụng khối lượng dữ liệu lớn (2001 đến nay)

Trong các giai đoạn trước, việc phát triển trí tuệ nhân tạo chủ yếu tập trung vào xây dựng thuật toán và các hệ thống dựa trên tri thức chuyên gia. Gần đây, do sự xuất hiện của Internet, thương mại điện tử, và một số lĩnh vực khoa học như sinh học phân tử, lượng dữ liệu số hóa được tạo ra tăng rất nhanh. Nhiều nghiên cứu cũng cho thấy việc sử dụng dữ liệu hợp lý quan trọng hơn việc xây dựng các thuật toán phức tạp. Một trong những ví dụ là tiến bộ của hệ thống dịch tự động của Google, được xây dựng dựa trên việc thống kê số lượng lớn các văn bản đơn ngữ và song ngữ, thay vì sử dụng luật và quy tắc ngữ pháp như trước đây.

Trong vài năm gần đây, xu hướng sử dụng *dữ liệu lớn* (big data), tức là các kỹ thuật ra quyết định dựa trên việc phân tích lượng lớn dữ liệu với bản chất đa dạng và thay đổi nhanh theo thời gian đang được coi là một trong những xu hướng có ảnh hưởng quan trọng tới sự phát triển và ứng dụng của trí tuệ nhân tạo. Nhiều ứng dụng quan trọng dựa trên dữ liệu lớn như các hệ thống hỗ trợ khuyến nghị của Amazon, ứng dụng trợ giúp Siri của Apple, chương trình dịch tự động và nhận dạng giọng nó của Google, chương trình trò chơi Watson của IBM là những ví dụ thành công điển hình của xu hướng này.

Thành công của việc sử dụng dữ liệu lớn cho thấy có thể giải quyết một vấn đề quan trọng của trí tuệ nhân tạo: đó là việc thu thập đủ lượng tri thức cần thiết cho hệ thống. Thay vì thu thập bằng tay như trước đây, tri thức có thể được tổng hợp, hay "học" từ dữ liệu. Đây cũng là hứa hẹn cho một giai đoạn phát triển mạnh các ứng dụng của trí tuệ nhân tạo.

Nội dung bài 1 đã trình bày các khái niệm cơ bản, các hiểu biết về TTNT cũng như vài trò và ứng dụng của TTNT trong thực tiễn.

Câu hỏi cuối chương

Câu 1: Trình bày khái niệm về TTNT.

Câu 2: Trình bày khái niệm trí tuệ nhân tạo theo hướng hệ thống hành động như người.

Câu 3: Trình bày khái niệm trí tuệ nhân tạo theo hướng hệ thống có thể suy nghĩ như người.

Câu 4: Trình bày khái niệm trí tuệ nhân tạo theo hướng hệ thống có thể suy nghĩ hợp lý.

Câu 5: Trình bày khái niệm trí tuệ nhân tạo theo hướng hệ thống hành động hợp lý.

Câu 6: Vai trò của TTNT đối với ngành CNTT là gì?

Câu 7: Trình bày lịch sử phát triển và hình thành TTNT?

Câu 8: Các ứng dụng của TTNT là gì?

Câu 9: Lợi ích của TTNT?

Câu 10: Em hãy trình bày 5 ứng dụng gần đây nhất của TTNT?

CHƯƠNG 2: CÁC CHIẾN LƯỢC TÌM KIẾM MÙ

Nội dung chính của chương

Chương này sẽ trình bày về kỹ thuật giải quyết vấn đề bằng tìm kiếm, còn được gọi là tìm kiếm trong không gian trạng thái. Đây là các phương pháp được dùng phổ biến trong một lớp lớn các bài toán và là lớp kỹ thuật giải quyết vấn đề quan trọng, được nghiên cứu và ứng dụng nhiều của trí tuệ nhân tạo. Trong phạm vi bài giảng, ta sẽ xem xét cách phát biểu một vấn đề dưới dạng bài toán tìm kiếm, trước khi chuyển sang các giải thuật đã được phát triển để giải quyết bài toán này. Các giải thuật tìm kiếm được chia thành ba nhóm lớn: tìm kiếm mù (không có thông tin), tìm kiếm heuristics (có thông tin), và tìm kiếm cục bộ. Giải thuật thuộc nhóm hai và nhóm ba, đặc biệt là nhóm ba, được sử dụng cho các bài toán thực tế với kích thước lớn.

Giải quyết vấn đề bằng tìm kiếm

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

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

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

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

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

- Các kỹ thuật tìm kiếm mù, trong đó chúng ta không có hiểu biết gì về các đối tượng để hướng dẫn tìm kiếm mà chỉ đơn thuần là xem xét theo một hệ thống nào đó tất cả các đối tượng để phát hiện ra đối tượng cần tìm.
- Các kỹ thuật tìm kiếm kinh nghiệm (tìm kiếm heuristic) trong đó chúng ta dựa vào kinh nghiệm và sự hiểu biết của chúng ta về vấn đề cần giải quyết để xây dựng nên hàm đánh giá hướng dẫn sự tìm kiếm.

- Các kỹ thuật tìm kiếm tối ưu.
- Các phương pháp tìm kiếm có đối thủ, tức là các chiến lược tìm kiếm nước đi trong các trò chơi hai người, chẳng hạn cờ vua, cờ tướng, cờ carô.

Mục tiêu cần đạt được của chương

Trong chương này, sinh viên cần hiểu được vấn đề tìm kiếm trong không gian trạng thái, nghiên cứu các chiến lược tìm kiếm mù (blind search): tìm kiếm theo bề rộng (breadth-first search) và tìm kiếm theo độ sâu (depth-first search). Hiệu quả của các phương pháp tìm kiếm này.

BÀI 2: CÁC CHIẾN LƯỢC TÌM KIẾM MÙ (Số tiết: 3 tiết)

2.1. Biểu diễn vấn đề trong không gian trạng thái

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

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

Một phạm vi rộng lớn các vấn đề, đặc biệt các câu đố, các trò chơi, có thể mô tả bằng cách sử dụng khái niệm trạng thái và toán tử (phép biến đổi trạng thái). Chẳng hạn, một khách du lịch có trong tay bản đồ mạng lưới giao thông nối các thành phố trong một vùng lãnh thổ (*hình 2.1*), du khách đang ở thành phố A và anh ta muốn tìm đường đi tới thăm thành phố B. Trong bài toán này, các thành phố có trong các bản đồ là các trạng thái, thành phố A là trạng thái ban đầu, B là trạng thái kết thúc. Khi đang ở một thành phố, chẳng hạn ở thành phố D anh ta có thể đi theo các con đường để nối tới các thành phố C, F và G. Các con đường nối các thành phố sẽ được biểu diễn bởi các toán tử. Một toán tử biến đổi một trạng thái thành một trạng thái khác. Chẳng hạn, ở trạng thái D sẽ có ba toán tử dẫn trạng thái D tới các trạng thái C, F và G. Vấn đề của du khách bây giờ sẽ là tìm một dãy toán tử để đưa trạng thái ban đầu A tới trạng thái kết thúc B.

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

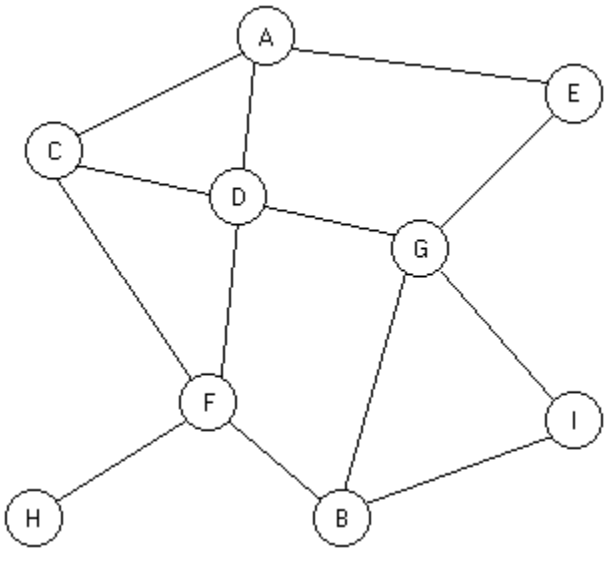
Như vậy muốn biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:

- Trạng thái ban đầu.
- Một tập hợp các toán tử. Trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.

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

Ta sẽ ký hiệu không gian trạng thái là U, trạng thái ban đầu là u_0 ($u_0 \in U$). Mỗi toán tử R có thể xem như một ánh xạ R: U→U. Nói chung R là một ánh xạ không xác định khắp nơi trên U.

- Một tập hợp T các trạng thái kết thúc (trạng thái đích). T là tập con của không gian U. Trong vấn đề của du khách trên, chỉ có một trạng thái đích, đó là thành phố B. Nhưng trong nhiều vấn đề (chẳng hạn các loại cờ) có thể có nhiều trạng thái đích và ta không thể xác định trước được các trạng thái đích. Nói chung trong phần lớn các vấn đề hay, ta chỉ có thể mô tả các trạng thái đích là các trạng thái thỏa mãn một số điều kiện nào đó.



Hình 2.1. Mạng lưới giao thông nối các thành phố Tìm đường đi từ A đến B

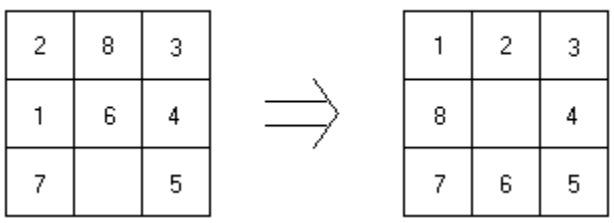
Khi chúng ta biểu diễn một vấn đề thông qua các trạng thái và các toán tử, thì việc tìm nghiệm của bài toán được quy về việc tìm đường đi từ trạng thái ban đầu tới trạng thái đích. (Một đường đi trong không gian trạng thái là một dãy toán tử dẫn một trạng thái tới một trạng thái khác).

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

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

Ví dụ 1: Bài toán 8 số.

Chúng ta có bảng 3x3 ô và tám quân mang số hiệu từ 1 đến 8 được xếp vào tám ô, còn lại một ô trống, chẳng hạn như trong *hình 2.2* bên trái. Trong trò chơi này, bạn có thể chuyển dịch các quân ở cạch ô trống tới ô trống đó. Vấn đề của bạn là tìm ra một dãy các chuyển dịch để biến đổi cảnh huống ban đầu thành một cảnh huống xác định nào đó, chẳng hạn cảnh huống trong hình



Hình 2.2. Trạng thái ban đầu và trạng thái kết thúc của bài toán 8 số

Trong bài toán này, trạng thái ban đầu là cảnh huống ở bên trái *hình 2.2*, còn trạng thái kết thúc ở bên phải hình *2.2.* Tương ứng với các quy tắc chuyển dịch các quân, ta có bốn toán tử: ***up*** (đẩy quân lên trên), ***down*** (đẩy quân xuống dưới), ***left*** (đẩy quân sang trái), ***right*** (đẩy quân sang phải). Rõ ràng là, các toán tử này chỉ là các toán tử bộ phận; chẳng hạn, từ trạng thái ban đầu (*hình 2.2 bên trái*), ta chỉ có thể áp dụng các toán tử ***down, left, right***.

Qua ví dụ trên, việc tìm ra một biểu diễn thích hợp để mô tả các trạng thái của vấn đề là khá dễ dàng và tự nhiên. Song trong nhiều vấn đề việc tìm hiểu được biểu diễn thích hợp cho các trạng thái của vấn đề là hoàn toàn không đơn giản. Việc tìm ra dạng biểu diễn tốt cho các trạng thái đóng vai trò hết sức quan trọng trong quá trình giải quyết một vấn đề. Có thể nói rằng, nếu ta tìm được dạng biểu diễn tốt cho các trạng thái của vấn đề, thì vấn đề hầu như đã được giải quyết.

Ví dụ 2: Bài toán triệu phú và kẻ cướp.

Có ba nhà triệu phú và ba tên cướp ở bên bờ tả ngạn một con sông, cùng một chiếc thuyền chở được một hoặc hai người. Hãy tìm cách đưa mọi người qua sông sao cho không để lại ở bên bờ sông kẻ cướp nhiều hơn triệu phú. Đương nhiên trong bài toán này, các toán tử tương ứng với các hành động chở 1 hoặc 2 người qua sông. Nhưng ở đây ta cần lưu ý rằng, khi hành động xảy ra (lúc thuyền đang bơi qua sông) thì ở bên bờ sông thuyền vừa dời chỗ, số kẻ cướp không được nhiều hơn số triệu phú. Tiếp theo ta cần quyết định cái gì là trạng thái của vấn đề. Ở đây ta không cần phân biệt các nhà triệu phú và các tên cướp, mà chỉ số lượng của họ ở bên bờ sông là quan trọng. Để biểu diễn

các trạng thái, ta sử dụng bộ ba *(a, b, k)*, trong đó *a* là số triệu phú, *b* là số kẻ cướp ở bên bờ tả ngạn vào các thời điểm mà thuyền ở bờ này hoặc bờ kia, *k = 1* nếu thuyền ở bờ tả ngạn và *k = 0* nếu thuyền ở bờ hữu ngạn. Như vậy, không gian trạng thái cho bài toán triệu phú và kẻ cướp được xác định như sau:

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

2.2 Các chiến lược tìm kiếm

Như ta đã thấy trong mục 2.1, để giải quyết một vấn đề bằng tìm kiếm trong không gian trạng thái, đầu tiên ta cần tìm dạng thích hợp mô tả các trạng thái cảu vấn đề. Sau đó cần xác định:

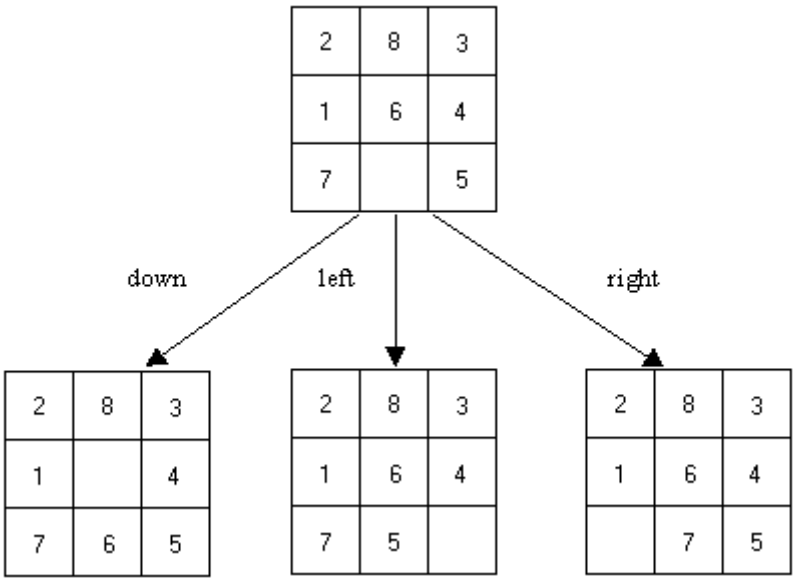
- Trạng thái ban đầu.
- Tập các toán tử.
- Tập T các trạng thái kết thúc. (T có thể không được xác định cụ thể gồm các trạng thái nào mà chỉ được chỉ định bởi một số điều kiện nào đó).

Giả sử *u* là một trạng thái nào đó và R là một toán tử biến đổi *u* thành *v*. Ta sẽ gọi *v* là trạng thái kề *u*, hoặc *v* được sinh ra từ trạng thái *u* bởi toán tử R. Quá trình áp dụng các toán tử để sinh ra các trạng thái kề *u* được gọi là phát triển trạng thái *u*. Chẳng hạn, trong bài toán toán số, phát triển trạng thái ban đầu (*hình 2.2 bên trái*), ta nhận được ba trạng thái kề

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

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

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



Hình 2.3. Phát triển một trạng thái

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

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

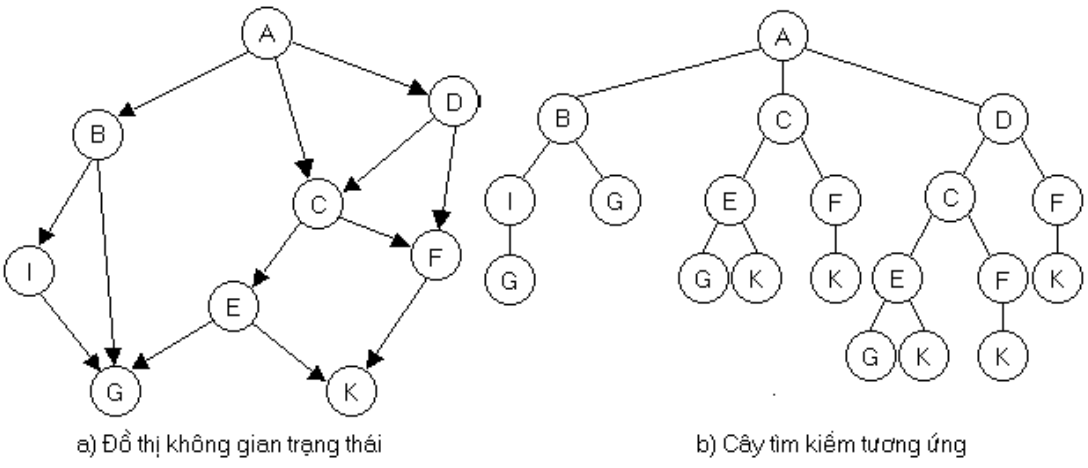
- ***Tìm kiếm kinh nghiệm (tìm kiếm heuristic)***. Trong rất nhiều vấn đề, chúng ta có thể dựa vào sự hiểu biết của chúng ta về vấn đề, dựa vào kinh nghiệm, trực giác, để đánh giá các trạng thái. Sử dụng sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm: trong quá trình phát triển các trạng thái, ta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

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

Cây tìm kiếm

Chúng ta có thể nghĩ đến quá trình tìm kiếm như quá trình xây dựng ***cây tìm kiếm***. Cây tìm kiếm là cây mà các đỉnh được gắn bởi các trạng thái của không gian trạng thái. Gốc của cây tìm kiếm tương ứng với trạng thái ban đầu. Nếu một đỉnh ứng với

trạng thái u, thì các đỉnh con của nó ứng với các trạng thái v kề u. Hình 2.4a là đồ thị biểu diễn một không gian trạng thái với trạng thái ban đầu là A, hình 2.4b là cây tìm kiếm tương ứng với không gian trạng thái đó.



Hình 2.4. Đồ thị không gian trạng thái và cây tìm kiếm tương ứng

Mỗi chiến lược tìm kiếm trong không gian trạng thái tương ứng với một phương pháp xây dựng cây tìm kiếm. Quá trình xây dựng cây bắt đầu từ cây chỉ có một đỉnh là trạng thái ban đầu. Giả sử tới một bước nào đó trong chiến lược tìm kiếm, ta đã xây dựng được một cây nào đó, các lá của cây tương ứng với các trạng thái chưa được phát triển. Bước tiếp theo phụ thuộc vào chiến lược tìm kiếm mà một đỉnh nào đó trong các lá được chọn để phát triển. Khi phát triển đỉnh đó, cây tìm kiếm được mở rộng bằng cách thêm vào các đỉnh con của đỉnh đó. Kỹ thuật tìm kiếm theo bề rộng (theo độ sâu) tương ứng với phương pháp xây dựng cây tìm kiếm theo bề rộng (theo độ sâu).

2.3 Các chiến lược tìm kiếm mù

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

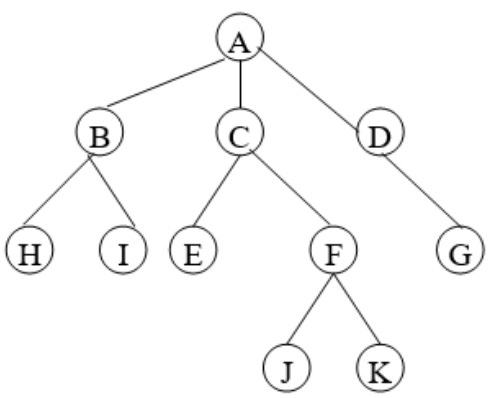
Chúng ta sử dụng danh sách L để lưu các trạng thái đã được sinh ra và chờ được phát triển. Mục tiêu của tìm kiếm trong không gian trạng thái là tìm đường đi từ trạng thái ban đầu tới trạng thái đích, do đó ta cần lưu lại vết của đường đi. Ta có thể sử dụng hàm father để lưu lại cha của mỗi đỉnh trên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

2.3.1 Tìm kiếm theo chiều rộng

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

procedure *Breadth_First_Search***;**
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
2. **loop do**
2.1 **if** *L rỗng* **then**
{*thông báo tìm kiếm thất bại***; stop};**
*2.2 Loại trạng thái u ở đầu danh sách L***;**
2.3 **if** *u là trạng thái kết thúc* **then**
{*thông báo tìm kiếm thành công***; stop};**
2.4 **for** *mỗi trạng thái v kề u* **do {**
*Đặt v vào cuối danh sách L***;**
*father(v) <- u***}**
end;

Ví dụ 3: Cho đồ thị, với u_0 = A, T = {I, E, K}

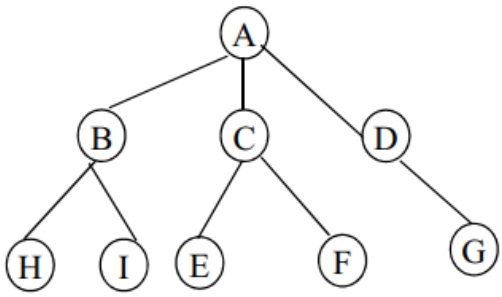


Áp dụng thuật toán tìm kiếm theo chiều rộng với đồ thị trên (trình bày từng bước; vẽ cây tìm kiếm).

Giải:

Lần lặp	L=∅	u	u∈T	v	L
0					A
1	False	A	False	B, C, D	B,C D
2	False	B	False	H, I	C, D, H, I
3	False	C	False	E, F	D, H, I, E, F
4	False	D	False	G	H, I, E, F,G
5	False	H	False		I, E, F, G
6	False	I∈T	True		

Quá trình tìm kiếm thành công, cây tìm kiếm là:



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

- Trong tìm kiếm theo chiều rộng, trạng thái nào được sinh ra trước sẽ được phát triển trước, do đó danh sách L được xử lý như hàng đợi. Trong bước 2.3, ta cần kiểm tra xem u có là trạng thái kết thúc hay không. Nói chung các trạng thái kết thúc được xác định bởi một số điều kiện nào đó, khi đó ta cần kiểm tra xem u có thỏa mãn các điều kiện đó hay không.

- Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái ban đầu tới trạng thái đích), thì thuật toán tìm kiếm theo chiều rộng sẽ tìm ra nghiệm, đồng thời đường đi tìm được sẽ là ngắn nhất. Trong trường hợp bài toán vô nghiệm và không gian trạng thái hữu hạn, thuật toán sẽ dừng và cho thông báo vô nghiệm.

Đánh giá tìm kiếm theo chiều rộng

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

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

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

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

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

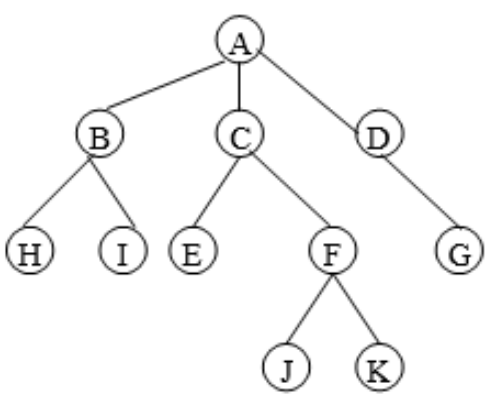
Để thấy rõ tìm kiếm theo chiều rộng đòi hỏi thời gian và không gian lớn tới mức nào, ta xét trường hợp nhân tố nhánh b = 10 và độ sâu d thay đổi. Giả sử để phát hiện và kiểm tra 1000 trạng thái cần 1 giây, và lưu giữ 1 trạng thái cần 100 bytes. Khi đó thời gian và không gian mà thuật toán đòi hỏi được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

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

2.3.2 Tìm kiếm theo độ sâu

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

Ví dụ 4: Cho đồ thị với u_0 = A, T = {I, E, G}

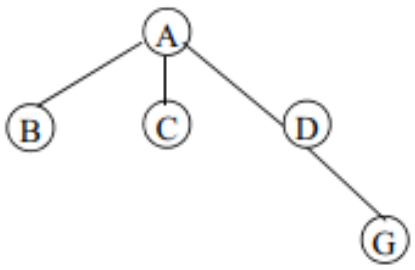


Áp dụng thuật toán tìm kiếm theo độ sâu với đồ thị trên (trình bày từng bước; vẽ cây tìm kiếm).

Giải

Lần lặp	L=∅	u	u∈T	v	L
0					A
1	False	A	False	B, C, D	D,C,B
2	False	D	False	G	G,C,B
3	False	G	True		

Quá trình tìm kiếm thành công, cây tìm kiếm là:



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

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

- Độ phức tạp của thuật toán tìm kiếm theo độ sâu.

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

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

2.3.3 Các trạng thái lặp

Như ta thấy trong mục 2, cây tìm kiếm có thể chứa nhiều đỉnh ứng với cùng một trạng thái, các trạng thái này được gọi là trạng thái lặp. Chẳng hạn, trong cây tìm kiếm hình 4b, các trạng thái C, E, F là các trạng thái lặp. Trong đồ thị biểu diễn không gian

trạng thái, các trạng thái lặp ứng với các đỉnh có nhiều đường đi dẫn tới nó từ trạng thái ban đầu. Nếu đồ thị có chu trình thì cây tìm kiếm sẽ chứa các nhánh với một số đỉnh lập lại vô hạn lần. Trong các thuật toán tìm kiếm sẽ lãng phí rất nhiều thời gian để phát triển lại các trạng thái mà ta đã gặp và đã phát triển. Vì vậy trong quá trình tìm kiếm ta cần tránh phát sinh ra các trạng thái mà ta đã phát triển. Chúng ta có thể áp dụng một trong các giải pháp sau đây:

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

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

Để thực hiện giải pháp thứ 3 ta cần lưu các trạng thái đã phát triển vào tập Q, lưu các trạng thái chờ phát triển vào danh sách L. Đương nhiên, trạng thái v lần đầu được sinh ra nếu nó không có trong Q và L. Việc lưu các trạng thái đã phát triển và kiểm tra xem một trạng thái có phải lần đầu được sinh ra không đòi hỏi rất nhiều không gian và thời gian.

2.3.4 Tìm kiếm sâu lặp

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

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

```
procedure Depth_Limited_Search(d);
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu u0;
   depth(u0) ← 0;
2. loop do
```

```
    2.1 if L rỗng then
                {thông báo thất bại; stop};
    2.2 Loại trạng thái u ở đầu danh sách L;
    2.3 if u là trạng thái kết thúc then
            {thông báo thành công; stop};
    2.4 if depth(u) <= d then
            for mỗi trạng thái v kề u do
                {Đặt v vào đầu danh sách L;
                 depth(v) ← depth(u) + 1};
end;
```

Thuật toán tìm kiếm sâu lặp

```
procedure Depth_Deepening_Search;
begin
for d ← 0 to max do
  {Depth_Limited_Search(d);
   if thành công then exit}
end;
```

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

- Cũng như tìm kiếm theo bề rộng, tìm kiếm sâu lặp luôn luôn tìm ra nghiệm (nếu bài toán có nghiệm), miễn là ta chọn độ sâu mã đủ lớn.
- Tìm kiếm sâu lặp chỉ cần không gian nhớ như tìm kiếm theo độ sâu.
- Trong tìm kiếm sâu lặp, ta phải phát triển lặp lại nhiều lần cùng một trạng thái.

Điều đó làm cho ta có cảm giác rằng, tìm kiếm sâu lặp lãng phí nhiều thời gian. Thực ra thời gian tiêu tốn cho phát triển lặp lại các trạng thái là không đáng kể so với thời gian tìm kiếm theo bề rộng. Thật vậy, mỗi lần gọi thủ tục tìm kiếm sâu hạn chế tới mức d, nếu cây tìm kiếm có nhân tố nhánh là b, thì số đỉnh cần phát triển là:

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

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

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

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

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

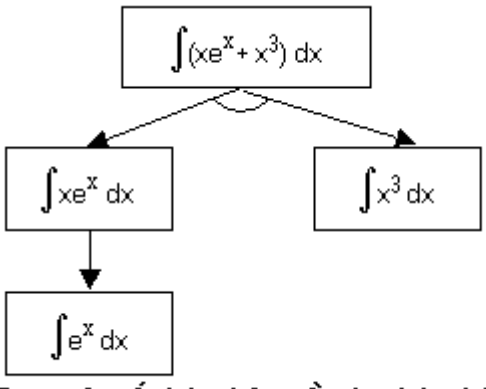
2.4 Quy vấn đề về các vấn đề con. Tìm kiếm trên đồ thị and/or.

2.4.1 Quy vấn đề về các vấn đề con:

Trong mục 2.1, chúng ta đã nghiên cứu việc biểu diễn vấn đề thông qua các trạng thái và các toán tử. Khi đó việc tìm nghiệm của vấn đề được quy về việc tìm đường trong không gian trạng thái. Trong mục này chúng ta sẽ nghiên cứu một phương pháp luận khác để giải quyết vấn đề, dựa trên việc quy vấn đề về các vấn đề con. Quy vấn đề về các vấn đề con (còn gọi là rút gọn vấn đề) là một phương pháp được sử dụng rộng rãi nhất để giải quyết các vấn đề. Trong đời sống hàng ngày, cũng như trong khoa học kỹ thuật, mỗi khi gặp một vấn đề cần giải quyết, ta vẫn thường cố gắng tìm cách đưa nó về các vấn đề đơn giản hơn. Quá trình rút gọn vấn đề sẽ được tiếp tục cho tới khi ta dẫn tới các vấn đề con có thể giải quyết được dễ dàng. Sau đây chúng ta xét một số vấn đề.

Ví dụ 5: Vấn đề tính tích phân bất định

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



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

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

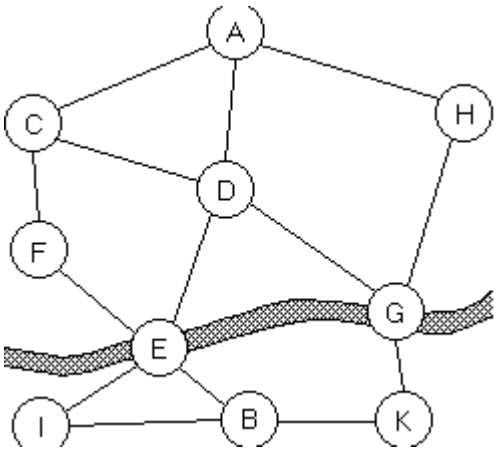
Chúng ta có thể biểu diễn việc quy một vấn đề về các vấn đề con cơ bởi các trạng thái và các toán tử. ở đây, bài toán cần giải là trạng thái ban đầu. Mỗi cách quy bài toán về các bài toán con được biểu diễn bởi một toán tử, toán tử $A \rightarrow B, C$ biểu diễn việc quy bài toán A về hai bài toán B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

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

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

***Ví dụ 6*: Vấn đề tìm đường đi trên bản đồ giao thông**

Bài toán này đã được phát triển như bài toán tìm đường đi trong không gian trạng thái (xem 2.1), trong đó mỗi trạng thái ứng với một thành phố, mỗi toán tử ứng với một con đường nối, nối thành phố này với thành phố khác. Bây giờ ta đưa ra một cách biểu diễn khác dựa trên việc quy vấn đề về các vấn đề con. Giả sử ta có bản đồ giao thông trong một vùng lãnh thổ (xem hình 2.5). Giả sử ta cần tìm đường đi từ thành phố A tới thành phố B. Có con sông chảy qua hai thành phố E và G và có cầu qua sông ở mỗi thành phố đó. Mọi đường đi từ A đến B chỉ có thể qua E hoặc G. Như vậy bài toán tìm đường đi từ A đến B được quy về:



Hình 2.5. Bản đồ giao thông giữa các thành phố

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

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

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

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

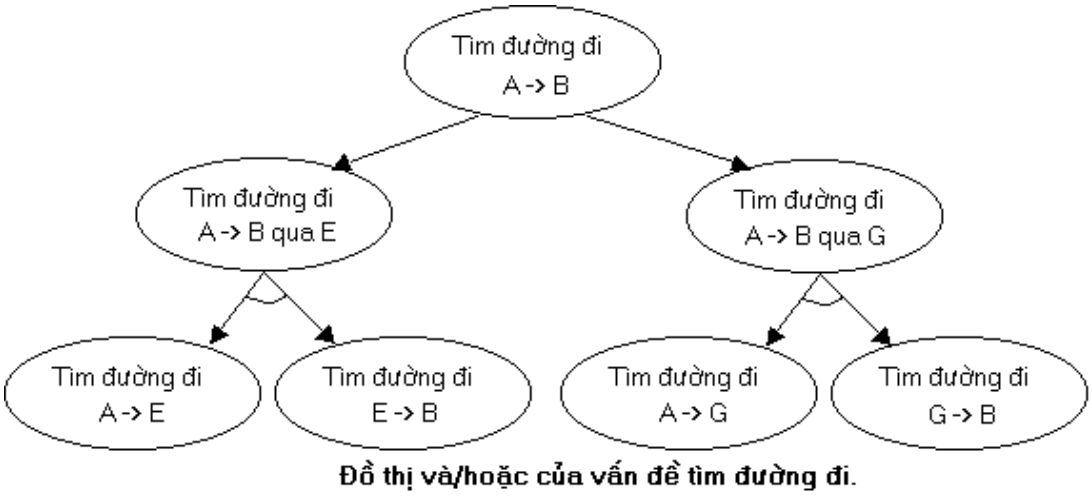
1.1 Tìm đường đi từ A đến E (và)

1.2 Tìm đường đi từ E đến B.

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

2.1 Tìm đường đi từ A đến G (và)

2.2 Tìm đường đi từ G đến B.



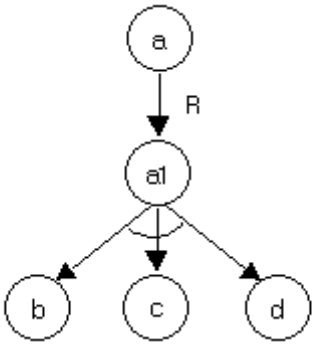
Hình 2.6: Đồ thị và/hoặc của vấn đề tìm đường đi

Quá trình rút gọn vấn đề như trên có thể biểu diễn dưới dạng đồ thị (đồ thị và/hoặc) trong hình trên. ở đây mỗi bài toán tìm đường đi từ một thành phố tới một thành phố khác ứng với một trạng thái. Các trạng thái kết thúc là các trạng thái ứng với

các bài toán tìm đường đi, chẳng hạn từ A đến C, hoặc từ D đến E, bởi vì đã có đường nối A với C, nối D với E.

2.4.2 Đồ thị and/or

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



Hình 2.7: Đồ thị biểu diễn toán tử R: $a \rightarrow b, c, d$

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

Ví dụ 7: Giả sử chúng ta có không gian trạng thái sau:

- Trạng thái ban đầu (bài toán cần giải) là a.
- Tập các toán tử quy gồm:

  R_1 : a →d, e, f

  R_2 : a →d, k

  R_3 : a →g, h

  R_4 : d →b, c

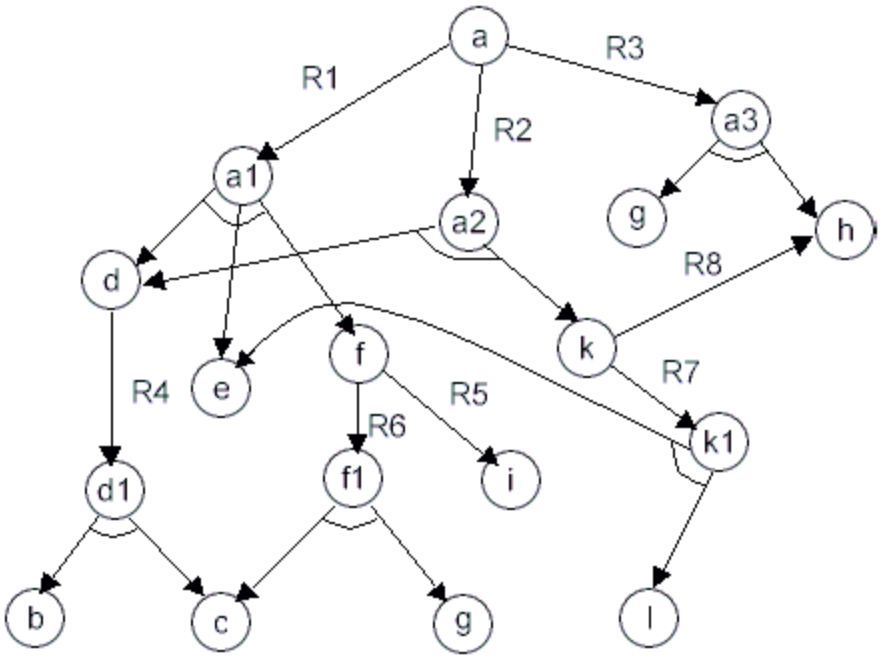
  R_5 : f →i

  R_6 : f →c, j

  R_7 : k →e, l

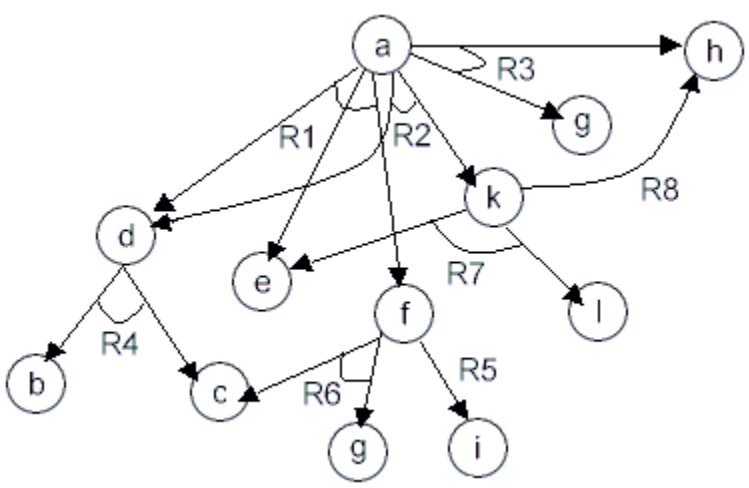
  R_8 : k →h

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



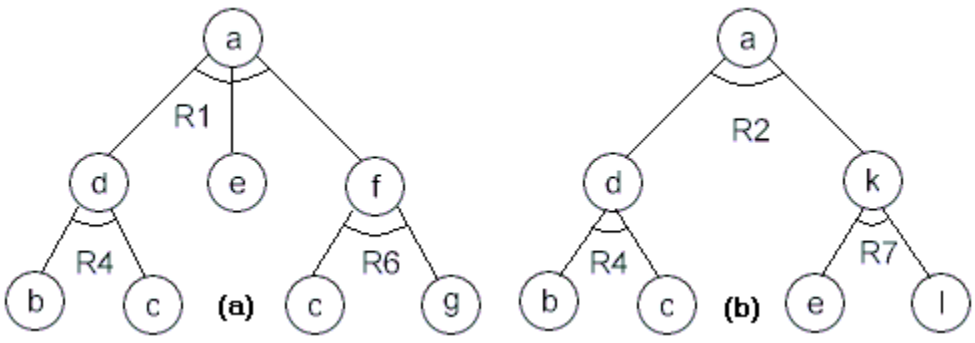
Hình 2.8: Đồ thị and/or – và/hoặc

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



Hình 2.9: Đồ thị rút gọn

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



Hình 2.10: Các cây nghiệm

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

Cây nghiệm là một cây, trong đó:

- Gốc của cây ứng với bài toán cần giải.
- Tất cả các lá của cây là các đỉnh kết thúc (đỉnh ứng với các bài toán sơ cấp).
- Nếu u là đỉnh trong của cây, thì các đỉnh con của u là các đỉnh kề u theo một toán tử nào đó.

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

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

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

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

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

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

Thứ tự giải các bài toán con trong một cây nghiệm là như sau. Bài toán ứng với đỉnh u chỉ được giải sau khi tất cả các bài toán ứng với các đỉnh con của u đã được giải. Chẳng hạn, với cây nghiệm trong hình trên, thứ tự giải các bài toán có thể là b, c, d, j, f, e, a.

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

2.4.3 Tìm kiếm trên đồ thị and/or

Ta sẽ sử dụng kỹ thuật tìm kiếm theo độ sâu trên đồ thị và/hoặc để đánh dấu các đỉnh. Các đỉnh sẽ được đánh dấu giải được hoặc không giải được theo định nghĩa đệ quy về đỉnh giải được và không giải được. Xuất phát từ đỉnh ứng với bài toán ban đầu, đi xuống theo độ sâu, nếu gặp đỉnh u là đỉnh kết thúc thì nó được đánh dấu giải được. Nếu gặp đỉnh u không phải là đỉnh kết thúc và từ u không đi tiếp được, thì u được đánh dấu không giải được. Khi đi tới đỉnh u, thì từ u ta lần lượt đi xuống các đỉnh v kề u theo một toán tử R nào đó. Nếu đánh dấu được một đỉnh v không giải được thì không cần đi tiếp xuống các đỉnh v còn lại. Tiếp tục đi xuống các đỉnh kề u theo một toán tử khác. Nếu tất cả các đỉnh kề u theo một toán tử nào đó được đánh dấu giải được thì u sẽ được đánh dấu giải được và quay lên cha của u. Còn nếu từ u đi xuống các đỉnh kề nó theo mọi toán tử đều gặp các đỉnh kề được đánh dấu không giải được, thì u được đánh dấu không giải được và quay lên cha của u.

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

- Biến Ok. Với mỗi toán tử R áp dụng được tại u, biến Ok nhận giá trị ***true*** nếu tất cả các đỉnh v kề u theo R đều giải được, và Ok nhận giá trị ***false*** nếu có một đỉnh v kề u theo R không giải được.

- Hàm Operator(u) ghi lại toán tử áp dụng thành công tại u, tức là Operator(u) = R nếu mọi đỉnh v kề u theo R đều giải được.

```
function Solvable(u);
begin
1. if u là đỉnh kết thúc then
  {Solvable ← true; stop};
2. if u không là đỉnh kết thúc và không có đỉnh kề then
  {Solvable(u) ← false; stop};
```

3. **for** *mỗi toán tử R áp dụng được tại u* **do**

{*Ok* ← *true***;**

for *mỗi v kề u theo R* **do**

if *Solvable(v)* = **false then {***Ok* ←**false; exit};**

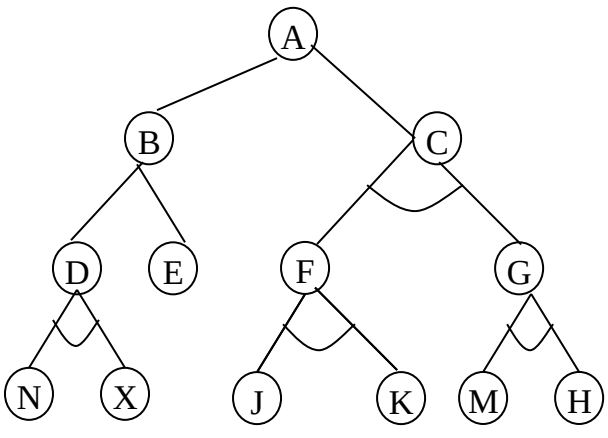
if *Ok* **then**

{*Solvable(u)***←true;** *Operator(u)***←** *R***; stop}}**

*4. Solvable(u)***←false;**

end;

Ví dụ 8: Cho đồ thị sau:



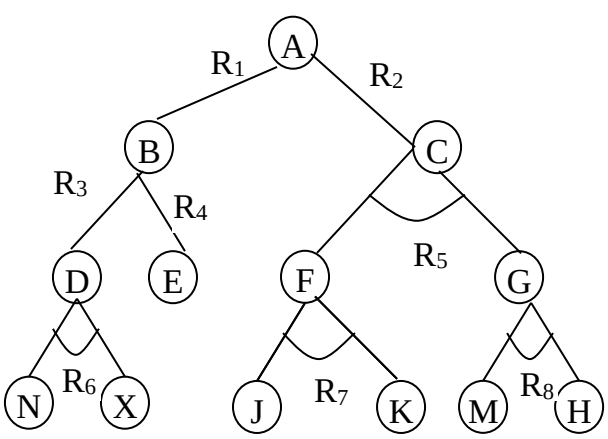
U_0=A

T = {X, H, J, K, M}.

Áp dụng thuật toán tìm kiếm trên đồ thị AND/OR gán nhãn giải được hoặc không giải được cho đỉnh U_0 của đồ thị trên. Từ đó kết luận bài toán ứng với đỉnh A có giải được không?

Giải:

Gán toán tử



Solvable(A)→R_1, R_2

R_1: Ok(R_1) = true, B

Solvable(B)→R_3, R_4

R_3: Ok(R_3) = true, D

Solvable(D)→R_6

R_6: Ok(R_6) = true, N, X

Solvable(N) = false

→Ok(R_6) = false

→Solvable(D) = false

→Ok(R_3) = false

R_4: Ok(R_4) = true, E

Solvable(E)= false

→Ok(R_3) = false

→Solvable(B) = false

→Ok(R_1) = false

R_2: Ok(R_2) = true, C

Solvable(C)→R_5

R_5: Ok(R_5) = true, F, G

Solvable(F)→R_7

R_7: Ok(R_7) = true, J, K

Solvable(J) = true

Solvable(K) = true

→Solvable(F) = true

Solvable(G) →R_8

R_8: Ok(R_8) = true, M, H

Solvable(M) = true

Solvable(H) = true

→Solvable(G) = true

→Solvable(C) = true

→Solvable(A) = true

Vậy đỉnh bài toán tương ứng với đỉnh A là giải được

Nhận xét

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

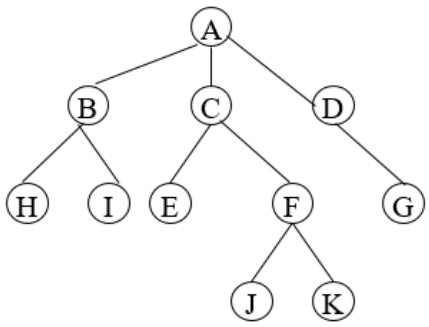
Nếu bài toán ban đầu giải được, thì bằng cách sử dụng hàm Operator ta sẽ xây dựng được cây nghiệm.

Câu hỏi thường gặp

1. Khi nào nên sử dụng DFS và khi nào nên sử dụng BFS?
2. Làm thế nào để thực hiện DFS và BFS trên đồ thị vô hướng?
3. Có vấn đề gì xảy ra khi thực hiện BFS hoặc DFS trên đồ thị có chu trình?
4. Tại sao DFS thường được thực hiện dưới dạng đệ quy?
5. Trong trường hợp nào nên sử dụng sâu lặp thay vì DFS hay BFS?
6. Tìm kiếm sâu lặp thích hợp cho bài toán nào?
7. Làm thế nào để biểu diễn một bài toán trên đồ thị AND/OR?
8. Trường hợp nào cần sử dụng đồ thị AND/OR thay vì đồ thị thông thường?
9. Làm thế nào để tìm kiếm tối ưu trên đồ thị AND/OR?
10. Đồ thị AND/OR có ứng dụng gì trong thực tế?
11. Đánh giá ưu nhược điểm của các phương pháp tìm kiếm theo chiều rộng, sâu và sâu lặp.

Bài tập cuối chương

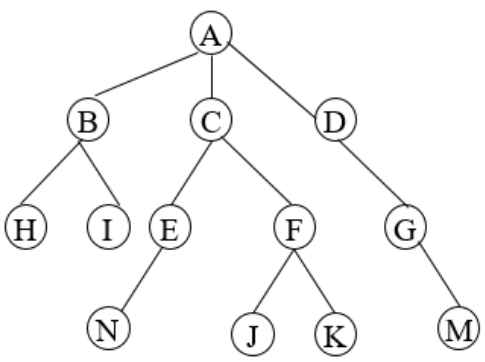
Bài 1: Cho đồ thị không gian trạng thái sau



u_0 = A ; T = { E,J,K}

- Áp dụng thuật toán tìm kiếm theo chiều rộng với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo chiều sâu với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo sâu hạn chế với d=2 với đồ thị trên

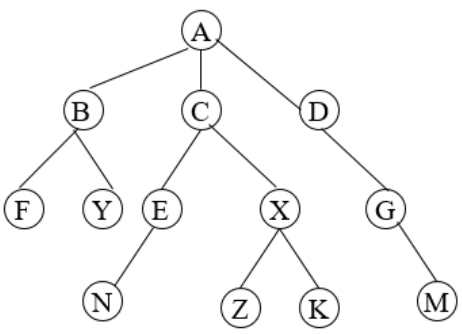
Bài 2: Cho đồ thị sau:



u_0 = A ; T = {K,N,I}

- Áp dụng thuật toán tìm kiếm theo chiều rộng với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo độ sâu với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo sâu hạn chế với d=3 với đồ thị trên

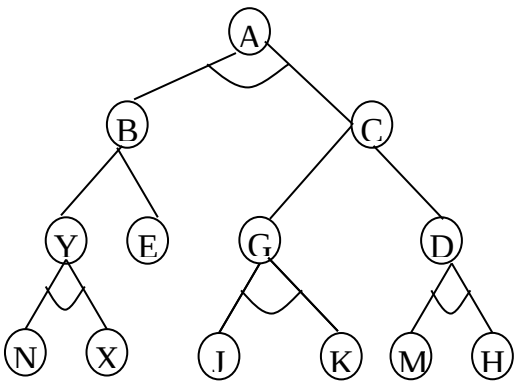
Bài 3: Cho đồ thị sau:



u_0 = A ; T = {N,Z,M}

- Áp dụng thuật toán tìm kiếm theo chiều rộng với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo độ sâu với đồ thị trên
- Áp dụng thuật toán tìm kiếm theo sâu hạn chế với d=3 với đồ thị trên

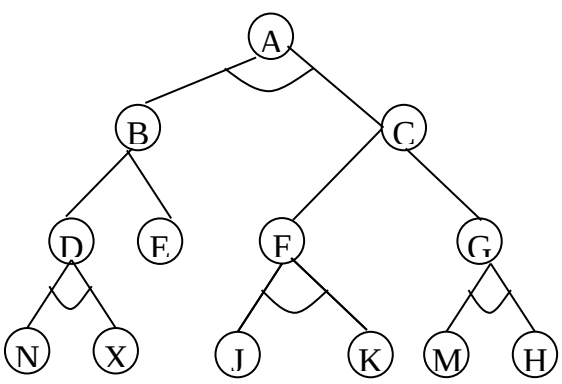
Bài 4: Cho đồ thị sau:



u_0=A ; T = {N, X, J, H, I}.

Áp dụng thuật toán tìm kiếm trên đồ thị AND/OR gán nhãn giải được hoặc không giải được cho các đỉnh của đồ thị trên. Từ đó kết luận bài toán ứng với đỉnh A có giải được không?

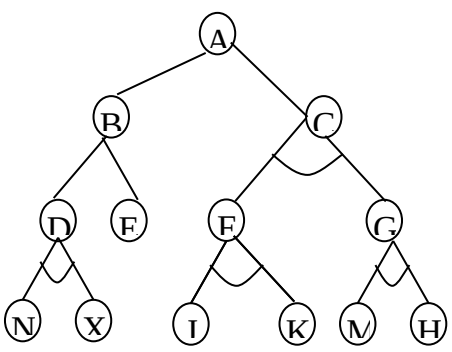
Bài 5: Cho đồ thị sau:



U_0=A ; T = {N, X, H, K, M}.

Áp dụng thuật toán tìm kiếm trên đồ thị AND/OR gán nhãn giải được hoặc không giải được cho đỉnh u_0 của đồ thị trên. Từ đó kết luận bài toán ứng với đỉnh A có giải được không?

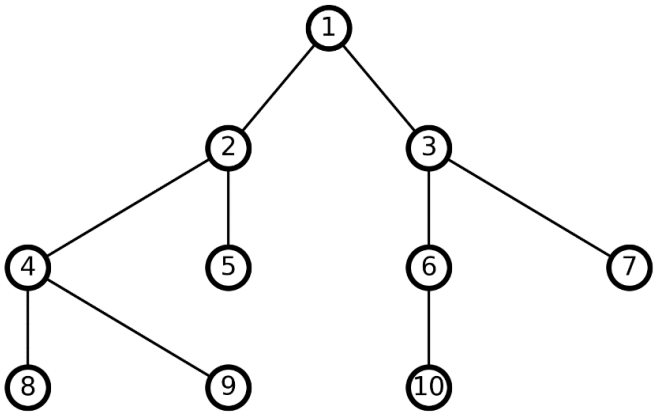
Bài 6: Cho đồ thị sau:



U_0=A ; T = {N, X, J, E, M}.

Áp dụng thuật toán tìm kiếm trên đồ thị AND/OR gán nhãn giải được hoặc không giải được cho đỉnh u_0 của đồ thị trên. Từ đó kết luận bài toán ứng với đỉnh A có giải được không?

Bài 7 : Cho đồ thị sau



U_0 =1 ; T = {9,10}

- Áp dụng thuật toán tìm kiếm theo chiều rộng với đồ thị trên

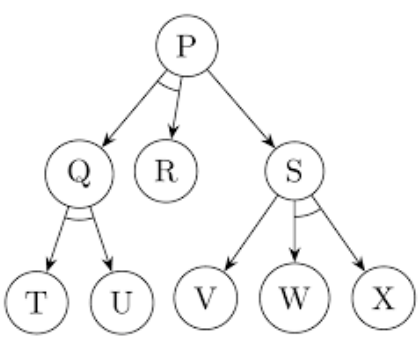
Bài 8 : Sử dụng đồ thị đã cho ở bài 7

- Áp dụng thuật toán tìm kiếm theo độ sâu với đồ thị trên

Bài 9 : Sử dụng đồ thị đã cho ở bài 7

- Áp dụng thuật toán tìm kiếm theo sâu hạn chế với d=3 với đồ thị trên

Bài 10 : Cho đồ thị sau



U_0=P ; T = {T,U}.

Áp dụng thuật toán tìm kiếm trên đồ thị AND/OR gán nhãn giải được hoặc không giải được cho đỉnh u_0 của đồ thị trên. Từ đó kết luận bài toán ứng với đỉnh P có giải được không?

CHƯƠNG 3. CÁC CHIẾN LƯỢC TÌM KIẾM KINH NGHIỆM

Nội dung chính của chương

Trong chương 2, chúng ta đã nghiên cứu việc biểu diễn vấn đề trong không gian trạng thái và các kỹ thuật tìm kiếm mù. Các kỹ thuật tìm kiếm mù rất kém hiệu quả và trong nhiều trường hợp không thể áp dụng được. Trong chương này, chúng ta sẽ nghiên cứu các phương pháp tìm kiếm kinh nghiệm (tìm kiếm heuristic), đó là các phương pháp sử dụng hàm đánh giá để hướng dẫn sự tìm kiếm.

Trong bài học này chúng ta sẽ tìm hiểu một số nội dung sau:

1. Hàm đánh giá và tìm kiếm kinh nghiệm

2. Các chiến lược tìm kiếm kinh nghiệm: tìm kiếm tốt nhất – đầu tiên; tìm kiếm leo đồi; tìm kiếm beam

Mục tiêu cần đạt được của chương

- Học viên cần hiểu được vấn đề tìm kiếm kinh nghiệm, sự khác biệt giữa tìm kiếm kinh nghiệm và tìm kiếm mù đã tìm hiểu ở bài 2.
- Học viên cần nắm được tư tưởng, các bước thực hiện của các thuật toán tìm kiếm tốt nhất – đầu tiên; tìm kiếm leo đồi; tìm kiếm beam.
- Áp dụng được trong một số bài tập cụ thể.

BÀI 3: CÁC CHIẾN LƯỢC TÌM KIẾM KINH NGHIỆM (Số tiết: 3 tiết)

3.1 Hàm đánh giá

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

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

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

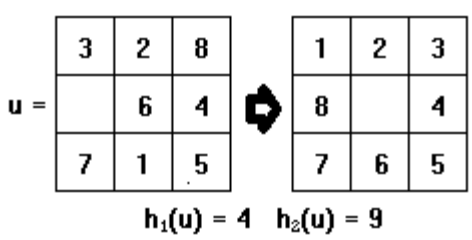
Hàm đánh giá

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

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

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

Hàm h_1: Với mỗi trạng thái u thì $h_1(u)$ là số quân không nằm đúng vị trí của nó trong trạng thái đích. Chẳng hạn trạng thái đích ở bên phải hình sau, và u là trạng thái ở bên trái hình 2.1, thì $h_1(u) = 4$, vì các quân không đúng vị trí là 3, 8, 6 và 1.



Hình 3.1: Đánh giá trạng thái u

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

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

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

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

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

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

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

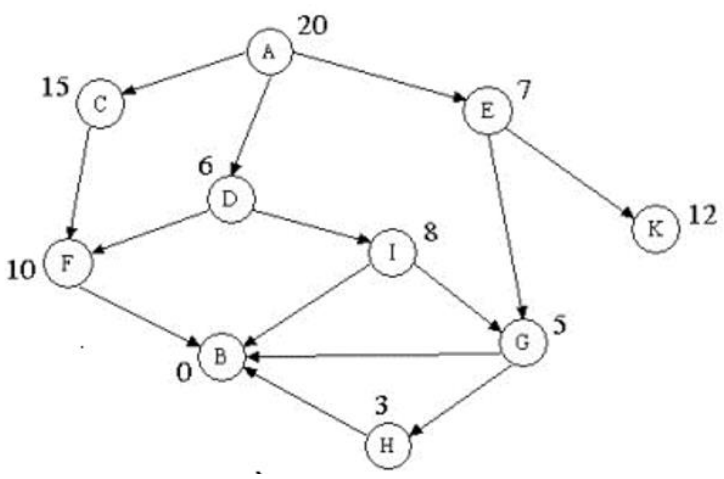
3.2 Các chiến lược tìm kiếm kinh nghiệm

3.2.1. Tìm kiếm tốt nhất - đầu tiên:

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

Thông thường có thể gặp và chia thành 3 biến thể:

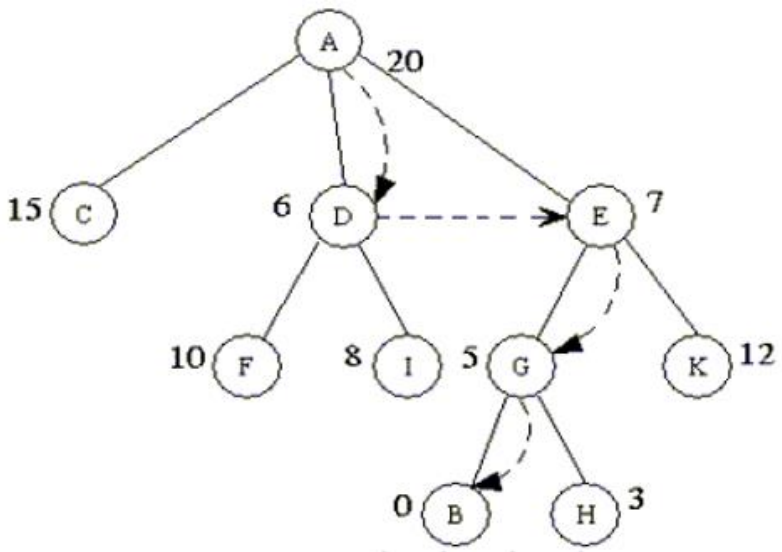
- Nếu hàm đánh giá chi phí từ trạng thái đầu đến trạng thái đang xét (u) – g(u): Uniform search (UCS)
- Nếu hàm đánh giá là từ trạng thái đang xét (u) đến trạng thái đích – h(u): Greedy search (tham lam) hay còn gọi là Best First Search, trong phạm vi bài học này chúng ta áp dụng hàm đánh giá h(u).
- Nếu sử dụng g(u) + h(u): A* search



Hình 3.2: Đồ thị không gian trạng thái

Ví dụ 9: Xét không gian trạng thái được biểu diễn bởi đồ thị trong *hình 3.2*, trong đó trạng thái ban đầu là A, trạng thái kết thúc là B. Giá trị của hàm đánh giá là các số ghi

cạnh mỗi đỉnh. Quá trình tìm kiếm tốt nhất - đầu tiên diễn ra như sau: Đầu tiên phát triển đỉnh A sinh ra các đỉnh kề là C, D và E. Trong ba đỉnh này, đỉnh D có giá trị hàm đánh giá nhỏ nhất, nó được chọn để phát triển và sinh ra F, I. Trong số các đỉnh chưa được phát triển C, E, F, I thì đỉnh E có giá trị đánh giá nhỏ nhất, nó được chọn để phát triển và sinh ra các đỉnh G, K. Trong số các đỉnh chưa được phát triển thì G tốt nhất, phát triển G sinh ra B, H. Đến đây ta đã đạt tới trạng thái kết thúc. Cây tìm kiếm tốt nhất - đầu tiên được biểu diễn trong *hình 3.3.*



Hình 3.3. Cây tìm kiếm tốt nhất - đầu tiên

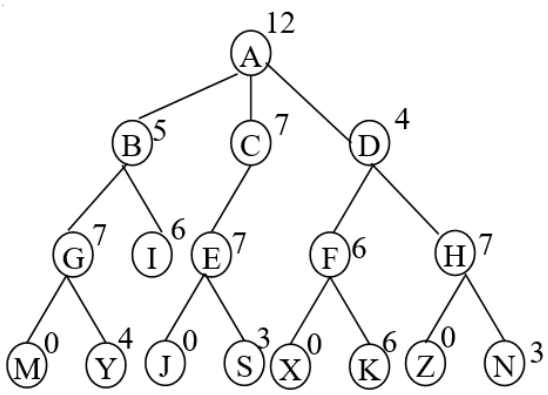
Sau đây là thủ tục tìm kiếm tốt nhất - đầu tiên. Trong thủ tục này, chúng ta sử dụng danh sách L để lưu các trạng thái chờ phát triển, danh sách được sắp theo thứ tự tăng dần của hàm đánh giá sao cho trạng thái có giá trị hàm đánh giá nhỏ nhất ở đầu danh sách.

procedure *Best_First_Search*;
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
2. **loop do**
2.1 **if** *L rỗng* **then**
{*thông báo thất bại*; **stop};**
2.2 Loại trạng thái u ở đầu danh sách L;
2.3 **if** *u là trạng thái kết thúc* **then**
{*thông báo thành công*; **stop}**
2.4 **for** *mỗi trạng thái v kề u* **do**
Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần của hàm đánh giá;

```
    end;
```

Ví dụ 10:

Cho đồ thị sau:



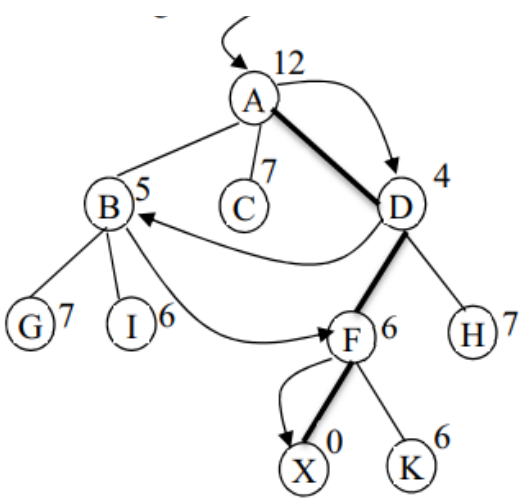
U_0=A; T={M, Z, J, X}

Áp dụng thuật toán tốt nhất-đầu tiên với đồ thị trên.

Giải:

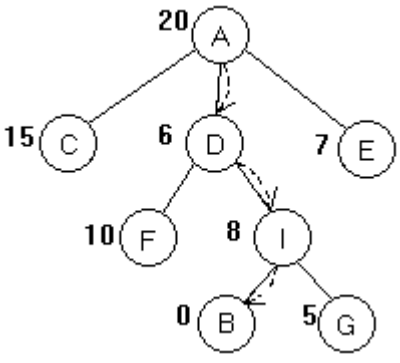
Lần lặp	L=ϕ	u	u∈T	v	L
0					A_{12}
1	False	A	False	B_5, C_7, D_4	D_4, B_5, C_7
2	False	D	False	F_6, H_7	B_5, F_6, C_7, H_7
3	False	B	False	G_7, I_6	F_6, I_6, G_7, C_7, H_7
4	False	F	False	X_0, K_6	X_0, K_6, I_6, G_7, C_7, H_7
5	False	X	True		

Quá trình tìm kiếm thành công, cây tìm nghiệm là:



3.2.2 Tìm kiếm leo đồi

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



Hình 3.4. Cây tìm kiếm leo đồi

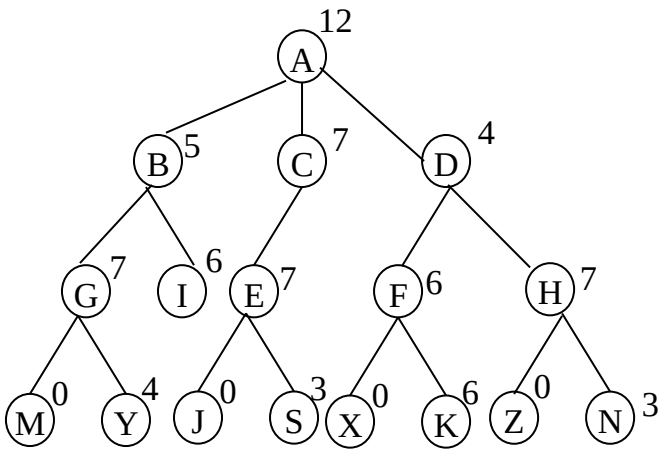
Ví dụ 11: Ta lại xét đồ thị không gian trạng thái trong hình 3.2. Quá trình tìm kiếm leo đồi được tiến hành như sau. Đầu tiên phát triển đỉnh A sinh ra các đỉnh con C, D, E. Trong các đỉnh này chọn D để phát triển vì h(D) = 6 (min), sinh ra các đỉnh F,I. Trong số các đỉnh con của D chọn I vì h(I)=8 nhỏ nhất. Với I được chọn sinh ra các đỉnh con B, G (B là trạng thái kết thúc). Quá trình tìm kiếm kết thúc. Cây tìm kiếm leo đồi được cho trong *hình 3.4*.

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

procedure *Hill_Climbing_Search*;
begin
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
2. **loop do**
2.1 **if** *L rỗng* **then**
{*thông báo thất bại*; **stop};**
2.2 Loại trạng thái u ở đầu danh sách L;
2.3 **if** *u là trạng thái kết thúc* **then**
{*thông báo thành công*; **stop};**
2.4 **for** *mỗi trạng thái v kề u do đặt v vào* L_1;
2.5 Sắp xếp L_1 *theo thứ tự tăng dần của hàm đánh giá*;
2.6 Chuyển danh sách L_1 *vào đầu danh sách L*;

```
    end;
```

***Ví dụ 12*:** Cho đồ thị sau:



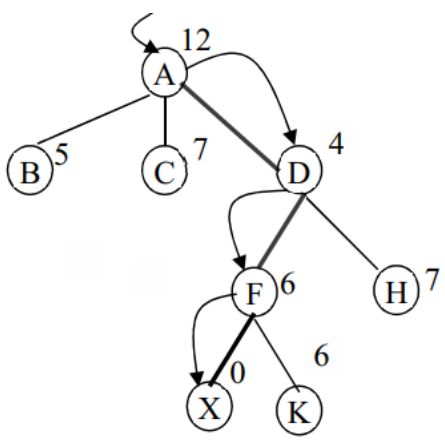
U_0=A; T={M, Z, J, X}

Áp dụng thuật toán leo đồi với đồ thị trên.

Giải

Lần lặp	L=ϕ	u	u∈T	v	L_1	L
0						A_{12}
1	False	A	False	B_5, C_7, D_4	D_4, B_5, C_7	D_4, B_5, C_7
2	False	D	False	F_6, H_7	F_6, H_7	F_6, H_7, B_5, C_7
3	False	F	False	X_0, K_6	X_0, K_6	X_0, K_6, H_7, B_5, C_7
4	False	X	True			

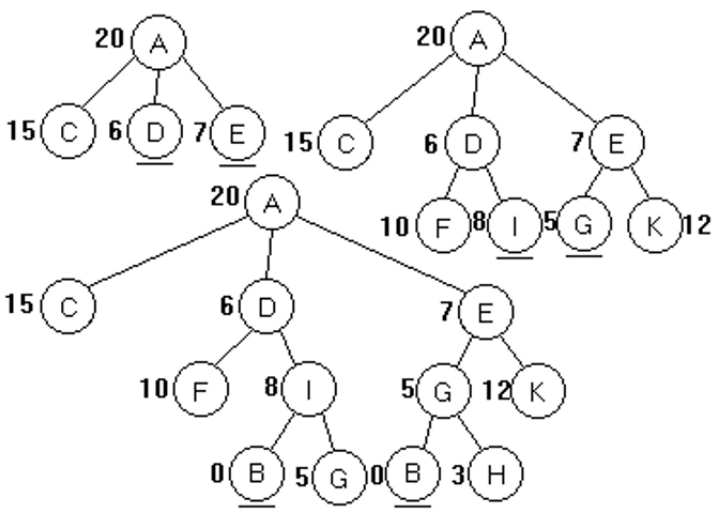
Quá trình tìm kiếm thành công, cây tìm kiếm là:



3.2.3 Tìm kiếm beam

Tìm kiếm beam (beam search) là một thuật toán tìm kiếm heuristic, giống như tìm kiếm theo bề rộng, nó phát triển các đỉnh ở một mức rồi phát triển các đỉnh ở mức tiếp theo. Tuy nhiên, trong tìm kiếm theo bề rộng, ta phát triển tất cả các đỉnh ở một mức, còn trong tìm kiếm beam, ta hạn chế chỉ phát triển k đỉnh tốt nhất (các đỉnh này được xác định bởi hàm đánh giá). Do đó trong tìm kiếm beam, ở bất kỳ mức nào cũng chỉ có nhiều nhất k đỉnh được phát triển, trong khi tìm kiếm theo bề rộng, số đỉnh cần phát triển ở mức d là b^d (b là nhân tố nhánh). Beam search được sử dụng trong các bài toán như dịch máy, nhận dạng giọng nói, tóm tắt văn bản,… Đó là các bài toán NLP có đầu ra liên quan đến việc tạo một chuỗi các từ

Ví dụ 13: Chúng ta lại xét đồ thị không gian trạng thái trong hình 3.2. Chọn k = 2. Khi đó cây tìm kiếm beam được cho như hình 3.5. Các đỉnh được gạch dưới là các đỉnh được chọn để phát triển ở mỗi mức.



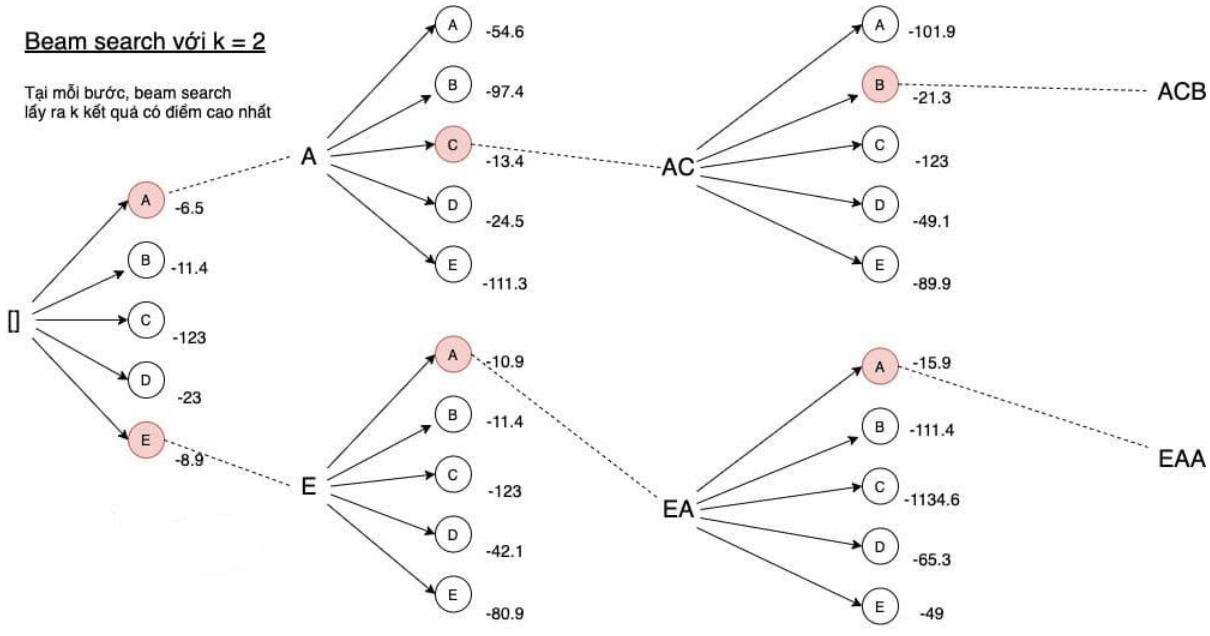
Hình 3.5. Cây tìm kiếm Beam

Ví dụ 14: *Xét bài toán NLP (N*atural language processing) như dịch máy (machine translation), tạo caption ảnh tự động (image caption generation), tóm tắt văn bản (text summarization), tổng hợp tiếng nói (auto speech recognition), … yêu cầu đầu ra của mô hình là chuỗi các từ có trong từ điển.

Thường thì mỗi từ trong chuỗi từ mà mô hình của các bài toán như trên dự đoán (predict) sẽ đi kèm theo một phân phối xác suất tương ứng. Tìm kiếm chuỗi từ phù hợp nhất yêu cầu chúng ta cần duyệt qua tất cả các chuỗi từ có thể có từ dự đoán của mô hình. Thường thì từ điển của chúng ta sẽ có kích thước rất lớn, không gian tìm kiếm của chúng ta là kích thước từ điển lũy thừa với độ dài của chuỗi. Do không gian tìm kiếm là quá lớn. Trên thực tế, chúng ta thường dùng một thuật toán tìm kiếm heuristic để có

được một kết quả tìm kiếm đủ tốt (good enough) cho mỗi dự đoán thay vì phải tìm kiếm toàn cục.

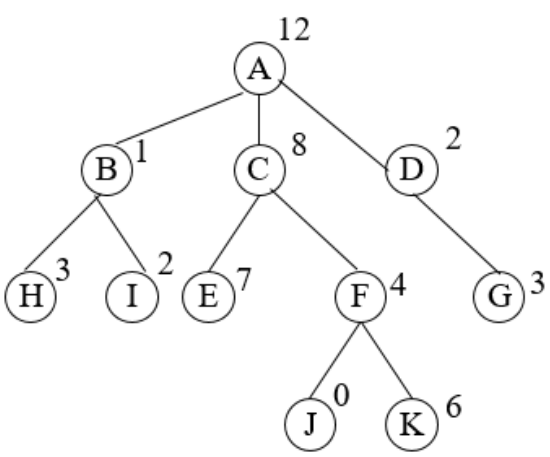
Mỗi chuỗi từ sẽ được gán một số điểm dựa trên xác suất phân bố của chúng, thuật toán tìm kiếm sẽ dựa trên điểm số này để đánh giá các chuỗi từ này. Có 2 thuật toán tìm kiếm phổ biến giúp tìm ra chuỗi từ "đủ tốt" đó là tìm kiếm tham lam (greedy search) và beam search. Dễ dàng nhận ra là nếu k =1 thì Beam search chính là Greedy search.



Mô phỏng hoạt động của Beam search với k =2.

Bài tập cuối chương

Bài 1: Cho đồ thị



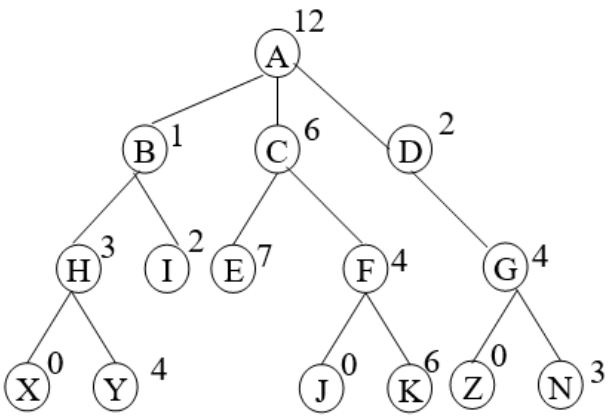
U_0=A; T={J}

+ Áp dụng thuật toán tốt nhất-đầu tiên với đồ thị trên

Bài 2: Sử dụng đồ thị và dữ liệu của Bài 1.

+ Áp dụng thuật toán leo đồi.

Bài 3: Cho đồ thị sau:



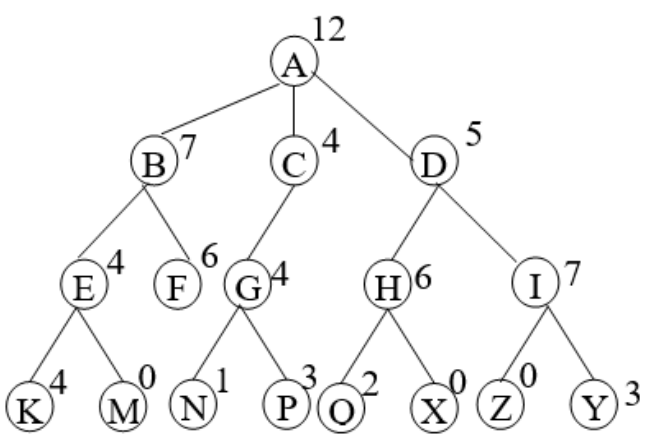
U_0=A; T={X, J, Z}

+ Áp dụng thuật toán tốt nhất-đầu tiên với đồ thị trên

Bài 4: Sử dụng đồ thị và dữ liệu của Bài 3.

+ Áp dụng thuật toán leo đồi.

Bài 5: Cho đồ thị



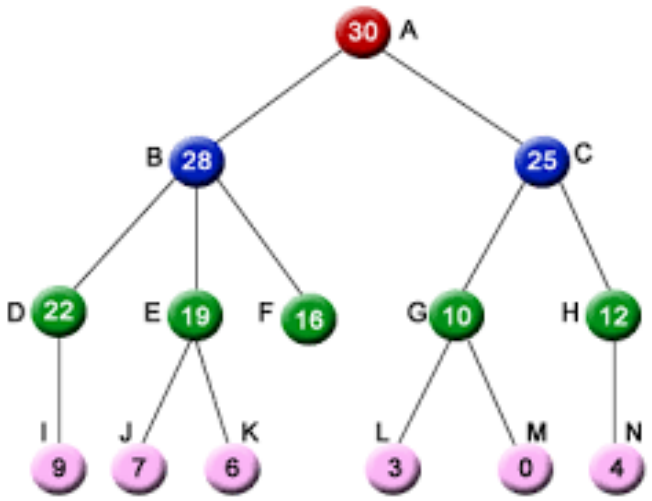
u_0=A; T={M,X,Z}

+ Áp dụng thuật toán tốt nhất-đầu tiên với đồ thị trên

Bài 6: Sử dụng đồ thị và dữ liệu của Bài 5.

+ Áp dụng thuật toán leo đồi.

Bài 7: Cho đồ thị sau:



U_0=A; T={M, N, J, K}

+ Áp dụng thuật toán tốt nhất - đầu tiên với đồ thị trên

Bài 8: Sử dụng đồ thị và dữ liệu của Bài 7.

+ Áp dụng thuật toán leo đồi.

Bài 9: Có n^2-1 số mang các giá trị từ 1 tới n^2-1 được sắp xếp vào một lưới các ô vuông kích thước n x n. Mỗi số đó được gọi là một quân cờ và lưới ô đó được gọi là bàn cờ. Có một vị trí của bàn cờ bỏ trống. Mỗi lần di chuyển quân, người chơi được phép chuyển một quân ở vị trí ô tiếp giáp cạnh với ô trống vào ô trống.

Từ một trạng thái ban đầu (sự sắp xếp ban đầu của các quân trên bàn cờ), thực hiện các nước đi hợp lệ để thu được trạng thái kết thúc (trạng thái đích cần đạt được).Bài toán 8-puzzle có n=8, ta minh họa trạng thái ban đầu và trạng thái kết húc qua các hình vẽ dưới đây:

2	8	3
1	6	4
	7	5

1	2	3
8		4
7	6	5

Hãy xây dựng hàm đánh giá và sử dụng thuật toán leo đồi giải bài toán trên:

Bài 10. Tương tự bài 9, hãy xây dựng hàm đánh giá và sử dụng thuật toán Best_first_search để giải quyết bài toán.

CHƯƠNG 4. CÁC CHIẾN LƯỢC TÌM KIẾM TỐI ƯU

Nội dung chính của chương

Đối với tìm kiếm mù, các nút biên lần lượt được mở rộng theo một thứ tự nhất định mà không tính tới việc ưu tiên các nút có khả năng dẫn tới lời giải nhanh hơn. Kết quả của việc tìm kiếm như vậy là việc di chuyển trong không gian tìm kiếm không có định hướng, dẫn tới phải xem xét nhiều trạng thái. Đối với những bài toán thực tế có không gian trạng thái lớn, tìm kiếm mù thường không thực tế do có độ phức tạp tính toán và yêu cầu bộ nhớ lớn.

Để giải quyết vấn đề trên, chiến lược tìm kiếm ***kinh nghiệm*** *hay còn được gọi là tìm kiếm heuristic sử dụng thêm thông tin từ bài toán để định hướng tìm kiếm, cụ* ***thể là lựa chọn thứ tự mở*** *rộng nút theo hướng* **nhanh** *dẫn tới đích hơn.*

Tuy nhiên các chiến lược tìm kiếm kinh nghiệm sẽ cho kết quả tìm kiếm nhanh hơn các chiến lược tìm kiếm mù nhưng nó cũng không phải là kết quả tốt nhất. Trong một số lĩnh vực người ta cần kết quả tìm kiếm là tối ưu theo một điều kiện nào đó, có thể là chi phí phải trả, có thể là tốc độ tìm kiếm nhanh nhất. Từ đó phát triển lên các chiến lược tìm kiếm tối ưu.

Nguyên tắc chung của tìm kiếm tối ưu là sử dụng một hàm mục tiêu $f(n)$ để đánh giá độ "tốt" tiềm năng của nút n, từ đó chọn nút n có hàm f tốt nhất để mở rộng trước. Thông thường, độ tốt được đo bằng giá thành đường đi tới đích, do vậy nút có hàm $f(n)$ nhỏ được ưu tiên mở rộng trước.

Trên thực tế, việc xây dựng hàm $f(n)$ phản ánh chính xác độ tốt của nút thường không thực hiện được, thay vào đó ta chỉ có thể ước lượng hàm $f(n)$ dựa vào thông tin có được từ bài toán. Như sẽ thấy trong các phần sau, hàm $f(n)$ thường chứa một thành phần là hàm heuristic $h(n)$, là hàm ước lượng khoảng cách từ nút n tới đích.

Trong bài học này chúng ta sẽ tìm hiểu các nội dung sau :

1. Hàm mục tiêu f

2. Các chiến lược tìm kiếm tối ưu : Trong chương này chúng ta sẽ nghiên cứu các thuật toán tìm kiếm đường đi ngắn nhất trong không gian trạng thái: Thuật toán A*; Thuật toán nhánh_và_cận.

Mục tiêu cần đạt được của chương

Sau khi học xong bài này học viên cần hiểu được hàm mục tiêu là gì, phương pháp xây dựng hàm mục tiêu, như thế nào là đường đi ngắn nhất. Hiểu được tư tưởng, thuật toán của Một số thuật toán tìm đường đi ngắn nhất như : A*, nhánh và cận. Áp dụng cho một số bài toán cụ thể.

BÀI 4: CÁC CHIẾN LƯỢC TÌM KIẾM TỐI ƯU (Số tiết: 3 tiết)

4.1 Hàm mục tiêu f

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

Tìm đường đi ngắn nhất.

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

Chúng ta có thể giải quyết vấn đề đặt ra bằng cách tìm tất cả các đường đi có thể có từ trạng thái ban đầu tới trạng thái đích (chẳng hạn, sử sụng các kỹ thuật tìm kiếm mù), sau đó so sánh độ dài của chúng, ta sẽ tìm ra đường đi ngắn nhất. Thủ tục tìm kiếm này thường được gọi là thủ tục bảo tàng Anh Quốc (British Museum Procedure). Trong thực tế, kỹ thuật này không thể áp dụng được, vì cây tìm kiếm thường rất lớn, việc tìm ra tất cả các đường đi có thể có đòi hỏi rất nhiều thời gian. Do đó chỉ có một cách tăng hiệu quả tìm kiếm là sử dụng các hàm đánh giá đề hướng dẫn sự tìm kiếm.

Các phương pháp tìm kiếm đường đi ngắn nhất mà chúng ta sẽ trình bày đều là các phương pháp tìm kiếm heuristic.

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

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

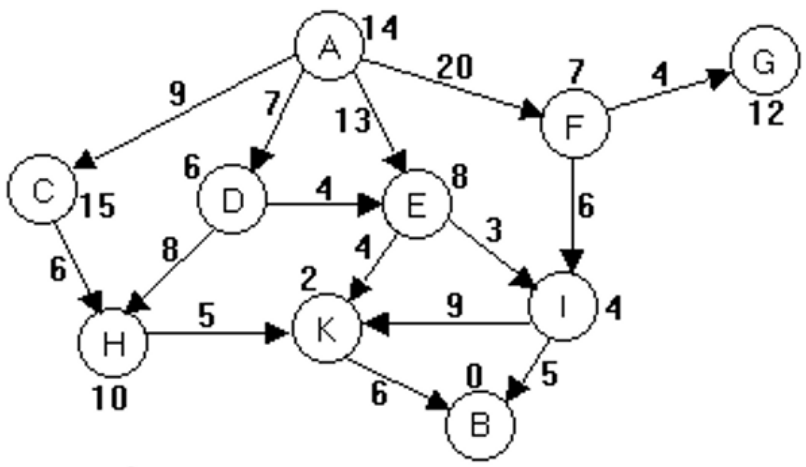
Hàm *h(u)* được gọi là ***chấp nhận được*** (hoặc đánh giá thấp***)*** nếu với mọi trạng thái *u*, *h(u)* ≤ độ dài đường đi ngắn nhất thực tế từ *u* tới trạng thái đích. Chẳng hạn trong bài toán tìm đường đi ngắn nhất trên bản đồ giao thông, ta có thể xác định *h(u)* là độ dài đường chim bay từ *u* tới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

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

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

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

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



Hình 4.1. Đồ thị không gian trạng thái với hàm đánh giá

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

4.2 Chiến lược tìm kiếm tối ưu A*

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

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

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

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

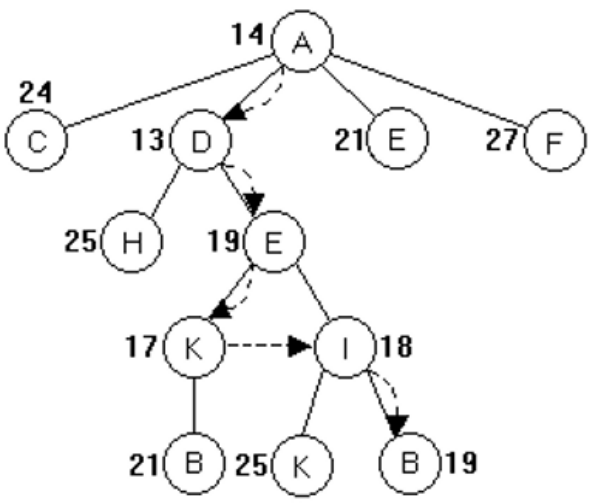
Như vậy đỉnh tốt nhất là D (vì f(D) = 13 là nhỏ nhất). Phát triển D, ta nhận được các đỉnh con H và E. Ta đánh giá H và E (mới):

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

Đường đi tới E qua D có độ dài:

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

Vậy đỉnh E mới có đánh giá là f(E) = g(E) + h(E) = 11 + 8 = 19. Trong số các đỉnh chờ phát triển, thì đỉnh E với đánh giá f(E) = 19 là đỉnh tốt nhất. Phát triển đỉnh này, ta nhận được các đỉnh con của nó là K và I. Chúng ta tiếp tục quá trình trên cho tới khi đỉnh được chọn để phát triển là đỉnh kết thúc B, độ dài đường đi ngắn nhất tới B là g(B) = 19. Quá trình tìm kiếm trên được mô tả bởi cây tìm kiếm trong hình 4.2, trong đó các số ghi cạnh các đỉnh là các giá trị của hàm đánh giá f(u).



*Hình 4.2. Cây tìm kiếm theo thuật toán A**

procedure *A**;

begin

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

2. **loop do**

2.1 **if** *L rỗng* **then**

{*thông báo thất bại***; stop};**

2.2 Loại trạng thái u ở đầu danh sách L;

2.3 **if** *u là trạng thái đích* **then**

{*thông báo thành công***; stop}**

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

{*g(v)* ← *g(u)* + *k(u,v)***;**

f(v) ← *g(v)* + *h(v)***;**

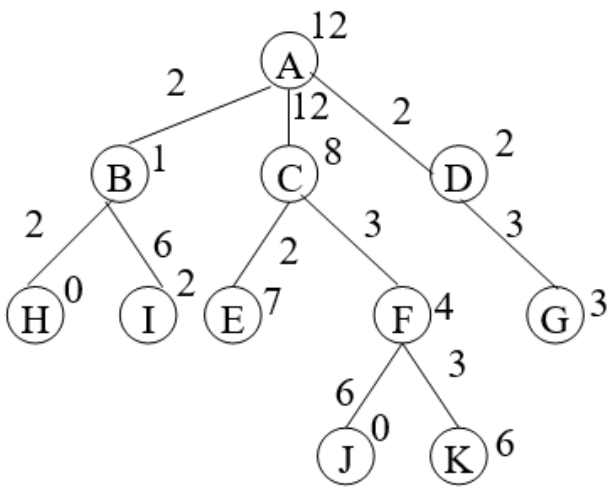
*Đặt v vào danh sách L***;}**

2.5 Sắp xếp L theo thứ tự tăng dần của hàm f sao cho trạng thái có giá trị của hàm f nhỏ nhất ở đầu danh sách;

end;

Ví dụ 15:

Cho đồ thị không gian trạng thái sau:



U_0=A; T={H, J}

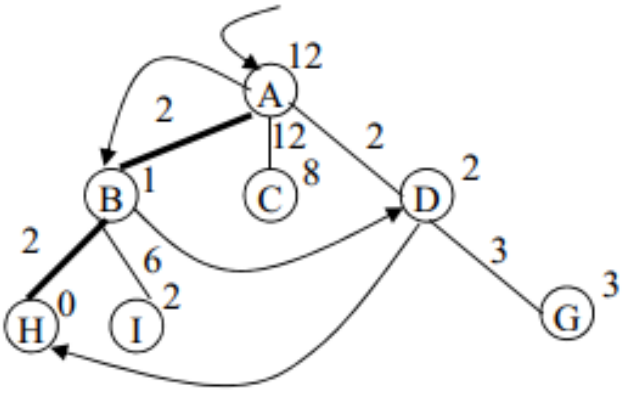
Áp dụng thuật toán A* đồ thị trên

Giải

Lần lặp	L=ϕ	u	u∈T	v	L
0					A_{12}^{0}
1	False	A	False	$B_3^2, C_{20}^{12}, D_4^2$	$B_3^2, D_4^2, C_{20}^{12}$
2	False	B	False	H_4^4, I_{10}^8	$H_4^4, D_4^2, I_{10}^8, C_{20}^{12}$
3	False	D	True	G_8^5	$H_4^4, G_8^5, I_{10}^8, C_{20}^{12}$
4	False	H	True		

Quá trình tìm kiếm thành công. Đường đi ngắn nhất từ A ->B->H, cost=4

Cây tìm kiếm là:



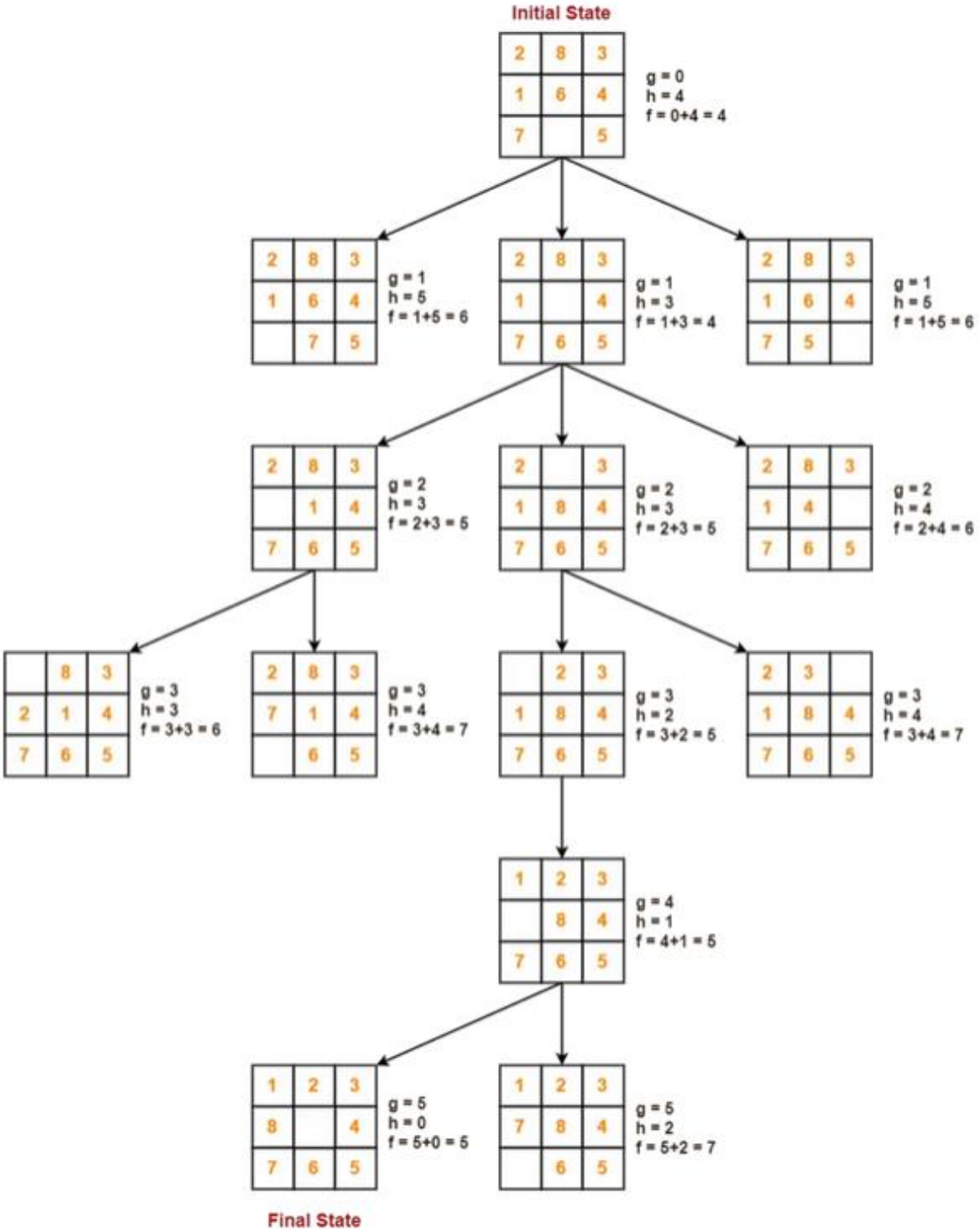
Ví dụ 16: Bài toán 8-puzzle sử dụng thuật toán A*

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Final State

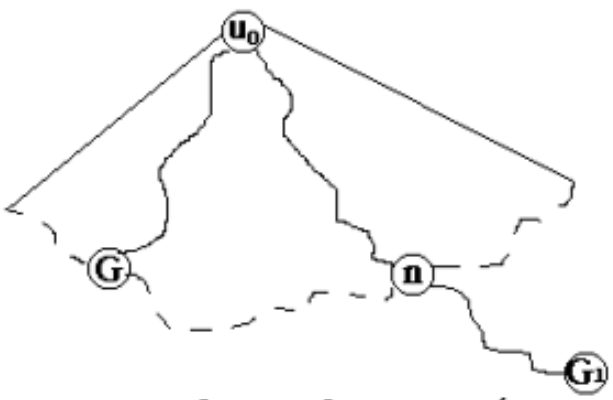


Chúng ta đưa ra một số nhận xét về thuật toán A*:

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

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

Giả sử thuật toán dừng lại ở đỉnh kết thúc G với độ dài đường đi từ trạng thái ban đầu u_0 tới G là g(G). Vì G là đỉnh kết thúc, ta có h(G) = 0 và f(G) = g(G) + h(G) = g(G). Giả sử nghiệm tối ưu là đường đi từ u_0 tới đỉnh kết thúc G_1 với độ dài l. Giả sử đường đi này "thoát ra" khỏi cây tìm kiếm tại đỉnh lá n (Xem hình 4.5). Có thể xảy ra hai khả năng: n trùng với G_1 hoặc không. Nừu n là G_1 thì vì G được chọn để phát triển trước G_1, nên $f(G) \le f(G_1)$, do đó $g(G) \le g(G_1) = l$. Nếu $n \ne G_1$ thì do h(u) là hàm đánh giá thấp, nên $f(n) = g(n) + h(n) \le l$. Mặt khác, cũng do G được chọn để phát triển trước n, nên $f(G) \le f(n)$, do đó, $g(G) \le l$. Như vậy, ta đã chứng minh được rằng độ dài của đường đi mà thuật toán tìm ra g(G) không dài hơn độ dài l của đường đi tối ưu. Vậy nó là độ dài đường đi tối ưu.



Hình 4.5. Đỉnh lá n của cây tìm kiếm nằm trên đường đi tối ưu

- Trong trường hợp hàm đánh giá h(u) = 0 với mọi u, thuật toán A* chính là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(u) mà ta đã nói đến.

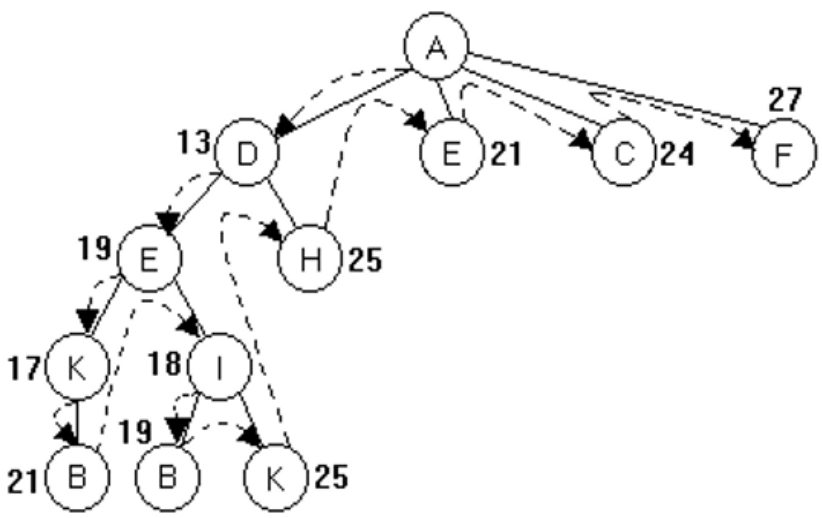
- Thuật toán A* đã được chứng tỏ là thuật toán hiệu quả nhất trong số các thuật toán đầy đủ và tối ưu cho vấn đề tìm kiếm đường đi ngắn nhất.

4.3 Chiến lược tìm kiếm tối ưu nhánh và cận.

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

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

Ví dụ 15: Chúng ta lại xét không gian trạng thái trong hình 4.1. Phát triển đỉnh A, ta nhận được các đỉnh con C, D, E và F, f(C) = 24, f(D) = 13, f(E) = 21, f(F) = 27. Trong số này D là tốt nhất, phát triển D, sinh ra các đỉnh con H và E, f(H) = 25, f(E) = 19. Đi xuống phát triển E, sinh ra các đỉnh con là K và I, f(K) = 17, f(I) = 18. Đi xuống phát triển K sinh ra đỉnh B với f(B) = g(B) = 21. Đi xuống B, vì B là đỉnh đích, vậy ta tìm được đường đi tối ưu tạm thời với độ dài 21. Từ B quay lên K, rồi từ K quay lên cha nó là E. Từ E đi xuống J, f(J) = 18 nhỏ hơn độ dài đường đi tạm thời (là 21). Phát triển I sinh ra các con K và B, f(K) = 25, f(B) = g(B) = 19. Đi xuống đỉnh B, vì đỉnh B là đích ta tìm được đường đi đầy đủ mới với độ dài là 19 nhỏ hơn độ dài đường đi tối ưu tạm thời cũ (21). Với độ dài đường đi tối ưu tạm thời bây giờ là 19. Bây giờ từ B ta lại quay lên các đỉnh còn lại chưa được phát triển. Song các đỉnh này đều có giá trị hàm đánh giá lớn hơn 19, do đó không có đỉnh nào được phát triển nữa. Như vậy, ta tìm được đường đi tối ưu với độ dài 19. Cây tìm kiếm được biểu diễn trong *hình 4.5*.



Hình 4.6. Cây tìm kiếm thuật toán nhánh – và – cận

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

Procedure *Branch_and_Bound***;**
begin
*1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu***;**
*Gán giá trị ban đầu cho cost***;**
2. **loop do**
2.1 **if** *L rỗng* **then stop;**
*2.2 Loại trạng thái u ở đầu danh sách L***;**
2.3 **if** *u là trạng thái kết thúc* **then**

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

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

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

{*g(v) ← g(u) + k(u,v)***;**

*f(v) ← g(v) + h(v)***;**

Đặt v vào danh sách L_1**};**

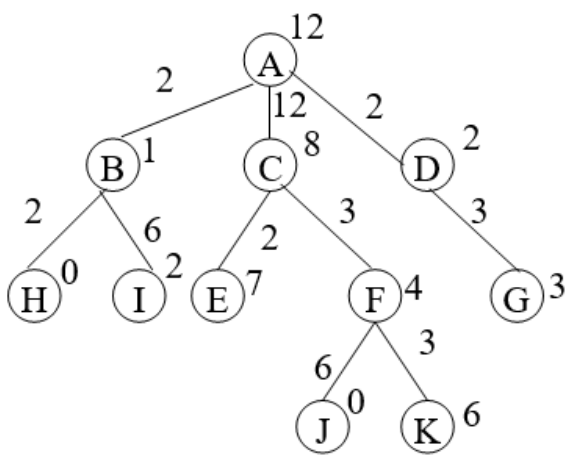
2.6 Sắp xếp L_1 *theo thứ tự tăng của hàm f***;**

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

end;

Ví dụ 17:

Cho đồ thị không gian trạng thái:



U_0=A; T={H, J}

Áp dụng thuật toán nhánh cận đồ thị trên

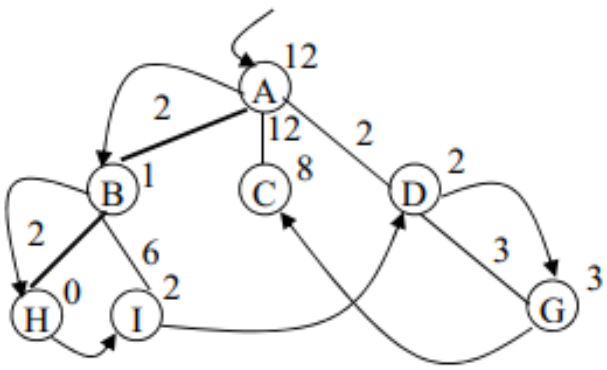
Giải:

Lần lặp	L=ϕ	u	u∈T	v	L_1	L	Cost
0						A_{12}^0	+∞
1	False	A	False	$B_3^2, C_{20}^{12}, D_4^2$	$B_3^2, D_4^2, C_{20}^{12}$	$B_3^2, D_4^2, C_{20}^{12}$	
2	False	B	False	H_4^4, I_{10}^8	H_4^4, I_{10}^8	$H_4^4, I_{10}^8, D_4^2, C_{20}^{12}$	
3	False	H	True	-	-	$I_{10}^8, D_4^2, C_{20}^{12}$	4

4	False	I	False	ϕ	ϕ	D_4^2, C_{20}^{12}	
5	False	D	False	G_8^5	G_8^5	G_8^5, C_{20}^{12}	
6	False	G	False	ϕ	ϕ	C_{20}^{12}	
7	False	C	False	ϕ	ϕ	ϕ	
8	True						

Quá trình tìm kiếm thành công. Đường đi ngắn nhất từ A ->B->H, cost=4

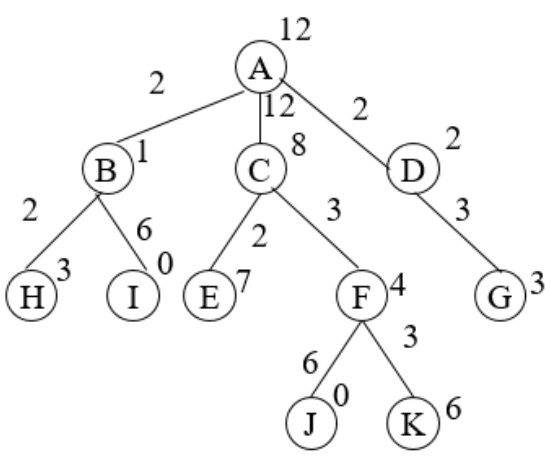
Cây tìm kiếm là:



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

Bài tập cuối chương

Bài 1: Cho đồ thị

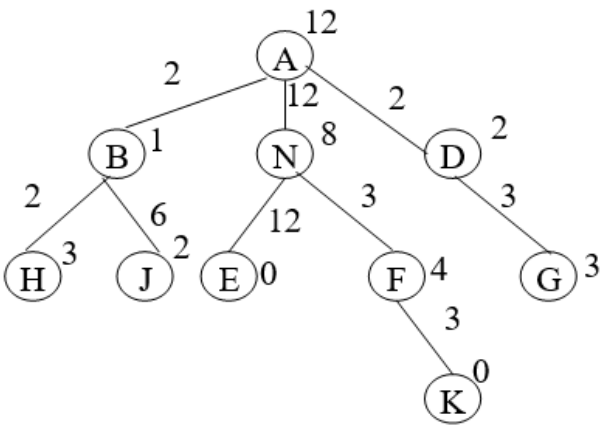


U_0=A; T={I,J}

+ Áp dụng thuật toán A* trên đồ thị

+ Áp dụng thuật toán nhánh cận đồ thị trên

Bài 2: Cho đồ thị

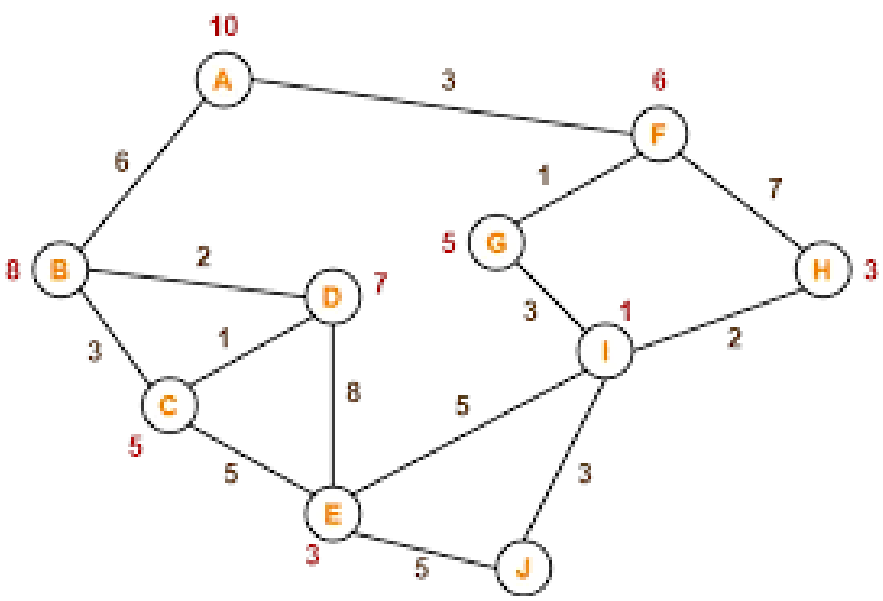


u_0=A; T={E,K}

+ Áp dụng thuật toán A* với đồ thị trên

+ Áp dụng thuật toán nhánh cận với đồ thị trên

Bài 3: Cho đồ thị

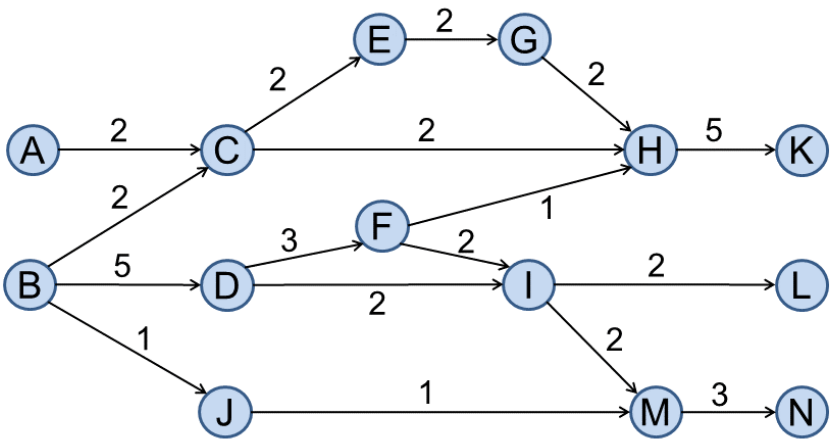


u_0=A; T={E,H}

+ Áp dụng thuật toán A* với đồ thị trên

+ Áp dụng thuật toán nhánh cận với đồ thị trên

Bài 4: Cho đồ thị có hướng

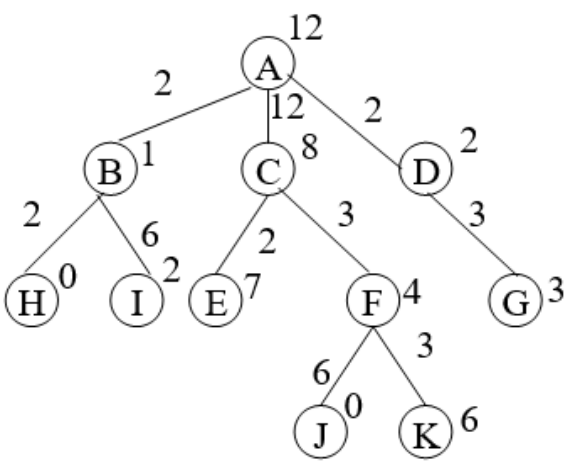


u_0=A; T={N, K}

+ Áp dụng thuật toán A^* với đồ thị trên

+ Áp dụng thuật toán nhánh cận với đồ thị trên

Bài 5: Cho đồ thị



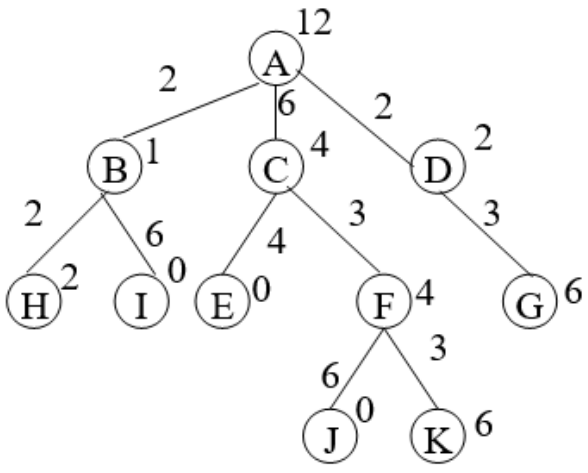
U_0=A; T={H, J}

+ Áp dụng thuật toán A^* với đồ thị trên

Bài 6: Sử dụng đồ thị đã cho của bài 5

+ Áp dụng thuật toán nhánh cận

Bài 7: Cho đồ thị



U_0=A; T={I, E, J}

+ Áp dụng thuật toán A^* với đồ thị trên

Bài 8: Sử dụng đồ thị đã cho của bài 7

+ Áp dụng thuật toán nhánh cận

Bài 9: Có n^2-1 số mang các giá trị từ 1 tới n^2-1 được sắp xếp vào một lưới các ô vuông kích thước n x n. Mỗi số đó được gọi là một quân cờ và lưới ô đó được gọi là bàn cờ. Có một vị trí của bàn cờ bỏ trống. Mỗi lần di chuyển quân, người chơi được phép chuyển một quân ở vị trí ô tiếp giáp cạnh với ô trống vào ô trống.

Từ một trạng thái ban đầu (sự sắp xếp ban đầu của các quân trên bàn cờ), thực hiện các nước đi hợp lệ để thu được trạng thái kết thúc (trạng thái đích cần đạt được).Bài toán 8-puzzle có n=8, ta minh họa trạng thái ban đầu và trạng thái kết húc qua các hình vẽ dưới đây:

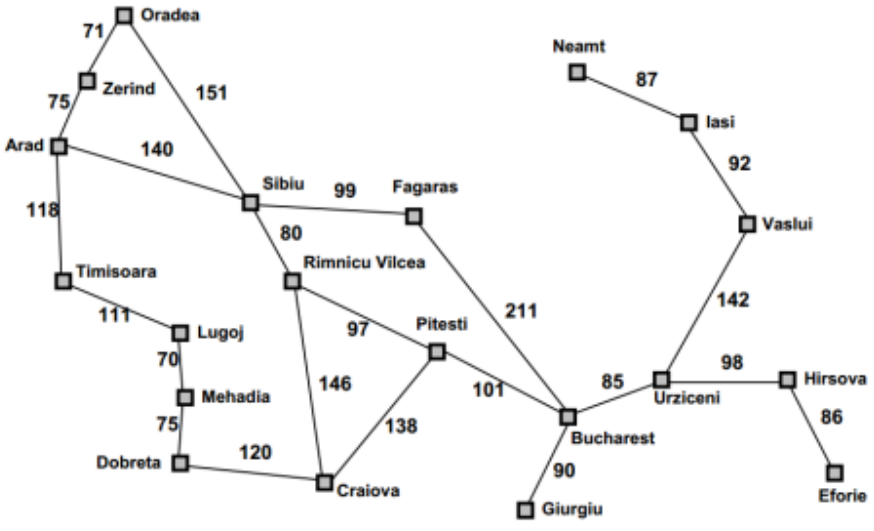
2	8	3
1	6	4
	7	5

1	2	3
8		4
7	6	5

Hãy xây dựng hàm đánh giá và sử dụng thuật toán A* giải bài toán trên

Bài 10: Sử dụng thuật toán A* tìm đường đi từ Orasea đến Bucharest với đồ thị cho như sau:

Romania with step costs in km



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

CHƯƠNG 5. CÁC CHIẾN LƯỢC TÌM KIẾM CÓ ĐỐI THỦ

Nội dung chính của chương

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

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

Mục tiêu cần đạt được của chương

Học viên cần hiểu về vấn đề tìm kiếm có đối thủ, cây trò chơi và tìm kiếm trên cây trò chơi. Hiểu được tư tưởng và thuật toán của một số chiến lược tìm kiếm như: tìm kiếm Minimax; tìm kiếm cắt cụt alpha – beta.

BÀI 5: CÁC CHIẾN LƯỢC TÌM KIẾM CÓ ĐỐI THỦ (Số tiết: 2 tiết)

5.1 Cây trò chơi và tìm kiếm trên cây trò chơi.

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

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

người chơi không biết được đối thủ của mình sẽ đi nước nào trong tương lai. Sau đây chúng ta sẽ phát biểu chính xác hơn vấn đề tìm kiếm này.

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

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

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

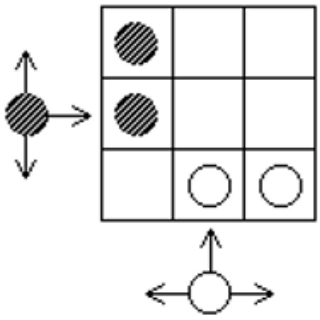
Để thuận lợi cho việc nghiên cứu các chiến lược chọn nước đi, ta biểu diễn không gian trạng thái trên dưới dạng cây trò chơi.

Cây trò chơi

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

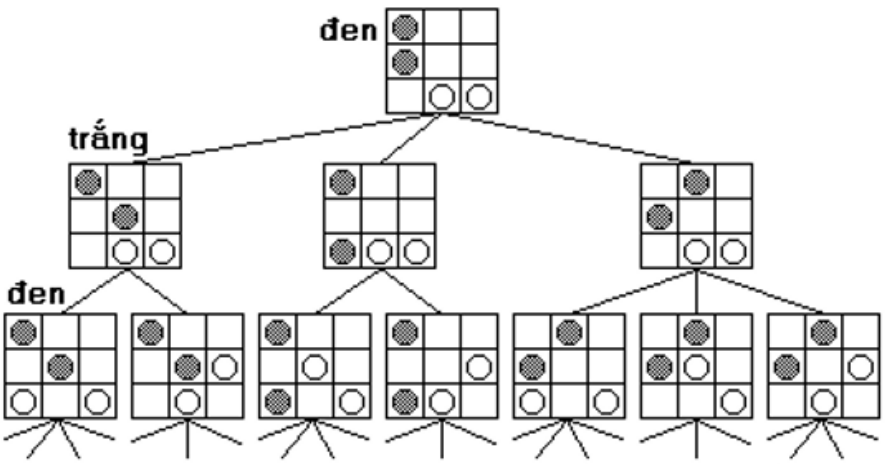
Ví dụ 17:

Xét trò chơi Dodgen (được tạo ra bởi Colin Vout). Có hai quân Trắng và hai quân Đen, ban đầu được xếp vào bàn cờ 3*3 (*Hình 5.1*). Quân Đen có thể đi tới ô trống ở bên phải, ở trên hoặc ở dưới. Quân Trắng có thể đi tới trống ở bên trái, bên phải, ở trên. Quân Đen nếu ở cột ngoài cùng bên phải có thể đi ra khỏi bàn cờ, quân Trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ. Ai đưa hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình thế bắt đối phương không đi được cũng sẽ thắng.



Hình 5.1. Trò chơi Dodgen

Giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như trong *hình 5.2*.



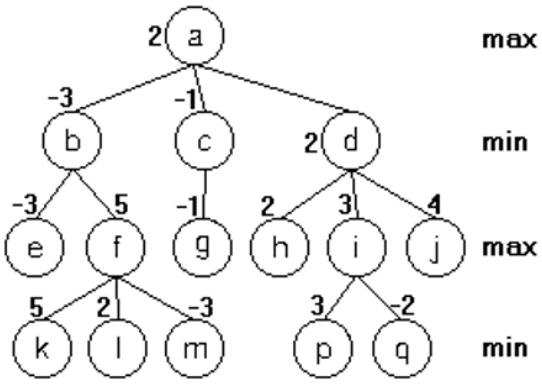
Hình 5.2. Cây trò chơi Dodgen với Đen đi trước

5.2 Chiến lược tìm kiếm Minimax

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

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

đã được xác định. Khi đó nếu v là đỉnh Trắng thì giá trị của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con. Còn nếu v là đỉnh Đen thì giá trị của nó là giá trị nhỏ nhất trong các giá trị của các đỉnh con.



Hình 5.3. Gán giá trị cho các đỉnh của cây trò chơi

Ví dụ 18: Xét cây trò chơi trong *hình 5.3*, gốc a là đỉnh Trắng. Giá trị của các đỉnh là số ghi cạnh mỗi đỉnh. Đỉnh i là Trắng, nên giá trị của nó là max(3,-2) = 3, đỉnh d là đỉnh Đen, nên giá trị của nó là min(2, 3, 4) = 2.

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

```
// Hàm MaxVal xác định giá trị cho các đỉnh Trắng
function MaxVal(u);
begin
  if u là đỉnh kết thúc then MaxVal(u) ← f(u)
  else MaxVal(u) ← max{MinVal(v) | v là đỉnh con của u}
end;
  // Hàm MinVal xác định giá trị cho các đỉnh Đen.
function MinVal(u);
begin
  if u là đỉnh kết thúc then MinVal(u) ← f(u)
  else MinVal(u) ← min{MaxVal(v) | v là đỉnh con của u}
end;
```

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

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

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

Thuật toán Minimax là thuật toán tìm kiếm theo độ sâu, ở đây ta đã cài đặt thuật toán Minimax bởi các hàm đệ quy.

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

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

Hàm đánh giá

Hàm đánh giá eval ứng với mỗi trạng thái u của trò chơi với một giá trị số eval(u), giá trị này là sự đánh giá "độ lợi thế" của trạng thái u. Trạng thái u càng thuận lợi cho Trắng thì eval(u) là số dương càng lớn; u càng thuận lợi cho Đen thì eval(u) là số âm càng nhỏ; eval(u) ≈ 0 đối với trạng thái không lợi thế cho ai cả.

Chất lượng của chương trình chơi cờ phụ thuộc rất nhiều vào hàm đánh giá. Nếu hàm đánh giá cho ta sự đánh giá không chính xác về các trạng thái, nó có thể hướng dẫn ta đi tới trạng thái được xem là tốt, nhưng thực tế lại rất bất lợi cho ta. Thiết kế một hàm

đánh giá tốt là một việc khó, đòi hỏi ta phải quan tâm đến nhiều nhân tố: các quân còn lại của hai bên, sự bố trí của các quân đó, ... ở đây có sự mâu thuẫn giữa độ chính xác của hàm đánh giá và thời gian tính của nó. Hàm đánh giá chính xác sẽ đòi hỏi rất nhiều thời gian tính toán, mà người chơi lại bị giới hạn bởi thời gian phải đưa ra nước đi.

Ví dụ 19 : Sau đây ta đưa ra một cách xây dựng hàm đánh giá đơn giản cho cờ vua. Mỗi loại quân được gán một giá trị số phù hợp với "sức mạnh" của nó. Chẳng hạn, mỗi tốt Trắng (Đen) được cho 1 (-1), mã hoặc tượng Trắng (Đen) được cho 3 (-3), xe Trắng (Đen) được cho 5 (-5) và hoàng hậu Trắng (Đen) được cho 9 (-9). Lấy tổng giá trị của tất cả các quân trong một trạng thái, ta sẽ được giá trị đánh giá của trạng thái đó. Hàm đánh giá như thế được gọi là hàm tuyến tính có trọng số, vì nó có thể biểu diễn dưới dạng: $s_1w_1 + s_2w_2 + \ldots + s_nw_n$.

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

Ví dụ 20: Bây giờ ta đưa ra một cách đánh giá các trạng thái trong trò chơi Dodgem. Trong *hình 5.4*, mỗi quân Trắng ở một vị trí trên bàn cờ được cho một giá trị tương ứng trong bảng bên trái. Còn mỗi quân Đen ở một vị trí sẽ được cho một giá trị tương ứng trong bảng bên phải.

30	35	40
15	20	25
0	5	10

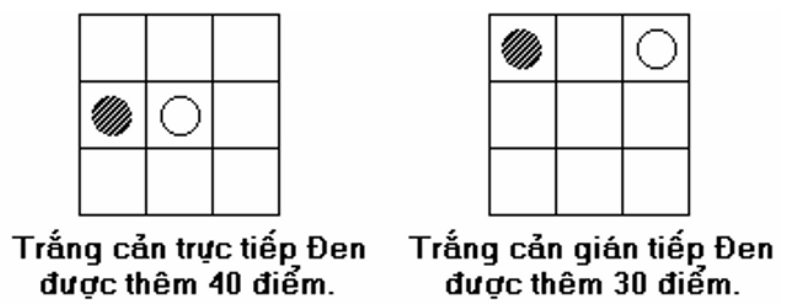
Giá trị quân Trắng.

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

Giá trị quân Đen.

Hình 5.4. Đánh giá các quân trong trò chơi Dodgem

Ngoài ra, nếu quân Trắng cản trực tiếp một quân Đen, nó được thêm 40 điểm, nếu cản gián tiếp nó được thêm 30 điểm (Xem *hình 5.5*). Tương tự, nếu quân Đen cản trực tiếp quân Trắng nó được thêm -40 điểm, còn cản gián tiếp nó được thêm -30 điểm.

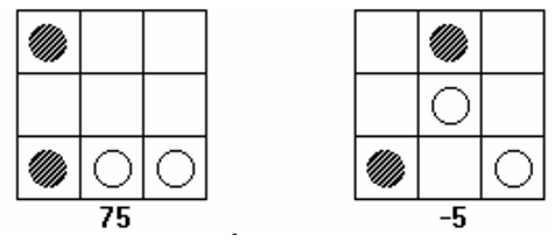


Trắng cản trực tiếp Đen được thêm 40 điểm.

Trắng cản gián tiếp Đen được thêm 30 điểm.

Hình 5.5. Đánh giá sự tương quan giữa quân Trắng và Đen

Áp dụng các qui tắc trên, như trong hình 5.6, ta tính được giá trị của trạng thái ở bên trái là 75, giá trị của trạng thái bên phải là -5.



Hình 5.6. Giá trị của một số trạng thái trong trò chơi Dodgem

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

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

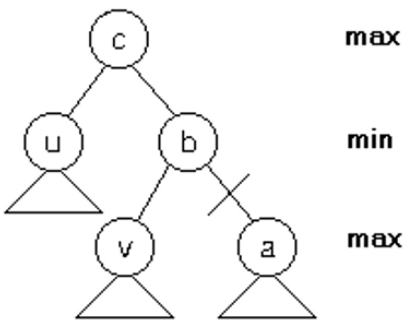
5.3 Chiến lược tìm kiếm cắt cụt alpha - beta

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

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

Tư tưởng của kỹ thuật cắt cụt alpha-beta là như sau: Nhớ lại rằng, chiến lược tìm kiếm Minimax là chiến lược tìm kiếm theo độ sâu. Giả sử trong quá trình tìm kiếm ta đi xuống đỉnh a là đỉnh Trắng, đỉnh a có người anh em v đã được đánh giá. Giả sử cha của đỉnh a là b và b có người anh em u dã được đánh giá, và giả sử cha của b là c (Xem *hình 5.7*). Khi đó ta có giá trị đỉnh c (đỉnh Trắng) ít nhất là giá trị của u, giá trị của đỉnh b (đỉnh Đen) nhiều nhất là giá trị v. Do đó, nếu eval(u) > eval(v), ta không cần đi xuống để đánh giá đỉnh a nữa mà vẫn không ảnh hưởng gì đến đánh giá đỉnh c. Hay nói cách khác ta có thể cắt bỏ cây con gốc a. Lập luận tương tự cho trường hợp a là đỉnh Đen, trong trường hợp này nếu eval(u) < eval(v) ta cũng có thể cắt bỏ cây con gốc a.

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



Hình 5.7. Cắt bỏ cây gốc a nếu eval(u) > eval(v)

Thuật toán maxval - xác định giá trị của đỉnh Trắng

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

//Thuật toán maxval - xác định giá trị của đỉnh Đen

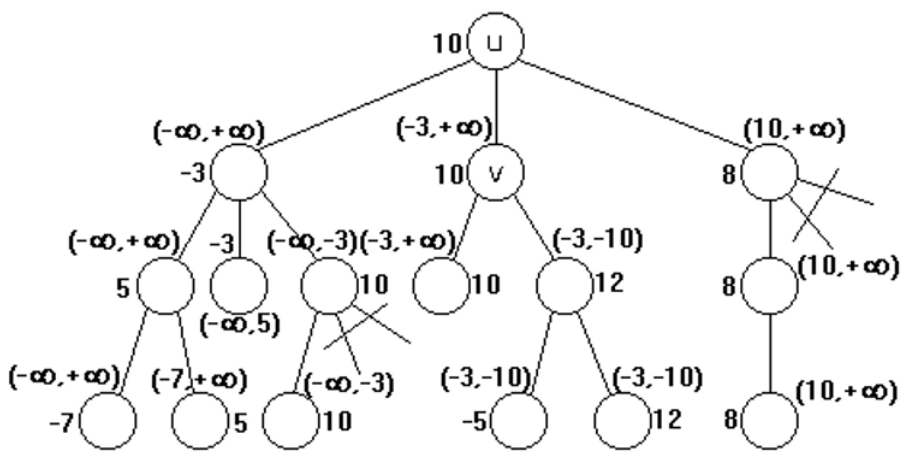
```
function MinVal(u, α, β);
begin
if u là lá của cây hạn chế hoặc u là đỉnh kết thúc
then MinVal ← eval(u)
else for mỗi đỉnh v là con của u do
        {β ← min[β, MaxVal(v, α, β)];
         // Cắt bỏ các cây con từ các đỉnh v còn lại
```

```
            if α ≥ β then exit};
    MinVal ← β;
    end;
```

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

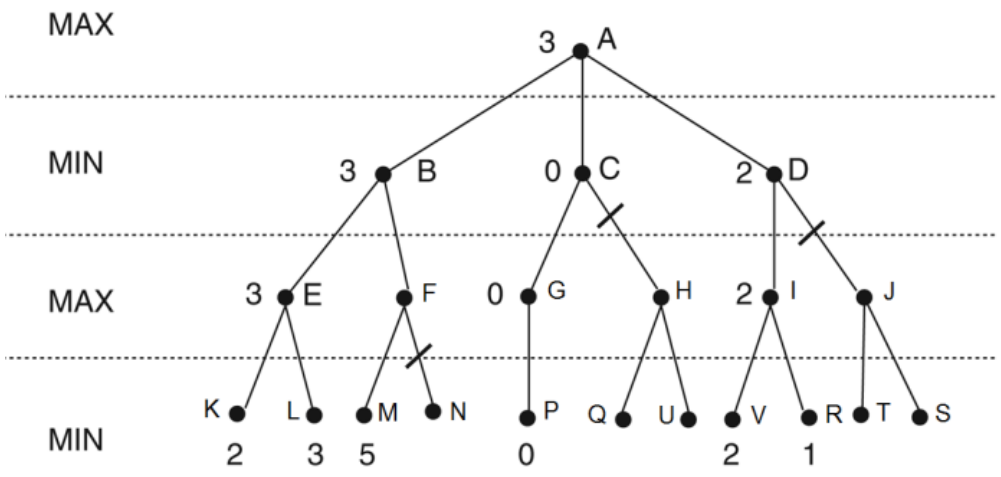
```
procedure Alpha_beta(u,v);
begin
  α ← -∞;
  β ← ∞;
  for mỗi đỉnh w là con của u do
  if α ≤ MinVal(w, α, β) then
        {α ← MinVal(w, α, β);
         v ← w;}
end;
```

Ví dụ 21: Xét cây trò chơi gốc u (đỉnh Trắng) giới hạn bởi độ cao h = 3 (hình 5.8). Số ghi cạnh các lá là giá trị của hàm đánh giá. Áp dụng chiến lược Minimax và kỹ thuật cắt cụt, ta xác định được nước đi tốt nhất cho Trắng tại u, đó là nước đi dẫn tới đỉnh v có giá trị 10. Cạnh mỗi đỉnh ta cũng cho giá trị của cặp tham số (α, β). Khi gọi các hàm MaxVal và MinVal để xác định giá trị của đỉnh đó. Các nhánh bị cắt bỏ được chỉ ra trong hình 5.8:



Hình 5.8. Xác định giá trị các đỉnh bằng kỹ thuật cắt cụt alpha-beta

Ví dụ 22: Cho không gian trạng thái như hình sau:



- Xét K(2) và L(3), khi đó E có giá trị 3=max(2,3)
- Vì M=5, nên ít nhất F=5, do đó, không cần xét nhánh N, có thể kết luận B=3 (cắt bỏ beta)
- Tương tự, xét P(0), suy ra G=0. Do chọn Min, suy ra C nhiều nhất =0. Vì A chọn max, nên không cần xét H
- Tương tự, D nhiều nhất =2, mà chọn A theo max, B > I, nên không cần xét J

Câu hỏi thường gặp

Câu 1. Tại sao một số trò chơi như cờ vua, cờ caro lại được coi là trò chơi có đối thủ?

Câu 2. Cây trò chơi là gì?

Câu 3. Tại sao việc tìm kiếm trên cây trò chơi lại giải quyết được bài toán liên quan đến trò chơi có đối thủ?

Câu 4. Tại sao chúng ta cần xây dựng hàm đánh giá để ước lượng lợi thế cho mỗi bên ở một trạng thái nhất định trong không gian trạng thái?

Câu 5. Chiến lược tìm kiếm minimax có ý nghĩa như thế nào đối với trò chơi có đối thủ?

Câu 6. Tại sao chiến lược tìm kiếm cắt cụt alpha - beta lại giúp cải thiện tốc độ tìm kiếm lời giải trên cây trò chơi?

Câu 7. Cây tìm kiếm có độ sâu quá lớn ảnh hưởng như thế nào đến chiến lược tìm kiếm minimax?

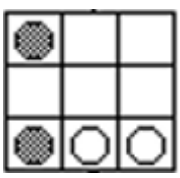
Câu 8. Trong chiến lược tìm kiếm minimax hàm đánh giá được sử dụng để ước lượng cho những trạng thái nào của cây tìm kiếm?

Câu 9. Trong tìm kiếm minimax chúng ta có phải tính giá trị của của tất cả các trạng thái bằng hàm đánh giá đã xây dựng được hay không?

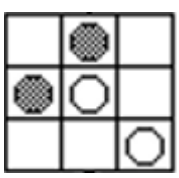
Câu 10. Nhưng tiêu chí gì cần có khi xây dựng hàm đánh giá nhằm ước lượng lợi thế cho mỗi bên trong bài toán tìm kiếm có đối thủ?

Bài tập cuối chương

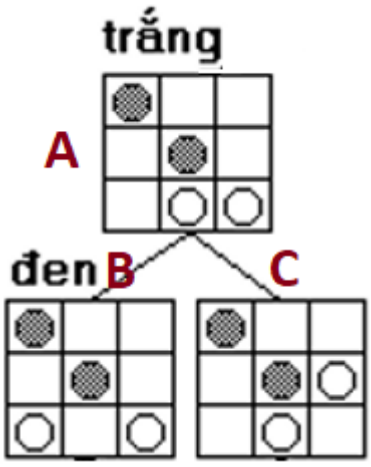
Bài 1. Hãy tính giá trị của hàm đánh giá cho trạng thái dưới đây của trò chơi Dodgem



Bài 2. Với trạng thái dưới đây của trò chơi Dodgem, theo bạn trạng thái này có lợi cho bên đen hay bên trắng? tại sao?



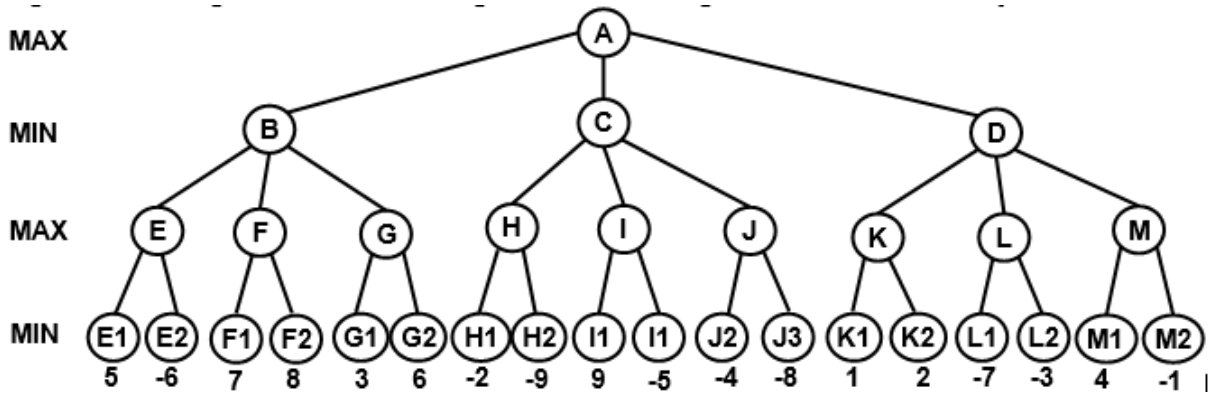
Trong trò chơi Dogem, cho không gian trạng thái như hình dưới:



Bài 3. Hãy tính giá trị của Hàm đánh giá cho các trạng thái A, B, C

Bài 4. Hãy xác định nước đi tốt nhất cho quân trắng ở trạng thái A

Cho cây trò chơi như hình dưới, các giá trị ở nút lá là giá trị của hàm đánh giá độ tốt của cây trò chơi tại nút này.

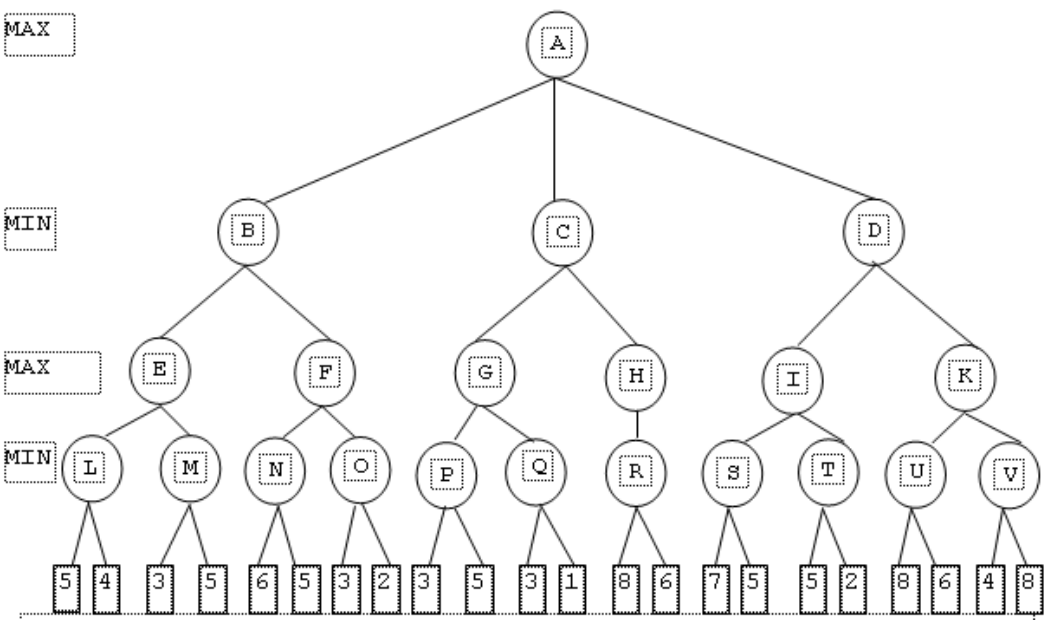


Bài 5. Hãy tính giá trị minimax cho các nút. Xác định nước đi tốt nhất cho Max.

Bài 6. Liệt kê các nút bị cắt khi dùng thuật toán cắt cụt *alpha-beta,* với thứ tự các nút con được thăm từ trái qua phải.

Bài 7. Liệt kê các nút bị cắt khi dùng thuật toán cắt cụt *alpha-beta,* với thứ tự các nút con được thăm từ phải qua trái.

Cho cây trò chơi như hình dưới, các giá trị ở nút lá là giá trị của hàm đánh giá độ tốt của cây trò chơi tại nút này.



Bài 8. Hãy tính giá trị minimax cho các nút. Xác định nước đi tốt nhất cho Max.

Bài 9. Liệt kê các nút bị cắt khi dùng thuật toán cắt cụt *alpha-beta,* với thứ tự các nút con được thăm từ trái qua phải.

Bài 10. Liệt kê các nút bị cắt khi dùng thuật toán cắt cụt *alpha-beta,* với thứ tự các nút con được thăm từ phải qua trái.

CHƯƠNG 6: BIỂU DIỄN TRI THỨC BẰNG LOGIC

Nội dung chính của chương

Giới thiệu về ngôn ngữ biểu diễn tri thức.

Biểu diễn tri thức bằng logic mệnh đề

Biểu diễn tri thức bằng logic vị từ cấp 1

Mục tiêu cần đạt được của chương

Sau khi học xong chương này, sinh viên có thể:

Hiểu và nắm được các thành phần của ngôn ngữ biểu diễn tri thức.

Vận dụng được cú pháp, ngữ nghĩa và các luật suy diễn trong logic mệnh đề để biểu diễn tri thức và lập luận.

Vận dụng được cú pháp, ngữ nghĩa và các luật suy diễn trong logic vị từ cấp 1 để biểu diễn tri thức và lập luận.

Bài 6: BIỂU DIỄN TRI THỨC BẰNG LOGIC MỆNH ĐỀ (Số tiết: 3 tiết)

6.1. Ngôn ngữ biểu diễn tri thức

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

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

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

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

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

Ngữ nghĩa của ngôn ngữ cho phép ta xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực. Chẳng hạn, trong ngôn ngữ các biểu thức số học, dãy ký hiệu (x+y)*z là một câu viết đúng cú pháp. Ngữ nghĩa của ngôn ngữ này cho phép ta hiểu rằng, nếu x, y, z, ứng với các số nguyên, ký hiệu + ứng với phép toán cộng, còn * ứng với phép nhân, thì biểu thức (x+y)*z biểu diễn quá trình tính toán: lấy số nguyên x cộng với số nguyên y, kết quả được nhân với số nguyên z.

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

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

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

Trong tài liệu này, chúng ta sẽ tập trung nghiên cứu logic vị từ cấp một (first-order predicate logic hoặc first-order predicate calculus) - một ngôn ngữ biểu diễn tri thức, bởi vì logic vị từ cấp một có khả năng biểu diễn tương đối tốt, và hơn nữa nó là cơ sở cho nhiều ngôn ngữ biểu diễn tri thức khác, chẳng hạn toán hoàn cảnh (situation

calculus) hoặc logic thời gian khoảng cấp một (first-order interval tempral logic). Nhưng trước hết chúng ta sẽ nghiên cứu logic mệnh đề (propositional logic hoặc propositional calculus). Nó là ngôn ngữ rất đơn giản, có khả năng biểu diễn hạn chế, song thuận tiện cho ta đưa vào nhiều khái niệm quan trọng trong logic.

6.2. Logic mệnh đề

6.2.1. Cú pháp

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

* Các ký hiệu

Hai hằng logic True và False.

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

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

Các dấu mở ngoặc (và đóng ngoặc).

* Các quy tắc xây dựng các công thức

Các biến mệnh đề là công thức.

Nếu A và B là công thức thì:

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

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

($\neg$A) (đọc "phủ định A")

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

(A$\Leftrightarrow$B) (đọc "A và B kéo theo nhau")

là các công thức.

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

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

6.2.2. Ngữ nghĩa

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

Một cách chính xác hơn, cho ta hiểu một minh họa là một cách gán cho mỗi ký hiệu mệnh đề một giá trị chân lý **True** hoặc **False.** Trong một minh họa, nếu kí hiệu mệnh đề P được gán giá trị chân lý **True/False** (P ←**True**/ P←**False**) thì ta nói mệnh đề P **đúng/sai** trong minh họa đó. Trong một minh họa, ý nghĩa của các câu phức hợp được xác định bởi ý nghĩa của các kết nối logic. Chúng ta xác định ý nghĩa của các kết nối logic trong các bảng chân lý (xem bảng 6.1)

Bảng 6.1. Bảng giá trị chân lý của các kết nối logic

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

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

Bảng chân lý cho phép ta xác định ngẫu nhiên các câu phức hợp. Chẳng hạn ngữ nghĩa của các câu P∧Q trong minh họa {P ← **True** , Q← **False** } là **False**. Việc xác định ngữ nghĩa của một câu (P ∨ Q) ∧ ⌉S trong một minh họa được tiến hành như sau:

đầu tiên ta xác định giá trị chân lý của P ∨ Q và ⌉ S , sau đó ta sử dụng bảng chân lý ∧ để xác định giá trị (P∨Q) ∧⌉S

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

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

Bằng cách lập bảng chân lý (*phương pháp bảng chân lý*) là ta có thể xác định được một công thức có *thoả được* hay không. Trong bảng này, mỗi biến mệnh đề đứng đầu với một cột, công thức cần kiểm tra đứng đầu một cột, mỗi dòng tương ứng với một minh họa. Chẳng hạn bảng 6.2 là bảng chân lý cho công thức (P⇒Q) ∧S. Trong bảng chân lý này ta cần đưa vào các cột phụ ứng với các công thức con của các công thức cần kiểm tra để việc tính giá trị của công thức này được dễ dàng. Từ bảng chân lý ta thấy rằng công thức (P⇒Q) ∧S là *thoả được* nhưng không *vững chắc*.

Bảng 6.2. Bảng chân lý cho công thức (P⇒Q) ∧S

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

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

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

6.2.3. Dạng chuẩn tắc

Sự tương đương của các công thức

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

$A \Rightarrow B \equiv \neg A \vee B$

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

$\neg(\neg A) \equiv A$

Luật De Morgan

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

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

Luật giao hoán

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

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

Luật kết hợp

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

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

Luật phân phối

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

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

*** Dạng chuẩn tắc :**

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

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

Ví dụ 23: Ta chuẩn hóa công thức $(P \Rightarrow Q) \vee \neg (R \vee \neg S)$:

$(P \Rightarrow Q) \vee \neg(R \vee \neg S) \equiv (\neg P \vee Q) \vee (\neg R \wedge S) \equiv ((\neg P \vee Q) \vee \neg R) \wedge ((\neg P \vee Q) \vee S) \equiv (\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee S)$.

Như vậy công thức $(P \Rightarrow Q) \vee \neg(R \vee \neg S)$ được đưa về dạng chuẩn hội

$(\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee S)$.

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

Các câu Horn:

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

$$\neg P_1 \vee ... \vee \neg P_m \vee Q_1 \vee ... \vee Q_m$$

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

$(\neg P1 \vee ... \vee \neg Pm) \vee (Q1 \vee ... \vee Qn)$

$\equiv \neg (P1 \wedge \ldots \wedge Pm) \vee (Q1 \vee ... \vee Qn)$

$\equiv (P1 \wedge \ldots \wedge Pm) \Rightarrow (Q1 \vee ... \vee Qn)$

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

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

Nếu m>0, n=1, câu Horn có dạng :

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

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

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

Khi m=0, n=1 câu Horn trở thành câu đơn Q, hay sự kiện Q. Nếu m>0, n=0 câu Horn trở thành dạng $\neg P_1 \vee ... \vee \neg P_m$ hay tương đương $\neg (P1 \wedge \ldots \wedge Pm)$. Cần chú ý rằng, không phải mọi công thức đều có thể biểu diễn dưới dạng hội của các câu Horn. Tuy nhiên trong các ứng dụng, cơ sở tri thức thường là một tập nào đó các câu Horn (tức là một tập nào đó các luật if-then).

Câu hỏi

Câu 1. Ngôn ngữ biểu diễn tri thức ?

Câu 2. Cú pháp của ngôn ngữ biểu diễn tri thức ?

Câu 3. Ngữ nghĩa của ngôn ngữ biểu diễn tri thức ?

Câu 4. Cú pháp của logic mệnh đề ?

Câu 5. Hai công thức tương đương ?

Bài tập

Bài 1. Hãy biểu diễn tri thức sau bởi logic mệnh đề :

Nam học giỏi, thông minh, đẹp trai

Nam học giỏi hoặc thông minh

Nam thông minh thì học giỏi

Bài 2. Hãy chuẩn hóa các công thức sau

a. $((A \Rightarrow B) \wedge C) \Rightarrow (\neg D \vee E)$

b. $(A \vee B) \Rightarrow (\neg C \vee \neg D \vee E)$

c. $(A \vee B) \Rightarrow ((C \Rightarrow D) \Rightarrow E)$

d. $(A \vee B) \Rightarrow ((C \wedge D) \Rightarrow E)$

e. $(A \Rightarrow B) \Rightarrow ((C \wedge D) \Rightarrow E)$

f. $(A \Rightarrow B) \vee ((C \Rightarrow D) \rightarrow E)$

g. $(A \Rightarrow B) \vee ((C \Rightarrow D) V E)$

Bài 7: BIỂU DIỄN TRI THỨC BẰNG LOGIC MỆNH ĐỀ (TIẾP) (Số tiết: 3 tiết)

6.2. Logic mệnh đề (Tiếp)

6.2.4. Luật suy diễn

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

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

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

1. Luật Modus Ponens

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

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

2. Luật Modus Tollens

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

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

3. Luật bắc cầu

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

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

4. Luật loại bỏ hội

$$\frac{\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_n}{\alpha_i}$$

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

5. Luật đưa vào hội

$$\frac{\alpha_1, \alpha_2, ..., \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_n}$$

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

6. Luật đưa vào tuyển

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee ... \vee \alpha_n}$$

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

7. Luật giải

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

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

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

Bằng phương pháp bảng chân lý, ta có thể kiểm chứng được các luật suy diễn nêu trên đều là tin cậy. Bảng chân lý của luật giải được cho trong bảng 6.3. Từ bảng này ta thấy rằng, trong bất kỳ một minh họa nào mà cả hai giả thiết $\alpha \vee \beta$, $\neg\beta \vee \gamma$ đúng thì kết luận $\alpha \vee \gamma$ cũng đúng. Do đó luật giải là luật suy điễn tin cậy.

Bảng 6.3 Bảng giá trị chân lý chứng minh tính tin cậy của luật giải.

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

False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

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

Tiên đề định lý chứng minh.

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

Ví dụ 24: Giả sử ta có các công thức sau :

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

$P \Rightarrow Q$ (2)

$R \Rightarrow S$ (3)

P (4)

R (5)

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

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

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

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

Phương pháp chứng minh bác bỏ

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

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

6.2.5. Luật phân giải và chứng minh bác bỏ bằng luật giải

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

* Luật giải trên các câu tuyển

$$\frac{A_1 \vee A_2 \vee ... \vee A_m \vee C,\ \neg C \vee B_1 \vee B_2 \vee ... \vee B_n}{A_1 \vee A_2 \vee ... \vee A_m \vee B_1 \vee B_2 \vee ... \vee B_n}$$

trong đó Ai, Bj và C là các literal.

* Luật giải trên các câu Horn:

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

$$\frac{P_1 \wedge P_2 \wedge ... \wedge P_m \wedge S \Rightarrow Q,\ R_1 \wedge R_2 \wedge ... \wedge R_m \Rightarrow S}{P_1 \wedge P_2 \wedge ... \wedge P_m \wedge R_1 \wedge R_2 \wedge ... \wedge R_m \Rightarrow Q}$$

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

$$\frac{P_1 \wedge P_2 \wedge ... \wedge P_m \wedge S \Rightarrow Q,\ S}{P_1 \wedge P_2 \wedge ... \wedge P_m \Rightarrow Q}$$

Khi ta có thể áp dụng luật giải cho hai câu, thì hai câu này được gọi là *hai câu giải được* và kết quả nhận được khi áp dụng luật giải cho hai câu đó được gọi là *giải thức* của chúng. Giải thức của hai câu A và B được kí hiệu là res(A,B). Chẳng hạn, hai câu tuyển giải được nếu một câu chứa một literal đối lập với một literal trong câu kia. Giải

thức của hai literal đối lập nhau (P và ⌝P) là câu rỗng, chúng ta sẽ ký hiệu câu rỗng là □, câu rỗng là không thoả được.

Giả sử G là một tập các câu tuyển (Bằng cách chuẩn hoá ta có thể đưa một tập các công thức về một tập các câu tuyển). Ta sẽ ký hiệu R(G) là tập câu bao gồm các câu thuộc G và tất cả các câu nhận được từ G bằng một dãy áp dụng luật giải.

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

Định lý giải:

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

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

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

procedure Resolution;

Input: tập G các câu tuyển ;

begin

1.*Repeat*

1.1 Chọn hai câu A và B thuộc G ;

1.2 *if* A và B giải được *then* tính Res (A,B);

1.3 *if* Res (A,B) là câu mới *then* thêm Res (A,B) vào G;

until

nhận được □ hoặc không có câu mới xuất hiện;

2. *if* nhận được câu rỗng *then* thông báo G không thoả được

e lse thông báo G thoả được ;

end;

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

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

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

Procedure Refutation_Proof;

input : Tập G các công thức;

Công thức cần chứng minh H;

Begin

1. Thêm ⌝H vào G;
2. Chuyển các công thức trong G về dạng chuẩn hội;
3. Từ các dạng chuẩn hội ở bước hai, thành lập tạp các câu tuyển *G'*;
4. áp dụng thủ tục giải cho tập câu *G'*;
5. *if* G' không thoả được *then* thông báo H là hệ quả logic
   else thông báo H không là hệ quả logic của G ;

end;

***Ví dụ 25*:** Giả giử G là tập hợp các câu tuyển sau

⌝A ∨ ⌝B ∨ P (1)

⌝C ∨ ⌝D ∨ P (2)

⌝E ∨ C (3)

A (4)

E (5)

D (6)

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

⌝P (7)

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

⌝C ∨ ⌝D (8)

Từ câu (6) và (8) ta nhận được câu:

⌝C (9)

Từ câu (3) và (9) ta nhận được câu:

⌝E (10)

Tới đây đã xuất hiện mâu thuẫn, vì câu (5) và (10) đối lập nhau. Từ câu (5) và (10) ta nhận được câu rỗng □. Vậy P là hệ quả logic của tập các câu tuyển thuộc G đã cho.

Câu hỏi

Câu 1. Một công thức H là hệ quả logic của tập công thức $G = \{G_1,...,G_m\}$?

Câu 2. Trình bày các luật suy diễn cơ bản trong logic mệnh đề.

Câu 3. Định lý phân giải ?

Câu 4. Tư tưởng của phương pháp chứng minh bác bỏ ?

Câu 5. Thuật toán chứng minh bác bỏ bằng luật phân giải.

Bài tập

Bài 1

Cho tập công thức sau

$A \vee B$

$B \Rightarrow (C \vee D)$

$C \Rightarrow (E \wedge F)$

$\neg E$

$\neg D$

Hãy CM: A là hệ quả logic

Bài 2

Cho tập công thức sau

$A \Rightarrow (B \wedge I)$

$(C \wedge D) \Rightarrow A$

$(B \vee E) \Rightarrow F$

D

C

Hãy CM: F là hệ quả logic

Bài 8: BIỂU DIỄN TRI THỨC BẰNG LOGIC VỊ TỪ CẤP 1 (Số tiết: 3 tiết)

6.3. Logic vị từ cấp một

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

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

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

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

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

6.3.1. Cú pháp và ngữ nghĩa của logic vị từ cấp 1

**** Cú pháp:***

+ Các ký hiệu.

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

Các ký hiệu hằng: a, b, c, An, Ba, John,...

Các ký hiệu biến: x, y, z, u, v, w,...

Các ký hiệu vị từ: P, Q, R, S, Like, Havecolor, Prime,...

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

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

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

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

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

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

Các hạng thức

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

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

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

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

Các công thức phân tử

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

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

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

Nếu $t_1, t_2,...,t_n$ là n hạng thức và p là vị từ của n biến thì p($t_1,t_2,...,t_n$) là câu đơn. Chẳng hạn, Hoa là một ký hiệu hằng, Love là một vị từ của hai biến, husband là hàm của một biến, thì Love (Hoa, husband(Hoa)) là một câu đơn.

Các công thức

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

Các công thức được xác định đệ quy như sau:

Các công thức phân tử là công thức.

Nếu G và H là các công thức, thì các biểu thức ($G \wedge H$), ($G \vee H$), ($\neg G$), ($G \Rightarrow H$), ($G \Leftrightarrow H$) là công thức.

Nếu G là một công thức và x là biến thì các biểu thức ($\forall$ x G), ($\exists$ x G) là công thức.

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

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

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

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

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

Ví dụ 26:

Công thức $\forall$xP(x, f(a, x)) $\wedge$ $\exists$y Q(y) là công thức đóng

Công thức $\forall$xP(x, f(y, x)) không phải là công thức đóng, vì sự xuất hiện của biến y trong công thức này không chịu ràng buộc bởi một lượng tử nào cả (Sự xuất hiện của y gọi là *sự xuất hiện tự do*).

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

**** Ngữ nghĩa.***

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

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

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

+ Ngữ nghĩa của các câu đơn .

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

+ Ngữ nghĩa của các câu phức hợp.

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

Ví dụ 25:

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

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

+ Ngữ nghĩa của các câu chứa các lượng tử.

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

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

Ngữ nghĩa của công thức ∃xG được xác định như là ngữ nghĩa của công thức là tuyển của tất cả các công thức nhận được từ G bằng cách thay x bởi một đối tượng trong miền đối tượng.

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

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

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

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

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

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

- Đối tượng: một cái bàn, một cái nhà, một cái cây, một con người, một con số...
- Tính chất: Cái bàn có thể có tính chất: có bốn chân, làm bằng gỗ, không có ngăn kéo. Con số có thể có tính chất là số nguyên, số hữu tỉ, là số chính phương...

- Quan hệ: cha con, anh em, bè bạn (giữa con người); lớn hơn nhỏ hơn, bằng nhau (giữa các con số); bên trong, bên ngoài nằm trên nằm dưới (giữa các đồ vật)...
- Hàm: Một trường hợp riêng của quan hệ là quan hệ hàm. Chẳng hạn, vì mỗi người có một mẹ, do đó ta có quan hệ hàm ứng mỗi người với mẹ của nó.

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

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

6.3.1. Cú pháp và ngữ nghĩa của logic vị từ cấp 1

6.3.1.1. Cú pháp

Các ký hiệu.

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

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

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

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

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

- Các ký hiệu kết nối logic: $\wedge$ (hội), $\vee$ (tuyển), $\neg$ (phủ định), $\Rightarrow$(kéo theo), $\Leftrightarrow$ (kéo theo nhau).
- Các ký hiệu lượng tử: $\forall$ (với mọi), $\exists$ (tồn tại).
- Các ký hiệu ngăn cách: dấu phẩy, dấu mở ngoặc và dấu đóng ngoặc.

Các hạng thức

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

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

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

Các công thức phân tử

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

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

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

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

Các công thức

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

Các công thức được xác định đệ quy như sau:

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

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

- Lượng tử phổ dụng ($\forall$) cho phép mô tả tính chất của cả một lớp các đối tượng, chứ không phải của một đối tượng, mà không cần phải liệt kê ra tất cả các đối tượng trong lớp. Chẳng hạn sử dụng vị từ Elephant(x) (đối tượng x là con voi) và vị từ Color(x, Gray) (đối tượng x có mầu xám) thì câu " tất cả các con voi

đều có mầu xám" có thể biểu diễn bởi công thức $\forall x$ (Elephant(x) $\Rightarrow$ Color(x, Gray)).

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

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

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

Ví dụ 27:

Công thức $\forall x$P(x, f(a, x)) $\wedge$ $\exists y$ Q(y) là công thức đóng

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

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

6.3.1.2. Ngữ nghĩa.

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

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

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

<u>Ngữ nghĩa của các câu đơn .</u>

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

Ngữ nghĩa của các câu phức hợp.

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

Ví dụ 28:

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

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

Ngữ nghĩa của các câu chứa các lượng tử.

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

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

Ngữ nghĩa của công thức $\exists$x G được xác định như là ngữ nghĩa của công thức là tuyển của tất cả các công thức nhận được từ G bằng cách thay x bởi một đối tượng trong miền đối tượng.

Chẳng hạn, nếu ngữ nghĩa của câu Younger(x,20) là "x trẻ hơn 30 tuổi" và miền đối tượng gồm ba người {Lan, An, Hoa} thì ngữ nghĩa của câu $\exists$x Yourger(x,20) là ngữ nghĩa của câu Yourger(Lan,20) $\vee$ Yourger(An,20) $\vee$ Yourger(Hoa,20). Câu này nhận giá trị True nếu và chỉ nếu ít nhất một trong ba người Lan, An, Hoa trẻ hơn 20.

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

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

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

Câu hỏi

Câu 1. Cú pháp của logic vị từ cấp 1?

Câu 2. Ngữ nghĩa của câu đơn trong logic vị từ cấp 1?

Câu 3. Ngữ nghĩa của câu phức trong logic vị từ cấp 1?

Câu 4. Ngữ nghĩa của câu chứa lượng tử trong logic vị từ cấp 1?

Bài tập

Bài 1. Hãy biểu diễn tri thức sau trong logic vị từ cấp 1

An là sinh viên

Mọi sinh viên đều vui vẻ

Sinh viên Khoa CNTT giỏi lập trình

Bài 2. Hãy biểu diễn tri thức sau bởi logic vị từ cấp 1

Hải quen Nam

Nam là sinh viên

Mọi sinh viên đều chăm chỉ

Bài 9: BIỂU DIỄN TRI THỨC BẰNG LOGIC VỊ TỪ CẤP 1 (TIẾP) (Số tiết: 3 tiết)

6.3. Logic vị từ cấp một (tiếp)

6.3.2. Chuẩn hóa công thức

**** Các công thức tương đương***

Cũng như trong logic mệnh đề, ta nói hai công thức G và H tương đương (viết là G ≡ H) nếu chúng cùng đúng hoặc cùng sai trong mọi minh hoạ. Ngoài các tương đương đã biết trong logic mệnh đề, trong logic vị từ cấp một còn có các tương đương khác liên

quan tới các lượng tử. Giả sử G là một công thức, cách viết G(x) nói rằng công thức G có chứa các xuất hiện của biến x. Khi đó công thức G(y) là công thức nhận được từ G(x) bằng cách thay tất cả các xuất hiện của x bởi y. Ta nói G(y) là công thức nhận được từ G(x) bằngcách *đặt tên lại* (biến x được đổi tên lại là y).

Chúng ta có các tương đương sau đây:

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

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

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

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

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

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

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

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

**** Chuẩn hóa công thức***

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

Trong mục này chúng ta sẽ trình bày thủ tục chuyển một câu phức hợp thành một câu ở dạng chuẩn tắc hội tương đương. Thủ tục chuẩn hoá các công thức gồm các bước sau:

Bước 1. Loại bỏ các kéo theo

Để loại bỏ các kéo theo hoặc kéo theo nhau, ta chỉ cần:

- Thay công thức $P \Rightarrow Q$ bởi công thức tương đương $\neg P \vee Q$
- Thay $P \Leftrightarrow Q$ bởi $(\neg P \vee Q) \wedge (\neg Q \vee P)$

Bước 2. Chuyển các phủ định tới các phân tử

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

$$\neg(\neg P) \equiv P$$
$$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$$
$$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$$
$$\neg(\exists x\, P) \equiv \forall x\, (\neg P)$$
$$\neg(\forall x\, P) \equiv \exists x\, (\neg P)$$

Bước 3. Loại bỏ các lượng tử tồn tại

Giả sử *P(x,y)* là các vị từ có nghĩa: "*y* lớn hơn *x*" trong miền các số. Khi đó, công thức$\forall x\, \exists y(P(x,y))$ có nghĩa là "với mọi số *x*, tồn tại *y* sao cho số *y* lớn hơn *x*". Ta có thể

xem *y* trong công thức đó là hàm của đối số *x*. Chẳng hạn, loại bỏ lượng tử ∃*y*, công thức đang xét trở thành ∀*x(P(x,f(x)))*.

Một cách tổng quát, giả sử ∃*y (G)* là một công thức con của công thức đang xét và nằmtrong miền tác dụng của các lượng tử ∀*x1,...,*∀*xn*. Khi đó, có thể xem *y* là hàm của *n* biến *x1,…,xn.*của ví dụ *f(x1...xn)*. Sau đó, thay các xuất hiện của *y* trong công thức *G* bởi hạng thức *(x1...xn)* và loại bỏ các lượng tử tồn tại. Hàm *f* được đưa vào để loại bỏ các lượng tử tồn tại được gọi là ***hàm Skolem*.**

Ví dụ 29: xét công thức sau:

∀*x (*∃*y (P(x,y)* ∨ ∀*u (*∃**v** *(Q(a,* **v***)* ∧ ∃*y* ⌉ *R(x,y)))* *(1)*

Công thức con ∃y P(x,y) nằm trong miền tác dụng của lượng tử ∀x, ta xem y là hàm của x: f(x).

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

∀*x (P(x,f(x))* ∨ ∀*u (Q(a,g(x,u))* ∧ ⌉ *R(x,h(x,u))))* *(2)*

Bước 4. Đổi tên các biến (nếu cần)

Đổi tên các biến sao cho các biến sau dấu ∀ có tên khác nhau. Ví dụ: cho công thức ∀xP(x) ∧ ∀x∀yQ(x,y) thì chuyển thành ∀xP(x) ∧ ∀z∀yQ(z,y)

Bước 5. Loại bỏ các lượng tử phổ dụng

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

P(x,f(x)) ∨ *(Q(a,g(x,u))* ∧⌉*R(x,h(x,u)))* (3)

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

Bước 6. Chuyển các tuyển tới các literal

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

(P(x,f(x)) ∨ *(Q(a,g(x,u)))* ∧ *(P(x,f(x))* ∨ ⌉*R(x,h(x,u)))* (4)

Bước 7. Loại bỏ các hội

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

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

Bước 8. Đặt tên lại các biến

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

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

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

6.3.3. Các luật suy diễn

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

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

- **Luật thay thế phổ dụng:**

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

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

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

• Hợp nhất

Trong luật thay thế phổ dụng, ta cần sử dụng phép thế các biến bởi các hạng thức để nhận được các công thức mới từ công thức chứa các lượng tử phổ dụng. Ta có thể sử dụng phép thế để hợp nhất các câu phân tử (tức là để các câu trở thành đồng nhất). Chẳng hạn xét hai câu phân tử Like(An, y), Like(x, Football). Cần lưu ý rằng hai câu này là hai câu ∀y Like(An,y) và ∀xLike(x,Football) mà để cho đơn giản ta bỏ đi các lượng tử phổ dụng. Sử dụng phép thế [x/An, y/Football] hai câu trên trở thành đồng nhất Like(An,Football). Trong các suy diễn, ta cần sử dụng phép hợp nhất các câu bởi các phép thế. Chẳng hạn, cho trước hai câu Friend(x,Ba) Good(x) (Mọi bạn của Ba đều là người tốt) Friend(Lan,y) (Lan là bạn của tất cả mọi người) Ta có thể hợp nhất hai câu Friend(x,Ba), Good(x) và Friend(Lan,y) bởi phép thay thế [x/Lan,y/Ba]. áp dụng luật thay thế phổ dụng với phép thay thế này ta nhận được hai câu: Friend(Lan,Ba), Good(Lan) Friend(Lan,Ba). Từ hai câu này, theo luật Modus Ponens, ta suy ra câu Good(Lan) (Lan là người tốt).

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

Với hai câu phân tử G và H mà tồn tại phép thế θ sao cho Gθ và Hθ trở thành đồng nhất (Gθ=Hθ) thì G và H được gọi là ***hợp nhất được***, phép thế θ được gọi là ***hợp nhất tử*** của G và H.

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

• Luật Modus Ponens tổng quát.

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

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

Trong đó Q' =Qθ.

Ví dụ 30: Giả sử ta có các câu (Student(x)∧Male(x)∧ Like (x,Football)) và Student(Anh), Male(Anh). Với phép thế θ=[x|Anh], các cặp câu Student(x),Student(Anh) và Male(x), Male(Anh) hợp nhất được. Do đó ta suy ra câu Like(Anh,Football).

Luật phân giải tổng quát

• Luật phân giải trên các câu tuyển

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

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

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

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

• Luật phân giải trên các câu Horn:

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

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

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

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

$$\frac{(P_1 \wedge \ldots \wedge P_m \wedge S) \Rightarrow Q,\quad (R_1 \wedge \ldots \wedge R_n) \Rightarrow T}{(P_1' \wedge \ldots \wedge P_m' \wedge R_1' \wedge \ldots \wedge R_n) \Rightarrow Q'}$$

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

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

$$\frac{(P_1 \wedge \ldots \wedge P_m \wedge S) \Rightarrow Q,\quad T}{(P_1' \wedge \ldots \wedge P_m') \Rightarrow Q'}$$

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

Ví dụ 31: Xét hai câu (Student(x) ∧ Male(x))⇒Play(x,Football) và Male(Ba). Hai câu Male(Ba) và Male(x) hợp nhất được với phép thế [x|Ba], do đó từ hai câu trên ta suy ra Student (Ba) ⇒ Play (Ba, Football).

Câu hỏi

Câu 1. Các bước chuẩn hóa công thức trong logic vị từ cấp 1

Câu 2. Sự tương đương của các công thức ?

Câu 3. Liệt kê các luật suy diễn ?

Câu 4. Phân biệt bước 4 và bước 8 trong thủ tục chuẩn hóa.

Câu 5. Phép thế ?

Bài tập

Đưa các câu sau về dạng chuẩn tắc hội.

a) $\exists x(\neg\exists yP(x,y)\wedge\forall yQ(x,y)) \Rightarrow\forall xR(x)$

b) $\forall x\forall y(\exists zP(x,y,z)\wedge Q(x,y))\vee\forall xR(x)$

c) $\forall x\forall y((\exists zP(x,y,z)\wedge Q(x,y))\wedge\forall xR(x)$

d) $(\forall xP(x) \Rightarrow\exists x\forall yQ(x,y)) \Rightarrow\forall yR(y)$

e) $(\exists xP(x)\vee\exists xQ(x)) \Rightarrow (\forall x\exists yR(x,y)\wedge\forall xA(x))$

Bài 10: BIỂU DIỄN TRI THỨC BẰNG LOGIC VỊ TỪ CẤP 1 (TIẾP) (Số tiết: 3 tiết)

6.3. Logic vị từ cấp một (tiếp)

6.3.4. Thuật toán hợp nhất

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

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

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

Know(An, x)

Know(y, husband(z))

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

Know(An, husband(z))

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

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

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

…

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

Ví dụ 32: Xét hai phép thế:

θ= [x|a, v|g(y,z)]

η = [x|b, y|c, z|f(x), v|h(a)]

khi đó hợp thành của chúng là phép thế

θη= [x|a, v|g(c,f(x)), y|c, z|f(x)]

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

Chẳng hạn, phép thế đã đưa ra ở trên:

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

tổng quát hơn phép thế:

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

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

θ=[y|An, x|husband(z)]

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

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

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

Ví dụ 33: Sự khác biệt giữa hai câu Like(x,f(a,g(z))) và Like(x,y) là cặp (f(a,g(z)),y) còn sự khác biệt giữa hai câu Know(x,f(a,u)) và Know(x,f(a,g(b)) là cặp (u,g(b)).

Procedure Unify(E,F,θ);
Begin
1.Xác định sự khác biệt giữa E và F;
2.if không có sự khác biệt then{thông báo thành công;stop};
3.if sự khác biệt là cặp (x,t),trong đó x là biến, t là hạng thức không chứa x then
{3.1 E←E[x|t]; F←F[x|t];
// tức là áp dụng phép thế {x|t} vào các biểu thức E và F
3.2 θ←θ [x/t]; // hợp thành của phép thế q và phép thế [x|t]
3.3 Unify(E,F, θ);
}
else {thông báo thất bại; stop}
end;

Nhận xét:

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

Ví dụ 34: Xét hai biểu thức sau:

P(a, x, f(a,g(x,y)))
P(u,h(a), f(u,v)) (1)

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

P(a, x, f(a, g(x, y)))
P(a, h(a), f(a, v)) (2)

và phép thế

θ_1 = [u|a].

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

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

và phép thế: θ_2 = [u|a, x|h(a)]

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

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

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

θ = [u|a, x|h(a), v|g(h(a), y)]

6.3.5. Chứng minh bác bỏ bằng luật phân giải

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

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

Ví dụ 35:

Giả sử CSTT gồm các câu tuyển sau:

$\neg P(w) \vee Q(w)$ (1)

$P(x) \vee R(x)$ (2)

$\neg Q(y) \vee S(y)$ (3)

$\neg R(z) \vee S(z)$ (4)

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

$P(a) \vee S(a)$ (5)

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

$\neg P(a) \vee S(a)$ (6)

Áp dụng luật phân giải cho câu (5) và (6), ta suy ra câu $S(a) \vee S(a)$. Câu này tương đương với câu S(a). Chứng minh bằng cách áp dụng các luật suy diễn để dẫn tới điều cần phải chứng minh (như chứng minh trên) được gọi là chứng minh diễn dịch (deduction proof). Nhưng cần biết rằng, luật phân giải không phải là luật đầy đủ cho diễn dịch (deduction_complete), tức là từ một tập các tiên đề, chỉ sử dụng luật phân giải chúng ta không thể sinh ra tất cả các câu là hệ quả logic của các tiên đề đã cho.

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

Để chứng minh câu H là hệ quả logic của tập các câu {G1, G2,…, Gn}(các tiên đề), ta có thể áp dụng phương pháp chứng minh bác bỏ, tức là chứng minh tập câu {G1,G2,…, Gn, ⌉H} không thoả được. Mặt khác, ở trên ta đã chỉ ra rằng, luật phân giải cho phép ta xác định được một tập câu là thoả được hay không thoả được. Vì vậy luật phân giải được xem là luật đầy đủ cho bác bỏ (refutation -complete).

Ví dụ 36:

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

⌉S(a) (7)

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

⌉Q(a) (8)

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

⌉P(a) (9)

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

⌉R(a) (10)

Từ câu (2) và (10) với phép thế [x|a] ta nhận được câu:P(a)(11)

Áp dụng luật phân giải cho câu (9) và (11) ta nhận được câu rỗng (mâu thuẫn: ⌉P(a) và P(a)). Bây giờ chúng ta trình bày thủ tục tổng quát sử dụng luật phân giải để chứng minh một công thức H là hoặc không là hệ quả logic của một tập công thức G={G1,G2,…, Gn}: thủ tục chứng minh bằng luật phân giải.

Procedure Proof_by_Resolution

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

H_Công thức cần chứng minh;

Begin

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

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

3. Repeat

3.1. Chọn ra hai câu A và B trong C;

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

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

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

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

else thông báo H sai;

End

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

1. Để thực hiện bước 1, ta cần áp dụng thủ tục chuẩn hoá các câu đã được đưa ra trong mục 6.3.4.

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

3. Bước 3.1 là bước chưa được xác định rõ ràng. Có rất nhiều phương pháp chọn ra hai câu A và B từ tập câu C. Mục sau chúng ta sẽ trình bày các chiến lược, nó cho phép ta lấy ra hai câu ở mỗi bước 3 sao cho vòng lặp ở bước 3 được thực hiên hiệu quả. Các chiến lược này được gọi là chiến lược phân giải.

Trong nhiều ứng dụng, CSTT (tập câu G) chỉ gồm các câu Horn (các luật If_then). Trong các trường hợp đó, chúng ta sẽ có các thủ tục suy diễn hiệu quả hơn. Một ví dụ chứng minh:

Ví dụ 37:

Giả sử chúng ta biết các thông tin sau đây:

1) Ông Ba nuôi một con chó.

2) Hoặc ông Ba hoặc ông Am đã giết con mèo Bibi.

Chúng ta cũng biết rằng:

3) Mọi người nuôi chó đều yêu quý động vật.

4) Ai yêu quý động vật cũng không giết động vật.

Và đương nhiên là:

5) Chó mèo đều là động vật.

Để biểu diễn các tri thức trên trong logic vị từ cấp một, chúng ta cần sử dụng các hằng D, Ba, Am, Bibi, các vị từ Dog(x) (x là chó), Cat(y), (y là mèo), Rear(u,v) (u nuôi v), AnimalLover(u) (u là người yêu quý động vật), Kill(u,v) (u giết v), Animal(x) (x là

động vật). Sử dụng các hằng và các vị từ trên, chúng ta có thể chuyển các câu 1_5 thành các câu trong logic vị từ cấp một như sau:

1. Dog(D) ∧ Rear(Ba,D)

2. Cat(Bibi) ∧ (Kill(Ba,Bibi) ∨ Kill(Am,Bibi))

3. ∀x (∀y(Dog(y) ∧ Rear(x,y)) ⇒ AnimalLover(x)))

4. ∀u (AnimalLover(u) ⇒ (∀v (Animal(v) ⇒ ¬ Kill(u,v)))

5. ∀z (Dog(z) ⇒ Animal(z)) ∧ ∀w(Cat(w) ⇒ Animal(w)).

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

3'. ¬ Dog(y) ∨ ¬ Rear(x,y) ∨ AnimalLover(x)

4'. ¬ AnimalLover(u) ∨ ¬ Animal(v) ∨ ¬ Kill(u,v)

5'.(¬ Dog(z) ∨ Animal(z)) ∧ (¬ Cat(w) ∨ Animal(w))

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

(1) Dog(D)

(2) Rear(Ba, D)

(3) Cat(Bibi)

(4) Kill(Ba,Bibi) ∨ Kill(Am, Bibi)

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

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

(7) ¬ Dog(z) ∨ Animal(z)

(8) ¬ Cat(w) ∨ Animal(w)

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

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

Thêm vào các câu (1)_(8) câu:

(9) ¬ Kill (t,Ba)

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

(10) Kill (Ba,Bibi)

Từ (6) và (10) với phép thế [u|Ba,v|Bibi],

ta nhận được câu:

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

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

(12) Animal (Bibi)

Từ (11) và (12) ta nhận được câu:

(13) ¬ AnimalLover (Ba)

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

(14) ⌉Rear (x,D) ∨ AnimalLover(x)

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

(15) AnimalLover(Ba)

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

6.3.6. Các chiến lược phân giải

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

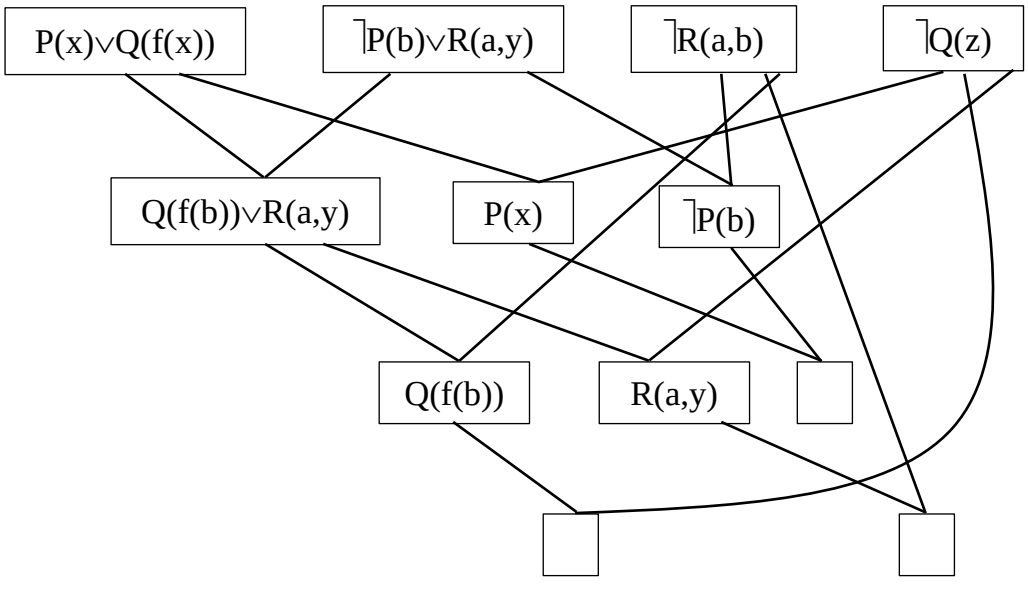
Chúng ta có thể xem thủ tục chứng minh bằng luật phân giải như thủ tục tìm kiếm trên ***đồ thị phân giải***. Đỉnh của đồ thị này là các câu, các cung đi từ các câu cha tới câu là phân giải thức của các câu cha.

Ví dụ 38:

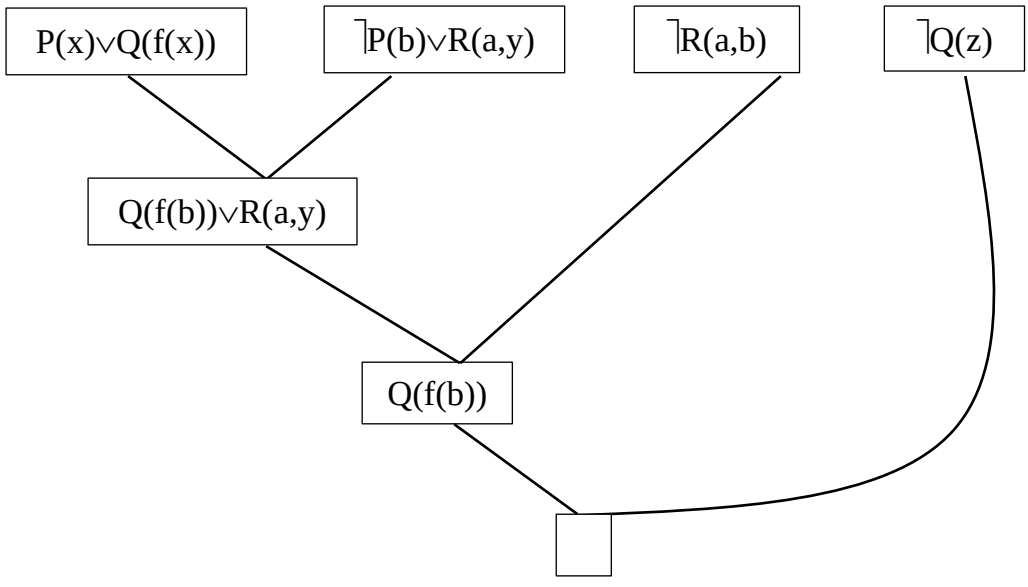
Giả sử chúng ta có một tập hợp các câu sau:

P(x)∨Q(f(x)); ⌉P(b) ∨R(a,y); ⌉R(a,b); ⌉Q(z)

Đồ thị phân giải trên tập câu đã cho được biểu diễn trong hình 6.1 dưới đây.



Hình 6.1. Đồ thị phân giải



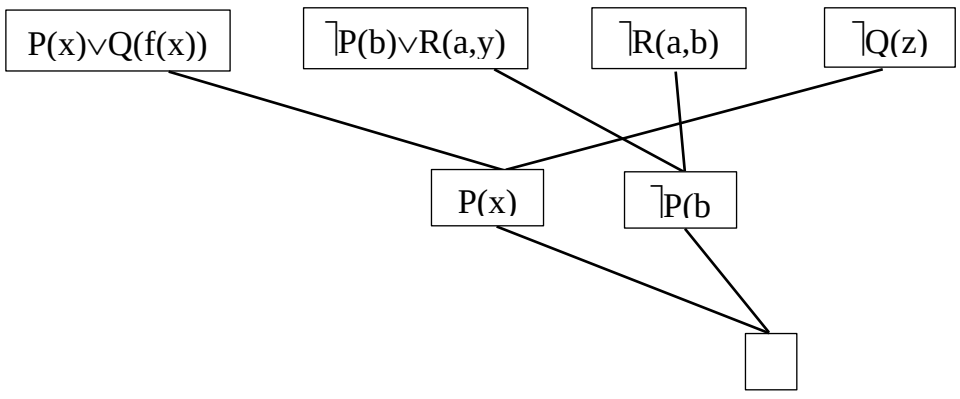
Hình 6.2. Một cây chứng minh từ đồ thị phân giải

Từ một tập câu tuyển đã cho, nếu ta tìm được một dãy các giải thức dẫn tới câu rỗng (□) thì ta đã chứng minh được tập câu tuyển đã cho là không thỏa được. Khi đó, dãy các giải thức kết thúc bởi câu rỗng tạo thành một cây nhị phân, gốc là câu rỗng. Cây này được gọi là ***cây chứng minh***. Hình 6.2 biểu diễn một cây chứng minh của tập câu trong ví dụ trên không thỏa được. Tuy nhiên, trong đồ thị phân giải có thể có nhiều cây chứng minh và đồ thị phân giải hình 6.1 có 5 cây chứng minh, hình 6.2 là một trong các cây chứng minh đó.

Mục đích của các chiến lược phân giải hoặc chiến lược sinh ra các giải thức theo một quy tắc nhất định nào đó luôn hướng tới mục tiêu sinh ra câu rỗng. Một chiến lược được xem là đầy đủ nếu nó đảm bảo sinh ra câu rỗng nếu một tập các câu tuyển cho trước là không thỏa được. Sau đâu chúng ta sẽ tìm hiểu 3 chiến lược phân giải: Chiến lược phân giải theo bề rộng; Chiến lược phân giải sử dụng tập hỗ trợ; Chiến lược phân giải tuyến tính.

a) Chiến lược phân giải theo bề rộng

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



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

b) Chiến lược phân giải sử dụng tập hỗ trợ

Thủ tục chứng minh sẽ hiệu quả hơn nhiều nếu chúng ta hạn chế sinh ra các giải thức mà vẫn không ảnh hưởng gì đến việc sinh ra câu rỗng. Chiến lược phân giải sử dụng tập hỗ trợ nhằm mục đích đó. Đầu tiên từ tập các câu tuyển đã cho, ta chọn ra một tập con các câu S- tập này được gọi là tập hỗ trợ. Chiến lược phân giải sử dụng tập hỗ trợ S đặt ra hạn chế sau: một phân giải thức chỉ được sinh ra nếu một câu cha của nó thuộc S hoặc là hậu thế của một câu thuộc S. Do đó nếu tập S là nhỏ so với tập ban đầu thì không gian tìm kiếm sẽ được thu hẹp hơn. Trật tự các câu được sinh ra có thể điều khiển bởi chiến lược bề rộng.

Các cách chọn tập hỗ trợ:

+ S gồm tất cả các câu thuộc tập câu tuyển ban đầu (ξ) không chứa literal âm. Khi đó ξ -S sẽ gồm các câu chứa literal âm, và do đó ξ -S là thoả được, vì một minh hoạ mà tất cả các literal âm trong ξ-S nhận giá trị true là một mô hình của ξ-S.

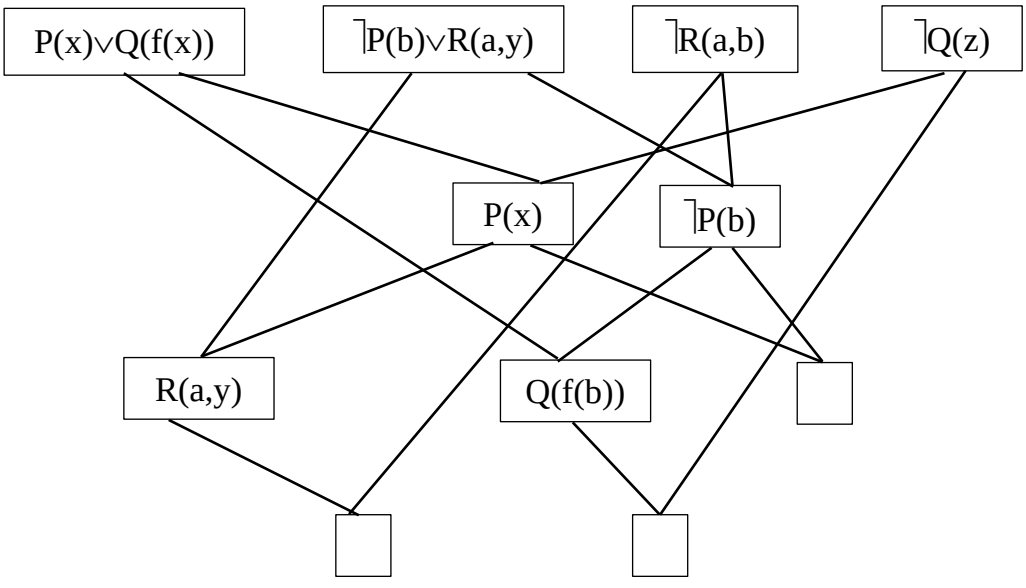
+ S gồm tất cả các câu thuộc ξ không chứa các literal dương. Tương tự như trên, tập các ξ-S là thoả được.

+ Chọn S là phủ định của kết luận cần chứng minh

Ví dụ 39:

Cho ξ={P(x)∨Q(f(x)); ¬P(b) ∨R(a,y); ¬R(a,b); ¬Q(z)}

Nếu chọn tập hỗ trợ S={¬R(a,b); ¬Q(z)}. Ta có đồ thị phân giải theo chiến lược sử dụng tập hỗ trợ:



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

c) Chiến lược tuyến tính

Giả sử ξ là tập câu mà ta cần chứng minh là không thoả được. Chọn một câu ξ_0 thuộc ξ, ξ_0 được gọi là câu trung tâm xuất phát. Các giải thức là hậu thế của ξ_0 cũng được gọi là câu trung tâm. Chiến lược tuyến tính đặt ra hạn chế sau: chỉ sinh ra các giải thức mà một trong hai câu cha là câu trung tâm, đồng thời hai câu trung tâm chỉ được phân giải cùng nhau khi một câu là hậu thế của một câu khác. Tính chất "tuyến tính" thể hiện ở chỗ, hai câu trung tâm không được phép phân giải cùng nhau trừ khi một câu là hậu thế của câu kia.

Chiến lược phân giải tuyến tính sẽ đầy đủ, nếu câu trung tâm xuất phát ξ_0 được chọn từ tập ξ sao cho các câu còn lại thoả được. Trong thực tế, nếu cần chứng minh H là hệ quả logic của tập các câu tuyển ξ, thì người ta lấy $\xi_0 = \neg H$.

Ví dụ 40:

Cho ξ={P(x)∨Q(f(x)); ¬P(b) ∨R(a,y); ¬Q(z)}

Giả sử rằng cần chứng minh R(a,b) là hệ quả logic của tập công thức đã cho.

Để chứng minh R(a,b) ta sẽ thêm ¬R(a,b) vào tập công thức đã cho và chọn ξ_0=¬R(a,b) là câu trung tâm. Khi đó ta có đồ thị phân giải theo chiến lược tuyến tính.

Câu hỏi

Câu 1. Hai biểu thức hợp nhất được khi nào ?

Câu 2. Tư tưởng của thuật toán hợp nhất. Cho ví dụ minh họa.

Câu 3. Tư tưởng chứng minh bác bỏ bằng luật phân giải trong logic vị từ cấp 1

Câu 4. Thủ tục chứng minh bác bỏ bằng luật phân giải trong logic vị từ cấp 1. Cho ví dụ.

Câu 5. Trình bày các chiến lực phân giải. Cho ví dụ

Bài tập

Vẽ cây chứng minh

a. Vẽ cây chứng minh theo tập hướng dẫn T={R(x), P(a)}

{P(x) ∨Q(x,y), ⌝Q(a,b)∨R(a), R(x), P(a)}

b. Vẽ cây chứng minh theo bề rộng

{⌝P(x) ∨Q(x,y), ⌝Q(a,b) ∨⌝R(a), R(x), P(a)}

Bài tập cuối chương 6

Bài 1. Đưa câu sau về dạng chuẩn hội:

a) ((A ⇒ B) ∨ C) ⇒ (⌝D∧ E)

b) (A ∧ B) ⇒ (⌝C∨ ⌝D∧E)

c) (A ∨ B) ⇒ ((C ⇒ D) ⇒ E)

d) (A ∨ B) ⇒ ((C ∨ D) ⇒ E)

e) (A ⇒ B) ⇒ ((C ∨ D) ⇒ E)

f) (A⇒B)∧((C⇒D) ⇒E)

g) (A⇒B)∨((C⇒D)∨E)

Bài 2. Đưa các câu sau về dạng chuẩn tắc hội.

1/ ∃x(⌝∃yP(x,y)∧∀yQ(x,y)) ⇒∀xR(x)

2/ ∀x∀y(∃zP(x,y,z)∧Q(x,y))∨∀xR(x)

3/ ∀x∀y((∃zP(x,y,z)∧Q(x,y))∧∀xR(x)

4/ (∀xP(x) ⇒∃x∀yQ(x,y)) ⇒∀yR(y)

5/ (∃xP(x)∨∃xQ(x)) ⇒ (∀x∃yR(x,y)∧∀xA(x))

6/ (∀xP(x)∨∀x∃yQ(x,y))∨(∃xR(x) ⇒∀xA(x))

7/ (∀xPx⇒∀xQx) ⇒(∀x∃yR(x,y)Λ∀xAx)

8/ ((∀x∃yP(x,y) ∨ ∀x∀yQ(x,y)) ⇒ ∀xRx) ⇒ ∀x∃yA(x,y)

9/ (∃xP(x)∨ ∀xQ(x)) ⇒ (∀x∃yR(x,y)∧∀xA(x))

10/ ((∀x∃yP(x,y) ∨ ∀x∀yQ(x,y)) ∧ ∀xRx) ⇒ ∀x∃yA(x,y)

Bài 3: Chứng minh (sử dụng các cách chứng minh)

a) Cho tập công thức sau

A∨B

B→(C∨D)

C→(E∧F)

⌝E

⌝D

b) Cho tập công thức sau

A→(B∨C)

B→D

⌝C

⌝D

Hãy CM: A là hệ quả logic

Hãy CM: ¬A là hệ quả logic

c) Cho tập công thức sau

$Q \wedge S \Rightarrow G \vee H$

$P \Rightarrow Q$

$R \Rightarrow S$

P

R

Hãy CM: G∨H là hệ quả logic

d) Cho tập công thức sau

$(B \Rightarrow C) \Rightarrow (G \wedge F)$

$B \Rightarrow E$

$F \Rightarrow I$

$F \wedge \neg E$

Hãy CM: G∧I là hệ quả logic

Bài 4: Vẽ cây chứng minh theo tập hướng dẫn

a. T={¬R(x), ¬P(a)}; {P(x)vQ(x,y), ¬Q(a,b)vR(a), ¬R(x), ¬P(a)}

b. T={¬R(z,x), ¬P(c,a,b)}; {P(x,y,z)vQ(y,z), ¬Q(a,b)vR(b,c), ¬R(z,x), ¬P(c,a,b)}

Bài 6: Vẽ cây chứng minh theo bề rộng

a. {P(x)vQ(x,y), ¬Q(a,b)vR(a), ¬R(x), ¬P(a)}

b. {P(x,y,z)vQ(y,z), ¬Q(a,b)vR(b,c), ¬R(z,x), ¬P(c,a,b)}

Bài 7: Chứng minh (sử dụng các phương pháp chứng minh)

a) Cho tập công thức: P(x,y,z)∨Q(y,z) ¬Q(a,b) ∨R(b,c) ¬R(z,x)	b) Cho tập công thức: P(x)∨Q(x,y)vR(y) ¬P(a) ∨H(a,b) ¬H(x,y) ¬R(b)	c) Cho tập công thức: P(x)∨Q(x,y) ¬R(a) ¬Q(f(a),b)∨R(a)
Hãy chứng minh P(c,a,b) là hệ quả logic của tập công thức trên	Hãy chứng minh Q(a,b) là hệ quả logic của tập công thức trên	Hãy chứng minh P(b) là hệ quả logic của tập công thức trên

d) Cho tập công thức: P(x)∨Q(f(x),y) ¬P(a) ¬Q(f(a),b)∨R(y)	e) Cho tập công thức: B(c)∨A(c,b,f(c)) B(x)∨Q(x,y) ¬A(x,y,f(x)) ∨ ¬R(y) R(b)	f) Cho tập công thức: ¬P(x,y)∨ ¬Q(x) P(a,b) ∨H(a,b) ¬H(x,y) ∨R(y) ¬R(b)
Hãy chứng minh R(b) là hệ quả logic của tập công thức trên	Hãy chứng minh Q(c,b) là hệ quả logic của tập công thức trên	Hãy chứng minh ¬Q(a) là hệ quả logic của tập công thức trên

Bài 8.

Hãy biểu diễn các câu sau bởi logic vị từ cấp 1

Tuấn là một sinh viên của ICTU

Mọi sinh viên của ICTU đều học môn Đại số

Vì Tuấn là một sinh viên của ICTU, nên Tuấn học môn Đại số.

Bài 9.

Cho cơ sở tri thức

1. Hùng thích tất cả các loại thực phẩm
2. Táo là thực phẩm
3. Gà là thực phẩm
4. Bất cứ thứ gì mọi người ăn và không bị hại đó là thực phẩm
5. Phong ăn đậu phộng và đậu tương
6. Lan ăn bất cứ thứ gì Phong ăn

Hãy chứng minh Hùng thích đậu phộng

Bài 10.

Cho phát biểu sau:

Nam đẹp trai, giàu có.

Do vậy, Nam hoặc là phung phí hoặc là nhân từ và giúp người.

Thực tế, Nam không phung phí hoặc cũng không kêu căng.

Do vậy, có thể nói Nam là người nhân từ.

Kiểm chứng kết quả suy luận trên, bằng luật phân giải.

Câu hỏi thường gặp chương 6

1. Ngôn ngữ biểu diễn tri thức?

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

2. Ngữ nghĩa của ngôn ngữ biểu diễn tri thức?

Câu trả lời: Ngữ nghĩa của ngôn ngữ cho phép ta xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực.

3. Cú pháp của một ngôn ngữ biểu diễn tri thức?

Câu trả lời: Cú pháp của một ngôn ngữ bao gồm các ký hiệu về các quy tắc liên kết các ký hiệu (các luật cú pháp) để tạo thành các câu (công thức) trong ngôn ngữ. Các câu ở đây là biểu diễn ngoài, cần phân biệt với biểu diễn bên trong máy tính. Các câu sẽ được chuyển thành các cấu trúc dữ liệu thích hợp được cài đặt trong một vùng nhớ nào đó của máy tính, đó là biểu diễn bên trong.

4. Sự tương đương của các công thức

Câu trả lời: Hai công thức A và B được xem là *tương đương* nếu chúng có cùng một giá trị chân lý trong mọi minh họa. Để chỉ A tương đương với B ta viết $A \equiv B$.

5. Dạng chuẩn tắc hội?

Câu trả lời: Một công thức ở dạng chuẩn hội nếu nó là **hội của các câu tuyển** có dạng $A_1 \vee ... \vee A_m$ trong đó các A_i là *literal.*

6. Các bước chuẩn hóa một công thức trong logic mệnh đề

Câu trả lời: Có 3 bước

1. Bỏ các dấu kéo theo ($\Rightarrow$) bằng cách thay ($A \Rightarrow B$) bởi ($\neg A \vee B$).
2. Chuyển các dấu phủ định ($\neg$) vào sát các kết hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay $\neg(\neg A)$ bởi A.
3. Áp dụng luật phân phối, thay các công thức có dạng

   $A \vee (B \wedge C)$ bởi $(A \vee B) \wedge (A \vee C)$.

7. Câu Horn?

Câu trả lời: câu Horn có dạng :

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

Trong đó P_i, Q là các literal dương. Các P_i được gọi là các điều kiện (hoặc giả thiết), còn Q được gọi là kết luận (hoặc hệ quả).

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

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

8. Luật suy diễn?

Câu trả lời: Một công thức H được xem là *hệ qủa logic* (logical consequence) của một tập công thức $G = \{G_1,...,G_m\}$ nếu trong bất kỳ minh họa nào mà $\{G_1,...,G_m\}$ đúng

thì H cũng đúng, hay nói cách khác bất kỳ một mô hình nào của *G* cũng là mô hình của H.

9. Luật suy diễn tin cậy?

Câu trả lời: Một luật suy diễn được xem là *tin cậy* (secured) nếu bất kỳ một mô hình nào của giả thiết của luật cũng là mô hình kết luận của luật.

10. Tiên đề, định lý, chứng minh?

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

11. Chứng minh bác bỏ?

Câu trả lời: Phương pháp chứng minh bác bỏ (refutation proof hoặc proof by contradiction) là một phương pháp thường xuyên được sử dụng trong các chứng minh toán học. Tư tưởng của phương pháp này là như sau: Để chứng minh P đúng, ta giả sử P sai (thêm ¬P vào các giả thiết) và dẫn tới một mâu thuẫn.

12. Định lý giải?

Câu trả lời: Một tập câu tuyển là không thỏa được nếu và chỉ nếu câu rỗng □ ∈ R(G).

13. Logic mệnh đề ?

Câu trả lời : Logic mệnh đề cho phép ta biểu diễn các sự kiện, mỗi kí hiệu trong logic mệnh đề được minh họa như là một sự kiện trong thế giới hiện thực, sử dụng các kết nối logic ta có thể tạo ra các câu phức hợp biểu diễn các sự kiện mang ý nghĩa phức tạp hơn.

14. Logic vị từ cấp một?

Câu trả lời: Logic vị từ cấp một là mở rộng của logic mệnh đề. Nó cho phép ta mô tả thế giới với các đối tượng, các thuộc tính của đối tượng và các mối quan hệ giữa các đối tượng. Nó sử dụng các biến (biến đối tượng) để chỉ một đối tượng trong một miền đối tượng nào đó. Để mô tả các thuộc tính của đối tượng, các quan hệ giữa các đối tượng, trong logic vị từ, người ta dựa vào các *vị từ* (predicate). Ngoài các kết nối logic như trong logic mệnh đề, logic vị từ cấp một còn sử dụng các *lượng tử*.

15. Lượng tử phổ dụng (∀)?

Câu trả lời: Lượng tử phổ dụng (∀) cho phép mô tả tính chất của cả một lớp các đối tượng, chứ không phải của một đối tượng, mà không cần phải liệt kê ra tất cả các đối tượng trong lớp. Chẳng hạn sử dụng vị từ Elephant(x) (đối tượng x là con voi) và vị từ

Color(x, Gray) (đối tượng x có mầu xám) thì câu " tất cả các con voi đều có mầu xám" có thể biểu diễn bởi công thức $\forall x$ (Elephant(x) $\Rightarrow$ Color(x, Gray)).

16. Lượng tử tồn tại ($\exists$) ?

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

17. Có mấy bước chuẩn hóa một công thức trong logic vị từ cấp một.

Câu trả lời : 8 bước

Chương 7. BIỂU DIỄN TRI THỨC BỞI CÁC LUẬT VÀ LẬP LUẬN

Nội dung chính của chương

Trong chương 6, chúng ta đã biết rằng, với một cơ sở tri thức gồm các câu trong logic vị từ cấp một, ta có thể chứng minh được một công thức có là hệ quả logic của cơ sở tri thức hay không, bằng phương pháp chứng minh bác bỏ và thủ tục giải. Tuy nhiên, thủ tục chứng minh tổng quát này có độ phức tạp lớn và đòi hỏi chiến lược giải thích hợp. Chính vì lý do này mà các nhà nghiên cứu cố gắng tìm các tập con của logic vị từ cấp một, sao cho chúng đủ khả năng biểu diễn cơ sở tri thức trong nhiều lĩnh vực áp dụng, và có thể đưa ra các thủ tục suy diễn hiệu quả. Các tập con này của logic vị từ cấp một sẽ xác định các ngôn ngữ biểu diễn tri thức đặc biệt. Trong chương này chúng ta sẽ nghiên cứu ngôn ngữ chỉ bao gồm các câu Horn (các luật *nếu - thì*). Chỉ sử dụng các luật *nếu - thì* chúng ta không thể biểu diễn được mọi điều mà chúng ta có thể biểu diễn được trong logic vị từ cấp một. Tuy nhiên với các luật *nếu - thì* ta có thể biểu diễn được một khối lượng lớn tri thức trong nhiều lĩnh vực áp dụng khác nhau, và có thể thực hiện các thủ tục suy diễn hiệu quả.

Nội dung của chương này bao gồm :

- Biểu diễn tri thức bởi các luật và hệ luật
- Lập luận tiến
- Lập luận lùi

Mục tiêu cần đạt được của chương

Sau khi học xong chương này học viên cần hiểu được biểu diễn tri thức bởi luật và hệ luật là gì ? Áp dụng để biểu diễn tri thức thành các luật. Sau đó áp dụng các thuật toán lập luận tiến và lập luận lùi để thực hiện quá trình suy diễn

BÀI 11 : BIỂU DIỄN TRI THỨC BỞI CÁC LUẬT VÀ LẬP LUẬN (Số tiết: 3 tiết)

7.1 Biểu diễn tri thức bởi luật và hệ luật

Hiện nay đa số các hệ chuyên gia sử dụng mô hình biểu diễn tri thức bởi các luật. Trong đó tri thức của các chuyên gia trong một lĩnh vực đối tượng nào đó (chẳng hạn trong y học, nông nghiệp, sửa chữa máy, ...) được biểu diễn bởi các luật ***if - then***. Mỗi luật gồm hai phần. Phần ***if*** gồm một số điều kiện (condition), phần ***then*** gồm một số hệ quả (cosnequent), hoặc gồm một số hành động (reaction):

if <conditions>

then <consequents>

hoặc

if <conditions>

then <reactions>

Một vài ví dụ về luật :

if 'cụ già đi bộ qua đường cao tốc"

then "cụ già có thể bị xe nghiến'

if 'x là số thực khác không'

and 'y là số thực khác không cùng dấu với x'

then "tích xy là số thực dương'

Chúng ta sẽ gọi sự kiện là một khẳng định về một điều gì đó. Chẳng hạn, 'nhiệt độ lúc 12h trưa hôm qua là $37^{O}C$ ' hoặc ' số pi là số vô tỉ ' là các sự kiện.

Một hệ chuyên gia sử dụng tri thức được biểu diễn dưới dạng các luật được gọi là *hệ luật*

Một hệ luật bao gồm ba thành phần chính sau đây :

1. *Cơ sở luật* (rule base) : Đó là tập hợp các luật được cung cấp bởi các chuyên gia.

2. *Bộ nhớ làm việc* (working memory) hoặc cơ sở sự kiện (fact base). Bộ nhớ làm việc lưu giữ các sự kiện liên quan đến một vấn đề cụ thể của người sử dụng hệ. Nó cùng lưu giữ các kết luận rút ra được trong quá trình suy diễn.

3. *Cơ chế suy diễn* : (inference engine hoặc deductive machine). Cơ chế suy diễn sử dụng các luật trong cơ sở luật và đối sánh với nội dung của bộ nhớ làm việc để rút ra các kết luận. Trong các mục sau, chúng ta sẽ nghiên cứu kỹ các phương pháp suy diễn trong các hệ sản xuất.

Trong một hệ sản xuất, nếu phần ***then*** của các luật xác định các khẳng định mới và sẽ được đặt vào bộ nhớ làm việc trong quá trình suy diễn, thì hệ được gọi là *hệ diễn dịch* (deduction system).

Có những hệ mà phần ***then*** không xác định các khẳng định mà là các hành động cần tiến hành khi các điều kiện trong phần ***if*** được thoả mãn. Các hệ như thế được gọi là các hệ hành động (reaction system).

7.2 Lập luận tiến

Trong các hệ sản xuất (các hệ dựa trên luật), ta hiểu suy diễn là quá trình móc nối các luật để rút ra các sự kiện mới từ các sự kiện đã biết.

Người ta phân biệt hai chiến lược suy diễn chính, tuỳ thuộc vào cách người ta lấy các sự kiện ở vế nào trong luật dùng làm các sự kiện xuất phát hoặc sự kiện được suy ra. Với ý nghĩa đó, người ta thường sử dụng hai chiến lược suy diễn (strategies for chaining) : suy diễn tiến (forward chaining) và suy diễn lùi (backward chaining).

Tư tưởng cơ bản của suy diễn tiến là như sau. Các sự kiện được xem là đúng trong một lĩnh vực áp dụng (các sự kiện được lưu trong bộ nhớ làm việc) được dùng làm các 'tiền đề' cho suy diễn. Trong mỗi bước, người ta xem xét một luật, đối sánh các điều kiện trong phần ***if*** của luật với nội dung của bộ nhớ làm việc. Nếu tất cả các điều kiện trong phần ***if*** của luật đều được thoả mãn (đều có trong bộ nhớ làm việc), thì luật được xem là có thể *cháy* được. Khi đó các kết luận trong phần ***then*** của luật được xem là 'hệ quả logic' của các sự kiện trong phần ***if*** và được xem là sự kiện được suy ra. Nếu các sự kiện được suy ra là mới (chưa có trong bộ nhớ làm việc) thì chúng được đưa vào bộ nhớ làm việc.

Quá trình trên được lặp lại cho tới khi không có luật nào sinh ra các sự kiện mới.

Để thấy rõ quá trình suy diễn tiến xẩy ra như thế nào, ta xét ví dụ sau:

Ví dụ 41:

Từ các tri thức của các nhà động vật học, ta có thể đưa ra một cơ sở luật (cơ sở luật về các con vật trong sở thú):

Luật 1 : ***if*** ' x có lông mao'

then 'x là động vật có vú '

Chú ý rằng, trong một luật có thể chứa một hoặc nhiều biến nó chỉ một đối tượng bất kỳ trong một lĩnh vực áp dụng nào đó. Chẳng hạn, trong luật 1 có một biến được ký hiệu là x, ở đây x chỉ một con vật bất kỳ trong sở thú.

Luật 2 : ***if*** 'x cho sữa'

then 'x là động vật có vú'

Luật 3 : ***if*** 'x có lông vũ'

then 'x là chim'

Luật 4 : ***if*** 1) ‘x biết bay’

2) ‘x đẻ trứng’

then ‘x là chim’

Luật 5 : ***if*** 1) ‘x là động vật có vú’

2) ‘x ăn thịt’

then ‘x là thú ăn thịt’

Luật 6 : ***if*** 1) ‘x là động vật có vú’

2) ‘x có răng nhọn’

3) ‘x có móng vuốt’

then ‘x là thú ăn thịt’

Luật 7 : ***if*** 1) ‘x là thú ăn thịt’

2) ‘x có màu vàng hung’

3) ‘x có đốm sẫm’

then ‘x là báo Châu Phi’

Luật 8 : ***if*** 1) ‘x là thú ăn thịt’

2) ‘x có màu vàng hung’

3) ‘ x có vằn đen’

then ‘x là hổ’

Luật 9 : ***if*** 1) ‘ x là chim’

2) ‘x không biết bay’

3) ‘x có chân dài’

4) ‘x có cổ dài’

then ‘x là đà điểu’

Luật 10 : ***if*** 1) ' x là chim '

2) ‘x jgiibg biết bay’

3) ‘x biết bơi’

then ‘x là chim cánh cụt’

Luật 11 : ***if*** 1) ‘x là chim’

2) ‘x bay giỏi’

then ‘x là hải âu’

Ngoài các luật trên, còn có nhiều các luật khác.

Bây giờ giả sử rằng, một em bé đứng trước chuồng nuôi một con thú (con thú có tên là Lôcô). Em quan sát thấy nó có các đặc điểm sau đây:

Lôcô có lông mao

Lôcô ăn thịt

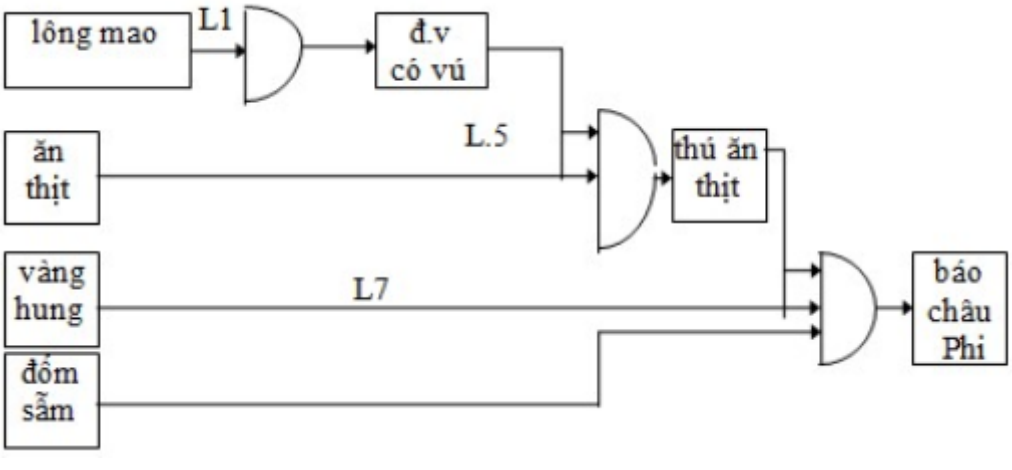
Lôcô có màu vàng hung

Lôcô có đốm sẫm

Các khẳng định này là nội dung của bộ nhớ làm việc. Ta thử xét xem hệ Sở - thú sẽ suy diễn như thế nào để giúp em bé xác định được Lôcô là con vật gì.

Vì 'Lôcô có lông mao', từ luật 1 suy ra rằng 'Lôcô là động vật có vú'. Vì 'Lôcô là động vật có vú' và 'Lôcô ăn thịt', từ luật 5 suy ra rằng 'Locô là thú ăn thịt'.

Tiếp theo ta thấy ba điều kiện trong phần if của luật 7 đều đúng. Do đó có thể khẳng định 'Lôcô là báo châu Phi'. Quá trình xâu chuỗi các luật để rút ra khẳng định 'Lôcô là báo châu Phi', có thể biểu diễn bởi sơ đồ trong hình sau.



Hình 7.1. Quá trình xâu chuỗi các luật

Trong quá trình suy diễn trên, ta đã sử dụng các khẳng định trong bộ nhớ làm việc như các tiên đề. Với mỗi luật ta đi từ phần ***if*** tới phần ***then.*** Khi các điều kiện trong phần ***if*** là đúng thì các hệ quả ở phần ***then*** được xem là các sự kiện mới được suy ra. Chính vì lẽ đó mà suy diễn tiến còn được gọi là lập luận định hướng dữ liệu

Các thủ tục thực hiện các chiến lược đó. Giả sử trong hệ sản xuất chứa một cơ sở luật là một tập nào đó các luật $\Re = \{R_1, R_2, ...\}$, một bộ nhớ làm việc WM. Với mỗi luật $R \in \Re$, ta sẽ ký hiệu cond (R) là tập các điều kiện trong phần ***if*** của luật R, còn conc (R) là tập các kết luận trong phần ***then*** của R.

Trong suy diễn tiến, thủ tục cơ bản là thủ tục FIRING (R), trong đó R là một luật nào đó (thủ tục đốt cháy luật R):

procedure FIRING (R);

1. Kiểm tra xem R có thể cháy được hay không bằng cách đối sánh các điều kiện trong phần ***if*** của luật R với các sự kiện trong bộ nhớ làm việc;
2. Nếu R cháy được, thì đưa các sự kiện mới (nếu có) trong phần then của R vào bộ nhớ làm việc, tức là WM ← WMU conc (R);

Thủ tục trên được áp dụng cho các luật $R \in \Re$ cho tới khi nào không có sự kiện nào mới được phát sinh.

Chúng ta cần chi tiết hoá thủ tục trên. Để dễ hiểu, ta xét ví dụ sau. Giả sử WM chứa các sự kiện.

Ví dụ 42:

Giôn là con ngựa

Ken là con ngựa

Giôn là mẹ của Bin

Giôn là mẹ của Ken

Ken là béo

Bin là mẹ của Kit

Kít là béo

Kít là con ngựa

Bin là con ngựa

và giả sử R là luật sau (luật mẹ)

if 1) 'x là con ngựa'

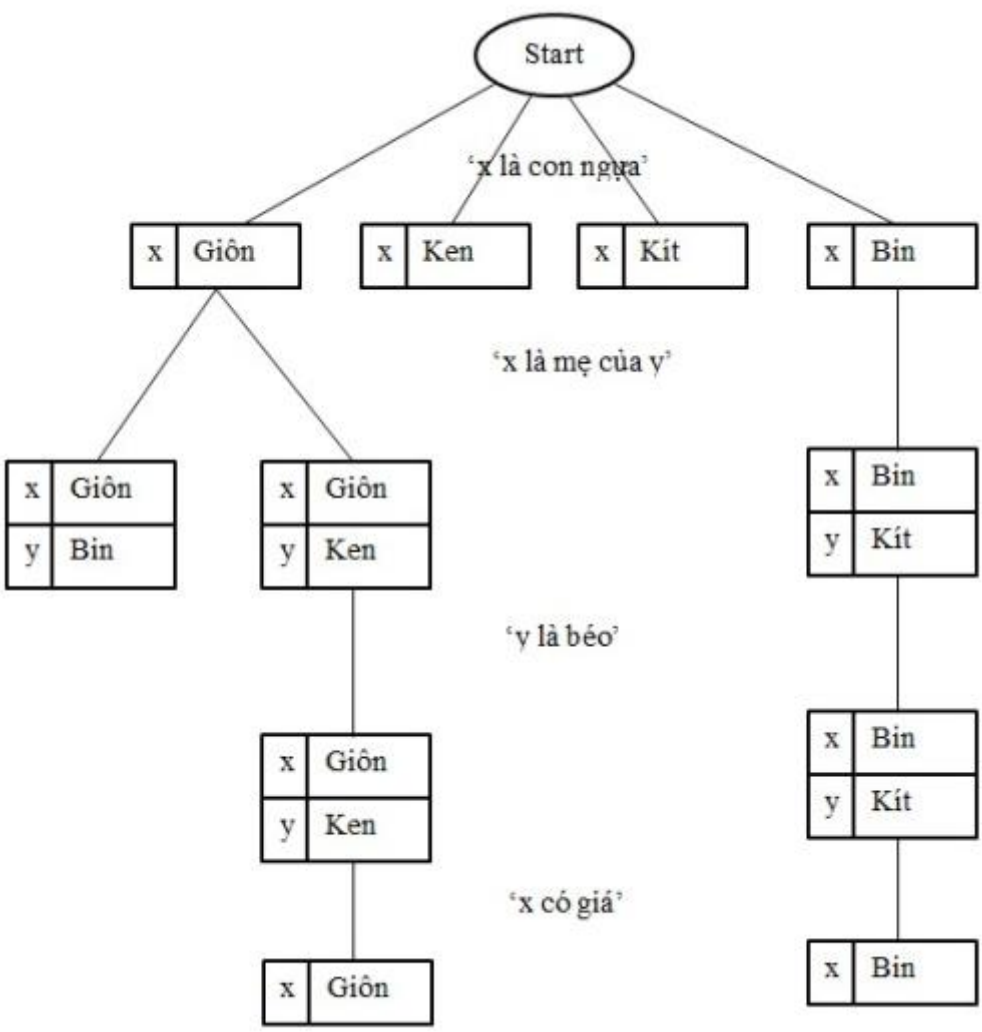
2) ' x là mẹ của y'

3) ' y là béo'

then ' x có giá'

Thông thường trong một luật có thể chứa các biến, chẳng hạn luật trên chứa hai biến là x và y. Ta gọi một ràng buộc biến (variable binding) là sự gắn cho mỗi biến một đối tượng cụ thể nào đó. Nói cách khác, một ràng buộc biến là một ánh xạ một phần tử

từ tập các biến vào tập các đối tượng. Bây giờ ta thử xét xem thủ tục FIRING cần làm những gì với luật trên (luật mẹ) để sinh ra các sự kiện mới. Kết hợp với luật đang xét, ta sẽ xây dựng một cây, gọi là cây suy diễn tiến Forward Tree (Forward chaining Tree). Gốc của cây là start. Xét điều kiện thứ nhất của luật ' x là con ngựa'. Đối chiếu với bộ nhớ làm việc, x có thể gắn với Gion, với Ken, với Kit, với Bin. Đỉnh start có 4 đỉnh con là (x → Gion), (x → Ken), (x → Kit) và (x → Bin). Xét tiếp điều kiện thứ hai của luật ' x là mẹ của y'. Đối sánh với WM ta thấy nếu x gắn với Gion thì y được gắn với Bin, hoặc với Ken. Do đó đỉnh (x → Giôn) có hai con là (x → Giôn, y → Bin) và (x → Giôn, y → Ken). Tương tự, đỉnh (x → Bin) có một con (x → Bin, y → Kít), các đỉnh (x → ken), (x → Kít) không có đỉnh con. Xét tiếp điều kiện thứ ba 'y là béo". Trong các đỉnh ở mức hai (x → Giôn, y → Bin), (x → Giôn, y → Ken) (x → Bin, y → Kít), vì chỉ có Ken và Kít là béo, nên đỉnh (x → Giôn, y → Bin) không có con, còn mỗi đỉnh (x → Giôn, y → Ken) và (x → Bin, Y → Kít) có một con. Ta nhận được cây trong hình sau.



Hình 7.2. cây suy diễn tiến kết hợp với một luật

Xét các đỉnh ở mức ba, ta thấy x được gắn với Giôn và với Bin. Do đó từ khẳng định " x có giá " trong phần then của luật mẹ, ta suy ra "Giôn có giá " và "Bin có giá'.

Nhận xét.

- Mỗi đỉnh của cây suy diễn (trừ đỉnh start) biểu diễn một ràng buộc biến nào đó. Các đỉnh ở mức 1 là các ràng buộc biến mà điều kiện thứ nhất của luật được thoả mãn. Các đỉnh ở mức thứ i biểu diễn các ràng buộc biến mà các điều kiện từ thứ nhất đến thứ i của luật đều được thoả mãn.
- Độ cao của cây suy diễn kết hợp với một luật bằng số điều kiện có trong phần if của luật đó.

Ta đưa ra các ký hiệu sau. Giả sử u là một đỉnh trong cây suy diễn kết hợp với luật R nào đó. Ta gọi B(u) là ràng buộc biến ở đỉnh u. Thủ tục sau đây xây dựng cây suy diễn để rút ra các kết luận. Cây được xây dựng theo bề rộng.

procedure Forward Tree (R: rule) ;

1. Các đỉnh con của gốc là các đỉnh biểu diễn các ràng buộc biến mà điều kiện thứ nhất của luật được thoả mãn.
2. Xây dựng các đỉnh ở mức thứ i. Giả sử u là một đỉnh ở mức thứ i-1. Các con của đỉnh u được xác định như sau. Thay các biến trong điều kiện thứ i của luật R bởi các đối tượng gắn với nó trong ràng buộc biến B(u) và đối sánh với các sự kiện trong WM, ta nhận được các ràng buộc biến mà điều kiện thứ i được thoả mãn. Mỗi ràng buộc biến này được biểu diễn bởi một đỉnh con của đỉnh u.
3. Lặp lại bước 2 cho tất cả các điều kiện của luật R.
4. Xét mỗi lá của cây. Thay mỗi biến có trong các kết luận ở phần then của luật bởi các đối tượng gắn với nó trong các ràng buộc biến ở mỗi lá ta nhận được các sự kiện.
5. Nếu các sự kiện nhận được ở bước 4 là mới thì đặt nó vào WM.

7.3 Lập luận lùi

Trong các hệ sản xuất, người ta còn sử dụng chiến lược suy diễn lùi.

Trong suy diễn lùi, ngoài các sự kiện trong bộ nhớ làm việc, người ta đưa vào các giả thiết (hypothesis), đó là các khẳng định cần được đánh giá. Qua quá trình suy diễn, khẳng định có thể được xem là đúng (hoặc được ủng hộ bởi các sự kiện trong bộ nhớ làm việc) hoặc có thể được xem là bị bác bỏ (bởi các sự kiện trong bộ nhớ làm việc).

Quá trình suy diễn như sau. Với mỗi giả thiết, ta đối sánh với phần ***then*** của các luật. Nếu giả thiết khớp với phần ***then*** của một luật thì ta đi ngược lại phần ***if*** của luật này. Xét các điều kiện trong phần ***if***. Nếu một điều kiện nào đó không có trong bộ nhớ làm việc thì nó được xem là một giả thiết mới xuất hiện. Giả thiết mới này lại được đối sánh với phần ***then*** của các luật. Quá trình trên cứ tiếp tục cho tới khi ta không thể phát triển thêm được nữa.

Nếu tất cả các điều kiện được sinh ra trong quá trình phát triển các giả thiết bởi lựa chọn một luật thích hợp đều được thoả mãn (đều có trong bộ nhớ làm việc), thì giả thiết được xem là đúng. Ngược lại, nếu dù áp dụng luật nào để phát triển các giả thiết cũng dẫn đến các điều kiện không có trong bộ nhớ làm việc (các điều kiện này được xem là sai) thì giả thiết được xem là bị bác bỏ bởi các sự kiện đã biết.

Để sáng tỏ tư tưởng của suy diễn lùi ta xét ví dụ sau. Cơ sở luật là cơ sở luật Sở - thú gồm 11 luật đã trình bày. Giả sử trong bộ nhớ làm việc được đưa vào các sự kiện sau.

Bibi có lông vũ

Bibi có chân dài

Bibi có cổ dài

Bibi không biết bay

Giả thiết sau đây được đưa ra :

Bibi là đà điểu

Đối sánh giả thiết này với phần ***then*** của các luật, ta thấy nó trùng với phần then của luật 9 khi x được thay bởi Bibi. Từ phần ***if*** của luật 9, ta suy ra rằng, giả thiết 'Bibi đà điểu' đòi hỏi các điều kiện :

1) Bibi là chim

2) Bibi không biết bay

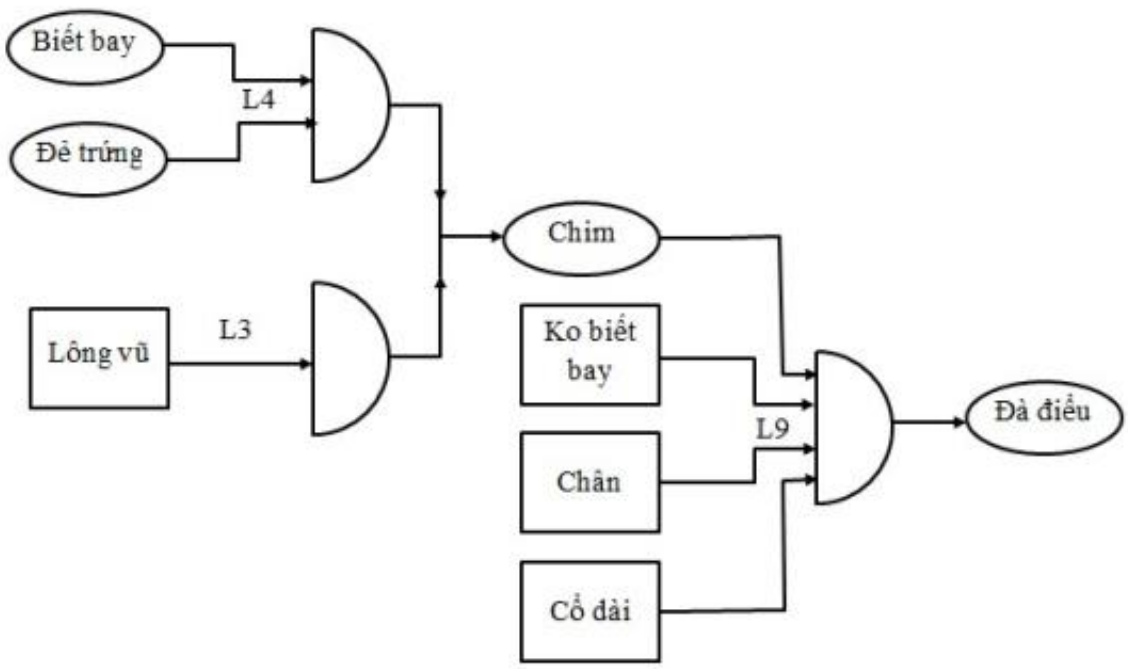
3) Bibi có chân dài

4) Bibi có cổ dài.

Ba điều kiện 2, 3 và 4 đều có trong bộ nhớ làm việc, trừ điều kiện 1) Bibi là chim. 'Bibi là chim' được xem là giả thiết mới. Lại đối sách giả thiết mới này với phần ***then*** của các luật. Ta thấy nó trùng với phần ***then*** của luật 3 và luật 4. Xét luật 4, đi ngược lên phần ***if*** của luật này, ta có các điều kiện sinh ra là 'Bibi biết bay' và 'Bibi đẻ trứng'. Cả hai điều kiện này đều không có trong bộ nhớ làm việc. Vì vậy chúng được xem như các giả thiết mới. Nhưng các giả thiết này đều không khớp với phần ***then*** của một luật

nào cả. Do đó, ta không thể phát triển tiếp các giả thiết mới này được nữa. Chuyển sang xét luật 3, để 'Bibi là chim' luật này đòi hỏi điều kiện "Bibi có lông vũ'. Điều kiện này có trong bộ nhớ làm việc. Vậy giả thiết 'Bibi là đà điểu' là đúng.

Quá trình suy diễn trên có thể mô tả bởi sơ đồ trong hình sau. Trong sơ đồ này, các khẳng định ghi trong các hình còn là các giả thiết. Nhìn sơ đồ này ta thấy, khi ta phát triển giả thiết 'Bibi là đà điểu' bởi luật 9, dẫn tới phát triển giả thiết 'Bibi là chim'. Nếu phát triển giả thiết này bởi luật 3, sẽ phát sinh điều kiện 'Bibi có lông vũ'. Tất cả các điều kiện phát sinh, khi ta phát triển giả thiết 'Bibi là đà điểu bởi luật 9, giả thiết 'Bibi là chim' bởi luật 3, đều là các sự kiện có trong bộ nhớ làm việc. Do đó giả thiết 'Bibi là đà điểu' là đúng.



Hình 7.3. Quá trình suy diễn

Trong quá trình suy diễn như trên, với mỗi luật ta luôn luôn đi từ phần ***then*** lùi tới phần ***if*** để sinh ra các điều kiện hoặc là giả thiết mới, hoặc là sự kiện đã có. Vì vậy mới có thuật ngữ suy diễn lùi (backward chaining, từ 'chaining' chỉ sự móc nối, liên kết các luật). Suy diễn lùi còn được gọi là lập luận định hướng mục đích (goal oriented reasoning).

Trong suy diễn lùi, một giả thiết H được đưa ra và ta cần kiểm tra xem giả thiết này là đúng hay sai. Giả sử giả thiết không chứa biến, chẳng hạn giả thiết 'Ken có giá', và các luật cũng không chứa biến, khi đó quá trình suy diễn có thể mô tả bởi hàm Eval (H) (hàm đánh giá giả thiết H). Hàm này nhận một trong hai giá trị {true, false}.

Gọi $\Re$(H) là tập các luật có phần kết luận trùng với giả thiết H.

function Eval (H : Hypothesis)

1. **if** H ∈ WM **then** Eval ← true **else** if ℜ (H) = ϕ **then** Eval ← false **else** đi tới bước 2;
2. OK ← false ;
3. **While** (ℜ(H) ≠ ϕ) and (not OK) **do**

   3.1. Lấy R ∈ℜ (H) và loại R khỏi ℜ(H) ;

   3.2. flag ← true ;

   3.3. **for** X ∈cond (R) **do**

   if Eval (X) = false **then** flag ← false

   3.4. **if** flag **then** OK ← true ;
4. Eval ← OK ;

Nhận xét: Quá trình thực hiện hàm Eval chẳng qua là quá trình tìm kiếm trên đồ thị AND/OR (xem mục 2.4) để tìm đồ thị con chứng minh giả thiết H.

Giả sử ta muốn hỏi hệ cho biết 'con ngựa nào có giá', tức là ta phải tìm ra tất cả các con ngựa có giá trị, trên cơ sở các luật và các sự kiện đã biết trong bộ nhớ làm việc. Câu hỏi trên có thể chuyển thành giả thiết chứa biến 'x có giá trị'. Giả sử WM ngoài các sự kiện trong ví dụ ở mục 7.2, còn chứa hai sự kiện

Bin là thắng cuộc.

Ken là thắng cuộc.

Cơ sở luật chứa thêm luật sau (luật thắng cuộc)

if ' w là thắng cuộc '.

then 'w là béo '

Trong trường hợp giả thiết H chứa biến và các luật của cơ sở luật cũng chứa biến, ta không thể áp dụng hàm đánh giá Eval(H) được. Trong trường hợp này, ứng với giả thiết H ta sẽ xây dựng cây suy diễn lùi Backward Tree. Để làm ví dụ ta xét giả thiết H = 'x có giá '.

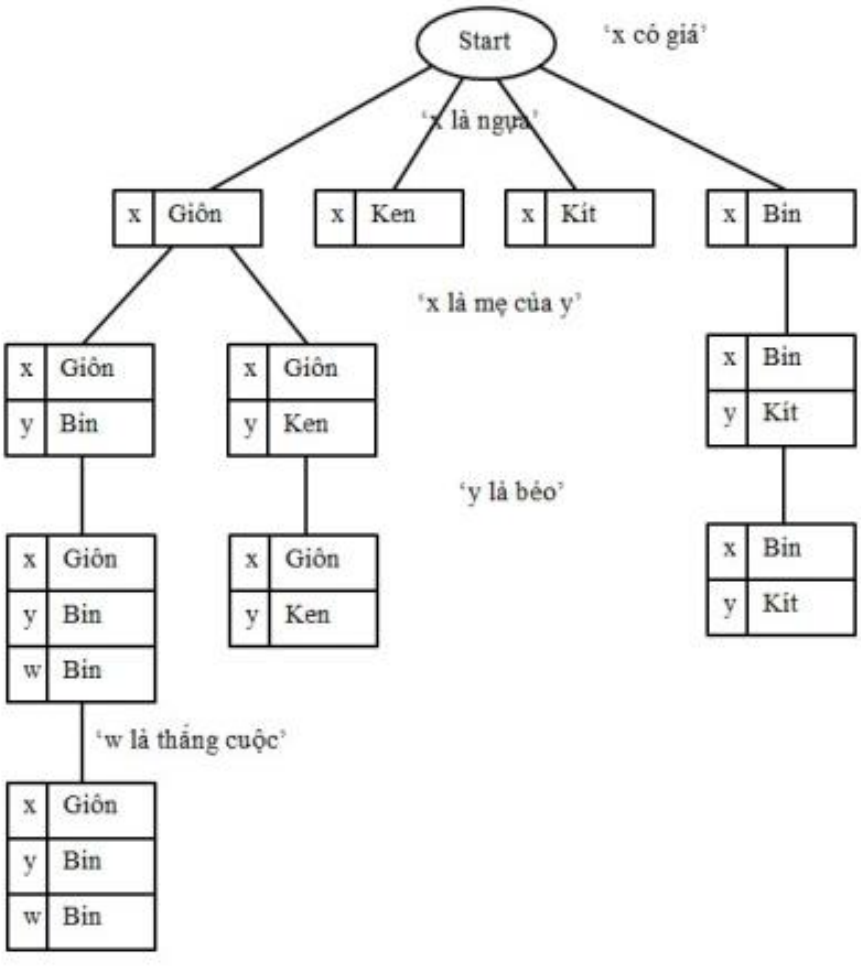
Cây suy diễn kết hợp với H được xây dựng như sau. Gốc của cây là H.

Đối sánh giả thiết H với phần kết luận của các luật để tìm ra các luật có phần kết luận trùng với H. Ở đây ta tìm ra luật mẹ. Đối sánh điều kiện thứ nhất 'x là con ngựa' của luật mẹ với WM, ta nhận được các ràng buộc biến (x → Giôn), (x → Ken), (x → Kit), (x → Bin). Gốc H có bốn đỉnh con ứng với bốn ràng buộc biến trên. Xét tiếp điều

kiện thứ hai 'x là mẹ của y'. Thay x bởi Giôn và đối sánh với WM, ta thấy điều kiện sẽ đúng khi y gắn với Bin hoặc Ken. Do đó đỉnh (x → Giôn) có hai con là (x → Giôn, y → Bin) và (x → Giôn, y → Ken). Tương tự đỉnh (x → Bin) có một con (x → Bin, y → Kit).

Xét đỉnh (x → Ken), điều kiện hai trở thành 'Ken là mẹ của y'. Đối sánh với WM, biến y không thể gắn với đối tượng nào cả. Xem 'Ken là mẹ của y' như giả thiết mới xuất hiên. Giả thiết này không trùng với phần kết luận của một luật nào cả. Trong trường hợp này ta đánh dấu đỉnh (x → Ken) là sai. Tương tự đỉnh (x → Kit) cũng được đánh dấu là sai. Các đỉnh được đánh dấu là sai sẽ không được phát triển tiếp.

Xét tiếp điều kiện thứ ba của luật mẹ 'y là béo' ở đây ta có ba đỉnh cần phát triển tiếp là (x → Gion, y → Bin), (x → Giôn, y → Ken) và (x → Bin, y → Kit). Điều kiện 'y là béo' trở thành các sự kiện trong WM khi y gắn với Ken và Kit. Mỗi đỉnh (x→Giôn, y → Ken) và (x → Bin, y → Kit) có một đỉnh con biểu diễn ràng buộc biến như cha nó. Với y gắn với Bin ta có khẳng định 'Bin là béo'. Khẳng định này không có trong WM. Xem nó như giả thiết mới. Giả thiết này là kết luận của luật như giả thiết mới. Giả thiết này là kết luận của luật thắng cuộc khi w được thay bởi Bin: Đỉnh (x → Giôn, y → Bin) có một đỉnh con là (x → Giôn, y → Bin, w → Bin). Xét điều kiện của luật thắng cuộc 'w là thắng cuộc' khi thay w bởi Bin ta nhận được 'Bin là thắng cuộc', đó là một sự kiện trong bộ nhớ làm việc. Đỉnh (x → Giôn, y → Bin, w → Bin) có một con biểu thị ràng buộc biến như cha nó. Ta nhận được cây suy diễn như trong hình sau.



Hình 7.4. Một cây suy diễn lùi

Từ các ràng buộc biến ở các lá không bị đánh dấu là sai, ta thấy x được gắn với Giôn và Bin. Như vậy các con ngựa có giá trị là Giôn và Bin.

Sau đây là thủ tục xây dựng cây suy diễn lùi Backward Tree liên kết với giả thiết H. Giả sử $\Re(H)$ là tập các luật có phần kết luận trùng với H.

procedure Backward Tree (H: Hypothesis) ;

1. Gốc của cây là H
2. Nếu $\Re(H) = \phi$ thì H được đánh dấu là sai,
3. **for** $R \in \Re(H)$ **do** {xây dựng cây gốc H tương ứng với luật R}

for $C \in$ cond(R) **do**

3.1. Gọi L là tập các lá của cây không bị đánh dấu là sai mà biểu diễn một ràng buộc biến nào đó.

3.2. for $u \in L$ **do**

3.2.1 Thay các biến trong điều kiện C bởi các đối tượng gắn với nó trong ràng buộc biến ở đỉnh u, ta nhận được khẳng định H(u).

3.2.2. Đối sánh H (u) với bộ nhớ làm việc WM, ta nhận được các ràng buộc biến mà điều kiện C được thoả mãn. Mỗi ràng buộc biến mới này là một đỉnh con của u.

3.2.3. Nếu không có ràng buộc biến nào để điều kiện C được thoả mãn thì ta xem H(u) là giả thiết mới. H(u) là đỉnh con của u. Gọi đệ qui Backward Tree (H (u)).

4. Xét các lá của cây không bị đánh dấu là sai. Thay các biến trong giả thiết H bởi các đối tượng gắn với nó trong các ràng buộc biến ở các lá, ta nhận được các câu trả lời cho câu hỏi được biểu diễn bởi giả thiết H.

Bài tập cuối chương

Bài 1: Cho cơ sở tri thức sau:

R_1: Nếu động vật có mỏ và động vật có lông vũ thì động vật là gia cầm.

R_2: Nếu động vật là gia cầm và động vật thích bơi lội thì động vật là thuỷ cầm (F6)

R_3: Nếu động vật là gia cầm và động vật có màng chân thì động vật là vịt

R_4: Nếu động vật là vịt và động vật là thuỷ cầm thì động vật là vịt trời

R_5: Nếu động vật là thuỷ cầm và động vật biết bay thì động vật là vịt trời

R_6: Nếu động vật là thuỷ cầm và động vật không biết bay thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Cun cút có mỏ

F_2: Cun cút có lông vũ

F_3: Cun cút không biết bay

F_4: Cun cút thích bơi lội

Trình bày quá trình lập luận tiến đối với cơ sở tri thức trên

Bài 2: Cho cơ sở tri thức sau:

Cơ sở luật RB gồm các luật sau:

R1: Nếu trời mưa thì đường ướt

R2: Nếu rửa đường thì đường ướt

R3: Nếu chuồn chuồn bay thấp thì trời mưa

R4: Nếu chuồn chuồn bay cao thì trời không mưa

R5: Nếu đường ướt và mây mù thì trời xấu

R6: Nếu đường ướt và quang mây thì trời không xấu

Cơ sở sự kiện sau:

F1: Chuồn chuồn bay thấp

F2: Mây mù

Trình bày quá trình lập luận tiến đối với cơ sở tri thức trên

Bài 3: Cho cơ sở tri thức sau:

R_1: Nếu động vật có 2 chân thì động vật là gia cầm.

R_2: Nếu động vật 4 chân thì động vật là gia súc.

R_3: Nếu động vật là gia súc và động vật có móng guốc thì động vật là loài móng guốc

R_4: Nếu động vật là loài móng guốc và động vật có 1 móng thì động vật là ngựa

R_5: Nếu động vật là loài móng guốc và động vật có 2 móng thì động vật là giống trâu bò

R_6: Nếu động vật là giống trâu bò và động vật thích cỏ khô thì động vật là bò

R_7: Nếu động vật là giống trâu bò và động vật thích cỏ ướt thì động vật là trâu

Cho tập sự kiện:

Bi có 4 chân

Bi có móng guốc

Bi có 1 móng

Trình bày quá trình lập luận tiến đối với cơ sở tri thức trên

Bài 4: Cho cơ sở tri thức sau:

R_1: Nếu động vật có lông vũ thì động vật là chim.

R_2: Nếu động vật có lông mao thì động vật là loài có vú.

R_3: Nếu động vật là loài có vú và động vật thịt thì động vật là thú ăn thịt

R_4: Nếu động vật là loài có vú và động vật có răng nhọn và động vật có móng vuốt thì động vật là thú ăn thịt

R_5: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có đốm sẫm thì động vật là báo châu Phi

R_6: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có văn đen thì động vật là hổ

R_7: Nếu động vật là chim và động vật không biết bay và động vật biết bơi và động vật có màu lông đen trắng thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Key có lông mao

F_2: Key có răng nhọn

F_3: Key có móng vuốt

F_4: Key có màu lông vàng hung

F_5: Key có đốm sẫm

Trình bày quá trình lập luận tiến đối với cơ sở tri thức và cơ sở sự kiện trên tìm kết luận Key là con gì?

Bài 5: Cho cơ sở tri thức sau:

R_1: Nếu động vật có lông vũ thì động vật là chim.

R_2: Nếu động vật có lông mao thì động vật là loài có vú.

R_3: Nếu động vật là loài có vú và động vật thịt thì động vật là thú ăn thịt

R_4: Nếu động vật là loài có vú và động vật có răng nhọn và động vật có móng vuốt thì động vật là thú ăn thịt

R_5: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có đốm sẫm thì động vật là báo châu Phi

R_6: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có văn đen thì động vật là hổ

R_7: Nếu động vật là chim và động vật không biết bay và động vật biết bơi và động vật có màu lông đen trắng thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Tom có lông mao

F_2: Tom có răng nhọn

F_3: Tom có móng vuốt

F_4: Tom có màu lông vàng hung

F_5: Tom có vằn đen

Trình bày quá trình lập luận tiến đối với cơ sở tri thức và cơ sở sự kiện trên tìm kết luận Tom là con gì?

Bài 6: Cho cơ sở tri thức sau:

R_1: Nếu động vật có lông vũ thì động vật là chim.

R_2: Nếu động vật có lông mao thì động vật là loài có vú.

R_3: Nếu động vật là loài có vú và động vật thịt thì động vật là thú ăn thịt

R_4: Nếu động vật là loài có vú và động vật có răng nhọn và động vật có móng vuốt thì động vật là thú ăn thịt

R_5: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có đốm sẫm thì động vật là báo châu Phi

R_6: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có văn đen thì động vật là hổ

R_7: Nếu động vật là chim và động vật không biết bay và động vật biết bơi và động vật có màu lông đen trắng thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Min có lông vũ

F_2: Min không biết bay

F_3: Min có màu lông đen trăng

F_4: Min biết bơi

Trình bày quá trình lập luận tiến đối với cơ sở tri thức và cơ sở sự kiện trên tìm kết luận Min là con gì?

Bài 7: Cho cơ sở tri thức sau:

R_1: Nếu động vật có 2 chân thì động vật là gia cầm.

R_2: Nếu động vật 4 chân thì động vật là gia súc.

R_3: Nếu động vật là gia súc và động vật có móng guốc thì động vật là loài móng guốc

R_4: Nếu động vật là loài móng guốc và động vật có 1 móng thì động vật là ngựa

R_5: Nếu động vật là loài móng guốc và động vật có 2 móng thì động vật là giống trâu bò

R_6: Nếu động vật là giống trâu bò và động vật thích cỏ khô thì động vật là bò

R_7: Nếu động vật là giống trâu bò và động vật thích cỏ ướt thì động vật là trâu

Cho tập sự kiện:

F_1: Bi có 4 chân

F_2: Bi có móng guốc

F_3: Bi có 1 móng

Cho giả thuyết Bi là ngựa

Hãy trình bày quá trình lập luận lùi để chứng minh hoặc bác bỏ giả thuyết trên.

Bài 8: Cho cơ sở tri thức sau:

R_1: Nếu động vật có 2 chân thì động vật là gia cầm.

R_2: Nếu động vật 4 chân thì động vật là gia súc.

R_3: Nếu động vật là gia súc và động vật có móng guốc thì động vật là loài móng guốc

R_4: Nếu động vật là loài móng guốc và động vật có 1 móng thì động vật là ngựa

R_5: Nếu động vật là loài móng guốc và động vật có 2 móng thì động vật là giống trâu bò

R_6: Nếu động vật là giống trâu bò và động vật thích cỏ khô thì động vật là bò

R_7: Nếu động vật là giống trâu bò và động vật thích cỏ ướt thì động vật là trâu

Cho tập sự kiện:

F_1: Bi có 4 chân

F_2: Bi có móng guốc

F_3:Bi có 2 móng

F_4:Bi thích cỏ ướt

Cho giả thuyết Bi là trâu

Hãy trình bày quá trình lập luận lùi để chứng minh hoặc bác bỏ giả thuyết trên.

Bài 9: Cho cơ sở tri thức sau:

R_1: Nếu động vật có lông vũ thì động vật là chim.

R_2: Nếu động vật có lông mao thì động vật là loài có vú.

R_3: Nếu động vật là loài có vú và động vật thịt thì động vật là thú ăn thịt

R_4: Nếu động vật là loài có vú và động vật có răng nhọn và động vật có móng vuốt thì động vật là thú ăn thịt

R_5: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có đốm sẫm thì động vật là báo châu Phi

R_6: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có văn đen thì động vật là hổ

R_7: Nếu động vật là chim và động vật không biết bay và động vật biết bơi và động vật có màu lông đen trắng thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Tata có lông mao

F_2: Tata có răng nhọn

F_3: Tata có móng vuốt

F_4: Tata có màu lông vàng hung

F_5: Tata có vằn đen

Cho giả thuyết Tata là hổ

Hãy trình bày quá trình lập luận lùi để chứng Tata hoặc bác bỏ giả thuyết trên.

Bài 10: Cho cơ sở tri thức sau:

R_1: Nếu động vật có lông vũ thì động vật là chim.

R_2: Nếu động vật có lông mao thì động vật là loài có vú.

R_3: Nếu động vật là loài có vú và động vật thịt thì động vật là thú ăn thịt

R_4: Nếu động vật là loài có vú và động vật có răng nhọn và động vật có móng vuốt thì động vật là thú ăn thịt

R_5: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có đốm sẫm thì động vật là báo châu Phi

R_6: Nếu động vật là thú ăn thịt và động vật có màu lông vàng hung và động vật có văn đen thì động vật là hổ

R_7: Nếu động vật là chim và động vật không biết bay và động vật biết bơi và động vật có màu lông đen trắng thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Min có lông vũ

F_2: Min không biết bay

F_3: Min có màu lông đen trăng

F_4: Min biết bơi

Cho giả thuyết Min là chim cánh cụt

Hãy trình bày quá trình lập luận lùi để chứng minh hoặc bác bỏ giả thuyết trên.

Bài 11: Cho cơ sở tri thức sau:

R_1: Nếu động vật có mỏ và động vật có lông vũ thì động vật là gia cầm.

R_2: Nếu động vật 4 chân thì động vật là gia súc.

R_3: Nếu động vật là gia cầm và động vật thích bơi lội thì động vật là thuỷ cầm

R_4: Nếu động vật là gia cầm và động vật có màng chân thì động vật là vịt

R_5: Nếu động vật là vịt và động vật là thuỷ cầm thì động vật là vịt trời

R_6: Nếu động vật là thuỷ cầm và động vật biết bay thì động vật là vịt trời

R_7: Nếu động vật là thuỷ cầm và động vật không biết bay thì động vật là chim cánh cụt

Cho tập sự kiện:

F_1: Toto có mỏ

F_2: Toto có lông vũ

F_3: Toto biết bay

F_4: Toto thích bơi lội

Cho giả thuyết Toto là vịt trời

Hãy trình bày quá trình lập luận lùi để chứng minh hoặc bác bỏ giả thuyết trên.

Chương 8. LƯỚI NGỮ NGHĨA VÀ HỆ KHUNG

Nội dung chính của chương:

Trong chương này chúng ta sẽ xét khái niệm lưới ngữ nghĩa và vấn đề suy diễn bằng thừa kế trong lưới ngữ nghĩa. Tiếp theo, nghiên cứu các thành phần cấu thành các khung và vấn đề suy diễn trong các hệ khung.

Lưới ngữ nghĩa (semantic network) là một trong các mô hình cơ bản biểu diễn tri thức, lưới ngữ nghĩa cho phép ta mô tả các đối tượng, các khái niệm (một lớp đối tượng) và các mối quan hệ giữa chúng. Trong chương này chúng ta sẽ xét khái niệm lưới ngữ nghĩa, vấn đề biểu diễn tri thức bởi lưới ngữ nghĩa và vấn đề suy diễn bằng thừa kế trong lưới ngữ nghĩa.

Trong phương pháp biểu diễn tri thức bởi các khung (frame), các sự kiện liên quan đến một đối tượng (hoặc một khái niệm) được tập hợp lại để tạo thành một khung. Khung là một đơn vị có cấu trúc để biểu diễn tri thức. Chúng ta sẽ nghiên cứu các thành phần cấu thành các khung và vấn đề suy diễn trong các hệ khung.

Mục tiêu cần đạt được của chương

Sau khi học xong chương này học viên cần nắm được khái niệm về khung, lưới ngữ nghĩa, ngôn ngữ mô tả khái niệm, Tính kế thừa trong lưới ngữ nghĩa. Cần hiểu và nắm được khái niệm về hệ khung, thiết lập khung cơ bản và cơ chế suy diễn trong hệ khung.

BÀI 12: LƯỚI NGỮ NGHĨA VÀ HỆ KHUNG (Số tiết: 2 tiết)

8.1 Ngôn ngữ mô tả khái niệm

Thủ tục chứng minh bác bỏ trong logic vị từ cấp 1 có độ phức tạp tính toán cao. Do đo, cần tìm kiếm ngôn ngữ con đặc biệt của logic vị từ cấp 1 mà có khả năng biểu diễn trong nhiều lĩnh vực áp dụng và đặc biệt thể hiện các thủ tục suy diễn hiệu quả. Đó chính là ngôn ngữ mô tả khái niệm.

Trong các ngành khoa học, khái niệm là đơn vị tri thức cơ sở. Người ta phát biểu các quy luật thông qua các khái niệm được đưa ra. Khi đưa ra một khái niệm mới, thường xác định các mối quan hệ của nó với các khái niệm đã biết và nêu ra các đặc tính của nó. Ví dụ: số nguyên tố là số nguyên dương chỉ chia hết cho 1 và chính nó.

Các khái niệm như “xe đạp”, “tam giác”,…có thể xem như sự mô tả các lớp đối tượng. Lớp đối tượng được xác định bởi một khái niệm là lớp con của lớp các đối tượng được xác định bởi các khái niệm tổng quát hơn. Ví dụ tam giác là lớp con của đa giác.

Ngôn ngữ mô tả khái niệm là ngôn ngữ đặc biệt cho phép mô tả các đối tượng, các lớp đối tượng, các mối quan hệ giữa chúng, các tính chất của các đối tượng và tính chất của các lớp.

Ngôn ngữ mô tả khái niệm chỉ sử dụng ba vị từ:

- isa(x,y) có nghĩa đối tượng x là thành viên của lớp y,
- ako(y,z) có nghĩa y là lớp con của lớp z,
- feature(x,p,v) có nghĩa đối tượng x có thuộc tính p nhận giá trị v

Các cá thể và các lớp tuân theo một số quy luật tổng quát sau:

- Nếu một đối tượng là thành viên của một lớp thì nó là thành viên của tất cả các lớp mẹ của lớp này.

  $$isa(x,y) \wedge ako(y,z) \Rightarrow isa(x,z)$$

- Quan hệ "là lớp con" có tính chất bắc cầu

  $$ako(x,y) \wedge ako(y,z) \Rightarrow ako(x,z)$$

- Nếu một lớp có đặc tính thì các thành viên của lớp cũng có đặc tính đó

  $$isa(x,y) \wedge feature(y,p,v) \Rightarrow feature(x,p,v)$$

Cần lưu ý các quy luật này có thể có các ngoại lệ

Ví dụ 43: Giả sử chúng ta có các câu sau đây mô tả một số khái niệm và một số cá thể.

- Động vật có vú là động vật có lông mao, có 4 chân và cho sữa

Chúng ta có thể biểu diễn các mô tả về loài vật và con vật bởi các câu trên ngôn ngữ mô tả khái niệm như sau:

Ako(Động vật có vú, động vật)

Feature(Động vật có vú, số chân, 4)

Feature(Động vật có vú, cho, sữa)

- Bin là chó cái màu xám

Chúng ta có thể biểu diễn bởi các câu trên ngôn ngữ mô tả khái niệm như sau:

Isa(Bin, chó)

Feature(Bin, màu lông, xám)

Feature(Bin, giới tính, cái)

Khi đã có một cơ sở tri thức gồm các câu trong ngôn ngữ mô tả khái niệm chúng ta cần có thủ tục suy ra các đặc tính mới của cá thể hay khái niệm được hỏi. Ta dùng các quy luật tổng quát trên.

Để thuận tiện cho quá trình suy diễn, chúng ta sẽ chuyển các tri thức được mô tả trong ngôn ngữ mô tả khái niệm sang dạng đồ thị đặc biệt là lưới ngữ nghĩa.

8.2 Lưới ngữ nghĩa

Trong mục này chúng ta sẽ xét lưới ngữ nghĩa như một mô hình biểu diễn tri thức. Sau đó chúng ta sẽ nghiên cứu phương pháp suy diễn bằng thừa kế trong lưới ngữ nghĩa.

8.2.1 Khái niệm

Đồ thị diễn tả thế giới sẽ trực quan và đỡ trừu tượng hơn các phương tiện thể hiện khác. Lưới ngữ nghĩa là một trong những kĩ thuật thể hiện tri thức đầu tiên mà con người sử dụng khi nghiên cứu các hệ thống tri thức.

***Định nghĩa**

Phương pháp thể hiện tri thức bằng cách dùng đồ thị gồm các đỉnh và các cung; đỉnh thể hiện đối tượng và cung thể hiện quan hệ giữa các đối tượng

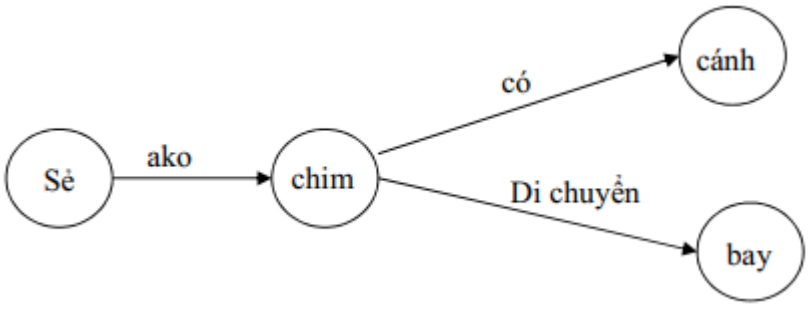
Lưới ngữ nghĩa là một đồ thị định hướng không có chu trình với các đỉnh và các cung được gắn nhãn:

- Các đỉnh biểu diễn các đối tượng hoặc các lớp đối tượng (các khái niệm), hoặc các giá trị của các thuộc tính.
- Các cung biểu diễn các quan hệ. Cung gắn nhãn isa đi từ đỉnh biểu diễn các đối tượng tới đỉnh biểu diễn lớp mà đối tượng là thành viên của lớp. Cung gắn nhãn ako đi từ đỉnh biểu diễn một lớp này tới đỉnh biểu diễn một lớp khác nếu lớp ứng với đỉnh nguồn của cung là lớp con của lớp ứng với đỉnh đích của cung. Các cung khác được gắn nhãn bởi tên của các thuộc tính, các cung này đi từ các đỉnh biểu diễn các đối tượng (hoặc các lớp), tới các đỉnh biểu diễn giá trị của thuộc tính.

Thông qua lưới ngữ nghĩa người ta dễ thấy đối tượng nào là quan trọng, nhận ra các thuộc tính và mối quan hệ của các đối tượng một cách trực quan. Đồ thị gồm nút và các cung nối các nút; nút và cung đều có nhãn ứng với đối tượng và mối quan hệ bản chất giữa các đối tượng. Nút thể hiện đối tượng, thuộc tính hoặc giá trị của thuộc tính. Các cung thể hiện mối quan hệ giữa các nút.

Từ định nghĩa lưới ngữ nghĩa trên, chúng ta dễ dàng chuyển các câu trong ngôn ngữ mô tả khái niệm thành một lưới ngữ nghĩa.

Ví dụ 44: Hình 8.1 là lưới ngữ nghĩa biểu diễn các tri thức được mô tả bởi ngôn ngữ mô tả khái niệm: "Chim sẻ là loài chim có cánh và biết bay". Lưới ngữ nghĩa có các nút chim, bay, cánh thể hiện ý nghĩa chim có cánh và chim biết bay. Nút sẻ nối nút chim theo cung ISA thể hiện ý nghĩa con chim sẻ là chim



Hình 8.1. Một lưới ngữ nghĩa

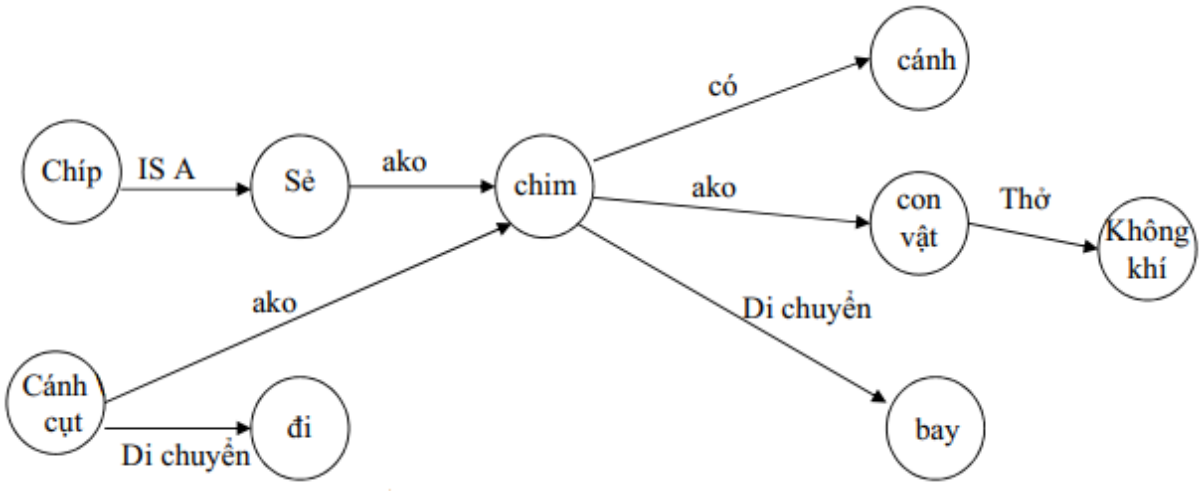
Người ta có thể nới rộng lưới ngữ nghĩa bằng cách thêm các nút và nối chúng vào đồ thị. Các nút mới ứng với các đối tượng bổ sung. Thông thường có thể nới rộng lưới ngữ nghĩa theo 1 trong 3 cách

Thêm một đối tượng tương tự

Thêm một đối tượng đặc biệt hơn

Thêm một đối tượng tổng quát hơn

Thứ nhất thêm chim cánh cụt thể hiện một loại chim mới. Thứ hai thêm chip có nghĩa là chip là con của sẻ và đồng thời nó là chim. Thứ ba có thể đưa ra đối tượng tổng quát như con vật. Lúc này người ta không những biết chim là con vật mà còn biết chip thở bằng không khí



Hình 8.2. Phát triển mạng ngữ nghĩa

8.2.2 Tính kế thừa

Khi chúng ta biểu diễn tri thức bởi lưới ngữ nghĩa, vấn đề quan trọng được đặt ra bây giờ là làm thế nào để xác định được giá trị của các thuộc tính của các đối tượng được hỏi.

Cơ chế suy diễn trong lưới ngữ nghĩa dựa trên ***nguyên lí kế thừa.*** Chẳng hạn, sự kiện "*chó có 4 chân*" được thừa kế từ sự kiện "*động vật có vú có 4 chân*". Tương tự, nhờ nguyên lí kế thừa, ta có thể suy ra rằng "*chim cánh cụt đẻ trứng*" và "*chim cánh cụt có 2 chân*". Một cách tổng quát, ta có thể phát biểu nguyên lý kế thừa như sau: ***Nếu một cá thể là thành viên của một lớp (hoặc một lớp là lớp con của lớp khác) thì nó có thể thừa kế các giá trị của các thuộc tính của lớp đó.***

Các giá trị của các thuộc tính của một lớp được gọi là các *giá trị ngầm định* (default value). Khi một đối tượng là thành viên của một lớp, nó có thể nhận giá trị ngầm định trong lớp đó cho các thuộc tính chưa được xác định. Chẳng hạn, giá trị ngầm định trong lớp chim của thuộc tính "*số chân*" là 2, chim cánh cụt thuộc lớp chim, do đó chim cánh cụt có 2 chân. Chúng ta có thể xác định các cá thể có các thuộc tính "*biết bay*" trong lớp chim là ***T***, nhưng chim cánh cụt là lớp con của lớp chim lại có giá trị ***F*** cho thuộc tính "*biết bay*". Trong trường hợp này, giá trị ngầm định của thuộc tính trong lớp mẹ không được lớp con (hoặc đối tượng con) kế thừa, giá trị ngầm định bị "*che lấp*" bởi giá trị thực tế đã được xác định trong lớp con (hoặc đối tượng con).

Bây giờ chúng ta sẽ xét cơ chế kế thừa trong lưới ngữ nghĩa. Trong lưới ngữ nghĩa, nếu chúng ta bỏ đi tất cả các cung biểu diễn các quan hệ thuộc tính – giá trị, chỉ giữ lại các các quan hệ *isa* và các quan hệ *ako*, chúng ta sẽ nhận được một đồ thị định hướng biểu diễn ***cấu trúc phân cấp kế thừa***. Chẳng hạn, từ lưới ngữ nghĩa trong hình 8.1, ta có cấu trúc phân cấp thừa kế trong hình 8.2.

Mục đích của suy diễn trong lưới ngữ nghĩa là: khi được hỏi về giá trị của một thuộc tính của một đối tượng (hoặc một lớp), ta cần xác định được giá trị của thuộc tính đó. Trước hết ta xét trường hợp đơn giản nhất, khi mà trong cấu trúc phân cấp thừa kế, từ một đỉnh chỉ có nhiều nhất một cung đi ra, tức là một đối tượng chỉ có thể là thành viên của một lớp duy nhất, một lớp chỉ có thể là lớp con của một lớp mẹ duy nhất. Chẳng hạn, lưới ngữ nghĩa trong hình 8.1 có các tính chất đó. Trong trường hợp này, để xác định giá trị của một thuộc tính cho một đối tượng, ta đi lên từ đỉnh ứng với đối tượng đó theo các cung được gắn nhãn **isa** hoặc **ako** cho tới lớp mà từ đó cung đi ra được gắn nhãn bởi thuộc tính được hỏi. Chẳng hạn, từ lưới ngữ nghĩa trong hình 8.1, để tìm câu trả lời cho câu hỏi "Sẻ có biết bay không?", ta đi từ đỉnh Sẻ đến đỉnh Chim, từ đỉnh này có cung đi ra nhãn biết bay (thuộc tính di chuyển) tới đỉnh biểu diễn giá trị bay. Do đo câu trả lời là Sẻ biết bay.

Tính chất quan trọng của lưới ngữ nghĩa là tính kế thừa. Nó cho phép các nút được bổ sung sẽ nhận các thông tin của nút liên quan ở đây là thông tin của nút đã có trước. Tính kế thừa của lưới ngữ nghĩa cho phép mã hoá tri thức một cách dễ dàng. Ví dụ khi thêm nút đặc biệt là chip vào mạng ngữ nghĩa, nó kế thừa thông tin theo cung ISA. Hơn nữa khi thêm nút đối tượng tổng quất con vật các nút khác kế thừa tính chất của nó. Đó chính là tính chất đặc biệt của mạng ngữ nghĩa

Phép toán trên mạng ngữ nghĩa

Để minh hoạ tính kế thừa, hãy xét 1 câu hỏi trên đồ thị. Chẳng hạn tại nút chim, người ta muốn hỏi xem con chíp chíp hoạt động như thế nào? Thông qua cung hoạt động người ta biết được nó bay

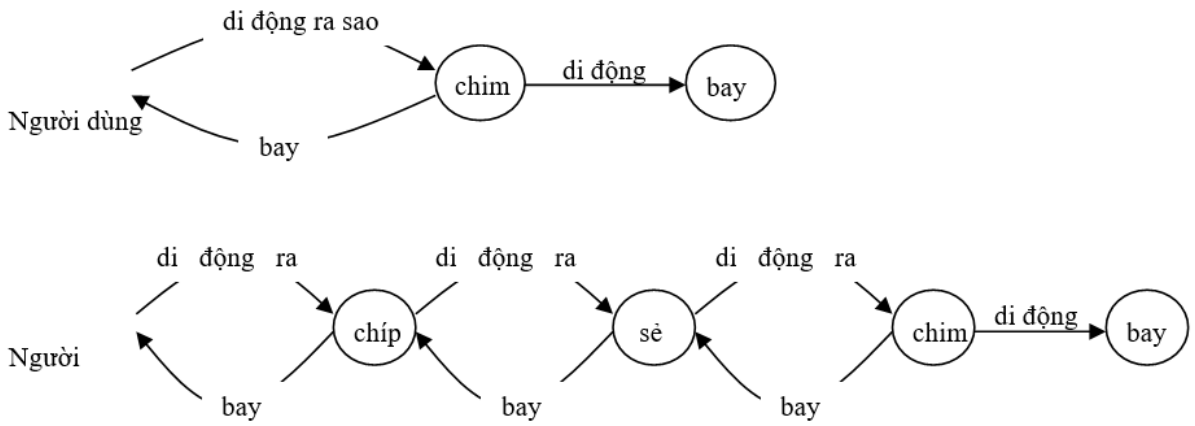
Nếu không có cung trực tiếp như vậy có thể tìm câu trả lời thông qua cung ISA. Khi cần biết chíp hoạt động như thế nào, người ta trả lời dần dần cho tới khi thu được bay thì trở về nút câu hỏi

Ngoại lệ

Tính chất kế thừa các thuộc tính của đối tượng này sang đối tượng khác trong lưới ngữ nghĩa tỏ ra thuận lợi đối với việc mô tả các đối tượng. Tuy nhiên đôi khi việc kế thừa lại áp đặt cứng nhắc các giá trị thuộc tính không chính xác cho một số đối tượng gây cho ta hiểu lầm về đối tượng. Chẳng hạn cố đi tìm khả năng bay của cánh cụt theo cung ISA. Để khỏi phải đi tìm mãi một điều phi lý là chim cánh cụt người ta dùng kĩ thuật gọi là ngoại lệ

Kĩ thuật này yêu cầu người dùng chỉ ra ngoại lệ để tránh một nút kế thừa thông tin không đúng. Khi ấy cần nối thêm một nút tới nó với thông tin mới; thông tin này cho phép cập nhật thông tin không đúng. Do vậy khi trả lời câu hỏi cánh cụt hoạt động như

thế nào người ta phát hiện ra thông tin đi quanh đối tượng; do vậy người ta không đi theo cung IS A để tìm xem nó hoạt động ra sao nữa. Kĩ thuật ngoại lệ tuy đơn giản nhưng được dùng một cách hiệu quả để tránh các sai sót



8.3 Hệ khung

8.3.1 Khái niệm

Một trong những kĩ thuật thể hiện tri thức là dùng khung. Khung được phát triển từ khái niệm lược đồ. Năm 1932 FC Barlett đề nghị dùng lược đồ để thay mạng ngữ nghĩa trong việc thể hiện tri thức. Một lược đồ được coi là khối tri thức điển hình về khái niệm hay đối tượng nào đó và gồm cả tri thức thủ tục và tri thức mô tả. Chẳng hạn trong lược đồ có tri thức về cánh và chân của chim cũng như cách săn bắt nó. Một lược đồ có nhiều thông tin về khái niệm sẽ phục vụ tốt cho các nghiên cứu đặc biệt.

Những người thiết kế hệ chuyên gia tiếp thu tư tưởng về lược đồ nhằm nắm bắt và thể hiện tri thức khái niệm trong hệ thống. Họ cụ thể hoá lược đồ dưới dạng hình thức của khung, là khái niệm do Marvin Minski đưa ra vào năm 1975.

Các khung được sử dụng để biểu diễn tri thức về một đối tượng cụ thể, một khái niệm (một lớp đói tượng), hoặc một hoàn cảnh nào đó. Trong mục này chúng ta sẽ xét khái niệm khung, việc biểu diễn tri thức trong một lĩnh vực bởi hệ khung và cơ chế suy diễn trong một hệ khung.

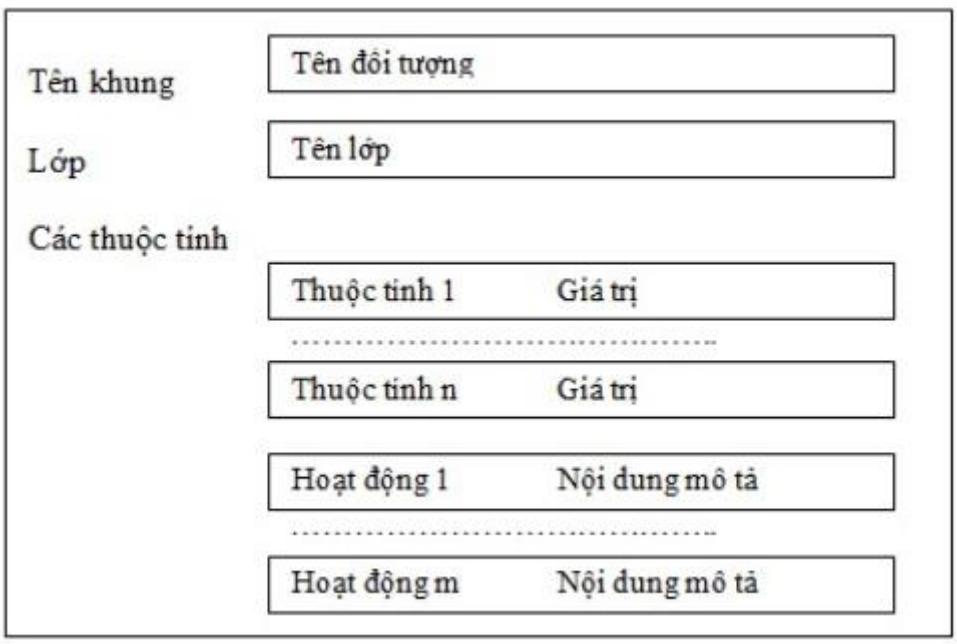
***Định nghĩa khung**

Cấu trúc dữ liệu để thể hiện tri thức đa dạng về khái niệm hay đối tượng nào đó

Ban đầu Minski mô tả khung như sau: khi một người vào hoàn cảnh mới (hay thay thế cách nhìn của người ấy về một vấn đề) họ tìm trong trí nhớ một cấu trúc gọi là khung. Đây là một khung được nhớ lại một cách thích nghi để phù hợp với thực tại bằng cách thay đổi các chi tiết cần thiết

8.3.2 Thiết kế khung cơ bản

Khung là đơn vị có cấu trúc để biểu diễn tri thức. Có thể hiểu khung như là một cấu trúc dữ liệu. Mỗi khung đều có tên, tên của khung là tên của đối tượng hoặc của khái niệm hoặc của hoàn cảnh mà khung biểu diễn. Mỗi khung gồm một số thành phần, các thành phần của khung được xem như là các lỗ (slot) chứa các thông tin về đối tượng (đối tượng ở đây cần được hiểu là một đối tượng cụ thể, một khái niệm, một hoàn cảnh điển hình hoặc một hoàn cảnh cá biệt nào đó). Mỗi lỗ có tên và giá trị. Các lỗ biểu diễn các quan hệ với các đối tượng khác hoặc biểu diễn các thuộc tính của đối tượng.



Hình 8.4 Mô hình hệ khung

Chúng ta sẽ quan tâm tới hai lỗ đặc biệt. Một lỗ đặc biệt có tên là **isa** (hoặc instance-of) biểu diễn đối tượng được mô tả là thành viên của một lớp đối tượng. Một lỗ đặc biệt khác có tên là **ako** biểu diễn một lớp là lớp con của một lớp khác (lớp mẹ trực tiếp). Các khung mô tả các đối tượng cụ thể sẽ được gọi là các cá thể (instance frame hoặc instance), các khung mô tả các khái niệm (chẳng hạn, khái niệm chim, bò sát,…), hoặc các hoàn cảnh điển hình (chẳng hạn, một phòng hội thảo điển hình) sẽ được gọi là các lớp (class).

Thông tin chứa đựng trong các lỗ của một khung có thể là một chỉ dẫn tới một khung khác, chẳng hạn thông tin chứa trong các lỗ có tên là isa hoặc ako. Thông tin chứa trong các lỗ biểu diễn các thuộc tính của đối tượng có thể là giá trị của thuộc tính đó, hoặc là một thủ tục cho phép ta tính giá trị của thuộc tính đó thông qua các thông tin khác.

Trong mô hình biểu diễn tri thức bởi các khung, các thông tin về một đối tượng, được bó lại để tạo thành một đơn vị khung biểu diễn đối tượng đó. Từ lưới ngữ nghĩa

đã cho, mỗi đỉnh (biểu diễn đối tượng) và các cung xuất phát từ nó có thể được gom lại thành một khung.

Ví dụ 45: từ lưới ngữ nghĩa trong hình 8.1 với tri thức về loài chim: Chim là động vật, biết bay, có hai cánh, ta có thể xây dựng lên một hệ khung.

KHUNG: CHIM	
Ako	Động vật
KN bay	Biết bay
Số cánh	2

Bây giờ chúng ta nghiên cứu cơ chế suy diễn trong các hệ khung. Cũng như đối với lưới ngữ nghĩa, vấn đề suy diễn trong các hệ khung là, khi được hỏi về giá trị của một thuộc tính đó. Cơ chế suy diễn trong các hệ khung dựa trên nguyên lý kế thừa.

Giả sử chúng ta có một số tri thức về một số loài chim như sau:

• *Chim là động vật, biết bay, có hai chân*

• *Hải âu là một loài chim, có lông màu đen - trắng, có cỡ trung bình là 115*

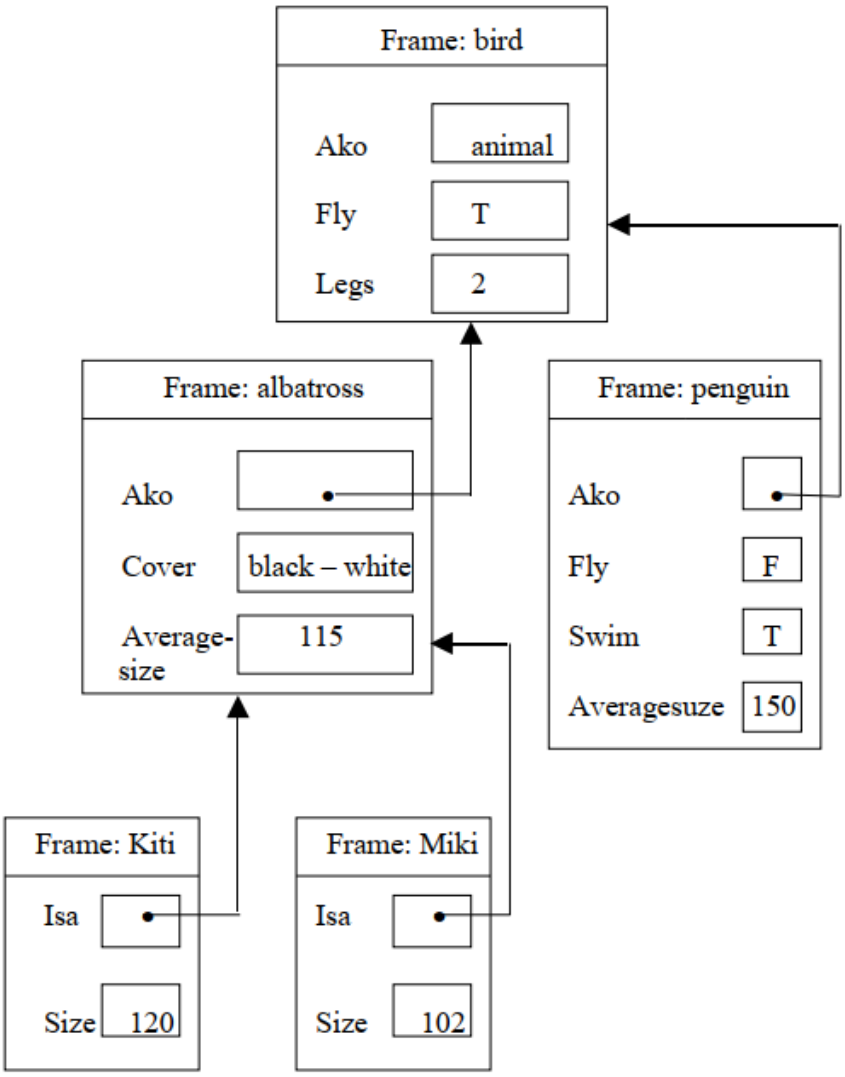
• *Chim cánh cụt (Penguin) là một loài chim, không biết bay, biết bơi, có cỡ trung bình là 150*

• *Kiti là một con hải âu, có cỡ 120*

• *Miki là một con hải âu, có cỡ 102*

Từ các tri thức trên, ta xây dựng được một hệ khung được cho trong hình 8.5.

Giả sử chúng ta được hỏi "Kiti có biết bay không?". Trong khung Kiti không có chỗ fly (điều đó có nghĩa là các thông tin trong khung Kiti không cho phép ta xác định được giá trị của thuộc tính được hỏi). Theo chỉ dẫn trong lỗ isa của khung Kiti, ta tìm đến khung **albatross(chim hải âu)**. Trong khung này cũng không có lỗ fly. Theo chỉ dẫn trong lỗ ako của khung này, ta tìm tới khung **bird(chim)**. Trong khung này, giá trị chứa trong lỗ fly là T (true). Khung Kiti kế thừa giá trị ngầm định T của lỗ fly từ khung mẹ **bird.** Như vậy ta đã tìm ra câu trả lời là "Kiti biết bay".



Hình 8.5 Một hệ khung

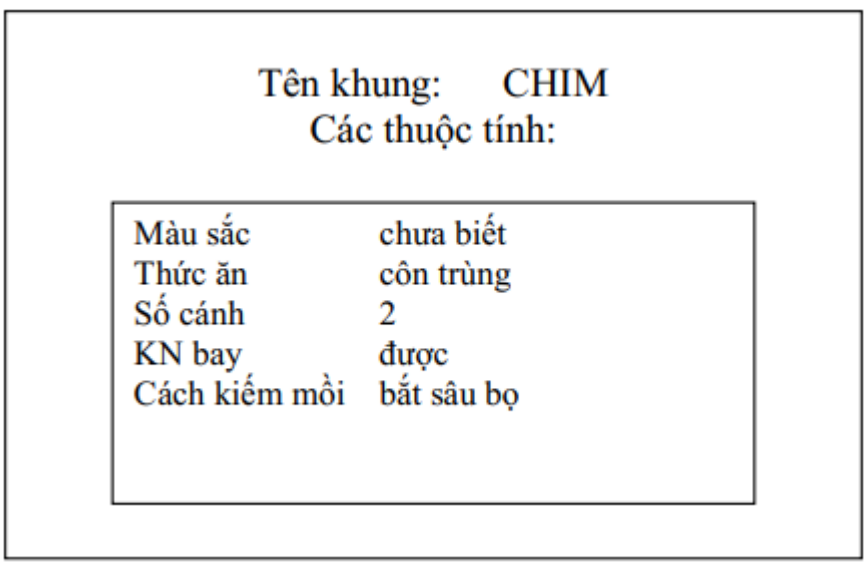
Tổng quát, để xác định giá trị của một thuộc tính của một đối tượng, ta có thể thực hiện như sau. Xuất phát từ khung biểu diễn đối tượng được hỏi. Nếu trong khung có lỗ biểu diễn thuộc tính được hỏi thì giá trị chứa trong lỗ đó là giá trị cần tìm. Nếu không, ta xác định giá trị của thuộc tính được hỏi bằng kế thừa. Ta đi từ khung hiện thời tới khung tổng quát hơn theo chỉ dẫn được chứa trong các lỗ isa hoặc ako cho tới khung mẹ mà tại đó có lỗ biểu diễn thuộc tính được hỏi. Giá trị ngầm định trong lỗ đó được lấy là giá trị của thuộc tính được hỏi.

Trên đây chúng ta mới chỉ đề cập đến sự kế thừa đơn, tức là mỗi khung chỉ có thể có một khung mẹ trực tiếp duy nhất, hay nói cách khác mỗi khung chỉ có một đường kế thừa duy nhất. Trong trường hợp kế thừa bội, khi mỗi khung có nhiều hơn một khung mẹ, giá trị một thuộc tính của khung có thể kế thừa từ nhiều khung mẹ khác nhau. Trong trường hợp này, cũng như trong lưới ngữ nghĩa, chúng ta cần có chiến lược quyết định khung mẹ nào có quyền cho kế thừa.

Các hệ khung đã được sử dụng trong nhiều hệ chuyên gia, trong các hệ nhận dạng, trong các hệ xử lý ngôn ngữ tự nhiên,… Lý thuyết các hệ khung là cơ sở của phương pháp luận lập trình định hướng đối tượng.

8.3.3 Khung lớp

Một khung lớp thể hiện các tính chất tổng quát của tập các đối tượng chung. Chẳng hạn người ta cần mô tả tính chất tổng quát như bay, có cánh, sống tự do...của cả loài



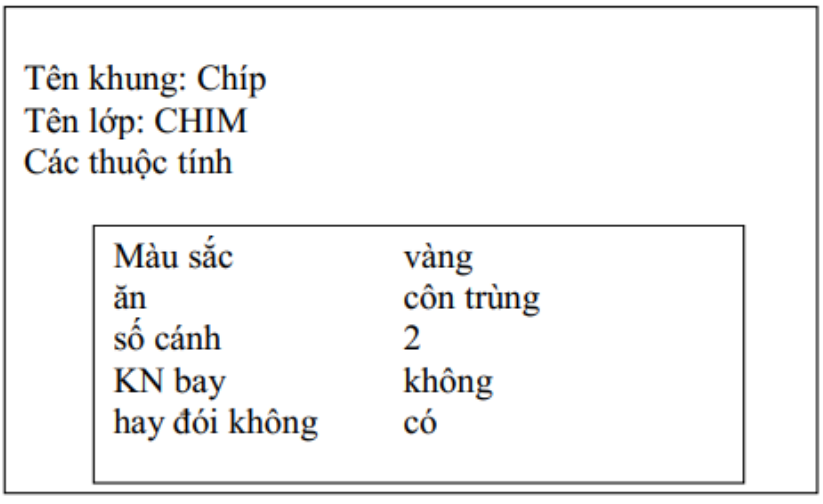
Hình 8.6. Khung lớp cho phép mô tả các đối tượng chim

Trong mỗi khung lớp người ta xác định các tính chất chung cho tất cả các đối tượng trong lớp; đôi khi có cả giá trị mặc định. Tính chất của các đối tượng được phân ra 2 loại điển hình là tĩnh hay động. Tính chất tĩnh mô tả khía cạnh không thay đổi giá trị của đối tượng. Tính chất động mô tả khía cạnh của đối tượng có giá trị thay đổi khi hệ thống hoạt động

Ví dụ 46:

Khung lớp có tên khung là chim mô tả đối tượng thuộc loại chim. Các tính chất mô tả đặc trưng chung của hầu hết loài chim. Tuy đặt trong cùng khối thuộc tính của khung nhưng cần phân biệt tính chất màu, mấy cánh như là tính chất tĩnh, đói, hoạt động như tính chất động

Người ta dùng một dạng khung khác để mô tả một thể hiện của khung lớp và được gọi là khung thể hiện. Khi tạo ra một thể hiện của một lớp, khung này kế thừa tính chất và giá trị của lớp. Có thể thay đổi giá trị để phù hợp với thể hiện cụ thể. Thậm chí người ta có thể thêm tính chất khác đối với khung thể hiện



Hình 8.7. Khung thể hiện ứng với một đối tượng cụ thể là Chíp thuộc lớp chim

8.3.4 Tính kế thừa của khung

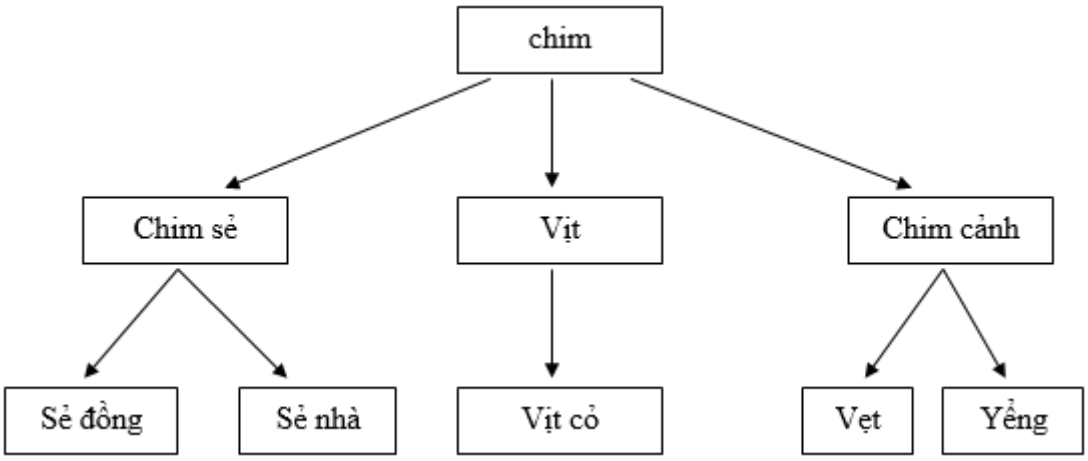
Cũng như tính chất kế thừa giữa các đối tượng trong mạng ngữ nghĩa, khung thể hiện nhận giá trị kế thừa từ khung lớp. Khi tạo một khung thể hiện người ta khẳng định khung đó là thể hiện của một khung lớp. Khẳng định này cho phép nó kế thừa các thông tin từ khung lớp.

Trong ví dụ chip kế thừa các tính chất của chim. Nói chung người ta cho phép một thể hiện nhận các giá trị mặc định hoặc nhận giá trị cố định đưa vào..

Bên cạnh việc kế thừa thông tin từ lớp của nó một thể hiện còn kế thừa hành vi. Do vậy trong khung lớp chứa cả thủ tục thường được gọi là phương pháp. Thủ tục này xác định hành động mà khung sẽ thực hiện. Chẳng hạn trong khung lớp chim người ta có thể bổ sung thủ tục chỉ hành động chim phải làm khi có thuộc tính đói hay không nhận giá trị có.

8.3.5 Cấu trúc phân cấp

Ngoài các khung lớp đơn giản và các thể hiện gắn với nó người ta có thể tạo ra cấu trúc khung phức tạp.



Ví dụ 47:

Dùng cấu trúc phân cấp các khung để mô tả thế giới loài chim. Cấu trúc này tổ chức khái niệm về chim theo các mức trừu tượng khác nhau. Khung ở mức cao mang thông tin chung về tất cả loài chim. Mức giữa có khung lớp con mang thông tin đặc thù hơn của nhóm chim. Mức cuối cùng là khung thể hiện ứng với đối tượng cụ thể.

Bài tập cuối chương

Bài 1: Xây dựng mạng ngữ nghĩa sau:

+ *Người* là động vật, có 2 chân, 2 tay, 2 mắt, 1 mũi, 1 mồm

+ *Động vật* thở bằng không khí, có lông

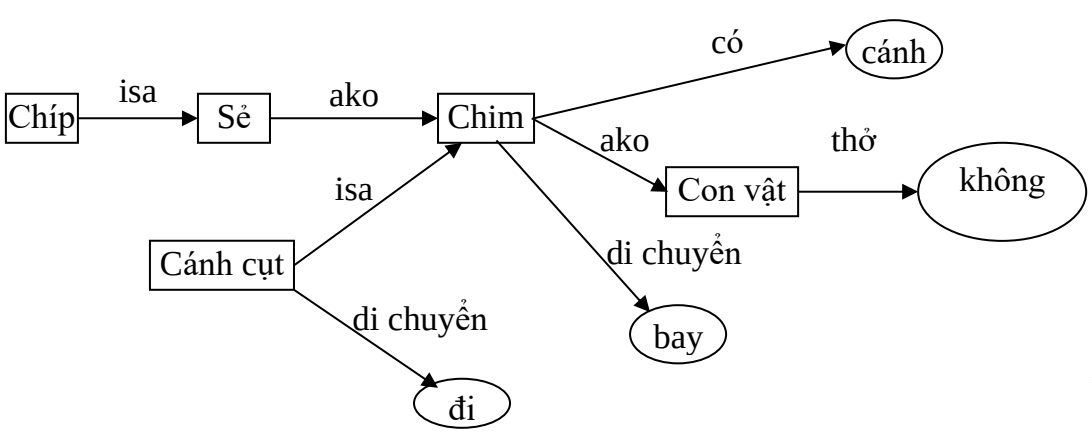
+ *Giáo sư* là *người*, giỏi chuyên môn

+ *Ông An* là *giáo sư* về lĩnh vực kinh tế

Bài 2: Xây dựng mạng ngữ nghĩa sau:

- Động vật có lông, có thể đi trên cạn
- Chim là động vật, có thể bay, có cánh, có lông vũ
- Vẹt là chim, có thể nói, mầu vàng
- Họa mi là chim, có thể hót, có đuôi

Bài 3: Cho mạng ngữ nghĩa sau:



Hãy xác định giá trị của các thuộc tính của chim Chíp

Bài 4: Thể hiện dưới dạng Frame với các tri thức sau:

+ Người là động vật, có 2 chân, 2 tay, 2 mắt, 1 mũi, 1 mồm

+ Động vật thở bằng không khí, có lông

+ Giáo sư là người, giỏi chuyên môn

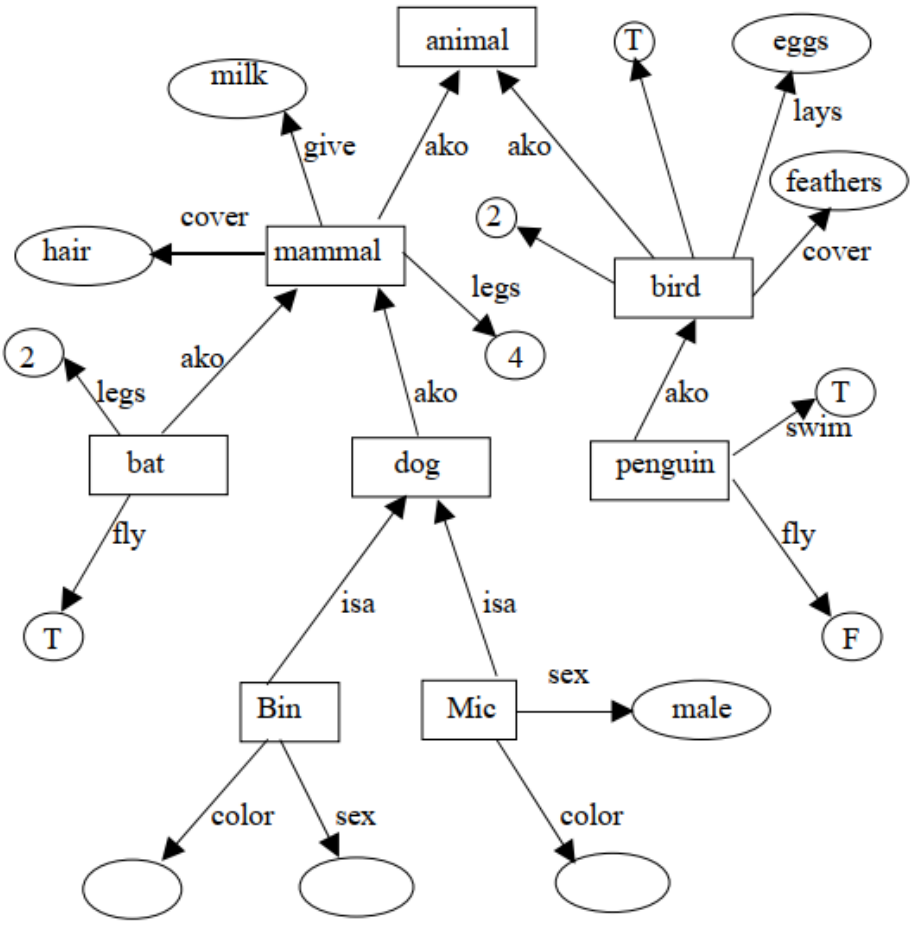
+ Ông An là giáo sư về lĩnh vực kinh tế

Bài 5: Cho các câu sau đây mô tả một số khái niệm và một số cá thể.

- Động vật có vú (mammal) là động vật (animal) có lông mao, có 4 chân cho sữa.
- Chim là động vật có lông vũ , có hai chân, biết bay, đẻ trứng.
- Chó là động vật có vú, có đức tính trung thành.
- Bin là chó cái, màu xám.
- Mic là chó đực, màu đen.
- Dơi (Bat) là động vật có vú, có hai chân, biết bay
- Chim cánh cụt (penguin) là loài chim biết bơi, không biết bay.

Yêu cầu biểu diễn các mô tả về một số loài vật và con vật trên bởi các câu trên ngôn ngữ mô tả khái niệm và xây dựng mạng ngữ nghĩa tương ứng.

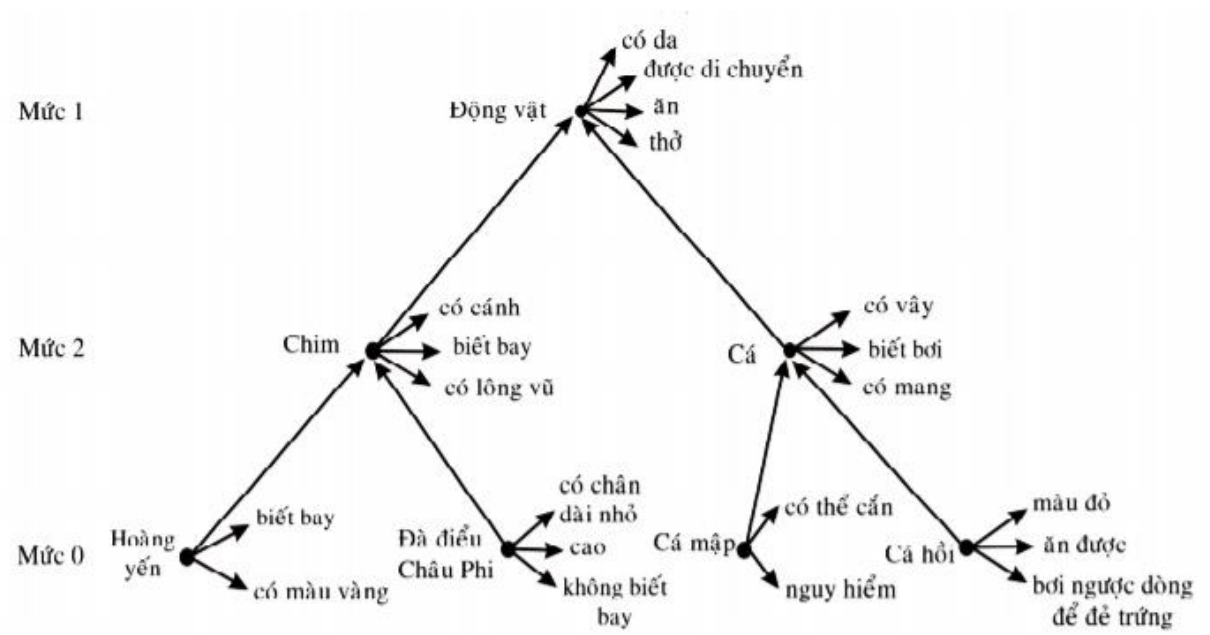
Bài 6 : Cho lưới ngữ nghĩa :



Hãy tìm câu trả lời cho câu hỏi "Mic có mấy chân?"

Bài 7 : Từ lưới ngữ nghĩa cho từ bài 5, hãy cho biết 'Bat có có cho sữa không ?'

Bài 8 : Cho mạng ngữ nghĩa sau :



Cho biết Hoàng yến có những đặc điểm gì ?

Bài 9 : Biểu diễn thông tin về động vật bằng hệ khung.

Biểu diễn thông tin về các loại động vật: "Cá", "Chim", và "Mèo". Sử dụng các hệ khung có các thuộc tính và giá trị khác nhau để biểu diễn loại động vật và các đặc điểm của chúng như sau:

1 Hệ Khung cho "Động Vật"

- Tên: Động Vật
- Sống: ?
- Ăn: ?
- Di chuyển: ?

2 Hệ Khung cho "Cá"

- Tên: Cá
- Sống: Nước
- Ăn: Thức ăn trong nước
- Di chuyển: Bơi
- Kế thừa từ: Động Vật

3 Hệ Khung cho "Chim"

- Tên: Chim
- Sống: Không trung, Mặt đất
- Ăn: Hạt, Côn trùng
- Di chuyển: Bay
- Kế thừa từ: Động Vật

4 Hệ Khung cho "Mèo"

- Tên: Mèo
- Sống: Nhà, Ngoài trời
- Ăn: Thịt
- Di chuyển: Đi bộ, Chạy
- Kế thừa từ: Động Vật

Các hệ khung cho "Cá", "Chim", và "Mèo" đều kế thừa các thuộc tính từ hệ khung "Động Vật".

Bài 10: Biểu diễn thông tin từ bài 9 tương ứng bằng mạng ngữ nghĩa.

CÂU HỎI THƯỜNG GẶP

Câu hỏi 1: Không gian trạng thái là gì? Biểu diễn vấn đề trong không gian trạng thái là gì?

Không gian trạng thái:

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

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

Biểu diễn vấn đề trong không gian trạng thái

Một phạm vi rộng lớn các vấn đề, đặc biệt các câu đố, các trò chơi, có thể mô tả bằng cách sử dụng khái niệm trạng thái và toán tử (phép biến đổi trạng thái). Biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:

- Trạng thái ban đầu.
- Một tập hợp các toán tử. Trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.

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

Ta sẽ ký hiệu không gian trạng thái là U, trạng thái ban đầu là u_0 ($u_0 \in U$). Mỗi toán tử R có thể xem như một ánh xạ R: U→U. Nói chung R là một ánh xạ không xác định khắp nơi trên U.

Một tập hợp T các trạng thái kết thúc (trạng thái đích). T là tập con của không gian U. Trong vấn đề của du khách trên, chỉ có một trạng thái đích, đó là thành phố B. Nhưng trong nhiều vấn đề (chẳng hạn các loại cờ) có thể có nhiều trạng thái đích và ta không thể xác định trước được các trạng thái đích. Nói chung trong phần lớn các vấn đề hay, ta chỉ có thể mô tả các trạng thái đích là các trạng thái thỏa mãn một số điều kiện nào đó.

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

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

Câu hỏi 2: Vấn đề tìm kiếm trong không gian trạng thái? Cây tìm kiếm là gì?

Vấn đề tìm kiếm trong không gian trạng thái

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

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

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

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

Cây tìm kiếm

Cây tìm kiếm là cây mà các đỉnh được gắn bởi các trạng thái của không gian trạng thái. Gốc của cây tìm kiếm tương ứng với trạng thái ban đầu. Nếu một đỉnh ứng với trạng thái u, thì các đỉnh con của nó ứng với các trạng thái v kề u.

Câu hỏi 3 : Làm thế nào để chọn giải thuật tìm kiếm phù hợp?

1. ***Phân tích bài toán:***

- Kích thước không gian tìm kiếm: Bài toán có một không gian tìm kiếm nhỏ, lớn hay vô hạn?
- Yêu cầu về thời gian: Cần giải quyết bài toán trong thời gian thực? Hay có thời gian dài để tìm kiếm?
- Tính chất của bài toán: Bài toán có một giải pháp duy nhất hay nhiều giải pháp? Có cần tìm giải pháp tối ưu hay chỉ cần giải pháp chấp nhận được?

2. ***Hiểu về giải thuật:***

- Độ phức tạp: Một số giải thuật có độ phức tạp thời gian và không gian cao, và có thể không phù hợp với bài toán có không gian tìm kiếm lớn.
- Tính chất của giải thuật: Một số giải thuật, như A*, đảm bảo tìm ra giải pháp tối ưu, trong khi một số giải thuật khác, như Greedy Search, chỉ tìm kiếm một giải pháp chấp nhận được.

3. ***Đánh giá heuristic (nếu sử dụng):***

- Khả năng hướng dẫn: Một heuristic tốt sẽ giúp giảm thiểu không gian và thời gian tìm kiếm.
- Tính không chệch (admissibility): Đối với giải thuật như A*, heuristic cần phải không chệch để đảm bảo tìm ra giải pháp tối ưu.

4. ***Thực nghiệm:***

- Không có giải thuật nào là tốt nhất cho mọi bài toán. Thường, việc thực hiện một số thử nghiệm trên dữ liệu thực tế hoặc dữ liệu mô phỏng sẽ giúp bạn hiểu rõ hơn về hiệu quả của mỗi giải thuật.

5. ***Tối ưu hóa và điều chỉnh:***

- Sau khi chọn một giải thuật, có thể cần phải điều chỉnh và tối ưu hóa nó dựa trên kết quả thực nghiệm.

6. ***Xem xét vấn đề bộ nhớ:***

- Một số giải thuật, như BFS, có thể tiêu thụ một lượng lớn bộ nhớ. Nếu bộ nhớ là một hạn chế, bạn cần xem xét việc sử dụng một giải thuật tìm kiếm khác hoặc tối ưu hóa giải thuật hiện tại.

7. ***Tính ứng dụng:***

- Trong một số ứng dụng, việc đưa ra quyết định nhanh chóng có thể quan trọng hơn việc tìm ra giải pháp tối ưu. Trong trường hợp đó, bạn có thể cần một giải thuật tìm kiếm nhanh chóng hơn là tối ưu.

Câu hỏi 4 : Làm thế nào để tìm ra một hàm heuristic tốt?

Heuristic là một hàm đánh giá giúp hướng dẫn quá trình tìm kiếm hoặc quyết định Chọn lựa heuristic phù hợp cho một bài toán cụ thể là khó. Tuy nhiên, dưới đây là một số tiêu chí và gợi ý để xây dựng hàm heuristic:

- ***Tính liên quan:*** Heuristic nên phản ánh một số khía cạnh quan trọng của bài toán. Ví dụ, trong bài toán Người du lịch (TSP), một heuristic phổ biến là khoảng cách Euclidean giữa hai thành phố.
- ***Tính không chệch (Admissibility):*** Đối với một số giải thuật như A*, heuristic được coi là không chệch nếu nó không bao giờ đánh giá quá cao chi phí thực tế để đạt đến mục tiêu từ một trạng thái cụ thể.
- ***Tính đơn điệu (Consistency):*** Một heuristic được coi là đơn điệu nếu chi phí ước lượng từ một trạng thái đến mục tiêu không tăng khi ta đi theo một bước tối ưu. Điều này quan trọng trong một số giải thuật như A*.
- ***Tính đơn giản và hiệu quả về mặt tính toán:*** Một heuristic tốt thường đơn giản và nhanh chóng để tính toán. Nếu mất quá nhiều thời gian để tính toán heuristic, điều này có thể làm mất lợi ích của việc sử dụng nó.
- ***Thử nghiệm và đánh giá:*** Một cách tốt để đánh giá heuristic là thử nghiệm nó trong bài toán thực tế. So sánh kết quả và hiệu suất của giải thuật khi sử dụng heuristic với khi không sử dụng.
- ***Kết hợp nhiều heuristic:*** Trong một số trường hợp, việc kết hợp thông tin từ nhiều heuristic có thể cung cấp kết quả tốt hơn so với việc sử dụng một heuristic duy nhất.

Câu 5: Ứng dụng nổi bật của các giải thuật heuristic tìm kiếm trong AI là gì?

Giải thuật heuristic được sử dụng rộng rãi trong trí tuệ nhân tạo (AI) vì chúng cung cấp một cách hiệu quả để giải quyết các bài toán phức tạp mà không cần phải tìm kiếm toàn bộ không gian trạng thái. Dưới đây là một số ứng dụng tiêu biểu của các giải thuật heuristic trong AI:

1. ***Tối ưu hóa và Lập kế hoạch:***
   - Bài toán người du lịch (TSP): Tìm kiếm một lộ trình ngắn nhất đi qua tất cả các điểm cho trước và quay trở lại điểm xuất phát.
   - Bài toán đóng gói – cái túi (Knapsack Problem): Chọn một tập hợp các vật phẩm sao cho tổng giá trị là cao nhất mà không vượt quá giới hạn trọng lượng cho phép.
2. ***Trò chơi và Chiến lược:***
   - Cờ vua, cờ caro, và trò chơi khác: Sử dụng heuristic để đánh giá các bước đi tiềm năng và quyết định nước đi tiếp theo.
   - Tìm kiếm cắt tỉa alpha-beta: Giảm thiểu không gian tìm kiếm bằng cách loại bỏ các nhánh không cần thiết trong trò chơi hai người.
3. ***Robotics và Điều hướng:***
   - Điều hướng robot trong môi trường có chướng ngại vật: Sử dụng heuristic để ước lượng khoảng cách từ robot đến mục tiêu.
   - Điều hướng xe không người lái: Tìm đường đi tối ưu dựa trên dữ liệu từ cảm biến và bản đồ.
4. ***Nhận dạng Mẫu và Học máy:***
   - Lựa chọn đặc trưng (Feature Selection): Chọn một tập con của các đặc trưng để cải thiện hiệu suất của mô hình học máy.
   - Phân cụm (Clustering): Sử dụng heuristic để xác định trung tâm cụm trong các thuật toán như K-means.
5. ***Thiết kế mạng và Tối ưu hóa mạng:***
   - Thiết kế mạng truyền thông: Tìm vị trí tối ưu để đặt các trạm phát sóng hoặc router.
   - Tối ưu hóa đường truyền trong mạng: Chọn lộ trình tối ưu cho dữ liệu để giảm thiểu độ trễ hoặc tải trên mạng.
6. ***Thời khóa biểu và Lập lịch:***

- Lập thời khóa biểu cho trường học hoặc tổ chức: Sắp xếp các lớp học, phòng học, và giáo viên một cách hiệu quả.
- Lập lịch sản xuất: Sắp xếp thứ tự sản xuất sao cho tối ưu hóa thời gian và tài nguyên.

Câu 6: Tìm kiếm mù và tìm kiếm heuristic có ưu và nhược điểm gì?

Tìm kiếm mù và tìm kiếm heuristic đều là phương pháp tìm kiếm quan trọng trong AI, nhưng chúng có những ưu và nhược điểm riêng biệt:

Tìm kiếm mù:

Ưu điểm:

1. ***Đơn giản và dễ hiểu:*** Không yêu cầu kiến thức cụ thể về bài toán, chỉ cần tìm kiếm dựa trên cấu trúc của không gian trạng thái.
2. ***Đảm bảo:*** Trong trường hợp của BFS (Tìm kiếm theo chiều rộng), nếu có giải pháp, giải thuật sẽ tìm ra giải pháp tối ưu.
3. ***Khái quát:*** Có thể áp dụng cho nhiều bài toán khác nhau mà không cần điều chỉnh nhiều.

Nhược điểm:

1. ***Không gian và thời gian tìm kiếm lớn***: Đối với các bài toán có không gian trạng thái lớn, tìm kiếm mù có thể trở nên không hiệu quả vì nó tìm kiếm qua tất cả các trạng thái.
2. ***Không luôn tối ưu:*** Trong trường hợp của DFS (Tìm kiếm theo chiều sâu), giải thuật có thể tìm ra một giải pháp nhưng không nhất thiết phải là tối ưu.

Tìm kiếm heuristic:

Ưu điểm:

1. ***Hiệu quả:*** Với một heuristic tốt, giải thuật có thể hướng dẫn tìm kiếm đến giải pháp một cách nhanh chóng, giảm thiểu không gian và thời gian tìm kiếm.
2. ***Tối ưu:*** Một số giải thuật như A* đảm bảo tìm ra giải pháp tối ưu nếu sử dụng heuristic không chệch.
3. ***Linh hoạt***: Heuristic có thể được thiết kế dựa trên kiến thức về bài toán, cho phép tận dụng thông tin đặc trưng của bài toán.

Nhược điểm:

1. ***Phụ thuộc vào heuristic***: Hiệu quả của giải thuật phụ thuộc nhiều vào việc chọn lựa heuristic. Một heuristic xấu có thể làm giảm hiệu suất tìm kiếm.

2. ***Không đảm bảo tối ưu***: Trong trường hợp của một số giải thuật như Greedy Search, giải thuật chỉ tìm một giải pháp chấp nhận được chứ không phải là tối ưu.
3. ***Phức tạp***: Việc thiết kế một heuristic tốt đòi hỏi sự hiểu biết sâu rộng về bài toán và có thể mất nhiều thời gian.

Câu 7: Tại sao Beam search và Greedy search được sử dụng rộng rãi trong bài toán xử lý ngôn ngữ tự nhiên NLP?

Beam search và Greedy search được sử dụng rộng rãi trong các bài toán Xử lý ngôn ngữ tự nhiên (NLP) vì các lý do sau đây:

Beam Search

1. ***Hiệu suất và Hiệu quả***: Beam search là một giải thuật heuristic hiệu quả trong việc giảm bớt không gian tìm kiếm mà vẫn có khả năng tìm kiếm các giải pháp gần đúng. Điều này là quan trọng trong NLP, nơi không gian trạng thái có thể rất lớn.
2. ***Tối ưu hóa chuỗi***: NLP thường đòi hỏi việc tối ưu hóa các chuỗi (ví dụ: câu, đoạn văn), và Beam search rất hiệu quả trong việc này.
3. ***Linh hoạt***: Beam search có thể được điều chỉnh (ví dụ, bằng cách thay đổi kích thước của "beam") để cân nhắc giữa độ phức tạp tính toán và chất lượng của giải pháp.

Greedy Search

1. ***Tính toán nhanh***: Greedy search rất nhanh và không đòi hỏi nhiều tài nguyên tính toán. Trong các ứng dụng thời gian thực của NLP, điều này có thể rất quan trọng.
2. ***Đơn giản và dễ triển khai***: Greedy search đơn giản cả về ý tưởng và triển khai. Điều này giúp nó trở thành lựa chọn tốt cho các ứng dụng cần giải pháp nhanh chóng và đơn giản.
3. ***Tốt cho các bài toán có giải pháp gần đúng:*** Trong một số trường hợp, việc tìm giải pháp tối ưu không cần thiết, và một giải pháp "tốt đủ" là đủ. Greedy search phù hợp trong các tình huống như vậy.

Cả hai giải thuật đều có ưu điểm riêng khi đối mặt với các bài toán NLP:

- Beam search thích hợp cho các bài toán cần giải pháp chất lượng cao và có thể đánh đổi thời gian tính toán.

- Greedy search hữu ích khi tài nguyên tính toán bị hạn chế hoặc khi cần có một giải pháp nhanh chóng.

Chính vì các lý do trên, Beam search và Greedy search được sử dụng rộng rãi trong các bài toán NLP.

Câu 8: Biểu diễn tri thức bằng logic có vai trò như thế nào trong trí tuệ nhân tạo?

Biểu diễn tri thức (Knowledge Representation) đóng một vai trò quan trọng trong trí tuệ nhân tạo (AI) vì nó là cầu nối giữa dữ liệu thô và các hệ thống có khả năng suy luận và học.

- Cung cấp cấu trúc cho dữ liệu: Biểu diễn tri thức giúp tổ chức dữ liệu một cách có ý nghĩa, giúp hệ thống dễ dàng truy cập và sử dụng chúng.
- Tạo điều kiện cho suy luận: Một biểu diễn tri thức tốt giúp hệ thống dễ dàng thực hiện suy luận, làm cho hệ thống có khả năng "hiểu" và "nghĩ" về dữ liệu, không chỉ là xử lý nó.
- Quản lý ngữ nghĩa: Biểu diễn tri thức cung cấp một phương tiện để mã hóa ý nghĩa của dữ liệu, điều này là rất quan trọng trong nhiều ứng dụng của AI như xử lý ngôn ngữ tự nhiên, học máy, và hệ thống đề xuất.
- Hỗ trợ tìm kiếm và tối ưu hóa: Khi tri thức được biểu diễn một cách có cấu trúc, các thuật toán tìm kiếm và tối ưu hóa có thể hoạt động hiệu quả hơn.
- Học và cập nhật: Một biểu diễn tri thức tốt cũng phải cho phép hệ thống cập nhật và học từ dữ liệu mới một cách dễ dàng.
- Tính di động và linh hoạt: Biểu diễn tri thức giúp hệ thống có khả năng xử lý các tình huống mới mà không cần phải được lập trình cụ thể cho chúng.
- Giao tiếp giữa máy và người: Biểu diễn tri thức giúp tạo ra một ngữ cảnh chung giữa con người và máy, làm cho việc giao tiếp và hiểu biết qua lại trở nên dễ dàng hơn.
- Tích hợp và kết hợp tri thức: Các hệ thống AI thường cần phải kết hợp tri thức từ nhiều nguồn khác nhau, và biểu diễn tri thức giúp việc này trở nên khả thi.
- Giảm thiểu độ phức tạp: Biểu diễn tri thức giúp tóm tắt và tổ chức dữ liệu một cách hiệu quả, giảm thiểu độ phức tạp và làm cho việc xử lý dữ liệu trở nên dễ dàng hơn.

Câu 9: Logic vị từ cấp 1 được sử dụng để biểu diễn tri thức như thế nào?

Logic vị từ cấp 1 (First-Order Predicate Logic) là một công cụ mạnh mẽ để biểu diễn tri thức trong trí tuệ nhân tạo. Nó cho phép chúng ta biểu diễn các đối tượng, các thuộc tính của đối tượng, và các quan hệ giữa các đối tượng một cách rõ ràng và có cấu trúc. Dưới đây là các cách cơ bản mà logic vị từ cấp 1 được sử dụng để biểu diễn tri thức:

Biểu diễn Đối tượng và Thuộc tính

- Đối tượng: Các đối tượng cụ thể như "John", "Apple", "Paris" có thể được biểu diễn.
- Thuộc tính: Các thuộc tính của đối tượng có thể được biểu diễn dưới dạng vị từ. Ví dụ, Human(John)Human(John) biểu diễn rằng "John là một con người".

Biểu diễn Quan hệ

- Các quan hệ giữa các đối tượng có thể được biểu diễn dễ dàng. Ví dụ, Loves(John,Mary)Loves(John,Mary) biểu diễn rằng "John yêu Mary".

Biểu diễn Hành động và Sự kiện

- Các hành động và sự kiện cũng có thể được biểu diễn. Ví dụ, Move(John,Paris)Move(John,Paris) có thể biểu diễn rằng "John di chuyển đến Paris".

Sử dụng Biến và Quantifiers

- Logic vị từ cấp 1 cho phép sử dụng các biến và quantifiers như "forall" (∀)(∀) và "exists" (∃)(∃) để biểu diễn các khái niệm phổ quát.
  - ∀x,Human(x)→Mortal(x)∀x,Human(x)→Mortal(x): "Mọi con người đều phải chết".
  - ∃x,Apple(x)∧Red(x)∃x,Apple(x)∧Red(x): "Có ít nhất một quả táo đỏ".

Biểu diễn Câu Điều kiện và Ràng buộc

- Câu điều kiện và ràng buộc có thể được biểu diễn dễ dàng, thích hợp cho việc suy luận.
  - Father(John,Mary)→Parent(John,Mary)Father(John,Mary)→Parent(John,Mary): "Nếu John là cha của Mary, thì John là phụ huynh của Mary".

Tích hợp và Kết hợp Tri Thức

- Logic vị từ cấp 1 cho phép tích hợp và kết hợp tri thức từ nhiều nguồn và trong nhiều dạng khác nhau, từ số đến từ ngữ.

Câu 10: Luật suy diễn có vai trò và ứng dụng như thế nào trong biểu diễn tri thức?

Luật suy diễn đóng một vai trò quan trọng trong việc biểu diễn và sử dụng tri thức trong trí tuệ nhân tạo. Các luật suy diễn cung cấp các quy tắc để hệ thống có thể rút ra các kết luận mới từ cơ sở tri thức hiện có. Dưới đây là một số vai trò và ứng dụng của luật suy diễn trong biểu diễn tri thức:

1. Kích hoạt Suy Luận Lôgic

- Luật suy diễn là cơ sở cho việc suy luận lôgic, giúp hệ thống có khả năng "hiểu" và "nghĩ" về dữ liệu một cách có lôgic.

2. Tạo điều kiện cho Tự động hóa

- Các luật suy diễn cho phép hệ thống tự động rút ra các kết luận mới mà không cần can thiệp từ người dùng.

3. Kiểm tra Tính Nhất Quán và Tính Đúng Đắn

- Sử dụng các luật suy diễn để kiểm tra tính nhất quán của cơ sở tri thức, và xác định xem các quy định hay các phát biểu mới có mâu thuẫn với tri thức hiện có hay không.

4. Tối ưu hóa và Cải tiến hiệu suất

- Các luật suy diễn có thể giúp tối ưu hóa quá trình tìm kiếm thông tin, làm cho việc truy vấn dữ liệu trở nên nhanh chóng và hiệu quả hơn.

5. Hỗ trợ trong Quyết Định và Dự đoán

- Các luật suy diễn có thể được sử dụng để hỗ trợ trong việc ra quyết định hoặc dự đoán các sự kiện trong tương lai dựa trên tri thức hiện có.

6. Cập nhật và Học

- Khi có tri thức mới, các luật suy diễn giúp cập nhật cơ sở tri thức một cách nhất quán và có lôgic.

7. Xây dựng và Hiệu chỉnh Mô hình

- Trong quá trình phát triển hệ thống, các luật suy diễn có thể giúp định rõ các ràng buộc và quy tắc, giúp cho việc xây dựng và hiệu chỉnh mô hình trở nên dễ dàng hơn.

Câu 11: Ưu nhược điểm của lưới ngữ nghĩa trong biểu diễn tri thức là gì?

Lưới ngữ nghĩa (Semantic Networks) là một công cụ biểu diễn tri thức mạnh mẽ, nhưng cũng có các ưu và nhược điểm cần xem xét.

Ưu điểm:

1. **Dễ Hiểu**: Cấu trúc đồ thị của lưới ngữ nghĩa giúp cho việc hiểu và diễn giải tri thức trở nên dễ dàng hơn.
2. **Tính Linh Hoạt**: Có thể dễ dàng thêm, xóa hoặc sửa đổi các đối tượng và quan hệ, giúp cập nhật và mở rộng cơ sở tri thức.
3. **Hỗ trợ Suy luận**: Lưới ngữ nghĩa có thể được sử dụng để hỗ trợ các quá trình suy luận, như việc tìm các quan hệ gián tiếp giữa các đối tượng.
4. **Tính Sắc thái Ngữ nghĩa**: Cung cấp một cách để biểu diễn các khái niệm và quan hệ có tính sắc thái ngữ nghĩa, như là "đối tượng A là một phần của đối tượng B" hoặc "đối tượng X kế thừa từ đối tượng Y".
5. **Tính Mô-đun hóa**: Các phần của lưới có thể được mô-đun hóa và tái sử dụng trong các cấu trúc tri thức khác nhau.
6. **Khả năng Tương tác**: Phù hợp cho các ứng dụng cần tương tác người - máy, như các hệ thống đề xuất hoặc các công cụ tìm kiếm thông minh.

Nhược điểm:

1. **Tính Mơ hồ và Không chính xác**: Lưới ngữ nghĩa có thể không đủ chính xác trong việc biểu diễn các khái niệm và quan hệ phức tạp hoặc mơ hồ.
2. **Khó Quản lý**: Khi cơ sở tri thức trở nên lớn và phức tạp, việc quản lý và duy trì lưới ngữ nghĩa có thể trở nên khó khăn.
3. **Thiếu Cơ chế Suy luận Tích hợp**: Mặc dù hỗ trợ suy luận đến một mức độ nào đó, lưới ngữ nghĩa thường không có cơ chế suy luận tích hợp mạnh mẽ như logic vị từ cấp 1.
4. **Rủi ro về Tính Nhất Quán**: Khi cơ sở tri thức phát triển, việc duy trì tính nhất quán trong lưới ngữ nghĩa có thể trở nên thách thức.
5. **Không biểu diễn được Thông tin Động**: Lưới ngữ nghĩa thường tập trung vào tri thức tĩnh và có thể không phản ánh hiệu quả sự thay đổi của thông tin theo thời gian.
6. **Tính Phân cấp Hạn chế**: Mặc dù lưới ngữ nghĩa có thể biểu diễn các quan hệ phân cấp, nhưng chúng có thể không hiệu quả trong việc biểu diễn các quan hệ phức tạp không phải là phân cấp.

Câu 12: Trường hợp nào nên sử dụng hệ khung để biểu diễn tri thức?

Hệ khung (Frames) là một công cụ rất mạnh mẽ để biểu diễn tri thức, và có nhiều trường hợp mà việc sử dụng hệ khung có thể là lựa chọn tốt nhất:

1. Tri Thức Phức Tạp và Đa Chiều

- Khi bạn cần biểu diễn tri thức có nhiều thuộc tính, quan hệ và hành vi, hệ khung là một lựa chọn tốt. Hệ khung có thể chứa nhiều thông tin trong cùng một cấu trúc, từ các thuộc tính đơn giản đến các quy tắc và hành vi phức tạp.

2. Mô phỏng và Mô hình hóa

- Trong các ứng dụng mô phỏng và mô hình hóa, hệ khung có thể giúp bạn biểu diễn các đối tượng và quá trình dễ dàng và có cấu trúc.

3. Hệ thống Chuyên gia

- Hệ khung rất hữu ích khi xây dựng các hệ thống chuyên gia có cơ sở tri thức phức tạp và cần được biểu diễn dưới dạng các đối tượng với nhiều thuộc tính và quan hệ.

4. Xử lý Ngôn ngữ Tự nhiên (NLP)

- Khi cần biểu diễn tri thức ngữ nghĩa và cú pháp trong xử lý ngôn ngữ tự nhiên, hệ khung có thể giúp biểu diễn các đối tượng và quan hệ một cách chi tiết.

5. Hệ thống Đề xuất và Tìm kiếm thông tin

- Trong các ứng dụng cần đề xuất thông tin dựa trên các thuộc tính và quan hệ đa dạng giữa các đối tượng, việc sử dụng hệ khung có thể là lựa chọn tốt.

6. Tự động hóa và Hành vi

- Khi bạn cần mô tả không chỉ các thuộc tính và quan hệ của các đối tượng, mà còn cả cách thức hệ thống tương tác với chúng, hệ khung có thể giúp bạn định rõ các quy tắc và hành vi.

7. Các Ứng dụng Có tính Tương tác Cao

- Hệ khung rất hữu ích trong các ứng dụng có tính tương tác cao với người dùng, như các hệ thống trợ giúp thông minh, bởi vì chúng cho phép biểu diễn tri thức dễ dàng và có cấu trúc.

8. Các Hệ thống Tích hợp Tri thức

- Khi có nhu cầu tích hợp tri thức từ nhiều nguồn và trong nhiều dạng, hệ khung có khả năng kết hợp và tổ chức tri thức một cách linh hoạt.

Câu 13: Trong trò chơi cờ caro đã sử dụng giải thuật Minimax, cắt tỉa Anpha-Beta, hàm tính toán heuristics. Hãy chỉ ra vai trò cụ thể của các giải thuật đó.

Trò chơi cờ Caro (còn được gọi là Gomoku) là một trò chơi trên bảng lưới, thường là 15x15 hoặc 19x19, trong đó hai người chơi luân phiên đặt các quân cờ của mình

(thường là quân đen và quân trắng) với mục tiêu là tạo ra một chuỗi liên tiếp gồm 5 quân cờ theo chiều ngang, dọc, hoặc chéo.

1. Thuật toán Minimax

Minimax là một thuật toán quyết định đối tác cổ điển được sử dụng trong các trò chơi có hai người chơi. Thuật toán này thăm các nút của cây trò chơi để đánh giá các điểm đi tốt nhất cho người chơi.

2. Cắt Tỉa Alpha-Beta

Cắt tỉa Alpha-Beta là một kỹ thuật cải tiến thuật toán Minimax bằng cách loại bỏ các nhánh không cần thiết trong cây trò chơi, giúp tăng hiệu suất.

3. Heuristic Evaluation

Để giúp thuật toán Minimax hoặc Alpha-Beta nhanh hơn, có thể sử dụng các hàm heuristic để đánh giá giá trị của các bảng cờ. Các hàm này có thể tính toán dựa trên số chuỗi cờ có tiềm năng, số lượng "mắt" cờ bị chặn, v.v.

Câu 14: Anh (chị) cho biết giải thuật cốt lõi của ứng dụng tìm kiếm Google map?

Google Maps sử dụng nhiều thuật toán và công nghệ để cung cấp dịch vụ tìm đường và dẫn đường, nhưng một trong những thuật toán cốt lõi thường được đề cập là A* (A-Star). Thuật toán A* là một thuật toán tìm kiếm đường đi trong đồ thị, và nó là một biến thể của thuật toán Dijkstra. Tuy nhiên, A* có hiệu suất tốt hơn do sử dụng một hàm heuristic để ước tính chi phí từ một đỉnh đến đích, giúp tập trung tìm kiếm vào các đỉnh có khả năng cao dẫn đến lời giải tốt nhất.

Câu 15: Điểm kỳ dị của công nghệ ("The Technological Singularity") là gì? Cho biết ý kiến, nhận định của anh (chị) về vấn đề này?

Điểm kỳ dị của công nghệ (hay "The Technological Singularity") là một ý tưởng hoặc dự đoán rằng sẽ có một thời điểm trong tương lai khi sự tiến bộ của công nghệ sẽ trở nên nhanh đến mức không thể đoán trước hay kiểm soát. Nói cách khác, đây là một thời điểm mà các máy, mạng nơ-ron, hoặc thuật toán AI sẽ trở nên thông minh đến mức có khả năng tự cải tiến và phát triển một cách độc lập, không cần sự can thiệp của con người.

TÀI LIỆU THAM KHẢO

[1]. Đinh Mạnh Tường, *Trí tuệ nhân tạo*, NXB Khoa học kỹ thuật, 2002

[2]. Nguyễn Thanh Thủy, Trí tuệ nhân tạo, NXB Giáo dục,1995

[3]. Đỗ Trung Tuấn, *Trí tuệ nhân tạo*, NXB Giáo dục,1998

[4]. J.Nilsson, *The question for Artificial Intelligence,* Cambridge University Press, 2009

[5]. S.Rusell; P.Norvig, *Artificial Intelligence – A modern approach,* Prentice Hall, 2003