MỤC LỤC

BÀI 1: KHÁI QUÁT VỀ LOGÍC 3
1. Giới thiệu: 3
2. Định nghĩa logic học: 3
3. Sự hình thành và phát triển của logic học: 3
4. Ứng dụng của logic học: 4
5. Đôi nét về logic mờ 5
BÀI 2: LOGÍC MỆNH ĐỀ 9
1. Định nghĩa : 10
2. Phân tích : 10
3. Các phép toán logic cơ bản : 11
Bảng chân trị 12
4. Công thức trong đại số logic : 12
4.1/ Công thức : 12
4.2/ Công thức tương đương : 13
4.3/ Các qui tắc thay thế: 14
5. Hệ quả logic và tương đương logic: 17
6. Công thức đối ngẫu 17
7. Tính đầy đủ của một hệ các phép toán 17
7. Ứng dụng logic mệnh đề để vẽ mạch điện tử 18
BÀI 3: LOGÍC TÍNH TOÁN 24
1. Khái niệm: 24
1.1/ Dạng tuyển chuẩn: 24
1.2/ Dạng hội chuẩn: 24
2. Số logic : 25
2.1/ Định nghĩa : 25
2.2 Hàm logic: 26
2.2 Tương đương logic: 28
3. Thuật toán biểu diễn một công thức logic dưới dạng tuyển chuẩn: 28
4. Thuật toán biểu diễn một công thức logic dưới dạng hội chuẩn: 30
Bài 4: CÀI ĐẶT MIả H HỌA 33
1. Thuật toán tính số logic của một công thức: 33
2. Chương trình minh họa việc kiểm tra 2 công thức tương đương: 39
BÀI 5: SUY DIễả LOGIC VÀ VỊ TỪ 42
1. Giới thiệu: 42
2. Định nghĩa qui tắc suy diễn: 42
3. Kiểm tra một qui tắc suy diễn: 44
4. Các qui tắc suy diễn cơ bản: 45
5. Các ví dụ áp dụng trong suy luận và chứng minh 48
6. Định nghĩa vị từ và ví dụ 50
6.1/ Định nghĩa: 50
6.2/ Các phép toán trên các vị từ 50

6.3/ Qui tắc phủ định mệnh đề có lượng từ 51
6.4/ Một số qui tắc dùng trong suy luận: 53
BÀI 6: Ị GÔI Ị GỮ PROLOG 58
1. Tư duy lập trình và định nghĩa vấn đề trên Prolog 58
2. Các clause, cách giải thích các vấn đề trên Prolog 60
3. Thực thi chương trình. - Đặt câu hỏi và nhận câu trả lời 62
4. Phép hợp nhất - Cơ chế tìm câu trả lời của Prolog. 65
4.1/ Phép hợp nhất 65
4.2/ Cơ chế tìm câu trả lời của Prolog 66
5. Sự quay lui - Khống chế số lượng lời giải -Vị từ nhát cắt và fail 68
6. Lập trình đệ quy với Prolog 72
7. Danh sách trên Prolog 74
8. Lập trình đệ quy với danh sách trên Prolog 75
9. Danh sách hai chiều 78
BÀI 7: LOGIC MỜ 81
1. Một số khái niệm 81
1.1/ Tập mờ (Fuzzy Sets) 81
1.2/ Số mờ (Fuzzy Ị umbers) 85
1.3/ Số logic dạng hình khối 86
1.4/ Số logic dạng tam giác 87
1.5/ Số logic dạng hình thang 89
2. Áp dụng của logic mờ trong dự đoán 90
2.1/ Giá trị trung bình trong thống kê 90
2.2/ Các phép toán với số tam giác vá số hình thang 91
2.3/ Trung bình trong logic mờ 93
2.4/ Dự đoán bằng phương pháp Delphi kết hợp logic mờ 95
2.5/ Phương pháp Fuzzy Delphi có trọng số: 99
2.6/ Ứng dụng Fuzzy Pert trong việc quản lý các đề án 100
ĐỀ TÀI CỘI G ĐIỂM CUỐI KỲ 110
TÀI LIỆU THAM KHẢO 111

BÀI 1: KHÁI QUÁT VỀ LOGÍC

1. Giới thiệu:

Logic là khoa học xuất hiện rất sớm trong lịch sử. Ị ó xuất hiện vào thế kỷ thứ IV trước công nguyên khi sự phát triển của khoa học nói riêng và tư duy nói chung đã đòi hỏi phải trả lời câu hỏi: làm thế nào để đảm bảo suy ra được kết luận đúng đắn, chân thực từ các tiền đề chân thực?

2. Định nghĩa logic học:

Từ "Logic" có nguồn gốc từ Hy lạp là từ "Logos", từ này có rất nhiều nghĩa trong đó có hai nghĩa thường dùng nhất là:

- ❖ Chỉ tính qui luật của sự tồn tại và phát triển của thế giới khách quan.
- ❖ Chỉ những qui luật đặc thù của tư duy.

Khi ta nói "trái đất quay quanh mặt trời", ta đã sử dụng nghĩa thứ nhất. Còn khi nói "anh ấy suy luận hợp logic", ta đã sử dụng nghĩa thứ hai.

3. Sự hình thành và phát triển của logic học:

Ị gười sáng lập ra logic học là nhà triết học Hy lạp vĩ đại Aristote(384-322 Tr.CỊ).

Ở thời cổ đại, logic học của Aristote được các học trò của ông tiếp tục phát triển sau khi ông mất nhưng học chỉ nêu ra một số qui tắc suy luận vớI tiền đề là phán đoán điều kiện và phán đoán lựa chọn nghiêm ngặt mà thôi. Các nhà triết học thuộc trường phái Megat và trường phái khắc kỷ đi xa hơn: họ nghiên cứu các quan hệ suy diễn. Để nghiên cứu vấn đề này, họ đưa ra quan hệ ***bao hàm*** (implication) và họ cũng đưa ra hình thức đầu tiên của định lý diễn dịch - định lý làm cơ sở cho các phép chứng minh trong các hệ thống hình thức hóa: một suy luận được gọi là hợp logic khi và chỉ khi công thức biểu thị nó là một công thức hằng đúng.

Các thành tựu quan trọng nhất ở thời La mã cổ đại là: hệ thống các thuật ngữ logic được sử dụng đến ngày nay: hình vuông logic (sau này được Boethius hoàn thiện); lý thuyết về tam đoạn luận phức hợp và tam đoạn luận với tiền đề là phán đoán quan hệ.

Vào thời phục hưng, logic học truyền thống bị chỉ trích mạnh mẽ. Một số nhà tư tưởng tiến bộ của thời kỳ này buộc tội logic là chỗ dựa cho tư tưởng kinh viện. Ị hà triết học người

Anh F. Bacon (1561-1626) cho rằng tam đoạn luận của Aristote hoàn toàn vô ích vì nó không cho phép tìm ra các thông tin mới từ các tiền đề đã có. Vậy nên khoa học sử dụng nó không thể phát hiện các qui luật mới thông qua việc nghiên cứu các sự kiện thực nghiệm đã biết. Ông xây dựng nên logic qui nạp mà về sau được một nhà triết học và logic học Anh khác là S.Mill (1806 – 1873) phát triển.

Về phần logic diễn dịch thì mãi đến thế kỷ XVII mới được nhà toán học và triết học người Pháp R.Descates (1596 – 1650) thanh minh và bảo vệ. Ông muốn xây dựng nó thành phương pháp nhận thức tổng hợp. Công lao rất lớn trong việc phát triển logic diễn dịch vẫn thuộc về nhà toán học và logic học người Đức Leibniz (1646 – 1716). Ông được coi là người đầu tiên đặt nền tảng cho logic kí hiệu. Ông đưa ra tư tưởng sử dụng các kí hiệu và phương pháp toán học vào logic. Ông chỉ ra rằng khi sử dụng các kí hiệu thay cho lời nói, không những chúng ta làm cho tư tưởng được trở nên rõ ràng hơn, chính xác hơn mà còn làm cho tư tưởng trở nên đơn giản hơn. Ông muốn xây dựng logic học thành phép tính (calculus rationator) – ngôn ngữ nhân tạo tổng quát trong đó các suy luận được hình thức hoá giống như các phép tính được hình thức hoá trong đại số. Tư tưởng của Leibniz về sau được các nhà toán học và logic học J. Boole (1815 – 1864) và De Morgan phát triển, họ đã xây dựng các hệ đại số logic.

Sự phát triển của logic hình thức trong thời hiện đại gắn liền với các tên tuổi của các nhà bác học lớn như G.Frege (1848 – 1925), Peano (1858 – 1932), B.Russel (1872 – 1970),…. Quá trình phát triển của logic học kể từ thời Leibniz và đặc biệt là từ Russel trở về sau liên quan rất chặt chẽ đến toán học. Ị gày nay, logic hình thức bao gồm nhiều nhánh khác nhau như logic cổ điển, logic tình thái, logic thời gian, logic kiến thiết, logic relevant, logic không đơn điệu, logic mờ,…

4. Ứng dụng của logic học:

Cùng với sự phát triển của khoa học và công nghệ, logic học ngày càng được ứng dụng rộng rãi. Logic giúp giải quyết các vấn đề nan giải của toán học, của điều khiển học, của nhiều vấn đề trong khoa học máy tính…Ị gười ta sử dụng logic vị từ để làm các ngôn ngữ lập trình cho trí tuệ nhân tạo (ví dụ ngôn ngữ PROLOG); ứng dụng logic mờ (Fuzzy logic) để phát triển công nghệ mờ…

5. Đôi nét về logic mờ

Ị gày nay khi nhìn lại lịch sử của logic mờ, người ta nhận thấy người đầu tiên đề cập tới logic mờ chính là Đức Phật (500 năm trước CỊ). Triết lý Phật giáo dựa trên tư tưởng rằng thế giới đầy những mâu thuẫn, "sắc không không sắc", mọi thứ đề chứa một phần đối lập của nó. Bước chân vào mỗi ngôi chùa chúng ta đều thấy ở ngay gian trước là hai vị Thiện — Ác, là hình ảnh hai mặt tốt và xấu trong mỗi con người. Ị ói theo lý thuyết logic mờ nghĩa là sự vật có thể đồng thời là A và không-A. Ở đây ta thấy có một mối liên hệ rõ ràng giữa triết lý Phật giáo và logic mờ hiện đại. Thuyết âm dương của người Trung Quốc cũng hàm chứa logic mờ! "Logo" bát quái thể hiện tư tưởng cốt yếu của thuyết: hình tròn thể hiện sự toàn vẹn của sự vật, trời đất; mỗi sự vật hiện tượng đều có hai mặt âm và dương đối lập nhau, cùng tồn tại, mặt này thịnh thì mặt kia suy (phần âm to ra thì phần dương nhỏ đi và ngược lại); dấu trắng trong phần đen và dấu đen trong phần trắng thể hiện trong âm có dương, trong dương có âm; dấu đen trong đầu to của phần trắng thể hiện khi dương cực thịnh thì chính là lúc trong lòng nó xuất hiện âm (và ngược lại).

Sau đức Phật 200 năm, nhà bác học Hy-lạp là Aristote phát triển logic nhị phân. Trái ngược với triết lý nhà Phật, Aristote cho rằng thế giới tạo bởi các đối nghịch, thí dụ nam-nữ, nóng-lạnh, khô-ướt. Mọi thứ hoặc là A hoặc là không-A, không thể cả hai. Logic nhị phân của Aristote trở thành nền tảng cho khoa học, nếu một thứ được chứng minh về mặt logic (nhị phân) thì nó được và vẫn sẽ được khoa học công nhận. Cho tới cuối thế kỷ 19, khi một nhà văn-nhà toán học người Anh, Russel, phát hiện ra một nghịch lý của logic nhị phân . . .

Russel (1872-1970), người khai sinh logic mờ

Bá tước Bertrand Arthur William Russel sinh ra trong một gia đình quý tộc Anh năm 1872. Ông có một cuộc đời dài và đầy biến động. Thời trẻ tuổi, ông nghiên cứu toán học và sau đó, cùng với một nhà toán học khác, viết một cuốn sách về những cơ sở của toán học. Trong sách, họ dành cả một trang chỉ để chứng minh 1 + 1 = 2. Trong quá trình nghiên cứu, ông đã phát hiện ra một nghịch lý mà ngày nay gọi là nghịch lý tập của Russell :

Trước hết chúng ta phân biệt hai loại tập: tập chứa chính nó và tập không chứa chính nó.

Xét thí dụ: một quả lê thuộc tập các quả lê, nhưng tập các quả lê không thuộc về tập các quả lê do bản thân nó không phải là một quả lê! Ị ghĩa là tập các quả lê không phải là một thành viên của chính nó.

Bây giờ ta xét một tập khác, tập mọi thứ không phải quả lê, gồm sách, chuột cống, hay cả tổng thống Bush nữa! Do trong tập này bạn tìm thấy mọi thứ không phải quả lê, nên bạn cũng có thể tìm thấy trong đó tập các quả lê và tập mọi thứ không phải quả lê ! Ị ghĩa là tập mọi thứ không phải quả lê là thành viên của chính nó.

Russel đi sâu hơn và xem xét tập của mọi tập mà không chứa chính nó. Trong tập này, bạn sẽ tìm thấy tập các quả lê, tập các tổng thống, và nhiều tập khác nữa. Ị hưng bạn sẽ không tìm thấy tập mọi thứ không phải quả lê, do tập đó chứa chính nó và do vậy không thoả mãn tiêu chuỊn đặt ra. Trong khi xem xét tập các tập không chứa chính nó này, Russell băn khoăn liệu nó có phải là một thành viên của chính nó?

Ị ếu nó là một thành viên của chính nó, thì không thoả mãn định nghĩa. Mặt khác, nếu nó không phải là thành viên của chính nó, thì theo định nghĩa về tập đó, thì nó lại thoả mãn và như vậy nó là thành viên của chính nó!

Vì vậy khi tìm ra nghịch lý này, Russell ngẫu nhiên chứng minh rằng logic nhị phân, mà ông nghĩ là cơ sở của toán học, không thể tự chứng minh nó. Tất nhiên ngày nay, chúng ta biết nghịch lý của Russell không phải là một trường hợp không giải được, nếu dùng logic mờ thì ta có câu trả lời ngay. Tuy nhiên, Russell không hề biết gì về logic mờ và đã vô cùng thất vọng với toán học. Ông từ bỏ toán học, những như thế không có nghĩa là ông đã dừng lại việc làm đảo lộn thế giới này. Trong suốt cuộc đời 97 năm, ông luôn truyền bá tư tưởng của mình; ông viết hàng tá sách, sách toán, triết luận, tiểu thuyết, thậm chí cả thứ sách lá cải nữa. Khi mất năm 1970, ông đã không chỉ khởi đầu một trang mới của logic học, mà còn đoạt cả một giải Ị obel văn học. Ông là một thí dụ điển hình cho thấy những người có tài năng lớn về toán học cũng có thể là những nhà văn lớn.

Zadeh, cha đẻ của logic mờ hiện đại.

Ị ăm 1964, giáo sư Zadeh bắt đầu suy nghĩ liệu có thứ logic tốt hơn nào dùng trong máy móc. Ông có ý tưởng liệu ta có thể bảo máy điều hoà làm việc nhanh hơn khi trời nóng lên, hay những vấn đề tương tự như thế, sẽ hiệu quả hơn việc đặt ra từng luật cho từng nhiệt

độ. Đây chính là bước đi đầu tiên của logic mờ hiện đại như chúng ta hiểu và ứng dụng ngày nay.

Phải mất một thời gian dài logic mờ mới được chấp nhận, mặc dù ngay từ đầu một số người đã rất quan tâm. Bên cạnh các kỹ sư, những nhà triết học, tâm lý học và xã hội học nhanh chóng áp dụng logic mờ vào ngành khoa học của mình.

Ị ăm 1987, Ị hật Bản đã xây dựng hệ thống tàu điện ngầm đầu tiên làm việc với hệ thống điều khiển hoạt động tàu tự động dựa trên logic mờ. Đây là một thành công lớn và dẫn tới sự phát triển bùng nổ của logic mờ. Các trường đại học và các hãng công nghiệp đua nhau phát triển những ý tưởng mới. Đầu tiên là ở Ị hật Bản, do tôn giáo ở Ị hật thừa nhận rằng mọi thứ có thể chứa phần đối lập của chính nó, chứ không coi là một thứ "kinh khủng" như hầu hết những nơi khác trên thế giới. Và logic mờ cũng hứa hẹn đem lại nhiều tiền bạc cho các hãng công nghiệp, tất nhiên là điều này được đón chào.

Logic mờ được công bố lần đầu tiên tại Mỹ vào năm 1965 do giáo sư Lotfi Zadeh. Kể từ đó, logic mờ đã có nhiều phát triển qua các chặng đường sau : phát minh ở Mỹ, áp dụng ở Châu Âu và đưa vào các sản phẹm thương mại ở Ị hật.

Ứng dụng đầu tiên của logic mờ vào công nghiệp được thực hiện ở Châu âu, khoảng sau năm 1970. Tại trường Queen Mary ở Luân Đôn – Anh, Ebrahim Mamdani dùng logic mờ để điều khiển một máy hơi nước mà trước đây ông ấy không thể điều khiển được bằng các kỹ thuật cổ điển. Và tại Đức, Hans Zimmermann dùng logic mờ cho các hệ ra quyết định. Liên tiếp sau đó, logic mờ được áp dụng vào các lĩnh vực khác như điều khiển lò xi măng, … nhưng vẫn không được chấp nhận rộng rãi trong công nghiệp.

Kể từ năm 1980, logic mờ đạt được nhiều thành công trong các ứng dụng ra quyết định và phân tích dữ liệu ở Châu âu. Ị hiều kỹ thuật logic mờ cao cấp được nghiên cứu và phát triển trong lĩnh vực này.

Cảm hứng từ những ứng dụng của Châu Âu, các công ty của Ị hật bắt đầu dùng logic mờ vào kỹ thuật điều khiển từ năm 1980. Ị hưng do các phần cứng chuẹn tính toán theo giải thuật logic mờ rất kém nên hầu hết các ứng dụng đều dùng các phần cứng chuyên về logic mờ. Một trong những ứng dụng dùng logic mờ đầu tiên tại đây là nhà máy xử lý nước của Fuji Electric vào năm 1983, hệ thống xe điện ngầm của Hitachi vào năm 1987.

Ị hững thành công đầu tiên đã tạo ra nhiều quan tâm ở Ị hật. Có nhiều lý do để giải thích tại sao logic mờ được ưa chuộng. Thứ nhất, các kỹ sư Ị hật thường bắt đầu từ những giải pháp đơn giản, sau đó mới đi sâu vào vấn đề. Phù hợp với việc logic mờ cho phép tạo nhanh các bản mẫu rồi tiến đến việc tối ưu. Thứ hai, các hệ dùng logic mờ đơn giản và dễ hiểu. Sự "thông minh" của hệ không nằm trong các hệ phương trình vi phân hay mã nguồn. Cũng như việc các kỹ sư Ị hật thường làm việc theo tổ, đòi hỏi phải có một giải pháp để mọi người trong tổ đều hiểu được hành vi của hệ thống, cùng chia sẽ ý tưởng để tạo ra hệ. Logic mờ cung cấp cho họ một phương tiện rất minh bạch để thiết kế hệ thống. Và cũng do nền văn hóa, người Ị hật không quan tâm đến logic Boolean hay logic mờ; cũng như trong tiếng Ị hật , từ "mờ' không mang nghĩa tiêu cực.

Do đó, logic mờ được dùng nhiều trong các ứng dụng thuộc lĩnh vực điều khiển thông minh hay xử lý dữ liệu. Máy quay phim và máy chụp hình dùng logic mờ để chứa đựng sự chuyên môn của người nghệ sĩ nhiếp ảnh. Misubishi thông báo về chiếc xe đầu tiên trên thế giới dùng logic mờ trong điều khiển, cũng như nhiều hãng chế tạo xe khác của Ị hật dùng logic mờ trong một số thành phần. Trong lĩnh vực tự động hóa, Omron Corp. có khoảng 350 bằng phát minh về logic mờ. Ị goài ra, logic mờ cũng được dùng để tối ưu nhiều quá trình hóa học và sinh học.

Ị ăm năm trôi qua, các tổ hợp Châu âu nhận ra rằng mình đã mất một kỹ thuật chủ chốt vào tay người Ị hật và từ đó họ đã nỗ lực hơn trong việc dùng logic mờ vào các ứng dụng của mình. Đến nay, có khoảng 200 sản phỊm bán trên thị trường và vô số ứng dụng trong điều khiển quá trình – tự động hóa dùng logic mờ.

BÀI 2: LOGÍC MỆNH ĐỀ

Trong đời sống hàng ngày, người ta cần có những lý luận để từ các điều kiện được biết hay được giả định (các tiền đề - premises) có thể suy ra các kết luận (conclusion) đúng. Hãy xét 2 lý luận sau :

- **Lý luận (1)** : Các tiền đề :

  + Ị ếu hôm nay trời đẹp thì tôi đi chơi.

  + Ị ếu tôi đi chơi thì hôm nay về trễ .

  Giả thiết : Hôm nay trời đẹp .

  Kết luận : Hôm nay tôi sẽ về trễ .

- **Lý luận (2)** : Các tiên đề :

  + Ị ếu hôm nay rạp hát không đóng cửa thi tôi sẽ xem phim.

  + Ị ếu tôi xem phim thì tôi sẽ không soạn kịp bài .

  Giả thiết : Hôm nay rạp hát không đóng cửa .

  Kết luận : Hôm nay tôi sẽ không soạn kịp bài.

Hai lý luận trên là đúng và có cùng dạng lý luận. Chúng đúng vì có dạng lý luận đúng, bất kể ý nghĩa mà chúng đề cập đến.

Còn lý luận sau :

- **Lý luận (3)** : Các tiền đề :

  + Ị ếu trời đẹp thì tôi đi chơi.

  + Ị ếu tôi đi chơi thì tôi sẽ về trễ.

  Giả thiết : Hôm nay tôi về trễ.

  Kết luận : Hôm nay trời đẹp .

  Là lý luận sai và mọi lý luận dạng như vậy đều sai .

Logic toán học quan tâm đến việc phân tích các câu (sentences), các mệnh đề (propositions) và chứng minh với sự chú ý đến dạng (form) lược bỏ đi sự việc cụ thể.

1. Định nghĩa :

- Một phán đoán là một suy nghĩ muốn khẳng định hay phủ định một điều gì đó có tính chính đúng hoặc sai mà không thể vừa đúng lại vừa sai.
- Mệnh đề toán học là diễn đạt phán đoán bằng một câu ngữ pháp.
- Mệnh đề đúng có giá trị chân lý là 1, mệnh đề sai có giá trị chân lý là 0.

Ví dụ :

4+3 > 2 là một mệnh đề có giá trị chân lý là 1
3+5 = 7 là một mệnh đề có giá trị chân lý là 0.

Các phát biểu sau đây không phải là các mệnh đề (toán học) vì tính đúng sai của chúng không xác định:

Ai đang đọc sách? (một câu hỏi)

Cho n là một số nguyên dương.

a là một số chính phương.

2. Phân tích :

Phân tích lý luận (1) ta thấy nó sử dụng các mệnh đề cơ sở sau :

- Hôm nay trời đẹp
- Tôi đi chơi
- Tôi sẽ về trễ.

Mỗi mệnh đề (proposition) là một phát biểu đúng (true) hay sai (false).

Biểu thị tượng trưng lần lượt các mệnh đề trên bởi các tên A, B, C, ta ghi lại dạng lý luận của (1) như sau :

Ị ếu A thì B (4)
Ị ếu B thì C
Có A kết luận được : C

Đây cũng là dạng lý luận của (2) .

- ❖ Thường một phát biểu sẻ gồm nhiều phát biểu nhỏ nối kết với nhau bằng các liên
- ❖ từ "và" , "hay" , "vì vậy " ,"kết quả là" ...
- ❖ Một mệnh đề đơn (simple proposition) là mệnh đề không chứa mệnh đề khác.

❖ Một mệnh đề phức (compound proposition) là mệnh đề được tạo thành từ hai hay nhiều mệnh đề đơn .Việc nối kết này được thực hiện bởi các liên từ logic.

Ví dụ : Xét các mệnh đề sau đây.

p = "15 chia hết cho 3".

q = "2 là một số nguyên tố và là một số lẻ".

Ta có p là một mệnh đề sơ cấp. Ị hưng q là một mệnh đề phức hợp, vì mệnh đề q được tạo thành từ hai mệnh đề "2 là một số nguyên tố" và "2 là một số lẻ" nhờ vào liên kết logic "và".

3. Các phép toán logic cơ bản :

Các phép toán logic được định nghĩa bằng ***bảng chân trị*** (truth table). Bảng chân trị chỉ ra rõ ràng chân trị của mệnh đề phức hợp theo từng trường hợp của các chân trị của các mệnh đề sơ cấp tạo thành mệnh đề phức hợp. Bảng chân trị của các phép toán logic tất nhiên là phản ánh ngữ nghĩa tự nhiên của các từ liên kết tương ứng. Về mặt tự nhiên của ngôn ngữ, trong nhiều trường hợp cùng một từ nhưng có thể có nghĩa khác nhau trong những ngữ cảnh khác nhau. Do đó, bảng chân trị không thể diễn đạt mọi nghĩa có thể có của từ tương ứng với ký hiệu phép toán. Điều nầy cho thấy rằng đại số logic là rõ ràng hoàn chỉnh theo nghĩa là nó cho ta một hệ thống logic đáng tin cậy. Đại số logic còn đặc biệt quan trọng trong việc thiết kế mạch cho máy tính.

Bảng chân trị không chỉ dùng để kê ra sự liên hệ chân trị giữa mệnh đề phức hợp với chân trị của các mệnh đề sơ cấp cấu thành nó, mà bảng chân trị còn được dùng với mục đích rộng hơn: liệt kê sự liên hệ chân trị giữa các mệnh với các mệnh đề đơn giản hơn cấu thành chúng.

Phép phủ định: $\neg$ (không)	Độ ưu tiên của các toán tử logic.
Phép hội: $\wedge$ (và)	$\neg$
Phép tuyển: $\vee$ (hay)	$\wedge$, $\vee$
Phép kéo theo: $\rightarrow$ (kéo theo)	$\rightarrow$, $\leftrightarrow$
Phép kéo theo 2 chiều: $\leftrightarrow$ (tương đương)	Các toán tử cùng dòng có cùng độ ưu tiên.

Ví dụ : $\neg p \vee q$ có nghĩa là $((\neg p) \vee q)$.

$\neg p \vee q \rightarrow r \wedge s$ có nghĩa là $(((\neg p) \vee q) \rightarrow (r \wedge s))$.

$\neg p \vee q \wedge r$ là không rõ ràng cần phải dùng các dấu ngoặc để chỉ rõ nghĩa.

Xét hai mệnh đề x và y, khi đó ta có:

Bảng chân tri của các phép toán mệnh đề

Mệnh đề p	Phủ đinh p	Mệnh đề p	Phép tuyển	Phép hội	Kéo theo	Tương đương
x	$\neg x$	y	$x \vee y$	$x \wedge y$	$x \rightarrow y$	$x \leftrightarrow y$
1	0	1	1	1	1	1
1	0	0	1	0	0	0
0	1	1	1	0	1	0
0	1	0	0	0	1	1

Ị goài ra ta còn có thêm một phép toán logic khác là "phép tuyển chọn" ứng với 2 mệnh đề sơ cấp p và q khác với phép tuyển đã cho ở trên được viết là p + q, hay $p \oplus q$ hay có thể biểu diễn như sau:

Bảng chân tri

x	y	x $\underline{\vee}$ y
1	1	0
1	0	1
0	1	1
0	0	0

4. Công thức trong đại số logic :

4.1/ Công thức :

Từ các biến mệnh đề sơ cấp, nhờ các phép toán logic cơ bản, ta lập được các mệnh đề phức hợp, chúng được gọi là các công thức . Ta thường ký hiệu công thức bởi các chữ F, G, H, R, ...

- **Ví dụ :**

$F = ((x \wedge y) \rightarrow z)$

$G = (x \rightarrow (y \rightarrow z))$

$R = (x \wedge y) \vee z)$

4.2/ Công thức tương đương :

Hai công thức F và G gọi là tương đương logic nếu chúng nhận cùng giá trị chân lý với mọi giá trị của các biến mệnh đề sơ cấp. Ký hiệu **F = G**

Ví dụ :

F = ((x ∧ y) → z)

G = (x→ (y→z)) (chứng minh như bài tập)

thì F = G

❖Các luật về phép phủ định:	◆ ¬ ¬ p ⇔ p (luật phủ định của phủ định) ◆ ¬ 1 ⇔ 0 ◆ ¬ 0 ⇔ 1
❖Luật giao hoán	◆ p ∧ q ⇔ q ∧ p ◆ p ∨ q ⇔ q ∨ p
❖Luật kết hợp	◆ p ∧ (q ∧ r) ⇔ (p ∧ q) ∧ r ◆ p ∨ (q ∨ r) ⇔ (p ∨ q) ∨ r
❖Luật phân bố	◆ p ∧ (q ∨ r) ⇔ (p ∧ q) ∨ (p ∧ r) ◆ p ∨ (q ∧ r) ⇔ (p ∨ q) ∧ (p ∨ r)
❖Luật De Morgan	◆ ¬ (p ∧ q) ⇔ ¬ p ∨ ¬ q ◆ ¬ (p ∨ q) ⇔ ¬ p ∧ ¬ q
❖Luật về phần tử bù	◆ p ∨ ¬ p ⇔ 1 ◆ p ∧ ¬ p ⇔ 0
❖Luật kéo theo	◆ p → q ⇔ ¬ p ∨ q
❖Luật tương đương	◆ p ↔ q ⇔ (p → q) ∧ (q → p)

❖Các luật đơn giản của phép tuyển

- $p \vee p \Leftrightarrow p$ (tính lũy đẳng của phép tuyển)
- $p \vee 1 \Leftrightarrow 1$ (luật này còn được gọi là luật thống trị)
- $p \vee 0 \Leftrightarrow p$ (luật này còn được gọi là luật trung hòa)
- $p \vee (p \wedge q) \Leftrightarrow p$ (luật này còn được gọi là luật hấp thụ)

❖Các luật đơn giản của phép hội

- $p \wedge p \Leftrightarrow p$ (tính lũy đẳng của phép hội)
- $p \wedge 1 \Leftrightarrow p$ (luật này còn được gọi là luật trung hòa)
- $p \wedge 0 \Leftrightarrow 0$ (luật này còn được gọi là luật thống trị)
- $p \wedge (p \vee q) \Leftrightarrow p$ (luật này còn được gọi là luật hấp thụ)

Một số luật trong các luật trình bày ở trên có thể được suy ra từ các luật khác. Các công thức tương đương logic khác:

$x \oplus y = (\neg x \wedge y) \vee (x \wedge \neg y)$

$x \oplus y = y \oplus x$

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$

$x(y \oplus z) = xy \oplus xz$

$\neg x \oplus 1 = x$

$x \oplus 1 = \neg x$

$x \oplus x = 0$

4.3/ Các qui tắc thay thế:

Dưới đây là các qui tắc để cho ta có thể suy ra những biểu thức logic mới hay tìm ra các biểu thức logic tương đương với một biểu thức logic cho trước.

- **Qui tắc 1:**

Trong một biểu thức logic E, nếu ta thay thế một biểu thức con bởi một biểu thức logic tương đương với biểu thức con đó thì ta sẽ được một biểu thức mới E' tương đương với biểu thức E.

Ví dụ : Cho biểu thức logic E = q ∨¬ p. Thay thế q trong biểu thức E bởi biểu thức ¬ ¬ q (tương đương với q) ta được một biểu thức mới E' = ¬ ¬ q ∨¬ p. Theo qui tắc thay thế 1 ta có:

$$q \vee \neg p \Leftrightarrow \neg \neg q \vee \neg p$$

Qui tắc 2:

Giả sử biểu thức logic E là một hằng đúng. Ị ếu ta thay thế một biến mệnh đề p bởi một biểu thức logic tuỳ ý thì ta sẽ được một biểu thức logic mới E' cũng là một hằng đúng.

Ví dụ : Ta có biểu thức E(p,q) = $(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$ là một hằng đúng. Thay thế biến q trong biểu thức E bởi biểu thức $q \wedge r$ ta được biểu thức logic mới:

$$E'(p,q,r) = (p \rightarrow (q \wedge r)) \Leftrightarrow (\neg p \vee (q \wedge r))$$

Theo qui tắc thay thế 2 ta có biểu thức E'(p,q,r) cũng là một hằng đúng.

Ví dụ áp dụng:

- **Ví dụ 1:** Chứng minh rằng

$$(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p).$$

Chứng minh :

$(p \rightarrow q) \Leftrightarrow \neg p \vee q$ (luật kéo theo)

$\Leftrightarrow q \vee \neg p$ (luật giao hoán)

$\Leftrightarrow \neg q \vee \neg p$ (luật phủ định)

$\Leftrightarrow \neg q \rightarrow \neg p$ (luật kéo theo)

- **Ví dụ 2:** Chứng minh rằng biểu thức sau là một hằng đúng.

$$((p \rightarrow q) \wedge p) \rightarrow q$$

Chứng minh.

$((p \to q) \wedge p) \to q$

$\Leftrightarrow ((p \to q) \wedge p) \vee q$ (luật kéo theo)

$\Leftrightarrow (\neg (p \to q) \vee \neg p) \vee q$ (luật De Morgan)

$\Leftrightarrow \neg (p \to q) \vee (\neg p \vee q)$ (luật kết hợp)

$\Leftrightarrow \neg (p \to q) \vee (p \to q)$ (luật kéo theo)

$\Leftrightarrow 1$ (luật về phần tử bù)

Vậy biểu thức $((p \to q) \wedge p) \to q$ là hằng đúng.

- **Ví dụ 3:** Chứng minh rằng biểu thức sau là một hằng đúng.

$$p \wedge q \to p$$

Chứng minh.

$p \wedge q \to p \Leftrightarrow \neg (p \wedge q) \vee p$ (luật kéo theo)

$\Leftrightarrow (\neg p \vee \neg q) \vee p$ (luật De Morgan)

$\Leftrightarrow (\neg q \vee \neg p) \vee p$ (luật giao hoán)

$\Leftrightarrow \neg q \vee (\neg p \vee p)$ (luật kết hợp)

$\Leftrightarrow \neg q \vee 1$ (luật về phần tử bù)

$\Leftrightarrow 1$ (luật đơn giản)

Vậy mệnh đề $p \wedge q \to p$ là hằng đúng.

- **Ví dụ 4:** Chứng minh rằng biểu thức sau là một hằng đúng.

$$p \to p \vee q$$

Chứng minh.

$p \to p \vee q \Leftrightarrow \neg p \vee (p \vee q)$ (luật kéo theo)

$\Leftrightarrow (\neg p \vee p) \vee q$ (luật kết hợp)

$\Leftrightarrow 1 \vee q$ (luật về phần tử bù)

$\Leftrightarrow$ 1 (luật đơn giản)

Vậy mệnh đề $p \to p \vee q$ là hằng đúng.

Nhận xét: Các ví dụ trên cho ta thấy một quan hệ khác giữa các mệnh đề phức hợp hay các mệnh đề : quan hệ "**suy ra**". Khi mệnh đề $p \to q$ là hằng đúng, ta nói rằng p suy ra q (về mặt logic). Chúng ta sẽ dùng ký hiệu $\Rightarrow$ để chỉ quan hệ "suy ra". Quan hệ suy ra nầy có tính truyền (hay bắc cầu), nhưng không có tính chất đối xứng.

5. Hệ quả logic và tương đương logic:

Ị ếu công thức $x \to y = 1$ thì mệnh đề y được gọi là hệ quả logic của x.

Ị ếu x là hệ quả logic của y và y và hệ quả logic của x thì x và y là tương đương logic.

Ví dụ: Ị ếu $F_1 = (x \to y) \wedge (y \to z)$

$G_1 = (x \to z)$

$F_2 = (x \to z) \wedge (y \to z)$

$G_2 = (x \vee y) \to z$

Khi đó G_1 là hệ quả logic của F_1, G_2 và F_2 là tương đương logic

6. Công thức đối ngẫu

Các phép toán logic hội và tuyển được gọi là hai phép toán đối ngẫu của nhau.

Hai công thức F và G được gọi là đối ngẫu của nhau nếu công thức này suy ra từ công thức kia bằng cách thay mọi phép toán tuyển và hội bằng các phép toán đối ngẫu của nó. Ký hiệu công thức đối ngẫu của công thức F là F^*

Ví dụ: Ị ếu $F = (x_1 \vee \overline{x_2}) \wedge x_3$ thì $F^* = (x_1 \wedge \overline{x_2}) \vee x_3$

7. Tính đầy đủ của một hệ các phép toán

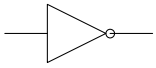
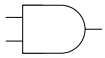
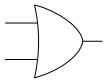
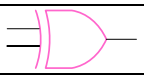
Một hệ thống $\sum$ bao gồm các phép toán logic được gọi là một hệ đầy đủ nếu mọi công thức logic đều có thể biểu diễn chỉ gồm các biến mệnh đề với chỉ các phép toán logic trong hệ.

Các hệ sau là thể hiện tính đầy đủ của các phép toán:

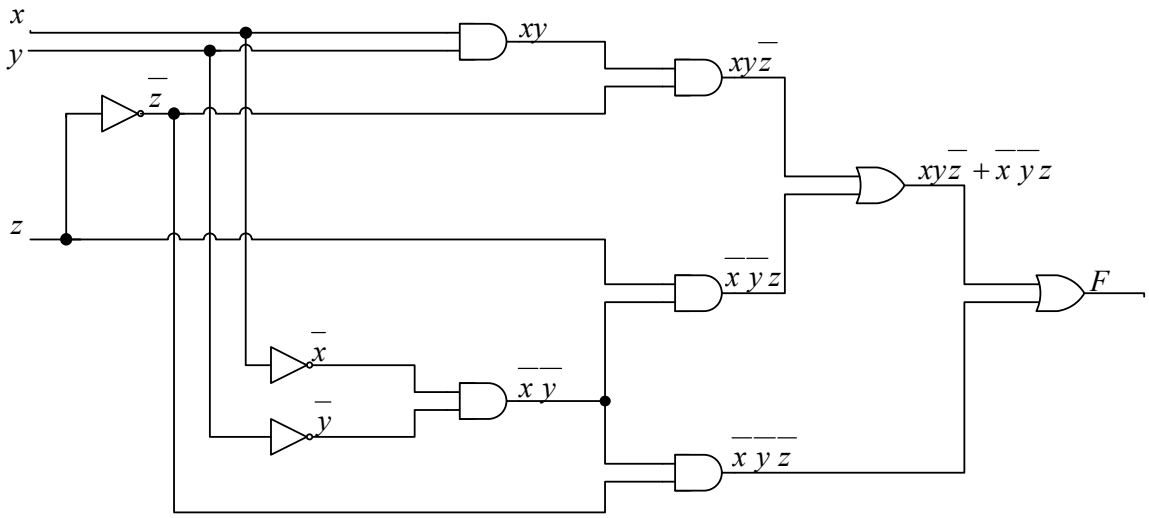
$$\sum_0 = \{\wedge, \vee, \neg\},\ \sum_1 = \{\wedge, \neg\},\ \sum_2 = \{\vee, \neg\},\ \sum_3 = \{\wedge, \oplus, \neg\}$$

7. Ứng dụng logic mệnh đề để vẽ mạch điện tử

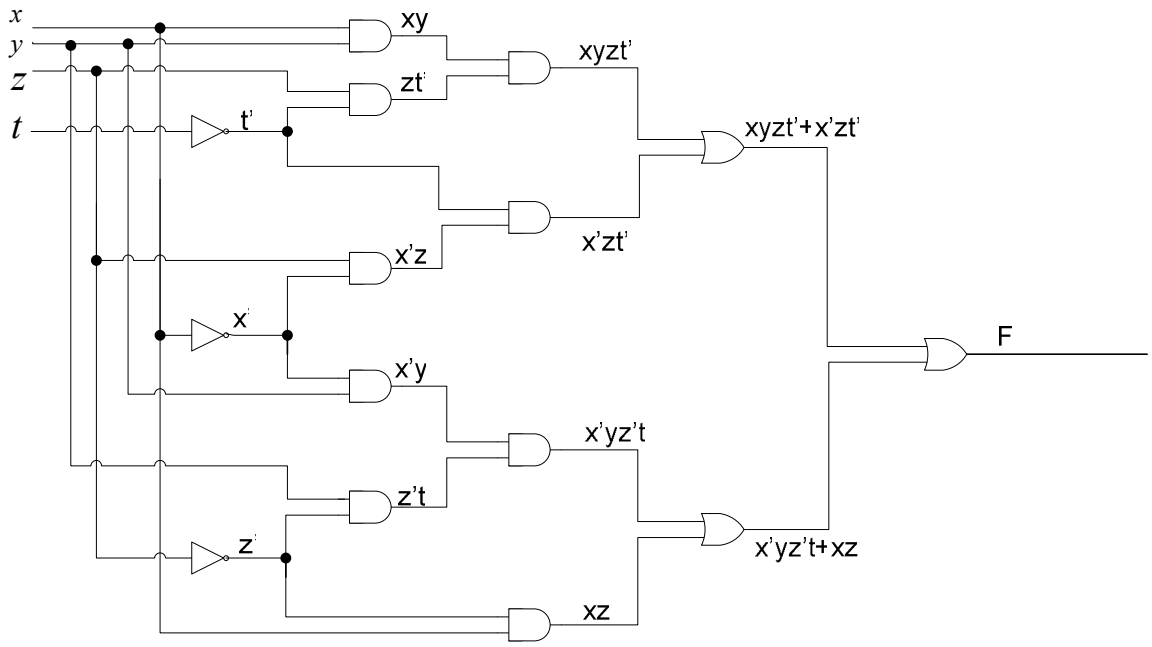
Ta có các cổng cơ bản sau để thiết kế mạch:

	Cổng inverter thể hiện phép toán phủ định 1 biến ngõ nhập
	Cổng AI D thể hiện phép toán hội giữa các biến ngõ nhập.
	Cổng OR thể hiện phép toán tuyển giữa các biến ngõ nhập
	Cổng XOR thể hiện cho phép toán tuyển chọn.

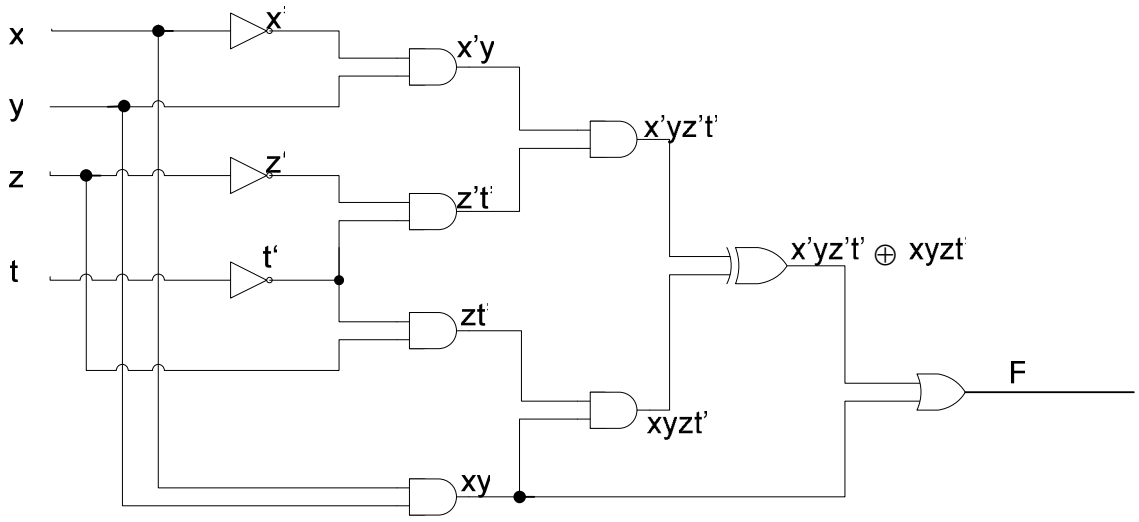
$$F = xy\bar{z} + \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z}$$



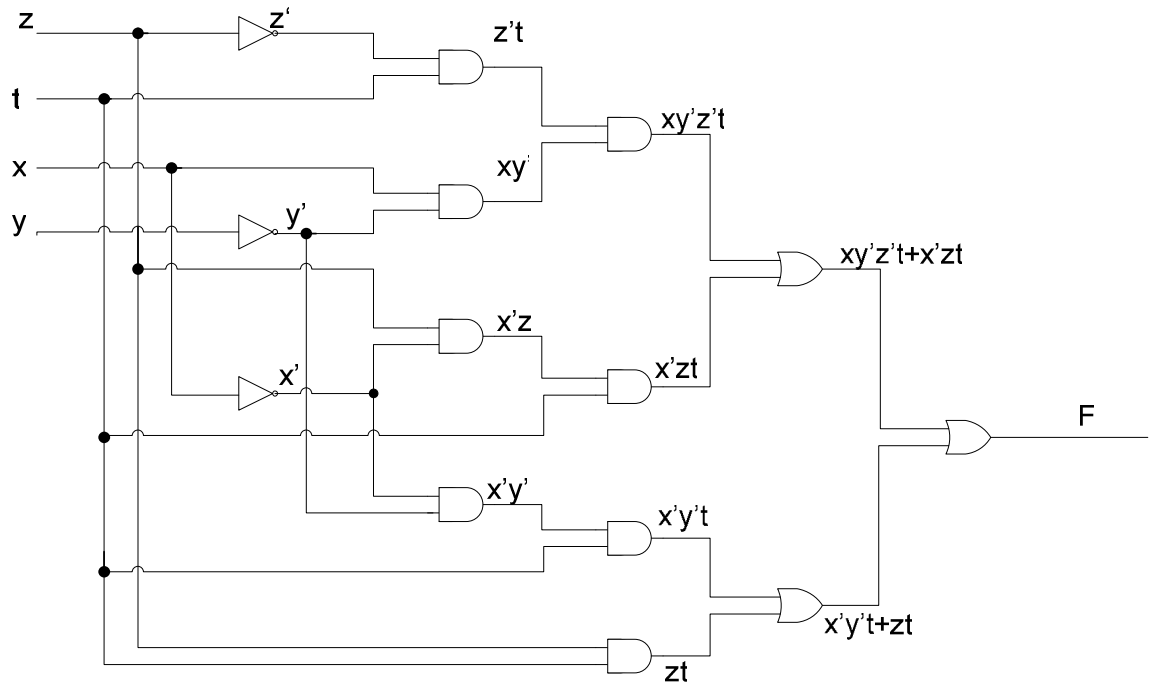
F = xyzt '+ x'zt' + x'yz't + xz'



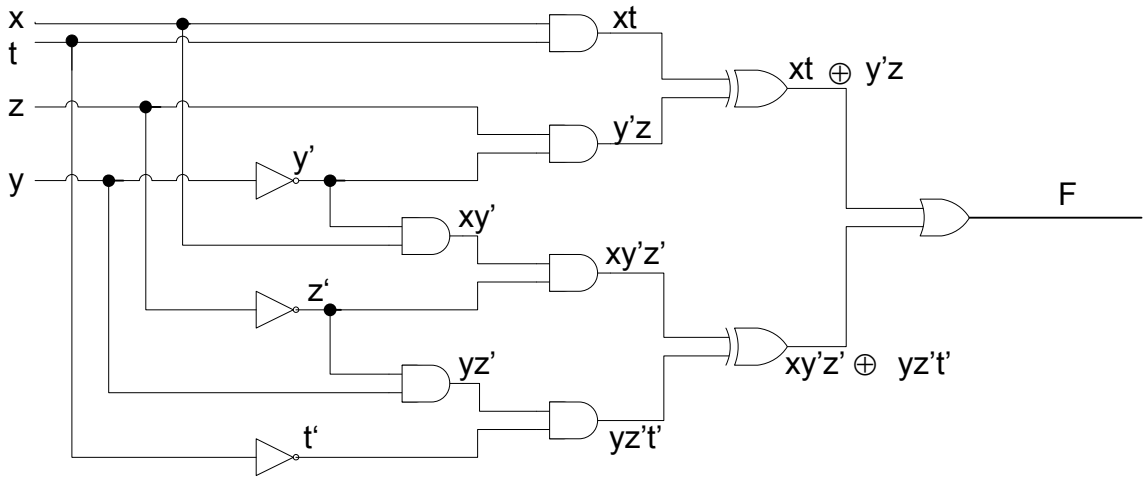
$$F=xy+x'yz't' \oplus xyzt'$$



$$F = x'zt + xy'z't + x'y't + zt$$



$$F=(xt \oplus y'z) + (xy'z' \oplus yz't')$$



BÀI TẬP

1. Cho biết những khẳng định nào dưới đây là mệnh đề:
   a. 2 là một số nguyên dương.
   b. 2k – 1 là một chẵn.
   c. Trời lạnh quá!

d. Ị ếu biết trước là khó khăn sao anh còn cố gắng?

2. Hãy viết các mệnh đề đã cho dưới đây dưới dạng hình thức có sử dụng phép nối:

   A: "Bình biết chạy xe đạp"

   B: "Bình không biết chạy xe máy"

   a. Bình không chạy được xe máy nhưng chạy được xe đạp.

   b. Bình không biết chạy xe đạp lẫn xe gắn máy.

   c. Ị ếu Bình biết chạy xe máy thì biết chạy xe đạp.

   d. Bình biết chạy xe máy và xe đạp hay Bình chạy được xe máy mà không chạy được xe đạp.

3. Cho biết chân trị các mệnh đề sau:

   a. $x \le 1$ và $x^2 + 1 > 2$

   b. $x^2 - 2x + 1 \le 0 \to x = 1$

   c. $x^2 - 2x + 1 \le 0 \leftrightarrow x = 0$

   d. $x^2 - 4x + 3 \le 0 \to x = 2$

4. Lập bảng chân trị và chỉ ra các mệnh đề luôn đúng:

   a. $m \to (n \to p)$

   b. $[(\overline{m} \to n) \vee (n \to \overline{m})] \to (m \wedge n)$

   c. $[m \to (n \to p)] \to (n \to p)$

   d. $[m \to (n \vee p)] \to [(m \to p) \vee (n \to p)]$

5. Áp dụng các luật logic để rút gọn các mệnh đề và chỉ ra đâu là công thức hằng:

   a. $\overline{(m \vee n)} \vee [(\overline{m} \wedge n) \vee \overline{n}]$

   b. $(m \to n) \wedge [\overline{n} \wedge (p \vee \overline{n})]$

   c. $m \wedge (n \vee p) \wedge (\overline{m} \vee \overline{n} \vee p)$

   d. $[[(m \wedge n) \wedge p] \vee [(m \wedge p) \wedge \overline{p}]] \vee \overline{n}$

6. Hãy $\overline{\mathrm{F}}$, F^*, $\overline{\mathrm{F}^*}$, $\overline{\mathrm{F}^*}(\overline{m}, \overline{n}, \overline{p}, \overline{q})$ của các công thức sau:

   a. $F(m,n) = \overline{(m \vee n)} \vee [(\overline{m} \wedge n) \vee \overline{n}]$

b. $F(m,n,p) = mn \wedge [\overline{n} \wedge (p \vee \overline{n})]$

c. $F(m,n,p) = \overline{(p \vee \overline{n})} \vee [(\overline{m} \wedge p) \vee \overline{n}]$

d. $F(m,n,p,q) = mq \wedge (n \vee p) \wedge (\overline{m} \vee \overline{n} \vee p)$

7. Hãy biểu diễn các công thức sau dưới tất các hệ đầy đủ các phép toán:

a. $G = \overline{n} \vee (m \wedge \overline{(\overline{m} \vee \overline{n} \vee p)})$

b. $F = \overline{n} \oplus (m \vee p)$

c. $G = m \leftrightarrow (n \oplus p)$

d. $G = (\overline{n} \oplus m) \leftrightarrow (\overline{m} \vee p)$

8. Một xạ thủ bắn cung vào một mục tiêu, kết quả được thể hiện theo các mệnh đề sau: P_k = { Phát thứ k trúng đích} k = 1, 2, 3

Hãy giải thích các mệnh đề phức hợp sau:

a. $P_1 \wedge P_2 \wedge P_3$

b. $P_1 \vee P_2 \vee P_3$

c. $(P_1 \wedge \overline{P_2} \wedge \overline{P_3}) \vee (\overline{P_1} \wedge P_1 \wedge \overline{P_3}) \vee (\overline{P_1} \wedge \overline{P_2} \wedge P_3)$

d. $(P_1 \wedge P_2) \vee (\overline{P_1} \wedge P_2 \wedge P_3) \vee (P_1 \wedge \overline{P_2} \wedge P_3)$

9. Cho 4 thí sinh A, B, C, D tham gia thi đấu xếp hạng. Kết quả xếp hạng như thế nào nếu:

- Ị gười thứ 1 dự đoán: B hạng nhì, C hạng ba.
- Ị gười thứ 2 dự đoán: A hạng nhì, C hạng tư.
- Ị gười thứ 3 dự đoán: B hạng nhất, D hạng nhì.
- Được biết là mỗi người có phần đúng phần sai.

10. Trong một chatroom, có tổng công 5 người An, Bình, Chinh, Dung, Yến đang thảo luận về đề tài logic toán với nhau trên mạng.

Biết rằng:

- Hoặc An, hoặc Bình hoặc là cả 2 đang thảo luận.
- Hoặc Chinh, hoặc Dung, nhưng không phải cả 2 cùng thảo luận.

- Ị ếu Yến đang thảo luận thì Chinh cũng vậy.
- Dung và An, hoặc cả 2 cùng thảo luận, hoặc không ai thảo luận.
- Ị ếu Bình đang thảo luận thì Yến và An cũng vậy.

  Hãy giải thích xem nếu tất cả khẳng định trên đều đúng thì hiện tại ai đang thảo luận?

11. Để có đủ chứng cứ buộc tội đối với thủ phạm trong 1 vụ án, 1 cảnh sát điều tra đi thu thập bằng chứng tại một tòa biệt thự. Anh ta lần lượt hỏi một số người sau rồi có khẳng định sau:

- Ị ếu người quản gia nói thật thì người đầu bếp cũng vậy.
- Ị gười đầu bếp và người làm vườn không thể cùng nói thật.
- Ị gười làm vườn và người hầu không thể cùng nói dối.
- Ị ếu người làm vườn nói thật thì người đầu bếp nói dối.

  Hỏi có thể xác định rõ ai nói thật ai nói dối không?

12. Một sinh viên làm bài thi giữa kỳ môn logic toán gồm 5 câu hỏi trắc nghiệm đúng/sai trên máy tính, anh ta không biết trả lời chính xác cho bất kỳ câu hỏi nào nhưng biết rằng:

- Câu 1 và câu 5 cùng đòi hỏi trả lời trái ngược nhau.
- Câu 2 và câu 4 cần trả lời như nhau.
- Ít nhất 1 trong 2 câu hỏi đầu cần trả lời là khẳng định.
- Ị ếu câu 4 trả lời là "đúng" thì câu 5 phải trả lời là "sai".
- Kinh nghiệm cho sinh viên này thấy rằng máy đặt câu hỏi cần trả lời "đúng" nhiều hơn "sai".

  Hãy xác định các đáp án cần chọn lựa.

13. Sau khi thu gọn, hãy tìm các bộ nghiệm (x, y, z, t) để các công thức hàm sau đạt giá là 0:

   a. $F = [(y\bar{t} \vee xz) \to (xt \to \bar{y}z)] \to [x\bar{y}t \to (y\bar{z} \vee \bar{x}t)]$

   b. $F = [(x\bar{y} \vee z\bar{t}) \to (yz \vee \bar{x}t)] \to [(xz \vee yt) \to (\bar{y}z \vee x\bar{t})]$

   c. $F = [(x\bar{y}z \vee y\bar{t}) \to (\bar{x}t \to yz)] \to [(y\bar{t} \vee xz) \to \bar{x}\bar{y}z]$

BÀI 3: LOGÍC TÍNH TOÁN

1. Khái niệm:

1.1/ Dạng tuyển chuẩn:

Giả sử $p_1, p_2, \dots, p_n$ là các biến mệnh đề. Một biểu thức logic F theo các biến mệnh đề $p_1, p_2, \dots, p_n$ được gọi là một biểu thức hội cơ bản nếu nó có dạng sau:

> **$F = q_1 \wedge q_2 \wedge \dots \wedge q_n$**
>
> với $q_j = p_j$ hoặc $q_j = \neg p_j$ $(j = 1, \dots, n)$.

❖ **Ví dụ**: Biểu thức $x \wedge \neg y \wedge z$ là một biểu thức hội cơ bản theo 3 biến mệnh đề x, y, z.

Biểu thức logic $E(p_1, p_2, \dots, p_n)$ theo các biến mệnh đề $p_1, p_2, \dots, p_n$ được nói là có dạng tuyển chuẩn khi E có dạng:

> $E = E_1 \vee E_2 \vee \dots \vee E_m$

Trong đó mỗi biểu thức con E_i đều có dạng tuyển chuẩn theo các biến $p_1, p_2, \dots, p_n$.

❖ **Ví dụ:** Các biểu thức sau đây có dạng tuyển chuẩn:

$E(x,y,z) = (x \wedge y \wedge z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge y \wedge z)$

$F(p_1,p_2,p_3,p_4) = (p_1 \wedge p_2 \wedge p_3 \wedge p_4) \vee (p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg p_4)$

Định lý :

Mọi biểu thức logic $E(p_1, p_2, \dots, p_n)$ đều có thể viết dưới dạng tuyển chuẩn duy nhất, không kể sự sai khác về thứ tự trước sau của các biểu thức hội cơ bản trong phép tuyển). Ị ói một cách khác, ta có duy nhất một tập hợp các biểu thức hội cơ bản $\{ E_1, E_2, \dots, E_m\}$ sao cho biểu thức $E(p_1, p_2, \dots, p_n)$ tương đương logic với biểu thức $(E_1 \vee E_2 \vee \dots \vee E_m.)$

1.2/ Dạng hội chuẩn:

Giả sử $p_1, p_2, \dots, p_n$ là các biến mệnh đề. Một biểu thức logic F theo các biến mệnh đề $p_1, p_2, \dots, p_n$ được gọi là một biểu thức tuyển cơ bản nếu nó có dạng sau:

> $F = q_1 \vee q_2 \vee \dots \vee q_n$
>
> với $q_i = p_i$ hoặc $q_i = \neg p_i$ $(j = 1, \dots, n)$.

❖ **Ví dụ:** Biểu thức $x \vee \neg y \vee z$ là một biểu thức tuyển cơ bản theo 3 biến mệnh đề x, y, z.

Biểu thức logic $E(p_1, p_2, \dots, p_n)$ theo các biến mệnh đề $p_1, p_2, \dots, p_n$ được nói là có dạng hội chuẩn khi E có dạng:

$$E = E_1 \wedge E_2 \wedge \dots \wedge E_m$$

Trong đó mỗi biểu thức con E_i đều có dạng tuyển chuẩn theo các biến $p_1, p_2, \dots, p_n$.

❖ **Ví dụ:** Các biểu thức sau đây có dạng hội chuẩn:

$E(x,y,z) = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee y \vee z)$

$F(p_1,p_2,p_3,p_4) = (p_1 \vee p_2 \vee p_3 \vee p_4) \wedge (p_1 \vee \neg p_2 \vee p_3 \vee \neg p_4)$

Định lý :

Mọi biểu thức logic $E(p_1, p_2, \dots, p_n)$ đều có thể viết dưới dạng hội chuẩn duy nhất, không kể sự sai khác về thứ tự trước sau của các biểu thức tuyển cơ bản trong phép hội). Ị ói một cách khác, ta có duy nhất một tập hợp các biểu thức tuyển cơ bản $\{ E_1, E_2, \dots, E_m\}$ sao cho biểu thức $E(p_1, p_2, \dots, p_n)$ tương đương logic với biểu thức $(E_1 \wedge E_2 \wedge \dots \wedge E_m.)$

2. Số logic :

2.1/ Định nghĩa :

- Số logic của một biến nguyên thủy A , kí hiệu #A là chân trị của biến đó xét trong không gian logic Ị chiều mà A tham gia biểu diễn.
- Ví dụ:

  Xét trong không gian một biến thì #A =(1,0)

  Trong không gian 2 biến: AB.

$$\begin{pmatrix} A & B \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Ma trận biểu diễn: [A,B]

Xét trong không gian 3 chiều: ABC

$$\begin{pmatrix} A & B & C \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Ma trận biểu diễn: [A,B,C]

- Mỗi không gian logic được biểu diễn bởi một số biến riêng. Sự tổ hợp các biến ra một giá trị của các biến nằm trong không gian logic.

2.2 Hàm logic:

- Hàm logic là một liên kết giữa các biến logic bởi các phép toán logic như: hợp (+), giao (.), phủ định (-), kéo theo ($\rightarrow$), tương đương ($\leftrightarrow$),….
- Ị hận xét: Một hàm logic có một số logic tương ứng: f(A,B,…) ≅ #f(A,B,…)

  Ví dụ: trong không gian 3 biến cho f(A,B,C) = $AB + \overline{B}C$

A	B	C	AB	$\overline{B}C$	f(A,B,C)
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
1	1	0	1	0	1
0	0	1	0	1	1
1	0	1	0	1	1
0	1	1	0	0	0
1	1	1	1	0	1

  Vậy #f(A,B,C) = #($AB + \overline{B}C$) = 00011101

- Hệ quả:
  - Số logic của một biến hợp (tổng) bằng tổng các số logic thành phần:

    #(A+B) = #A + #B

    hay #(A ∨ B) = #A ∨ #B

- Số logic của một tích bằng tích các số logic thành phần:

  #(A . B) = #A . #B

  hay #(A ∧ B) = #A ∧ #B

- $\#\overline{A} = \overline{\#A}$
- #(A ⇒ B) = #A ⇒ #B

♦ Ví dụ minh họa: chứng minh phát biểu sau ((A→B)∧(B→C)∧A) →C

Có nhiều cách để chứng minh phát biểu trên, chẳng hạn:

- Cách 1: dùng bảng chân trị

A	B	C	A→B	B→C	(A→B)∧(B→C)∧A	((A→B)∧(B→C)∧A) →C
0	0	0	1	1	0	1
1	0	0	0	1	0	1
0	1	0	1	0	0	1
1	1	0	1	0	0	1
0	0	1	1	1	0	1
1	0	1	0	1	0	1
0	1	1	1	1	0	1
1	1	1	1	1	1	1

Ị hận xét: cách này khá phức tạp nếu số các phép kết nối logic lớn

- Cách 2: dùng các luật logic của Vương hạo (1962) để chứng minh. Cách này không dễ.
- Cách 3: tính số logic của phát biểu, nếu số logic bằng I(1111…1) thì phát biểu trên là đúng.

Ta có #(((A→B)∧(B→C)∧A)→C)

= (#(A→B)∧#(B→C)∧#A)→#C

= ((#A→#B)∧(#B→#C)∧#A)→#C

= ((01010101→00110011)∧(00110011→00001111)∧01010101)→00001111

= (10111011∧11001111∧01010101) → 00001111

= 11111111 = I ⇒ phát triển trên là đúng.

2.2 Tương đương logic:

- Hai công thức logic $f(A_1,A_2,\ldots,A_m)$ và $g(A_1,A_2,\ldots,A_n)$ được gọi là tương đương khi và chỉ khi #f = #g xét trên không gian lớn nhất chứa f và g.
- Ví dụ: cho biết f(A,B) = A→B và $g(A,B,C) = \left(C \vee \overline{C}\right) \wedge \left(\overline{B} \to \overline{A}\right)$ có tương đương nhau hay không?
  - Ta có: #f = 10111011. Ở đây f cũng phải được xét trong không gian 3 biến vì nó cần phải so sánh với g có 3 biến
  - #g = 10111011

  Do #f = #g ⇒ f và g là tương đương nhau.

3. Thuật toán biểu diễn một công thức logic dưới dạng tuyển chuẩn:

Số logic của tích các biến: Gọi C_i là số logic của một tổ hợp các biến sao cho C_i chỉ nhận giá trị 1 tại cột i, các cột còn lại nhận giá trị 0.

Ví dụ: trong không gian 3 biến A,B,C; ta có biểu diễn như sau:

$\overline{A}\overline{B}\overline{C}$	1	0	0	0	0	0	0	0	C_1
$A\overline{B}\overline{C}$	0	1	0	0	0	0	0	0	C_2
$\overline{A}B\overline{C}$	0	0	1	0	0	0	0	0	C_3
$AB\overline{C}$	0	0	0	1	0	0	0	0	C_4
$\overline{A}\overline{B}C$	0	0	0	0	1	0	0	0	C_5
$A\overline{B}C$	0	0	0	0	0	1	0	0	C_6
$\overline{A}BC$	0	0	0	0	0	0	1	0	C_7
ABC	0	0	0	0	0	0	0	1	C_8

Thuật toán: cho công thức logic $f(A_1,A_2,\ldots,A_n)$.

Bước 1: Tính #f. Giả sử $\#f = (i_1,i_2,\ldots,i_2^n)$

k := 1

F := {}

Bước 2:

Nếu $i_k = 1$ thì $F := F + C_k$.

k := k+1

Bước 3: nếu $k \le 2^n$ thì quay lại bước 2, ngược lại dừng.

Khi thuật toán dừng, F chính là biểu diễn dạng tuyển chuẩn của công thức f đã cho ban đầu.

Ví dụ minh họa:

Tìm dạng tuyển chuẩn của công thức logic sau: **$f(A,B,C) = (A \to B) \wedge (B \to C)$**

Bước 1: #f = 10001011

k := 1

F := {}

Bước 2: do $i_k = 1 \Rightarrow F := F + \overline{A}\overline{B}\overline{C}$ (C_1);

k:= k+1 = 2

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 0 \Rightarrow F = \overline{A}\overline{B}\overline{C}$;

k:= k+1 = 3

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 0 \Rightarrow F = \overline{A}\overline{B}\overline{C}$;

k:= k+1 = 4

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 0 \Rightarrow F = \overline{A}\overline{B}\overline{C}$;

k:= k+1 = 5

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 1 \Rightarrow F := \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C$

k:= k+1 = 6

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 0 \Rightarrow F = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C$

k:= k+1 = 7

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 1 \Rightarrow F = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC$

k := k+1 = 8

Bước 3: do $k \le 8(2^3)$ quay lại bước 2

Bước 2: do $i_k = 1 \Rightarrow F = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + ABC$

$\Rightarrow$ Tuyển chuẩn của f là F = $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + ABC$

4. Thuật toán biểu diễn một công thức logic dưới dạng hội chuẩn:

Tích của các tổng: xét số 0 ở vị trí nào thì đưa tổ hợp tại vị trí đó vào.

Các tổ hợp hội chuẩn của 3 biến A,B,C:

$\overline{A}+\overline{B}+\overline{C}$	1	1	1	1	1	1	1	0	D_8
$A+\overline{B}+\overline{C}$	1	1	1	1	1	1	0	1	D_7
$\overline{A}+B+\overline{C}$	1	1	1	1	1	0	1	1	D_6
$A+B+\overline{C}$	1	1	1	1	0	1	1	1	D_5
$\overline{A}+\overline{B}+C$	1	1	1	0	1	1	1	1	D_4
$A+\overline{B}+C$	1	1	0	1	1	1	1	1	D_3
$\overline{A}+B+C$	1	0	1	1	1	1	1	1	D_2
$A+B+C$	0	1	1	1	1	1	1	1	D_1

Thuật toán xác định hội chuẩn:

Cho công thức logic f(A_1,A_2,…,A_n)

Bước 1: Tính #f. Giả sử #f = (i_1,i_2,…,i_2^n)

k := 1; F := {}

Bước 2:

Nếu i_k = 0 thì F := F + D_k

k := k+1

Bước 3: nếu k $\le 2^n$ thì quay lại bước 2; ngược lại dừng.

Khi thuật toán dừng, F chính là dạng hội chuẩn của công thức đã cho ban đầu.

Ví dụ minh họa:

Tìm công thức hội chuẩn của công thức sau:

f(A,B,C) = (A$\wedge$B) $\vee \overline{B} \vee C$

Bước 1: Tính #f = 11011111

k:=1

F := {}

Bước 2: do i_k = 1 $\Rightarrow$ F = {}

k := k+1 = 2

Bước 3: do k $\le 2^3$ $\Rightarrow$ quay lại bước 2

Bước 2: do $i_k = 1 \Rightarrow F = \{\}$

$k := k+1 = 3$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $i_k = 0 \Rightarrow F = D_3 = (A + \overline{B} + C)$

$k := k+1 = 4$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $C_k = 0 \Rightarrow F = D_3 = (A + \overline{B} + C)$

$k := k+1 = 5$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $C_k = 1 \Rightarrow F = (A + \overline{B} + C)$

$k := k+1 = 6$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $C_k = 1 \Rightarrow F = (A + \overline{B} + C)$

$k := k+1 = 7$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $C_k = 1 \Rightarrow F = (A + \overline{B} + C)$

$k := k+1 = 8$

Bước 3: do $k \le 2^3 \Rightarrow$ quay lại bước 2

Bước 2: do $C_k = 1 \Rightarrow F = (A + \overline{B} + C)$

$k := k+1 = 9$

Bước 3: $k > 2^3 \Rightarrow$ dừng

Vậy: dạng hội chuẩn của công thức ban đầu là $F = A + \overline{B} + C$

- Ị hận xét: tuỳ theo số logic của công thức f có số lượng số 0 hay số 1 ít hơn mà ta có thể đưa về dạng hội chuẩn hay tuyển chuẩn.

Bài tập

Chuyển tất cả các công thức sau thành dạng chuyển tuyển, chuẹn hội, chuyển tuyển hoàn toàn và chuẹn hội hoàn toàn:

1. $(m \vee \overline{n}) \wedge (\overline{n \to p})$
2. $[m \to (n \to p)] \to (n \to p)$
3. $\overline{n} \vee (m \wedge (\overline{m \vee \overline{n}} \vee p))$
4. $[(\overline{m} \to n) \vee (n \to \overline{m})] \to (m \wedge n)$
5. $(\overline{m \vee n}) \vee [(\overline{m} \wedge n) \vee \overline{n}]$
6. $(m \to n) \wedge [\overline{n} \wedge (p \vee \overline{n})]$
7. $m \wedge (n \vee p) \wedge (\overline{m} \vee \overline{n} \vee p)$
8. $[[(m \wedge n) \wedge p] \vee [(m \wedge p) \wedge \overline{p}]] \vee \overline{n}$
9. $[m \to (n \vee p)] \to [(m \to p) \vee (n \to p)]$
10. $\overline{n} \oplus (m \vee p)$
11. $m \leftrightarrow (n \oplus p)$
12. $(\overline{n} \oplus m) \leftrightarrow (\overline{m} \vee p)$

Bài 4: CÀI ĐẶT MINH HỌA

1. Thuật toán tính số logic của một công thức:

Input: Chuỗi s chứa công thức logic f (có tối đa 5 biến logic)

Output: Số logic tương ứng của f

❖Thuật toán:

unsigned long Tinh_sologic(char *s)

{ Bước 1: loại bỏ ngoặc thừa của chuỗi s

Bước 2: sử dụng biểu thức hậu tố Balan để tìm phép toán chính của biểu thức s, các phép toán cùng cấp ưu tiên sẽ được tính từ trái sang phải. Giả sử vị trí của phép toán chính là t(t=-1 có nghĩa là biểu thức không còn phép toán chính)

- Nếu t=-1:

  return #s

- Ngược lại: qua bước 3

Bước 3: Gọi s1 và s2 là 2 biểu thức bên trái và bên phải của phép toán chính của s

```
t1 = Tinh_sologic(s1);
t2 = Tinh_sologic(s2);
switch(phép toán chính)
{       case '&':  return t1 & t2;
        case 'v':  return t1 v t2;
        case '->': return t1 -> t2;
}
}
```

❖Chương trình hoàn chỉnh:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<string.h>
```

```
#define MAX 8
int SOLG[MAX],n;
char MD[8][5];// moi menh de co toi da 4 ki tu(gia su)
/*------------------------------*/
char* Bo_ngoac_2phia_thua(char *s)
{
        int l = strlen(s);
        if(s[0] != '(' || s[l-1] != ')') return s;
        int d  = 1;
        for(int i = 1; i < l-1; i++)
        {
          if(s[i]=='(') d++;
          else if(s[i]==')') d--;
          if(d==0)  return s;
        }
        char s1[100];
        strcpy(s1,s+1);
        s1[l-2] = 0;
        return s1;
}
/*----------------------------*/
int  Ktra_chua_ngoac_2phia(char *s)
{
        int l = strlen(s);
        if(s[0] != '(' || s[l-1] != ')') return 0;
        int d  = 1;
        for(int i = 1; i < l-1; i++)
        {
          if(s[i]=='(') d++;
          else if(s[i]==')') d--;
```

```
            if(d==0)  return 0;
          }
          return 1;
}
/*--------------------------------*/
void Chuan_hoa(char *mde)
{
          for(int i=strlen(mde)-1;i>=0;i--)
           if(mde[i]==32) strcpy(mde+i,mde+i+1);
}
/*--------------------------------*/
int Tim_phep_toan_chinh(char *bt)
{
  int s[100]; int t = 0, l = strlen(bt);
  for(int i = 0 ; i < l; i++)
          if(strchr(">&v()",bt[i]))
          {
                  if(bt[i]==')' )
                  {
                    while(bt[s[t-1]]!='(') t--;
                    t--;
                  }
                  else s[t++] = i;
          }
          if(t==0) return -1;
          return s[t-1];
}
/*--------------------------------*/
void SX(char md[][5],int n)
{
```

```
        for(int i = 0; i < n-1; i++)
          for(int j = i+1; j < n ;j++)
                if(strcmp(md[i],md[j])>0)
                {
                        char s[5];
                        strcpy(s,md[i]);
                        strcpy(md[i],md[j]);
                        strcpy(md[j],s);
                }
}
/*------------------------------*/
int Tim_vt(int n,char *s)
{
        for(int j = 0; j < n; j++)
          if(strcmp(s,MD[j])==0) return j;
        return -1;
}
/*------------------------------*/
int Tim_cac_mde(char *s)
{
  int cs = 0;
  int n = 0;
  int l = strlen(s);
  for(int i = 1; i <= l;i++)
        if(!strchr("&v->(",s[i]) && strchr("&v->(",s[i-1])) cs = i;
        else
         if((i==l || strchr("&v-)",s[i])) && !strchr("&v->()",s[i-1]))
         {  char s1[5]; strncpy(s1,s+cs,i-cs);
                s1[i-cs] = 0;
                if(Tim_vt(n,s1)==-1) strcpy(MD[n++],s1);
```

```
        }
  SX(MD,n);//sắp xếp để bảo đảm tính tăng của các biến nguyên thủy
                //điều này không quan trọng nhưng để khỏi bị hiểu nhầm
  return n;
}
/*-------------------*/
void Tinh_mang_so( )
{   int  *bit;
        int hmn = pow(2,n);
        bit = new int [hmn];
        bit[0]=1;
        for(int i = 1;i < hmn ; i++) bit[i] = bit[i-1]<<1;
        for(  i = 0; i< n ; i++)
        {
         int t = 0,hmi = pow(2,i);
         for(int j = 0; j < hmn; j+= (hmi<<1))
                for(int k = j; k < hmi+j;k++)
                  t |= bit[k];
         SOLG[i] = t;
        }
        delete []bit;
}
/*-------------------*/
unsigned long Tinh_sologic(char *s)
{
        s = Bo_ngoac_2phia_thua(s);
        int t = Tim_phep_toan_chinh(s);
        if(t==-1)
                if(s[0]=='-') return ~SOLG[Tim_vt(n,s+1)];
                else        return  SOLG[Tim_vt(n,s)];
```

```
        char ch = s[t];
        char s1[100],s2[100];
        strcpy(s1,s);
        strcpy(s2,s+t+1);
        s1[t] = 0;
        if(s1[t-1]=='-') s1[t-1] = 0;
        unsigned long t1 = Tinh_sologic(s1);
        unsigned long t2 = Tinh_sologic(s2);
        switch(ch)
        {
                case '&': return t1 & t2;
                case 'v': return t1 | t2;
                case '>': return (~t1)|t2;
        }
        return t;
}
/*----------------------*/
void Xuat_so_np(long t)
{
        int haimu = pow(2,n);
        for(int i=haimu-1;i>=0;i--)
         printf("%d",(t>>i)&1);
}
void main()
{
        clrscr();
        char s[100];
        printf("Ị hap cong thuc logic(cac toan tu logic la -,&,v,->):"); gets(s);
        Chuan_hoa(s);
        n = Tim_cac_mde(s);
```

```
        Tinh_mang_so();
        long t = Tinh_sologic(s);
        printf("So logic cua %s la:",s); Xuat_so_np(t);
        getch();
}
```

2. Chương trình minh họa việc kiểm tra 2 công thức tương đương:

- Thuật toán:

  Input: 2 công thức logic $f(A_1,A_2,\ldots,A_m)$ và $g(A_1,A_2,\ldots,A_n)$

  Ouput: True nếu f tương đương với g

  False nếu f không tương đương với g

  Phương pháp:

  - Tính #f
  - Tính #g
  - nếu #f = #g thì kết quả:=True ngược lại kết quả:=False

- **Chương trình hoàn chỉnh:**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<string.h>
#define MAX 8
int SOLG[MAX],n;
char MD[8][5];// moi menh de co toi da 4 ki tu(gia su)
/*-------------------------------*/
char* Bo_ngoac_2phia_thua(char *s);
/*-------------------------------*/
int  Ktra_chua_ngoac_2phia(char *s);
/*-------------------------------*/
void Chuan_hoa(char *mde);
/*-------------------------------*/
int Tim_phep_toan_chinh(char *bt);
```

Các hàm này đã được cài đặt trong phần tính số logic.

```
/*------------------------------*/
int Tim_vt(int n,char *s);
/*------------------------------*/
void SX(char md[][5],int n);
/*------------------------------*/
void Tinh_mang_so();
/*-------------------*/
unsigned long Tinh_sologic(char *s);
/*---------------------*/
void Xuat_so_np(long t);
/*----------------------------------*/
void Xuat_cac_md(char md[][5],int n);
/*----------------------------------*/
void Tim_cac_mde(char *s,int &n)
{  int cs = 0;
   int l = strlen(s);
   for(int i = 1; i <= l;i++)
         if(!strchr("&v->(",s[i]) && strchr("&v->(",s[i-1])) cs = i;
         else
          if((i==l || strchr("&v-)",s[i])) && !strchr("&v->()",s[i-1]))
          {  char s1[5]; strncpy(s1,s+cs,i-cs);
                s1[i-cs] = 0;
                if(Tim_vt(n,s1)==-1) strcpy(MD[n++],s1);
          }
}
```

Hàm tìm các mệnh đề có thay đổi so với phần tính số logic vì các mệnh đề ở đây là hội của các mệnh đề của f và g.

```
void main()
{
         clrscr();
```

```
        char s1[100],s2[100];
        printf("Chuong trinh so sanh 2 cong thuc logic tuong duong?\n");
        printf("Ị hap cong thuc logic f(cac toan tu logic la -,&,v,->):"); gets(s1);
        printf("Ị hap cong thuc logic g(cac toan tu logic la -,&,v,->):"); gets(s2);
        Chuan_hoa(s1); Chuan_hoa(s2);
        n = 0;
        Tim_cac_mde(s1,n);
        Tim_cac_mde(s2,n);
        SX(MD,n);//khong quan trong nhung de bi hieu lam nen phai sap xep
        Tinh_mang_so();
        unsigned long t1 = Tinh_sologic(s1);
        printf(" So logic cua f"); Xuat_cac_md(MD,n);
        printf(" = %s la:",s1);
        Xuat_so_np(t1);
        unsigned long t2 = Tinh_sologic(s2);
        printf("\n So logic cua g"); Xuat_cac_md(MD,n);
        printf(" = %s la:",s2);
        Xuat_so_np(t2);
        if(t1==t2){
                printf("\nCong thuc f");
                Xuat_cac_md(MD,n);
                printf(" = %s tuong duong voi cong thuc g",s1);
                Xuat_cac_md(MD,n);
                printf(" = %s",s2);
        }
        else   printf("\nHai menh de khong tuong duong");
        getch();
}
```

BÀI 5: SUY DIỄN LOGIC VÀ VỊ TỪ

1. Giới thiệu:

Một hệ thống toán học bao gồm các tiên đề, các định nghĩa, và những khái niệm không được định nghĩa. Các tiên đề được giả định là đúng. Các định nghĩa được sử dụng để xây dựng hay đưa ra những khái niệm mới trên cơ sở những khái niệm đã có. Một số thuật ngữ, khái niệm sẽ không được định nghĩa rõ ràng nhưng được ngầm định nghĩa bởi các tiên đề. Trong một hệ toán học chúng ta có thể suy ra được các định lý. Một định lý là một khẳng định được chứng minh là đúng. Một số loại định lý được xem là các bổ đề, các hệ quả.

Một lập luận (hay lý luận) chỉ ra được tính đúng đắn của mệnh đề phát biểu trong định lý được gọi là chứng minh. Logic là một công cụ cho việc phân tích các chứng minh. Trong phần nầy chúng ta sẽ đề cập đến việc xây dựng một chứng minh toán học. Để thực hiện được một lập luận hay một chứng minh chúng ta cần hiểu các kỹ thuật và các công cụ được sử dụng để xây dựng một chứng minh. Thông thường một chứng minh sẽ bao gồm nhiều bước suy luận mà ở mỗi bước ta đi đến (hay suy ra) một sự khẳng định mới từ những khẳng định đã biết.

Ví dụ về một bước suy diễn:

1/ Ị ếu một danh sách L là khác rỗng thì ta có thể lấy ra phần tử đầu trong danh sách.Vì danh sách L là rỗng nên theo sự khẳng định trên ta không thể lấy ra phần tử đầu trong danh sách.

2/ Ị ếu một danh sách L là khác rỗng thì ta có thể lấy ra phần tử đầu trong danh sách.Vì ta không thể lấy ra phần tử đầu trong danh sách L nên danh sách L là danh sách rỗng.

Trong 2 suy diễn ở ví dụ trên thì suy diễn 2/ là một suy luận đúng, nhưng suy diễn 1/ là không đúng. Vậy làm thế nào để biết được một suy diễn là đúng hay sai ? Một bước suy luận như thế phải dựa trên một qui tắc suy diễn hợp lý nào đó để nó được xem là một suy luận đúng. Các qui tắc suy diễn là cơ sở để tay biết được một lập luận hay một chứng minh là đúng hay sai. Trong các mục tiếp theo chúng ta sẽ xem xét chi tiết hơn về các qui tắc suy diễn và giới thiệu một số qui tắc suy dễn cơ bản thường được dùng trong việc suy luận và chứng minh.

2. Định nghĩa qui tắc suy diễn:

Tuy có nhiều kỹ thuật, nhiều phương pháp chứng minh khác nhau, nhưng trong chứng minh trong toán học ta thường thấy những lý luận dẫn xuất có dạng:

Nếu p_1 và p_2 và . . . và p_n thì q.

Dạng lý luận nầy được xem là hợp lý (được chấp nhận là đúng) khi ta có biểu thức

$(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \to q$ là hằng đúng.

Ta gọi dạng lý luận trên là một **luật suy diễn** và người ta cũng thường viết luật suy diễn trên theo các cách sau đây :

- Cách 1: Biểu thức hằng đúng $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \to q \Leftrightarrow 1$
- Cách 2: Dòng suy diễn $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Rightarrow q$
- Cách 3: Mô hình suy diễn

p_1

. . . .

p_n

$\therefore q$

Các biểu thức logic $p_1, p_2, \ldots, p_n$ trong luật suy diễn trên được gọi là giả thiết (hay tiền đề), và biểu thức q được gọi là kết luận. Ở đây chúng ta cũng cần lưu ý rằng lý luận trên đúng không có nghĩa là ta có q đúng và cũng không khẳng định rằng $p_1, p_2, \ldots, p_n$ đều đúng. Lý luận chỉ muốn khẳng định rằng nếu như ta có $p_1, p_2, \ldots, p_n$ là đúng thì ta sẽ có q cũng phải đúng.

Ví dụ : Giả sử p và q là các biến logic. Xác định xem mô hình sau đây có phải là một luật suy diễn hay không?

$p \to q$

p

$\therefore q$

Giải: Lập bảng chân trị ta có:

p	q	$p \to q$	$(p \to q) \wedge p$	$((p \to q) \wedge p) \to q$
1	1	1	1	1
1	0	0	0	1

0	1	1	0	1
0	0	1	0	1

Bảng chân trị cho thấy biểu thức $((p \rightarrow q) \wedge p) \rightarrow q$ là hằng đúng. Do đó, mô hình suy luận trên đúng là một luật suy diễn. Thật ra, ta chỉ cần nhìn vào các cột chân trị của p, q, và $p \rightarrow q$ trong bảng chân trị là ta có thể kết luận được rồi, vì từ bảng chân trị trên ta thấy rằng nếu các giả thiết $p \rightarrow q$ và p đúng (có giá trị bằng 1) thì kết luận q cũng đúng.

Ta có thể khẳng định được mô hình suy luận trên là một luật suy diễn mà không cần lập bảng chân trị. Giả sử $p \rightarrow q$ và p đúng. Khi đó q phải đúng, bởi vì nếu ngược lại (q sai) thì p cũng phải sai (sẽ mâu thuẫn với giả thiết).

3. Kiểm tra một qui tắc suy diễn:

Để kiểm tra một qui tắc suy diễn xem có đúng hay không ta có thể sử dụng một trong các phương pháp sau đây:

❖ Phương pháp 1: Lập bảng chân trị.

Theo phương pháp này, ta sẽ thiết lập biểu thức logic tương ứng của qui tắc suy diễn và lặp bảng chân trị của biểu thức đó để xem nó có phải là hằng đúng hay không. Trong trường hợp biểu thức logic là hằng đúng thì ta kết luận qui tắc suy diễn là đúng. Ị gược lại, ta kết luận qui tắc suy diễn là sai.

Ví dụ: Để kiểm tra qui tắc suy diễn sau: $(p \rightarrow q) \wedge p \Rightarrow q$

ta lập bảng chân trị của biểu thức logic $((p \rightarrow q) \wedge p) \rightarrow q$ theo 2 biến logic p và q. Từ bảng chân trị ta sẽ thấy biểu thức logic là hằng đúng và ta đi đến kết luận rằng qui tắc suy diễn trên là đúng.

❖ Phương pháp 2: Chứng minh bằng cách sử dụng các luật logic.

Theo phương pháp nầy, ta sẽ thiết lập biểu thức logic tương ứng của qui tắc suy diễn và chứng minh biểu thức là hằng đúng bằng cách áp dụng các luật logic và các qui tắc thay thế.

Ví dụ: Kiểm tra qui tắc suy diễn sau đây:

$(p \rightarrow q) \wedge p \Rightarrow q$

Giải: Thực hiện phép biến đổi tương đương logic, ta có:

$((p \rightarrow q) \wedge p) \rightarrow q$

$\Leftrightarrow ((\neg p \vee q) \wedge p) \rightarrow q$ (luật kéo theo)

$\Leftrightarrow ((\neg p \wedge p) \vee (q \wedge p)) \rightarrow q$ (luật phân bố)

$\Leftrightarrow (0 \vee (q \wedge p)) \rightarrow q$ (luật về phần tử bù)

$\Leftrightarrow (q \wedge p) \rightarrow q$ (luật đơn giản)

$\Leftrightarrow \neg (q \wedge p) \vee q$ (luật kéo theo)

$\Leftrightarrow (\neg q \vee \neg p) \vee q$ (luật De Morgan)

$\Leftrightarrow (\neg q \vee q) \vee \neg p$ (luật giao hoán và kết hợp)

$\Leftrightarrow 1 \vee \neg p$ (luật về phần tử bù)

$\Leftrightarrow 1$ (luật đơn giản)

Vậy biểu thức $((p \rightarrow q) \wedge p) \rightarrow q$ là hằng đúng, nên ta có qui tắc suy diễn $(p \rightarrow q) \wedge p \Rightarrow q$ là đúng.

Ghi chú: Để kiểm tra một qui tắc suy diễn ta còn có thể kết hợp 2 phương pháp trên và áp dụng cả những luật suy diễn đã biết trước.

4. Các qui tắc suy diễn cơ bản:

Trong mục nầy chúng ta nêu lên một số qui tắc suy diễn (đúng) thường được sử dụng mà ta có thể kiểm tra chúng bằng các phương pháp đã được trình bày trong mục trước.

Stt	Qui tắc	Mô hình suy diễn
1	**Qui tắc khẳng định (Modus Ponens)** $[(p \rightarrow q) \wedge p] \rightarrow q = 1$	$p \rightarrow q$ p ——— $\therefore q$
2	**Qui tắc phủ định (Modus Tollens)** $[(p \rightarrow q) \wedge q \rightarrow \neg p] = 1$	$p \rightarrow q$ $\neg q$

		–––––– ∴ ¬ p
3	**Tam đoạn luận giả thiết (Symlogism)** $[(p \to q) \wedge (q \to r) \to (p \to r)] = 1$	$p \to q$ $q \to r$ ––––– $\therefore p \to r$
3	**Tam đoạn luận tuyển** $[\neg p \wedge (p \vee q)] \to q = 1$	$\neg p$ $p \wedge q$ ––––– $\therefore q$
4	**Qui tắc nối liền** $p \wedge q \to p = 1$	$p \wedge q$ ––––– $\therefore q$
5	**Qui tắc phản đảo** $(p \to q) = (\neg q \to \neg p)$	$(p \to q)$ –––––––– $\therefore (\neg q \to \neg p)$
4	**Qui tắc chứng minh bằng phản chứng** $[(p \to q) \to (p \to \neg q)] \to 0$	Qui tắc nầy cho phép ta chứng minh $(p \to \neg q) \to 0$ thay cho $p \to q$. Ị ói cách khác, nếu ta thêm giả thiết phụ vào tiền đề p mà chứng minh được có sự mâu thuẫn thì ta có thể kết luận q từ tiền đề p.
5	**Qui tắc chứng minh theo trường hợp** $(p_1 \to q) \wedge (p_2 \to q) \wedge \ldots \wedge (p_n \to q) \Rightarrow (p_1 \vee p_2 \vee \ldots \vee p_n) \to q$	$p_1 \to q$ $p_2 \to q$ . . . $p_n \to q$ –––––––––– $\therefore (p_1 \vee p_2 \vee \ldots \vee p_n) \to q$

Kiểm tra một phép suy luận cụ thể

Để kiểm tra một suy luận cụ thể là đúng hay không, tức là có "hợp logic" hay không, ta có thể căn cứ vào các qui tắc suy diễn (luật suy diễn). Phép suy luận cụ thể có thể được xem như sự suy diễn trên các mệnh đề phức hợp. Các mệnh đề sơ cấp cụ thể (mà chân trị có thể đúng hoặc sai) trong phép suy luận sẽ được trừu tượng hóa (thay thế) bởi các biến logic. Ị hư thế phép suy luận được trừu tượng hóa thành một qui tắc suy diễn trên các biểu thức logic mà ta có thể kiểm tra xem qui tắc suy diễn là đúng hay không. Đây chính là biện pháp để ta biết được một suy luận cụ thể là đúng hay sai.

Ví dụ 1:

Xét sự suy luận sau đây: Ị ếu một danh sách L là khác rỗng thì ta có thể lấy ra phần tử đầu trong danh sách.Vì ta không thể lấy ra phần tử đầu trong danh sách L nên danh sách L là danh sách rỗng.

Trong phép suy luận, ta có các mệnh đề sơ cấp "danh sách L là khác rỗng", "ta có thể lấy ra phần tử đầu (từ danh sách L)". Thay thế các mệnh đề sơ cấp nầy bởi các biến logic p, q tương ứng thì phép suy luận cụ thể trên sẽ được trừu tượng hóa thành một suy diễn trên các biểu thức logic như sau:

$p \rightarrow q$

$\neg q$

_____

$\therefore \neg p$

Mô hình suy diễn nầy chính là qui tắc suy diễn Modus Tollens, đã được biết là đúng. Vậy phép suy luận trên là suy luận đúng.

Ví dụ 2: Xét xem suy luận sau đây có đúng hay không?

Ị ếu $\sqrt{2}$ là số hữu tĩ thì phương trình $m^2 = 2n^2$ có nghiệm nguyên dương m, n. Ị ếu phương trình $m^2 = 2n^2$ có nghiệm nguyên dương m và n thì ta có mâu thuẫn. Vậy $\sqrt{2}$ là số vô tĩ.

Trừu tượng hóa các mệnh đề sơ cấp " $\sqrt{2}$ là số hữu tĩ", " phương trình $m^2 = 2n^2$ có nghiệm nguyên dương m, n " thành các biến logic p, q tương ứng thì phép suy luận trên có dạng mộ hình suy diễn

$p \rightarrow q$

$q \rightarrow 0$

_____

$\therefore \neg p$

Kiểm tra mô hình suy diễn nầy ta sẽ thấy là đúng. Ị hư thế phép suy luận trên là đúng.

5. Các ví dụ áp dụng trong suy luận và chứng minh

Dưới đây ta trình bày chứng minh của một số mệnh đề mà không nêu lên một cách chi tiết về các qui tắc suy diễn đã được áp dụng. Ị gười đọc có thể tìm thấy các qui tắc suy diễn được sử dụng trong chứng minh một cách dễ dàng.

❖ **Mệnh đề 1:** Với mọi số nguyên n, $n^3 + 2n$ chia hết cho 3.

Suy nghĩ đầu tiên là ta thấy rằng không thể tìm thấy một thừa số 3 trong biểu thức $n^3 + 2n$. Ị hưng khi phân tích ra thừa số thì $n^3 + 2n = n(n^2 + 2)$. Phát biểu "$n^3 + 2n$ chia hết cho 3" sẽ đúng nếu n là bội số của 3. Còn các trường hợp khác thì sao?. Ta thử phương pháp phân chứng.

Chứng minh:

Ta có $n^3 + 2n = n(n^2 + 2)$, và số tự nhiên n có một trong 3 dạng ứng với 3 trường hợp dưới đây:

❖ Trường hợp 1: $n = 3k$, với k là một số nguyên.

$n^3 + 2n = 3k(9k^2 + 2)$ chia hết cho 3.

❖ Trường hợp 2: $n = 3k+1$, với k là một số nguyên.

$$n^3 + 2n = (3k+1)((3k+1)^2 + 2)$$
$$= (3k+1)(9k^2 +6k+3)$$
$$= (3k+1)3(3k^2 +2k+1) \text{ chia hết cho 3.}$$

❖ Trường hợp 3: $n = 3k+2$, với k là một số nguyên.

$$n^3 + 2n = (3k+2)((3k+2)^2 + 2)$$
$$= (3k+1)(9k^2 +12k+6)$$
$$= (3k+1)3(3k^2 +4k+2) \text{ chia hết cho 3.}$$

Trong mọi trường hợp (có thể có) ta đều có $n^3 + 2n$ đều chia hết cho 3.

Vậy ta kết luận $n^3 + 2n$ chia hết cho 3 đối với mọi số nguyên n.

- **Nhận xét :** Chứng minh trên có thể được trình bày ngắn gọn hơn bằng cách sử dụng phép đồng dư modulo 3.

❖ **Mệnh đề 2:** Ị ếu n^2 là một số chẵn thì n cũng là một số chẵn.

Suy nghĩ: Giả sử $n^2 = 2k$ (là số chẵn). Ta thấy khó suy ra n là số chẵn. Ị ếu biết thông tin gì đó về n thì suy ra điều gì đó về n^2 thì dễ hơn. Ta thử phương pháp phản chứng.

Chứng minh: Ta hãy chứng minh mệnh đề "Ị ếu n lẻ thì n^2 lẻ".

Cho n là một số lẻ, ta có $n = 2k+1$ (k là một số nguyên).
Do đó $n^2 = (2k+1)^2 = 4k^2+4k+1$ là một số lẻ.

Mệnh đề trong cặp nháy kép là đúng nên mệnh đề phản đảo của nó cũng đúng. Vậy, nếu n^2 là một số chẵn thì n cũng là một số chẵn.

Mệnh đề 3: Ị ếu $p > 3$ và p nguyên tố thì $p^2 - 1$ chia hết cho 3.

Chứng minh:

Ta có (p-1), p, (p+1) là 3 số nguyên liên tiếp. Trong 3 số nguyên nầy có một số chia hết cho 3. Ị hưng số đó không phải là p vì p là số nguyên tố lớn hơn 3. Do đó (p-1) chia hết cho 3 hoặc (p+1) chia hết cho 3. Suy ra (p-1)(p+1) chia hết cho 3, tức là $p^2 - 1$ chia hết cho 3.

❖ **Mệnh đề 4:** Số lượng các số nguyên tố là vô hạn.

Chứng minh:

Giả sử phát biểu trong mệnh đề là sai. Tức là chỉ có một số hữu hạn, k, số nguyên tố (dương). Ký hiệu k số nguyên tố là $p_1, p_2, \ldots, p_k$, ở đây k là số nguyên dương.

Đặt $n = p_1p_2 \ldots p_k + 1$.

Số n lớn hớn tất cả k số nguyên tố nên n không nguyên tố.

Do đó, từ định lý cơ bản của số học , n phải có một ước số nguyên tố p.

p phải là một trong k số nguyên tố. Do đó $p \mid (p_1p_2 \ldots p_k)$.

Suy ra $p \mid (n - p_1p_2 \ldots p_k)$, hay $p \mid 1$.

Ị hư thế, ta có p là một số nguyên tố và $p \mid 1$. Điều nầy là không thể, hay nói cách khác, ta có một mâu thuị n.

Vậy, Số lượng các số nguyên tố là vô hạn.

6. Định nghĩa vị từ và ví dụ

6.1/ Định nghĩa:

Một vị từ là một phát biểu p(x, y, …) phụ thuộc theo các biến x, y, … lấy giá trị trên các miền xác định A, B, … nào đó. Khi thay thế các biến trong vị từ bởi các giá trị cụ thể a, b, … thuộc các miền xác định thì ta được một mệnh đề p(a, b, …) có chân trị đúng hoặc sai.

Gọi B là tập hợp gồm có hai giá trị : Sai (ký hiệu bởi 0), và Đúng (ký hiệu bởi 1). Một vị từ p(x, y, …)

Ví dụ1: P(n) ≡ "n là một số nguyên tố" là một vị từ trên tập hợp các số tự nhiên (hoặc trên tập hợp các số nguyên). Ta có thể thấy rằng:

- P(1) = 0, tức là P(1) ≡ "1 là một số nguyên tố" là một mệnh đề sai.
- P(2) = 1, tức là P(2) ≡ "2 là một số nguyên tố" là một mệnh đề đúng.
- P(12) = 0, tức là P(12) ≡ "12 là một số nguyên tố" là một mệnh đề sai.
- P(17) = 1, tức là P(17) ≡ "17 là một số nguyên tố" là một mệnh đề đúng.

Vị từ "n là một số nguyên tố" có thể được xem là một ánh xạ đi từ tập hợp các số tự nhiên **N** vào tập hợp Boole B:

$P : \mathbf{N} \to B$

$$P(n) = \begin{cases} 1 & \text{khi n nguyên tố} \\ 0 & \text{khi n không nguyên tố} \end{cases}$$

Ví dụ2: p(m,n) ≡ "m là một ước số của n", với m và n là các biến số tự nhiên, cho ta một vị từ theo 2 biến m và n thuộc tập hợp các số tự nhiên. Ta có:

p(2,4) = 1, p(3,4) = 0.

6.2/ Các phép toán trên các vị từ

Cho p(x, y, …) là một vị từ theo các biến x, y, … . Phủ định của p, ký hiệu là ¬p, là một vị từ mà khi thay các biến x, y, … bởi các phần tử cụ thể a, b, … tương ứng thì ta được mệnh đề ¬(p(a, b, …)). Ị óI một cách khác, vị từ ¬p được định nghĩa bởi:

(¬ p) (x, y, …) = ¬(p(x, y, …))

Cho p(x, y, …) và q(x, y, …) là các vị từ theo các biến x, y, … . Phép hội của p và q, ký hiệu là p→ q, là một vị từ mà khi thay các biến x, y, … bởi các phần tử cụ thể a, b, … tương ứng thì ta được mệnh đề p(a, b, …) → q(a, b, …). Ị ói một cách khác, vị từ p ?q được định nghĩa bởi:

$(p \wedge q)(x, y, \ldots) = p(x, y, \ldots) \wedge q(x, y, \ldots)$

Một cách tương tự, các phép toán tuyển, kéo theo và tương đương của 2 vị từ p và q có thể được định nghĩa như sau:

- $(p \vee q)(x, y, \ldots) = p(x, y, \ldots) \vee q(x, y, \ldots)$
- $(p \rightarrow q)(x, y, \ldots) = p(x, y, \ldots) \rightarrow q(x, y, \ldots)$
- $(p \leftrightarrow q)(x, y, \ldots) = p(x, y, \ldots) \leftrightarrow q(x, y, \ldots)$

6.3/ Qui tắc phủ định mệnh đề có lượng từ

Dựa vào cách xác định chân trị của các mệnh đề có lượng từ theo ngữ nghĩa tự nhiên của các phát biểu, ta có các qui tắc phủ định mệnh đề có lượng từ sau đây:

$\neg (\forall x : P(x)) \equiv \exists x : \neg P(x)$ (1)

$\neg (\exists x : P(x)) \equiv \forall x : \neg P(x)$ (2)

Ví dụ 1: Tìm phủ định của mệnh đề "tồn tại một số thực x sao cho $x^2 < 0$".

Đặt $P(x) \equiv$ "$x^2 < 0$". Mệnh đề đã cho được viết dưới dạng ký hiệu như sau: $\exists x : P(x)$

Áp dụng luật phủ định mệnh đề có lượng từ, ta có mệnh đề phủ định cần tìm có dạng :

$\forall x : \neg P(x)$. Vậy mệnh đề phủ định là: "Với mọi số thực x, $x^2 \geq 0$".

Ghi chú:

Từ các qui tắc trên ta có thể nói chung về qui tắc phủ định mệnh đề có lượng từ như sau: Ị ếu trong một mệnh đề có lượng từ ta thay thế lượng từ ∀ bởi lượng từ ∃ , lượng từ ∃ bởi lượng từ ∀ , và biểu thức vị từ được thay thế bởi phủ định của nó thì ta sẽ được mệnh đề phủ định của mệnh đề có lượng từ ban đầu. Qui tắc nầy cũng áp dụng được cho các mệnh đề với nhiều lượng từ.

Ví dụ 2:

Cho p(x, y, z) là một vị từ phụ thuộc vào biến bộ ba $(x,y,z) \in AxBxC$. Miền xác định là tích Đê-Cat của 3 tập hợp A, B, C. Trong trường hợp nầy ta nói vị từ p là một vị từ theo 3 biến

x, y, z. Miền xác định tương ứng của 3 biến nầy là A, B, C. Hãy tìm phủ định của mệnh đề sau:

$\forall x \in A, \exists y \in B, \exists z \in C : p(x,y,z)$

Theo qui tắc chung ta có :

$\neg (\forall x \in A, \exists y \in B, \exists z \in C : p(x,y,z)) \equiv$

$\exists x \in A, \forall y \in B, \forall z \in C : \neg p(x,y,z)$

Thật ra nếu thực hiện từng bước theo các qui tắc (1) và (2) ta cũng đạt được mệnh đề phủ định như trên:

$\neg (\forall x \in A, \exists y \in B, \exists z \in C : p(x,y,z)) \equiv$

$\exists x \in A, \neg (\exists y \in B, \exists z \in C : p(x,y,z)) \equiv$

$\exists x \in A, \forall y \in B, \neg (\exists z \in C : p(x,y,z)) \equiv$

$\exists x \in A, \forall y \in B, \forall z \in C : \neg p(x,y,z)$

Ví dụ 3:

Với một hàm số f xác định ở một lân cận của điểm a $\in$ **R** (a là một số thực), ta có định nghĩa sự liên tục của f tại a như sau : f liên tục tại a nếu và chỉ nếu cho một số dương ε tùy ý, ta có một số dương δ sao cho $|x-a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon$. Ị hư vậy f liên tục tại a khi và chỉ khi mệnh đề sau đây đúng:

"cho số dương ε tùy ý, ta có một số dương δ sao cho với mọi x ta có

$|x-a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon$ ".

Hãy tìm phủ định của mệnh đề trên.

Mệnh đề trên được viết là :

$\forall \varepsilon > 0, \exists \delta > 0 : (\forall x : |x-a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon)$

Theo qui tắc phủ định mệnh đề có lượng từ, phủ của mệnh đề trên là:

$\exists \varepsilon > 0, \forall \delta > 0 : \neg (\forall x : |x-a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon)$

$\equiv \exists \varepsilon > 0, \forall \delta > 0 : (\exists x : \neg (|x-a| < \delta \rightarrow |f(x) - f(a)| < \varepsilon))$

$\equiv \exists \varepsilon > 0, \forall \delta > 0 : (\exists x : |x-a| < \delta \wedge \neg (|f(x) - f(a)| < \varepsilon))$

$\equiv \exists\, \varepsilon > 0, \forall\, \delta > 0 : (\exists\, x : |\, x-a\,| < \delta \wedge |\, f(x) - f(a)\,| \geq \varepsilon\,)$

$\equiv \exists\, \varepsilon > 0, \forall\, \delta > 0, \exists\, x : |\, x-a\,| < \delta \wedge |\, f(x) - f(a)\,| \geq \varepsilon$

Ị hư vậy ta có thể phát biểu mệnh đề phủ định như sau:: "Tồn tại một số dương ε sao cho ứng với số dương δ tùy ý có một số thực x thoả điều kiện $|\, x-a\,| < \delta$ và $|\, f(x) - f(a)\,| \geq \varepsilon$ ".

Như vậy ta có thể phát biểu mệnh đề phủ định như sau: "Tồn tại một số dương ε sao cho ứng với mỗi số dương δ tùy ý ta có một số thực x thỏa điều kiện $|\, x-a\,| < \delta$ và $|\, f(x) - f(a)\,| \geq \varepsilon$ ".

6.4/ Một số qui tắc dùng trong suy luận:

Thay đổi thứ tự lượng từ hóa của 2 biến

Cho một vị từ p(x, y) theo 2 biến x, y. Ị ếu lượng từ hóa cả 2 biến x, y trong đó ta lượng từ hóa biến y trước và lượng từ hóa biến x sau thì sẽ được 4 mệnh sau đây:

- $\forall\, x, \forall\, y : p(x,y)$
- $\exists\, x, \forall\, y : p(x,y)$
- $\forall\, x, \forall\, y : p(x,y)$
- $\forall\, x, \forall\, y : p(x,y)$

Tương tự ta cũng có 4 mệnh đề lượng từ hóa từ vị từ p(x, y) trong đó ta lượng từ hóa biến x trước và lượng từ hóa biến y sau:

- $\forall\, y, \forall\, x : p(x,y)$
- $\exists\, y, \forall\, x : p(x,y)$
- $\forall\, y, \forall\, x : p(x,y)$
- $\forall\, y, \forall\, x : p(x,y)$

Định lý dưới đây cho ta một số tính chất liên quan đến thứ tự của việc lượng từ hóa các biến trong các mệnh đề có lượng từ.

Định lý: Giả sử p(x, y) là một vị từ theo 2 biến x, y thì các mệnh đề sau là đúng

- $(\forall\, x, \forall\, y : p(x,y)\,) \leftrightarrow (\forall\, y, \forall\, x : p(x,y)\,)$
- $(\exists\, x, \exists\, y : p(x,y)\,) \leftrightarrow (\exists\, y, \exists\, x : p(x,y)\,)$

- $(\exists x, \forall y : p(x,y)) \rightarrow (\forall y, \exists x : p(x,y))$

Qui tắc đặc biệt hóa phổ dụng

❖ **Qui tắc:**

*G*iả sử một mệnh đề có lượng từ trong đó biến x với miền xác định là A, được lượng từ hóa và bị buộc bởi lượng từ phổ dụng $\forall$, và mệnh đề là đúng. Khi đó nếu thay thế x bởi a $\in$ A thì ta sẽ được một mệnh đề đúng.

Ví dụ1: Biết rằng phát biểu "mọi số nguyên tố lớn hơn 2 đều là số lẻ" là một mệnh đề đúng. Cho a là một số nguyên tố lớn hơn 2 (cố định nhưng tùy ý). Hãy chứng minh rằng a là một số lẻ.

Giải: Đặt p(n) ≡ "n là số nguyên tố lớn hơn 2", và q(n) ≡ "n là số lẻ". Ta có p(n) và q(n) là các vị từ theo biến số tự nhiên n, và ta có mệnh đề đúng sau đây:

$\forall n : p(n) \rightarrow q(n)$

Theo qui tắc trên ta suy ra $p(a) \rightarrow q(a) = 1$.

Theo giả thiết ta cũng có $p(a) = 1$.

Suy ra $q(a) = 1$ (qui tắc suy diễn Modus Ponens).

Vậy ta có mệnh đề "a là một số lẻ" là đúng.

Ví dụ 2: Trong các định lý Toán học ta thường thấy các khẳng định là các mệnh đề lượng từ hóa phổ dụng. Ví dụ như các trường hợp bằng nhau của 2 tam giác bất kỳ. Khi áp dụng ta sẽ đặc biệt hóa cho 2 tam giác cụ thể.

Qui tắc tổng quát hóa phổ dụng

❖ **Qui tắc:**

Ị ếu ta thay thế biến x trong vị từ P(x) bởi một phần tử a cố định nhưng tùy ý thộc miền xác định của biến x mà mệnh đề nhận được có chân trị là đúng, tức là P(a) = 1, thì mệnh đề lượng từ hóa

$\forall x : P(x)$ là một mệnh đề đúng.

❖ **Nhận xét:** Ị ếu xem vị từ P(x) như là một hàm lấy giá trị Bool trên miền xác định A của biến x thì ta có mệnh đề lượng từ hóa

$\forall x : P(x)$ là một mệnh đề đúng khi và chỉ khi P là hàm hằng 1.

Từ các qui tắc trên ta có thể chứng minh được một số tính chất suy diễn được phát biểu trong các mệnh đề sau đây:

- **Mệnh đề 1:** Cho p(x) và q(x) là các vị từ theo biến x lấy giá trị trong tập hợp A (miền xác định của biến x là tập hợp A), và a là một phần tử cố định tùy ý thuộc A. Khi ấy ta có qui tắc suy diễn sau đây:

  $\forall x : p(x) \to q(x)$

  $p(a)$

  _____

  $\therefore q(a)$

- **Mệnh đề 2:** Cho p(x), q(x) và r(x) là các vị từ theo biến x lấy giá trị trong tập hợp A (miền xác định của biến x là tập hợp A). Ta có qui tắc suy diễn sau đây:

  $\forall x : p(x) \to q(x)$

  $\forall x : q(x) \to r(x)$

  _____

  $\therefore \forall x : p(x) \to r(x)$

BÀI TẬP

1. Tìm chân trị của các vị từ ứng với các giá trị tương ứng sau:
   m(x): "x > 2"
   n(x): " x-1 lẻ "
   p(x): " x < 0 "
   a. m(2)
   b. $n(2^{10})$
   c. $m(-1) \vee \overline{n(3)}$
   d. $\overline{m(0) \wedge n(4)}$
   e. $[m(7) \to n(1)] \vee p(1)$
   f. $\overline{[m(-1) \vee n(-1)] \to p(-1)}$
   g. $[m(3) \leftrightarrow n(3)] \vee p(3)$
   h. $m(1) \to [n(1) \to p(1)]$

2. Xác định chân trị của vị từ sau:
   p(x,y) : "x là ước của y "

a. p(1,5)
b. p(3,4)
c. $\forall x, p(x,x)$
d. $\forall y, p(y,y)$
e. $\forall x \exists y, p(x,y)$
f. $\forall y \exists x, p(x,y)$
g. $\forall x \forall y, (p(x,y) \wedge p(y,x)) \rightarrow (x = y)$
h. $\forall x \forall y \forall z, (p(x,y) \wedge p(y,z)) \rightarrow p(x,z)$

3. Xác định chân trị của các vị từ sau:
m(x,y) : "2x > y-1"
n(x,y): "$y < x^2 + 1$"
a. m(1,3)
b. n(-1,2)
c. $m(2,3) \wedge n(-2,5)$
d. $\overline{m(3,2)} \rightarrow n(-1,1)$
e. $\overline{m(3,3) \rightarrow n(3,1)}$

4. Xác định chân trị của các mệnh đề sau :
a. $\exists x \in R, [(x^2 - 6x + 5 = 0) \rightarrow (x - 3 = 0)]$
b. $\forall x \in R, \exists y \in R, [(x - y = 2) \wedge (x + y = -3)]$
c. $\exists x \in R, \exists y \in R, \forall z \in R, [(x^2 = y^2 - z^2) \rightarrow (x \geq z)]$
d. $\exists x \in R, \exists y \in R, \exists z \in R, [(2x^2 = 2y^2 - z^2) \rightarrow (x < z)]$
e. $\exists x \in R, \exists y \in R, \forall z \in R, [(x - 2y \geq z) \vee (2x + y + z^2 \geq 0)]$
f. $\forall x \in R, \forall y \in R, \exists z \in R, [(x^2 = y^2 + z^2) \rightarrow (x > z)]$
g. $\forall x \in \mathbb{Z}, \exists y \in \mathbb{Z}, \exists z \in \mathbb{Z}, [(x - 2y = z) \wedge (2x + y + z = 0)]$
h. $\exists x \in R, \forall y \in R, \exists z \in R, [(x^2 = y^2 - z^2) \wedge (x \geq z)]$

5. Xác định các suy luận đúng và cho biết qui tắc suy diễn đã được áp dụng:
a. Mọi người đều quan tâm đến học vấn đều gia sức học tập để tích lũy kiến thức.
Huy chẳng cố gắng học tập.
Suy ra là Huy không quan tâm đến học vấn.
b. Ị ếu Bình đi chơi thì Bình không học logic toán. Ị ếu Bình không học bài thì sẽ thi trượt môn logic toán. Mà Bình lại đi chơi nên Bình thi trượt môn logic toán.
c. Mọi sinh viên lười học đều không chịu đến lớp học thường xuyên.
Tân đã đến lớp học thường xuyên.
Vì thế Vân là sinh viên không lười học.
d. Minh khẳng định rằng, nếu không được lĩnh lương dạy kèm thì phải kiếm việc khác làm. Mặt khác, nếu Minh nghỉ dạy kèm mà mẹ Minh không gửi tiền trợ cấp thì phải bán máy tính để đóng tiền học anh văn buổi tối. Cuối cùng Minh vẫn được trả lương nên Minh vẫn được tiếp tục học anh văn vào buổi tối.

e. Mọi sinh viên nghiêm túc đều không nộp bài chưa làm xong. Minh không nộp bài chưa làm xong. Vậy Minh là sinh viên nghiêm túc.
f. Mọi công dân quan tâm đến môi trường đều bỏ riêng túi nhựa bỏ đi vào một chỗ. Hùng không quan tâm đến môi trường. Vậy Hùng không bỏ riêng túi nhưa bỏ đi vào một chỗ.

6. Hãy phủ định rồi sau đó rút gọn các mệnh đề lượng từ hóa sau:

a. $\{(\forall x)[p_1(x) \to p_2(x)] \wedge (\forall x)[\overline{p_3(x)} \vee p_4(x)] \wedge (\forall x)[p_1(x) \vee p_3(x)]\} \to [\overline{p_2(x)} \to p_4(x)]$

b. $(\forall x)[\overline{p_2(x)} \to p_1(x)] \wedge (\forall x)[\overline{p_2(x)} \vee p_3(x)] \wedge (\forall x)[\overline{p_4(x)} \to \overline{p_3(x)}]\} \to (\forall x)[\overline{p_2(x)} \vee p_4(x)]$

c. $\{(\forall x)[\overline{p_2(x)} \to \overline{p_1(x)}] \wedge (\forall x)[\overline{p_2(x)} \to p_3(x)] \wedge (\forall x)[\overline{p_4(x)} \vee \overline{p_3(x)}]\} \vee (\forall x)[\overline{(p_2(x)} \wedge p_4(x)]$

7. Hãy chứng minh các công thức :

a. $0^2 + 1^2 + .. + n^2 = \dfrac{n(n+1)(2n+1)}{6}$

b. $0^3 + 1^3 + .. + n^3 = \dfrac{n^2(n+1)^2}{4}$

c. $1.2.3 + 2.3.4 + .. + n.(n+1)(n+2) = \dfrac{n(n+1)(n+2)(n+3)}{4}$

d. $\dfrac{1}{1.2.3} + \dfrac{1}{2.3.4} + .. + \dfrac{1}{n.(n+1)(n+2)} = \dfrac{n(n+3)}{4(n+1)(n+2)}$

8. Tìm nguyên dương n . Biết rằng trong 3 mệnh đề sau chỉ có duy nhất 1 mệnh đề sai :
   A = {n+51 là một số chính phương}
   B = {n có số tận cùng là 1}
   C = {n - 38 là một số chính phương}

9. Cho 2 số nguyên dương m và n. Biết rằng trong số 4 mệnh đề sau chỉ có duy nhất 1 mệnh đề sai:
   A = {m = 2n + 5}
   B = {m + 1 chia hết cho n}
   C = {m + n chia hết cho 3}
   D = {m + 7n là số nguyên tố}
   a. Hãy chỉ ra mệnh đề sai trong các mệnh đề trên.
   b. Hãy tìm tất các cặp (m, n) thõa các mệnh đề đúng còn lại.

10. Một giải cờ vua có n kỳ thủ tham dự thi đấu vòng tròn 1 lượt tính điểm. Trong mỗi trận thì người thắng được 2 điểm, hòa được 1 điểm, thua thì không có điểm. Các kỳ thủ có cùng số điểm sẽ xét thêm các chỉ số phụ nào đó. Khi kết thúc thì kỳ thủ hạng 1 được 8 điểm, kỳ thủ hạng 2 được 6 điểm và kỳ thủ hạng 3 được 5 điểm. Các kỳ thủ còn lại có số điểm khác nhau. Hãy cho biết có bao nhiêu kỳ thủ tham dự, và số điểm của họ bao nhiêu?

BÀI 6: NGÔN NGỮ PROLOG

1. Tư duy lập trình và định nghĩa vấn đề trên Prolog

Đối với Prolog, một chương trình có thể hiểu như là các tri thức được người lập trình cung cấp cho hệ thống Prolog. Ị hờ vào các kiến thức được cung cấp, hệ thống có thể trả lời được các câu hỏi được đặt ra, và câu trả lời có thể đạt được nhờ cơ chế suy luận của hệ thống dựa trên những kiến thức được cung cấp ban đầu.

Đơn vị kiến thức mà người lập trình cung cấp cho Prolog gọi là các vị từ (predicate). Các vị từ dùng để biểu diễn các khái niệm mà người lập trình muốn hệ thống dùng để suy luận để đạt được các kiến thức khác mà mình mong muốn. Về mặt kỹ thuật, các predicate có thể được xem như các hàm, nhưng giá trị trả về chỉ có thể là các giá trị luận lý - đúng hoặc sai. Và giá trị trả về này chỉ có thể sử dụng để suy luận, *Prolog không có cơ thế chồng chất hàm như các ngôn ngữ thủ tục* khác, chính điều này sẽ làm những người quen với việc lập trình thủ tục gặp khó khăn khi bước đầu lập trình với Prolog. Công việc đầu tiên khi lập trình trên Prolog là định nghĩa các vị từ - các khái niệm mà mình cần cung cấp cho chương trình.
Xét các ví dụ sau:

VD1:

Dữ kiện ban đầu: Mọi người đều phải chết. Socrates là người.
Yêu cầu:

Chúng ta muốn hệ thống phải có khả năng suy luận và trả lời được các vấn đề liên quan đến các khái niệm trên: ai là người, ai không là người, ai phải chết, ai không phải chết. Ở đây chúng ta có một sự suy luận thông minh đặc trưng cho sức mạnh của Prolog: hệ thống sẽ tự động suy luận rằng Socrates phải chết (điều không được cung cấp ban đầu).

Để biểu diễn các vấn đề trên bằng ngôn ngữ Prolog, chúng ta cần phải xác định cần phải *biểu diễn những khái niệm gì*. Trong vấn đề này chúng ta có hai khái niệm cần biểu diễn: một thực thể nào đó có
thể là người (hoặc không), và một thực thể nào đó có thể chết.

Ị hư vậy chúng ta biểu diễn vấn đề đầu tiên bằng ngôn ngữ Prolog như sau:
nguoi(symbol)

Symbol là một kiểu dữ liệu đặc biệt của Prolog, dùng để biểu diễn cho một thực thể, một khái niệm tổng quát. Chúng ta sẽ trở lại vấn đề này sau. Ị hư vậy chúng ta vừa định nghĩa một khái niệm: một symbol nào đó có thể là người, một symbol nào khác thì không.
Hiểu như một sự định nghĩa hàm, chúng ta có thể xem như định nghĩa một hàm mang tên nguoi, hàm này có thông số một biến thuộc kiểu dữ liệu symbol, và kết quả của hàm này, không cần phải khai báo thuộc về kiểu gì, vì chỉ có thể thuộc kiểu boolean, chỉ có thể đúng hoặc sai. Ị hiệm vụ của Prolog là phải trả lời được với giá trị symbol nhập vào, thì hàm này

cho ra kết quả đúng hoặc sai, tức symbol ấy có phải là người hay không. Prolog chỉ có thể làm được điều này nếu như nếu như chúng ta cung cấp cho hệ thống một cơ chế suy luận đúng đắn, tức là giải thích được cho Prolog hiểu *như thế nào là người?*
Tương tự như vậy, chúng ta định nghĩa về vấn đề một thực thể nào đó phải chết bằng vị từ sau
chet(symbol)
Ị hư vậy với bài toán đã nêu, chúng ta sẽ đặt ra hai vị từ
nguoi(symbol)
chet(symbol)

VD2:

Yêu cầu: tính giá trị giai thừa của một số nguyên bất kỳ.

Bài toán trên không cho biết dữ kiện ban đầu. Chúng ta phải cung cấp các dữ kiện ban đầu, để Prolog có thể dựa vào đó để suy luận, để từ đó hệ thống có thể giải quyết được yêu cầu của chúng ta. Việc cung cấp dữ kiện ban đầu cho hệ thống là rất quan trọng quyết định vấn đề giải quyết yêu cầu của chúng ta. Một trong những cách giải quyết có thể được lựa chọn là chúng ta sẽ cho hệ thống biết giá trị giai thừa của toàn bộ số nguyên: giai thừa của 0 là 1, giai thừa của 1 là 1, giai thừa của 2 là 2, giai thừa của 3 là 6, giai thừa của 4 là 24… Dễ dàng nhận thấy rằng cách này là không khả thi, và trong thực tế, *con người cũng không tiếp thu tri thức theo cách này*.

Chúng ta có thể cung cấp dữ kiện cho hệ thống theo cách khác: giai thừa của một số là tích các số từ 1 đến số đó. Ị hư vậy với cách giải quyết này, chúng ta có hai khái niệm cần phải cung cấp: giai thừa của một số là gì, và tích của các số nguyên tính từ 1 đến một số là gì?

Cách đặt vấn đề này có thể giải quyết được bài toán, tuy nhiên chúng ta có thể đặt vấn đề theo một cách khác đơn giản, và hợp với tinh thần của Prolog hơn: giai thừa của 0 là 1, và giai thừa của một số lớn hơn 0 là giai thừa của số liền trước nó nhân với chính nó.

Với cách đặt vấn đề này, chúng ta chỉ có một khái niệm phải biểu diễn: giai thừa của một số là gì? (thật ra chúng ta còn một số khái niệm phải đưa ra: một số đứng trước một số là gì, nhân hai số nghĩa là gì, tuy nhiên Prolog đã cung cấp các toán tử để giải quyết vấn đề này. Hiểu theo một nghĩa nào đó, các vấn đề trên là các tiên đề, không cần phải giải thích với hệ thống.)

Ị ếu quen với ngôn ngữ lập trình thủ tục, chúng ta có khuynh hướng khai báo vị từ diễn tả khái niệm giai thừa như sau:
giaithua(integer)
Ở đây cách đặt vấn đề như vậy là không thích hợp với ngôn ngữ Prolog, vì. Một vị từ chỉ có thể trả lời là đúng hoặc sai, trong khi chúng ta đang mong muốn kết quả trả về theo cách khai báo này một số. Ị gôn ngữ Prolog không có sự chồng chất hàm, nghĩa là kết quả của hàm (vị từ) không thể dùng như một thông số cho một vị từ khác, trong khi chúng ta đang định dùng kết quả của hàm này để tính tiếp giá trị cho một hàm khác.(Chúng ta định dùng hàm này để tính giai thừa của n -1 , rồi nhân tiếp cho n để ra kết quả cuối cùng).

Vị từ thích hợp sẽ được khai báo như sau:
giaithua(integer,integer)

Điều này, hiểu theo ngôn ngữ thủ tục, nghĩa là chúng ta khai báo một hàm có thông số là hai số nguyên, và kết quả trả về sẽ là đúng hoặc sai. Điều chúng ta muốn diễn tả có nghĩa là: giai thừa của một số nguyên (integer) sẽ là một số nguyên khác.

Ị ếu chúng ta giải thích được cho Prolog hiểu giai thừa của một số nguyên sẽ được tính như thế nào, hệ thống sẽ có khả năng trả lời cho cả câu hỏi thuận (giai thừa của một số nguyên là gì), câu hỏi nghịch (số nguyên nào có giai thừa bằng số nguyên này), và nghi vấn (giai thừa của một số nguyên X có phải là số nguyên Y hay không).

Tuy nhiên mục đích của chúng ta chỉ cung cấp các dữ kiện để hệ thống có thể trả lời câu hỏi thuận (và có thể trả lời thêm câu hỏi nghi vấn) mà thôi.

Tóm tắt:

- Lập trình trên Prolog là cung cấp cho hệ thống các khái niệm và diễn giải các khái niệm đó.
- Các khái niệm được cung cấp qua các vị từ.
- Các vị từ có thể xem như các hàm như chỉ trả về giá trị đúng hoặc sai.
- Việc hệ thống có thể trả lời được những câu hỏi nào liên quan đến khái niệm đã cung cấp phụ thuộc vào việc chúng ta diễn giải các khái niệm trên cho hệ thống

2. Các clause, cách giải thích các vấn đề trên Prolog

Sau khi đã cung cấp cho hệ thống các khái niệm cần thiết, chúng ta cần phải giải thích các khái niệm mình đã cung cấp, Prolog sẽ dùng các lời giải thích này để thực hiện việc suy luận và trả lời câu hỏi của chúng ta. Các lời giải thích này được gọi là các mệnh đề (clauses).
Có hai dạng mệnh đề: sự kiện (fact), và luật(rule).
Các sự kiện là những điều mà chúng ta công nhận là đúng. Luật là những quy tắc mà chúng ta xác định điều kiện đúng cho chúng.

VD3: hãy viết phần clause cho vị từ *nguoi* đã định nghĩa trong VD1

Dữ kiện ban đầu chỉ cung cấp cho chúng ta một vấn đề liên quan đến người: Socrates là người. Theo như cách tư duy trong không gian của bài toán, chỉ có một con người duy nhất: Socrates. Không ai khác là người. Ị hư vậy chúng ta sẽ viết phần clause cho vị từ này như sau:
nguoi(socrates).
Chúng ta vừa viết một sự kiện: socrates là người là điều chắc chắn đúng. Bất kỳ symbol nào có tên là socrates là người là chắc chắn đúng, không cần phải có một điều kiện ràng buộc nào kèm theo.

Lưu ý:

i/Có hai cách viết dạng hằng (literal) cho symbol trên Prolog:

- Một danh hiệu mở đầu bằng ký tự thường (socrates, sOCRATES…)

- Một chuỗi ký hiệu đặt trong cặp ký hiệu "," ("socrates","SOCRATES"," sOCRATES", "Socrates"…)

ii/ một mệnh đề luôn kết thúc bằng ký tự '.'

VD4: hãy viết phần clause cho vị từ *chet* trong VD1.

Dữ kiện ban đầu chỉ cung cấp cho chúng ta một sự kiện liên quan đến vấn đề này: symbol sẽ phải chết nếu (và chỉ nếu) đó là người. Điều này sẽ xác định một quy tắc: symbol sẽ chỉ phải chết, tức vị từ sẽ trả về kết quả true, nếu symbol đó là người. Vấn đề symbol nào là người và symbol nào không là người chúng ta đã đưa ra khái niệm và giải thích cho Prolog trong các ví dụ 1 và 3.

Ị hư vậy phần mệnh đề sẽ được viết như sau:
chet(X):-nguoi(X).

Mệnh đề dạng rule sẽ bao gồm hai phần, nằm ở hai bên cặp ký hiệu ":-". Phần bên trái cho biết vị từ đang được đề cập và các thông số tương ứng. Phần bên phải, xác định điều kiện trả lời đúng cho luật trên, bao gồm các lời gọi các vị từ khác, được ngăn cách bởi ký hiệu ',', gọi là các mệnh đề con (sub-clause). Trong ví dụ trên, chỉ có một sub-clause. *Một luật chỉ trả lời đúng nếu tất cả các sub-clause bên vế phải đều trả lời đúng.*

Trong ví dụ trên, chúng ta có một biến X. *Tất cả các thông số mở đầu bằng ký tự hoa đều được Prolog hiểu là biến*. Biến này là thông số của vị từ *chet.* Do đã khai báo ở phần vị từ, X sẽ được hiểu là một biến thuộc kiểu symbol. Kết quả sẽ trả về đúng nếu tất cả sub-clause bên vế phải đều trả lời là đúng. Trong trường hợp này, chỉ có một sub-clause xác định xem X có phải là người không. Ị hư vậy chúng ta đã biểu diễn được khái niệm một symbol sẽ phải chết nếu symbol đó là người, tức là tất cả những dữ kiện ban đầu được cung cấp.

VD5: Hãy viết phần clause cho vị từ *giaithua* ở VD2.

Từ các dữ kiện được cung cấp (do chúng ta tự cung cấp cho mình để giải bài toán), chúng ta thấy có một sự kiện chắc chắn đúng: giai thừa của 0 là 1, và có một luật suy diễn: giai thừa của n là (n-1)!*n.

Chúng ta sẽ viết phần mệnh đề cho vị từ này như sau:
giaithua(0,1).
*giaithua(X,Y) -: X1 = X -1, giaithua(X1,Y1), Y = Y1*X.*

Trước khi hiểu những điều được mô tả trong các ví dụ trên, chúng ta sẽ có một số nhận xét như sau:

- Trước tiên, chúng ta thấy vị từ *giaithua* được biểu diễn bằng hai mệnh đề: một sự kiện và một luật. Khi viết nhiều mệnh đề cho một vị từ, *các mệnh đề phải được viết liên tiếp nhau (không được xen mệnh đề của vị từ khác vào).*

- Chúng ta sẽ hiểu hai mệnh đề con đầu tiên *X1 = X -1, giaithua(X1,Y1)* biểu diễn cho công việc tính giai thừa của X-1. Tuy nhiên chúng ta không được viết *giaithua(X-1,Y1). Thông số của các mệnh đề con phải là biến, không được phép là biểu thức.*
- Chúng ta thấy sự xuất hiện của ký hiệu '= ' và sẽ hiểu như mệnh đề con X1 = X-1 là phép gán. *Trên Prolog không có phép gán.* Ký hiệu '=' biểu diễn cho một phép toán đặc biệt của Prolog, phép hợp nhất (unification), và cũng như các mệnh đề khác, phép hợp nhất này sẽ trả về kết quả đúng hoặc sai, biểu diễn cho kết quả công việc hợp nhất thành công hay không. Xem thêm về phép hợp nhất trong các phần sau.
- Phần vị từ trên biểu diễn cho việc sử dụng kỹ thuật lập trình đệ quy, sẽ là sức mạnh lập trình chủ yếu của Prolog. Xem thêm về phần lập trình đệ quy trên Prolog trong các phần sau.

Tóm tắt

- ❖ Các khái niệm được mô tả qua các vị từ sẽ được giải thích bằng các mệnh đề.
- ❖ Có hai loại mệnh đề: sự kiện và luật.
- ❖ Thông số được truyền trong lời gọi các mệnh đề con phải là biến.
- ❖ Các kỹ thuật chủ yếu để lập trình trên Prolog là hợp nhất và đệ quy.

3. Thực thi chương trình. - Đặt câu hỏi và nhận câu trả lời

Đến đây chúng ta đã có thể thực thi các chương trình trên. Trên cửa sổ Prolog, nhập nội dung đoạn chương trình vào vùng cửa sổ soạn thảo. (Chúng ta có thể luôn chuyển về vùng cửa sổ này bằng phím nóng Alt+E).

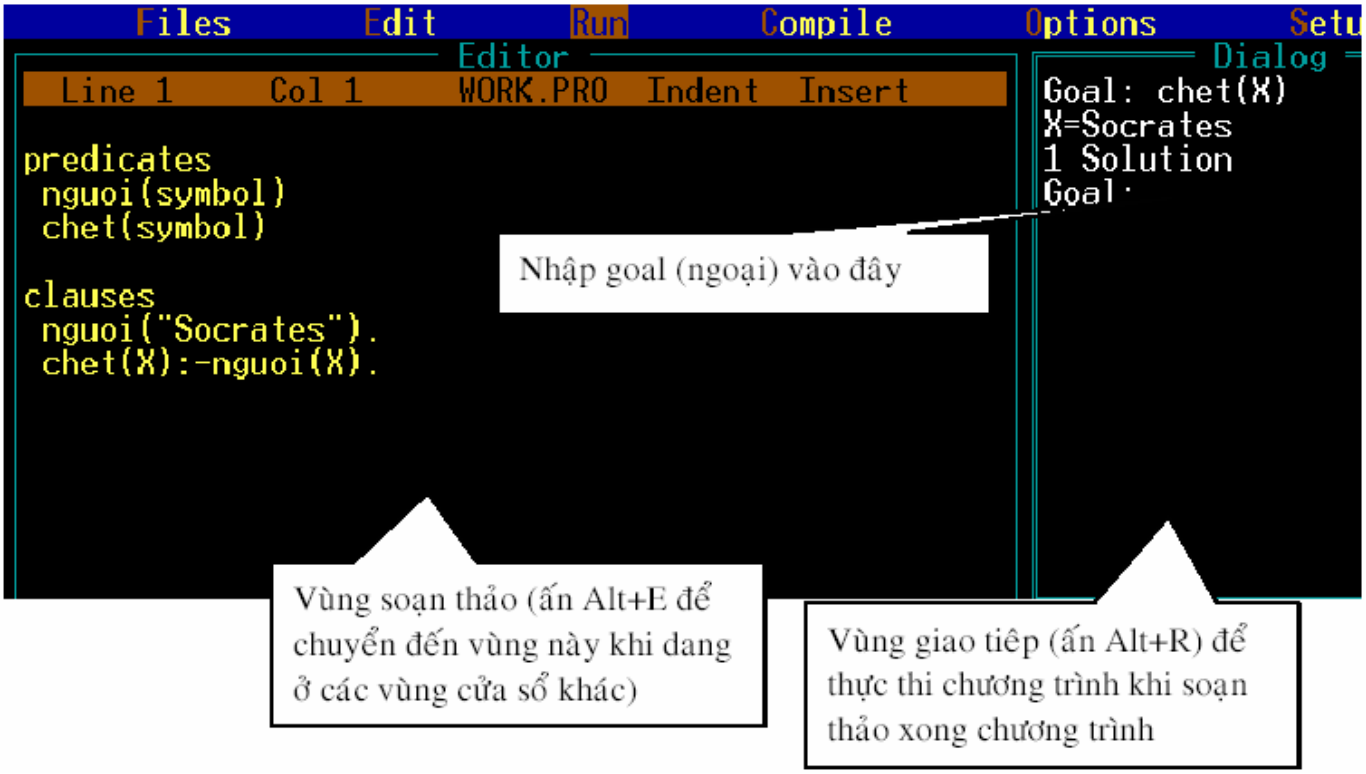
VD6: Viết chương trình hoàn chỉnh cho VD1.

Ị ội dung chương trình nhập hoàn chỉnh cho VD1 như sau:

```
predicates
nguoi(symbol)
chet(symbol)
clauses
nguoi("Socrates").
chet(X):-nguoi(X).
```

Ị hư vậy chúng ta thấy trong một chương trình Prolog, các phần khai báo các vị từ và hiện thực các mệnh đề được bắt đầu bằng các từ khoá *predicates* và *clauses*. Lưu ý: Phần predicates phải được viết trước phần dành cho clauses. Chúng ta sử dụng Turbo Prolog để thử các ví dụ trên:

Giao diện chương trình sẽ như sau:



Để thực thi chương trình, người sử dụng nhập yêu cầu (câu hỏi) của mình cho hệ thống.Yêu cầu này gọi là goal. Có hai loại goal:goal nội và goal ngoại. Phần này chỉ trình bày về goal ngoại. Để nhập goal ngoại, sau khi đã hoàn tất việc soạn thảo chương trình, chúng ta dùng phím tắt Alt+R để chuyển sang vùng giao tiếp của chương trình. Câu hỏi chúng ta đặt ra cho hệ thống phải chỉ dựa vào các tri thức mà chúng ta đã cung cấp cho hệ thống. Chúng ta đã cung cấp cho hệ thống các khái niệm *nguoi* và *chet*, như vậy chúng ta chỉ có thể đặt các câu hỏi liên quan đến hai khái niệm này. Ị gay sau dấu nhắc *Goal:* tại vùng cửa sổ này, chúng ta có thể nhập câu hỏi như sau: *nguoi("Socrates")*

Dựa trên tinh thần của của khái niệm, câu phát biểu của chúng ta có nghĩa là "Socrates là người", và vì đây là câu phát biểu trong vùng goal, nên hệ thống sẽ hiểu rằng chúng ta muốn đặt một câu hỏi nghi vấn "Socrates là người phải không?"

Sau khi ấn Enter, chúng ta sẽ thấy hệ thống có ngay câu trả lời: Yes.
Thay bằng một tên khác, ví dụ:
nguoi("Xeda")
Hệ thống sẽ trả lời Ị o.

Chúng ta thấy các câu trả lời của hệ thống dựa trên kiến thức mà chúng ta đã cung cấp. Dựa vào những gì mà chúng ta đã cung cấp, hệ thống chỉ biết có một người là Socrates, tất cả những symbol khác đều không phải là người.

Tuy nhiên, với cơ chế suy luận mà chúng ta cung cấp, hệ thống có thể suy luận ra những điều chưa được cung cấp sẵn. Đây chính là điểm tạo nên sức mạnh lập trình của Prolog.

Ị hập vào goal như sau:
chet("Socrates")
Câu trả lời là: Yes.
Với một tên người khác:
chet("Xeda")
Câu trả lời là: Ị o.

Hệ thống đã tự động suy luận theo nguyên lý mà chúng ta muốn nó phải "học": ai là người thì người đó phải chết. Ị goài những câu hỏi dạng Yes/Ị o, Prolog có thể trả lời các câu hỏi yêu cầu tìm đáp số.

Chúng ta nhập vào một goal như sau:
chet(X)

Đến đây, trong câu hỏi của chúng ta có một biến: X (nhắc lại: mọi danh hiệu mở đầu là ký tự hoa đều là biến). Khi trong câu hỏi của chúng ta chứa một (hoặc nhiều) biến, hệ thống sẽ tìm các giá trị có thể có của biến để cho câu phát biểu của ta là đúng. Hiểu ở mức ý niệm, câu hỏi của chúng ta là: ai là người? Kết quả trả lời của câu hỏi (ai) sẽ được chứa trong biến X.
Câu trả lời sẽ là: X = Socrates

Tương tự như trên, hệ thống sẽ dựa vào cơ chế suy luận đã được cung cấp để tìm ra lời giải với những câu hỏi dành cho các vị từ có các mệnh đề tương ứng là các luật. Ị hập vào goal như sau:
chet(X)
Hệ thống sẽ trả lời như sau:
X = Socrates.

VD7: Hoàn chỉnh và thực thi chương trình cho VD2

Chương trình hoàn chỉnh như sau:
predicates
giaithua(integer,integer)
clauses
giaithua(0,1).
*giaithua(X,Y):- X1 = X -1, giaithua(X1,Y1), Y = X*Y1.*

Chúng ta lưu ý là khi kết thúc mỗi mệnh đề đều có ký hiệu '.'
Chúng ta có thể đặt cho hệ thống goal dạng nghi vấn như sau:
giaithua(3,6)

Hiểu theo ngôn ngữ tự nhiên sẽ là: có phải giai thừa của 3 là 6 hay không?
Câu trả lời là: Yes

Hoặc chúng ta có thể đặt câu hỏi:
giaithua(3,8)

Câu trả lời sẽ là: Ị o.

Chúng ta sẽ đặt câu hỏi theo dạng tìm lời giải:
giaithua(3,X)
Câu trả lời sẽ là X = 6

Chúng ta cũng có thể đặt câu hỏi ngược:
giaithua(X,6)
Ý tưởng của câu hỏi sẽ là: giai thừa của số nào sẽ bằng 6.

Tuy nhiên chúng ta không cung cấp cho hệ thống cơ chế suy luận để trả lời câu hỏi này. Hệ thống sẽ trả lời: Ị o Solution.

Tất nhiên chúng ta có thể đặt câu hỏi như sau:
giaithua(X,Y)

Cả hai thông số đều là biến. Ị hư vậy câu hỏi có thể hiểu là: số nào (X) giai thừa thì thành một số khác (Y). Câu hỏi gần như vô nghĩa và những câu trả lời của hệ thống cũng sẽ chẳng mang một ý nghĩa thực sự có nghĩa nào.

Tóm tắt:

- Chương trình Prolog sẽ hoạt động theo cơ chế tương tác. Ị gười sử dụng sẽ cung cấp yêu cầu, gọi là goal, và hệ thống sẽ trả lời các yêu cầu này.
- Có hai loại goal: goal nội và goal ngoại.
- Ị ếu goal không chứa biến thì hệ thống sẽ kiểm tra phát biểu của chúng ta là đúng hoặc sai, ngược lại, hệ thống sẽ tìm các giá trị của các biến làm cho phát biểu của ta là đúng.

4. Phép hợp nhất - Cơ chế tìm câu trả lời của Prolog.

4.1/ Phép hợp nhất

Công việc quan trọng nhất của Prolog trong việc tìm câu trả lời là thực hiện việc hợp nhất. Phép hợp nhất được biểu diễn bằng dấu =, và nó có hai thành phần, tạm gọi là vế trái vế phải. Phép hợp nhất sẽ trả về kết quả true hoặc false.

Có các trường hợp hợp nhất sau:

a. Cả hai vế đều là hằng hoặc biểu thức chứa toàn hằng. Ị ếu giá trị của hai vế là bằng nhau thì phép hợp nhất thành công (đáp số là true), ngược lại phép hợp nhất sẽ thất bại (kết quả là false)
   7 = 7 _ true
   7 = 8 _ false
   "abc" = "abc" _ true
   "abcd" = "abc" _ false

7 = 6 +1 _ true
6 = 7 +1 _ false

b. Một trong hai vế là hằng hoặc trong biểu thức chứa toàn hằng, vế kia là biến hoặc biểu thức có chứa biến.

Trường hợp 1: Ị ếu tất cả các biến đều có giá trị (gọi là các biến ở tình trạng bound), chúng ta quay về trường hợp a
7 = X _ false nếu X đã có giá trị là 6
7 = X +1 _ true nếu X đã có giá trị là 6
Y = "Socrates" _ true nếy Y đã có giá trị là "Socrates"

Trường hợp 2: Ị ếu có biến chưa có giá trị (gọi là biến ở tình trạng unbound), Prolog sẽ gán giá trị cho biến sau cho hai vế có giá trị như nhau và trả về kết quả là true. Ị ếu không tìm giá trị như vậy, phép hợp nhất sẽ cho kết quả là false.
7 = X _ true nếu X chưa có giá trị, sau phép hợp nhất này, X sẽ có giá trị là 7-1 = X*X _ false vì không thể tìm cho X giá trị nào làm cho giá trị hai vế là như nhau.

c. Cả hai vế đều là biến hoặc các biểu thức có chứa biến

Trường hợp 1: tất cả các biến đều có chứa giá trị, chúng ta sẽ quay về trường hợp a
X = Y _ true nếu cả X và Y đều đã có giá trị và những giá trị này bằng nhau X -1 = Y _ false nếu X và Y đều đã có giá trị và X nhỏ hơn Y.
Trường hợp 2: tất cả các biến của một vế đều đã có giá trị, chúng ta sẽ quay về về trường hợp b
X = Y _ true nếu X chưa có giá trị và Y đã có giá trị, sau phép hợp nhất, X sẽ nhận giá trị của Y
X - 1 = Y _ true nếu X chưa có giá trị, Y đã có giá trị. Sau phép hợp nhất, X sẽ có giá trị bằngg Y +1
Trường hợp 3: cả hai vế đều còn chứa biến ở tình trạng unbound _ hợp nhất thất bại
X = Y _ false nếu cả X và Y đều chưa gán giá trị
X-1 = Y _ false nếu cả X và Y đều chưa gán giá trị

4.2/ Cơ chế tìm câu trả lời của Prolog

Ị ếu chúng ta đặt ra cho Prolog một câu hỏi, Prolog sẽ thực hiện công việc so trùng (match), tức là tìm mệnh đề đầu tiên đề cập đến khái niệm mà chúng ta muốn hỏi.

Trở lại VD6, sau khi đã hoàn tất chương trình, chúng ta đặt ra Goal như sau:
nguoi("Socrates")

Prolog sẽ tìm mệnh đề đầu tiên có liên quan đến khái niệm *nguoi*. Hiển nhiên, mệnh đề đầu tiên (và duy nhất có liên quan đến khái niệm này là:
nguoi("Socrates")

Ị hư vậy, khi đã có goal (*nguoi("Socrates")*) và tìm thấy mệnh đề liên quan (*nguoi("Socrates")*), Prolog sẽ tiến hành tìm kiếm lời giải, công việc này tiến hành bằng cách tạo mối liên kết giữa các thông số ở phần goal và các thông số ở phần mệnh đề. Có các trường hợp sau:

a. Cả hai thông số này đều là các biến unbound, trong trường hợp này Prolog sẽ xem cả hai thông số là 1.
b. Ở tất cả các trường hợp khác, Prolog sẽ tiến hành phép hợp nhất giữa hai loại thông số.

Sau khi đã tạo mối quan hệ giữa các thông số ở phần goal và phần clause, Prolog sẽ tiến hành các sub-clause (nếu mệnh đề này một luật). Ị ếu tất cả các sub-clause thành công và các biến ở phần goal đã ở tình trạng bound (tức là đã có giá trị), Prolog sẽ thông báo lời giải. Ị ếu là câu hỏi thuộc dạng Yes/Ị o như ví dụ trên, tức là câu hỏi không chứa biến, Prolog sẽ trả lời Yes nếu công việc hợp nhất như đã nói ở phần b thành công và các sub-clause đều thành công (nếu mệnh đề so trùng là một luật).

Quay trở lại với ví dụ của chúng ta, ở đây thông số của Goal là một hằng ("Socrates), và thông số của mệnh đề tương ứng cũng là một hằng ("Socrates), hai hằng này hợp nhất thành công, và kết quả trả lời là Yes.

Ị ếu chúng ta đặt ra câu hỏi khác:
nguoi("Xeda")
Prolog cũng chỉ tìm thấy một mệnh đề liên quan đến khai niệm này (*nguoi("Socrates")*), và vì sự hợp nhất giữa hai hằng "Socrates" và "Xeda" thất bại, đáp số sẽ trả lời là Ị o.

Chúng ta xét trường hợp câu hỏi của chúng ta có chứa biến:
nguoi(X)
Hệ thống sẽ tìm thấy mệnh đề có liên quan đến vấn đề này (*nguoi("Socrates")*) , vàtiến hành hợp nhất giữa X và "Socrates", và vì X chưa có giá trị (unbound) nên phép hợp nhất thành công, X có giá trị là "Socrates".

Vì việc hợp nhất giữa các thông số giữa phần goal và phần clause đã thành công, đây là một sự kiện nên không cần phải thực hiện phần sub-clause, và sau khi hợp nhất, tất cả các biến cần tìm đã có giá trị (ở đây chỉ có một biến là X), nên hệ thống sẽ công bố đã tìm ra lời giải và in ra giá trị của X (X = "Socrates")

Chúng ta xét trường hợp khi ở câu hỏi phần goal so trùng với một luật:
chet(Y)

Chúng ta hoàn toàn có thể đặt câu hỏi là *chet(X)*, nhưng chúng ta sẽ đặt tên biến khác để tiện phân biệt giữa biến trong câu hỏi của goal và thông số cục bộ ở mệnh đề.

Câu hỏi trong goal được so trùng với mệnh đề sau:
chet(X): - nguoi (X).

Vì hai biến X(thông số của mệnh đề) và Y(thông số của goal) đều chưa chứa giá trị, hệ thống sẽ xem cả hai biến là một, tức là, khi X có được giá trị thì Y cũng có giá trị đó và ngược lại. Do đây là một luật, nên hệ thống sẽ tiến hành thực hiện các sub-clause. Hệ thống sẽ thực hiện sub-clause đầu tiên *nguoi(X)*.

Quá trình thực hiện các sub-clause ở vế phải sẽ được thực hiện như sau:

a. Ị ếu sub-clause này có thông số là biến unbound, Prolog sẽ tìm giá trị của biến này để sub-clause có giá trị Yes, nếu không tìm được giá trị như vậy, sub-clause sẽ thất bại.
b. Ị ếu sub-clause có thông số đều là biến bound (đã có giá trị) hoặc là hằng, Prolog sẽ kiểm tra xem sub-clause có trả về giá trị Yes hay không, nếu không, subclause sẽ thất bại.

Các sub-clause sẽ được tiến hành từ trái qua phải, và nếu có một sub-clause thất bại, mệnh đề được so trùng sẽ thất bại.

Trong trường hợp trên, khi tiến hành sub-clause *nguoi(X)*, do biến X là unbound, nên chúng ta rơi vào trường hợp a, hệ thống sẽ tìm giá trị của X cho sub-clause trên là đúng. Cách tìm kiếm câu trả lời cho sub-clause này hòan tòan giống như cách hệ thống tìm câu trả lời khi chúng ta đặt câu hỏi này trong phần goal, và như vậy X sẽ có giá trị là "Socrates" sau khi sub-clause này thực hiện xong.

Do X và Y được xem như một, nên khi X có giá trị là "Socrates" thì Y cũng có giá trị này.
Do tất cả các sub-clause đã thực hiện xong, và Y đã có giá trị, nên Prolog công bố là đã tìm ra lời giải và in ra giá trị của Y.

Tóm tắt:

- Phép hợp nhất là nền tảng của mọi hoạt động của Prolog để tìm ra lời giải. Để trả lời câu hỏi, Prolog so trùng câu hỏi với mệnh đề và tạo mối liên quan giữa các thông số.
- Prolog tìm ra lời giải khi thực hiện thành công một mệnh đề và tất cả các biến nếu có trong các thông số của goal đều đã có giá trị.

5. Sự quay lui - Khống chế số lượng lời giải -Vị từ nhát cắt và fail

Goal nội và goal ngoại (internal goal và external goal)

Khi chúng ta sử dụng Alt-R để chuyển sang cửa sổ giao tiếp và nhập vào goal, goal này gọi là goal ngoại. Chúng ta có thể thêm phần goal này hẳn trong phần soạn thảo chương trình, goal này gọi là goal nội.

VD8: Viết lại chương trình giải quyết VD6 sử dụng goal nội

Chương trình được viết lại như sau:
predicates
nguoi(symbol)
chet(symbol)
clauses
nguoi("Socrates").
chet(X):-nguoi(X).
goal
nguoi(X),write(X)

Trong ví dụ này, chúng ta đã thêm phần goal vào trong chương trình. Khi thực thi, hệ thống sẽ không hỏi goal nữa, và tự động thực hiện các yêu cầu trong phần goal. Tuy nhiên khi thực hiện goal nội, hệ thống sẽ không tự động in kết quả nữa. Chúng ta phải gọi vị từ *write* để làm điều này. Vị từ này sẽ cho kết quả là đúng nếu các thông số nhập vào là đều là biến ở trạng thái bound hoặc là hằng.

Sự quay lui (back-tracing) trên Prolog

Hợp nhất là hòn đá nền tảng cho cơ chế suy luận của Prolog, tuy nhiên, để tìm ra lời giải đúng, Prolog phải sử dụng cơ chế quay lui, khi giá trị đầu tiên được gán cho thông số không phải là lời giải.

Chúng ta xét ví dụ sau:

VD9:
predicates
nguoi(symbol)
20
vua(symbol)
sungsuong(symbol)
clauses
nguoi("Socrates").
nguoi("Xeda").
vua("Xeda).
sungsuong(X) :- nguoi(X),vua(X).

Ị hư vậy trong ví dụ này, ngoài khái niệm về người, chúng ta đưa ra khái niệm về vua và sự sung sướng. Diễn giải những thông tin trong các dữ kiện trên thành ngôn ngữ tự nhiên, chúng ta có được các điều sau: "Thế giới mà chúng ta sống có hai người là Socrates và Xeda. Chúng ta có một vua la Xeda, và một thực thể nào đó chỉ sung sướng nếu thực thể đó vừa người vừa là vua." Lưu ý rằng trong ví dụ trên, các mệnh đề liên quan đến cùng một vị từ phải viết liên tiếp nhau.

Xét khi hệ thống trả lời câu hỏi sau:
sungsuong(X)

Trước tiên hệ thống sẽ so trùng goal trên với mệnh đề *sungsuong(X) :-nguoi(X),vua(X)*. Lưu ý rằng vào lúc này chúng ta có hai biến X: một biến X là thông số của goal và một biến X là thông số của mệnh đề. Về nguyên tắc, *hai biến X này hoàn toàn khác nhau*.

Tuy nhiên, khi so trùng goal với mệnh đề, do cả hai biến X lúc này đều chưa chứa giá trị, nên chúng sẽ được xem như một. Ị hưng cần chú ý rằng *biến X sử dụng trong các sub-clause là biến X thông số của mệnh đề.*

Sau đó Prolog sẽ tiến hành các sub-clause. Ở sub-clause đầu tiên, *nguoi(X)*, tương tự như VD6, Prolog sẽ tìm được câu trả lời là X = Socrates. Khi thực hiện sub-clause thứ hai, *vua(X)*, do X đã có giá trị (Socrates), Prolog sẽ kiểm tra xem giá trị này có làm giá trị của mệnh đề là true hay không.

Ị hư các ví dụ trên, việc tiến hành trả lời một sub-clause cũng tương tự như khi trả lời một goal, Prolog lại so trùng sub-clause với một mệnh đề cùng tên. Prolog tìm thấy một mệnh đề liên quan đến *vua* là *vua("Xeda")* và tiến hành hợp nhất giữa X và Xeda. Do X đã có giá trị là Socrates, việc hợp nhất thất bại.

Tuy nhiên khi sub-clause này thất bại, không có nghĩa rằng Prolog sẽ vội kết luận rằng mệnh đề này thất bại. Ở đây công việc tìm kiếm câu trả lời thất bại sau khi biến X được gán giá trị và chuyển từ trạng thái bound sang unbound. Hệ thống sẽ quay lại thời điểm biến X được gán giá trị (khi trả lời sub-clause *nguoi(X)*) , X được chuyển lại sang tình trạng unbound, và cố gắng tìm kiếm một giá trị khác của X để cho mệnh đề con này vịn đúng. Công việc này được gọi là back-tracing.

Do việc so trùng sub-clause này với mệnh đề *nguoi("Socrates")* thất bại, hệ thống sẽ so trùng với mệnh đề khác. *Nếu không còn mệnh đề nào khác liên quan đến subclause, việc thực hiện mệnh đề mới thật sự thất bại*, tuy nhiên ở đây hệ thống tìm thấy một mệnh đề khác liên quan đến khái niệm này là *nguoi("Xeda")*. Việc hợp nhất giữa X và "Xeda" lại được thực hiện, X sẽ có giá trị là Xeda và sau đó, khi lại tiếp tục thực hiện sub-clause *vua(X)* thì chúng ta sẽ dễ dàng thấy rằng sub-clause lần này được thực hiện thành công. Prolog đã tìm ra lời giải, tuy nhiên, ở trường hợp này, ngoài sự hợp nhất, Prolog còn sử dụng thêm một "vũ khí" mới, đó là sự quay lui.

Khống chế số lượng lời giải

Chúng ta xét ví dụ sau

VD10:

predicates

nguoi(symbol)

chet(symbol)

clauses

nguoi("Socrates").

nguoi("Xeda").

chet(X) :- nguoi(X).

Ví dụ không có gì phức tạp, so với VD6, chúng ta chỉ thêm một người mới là Xeda.
Khi sử dụng goal ngoại, với câu hỏi *nguoi(X)*
Chúng ta có hai lời giải:
X = Socrates
X = Xeda.

Chúng ta cảm thấy hai đáp số này là hiển nhiên. Tuy nhiên nếu chúng ta dùng goal nội tương tự VD8, chúng ta chỉ có một đáp số là Socrates.

Đây là một trong những điểm khác biệt căn bản của goal nội và goal ngoại. *Goal nộichỉ tìm câu trả lời đầu tiên còn goal ngoại tìm tất cả các câu trả lời có thể.* Để khống chế số lượng lời giải theo ý mình, chúng ta sử dụng hai vị từ đặc biệt là nhát cắt (cut) và fail, như các ví dụ sau:

VD11: Viết lại VD10, sử dụng vị từ fail để in ra tất cả các lời giải trong trường hợp dùng goal nội.

Chương trình sẽ được viết lại như sau:
predicates
nguoi(symbol)
chet(symbol)
clauses
nguoi("Socrates").
nguoi("Xeda").
chet(X) :- nguoi(X).
goal
nguoi(X),nl,write(X),fail

nl: là vị từ đặc biệt, luôn trả về kết quả là đúng, và chỉ đơn giản là xuống dòng trước khi in thông tin mới ra cửa sổ giao tiếp.
Fail: là một vị từ đặc biệt, luôn luôn trả về kết quả là sai.

Ị hư vậy để in ra tất cả các kết quả, chúng ta dùng một thủ thuật (trick) thường gặp khi lập trình trên Prolog: sau khi đã tìm thấy lời giải cho sub-goal *nguoi(X)* và in giá trị này ra bằng lời gọi vị từ *write(X)*, chúng ta gọi vị từ fail để nhận được kết quả là sai. Do cơ chế back-tracing đã nói ở trên, Prolog lại quay lại thời điểm gọi sub-goal *nguoi(X)* để tìm lời giải khác và in ra. Quá trình này cứ tiếp tục cho đến khi không thể tìm thấy thêm một lời giải nào khác. Bằng cách này, chúng ta đã in ra tất cả các lời giải cho câu hỏi *nguoi(X)*, tuy nhiên lưu ý rằng với goal chính thì không tìm ra lời giải (do chúng ta luôn gọi vị từ fail cho subgoal cuối cùng)
VD12: Viết lại VD10, dùng vị từ nhát cắt để để in ra một lời giải cho câu hỏi *chet(X)* cho trường hợp dùng goal ngoại.

Vị từ nhát cắt được viết là !, là vị từ đặc biệt, sẽ trả lời đúng khi goal chưa tìm thấy lời giải nào, ngược lại sẽ báo là sai.

Chương trình sẽ được viết lại như sau:
predicates
nguoi(symbol)
chet(symbol)
clauses
nguoi("Socrates").
nguoi("Xeda").
chet(X) :- nguoi(X),!.

Khi sử dụng goal ngoại là *chet(X)*, khi trả lời sub-clause *nguoi(X)*, hệ thống tìm ra đáp số là X = Socrates, và vì lúc này mệnh đề được so trùng *chet(X)* chưa có đáp số nào, vị từ ! tiếp theo sẽ trả lời là thành công.

Do chúng ta đang sử dụng goal ngoại, Prolog sẽ tìm tất cả các câu trả lời có thể có, nên hệ thống sẽ tìm một câu trả lời khác. Để làm điều đó, hệ thống sẽ tìm xem subclause *nguoi(X)* có đáp số nào khác không. Chúng ta sẽ dễ dàng nhận thấy rằng hệ thống tìm thấy một đáp số khác là X = Xeda. Tuy nhiên do goal đã có một lời giải, nên sub-clause tiếp theo là ! sẽ báo là thất bại, và lời giải thứ hai không được chấp nhận.

Tóm tắt
- Goal nội sẽ tìm lời giải đầu tiên, và goal ngoại sẽ tìm tất cả các lời giải có thể có.
- Prolog sẽ sử dụng cơ chế quay lui khi một biến khi chuyển từ trạng thái unbound sang bound sẽ dẫn đến sự thất bại trong việc truy tìm lời giải
- Vị từ fail luôn trả lời là sai, sử dụng khi chúng ta muốn in tất cả các lời giải với goal nội.
- Vị từ ! sẽ trả lời đúng khi goal chưa có lời giải và ngược lại.

6. Lập trình đệ quy với Prolog

Chúng ta nhớ lại rằng với VD2, chúng ta đã cố gắng né tránh cách đặt vấn đề để giải bài toán giai thừa theo cách nhân dồn các số từ 1 đến số cần tính giá trị giai thừa. Điều này sẽ dẫn đến một điểm yếu của Prolog: không cung cấp các cấu trúc điều hiển cần thiết, dẫn đến việc khó khăn khi thực hiện phép lặp. uy nhiên ví dụ này cũng cho thấy một kỹ thuật lập trình tạo nên sức mạnh chủ yếu của Prolog: lập trình đệ quy. Kỹ thuật này cũng phù hợp với suy nghĩ của con người khi tiếp cận giải quyết vấn đề và khiến cho việc lập trình trên Prolog có một sự uyển chuyển và nhẹ nhàng trong việc viết mã. Tuy vậy, nó tạo ra một sự khó khăn với những người quen lập trình thủ tục. Chúng ta sẽ xem xét lại từng bước trong việc gọi đệ quy để tìm ra lời giải.

VD13: Xét từng bước quá trình gọi đệ quy và hợp nhất của VD7 với goal là giaithua(2,X)

Ị hắc lại, chúng ta đã có đoạn chương trình như sau:
predicates
giaithua(integer,integer)

clauses
giaithua(0,1):-!.
*giaithua(X,Y):- X1 = X -1, giaithua(X1,Y1), Y = X*Y1*

Ở đây có một sự thay đổi nhỏ: chúng ta đặt nhát cắt để chuyển sự kiện đầu thành luật. Chúng ta muốn khẳng định: nếu số cần tìm giai thừa là 0 thì giai thừa của nó là 1, *và kết quả này là duy nhất*, không cần phải tiếp tục tìm các trường hợp khác. Với goal là *giaithua(2,X)*, hệ thống sẽ so trùng với mệnh đề *giaithua(0,1)* là mệnh đề đầu tiên tìm thấy có liên quan đến khái niệm *giaithua*.
Hệ thống sẽ hợp nhất các thông số theo thứ tự, 2 hợp nhất với 0 và X hợp nhất với 1.
Công việc hợp nhất X với 1 thành công, X có giá trị là 1, nhưng 2 hợp nhất với 0 thất bại.
Hệ thống sẽ tiếp tục tìm kiếm lời giải khác bằng cách so trùng với mệnh đề khác. Lần này hệ thống so trùng goal với mệnh đề *giaithua(X,Y)*. Khi tạo mối liên quan giữa các thông số, hệ thống hợp nhất 2 với X của mệnh đề và Y với X của goal. Chúng ta sẽ ký hiệu XG là X thông số của goal. Do Y và XG đều chưa có giá trị, Prolog sẽ xem hai biến này là một.
Sau đó hệ thống bắt đầu thực hiện từng sub-clause: *X1 = X - 2*
X1 là biến mới, và chưa có giá trị. X đã có giá trị là 2, nên X - 1 có giá trị là 1. Hợp nhất X1 với 1 ta sẽ có giá trị của X1 là 1.
giaithua(X1,Y1)

Ở đây mệnh đề giai thừa được gọi đệ quy. Lưu ý lúc này X1 đã có giá trị là 1, Y1 là biến mới chưa có giá trị, vì vậy nhiệm vụ của hệ thống là tìm giá trị của Y1 sao cho sub-clause *giaithua(X1,Y1)* cho giá trị là đúng. Và cũng như các ví dụ trên, cách thức Prolog trả lời một sub-clause cũng tương tự như khi trả lời câu hỏi từ goal, tức là lại so trùng câu hỏi với các mệnh đề đã biết

- So trùng với *giaithua(0,1)*, Prolog tiến hành hợp nhất X1 với 0, Y1 với 1, do X1 đã có giá trị là 1, việc hợp nhất với 0 thất bại, Prolog phải so trùng với mệnh đề khác.
- So trùng với *giaithua(X,Y)*, Prolog tiến hành hợp nhất X1 với X đồng nhất Y1 với Y. Chúng ta ký hiệu X và Y ở lần gọi đệ quy này là X2 và Y2, và sử dụng cách ký hiệu tương tự như vậy cho các biến còn lại ở lần gọi đệ quy này cũng như các lần gọi đệ quy tiếp theo. Ị hư vậy X2 sẽ có giá trị là 1 và Y1 sẽ có giá trị mà Y2 sẽ có. Tương tự ở lần gọi thứ nhất, các sub-clause của mệnh đề trên ở lần gọi thứ hai này sẽ lần lượt được gọi:
  - *X12 = X2 - 1*, hợp nhất X12 với X2 -1, ta có X12 có giá trị là 0.
  - *giaithua(X12,Y12)*, X12 đã có giá trị là 0, Prolog sẽ tìm giá trị của Y12 bằng việc tiếp tục so trùng *giaithua(X12,Y12)* với các mệnh đề có liên quan:
- So trùng *giaithua(X12,Y12)* với *giaithua(0,1)*. Do X12 đã có giá trị là 0, Prolog tiến hành hợp nhất X12 với 0 và Y12 với 1. Thực hiện tiếp sub-clause !, do câu hỏi *giaithua(X12,Y12)* chưa tìm được câu trả lời nào, nên sub-clause này trả lời là đúng. Việc thực hiện mệnh đề *giaithua(0,1)* thành công, và Y12 đã có giá trị là 1 nên câu hỏi *giaithua(X12,Y12)* đã có đáp số. Vị từ ! sẽ ngăn chặn việc tìm các đáp số khác, *vì vậy trong trường hợp này, Prolog không tiếp tục so trùng tiếp với mệnh đề giaithua(X,Y).*
  - *Y2 = X2 * Y12*, lúc này Y2 chưa có giá trị, X2 và Y12 đã có giá trị là 1 và 1 nên Prolog sẽ hợp nhất Y2 và 1. Kết quả sẽ là Y2 có giá trị là 1. Ị hư vậy đến đây các

sub-clause của mệnh đề *giaithua(X2,Y2)* đã thực thi thành công, và Y2 đã có giá trị là 1, và vì Y1 được đồng nhất với Y2, nên Y1 cũng sẽ có giá trị là 1.

- ❖ *Y = X* Y1*, lúc này Y chưa có giá trị, X và Y1 đã lần lượt có giá trị là 2 và 1, nên Prolog hợp nhất Y và 2*1, kết quả Y sẽ có giá trị là 2.

Ị hư vậy đến đây các sub-clause của mệnh đề *giaithua(X,Y)* đã thực thi thành công, và Y đã có giá trị là 2, và vì XG được đồng nhất với Y, nên XG cũng sẽ có giá trị la ø2, và lời giải của bài toán đã được tìm thấy.

Tóm tắt:

- ❖ Đệ quy là sức mạnh lập trình chủ yếu trên Prolog
- ❖ Mỗi lần gọi đệ quy, các thông số và biến cục bộ trong mỗi mệnh đề sẽ được tạo mới tương ứng với lần gọi đệ quy đó.
- ❖ Có thể dùng nhát cắt để ngăn chặn các lần gọi đệ quy thừa khi đã tìm ra đáp số

7. Danh sách trên Prolog

Danh sách là kiểu dữ liệu cấu trúc đặc biệt trên Prolog. Có thể hiểu danh sách như một kiểu dãy một chiều, và phần tử của danh sách có thể thuộc về kiểu dữ liệu bất kỳ, tuy nhiên các phần tử trong cùng một danh sách phải cùng kiểu.

Định nghĩa kiểu danh sách

Kiểu danh sách là một kiểu dữ liệu (user-defined type) do người dùng định nghĩa trên Prolog. Chúng ta cần phải định nghĩa một kiểu dữ liệu danh sách trước khi sử dụng. Phần định nghĩa kiểu dữ liệu mới sẽ được khai báo sau từ khoá *domains* và đặt ở đầu chương trình.

VD14: Khai báo một kiểu dữ liệu mới là một danh sách số nguyên trên Prolog có tên là list.

domains

*list = integer**

Ký hiệu * biểu hiện cho danh sách. *list* sẽ là kiểu dữ liệu danh sách có phần tử thuộc kiểu *integer*.

Cấu trúc của danh sách

Một danh sách trên Prolog bao gồm hai phần: phần đầu (head) là phần tử đầu tiên của danh sách và phần đuôi (tail) là danh sách các phần tử còn lại của danh sách.

Một danh sách có thể mô tả theo hai cách:

- Liệt kê các phần tử của danh sách, ví dụ: [1,2,3,4,5]
- Mô tả phần đầu và phần đuôi của danh sách, ngăn cách bởi dấu |, ví dụ [1|[2,3,4,5]]

VD15: Mô tả một danh sách bao gồm 5 số nguyên là 1,2,3,4,5

Danh sách trên có thể mô tả theo các cách sau:
[1,2,3,4,5]
[1|[2,3,4,5]]
[1|[2|[3,4,5]]]
[1|[2|[3|[4,5]]]]
[1|[2|[3|[4|[5]]]]]
[1|[2|[3|[4|[5|[]]]]]]

Lưu ý: danh sách rỗng có thể được mô tả như sau: []

VD16: Viết chương trình in ra phần đầu và phần đuôi của một danh sách.
Chương trình này thực chất chỉ giúp chúng ta nhìn rõ hơn khái niệm về danh sách.
Chương trình được viết như sau:
domains
*list = integer**
predicates
indanhsach(list,integer,list)
clauses
indanhsach(L,H,T):- L = [H|T].
Xét khi chúng ta nhập goal vào như sau:
indanhsach([1,2,3,4,5],X,Y)

Prolog sẽ so trùng goal với mệnh đề *indanhsach(L,H,T)*, L được hợp nhất với [1,2,3,4,5], X được đồng nhất với H, Y được đồng nhất với T. Khi thực hiện sub-clause *L = [H|T]*, L được hợp nhất với [H|T], như vậy phần đầu của L sẽ hợp nhất với H, phần đuôi sẽ hợp nhất với T. Do L đã có giá trị là [1,2,3,4,5], phần đầu của L sẽ có giá trị là 1, phần đuôi sẽ có giá trị là [2,3,4,5], vậy sau khi hợp nhất, H sẽ có giá trị là 1 và L sẽ có giá trị là [2,3,4,5]. Cũng tức là X sẽ có giá trị là 1 và Y có giá trị là [2,3,4,5]. Prolog đã tìm thấy lời giải và sẽ in ra lời giải này.

Lưu ý:
- Chương trình trên sẽ chạy sai nếu danh sách nhập vào là rỗng ([]) do lời giải của chúng ta chưa xử lý trường hợp này
- Phần mệnh đề cho vị từ *indanhsach* có thể viết lại là *indanhsach([H|T],H,T).*

Tóm tắt
- Danh sách là kiểu dữ liệu cấu trúc đặc biệt do người dùng định nghĩa trên Prolog
- Một danh sách bao gồm hai phần: phần đầu là phần tử đầu, phần đuôi là danh sách các phần tử còn lại của danh sách.
- Trong trường hợp danh sách rỗng, phần đầu của danh sách sẽ không có.

8. Lập trình đệ quy với danh sách trên Prolog

Khi xử lý danh sách trên Prolog, người lập trình phải từ bỏ phong cách dùng vòng lặp để xử lý dãy mà phải tận dụng kỹ thuật đệ quy để tìm ra lời giải. Chúng ta xét một số ví dụ sau đây:

VD17: Viết vị từ đếm xem một danh sách có bao nhiêu phần tử.
Đầu tiên chúng ta phải định nghĩa được công việc mà chúng ta định làm.
Chúng ta sẽ viết một vị từ như sau:
dem(list,integer)
Chúng ta đếm trong một danh sách có bao nhiêu phần tử. Ví dụ khi gọi goal dem([1,2,3,4],X), đáp số sẽ là X = 4
Tiếp theo chúng ta phải xác định một thuật giải phù hợp với tinh thần của Prolog.
Không thể dùng vòng lặp, chúng ta phải xây dựng một giải thuật đệ quy.
Một giải thuật đệ quy sẽ bao gồm hai phần: điều kiện dừng và lời gọi đệ quy.
Điều kiện dừng cho bài toán này rất dễ dàng: khi danh sách không có phần tử nào, thì hiển nhiên số phần tử của nó là 0. Vậy điều kiện dừng sẽ được viết như sau: *dem([],0):-!.*

Khi trường hợp danh sách không có phần tử nào, đáp số 0 là duy nhất, vậy chúng ta có thể dùng nhát cắt để yêu cầu Prolog không tìm thêm lời giải nào khác.
Phần đệ quy hơi khó đối với những ai chưa quen với danh sách trên Prolog. Prolog chỉ cung cấp cho chúng ta cơ chế chia danh sách thành hai phần: phần tử đầu và phần đuôi danh sách các phần tử còn lại. Ị hư vậy lời giải gần như đã tự nó hiện ra: chúng ta sẽ gọi đệ quy để đếm phần đuôi có bao nhiêu phần tử rồi cộng nó với 1 (phần tử đầu) sẽ ra số phần tử trong một danh sách.

Phần này sẽ được viết như sau:
dem([H|T],X):- dem(T,X1), X = X1+1.
Tư tưởng đệ quy đã được thể hiện rất rõ ràng: đếm phần đuôi T của danh sách để có được tổng X1, hợp nhất X với X1+1 sẽ cho đáp số cần tìm. Tuy nhiên chúng ta thấy ở đây biến H hòan tòan không cần dùng trong vế phải của mệnh đề. Prolog không coi đây là lỗi, nhưng sẽ "phàn nàn" về việc này. Xét về hiệu quả lập trình, hòan tòan không cần khai báo tên biến mới cho thành phần H trong trường hợp này. Có một nguyên tắc để nhận ra những biến "vô dụng" như vậy: đó là những biến chỉ xuất hiện 1 lần trong suốt mệnh đề. Đối với trường hợp này, ta nên dùng ký hiệu '_' thay cho tên biến để biểu diễn biến này không cần dùng trong thuật giải.

Vậy phần đệ quy sẽ được "tinh chế" như sau:
dem([_|T],X):- dem(T,X1), X = X1+1.
Ị hư vậy toàn bộ chương trình hoàn chỉnh sẽ như sau:
domains
*list = integer**
predicates
dem(list,integer)
clauses
dem([],0):-!.
dem([_|T],X):- dem(T,X1), X = X1+1.
VD18: Viết vị từ tính tổng các phần tử trong một danh sách
domains
*list = integer**
predicates

tong(list,integer)
clauses
tong([],0):-!.
tong([H|T],X):- tong(T,X1), X = X1+H.
Tư duy đệ quy ở đây là: chúng ta tính tổng phần đuôi của danh sách, rồi cộng nó với phần tử đầu để tính tổng danh sách.

VD19: Viết vị từ kiểm tra một số nguyên có nằm trong danh sách hay không.

Xác định vấn đề: chúng ta sẽ viết vị từ *ptu(integer,list)*, khi gọi *ptu(2,[1,3,5])* kết quả sẽ là Ị o, ngược lại khi gọi *ptu(3,[1,3,5]),* kết quả sẽ là Yes. Ở đây chúng ta phải xác định được các trường hợp một phần tử nằm trong một danh sách. Và chúng ta phải trình bày được các khái niệm này một cách đệ quy.

Sau đây là ý tưởng của thuật giải: có hai trường hợp để một số nguyên là phần tử của một danh sách: số nguyên này là phần tử đầu của danh sách hoặc nó là phần tử của phần đuôi danh sách.

Sau khi xác định được ý tưởng, chúng ta có bài giải như sau:
domains
*list = integer**
predicates
ptu(integer,list)
clauses
ptu(H,[H|_]):-!.
ptu(H,[_|T]):- ptu(H,T).

Lưu ý:
- Chúng ta đã thay thế các biến chỉ xuất hiện một lần trong một mệnh đề bằng ký hiệu '_' như đã nói.
- Ở mệnh đề đầu đã có ký hiệu nhát cắt, nên nếu mệnh đề này đúng, bài toán đã có lời giải và không so trùng đến mệnh đề thứ hai. *Như vậy mệnh đề thứ hai chỉ được so trùng khi mệnh đề thứ nhất thất bại*, và vì mệnh đề thứ nhất ứng với trường hợp số nguyên cần tìm bằng với phần tử đầu của danh sách, nên khi mệnh đề thứ nhất thất bại, tức là số nguyên cần tìm không bằng với phần tử đầu của danh sách, nên trong mệnh đề thứ hai, *chúng ta không cần quan tâm đến phần tử đầu của danh sách*.

VD20: Xác định phần tử thứ n của danh sách
Khi ta gọi ptn([1,**3**,5,7,9],**2**,X) -_ X = 3 (phần tử thứ 2 của danh sách). Vì Prolog chỉ cho chúng ta truy xuất phần tử đầu tiên của danh sách, nên chúng ta phải xây dựng thuật giải đệ quy dựa trên cơ sở này: nếu n bằng 1 thì phần tử cần tìm là phần tử đầu của danh sách, ngược lại thì đó sẽ là phần tử thứ n -1 của phần đuôi danh sách.

domains
*list = integer**

predicates
ptn(list,integer,integer)
clauses
ptn([H|_],1,H):-!.
ptn([_|T],N,X):- N1 = N - 1, ptn(T,N1,X).

VD 21: Tạo ra một danh sách bao gồm chỉ các phần tử lẻ của danh sách ban đầu.
Goal khi gọi sẽ có dạng ptle([1,2,3,4,5,6],L) _ L = [1,3,5]
Điều kiện dừng của bài toán rất đơn giản: khi danh sách là rỗng thì danh sách được tạo ra cũng là rỗng. Và phần đệ quy của bài toán phải dựa trên việc truy xuất từ phần tử đầu tiên của danh sách và gọi đệ quy cho phần đuôi. Chúng ta phải xét các trường hợp khác nhau của phần tử đầu.
domains
*list = integer**
predicates
ptle(list,list)
clauses
ptle([],[]):-!.
ptle([H|T],[H|T1]):- H mod 2 = 1, ptle(T,T1),!.
ptle([_|T],T1):- ptle(T,T1)
.
Ở đây chúng ta lưu ý đến mệnh đề thứ ba: do hai mệnh đề trên đều có nhát cắt, nên mệnh đề thứ ba chỉ được so trùng khi hai mệnh đề trên đều sai. Mệnh đề thứ hai đã xét trường hợp khi H lẻ, vì vậy ở mệnh đề thứ ba chắc chắn H không phải là số lẻ, và vì vậy không cần phải xét đến.

Tóm tắt:
- ❖ Đệ quy là công cụ chủ yếu để xử lý danh sách trên Prolog
- ❖ Việc gọi đệ quy trên danh sách thường dựa trên cơ sở chia danh sách thành phần đầu và phần đuôi.
- ❖ Sử dụng vị từ nhát cắt hợp lý sẽ khiến các mệnh đề trở nên gọn nhẹ và logic hơn.

9. Danh sách hai chiều

Danh sách hai chiều là một trường hợp đặc biệt của danh sách. Phần tử của danh sách lúc này cũng là một danh sách. Ta xét một ví dụ về khai báo danh sách hai chiều trên Prolog:

VD22: Khai báo danh sách hai chiều trên Prolog
domains
list = integer*
list2d = list*
Ị hư vậy kiểu dữ liệu list2d là kiểu dữ liệu danh sách mà phần tử của nó thuộc kiểu list, tức là một danh sách khác. Việc lập trình với danh sách hai hay nhiều chiều, về cơ bản, *hòan tòan không khác với việc lập trình danh sách một chiều*, tức cũng sử dụng các giải thuật xử lý danh

sách bình thường. Tuy nhiên chúng ta cần chú ý rằng phần tử của danh sách này cũng là một danh sách.

VD23: Viết chương trình đếm trên một danh sách hai chiều của các số nguyên L có bao nhiêu phần tử con có tổng các phần tử là số chẵn.

Với bài toán này, chúng ta phải viết thêm một vị từ phụ để có được lời giải:

```
domains
list = integer*
list2d = list*
predicates
tong(list,integer)
dem(list2d,integer)
clauses
tong([],0):-!.
tong([H|T],X):-tong(T,X1), X = H+X1.
dem([],0):-!.
dem([H|T],X):-tong(H,N), N mod 2 = 0, dem(T,X1), X = X1 + 1,!.
dem([_|T],X):- dem(T1,X).
```

Tóm tắt

- Danh sách nhiều chiều là một danh sách trong đó các phần tử cũng là danh sách
- Về mặt tư duy, lập trình để xử lý trên danh sách nhiều chiều không khác nhiều với xử lý danh sách một chiều.
- Các phần tử của danh sách nhiều chiều cũng là danh sách, nên có thể áp dụng các giải thuật xử lý danh sách để xử lý các phần tử này.

Bài tập Thực hành Prolog

Viết các chương trình giải quyết các công việc sau:

1. Viết chương trình để đếm số phần tử là số chẵn trên một danh sách.
2. Viết chương trình để đếm số phần tử là số chẵn trên một danh sách nhưng phần tử trước nó phải là một số lẻ.
3. Viết chương trình lấy phần tử chính giữa một danh sách (trường hợp số phần tử của danh sách là số chẵn thì trả về số 0)
4. Viết chương trình xử lý giữa hai danh sách: lấy các phần giao, hội, hiệu của hai danh sách.
5. Viết chương trình để lấy một danh sách chỉ bao gồm các phần tử có chỉ số chẵn trên một danh sách khác.
6. Viết chương trình nhằm chia danh sách thành hai phần. Phần đầu gồm n.a phần đầu của danh sách được sắp xếp theo thứ tự tăng dần, phần sau gồm n.a sau danh sách được sắp xếp theo thứ tự giảm dần.
7. Ví dụ: chia([3,52,7,312,25,66,34,45],X,Y) → X = [3,7,52,312], Y = [25,34,45,66]
8. Sinh viên nghĩ ra cách giải quyết cho trường hợp số phần tử của danh sách là số lẻ.

9. Viết chương trình để chia một danh sách thành n phần (n là thông số nhập), trong đó từng phần lần lượt được sắp theo thứ tự tăng dần rồi đến giảm dần.
10. Viết chương trình để sắp xếp các phần tử của một danh sách theo thứ tự phần dư của các phần tử này cho một số nguyên Ị .
11. Viết chương trình xóa đi sự xuất hiện đầu tiên của một phần tử trong một danh sách.
12. Viết chương trình xóa đi tất cả sự xuất hiện của một phần tử trong một danh sách.
13. Viết chương trình tìm kiếm một danh sách con trong một danh sách.
14. Viết chương trình xóa đi sự xuất hiện đầu tiên của một danh sách con trong một danh sách
15. Viết chương trình xóa đi toàn bộ sự xuất hiện của một danh sách con trong một danh sách.
16. Viết chương trình để tạo ra một tập hợp từ một danh sách các số nguyên (trong một tập hợp thì một phần tử xuất hiện nhiều nhất là một lần.)
17. Viết một chương trình để tạo ra một danh sách bao gồm Ị phần tử nhỏ nhất từ một danh sách.
18. Viết một chương trình tạo ra một danh sách các ước số của một số nguyên.
19. Viết chương trình tính toán tần suất xuất hiện của một danh sách trong một danh sách khác.
20. Ví dụ: tansuat([1,2,3],[1,2,3,4,1,2,3,5],X) → X = 2.
    tansuat([2,2],[1,2,2,2,2,2],X) → X = 4.
21. Viết chương trình kiểm tra xem một danh sách có đối xứng hay không.
22. Viết chương trình đảo ngược một danh sách.
23. Viết chương trình so sánh hai danh sách xem chúng có là hai tập hợp bằng nhau hay không.
24. Viết chương trình loại bỏ những số nguyên nào xuất hiện đúng Ị lần trong một danh sách.

BÀI 7: LOGIC MỜ

1. Một số khái niệm

1.1/ Tập mờ (Fuzzy Sets)

a. Định nghĩa tập mờ

- Tập rõ (crisp sets)

Cho tập hợp A là con của tập vũ trụ U và x là một phần tử. Khi đó hàm thành viên $\mu(x)$ có dạng như sau:

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \qquad (1.1)$$

Rõ ràng hàm thành viên μ_A chỉ nhận một trong hai giá trị là 0 hay 1. Ị hận giá trị 0 khi $x \notin A$ và nhận giá trị 1 khi $x \in A$. Vì thế $\mu_A(x) \in \{0,1\}$.

Ví dụ: Cho tập hợp $U = \{x_1, x_2, x_3, x_4, x_5\}$ và $A = \{x_2, x_3, x_5\}$ thì chỉ có 3 trong 5 phần tử thuộc về A. Ta có:

$$\mu_A(x_1) = 0, \quad \mu_A(x_2) = 1, \quad \mu_A(x_3) = 1$$

$$\mu_A(x4) = 0, \quad \mu_A(x_5) = 1$$

Vì thế hàm đặc trưng của A như sau:

$$\mu_A = \begin{cases} 1 & , x = x_2, x_3, x_5 \\ 0 & , x = x_1, x_4 \end{cases} \qquad (1.2)$$

- *Tập mờ*

Xét tập hợp A là con của tập vũ trụ U. Một tập mờ A được định nghĩa bởi một tập hợp hoặc các cặp xó thứ tự (một quan hệ nhị phân) như sau:

$$A = \{(x, \mu_A(x)) \mid x \in A, \ \mu_A(x) \in [0,1]$$

trong đó $\mu_A(x)$ là một hàm gọi là hàm thành viên chỉ định mức độ mà phần tử x thuộc về tập mờ A

Định nghĩa trên gắn kết mỗi phần tử x trong tập A với một số thực $\mu_A(x)$ trong khoảng [0, 1]. Giá trị $\mu_A(x)$ càng lớn thì mức độ x thuộc về A càng cao.

Xem xét công thức (1.2) ta thấy các phần tử x của cặp $(x, \mu_A(x))$ cho biết số lượng các phần tử trong tập cổ điển A, chúng thoả một số điều kiện (P) nào đó và hàm thành viên $\mu_A(x)$

có miền giá trị trong khoảng [0, 1] và đó là việc mở rộng mức độ mà các phần tử x thoả mã điều kiện *P*. Hàm thành viên được xem xét có thể là một hàm liên tục hoặc một hàm rời rạc.

Tập mờ A theo định nghĩa (1.2) thường tương đương với hàm thành viên $\mu_A(x)$. Do đó chúng ta có thể nhận biết được tập mờ thông qua hàm thành viên và có thể dùng hai khái niệm này thay thế cho nhau. Vì thế chúng ta có thể xem một tập mờ trên miền *A* là một ánh xạ từ tập *A* vào đoạn [0,1].

Các tập mờ thường được ký hiệu bằng các chữ cái in hoa *A, B, C*… và các hàm thành viên tương ứng được ký hiệu bằng $\mu_A(x)$, $\mu_B(x)$, $\mu_C(x)$,…Các phần tử với hàm thành viên chỉ mức độ bằng không thường không được liệt kê vào tập mờ.

Các tập rõ có thể được xem như là một trường hợp đặt biệt của tập mờ với các hàm thành viên chỉ mức độ bằng 1.

Một tập mờ *A* được gọi là được chuẩn hoá (*normalized*) khi có ít nhất một phần tử $x \in A$ mà tại đó hàm thành viên tại $\mu_A(x) = 1$. Ị ược lại tập mờ *A* được gọi là tập không được chuẩn hoá.

Giả sử tập hợp *A* là tập chưa được chuẩn hoá, khi đó max $(\mu_A(x)) < 1$. Để chuẩn hoá tập *A* có nghĩa là chuẩn hoá hàm thành viên ta chỉ việc lấy hàm thành viên đó chia cho max $(\mu_A(x))$, ta được các giá trị $\dfrac{\mu_A(x)}{\max(\mu_A(x))}$

Tập *A* được gọi là tập rỗng ký hiệu là ∅ nếu $\mu_A(x) = 0$ với mọi $x \in A$.

Tập mờ $A = \{(x_1, \mu_A(x_1))\}$ với x_1 là giá trị duy nhất trong tập $A \subset U$ và $\mu_A(x_1) \in [0,1]$ thì *A* được gọi là tập mờ đơn.

Ví dụ 1:

Xét tập mờ

$A = \{(x_1, 0.1), (x_2, 0.5), (x_3, 0.3), (x_4, 0.8), (x_5, 1), (x_6, 0.2)\}$

mà có thể được biểu diễn như sau:

$A = 0.1/x_1 + 0.5/x_2 + 0.3/x_3 + 0.8/x_4 + 1/x_5 + 0.2/x_6$

Đây là tập mờ rời rạc bao gồm 6 cặp. Các phần tử x_i, i = 1,…,6 không cần thiết phải đánh số, chúng thuộc về tập $A = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ và là tập con của U. Hàm thành viên $\mu_A(x)$ của A nhận các giá trị sau trong khoảng [0, 1]:

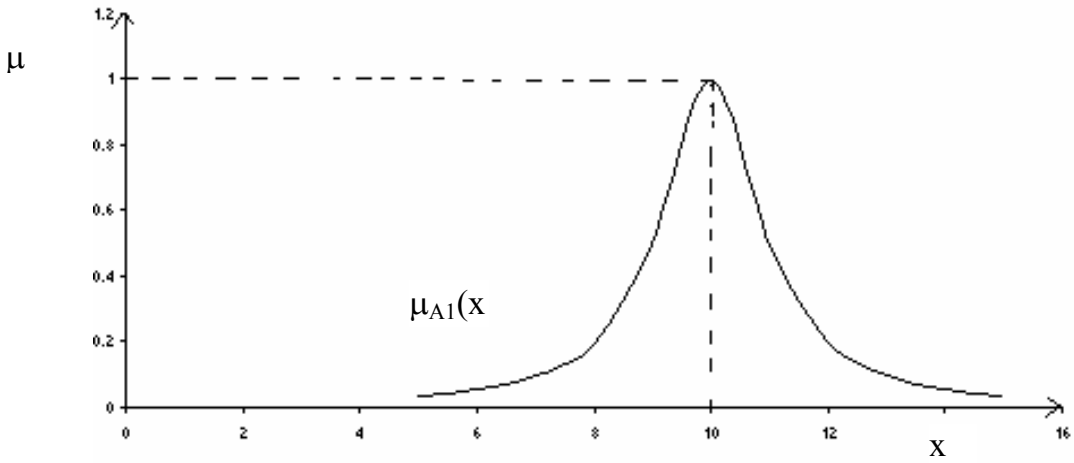
Ví dụ 2:

Chúng ta hãy mô tả các số lân cận 10

Trước tiên xem xét tập mờ

$$A_1 = \left\{(x, \mu_{A_1}(x)) \mid x \in [5,15], \mu_{A_1}(x) = \frac{1}{1+(x-10)^2}\right\}$$

trong đó $\mu_{A1}(x)$ là một hàm liên tục và được thể hiện trong hình sau:

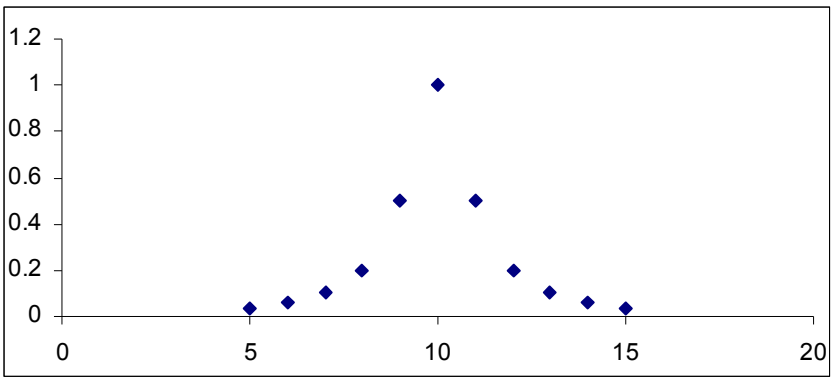


Hình 1: Các số thực lân cận 10

Ị hững số nguyên lân cận 10 có thể được mô tả bằng một tập mờ hữu hạn có 7 cặp có thứ tự như sau:

A_2 = { (7,0.1), (8,0.2), (9,0.5), (10,1), (11, 0.5),(12,0.2), (13, 0.1)}

Hàm thành viên của A_2 được thể hiện trong hình 2 bằng những dấu chấm. Đó là một hàm rời rạc.



Hình 2: Những số nguyên lân cận 10

- Cho tập mờ A, ta định nghĩa lát cắt α (α-cut) ký hiệu A_α như là một tập rõ các phần tử x mà nó thuộc A với mức độ ít nhất là α

$$A = \{ x \mid x \in R,\ \mu_A(x) \geq \alpha \}, \quad \alpha \in [0,1]$$

Lát cắt trên là một ngưỡng mà nó cho ta mức độ tin cậy α trong một quyết định hoặc khái niệm trong một tập mờ. Chúng ta có thể dùng ngưỡng này để loại bỏ ra khỏi tập những phần tử x thuộc về A với mức độ $\mu_A(x_1) < \alpha$

Ví dụ 3:

Cho tập T miêu tả độ cao của những người đàn ông. Tập này có vô số những khoảng cắt bên trong T_α với α thay đổi từ 0.22 đến 1. Cho một vài lát cắt sau đây:

$T_{0.22} = \{ x \mid x \in R, 160 \le x \le 200\}, \mu_T(x) \ge 0.22$

$T_{0.5} = \{ x \mid x \in R, 170 \le x \le 200\}, \mu_T(x) \ge 0.5$

$T_{0.78} = \{ x \mid x \in R, 180 \le x \le 200\}, \mu_T(x) \ge 0.78$

Chẳng hạn ta chọn một ngưỡng với lát cắt $T_{0.5}$ do đó ta sẽ loại những người đàn ông cao dưới 170 cm ra khỏi tập cần xem xét.

- Một tập mờ A với $U = R$ là một tập tập lồi nếu và chỉ nếu những khoảng cắt A_α là bao lồi đối với những khoảng cắt trong khoảng (0, 1]. Trong trường hợp này thì các khoảng cắt A_α chỉ gồm có một đoạn. Ị gược lại thì tập A không phải là tập lồi.

b. Các phép toán trên tập mờ

Xét hai tập mờ A và B trong tập vũ trụ U

$$A = \{(x, \mu_A(x))\}, \quad \mu_A(x) \in [0,1]$$

$$B = \{(x, \mu_B(x))\}, \quad \mu_B(x) \in [0,1]$$

Các phép toán giữa A và B được thực hiện thông qua hàm thành viên của nó là $\mu_A(x)$ và $\mu_B(x)$

• Tương đương

Tập mờ A và B được gọi là tương đương, ký hiệu là $A = B$ nếu và chỉ nếu

$$\mu_A(x) = \mu_B(x), \quad \forall x \in U$$

• Bao gồm (bao hàm)

Tập mờ A được bao hàm trong tập mờ B, ký hiệu là $A \subseteq B$ nếu

$$\mu_A(x) \le \mu_B(x), \quad \forall x \in U$$

Khi đó tập A được gọi là tập con của tập B

• Tập con thực sự

Tập mờ A được gọi là tập con thực sự của tập mờ B, ký hiệu là $A \subset B$ khi A là tập con của B và $A \ne B$.

$$\left.\begin{array}{ll} \mu_A(x) \le \mu_B(x), & \forall x \in U \\ \mu_A(x) < \mu_B(x), & \exists x \in U \end{array}\right\}$$

• Bù

Tập mờ A và $\overline{A}$ được gọi là bù nhau nếu

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x), \quad \forall x \in U \text{ hoặc } \mu_A(x) + \mu_{\overline{A}}(x) = 1, \quad \forall x \in U$$

Hai hàm thành viên $\mu_{\bar{A}}(x)$ và $\mu_A(x)$ đối xứng với nhau qua trục đối xứng là đường thẳng $\mu = 0.5$

• Giao

Phép giao của tập A và tập B ký hiệu là $A \cap B$ được định nghĩa bởi

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)), \quad x \in U$$

Ị ếu $a_1 < a_2$ thì **min**(a_1, a_2) = a_1. Ví dụ: **min**(0.5, 0.7) = 0.5.

• Hợp

Phép hợp của tập mờ A và tập mờ B ký hiệu là $A \cup B$ được định nghĩa

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)), \quad x \in U$$

Ị ếu $a_1 < a_2$ thì **max**(a_1, a_2) = a_2. Ví dụ: **max**(0.5, 0.7) = 0.7.

Ví dụ:

Xét tập vũ trụ U = {x_1, x_2, x_3, x_4,} và hai tập mờ A, B được định nghĩa bởi bảng sau:

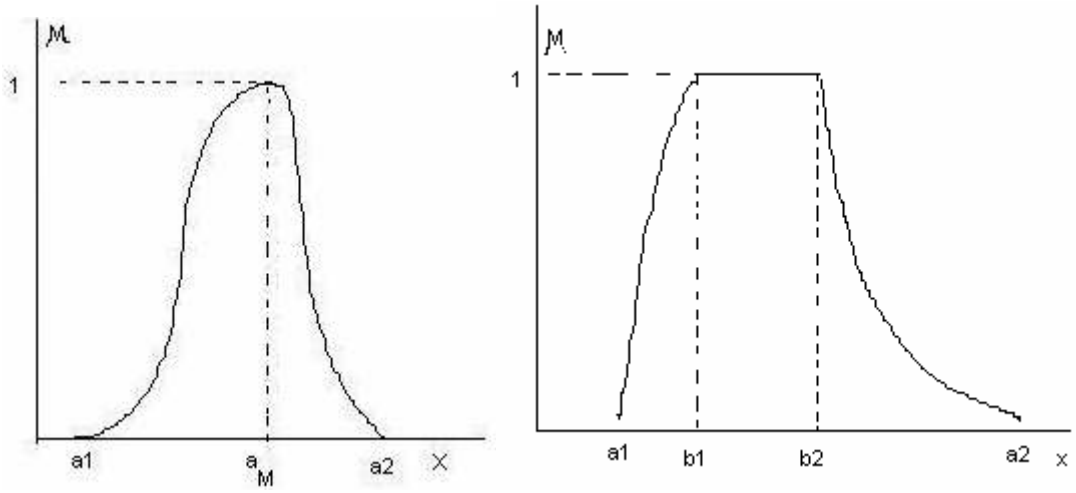
X	x1	x2	x3	x4
$\mu_A(x)$	0.2	0.7	1	0
$\mu_B(x)$	0.5	0.3	1	0.1

Khi đó ta có hợp và giao của $\mu_A(x)$ và $\mu_B(x)$ như sau:

X	x1	x2	x3	x4
$\mu_A(x) \cap \mu_B(x)$	0.2	0.3	1	0
$\mu_B(x) \cup \mu_B(x)$	0.5	0.7	1	0.1

1.2/ Số mờ (Fuzzy Numbers)

Một số mờ được định nghĩa dựa trên tập vũ trụ R là một tập lồi và đã được chuẩn hoá. Hình (2.a) và (2.b) thể hiện hai số logic:



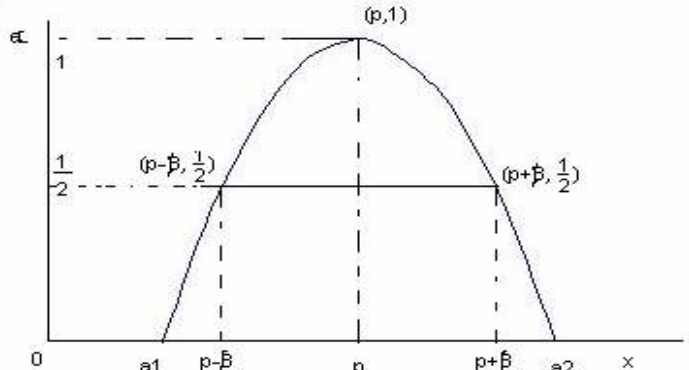
Hình (2a): số logic với một giá trị lớn nhất Hình (2b): số logic với nhiều giá trị lớn nhất

Khoảng $[a_1, a_2]$ được gọi là một khoảng xác định của số logic. Chẳng hạn với số logic $x = a_M$ trong hình (2a) là một giá trị lớn nhất trong khi một đoạn trong hình (2b) có độ cao lớn nhất là 1, thực chất đây là một lát cắt với độ cao tin cậy là 1.

Các số logic sẽ được ký hiệu bằng các chữ cái hoa được tin đậm như: **A, B, C**,… và các hàm thành viên được ký hiệu bằng $\mu_A(x)$, $\mu_B(x)$, $\mu_A(x)$, $\mu_C(x)$,…

1.3/ Số logic dạng hình khối

Hàm thành viên μ_A của một số logic dạng hình khối thể hiện trong hình (3) có dạng hình chuông, đối xứng qua đường thẳng $x = p$, xác định trong khoảng $[a_1, a_2]$ và được đặc trưng bởi hai tham số $p = \frac{1}{2}(a_1 + a_2)$ và $\beta \in (0,\ a_2 - p)$. Đỉnh chóp của hình chuông là điểm (p, 1); 2p được gọi là biên độ và được định nghĩa như là một đoạn cắt tại mức α giữa hai điểm $(p - \beta, \frac{1}{2})$ và $(p + \beta, \frac{1}{2})$ được gọi là những điểm giao nhau.



Hình (3): Số logic dạng hình khối

Đường cong trong hình (3) được mô tả bởi các phương trình sau:

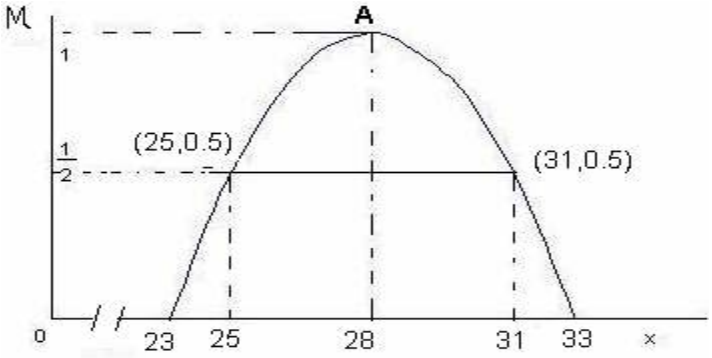
$$\mu_A = \begin{cases} \dfrac{1}{2(p-\beta-a_1)^2}(x-a_1)^2, & a_1 \le x \le p-\beta, \\ -\dfrac{1}{2\beta^2}(x-p)^2+1, & p-\beta \le x \le p+\beta, \\ \dfrac{1}{2(p+\beta-a_2)^2}(x-a_2)^2, & p+\beta \le x \le a_2 \\ 0 \end{cases} \qquad (2.1)$$

Số logic của (2.1) là một số thực lân cận *p*. Bởi vì từ "lân cận" là mơ hồ và trong ngữ cảnh logic nên không được định nghĩa một cách đơn điệu. Điều đó phụ thuộc vào khoảng xác định được chọn và biên độ được dùng để phản ánh một trường hợp cụ thể. Chẳng hạn tập mờ về chiều cao của những người đàn ông là một trường hợp cụ thể của (2.1) (nhánh trái) trong khoảng [160, 200] với $a_1 = 140$, p = 200 và β = 30.

Ví dụ:

Giá sản xuất của một sản phẩm lân cận 28 và có thể được mô tả bởi số logic A trong hình 2.2 như sau trong đó $a_1 = 23$, $a_2 = 33$, p = 28 và β = 3

Hàm thành viên $\mu_A(x)$ của (2.1) có thể nhận được từ (2.1) bằng việc thay thế a_1, a_2, p và β bằng những giá trị đặc biệt như sau:



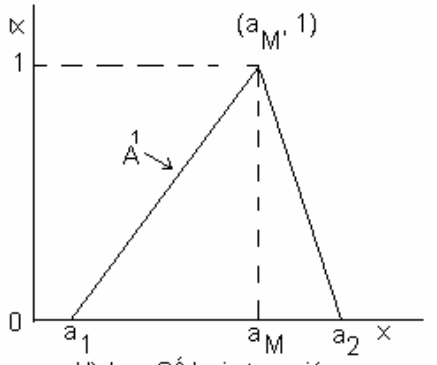
Hình 2.2 Giá sản phẩm lân cận 28

1.4/ Số logic dạng tam giác

Một số logic dạng tam giác hoặc đơn giản hơn là số logic tam giác với hàm thành viên $\mu_A(x)$ được định nghĩa trên *R* bởi:

$$A \overset{\Delta}{=} \mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_M-a_1}, & a_1 \le x \le a_M, \\ \dfrac{x-a_2}{a_M-a_1}, & a_M \le x \le a_2, \\ 0 \end{cases} \qquad (3.1)$$

trong đó $[a_1, a_2]$ là khoảng xác định và điểm $(a_M, 1)$ là đỉnh (xem hình)



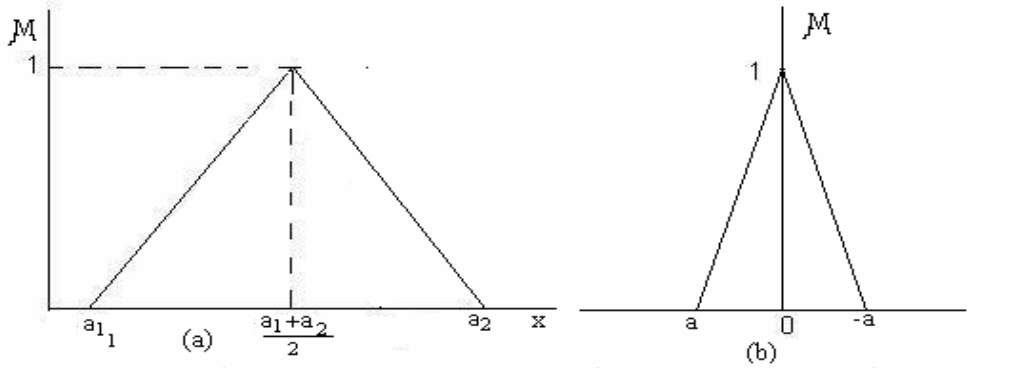
Hình Số logic tam giác

Thông thường trong các ứng dụng điểm $a_M \in (a_1, a_2)$ được đặt tại giữa của khoảng xác định, ví dụ như $a_M = \frac{a_1 + a_2}{2}$, sau đó thay thế giá trị này vào công thức (3.1) thu được

$$A \overset{\Delta}{=} \mu_A(x) = \begin{cases} 2\dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le \dfrac{a_1 + a_2}{2}, \\ 2\dfrac{x - a_2}{a_2 - a_1}, & \dfrac{a_1 + a_2}{2} \le x \le a_2, \\ 0 \end{cases} \qquad (3.2)$$

Ta nói rằng (3.2) biểu diễn một số logic tam giác ở giữa, tương tự như số logic hình khối và rất dễ để diễn tả từ lân cân (lân cận tới a_M)

Số logic tam giác rất thường được dùng trong các ứng dụng (điều khiển mờ, quản lý việc ra quyết định, thương nghiệp và tài chính, xã hội,…). Chúng có một hàm thành viên bao gồm hai đoạn thẳng tuyến tính A^l (trái) và A^r (phải) giao nhau tại điểm $(a_M, 1)$ tạo nên dạng hình học và các phép toán trên số logic tam giác rất đơn giản.



Hình . (a) Số logic tam giác ở giữa (b) Số logic tam giác ở giữa đối xứng qua M

Giả sử chúng ta đang giải quyết với một giá trị không chắc chắn, chúng ta có thể chỉ định giá trị lớn nhất và giá trị nhỏ nhất có thể được trong khoảng $[a_1, a_2]$. Ị ếu chúng ta có thể chỉ định một giá trị $a_M \in [a_1, a_2]$ đáng tin cậy để thể hiện một giá trị chắc chắn thì đỉnh của tam giác sẽ là $(a_M , 1)$. Vì thế với ba giá trị a_1, a_2 và a_M có thể tạo lập một số tam giác và mô tả hàm thành viên của nó. Đó là tại sao số tam giác còn được viết dưới dạng:

$A = (a_1, a_M, a_2)$ (3.3)

Số logic tam giác ở giữa là điểm đối xứng với trục μ. Ị ếu $a_1 = -a$, $a_2 = a$ thì $a_M = 0$. Do đó A = (-a, 0, a)

Ị hánh phải của A = (-a, 0, a) khi $0 \le x \le a$ được dùng để mô tả một số dương bé và được ký hiệu là $A^r = (0, 0, a)$. Tổng quát hơn, nhánh trái và nhánh phải của số tam giác có thể được ký hiệu tương ứng là $A^l = (a_1, a_M, a_M)$ và $A^r = (a_M, a_M, a_2)$ và được xem là số tam giác trái và số tam giác phải tương ứng.

1.5/ Số logic dạng hình thang

Một số logic có dạng hình thang hay gọi tắt là số logic hình thang được định nghĩa trên R như sau:

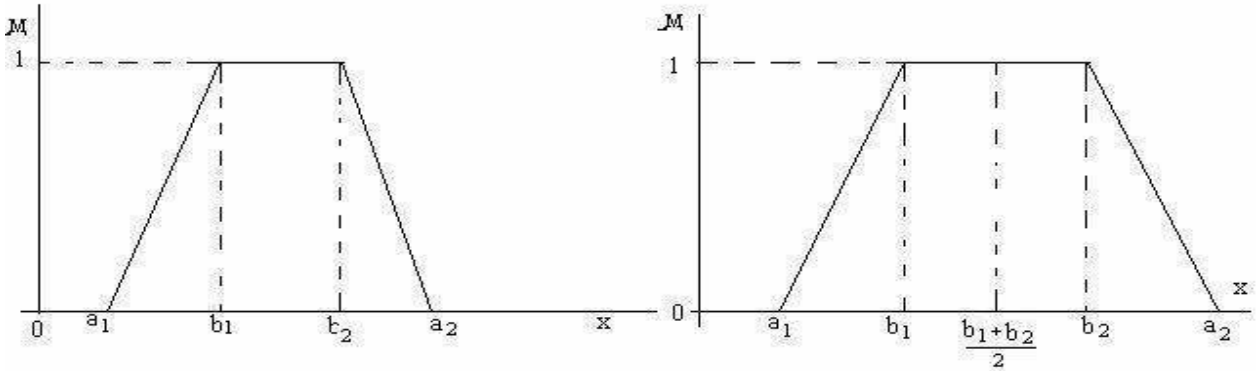
$$\overset{\Delta}{A} = \mu_A(x) = \begin{cases} \dfrac{x - a_1}{b_1 - a_1}, & a_1 \le x \le b_1 \\ 1, & b_1 \le x \le b_2 \\ \dfrac{x - a_2}{b_2 - a_2}, & b_2 \le x \le a_2 \\ 0 \end{cases}$$

Số logic hình thang là một trường hợp đặc biệt của số logic với đỉnh phẳng. Khoảng xác định của A là $[a_1, a_2]$ và đoạn đỉnh phẳng với mặt cắt $\alpha = 1$ có giá trị chiếu lên trục x tương ứng là b_1 và b_2. Với 4 giá trị a_1, a_2, b_1 và b_2 chúng ta có thể xây dựng một số logic hình thang và được ký hiệu là

$A = (a_1, b_1, b_2, a_2)$

Ị ếu $b_1 = b_2$, số logic hình thang biến thành số logic tam giác và được ghi là (a_1, a_M, a_M, a_2). Vì thế một số logic tam giác $A = (a_1, a_M, a_2)$ có thể được viết dưới dạng số logic hình thang là $A = (a_1, a_M, a_M, a_2)$. Khi đó $(a_1, a_M, a_2) = (a_1, a_M, a_M, a_2)$

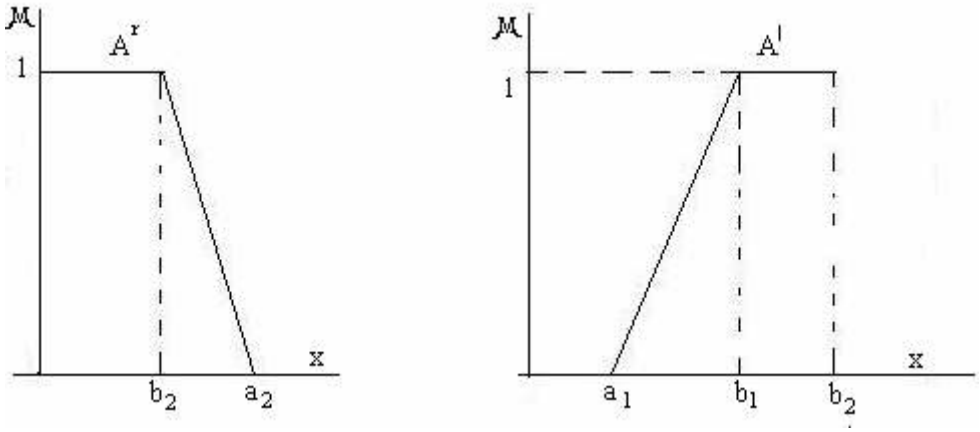
Ị ếu $[a_1, b_1] = [a_2, b_2]$ thì số logic hình thang là đối xứng qua đường $x = \frac{1}{2}(b_1 + b_2)$. Đó là một hình thang cân tương ứng trong khoảng $[b_1, b_2]$ và những số thực lân cận trong khoảng này.



Tương tự như số logic trái và số logic phải của tam giác, chúng ta có thể đưa ra số logic hình thang trái và số logic hình thang phải là các phần của số logic hình thang.

$A^l = (a_1, b_1, b_2, b_2)$

$A^r = (b_1, b_1, b_2, a_2)$



2. Áp dụng của logic mờ trong dự đoán

Việc dự đoán cung cấp những khái niệm cơ sở cho những việc sắp xảy ra. Khả năng dự đoán và ước lượng những sự kiện trong tương lai đòi hỏi phải nghiên cứu trên những nguồn dữ liệu mơ hồ và xuất phát từ những thay đổi nhanh chóng của môi trường, một công việc mà dùng logic mờ để giải quyết phù hợp hơn dùng phương pháp phân loại. Việc phân tích những tình huống phức tạp cần nhiều công sức và nhiều ý kiến của các chuyên gia. Ị hững ý kiến của các chuyên gia thường không trùng hợp nhau, ít nhiều cũng có những chỗ giống nhau và những chỗ đối lập nhau. Chúng sẽ được kết hợp lại hoặc tập hợp lại để tạo ra một quyết định. Trong phần này phương pháp học của logic mờ trung bình sẽ được đề cập và được dùng như một công cụ chính để tổp hợp nhiều mô hình dự đoán khác nhau (fuzzy Delphi, quản lý dự án, dự đoán theo yêu cầu).

2.1/ Giá trị trung bình trong thống kê

Một trong những khái niệm quan trọng nhất trong thống kê đó là giá trị trung bình (*average* hay *mean*) của n phép đo, n ý kiến hoặc n đại lượng được biểu diễn bởi các số thực $r_1, r_2, \ldots, r_n$ được định nghĩa bởi

$$r_{ave} = \frac{r_1 + r_2 + \ldots + r_n}{n} = \frac{\sum_{i=1}^{n} r_i}{n}; \qquad (3.1)$$

những đại lượng này được xem có cùng mức độ quan trọng. Giá trị trung bình tiêu biểu hoặc đại diện cho n đại lượng

Ị ếu các đại lượng này có mức độ quan trọng khác nhau được biểu diễn bởi các số thực $\lambda_1, \lambda_2, , , \lambda_n$ tương ứng thì khái niệm trọng số trung bình được tính theo công thức.

$$r_{ave}^{w} = \frac{\lambda_1 r_1 + ... + \lambda_n r_n}{\lambda_1 + ... \lambda_n} = w_1 r_1 + ... + w_n r_n = \sum_{i=1}^{n} w_i r_i$$

Ở đây W_i được gọi là trọng số và được cho bởi công thức sau:

$$W_i = \frac{\lambda_i}{\lambda_1 + ... + \lambda_n} . i = 1,2,..., n, \qquad W_1 + ... + W_n = \sum_{i=1}^{n} W_i$$

Trọng số phản ánh tầm quan trọng hoặc cường độ của r_i

Khái niệm trung bình có thể được tổng quát hoá bằng việc thay thế những số mờ bằng những con số thực r_i trong công thức (1) và (2). Ở đây chúng tôi giới hạn lại thủ tục tổng quát hoá đến số tam giác và số hình thang.

2.2/ Các phép toán với số tam giác vá số hình thang

a) *Phép cộng trên số tam giác*

Ị gười ta đã chứng minh được rằng tổng của hai số tam giác $A_1 = (a_1^{(1)}, a_M^{(1)}, a_2^{(1)})$ và $A_2 = (a_1^{(2)}, a_M^{(2)}, a_2^{(2)})$ cũng là một số tam giác

$$A_1 + A_2 = (a_1^{(1)}, a_M^{(1)}, a_2^{(1)}) + (a_1^{(2)}, a_M^{(2)}, a_2^{(2)})$$
$$= (a_1^{(1)} + a_1^{(2)}, a_M^{(1)} + a_M^{(2)}, a_2^{(1)} + a_2^{(2)}) \qquad (4)$$

Công thức tổng này có thể được mở rộng cho n số tam giác và có thể áp dụng cho số tam giác trái hoặc số tam giác phải.

Chẳng hạn như:

$$A_1^{r} + A_2 = (a_M^{(1)}, a_M^{(1)}, a_2^{(1)}) + (a_1^{(2)}, a_M^{(2)}, a_2^{(2)})$$
$$= (a_M^{(1)} + a_1^{(2)}, a_M^{(1)} + a_M^{(2)}, a_2^{(1)} + a_2^{(2)})$$

$$A_1^{l} + A_2 = (a_1^{(1)}, a_M^{(1)}, a_M^{(1)}) + (a_1^{(2)}, a_M^{(2)}, a_M^{(2)})$$
$$= (a_1^{(1)} + a_1^{(2)}, a_M^{(1)} + a_M^{(2)}, a_M^{(1)} + a_M^{(2)})$$

Ví dụ 1: Tính tổng của hai số tam giác sau:

$A_1 = (-5, -2, 1), \qquad A_2 = (-3, 4, 12)$

Dựa theo công thức (4) ta có:

$A_1 + A_2 = (-5 + (-3), -2 + 4, 1 + 12) = (-8, 2, 13)$

Dựa vào hình này ta có thể hiểu như sau: Ị ếu A_1 mô tả các số thực lân cận -2 và A_2 mô tả các số thực lân cận 4 thì $A_1 + A_2$ sẽ thể hiện các số thực lân cận -2+4 = 2.

b) *Nhân số tam giác với 1 số thực*

Phép nhân một số tam giác A với một số thực r cũng là một số tam giác:

$$A\ r = r\ A = r\ (a_1, a_M, a_2) = (r\ a_1, r\ a_M, r\ a_2)$$

c) *Chia số tam giác cho một số thực*

Phép chia số tam giác A cho số thực r được xem như là phép nhân số tam giác A với một số thực $\frac{1}{r}$ với $r \neq 0$.

$$\frac{A}{r} = \frac{1}{r}(a_1, a_M, a_2) = (\frac{a_1}{r}, \frac{a_M}{r}, \frac{a_{21}}{r})$$

Ví dụ:

- Ị hân số tam giác A = (2,4,5) với số thực 2 ta được:

$$2A = 2\ (2, 4, 5) = (4, 8, 10)$$

- Chia số tam giác A = (2,4,5) cho 2 ta được:

$$\frac{A}{2} = \frac{1}{2}(2,4,5) = (1,2,2.5)$$

- Vì thế

$$\frac{2A}{2} = \frac{(4,8,10)}{2} = A\ ; \qquad 2(\frac{A}{2}) = 2(1,2,2.5) = A$$

Các phép toán trên số hình thang được thực hiện tương tự như các phép toán trên số tam giác

d) *Cộng hai số hình thang*

Phép cộng hai số hình thang $A_1 = (a_1^{(1)}, b_1^{(1)}, b_2^{(1)}, a_2^{(1)})$ và số $A2 = (a_1^{(2)}, b_1^{(2)}, b_2^{(2)}, a_2^{(2)})$ cũng là một số hình thang.

$$A_1 + A_2 = (a_1^{(1)}, b_1^{(1)}, b_2^{(1)}, a_2^{(1)}) + (a_1^{(2)}, b_1^{(2)}, b_2^{(2)}, a_2^{(2)})$$
$$= (a_1^{(1)} + a_1^{(2)}, b_1^{(1)} + b_1^{(2)}, b_2^{(1)} + b_2^{(2)}, a_2^{(1)} + a_2^{(2)}) \qquad (7)$$

Công thức trên (3.7) có thể được tổng quát hoá cho n số hình thang và các số hình thang trái và các số hình thang phải.

e) *Nhân số hình thang với một số thực*

$$A\ r = r\ A = (r\ a_1,\ r\ b_1, r\ b_2, r\ a_2) \qquad (8)$$

f) *Chia số hình thang với một số thực*

$$\frac{A}{r} = \frac{1}{r}A = (\frac{a_1}{r}, \frac{b_2}{r}, \frac{b_2}{r}, \frac{a_2}{r}), \qquad r \neq 0 \qquad (9)$$

g) *Tổng số tam giác với số hình thang*

Xét số tam giác $A_1 = (a_1^{(1)}, a_M^{(1)}, a_2^{(1)})$ mà có thể hiện diện như một số hình thang $(a_1^{(1)}, a_M^{(1)}, a_M^{(1)}, a_2^{(1)})$ và số hình thang A2 = $(a_1^{(2)}, b_1^{(2)}, b_2^{(2)}, a_2^{(2)})$. Dùng công thức (7) ta có

$$A_1 + A_2 = (a_1^{(1)}, a_M^{(1)}, a_M^{(1)}, a_2^{(1)}) + (a_1^{(2)}, b_1^{(2)}, b_2^{(2)}, a_2^{(2)})$$
$$= (a_1^{(1)} + a_1^{(2)}, a_M^{(1)} + b_1^{(2)}, a_M^{(1)} + b_2^{(2)}, a_2^{(1)} + a_2^{(2)}) \quad (10)$$

2.3/ Trung bình trong logic mờ

a) *Công thức trung bình trong tam giác*

Xét n số tam giác $A_i = (a_1^{(i)}, a_M^{(i)}, a_2^{(i)})$, I = 1,…,n.

Sử dụng phép cộng các số tam giác và phép chia số tam giác với một số thực sẽ cho ta số trung bình trong tam giác A_{ave}

$$A_{ave} = \frac{A_1 + \ldots + A_n}{n}$$
$$= \frac{(a_1^{(1)}, a_M^{(1)}, a_2^{(1)}) + \ldots + (a_1^{(n)}, a_M^{(n)}, a_2^{(n)})}{n}$$
$$= \frac{(\sum_{i=1}^{n} a_1^{(i)}, \sum_{i=1}^{n} a_M^{(i)}, \sum_{i=1}^{n} a_2^{(i)})}{n}$$

Và cũng là một số tam giác

$$A_{ave} = (m_1, m_M, m_{2)} = \left(\frac{1}{n}\sum_{i=1}^{n} a_1^{(i)}, \frac{1}{n}\sum_{i=1}^{n} a_M^{(i)}, \frac{1}{n}\sum_{i=1}^{n} a_2^{(i)} \right) \quad (11)$$

Ví dụ:

(i) Cho số tam giác A_1 và A_2 như trong ví dụ 1. Khi đó số trung bình của hai số đó là:

$$A_{ave} = \frac{A_1 + A_2}{2} = \frac{(-8,2,13)}{2} = (-4,1,6.5)$$

(ii) Số tam giác A_1^r , A_2 và A_3^l trong ví dụ 2 có số trung bình là:

$$A_{ave} = \frac{A_1^r + A_2 + A_3^l}{3} = \frac{(4,9,12)}{3} = (1.33,3,4)$$

b) *Công thức trung bình trong tam giác có trọng số*

Ị ếu những số thực λ_i biểu diễn mức độ quan trọng của các số tam giác $A_i = (a_1^{(i)}, a_M^{(i)}, a_2^{(i)})$, i = 1,…n. Áp dụng công thức (2), (3) và (11) ta thu được trung bình trong tam giác có trọng số.

$$A_{ave}^w = \frac{\lambda_1 A_1 + .. + \lambda_n A_n}{\lambda_1 + .. + \lambda_n}$$
$$= w_1(a_1^{(1)}, a_M^{(1)}, a_2^{(1)} + .. + w_n(a_1^{(n)}, a_M^{(n)}, a_2^{(n)})$$
$$= (w_1 a_1^{(1)}, w_1 a_M^{(1)}, w_1 a_2^{(1)} + .. + (w_n a_1^{(n)}, w_n a_M^{(n)}, w_n a_2^{(n)})$$
$$= (w_1 a_1^{(1)} + .. + w_n a_1^{(n)}, w_1 a_M^{(1)} + .. + w_n a_M^{(n)}, w_1 a_2^{(1)} + .. + w_n a_2^{(n)})$$

và có thể được viết ngắn gọn lại như sau:

$$A_{ave}^w = (m_1^w, m_M^w, m_2^w) = \left(\sum_{i=1}^n w_i a_1^{(i)}, \sum_{i=1}^n w_i a_M^{(i)}, \sum_{i=1}^n w_i a_2^{(i)} \right)$$

Công thức trung bình đối với số hình thang có thể được chuyển hoá tương tự như (11) và (12) như sau:

c) *Công thức trung bình trong hình thang*

Giả sử $A_i = (a_1^{(i)}, b_1^{(i)}, b_2^{(i)}, a_2^{(i)}), i = 1,...,n$ là các số hình thang. Khi đó số trung bình là:

$$A_{ave} = (m_1, m_{M1}, m_{M2}, m_2)$$
$$= \frac{(a_1^{(1)}, b_1^{(1)}, b_2^{(1)}, a_2^{(1)}) + .. + (a_1^{(n)}, b_1^{(n)}, b_2^{(n)}, a_2^{(n)})}{n}$$
$$= \frac{(\sum_{i=1}^n a_1^{(i)}, \sum_{i=1}^n b_1^{(i)}, \sum_{i=1}^n b_2^{(i)}, \sum_{i=1}^n a_2^{(i)})}{n}$$

d) *Công thức trung bình trong hình thang có trọng số*

$$A_{ave}^w = (m_1^w, m_{M1}^w, m_{M2}^w, m_2^w)$$
$$= \frac{w_1(a_1^{(1)}, b_1^{(1)}, b_2^{(1)}, a_2^{(1)}) + .. + w_n(a_1^{(n)}, b_1^{(n)}, b_2^{(n)}, a_2^{(n)})}{n}$$
$$= \frac{(\sum_{i=1}^n w_i a_1^{(i)}, \sum_{i=1}^n w_i b_1^{(i)}, \sum_{i=1}^n w_i b_2^{(i)}, \sum_{i=1}^n w_i a_2^{(i)})}{n}$$

2.4/ Dự đoán bằng phương pháp Delphi kết hợp logic mờ

Fuzzy Delphi là một cải tiến của phương pháp Delphi cổ điển dùng để dự đoán các sự kiện trong khoa học. Phương pháp này được đề xuất vào thập niên 60 bởi Rand Corporation, Caliornia.

Các bước chính để dự đoán bằng phương pháp này như sau:

(i) Các chuyên gia có uy tín trong lĩnh vực cần dự báo sẽ đưa ra dự đoán về thời điểm diễn ra các sự kiện (khoa học, kỹ thuật, kinh tế…) một cách độc lập.

(ii) Các dữ liệu thu thập được phân tích và tính giá trị trung bình. Sau đó, các kết quả phân tích được gửi ngược lại cho các chuyên gia.

(iii) Các chuyên gia dựa trên kết quả nhận được để tiến hành dự đoán lại rồi gửi kết quả mới cho người tổ chức và kết quả phân tích mới sẽ được gởi lại cho các chuyên gia.

(iv) Quá trình này có thể được lặp lại nhiều lần cho đến khi kết quả dự đoán của các chuyên gia hội tụ về một kết quả mà người tổ chức vừa ý nhất. Thông thường thì chỉ cần lặp lại 2 hoặc 3 lần.

Tuy nhiên, việc dự đoán có một khó khăn là các thông tin có được thường không đầy đủ và thiếu tính chính xác. Hơn nữa mức độ chính xác của các dự đoán phụ thuộc rất nhiều vào quan điểm chủ quan cũng như khả năng của các chuyên gia. Do đó, thay vì sử dụng các con số chính xác trong quá trình dự đoán chúng ta sẻ sử dụng "số mờ". Số tam phần rất phù hợp cho việc đưa ra các dự đoán bằng cách cho 3 giá trị: giá trị nhỏ nhất, giá trị lớn nhất và giá trị tin tưởng nhất. Khi đó giá trị trung bình sẽ được tính dựa trên phương pháp tính trị trung bình của các "số mờ" đã trình bày bên trên.

Ý tưởng kết hợp phương pháp Delphi với logic mờ được Kaufman và Gupta đề xuất vào năm 1988 gồm các bước sau:

- Bước 1: Chuyên gia E_i được yêu cầu dự đoán thời điểm xảy ra một sự kiện bằng cách cho biết ngày sớm nhất, ngày trể nhất và ngày tin tưởng nhất. Thông tin các chuyên gia cung cấp sẽ có dạng:

$$A_i = \left(a_1^{(i)}, a_M^{(i)}, a_2^{(i)}\right), i = 1,\ldots,n$$

- Bước 2: Trước tiên chúng ta sẽ tính giá trị trung bình $A_{avg}\left(m_1, m_M, m_2\right)$ của các kết quả thu được. Sau đó, chúng ta tiến hành tính độ lệch tâm của từng kết quả dự đoán tương ứng của các chuyên gia theo công thức sau:

$$A_{avg} - A_i = \left(m_1 - a_1^{(i)}, m_M - a_M^{(i)}, m_2 - a_2^{(i)}\right)$$

$$= \left(\frac{1}{n}\sum_{i=1}^{n} a_1^{(i)} - a_1^{(i)}, \frac{1}{n}\sum_{i=1}^{n} a_M^{(i)} - a_M^{(i)}, \frac{1}{n}\sum_{i=1}^{n} a_2^{(i)} - a_2^{(i)}\right)$$

Giá trị lệch tâm có được sẽ được gửi trở lại cho các chuyên gia tương ứng để tiến hành dự đoán lại.

- Bước 3: Mỗi chuyên gia sau khi nhận được kết quả phản hồi sẽ tiến hành dự đoán lại và cho một kết quả mới dạng:

$$B_i = \left(b_1^{(i)}, b_M^{(i)}, b_2^{(i)}\right), i = 1,...,n$$

Quá trình này được lặp lại nhiều lần cho đến khi 2 kết quả trung bình liên tiếp là tương đối giống nhau.

- Bước 4: Công việc dự đoán có thể được tiến hành lại nếu sau khi có kết quả lại có thể những thông tin mới làm ảnh hưởng đến kết quả có được.

Phương pháp fuzzy delphi là phương pháp tiêu biểu cho cách dự đoán bằng cách kết hợp quan điểm của nhiều chuyên gia đầu ngành.

Bài tập minh họa 1: Ước lượng thời điểm ra đời sản phẩm mới

Kết quả dự đoán thời điểm xuất hiện máy tính có khả năng suy nghĩ của 15 chuyên gia máy tính được cho trong bảng sau (lần thứ nhất).

E_i	A_i	Thời điểm sớm nhất	Thời điểm có khả năng nhất	Thời điểm muộn nhất
E_1	A_1	$a_1^{(1)} = 1995$	$a_M^{(1)} = 2003$	$a_2^{(1)} = 2020$
E_2	A_2	$a_1^{(2)} = 1997$	$a_M^{(2)} = 2004$	$a_2^{(2)} = 2010$
E_3	A_3	$a_1^{(3)} = 2000$	$a_M^{(3)} = 2005$	$a_2^{(3)} = 2010$
E_4	A_4	$a_1^{(4)} = 1998$	$a_M^{(4)} = 2003$	$a_2^{(4)} = 2008$
E_5	A_5	$a_1^{(5)} = 2000$	$a_M^{(5)} = 2005$	$a_2^{(5)} = 2015$
E_6	A_6	$a_1^{(6)} = 1995$	$a_M^{(6)} = 2010$	$a_2^{(6)} = 2015$
E_7	A_7	$a_1^{(7)} = 2010$	$a_M^{(7)} = 2018$	$a_2^{(7)} = 2020$
E_8	A_8	$a_1^{(8)} = 1995$	$a_M^{(8)} = 2007$	$a_2^{(8)} = 2013$
E_9	A_9	$a_1^{(9)} = 1995$	$a_M^{(9)} = 2002$	$a_2^{(9)} = 2007$
E_{10}	A_{10}	$a_1^{(10)} = 2008$	$a_M^{(10)} = 2009$	$a_2^{(10)} = 2020$
E_{11}	A_{11}	$a_1^{(11)} = 2010$	$a_M^{(11)} = 2020$	$a_2^{(11)} = 2024$
E_{12}	A_{12}	$a_1^{(12)} = 1996$	$a_M^{(12)} = 2002$	$a_2^{(12)} = 2006$
E_{13}	A_{13}	$a_1^{(13)} = 1998$	$a_M^{(13)} = 2006$	$a_2^{(13)} = 2010$
E_{14}	A_{14}	$a_1^{(14)} = 1997$	$a_M^{(14)} = 2005$	$a_2^{(14)} = 2012$
E_{15}	A_{15}	$a_1^{(15)} = 2002$	$a_M^{(15)} = 2010$	$a_2^{(15)} = 2020$
Tổng cộng		29996	30109	30210

Khi đó ta có: $A_{avg} = \left(\frac{1}{15}\sum_{i=1}^{15} a_1^{(i)}, \frac{1}{15}\sum_{i=1}^{15} a_M^{(i)}, \frac{1}{15}\sum_{i=1}^{15} a_2^{(i)}\right)$

$$= \left(\frac{29996}{15}, \frac{30109}{15}, \frac{30206}{15} \right) = (1999.7, 2007.3, 2014)$$

hay lấy gần đúng ta có: $A^a_{avg} = (2000, 2007, 2014)$

Độ lệch tâm của các dự đoán của các chuyên gia tính được như trong bảng sau.

E_i	$m_1 - a_1^{(i)}$	$m_M - a_M^{(i)}$	$m_2 - a_2^{(i)}$
E_1	5	4	-6
E_2	3	3	4
E_3	0	2	4
E_4	2	4	6
E_5	0	2	-1
E_6	5	-3	-1
E_7	-10	-11	-6
E_8	5	0	1
E_9	5	5	7
E_{10}	-8	-2	-6
E_{11}	-10	-13	-10
E_{12}	4	5	8
E_{13}	2	1	4
E_{14}	3	2	2
E_{15}	-2	-3	-6

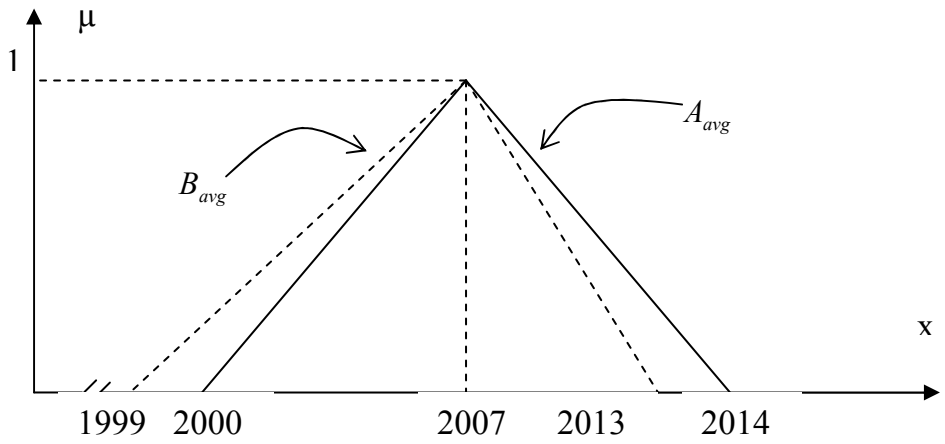
Giả sử rằng người tiến hành cuộc phỏng vấn chưa hài lòng với kết quả thu được. Khi đó, độ lệch tâm tính được trong bảng trên được gửi lại cho các chuyên gia tương ứng xem xét và cho ra một dự đoán mới với kết quả như trong bảng sau.

E_i	B_i	Thời điểm sớm nhất	Thời điểm có khả năng nhất	Thời điểm muộn nhất
E_1	B_1	$b_1^{(1)} = 1996$	$b_M^{(1)} = 2004$	$b_2^{(1)} = 2018$
E_2	B_2	$b_1^{(2)} = 1997$	$b_M^{(2)} = 2004$	$b_2^{(2)} = 2011$
E_3	B_3	$b_1^{(3)} = 2000$	$b_M^{(3)} = 2005$	$b_1^{(3)} = 2011$
E_4	B_4	$b_1^{(4)} = 1998$	$b_M^{(4)} = 2003$	$b_2^{(4)} = 2010$

E_5	B_5	$b_1^{(5)} = 2000$	$b_M^{(5)} = 2005$	$b_2^{(5)} = 2015$
E_6	B_6	$b_1^{(6)} = 1997$	$b_M^{(6)} = 2009$	$a_2^{(6)} = 2015$
E_7	B_7	$b_1^{(7)} = 2005$	$b_M^{(7)} = 2015$	$b_2^{(7)} = 2016$
E_8	B_8	$b_1^{(8)} = 1996$	$b_M^{(8)} = 2007$	$b_2^{(8)} = 2013$
E_9	B_9	$b_1^{(9)} = 1997$	$b_M^{(9)} = 2004$	$b_2^{(9)} = 2010$
E_{10}	B_{10}	$b_1^{(10)} = 2004$	$b_M^{(10)} = 2009$	$b_2^{(10)} = 2017$
E_{11}	B_{11}	$b_1^{(11)} = 2004$	$b_M^{(11)} = 2015$	$b_2^{(11)} = 2016$
E_{12}	B_{12}	$b_1^{(12)} = 1996$	$b_M^{(12)} = 2004$	$b_2^{(12)} = 2006$
E_{13}	B_{13}	$b_1^{(13)} = 1998$	$b_M^{(13)} = 2006$	$b_2^{(13)} = 2010$
E_{14}	B_{14}	$b_1^{(14)} = 1997$	$b_M^{(14)} = 2004$	$b_2^{(14)} = 2012$
E_{15}	B_{15}	$b_1^{(15)} = 2001$	$b_M^{(15)} = 2009$	$b_2^{(15)} = 2015$
Tổng cộng		29986	30103	30198

Khi đó ta có: $B_{avg} = (1999.07, 2006.9, 2013.2)$

Hay gần đúng ta có: $B_{avg}^{a} = (1999, 2007, 2013)$



Ị hư vậy ta có kết quả trung bình của 2 lần liên tiếp là tương đối giống nhau. Do đó, chúng ta có thể ngưng việc khảo sát và chuyển sang giai đoạn diễn giải kết quả thu được. Từ kết quả thu được chúng ta có thể dự đoán là sản phỊm mới có thể xuất hiện trong khoảng thời gian từ 1999 đến 2013. Hơn nữa việc giải mờ kết quả thu được chúng ta có thời điểm có khả năng nhất là vào khoảng năm 2007.

2.5/ Phương pháp Fuzzy Delphi có trọng số:

Sự thật là trong các lĩnh vực dự đoán khác nhau, khả năng am hiểu của từng chuyên gia cũng khác nhau. Do đó, các dự đoán của các chuyên gia khác nhau không thể có cùng một độ tin cậy như nhau. Ị hư vậy mức độ ảnh hưởng của các dự đoán của các chuyên gia tương ừng đối với kết quả cuối cùng là hoàn toàn khác nhau. Điều này dẫn đến một cải tiến của phương pháp Fuzzy Delphi bằng cách thêm vào một trọng số w_i cho từng chuyên gia với điều kiện $\sum_{i=1}^{n} w_i = 1$. Khi đó các bước của phương pháp Fuzzy Delphi vẫn không đổi, chỉ có khác là trước khi tính giá trị trung bình chúng ta phải nhân kết quả dự đoán của các chuyên gia với trọng số tương ứng của chuyên gia đó.

Bài tập minh họa 2: Với kết quả dự đoán trong bài tập trước nhưng giờ đây chúng ta có thêm bảng trọng số của từng chuyên gia cụ thể. Khi đó các kết quả tính toán có được như trong bảng sau.

E_i	w_i	$w_i \times a_1^{(i)}$	$w_i \times a_M^{(i)}$	$w_i \times a_2^{(i)}$
E1	0.1	199.5	200.3	202
E2	0.05	99.85	100.2	100.5
E3	0.5	200	200.5	201
E4	0.05	99.9	100.15	100.4
E5	0.1	200	200.5	201.5
E6	0.05	99.75	100.5	100.75
E7	0.05	100.5	100.9	101
E8	0.1	199.5	200.7	201.3
E9	0.05	99.75	100.1	100.35
E10	0.05	100.4	100.45	101
E11	0.05	100.5	101	101.2
E12	0.05	99.8	100.1	100.3
E13	0.1	199.8	200.6	201
E14	0.05	99.85	100.25	100.6
E15	0.05	100.1	100.5	101
Tổng	1	1999.2	2006.75	2013.9

Ị hư vậy ta có: $A_{avg}^{w} = (1999.2, 2006.75, 2013.9) = (1999, 2007, 2014)$

2.6/ Ứng dụng Fuzzy Pert trong việc quản lý các đề án

Quản lý đề án là một công việc phức tạp, đặc biệt là việc xác định tiến trình thực hiện các công việc. Mỗi đề án có một thời điểm bắt đầu và một thời điểm kết thúc xác định. Tương tự mỗi công việc bên trong đề án cũng có một thời điểm bắt đầu và kết thúc xác định. Các công việc trong đề án phải được thực hiện theo đúng một trình tự nhất định. Do đó, thời điểm hoàn tất của các công việc cũng phải được dự đoán trước.

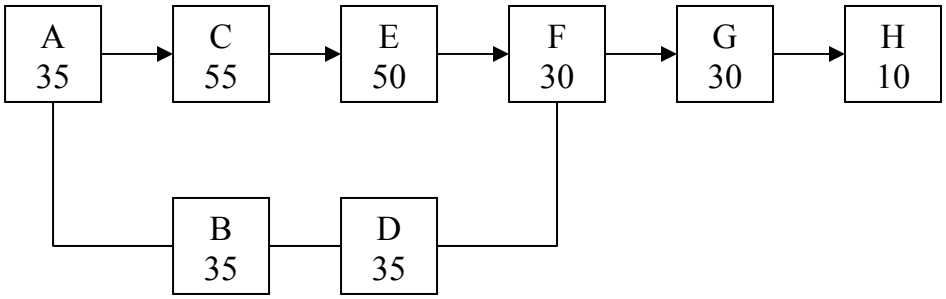
PERT và CPM cổ điển

Hai kỹ thuật cổ điển dùng để hoạch định và kiểm soát đồ án cổ điển là: PERT (Project Evaluation and Review Technique) và CPM (Crictical Path Method). PERT được đề xuất bởi hải quân Mỹ trong quá trình lên kế hoạch sản xuất tài ngầm nguyên tử Polaris. CPM cũng được đề xuất trong cùng thời gian bởi các nhà nghiên cứu ở Remington Rand và DuPont để bảo quản nhà máy hóa chất. Có vài sự tương đồng giữa 2 kỹ thuật này và chúng thường được dùng chung với nhau như là một phương pháp thống nhất.

Để minh họa cách sử dụng PERT và CPM, chúng ta sẽ khảo sát qua một đề án được thực hiện bởi Fogatty và Hoffmann vào năm 1983. Lịch trình thực hiện các công việc được cho trong bảng sau.

Hành động	Hành động phía trước	Hành động đồng thời	Hành động phía sau	Thời gian dự kiến hoàn tất (ngày)
A			B,C	35
B	A	C	D	35
C	A	B	SE	55
D	B	C,E	F	35
E	C	D	F	50
F	D,E		G	30
G	F		G	30
H	F			10

Từ những thông tin trong bảng trên, chúng ta có kế hoạch thực hiện các công việc như trong sơ đồ sau.



Sơ đồ kế hoạch thực hiện các công việc cho chúng ta thấy một cách rõ ràng mối lệ thuộc về thời gian thực hiện các công việc trong toàn bộ đề án.

Đường găng của sơ đồ chính là chuỗi các các công việc theo trình tự thời gian từ đầu cho đến kến thúc đề án mà có thời gian cần thiết để hoàn tất là dài nhất. Ị hư vậy thời gian cần thiết để hoàn tất đề án cũng chính là tổng thời gian của đường găng.

Dựa vào sơ đồ kế hoạch thực hiện các công việc, chúng ta có thể xác định đường găng của đề án. Trong sơ đồ trên, đường găng chính là đường nối bởi các dấu mũi tên: A, C, E, F, G và H. Khi đó ta có tổng thời gian để hoàn tất đề án là 35 + 55 + 50 + 30 + 30 +10 = 210 ngày. Chúng ta có thể thấy rằng 2 công việc B và D không nằm trên đường găng. Hai công việc này có thể không hoàn thành đúng tiến độ nhưng thời gian trể không quá 35 ngày. Ị ếu không thì công việc F trên đường găng sẽ bị ảnh hưởng.

PERT dựa trên niềm tin

Chúng ta không thể dự đoán chính xác thời điểm hoàn tất các công việc. Để giải quyết vấn đề này các nhà nghiên cứu mở rộng đã tăng cường độ chính xác của kỹ thuật PERT bằng cách sử dụng xác suất và thống kê.Để tiến hành dự đoán bằng kỷ thuật PERT, các chuyên gia phải đưa ra 3 dự đoán về thời điểm hoàn tất các công việc: thời điểm tốt nhất t1 nếu như mọi chuyện diễn ra tốt đẹp, thời điểm khả thi nhất nếu mọi chuyện diễn ra theo đúng kế hoạch và thời điểm trong trường hợp xấu nhất nếu như có sự cố xảy ra. Khi đó thời điểm hoàn tất từng công việc được tính theo công thức:

$$t_e = \frac{t_1 + 4t_M + t_2}{6} \qquad (3.22)$$

Thời gian tổng cộng đề hoàn tất cả đề án T_e chính là tổng các t_e của các công việc trên đường găng. Kết quả ước lượng tính theo công thức 3.22 sẽ gần giống với các giá trị trong các ô vuông ghi trên sơ đồ hoạch địnnh các công việc và thời gian ước lượng hoàn tất các công việc cũng sẽ mang nhiều tính thực tế hơn. Tiếp theo chúng ta sẽ phải tính độ lệch tâm cho các t_e và tiến hành những hoạt động phân tích thống kê khác. Để đơn giản, chúng ta đề xuất một giải pháp khác thay thế cho việc sử dụng khái niệm xác suất trong PERT.

Các giá trị ước lượng t_1, t_M, và t_2 do các chuyên gia đưa ra dựa trên kinh nghiệm và kiến thức tuy mang nhiều tính chủ quan nhưng không hề áp đặt. Chính vì vậy bản chất của việc không chắc chắn trong các dự đoán là do tính không rõ ràng của sự việc hơn xác suất xảy ra sự việc. PERT không đề xuất tính các giá trị t_1,t_M và t_2 mà chỉ phát biểu rằng các giá trị đó cần được ước lượng và kết hợp lại để cho ra kết quả cuối cùng dựa vào công thức 3.22.

Sử dụng PERT kết hợp với logic mờ để dự đoán

Chúng ta sẽ cải tiến PERT sử dụng Fuzzy Delphi để ước lượng t_1,t_M và t_2 cho từng công việc. Các chuyên gia sẽ đưa ra dự đoán về thời điểm hoàn thành các công việc dưới dạng số tam phần. Khi đó ứng với từng công việc chúng ta sẽ có một con số trung bình cũng có dạng số tam phần. Tiếp theo chúng ta sẽ giải mờ con số này để được một giá trị thực dự đoán thời gian cần thiết đề hoàn tất công việc.

Bài tập áp dụng:

Chúng ta xem xét lại ví dụ của bài tập trước và bỏ đi các giá trị dự đoán theo PERT cổ điển. Giả sử thời gian cần thiết để thực hiện mỗi công việc được dự đoán bởi 3 chuyên gia khác nhau. Mỗi công việc trong đề án sẽ có một con số ước lượng dạng triangular.Trong bảng sau là kết quả ước lượng cho công việc A.

Chuyên gia	T_i^A	Thời gian nhanh nhất	Thời gian có thể nhất	Thời gian lâu nhất
E_1	T_1^A	33	35	38
E_2	T_2^A	33	34	37
E_3	T_3^A	32	36	39
Tổng số	$\sum_{i=1}^{3} T_i^A$	98	105	114

Tổng kết các kết quả dự đoán chúng ta được một giá trị trung bình của thời gian cần thiết để hoàn thành công việc A

$$T_{ave}^A = \left(\frac{98}{3}, \frac{105}{3}, \frac{114}{3}\right) = (32.67, 35, 38) \approx (33, 35, 38)$$

Để giải mờ T_{ave}^A, chúng ta sử dụng một trong các công thức của công thức 3.15

$$t_{\max}^{(1)} = \frac{32.67 + 35 + 38}{3} = 35.22$$

$$t_{\max}^{(2)} = \frac{32.67 + 2*35 + 38}{4} = 35.17$$

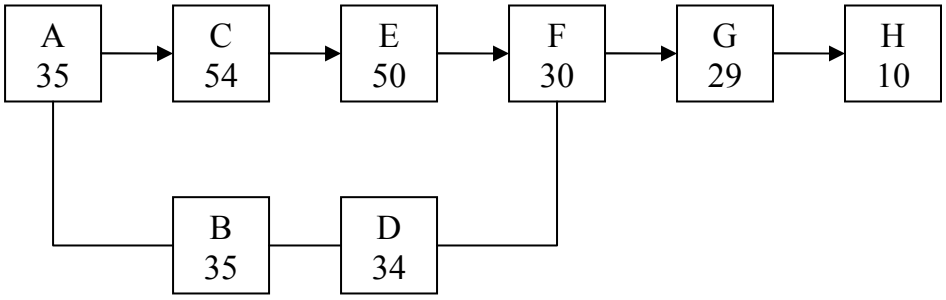
$$t_{\max}^{(3)} = \frac{32.67 + 4*35 + 38}{6} = 35.11$$

Chúng ta có thể thấy rằng dù tính theo công thức nào chúng ta cũng có giá trị gần với con số 35 ngày.

Tương tự các nhóm chuyên gia khác sẽ có kết quả ước lượng cho các công việc còn lại như trong bảng sau.

Công việc	Thời gian trung bình	Thời gian tốt nhất t_1	Thời gian dự kiến t_M	Thời gian lâu nhất t_2
A	T_{ave}^{A}	33	35	38
B	T_{ave}^{B}	32	35	38
C	T_{ave}^{C}	51	54	58
D	T_{ave}^{D}	32	34	36
E	T_{ave}^{E}	46	50	53
F	T_{ave}^{F}	27	30	33
G	T_{ave}^{G}	27	29	32
H	T_{ave}^{H}	7	10	12

Khi giải mờ các giá trị trung bình trong bảng trên chúng ta có sơ đồ thực hoạch định công việc như sau.



Số tam phần T cần thiết để hoàn tất dự án chính là tổng T_{ave}^{i} ,i = A, C, E, F, G, H.

$$T = T_{ave}^{A} + T_{ave}^{C} + T_{ave}^{E} + T_{ave}^{F} + T_{ave}^{G} + T_{ave}^{H} = (192,208,226)$$

Ị hư vậy thời gian cần thiết để hoàn thành dự án khoảng từ 192 đến 226 ngày, mà nhiều khả năng là 208 ngày có được bằng cách giải mờ số T.

Lên lịch cấp phát tài nguyên

Khoảng thời gian cần thiết để hoàn tất công việc có ảnh hưởng mật thiết đến công việc phân công nhân vật lực. Tất cả mọ người đều có cảm nhận rằng cần phải xác định đường găng của sơ đồ hoạch định công việc trước khi tiến hành phân công tài nguyên thực hiện các công việc. Việc dự đoán thời gian cần thiết để hoàn thành dự án được thực hiện với giả thuyết là

các tài nguyên cần thiết luôn luôn sẵn sàn cấp phát khi công việc cần đến. Tuy nhiên trong thực tế có rất nhiều vấn đề phức tạp nảy sinh. Thông thường người quản lý dự án có thể trang bị thêm các tài nguyên cần thiết để rút ngắn thời gian hoàn tất dự án. Song điều đó đồng nghĩa với việc tăng chi phí thực hiện dự án. Ị hư vậy phương cách tốt nhất là chúng ta tìm cách để rút ngắn thời gian tiến hành dự án.

Bài tập áp dụng (Phần 2)

Trước khi tiến hành thực hiện việc rút ngắn thời gian, chúng ta có các quy ước như sau.

t_n: thời gian dự định để hoàn tất công việc

t_c: thời gian ngắn nhất có thể hoàn tất công việc

C_n: chi phí dự kiến để hoàn tất công việc

C_c: chi phí thấp nhất có thể

Để rút ngắn thời gian hoàn tất dự án thực chất là đi rút ngắn đường găng của sơ đồ hoạch định các công việc. Việc rút ngắn thời gian thực hiện các công việc B và D hoàn toàn không làm giảm thời gian hoàn tất dự án. Song, chúng ta có thể sử dụng một phần tài nguyên để thực hiện B và D dùng cho việc thực hiện công việc C và E. Tuy nhiên ở đây chúng ta giả sử không thực hiện giải pháp này.

Giả sử rằng kết quả dự đoán thòi gian hoàn tất từng công việc đã được thực hiện và có kết quả như trong bài tập trên.

Ị hư vậy chúng ta còn phải tiến hành dự đoán thời gian ngắn nhất có thể t_c, chi phí dự kiến C_n và chi phí thấp nhất C_c của từng công việc tương bằng cách sử dụng Fuzzy Delphi. Giá trị sau khi đã giải mờ cho các đại lượng trên lần lượt là $t_{n_{\max}}$, $t_{c_{\max}}$, $C_{n_{\max}}$ và $C_{c_{\max}}$

Tiếp theo chúng ta xét qua ví dụ về ước lượng giá trị C_n cho hành động A. Các gía trị còn lại được ước lượng bằng cách tương tự.

Việc ước lượng giá trị C_n được tiến hành bởi 3 chuyên gia. Kết quả ước lượng của từng chuyên gia là một con số dạng triangular $C_n = (C_{n1}, C_{nM}, C_{n2})$. Trong đó C_{n1} là chi phí cho trường hợp tốt nhất, C_{nM} là chi phí trong trường hợp bình thường và C_{n2} là chi phí cho trường hợp xấu nhất.

Chuyên gia	C_{n1}	C_{nM}	C_{n2}
E_1	18000	20000	22000
E_2	19500	21000	22000
E_3	17000	19500	21000
Tổng	54500	60500	65000

Khi đó ta có: $C^A_{n_{ave}} = (18166.67, 20166.67, 21166.67) \approx (18000, 20000, 21500)$

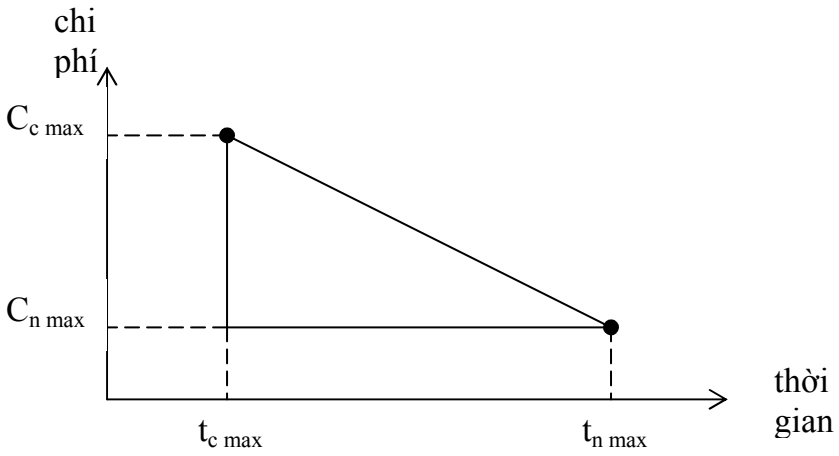
Tiến hành giải mờ con số trên ta được chi phí ước tính khoảng 20000.

Các nhóm chuyên gia khác sẽ ước lượng các giá trị còn lại cho tất cả các công việc trên đường găng của sơ đồ hoạch định công việc với kết quả như trong bảng bên dưới.

Để tiến hành rút ngắn thời gian của đường găng, PERT đưa ra khái niệm độ trượt giá được tính như trong công thức sau

$$k = \left| \frac{C_{n\max} - C_{c\max}}{t_{n\max} - t_{c\max}} \right|$$

Trong hình bên dưới, chúng ta có thể thấy rõ rằng khi giá trị $t_{n\ max}$ giảm về $t_{c\ max}$ thì giá trị $C_{n\ max}$ sẽ tăng lên giá trị $C_{c\ max}$



Giả sử rằng việc kết quả ước lượng của các chuyên gia và hệ số k tương ứng của các công việc trong dự án như trong bảng sau

Công việc	$t_{n\ max}$	$t_{c\ max}$	$C_{n\ max}$	$C_{c\ max}$	Hệ số k ($/ngày)
A	35	25	20000	26000	600
C	54	30	30500	40500	417
E	50	32	28000	35000	389
F	30	22	18500	25000	813
G	29	20	15000	19000	444
H	10	8	7000	8000	500

Ị ói chung việc trang bị thêm tài nguyên để rút ngắn thời gian hoàn tất dự án nên được bắt đầu với những công việc có hệ số trượt giá thấp nhất.

Công việc	Thời gian rút ngắn được $t_{n\ max}$	Chi phí vượt trội $C_{c\ max}$	Hệ số trượt giá ($/ngày)

	$-t_{c\ max}$	$-C_{n\ max}$	
E	18	7000	389
C	24	10000	417
G	9	4000	444
H	2	1000	500
A	10	6000	600
F	8	6500	813

Giả sử rằng người quản lý dự án muốn rút ngắn đội dài đường găng từ 208 ngày xuống còn 108 ngày, tức cần giảm 28 ngày. Khi đó với số liệu như bảng trên chúng ta thấy rằng trong số các công việc trên đường găng thì công việc E có hệ số trượt giá thấp nhất. Ị hư vậy bằng cách đầu tư thêm 7000$ chúng ta có thể rút ngắn được 18 ngày do đó cần phải rút thêm 10 ngày nữa. Công việc có hệ số k thấp nhất tiếp theo là công việc C và để rút ngắn thêm 10 ngày chúng ta cần đầu tư thêm 10 x 417 = 4170$.

BÀI TẬP

1. Cho 2 tập mờ A, B, C trong X = {1, 2, 3, 4, 5, 6, 7,}

X	1	2	3	4	5	6	7
μA	0.4	0.1	0.8	0	0.7	1	0.9
μB	0.7	0.1	0.3	0.4	0.8	0.9	0

   Tìm $A \cup B$, $A \cap B$, $A \backslash B$, $\neg A \cap \neg B$

2. Cho 3 tập mờ A, B, C trong X = {1, 2, 3, 4, 5, 6, 7, 8}

X	1	2	3	4	5	6	7	8
μA	0.8	0.3	0.5	0	0.4	1	0.6	0.1
μB	0.5	0.9	0.2	0.4	0.8	0.5	0	0.7
μC	0.2	0.1	0.8	0.5	0.3	0.7	0.9	0.3

   a. Kiểm chứng công thức
   $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
   $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

b. Tìm $A\backslash B$, $\neg A \cap B$, $\neg A \cap (\neg B \cup \neg C)$

3. Cho các tập mờ sau:

Tập mờ A định nghĩa dựa theo hàm thành viên: $\mu_A = \begin{cases} \frac{x}{3} & 0 \le x \le 3 \\ 1 & 3 \le x \le 5 \\ \frac{7-x}{2} & 5 \le x \le 7 \\ 0 & x \ge 7 \end{cases}$

Tập mờ B định nghĩa dựa theo hàm thành viên: $\mu_B = \begin{cases} \frac{x}{2} & 0 \le x \le 2 \\ \frac{4-x}{2} & 2 \le x \le 4 \\ 0 & x \ge 4 \end{cases}$

a. Vẽ đồ thị biểu diễn các tập mờ A và B.
b. Biểu diễn $\overline{A}, A \cap B, \overline{A \cup B}$ trên đồ thị.

Đề thi tham khảo trắc nghiệm giữa kỳ (30% tổng điểm môn học)

Trắc nghiệm:

1. Phát biểu nào là mệnh đề?
   a. Hôm này trời thật đẹp quá!
   b. Ị gày mai là thứ 7, bạn có biết không?
   c. Một con ngựa đau cả tàu bỏ cỏ.
   d. Thôi em hãy về, quê hương đang chờ em đó.
2. Chân trị của mệnh đề $\forall x \in R, [(x-3=0) \to (x^2-4x+3=0)]$ là:
   c. Đúng. b. Sai.
3. Chân trị của mệnh đề lượng tự hóa $\exists x \in R, \exists y \in R, [(x-2y=5) \wedge (2x+y=4)]$ là:
   a. Đúng. b. Sai.
4. Phủ định của mệnh đề $\exists x \in R, \forall y \in R, \forall z \in R, [(x^2 = y^2 + z^2) \to (x \ge z)]$ là:
   a. $\forall x \in R, \forall y \in R, \exists z \in R, [(x^2 = y^2 + z^2) \to (x \ge z)]$
   b. $\exists x \in R, \exists y \in R, \exists z \in R, [(x^2 \ne y^2 + z^2) \wedge (x \ge z)]$
   c. $\forall x \in R, \exists y \in R, \exists z \in R, [(x^2 = y^2 + z^2) \wedge (x < z)]$
   d. $\forall x \in R, \exists y \in R, \exists z \in R, [(x^2 \ne y^2 + z^2) \to (x < z)]$
   e. Tất cả đều sai
5. Phủ định của biểu thức mệnh đề $\overline{x}z \vee [(y \vee \overline{z}) \to xy]$
   a. $x\overline{z} \vee [(\overline{y} \vee z) \to \overline{x}\overline{y}]$
   b. $x\overline{z} \wedge [(\overline{y} \wedge z) \leftrightarrow \overline{x}\overline{y}]$

c. $x\bar{z} \to [(\bar{y} \wedge z) \wedge \bar{x}\bar{y}]$

d. $\bar{x}z \to [(y \vee \bar{z}) \to \bar{x} \wedge \bar{y}]$

e. Tất cả đều sai

6. F = $x\bar{y} \to [yz \vee yx)]$, biểu diễn F dưới dạng chuẩn hội hoàn toàn là:

a. $f = (\bar{x} \vee \bar{y} \vee z)(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee \bar{y} \vee \bar{z})$

b. $f = (x \vee \bar{y} \vee z)(x \vee \bar{y} \vee \bar{z})$

c. $f = (x \vee \bar{y} \vee z)(x \vee y \vee \bar{z})$

d. $f = (\bar{x} \vee y \vee z)(x \vee \bar{y} \vee \bar{z})(x \vee y \vee \bar{z})$

e. Tất cả đều sai

7. F = $xz \vee (y\bar{z} \to \bar{y}\bar{z})$, biểu diễn F dưới dạng chuẩn tuyển hoàn toàn là:

a. $f = xyz \vee x\bar{y}z \vee \bar{x}\bar{y}\bar{z} \vee x\bar{y}\bar{z} \vee \bar{x}\bar{y}z$

b. $f = xyz \vee x\bar{y}z \vee \bar{x}\bar{y}\bar{z} \vee x\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee x\bar{y}z$

c. $f = xyz \vee x\bar{y}z \vee \bar{x}\bar{y}\bar{z} \vee x\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee x\bar{y}z$

d. $f = xyz \vee x\bar{y}z \vee \bar{x}\bar{y}z \vee \bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}\bar{z} \vee x\bar{y}z$

e. Tất cả đều sai

8. F = $xy\bar{z} \vee \bar{x}z \vee x\bar{y}$, khi đó công thức đối ngẫu của F là:

a. F * = $\bar{x}\bar{y}z \vee x\bar{z} \vee \bar{x}y$

b. F * = $(\bar{x} \vee \bar{y} \vee z)(\bar{x} \vee z)(x \vee \bar{y})$

c. F * = $(\bar{x} \vee \bar{y} \vee z)(\bar{x} \vee z)(x \vee \bar{y})$

d. F * = $(x \vee y \vee \bar{z})(\bar{x} \vee z)(x \vee \bar{y})$

e. Tất cả đều sai

9. Công thức là $[x \to (y \vee x)] \wedge [(x \vee z) \to (xy)] \wedge (\bar{x} \to \bar{z})$

a. Công thức hằng đúng.

b. Công thức hằng sai

c. Công thức khả đúng/sai

10. Công thức là $(x \to y) \wedge (xz \to xy) \wedge (y \to z) \wedge \bar{x}$

a. Công thức hằng đúng.

b. Công thức hằng sai

c. Công thức khả đúng/sai

ĐỀ THI THAM KHẢO HẾT HỌC PHẦN (90 phút)

Câu 1(2đ). Xác định chân trị của các mệnh đề lượng từ hóa sau:

a. $\forall x \in R, [(x^2 - 6x + 5 = 0) \to (x - 3 < 2)]$

b. $\forall x \in R, \exists y \in R, [(x - y = 2) \wedge (x + y = -3)]$

c. $\exists x \in R, \exists y \in R, \forall z \in R, [(x^2 = y^2 - z^2) \to (x \geq z)]$

d. $\exists x \in R, \exists y \in R, \exists z \in R, [(2x^2 = 2y^2 - z^2) \to (x < z)]$

Câu 2 (1đ): Dùng các qui tắc suy diễn để chứng minh kết luận sau:

$\{\neg p \vee q\ ,\ (s \vee \neg\ q)\ ,\ (r \vee \neg s)\ ,\ p \wedge u\ \}$

$\therefore$ r, u

Câu 3 (2đ): Trong 1 trận thi đấu đối kháng võ thuật Vovinam, thí sinh sẽ được tính là 1 điểm nếu như có ít nhất 2 trong số 3 trọng tài phất cờ. Ị gười ta thiết kế 1 máy chấm điểm cho các trận đấu với thể thức thi đấu như vậy.

a. Tìm công thức logic tương ứng với máy chấm điểm này.

b. Hãy vẽ mạch điện tử tương ứng với công thức này (không rút gọn công thức).

Câu 4 (1đ): Cho $G = \overline{n} \to [m \to (\overline{m \vee \overline{n}} \vee p)]$, hãy đưa G về dạng thức chỉ dùng phép toán hội và phủ định để biểu diễn cho công thức trên.

Câu 5 (1đ): Hàm mờ $F(x,y) = \dfrac{2(x+y)}{2-xy}$ trong logic mờ là hàm chuẹn hay hàm đối chuẹn chuẹn ?

Câu 6 (3đ): Cho các tập mờ sau:

A = {(x1, 0.3), (x2, 0.3), (x3, 0.1), (x4, 1), (x5, 0), (x6, 0.3), (x7, 0.6)}

B = {(x1, 0.4), (x2, 0,6), (x3, 0.5), (x4, 0.8), (x5, 0.2), (x6, 0.9), (x7, 0.3)}

C = {(x1, 0), (x2, 0.2), (x3, 0.6), (x4, 1), (x5, 0.7), (x6, 0), (x7, 1)}

a. Tập mờ nào được gọi là đạt chuẹn ?

b. Tìm hgt, supp, core, card của các tập mờ trên.

c. Tìm $\overline{A} \cup C, (A \cap B \cap C), (\overline{A \cup B}) \cap C$.

ĐỀ TÀI CỘNG ĐIỂM CUỐI KỲ

1. Hãy tìm hiểu về các công cụ để vẽ mạch điện trong phần mềm MS. Visio và thực hiện vẻ tất cả các công thức hàm logic sau:
   a. $F = x\bar{y}z \vee \bar{x}\,z$
   b. $F = x\bar{y}z \oplus \bar{x}y\bar{z}$
   c. $F = x\bar{y} \leftrightarrow (y\bar{z} \oplus \bar{x}y)$
   d. $F = x\bar{y}zt \oplus \bar{x}yz \oplus \bar{z}t$
   e. $F = \overline{\bar{y}z \oplus \bar{x}y} \vee xyz$
   f. $F = x\bar{y}zt \vee \bar{x}y \vee xzt \vee x\bar{y}\bar{t}$
2. Tìm hiểu về sự phát triển và các thành tựu của ngành toán ứng dụng trong tin học.
3. Tìm hiểu về hình thành và phát triển của ngôn ngữ lập trình prolog.
4. Tìm hiểu về hình thành và phát triển của ngôn ngữ lập trình lisp.
5. Tìm hiểu những thành tựu có được từ việc ứng dụng logic mờ trong thời gian gần đây.
6. Lập trình trên ngôn ngữ prolog tất cả các bài tập về danh sách trong bài học số 6.

TÀI LIỆU THAM KHẢO

1. Toán học rời rạc ứng dụng trong tin học – Kenneth H. Rosen.
2. Toán rời rạc – ĐH KHTỊ – GS TS. Ị guyễn Hữu Anh.
3. Logic toán – Ị guyễn Văn Vĩnh, Ị guyễn Đức Đồng, Ị XB Thanh Hóa 2001.
4. Bài giảng môn Phương pháp toán trong tin học (phần logic) – ĐH KHTỊ 2004 của GS. TSKH Hoàng Kiếm
5. Hướng dẫn giải bài tập toán rời rạc – Đỗ Đức Giáo