Khoa Coâng Nghe Thoâng Tin

Tröôøng Ñaïi hoïc Caàn Thô

TRÍ TUỆ NHÂN TẠO

Artificial Intelligence: Structure and Strategies for Complex Problem Solving. (3rd edition - 1997)

George F. Luger, William A. Stubblefield

Giáo viên: Trần Ngân Bình

Nội Dung

- **Chương 1**. Giới thiệuTTNT
- **Chương 2.** Phép tính vị từ
- **Chương 3.** Cấu trúc và chiến lược dùng cho tìm kiếm trên không gian trạng thái (TK-KGTT)
- **Chương 4.** Tìm kiếm heuristic
- **Chương 5.** Điều khiển và cài đặt TK-KGTT
- **Chương 6:** Giải quyết vấn đề tri fthức chuyên sâu
- **Chương 7:** Suy luận với thông tin không chính xác hoặc không đầy đủ.
- **Chương 8.** Suy luận tự động (Automatic reasoning)
- **Chương 9.** Học máy

Trí Tuệ Nhân Tạo là gì?

- Là một nhánh của khoa học máy tính liên quan đến *sự tự động hóa hành vi thông minh*.

  Trí tuệ là gì?

- Các câu hỏi chưa có câu trả lời:
  - Liệu trí tuệ có phải là một khả năng duy nhất hay chỉ là một tên gọi cho một tập hợp các hành vi phân biệt và độc lập nhau?
  - Thế nào là khả năng sáng tạo?
  - Thế nào là trực giác?
  - Điều gì diễn ra trong quá trình học?
  - Có thể kết luận ngay về tính trí tuệ từ việc quan sát một hành vi hay không hay cần phải có biểu hiện của một cơ chế nào đó nằm bên trong ?

Định Nghĩa AI

- Rich, E. and K. Knight . 1991. *Artificial Intelligence.* New York: McGraw-Hill.

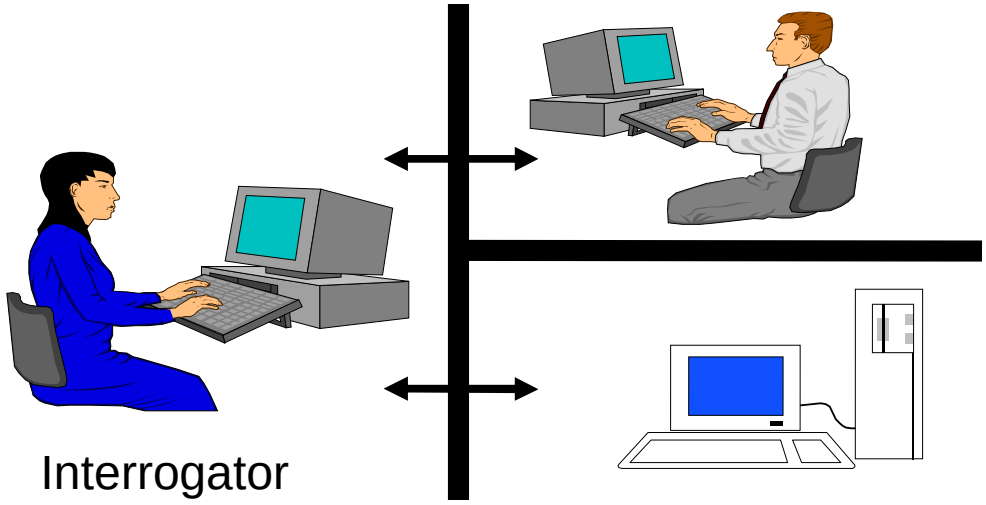
  "Artificial intelligence (AI) is the study of how to make computers do things which at the moment, people do better."

- George Luger:

"An AI approach problem-solving is one which:

- uses domain-specific knowledge
- to find a good-enough solution
- to a hard problem
- in a reasonable amount of time."

Turing Test



- Ưu điểm của Turing Test
  - Khái niệm khách quan về trí tuệ
  - Tránh đi những thảo luận về quá trình bên trong và ý thức
  - Loại trừ định kiến thiên vị của người thẩm vấn

Các ý kiến phản đối Turing Test

- Thiên vị các nhiệm vụ giải quyết vấn đề bằng ký hiệu
- Trói buộc sự thông minh máy tính theo kiểu con người, trong khi con người có:
  - Bộ nhớ giới hạn
  - Có khuynh hướng nhầm lẫn

Tuy nhiên, trắc nghiệm Turing đã cung cấp một cơ sở cho nhiều sơ đồ đánh giá dùng thực sự cho các chương trình TTNT hiện đại.

Các Ứng Dụng của TTNT

1. Trò chơi và các bài toán đố
2. Suy luận và chứng minh định lý tự động
3. Các hệ chuyên gia (các hệ tri thức)
4. Xử lý ngôn ngữ tự nhiên
5. Lập kế hoạch và người máy
6. Máy học
7. Mạng Neuron và giải thuật di truyền
8. …

Trí Tuệ Nhân Tạo - Đặc Điểm

- Sử dụng máy tính vào *suy luận trên các ký hiệu, nhận dạng qua mẫu, học, và các suy luận khác…*
- Tập trung vào các vấn đề "khó" *không thích hợp với các lời giải mang tính thuật toán.*
- Quan tâm đến các kỹ thuật giải quyết vấn đề sử dụng *các thông tin không chính xác, không đầy đủ, mơ hồ…*
- Cho lời giải 'đủ tốt' *chứ không phải là lời giải chính xác hay tối ưu.*
- Sử dụng heuristics – "bí quyết"
- Sử dụng *tri thức chuyên môn*
- …

Những vấn đề chưa được giải quyết

- Chương trình chưa tự sinh ra được heuristic
- Chưa có khả năng xử lý song song của con người
- Chưa có khả năng diễn giải một vấn đề theo nhiều phương pháp khác nhau như con người.
- Chưa có khả năng xử lý thông tin trong môi trường liên tục như con người.
- Chưa có khả năng học như con người.
- Chưa có khả năng tự thích nghi với môi trường.

TTNT =
Biểu Diễn + tìm kiếm

Hệ thống ký hiệu vật lý

- Hệ thống ký hiệu = tập hợp các ***mẫu*** và các ***quá trình***, trong đó các quá trình sản xuất, triệt tiêu và thay đổi các mẫu.
- Các *hành vi thông minh* đạt được bằng việc sử dụng:
  1. Các ***mẩu ký hiệu*** để biểu diễn các khía cạnh quan trọng của lĩnh vực bài toán.
  2. Các ***phép toán*** trên những mẫu này để sinh ra các lời giải có khả năng của bài toán..
  3. ***Tìm kiếm*** một lời giải trong số các khả năng này.

Giả thuyết về hệ thống ký hiệu vật lý

- "Một hệ thống ký hiệu vật lý có các phương tiện cần và đủ cho một hành vi thông minh tổng quát" (Nowell và Simon)

  Allen Newell and Herbert A. Simon, *Computer Science as Empirical Inquiry: Symbols and Search*, **Communications of the ACM** (March 1976)

TTNT như là sự biểu diễn và tìm kiếm

Sự biểu diễn phải:

- Cung cấp một cơ cấu tự nhiên để thể hiện tri thức/thông tin/ dữ liệu một cách đầy đủ => **Tính biểu đạt**
- Hỗ trợ việc thực thi một cách hiệu quả việc tìm kiếmđáp án cho một vấn đề => **Tính hiệu quả**

Liệu việc tìm kiếm:

– Có kết thúc không?

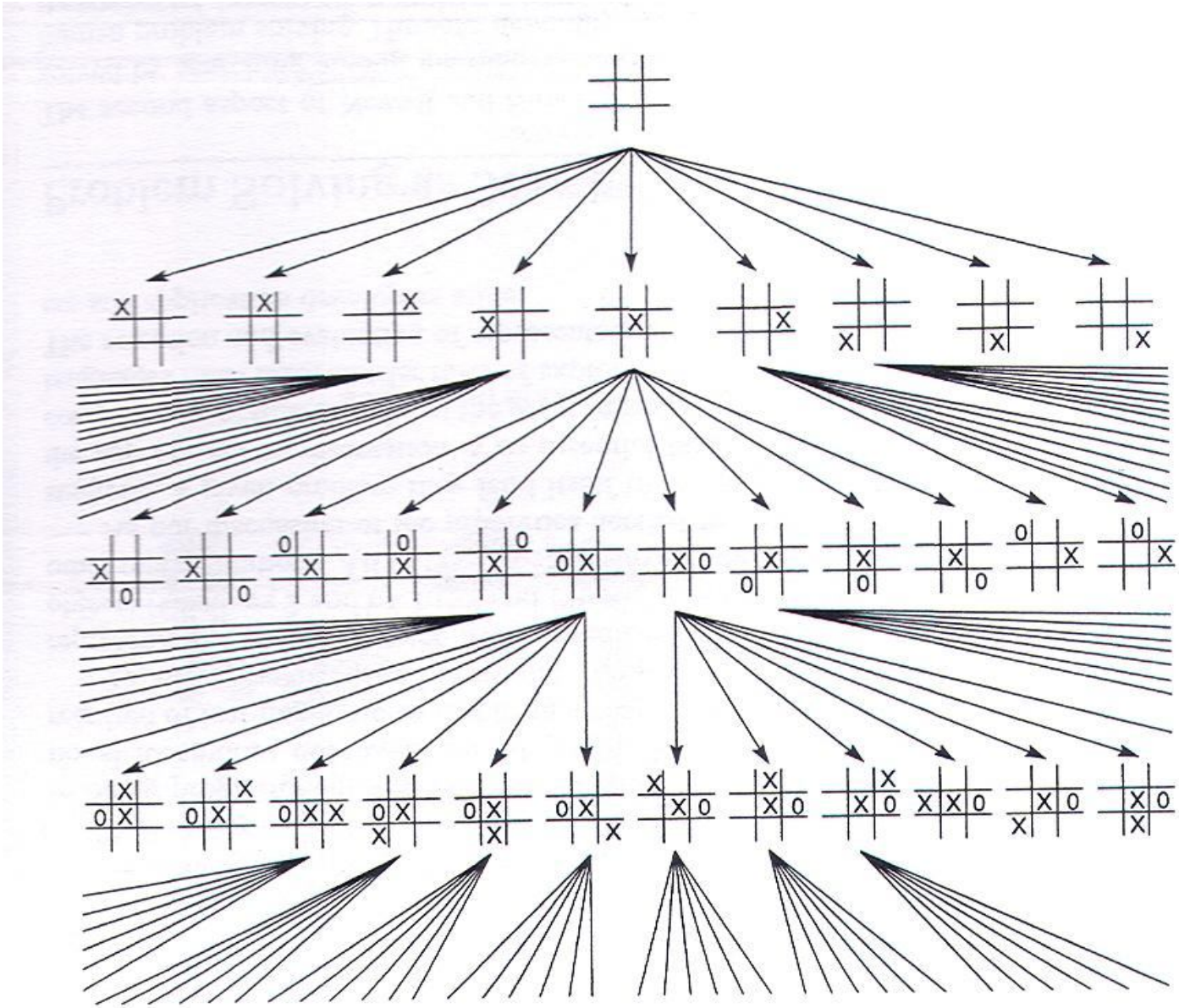
– Có chắc chắn sẽ tìm được lời giải không?

– Có chắc chắn sẽ tìm được lời giải tối ưu không?

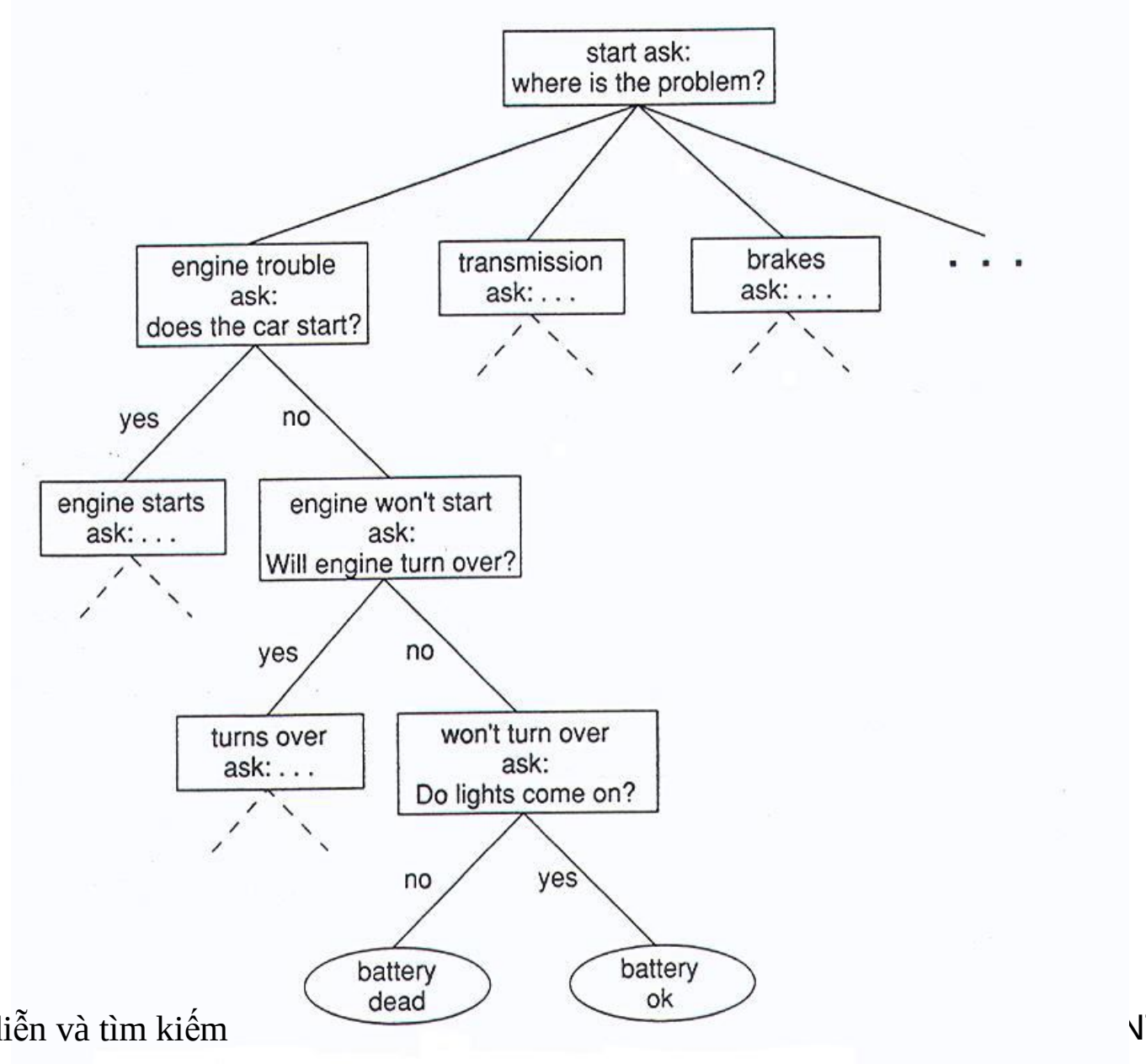
TTNT như là biểu diễn & tìm kiếm

- Giải quyết vấn đề như là sự tìm kiếm lời giải trong một *đồ thị không gian trạng thái*:
  - Nút ~ trạng thái (node ~ state)
  - Liên kết (link)
- Ví dụ:
  - Trò chơi tic-tac-toe
  - Chẩn đoán trục trặc máy móc trong ô tô

KGTT của Trò Chơi Tic-Tac-Toe



Chẩn đoán trục trặc máy móc trong ô tô



Chương 2 – Phép tính vị từ

- **Logic hình thức**
  - Logic hình thức = Biểu diễn + suy luận
  - Dùng như là một cơ chế biểu diễn tri thức
  - Dùng như là tìm kiếm không gian trạng thái trong các đồ thị And/Or
  - Dùng để hình thức hóa các luật heuristic
- **Có hai ngôn ngữ**:
  - Phép tính mệnh đề
  - Phép tính vị từ

Phép tính mệnh đề (1)

- **Mệnh đề:** là các câu khẳng định về thế giới.
- Mệnh đề có thể đúng (true) hoặc sai (false).
- Mệnh đề đơn giản:

Đồng là một kim loại	=>	Đúng
Gỗ là một kim loại	=>	Sai
Hôm nay là thứ Hai	=>	Sai

- Ký hiệu trong phép tính mệnh đề:
  - Ký hiệu mệnh đề: P, Q, R, S,...
  - Ký hiệu chân lý: true, false
  - Các phép toán logic: $\wedge$ (hội), $\vee$ (tuyển), $\neg$ (phủ định), $\Rightarrow$ (kéo theo) , = (tương đương)

Phép tính mệnh đề (2)

- Định nghĩa ***câu*** trong phép tính mệnh đề:
  - Mỗi ký hiệu mệnh đề, ký hiệu chân lý là một câu.
  - Phủ định của một câu là một câu.
  - Hội, tuyển, kéo theo, tương đương của hai câu là một câu.
- Ký hiệu ( ), [ ] được dùng để nhóm các ký hiệu vào các biểu thức con.
- Một *biểu thức mệnh đề* được gọi là một *câu* (hay *công thức dạng chuẩn*- WFF) $\Leftrightarrow$ nó có thể được tạo thành từ những ký hiệu hợp lệ thông qua một dãy các luật trên.

  Ví dụ: $((P \wedge Q) \Rightarrow R) = \neg P \vee \neg Q \vee R$

Ngữ Nghĩa của Phép Tính MĐ

- **Sự thông dịch** (Intepretation):
  - Là sự *gán giá trị chân lý* (T / F) cho các câu mệnh đề.
  - Là một sự khẳng định chân lý của các câu mệnh đề trong một thế giới khả hữu nào đó.
- Sự thông dịch của một câu kép thường được xác định bằng ***bảng chân lý***:

P	Q	$\neg$P	P$\wedge$Q	P$\vee$Q	P$\Rightarrow$Q	P=Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Sự Tương Đương của Phép Tính MĐ

- $\neg(\neg P) = P$
- $(P \vee Q) = (\neg P \Rightarrow Q)$
- Luật tương phản: $(P \Rightarrow Q) = (\neg Q \Rightarrow \neg P)$
- Luật De Morgan: $\neg(P \vee Q) = (\neg P \wedge \neg Q)$, và
  $\neg(P \wedge Q) = (\neg P \vee \neg Q)$
- Luật giao hoán: $(P \wedge Q) = (Q \wedge P)$, và $(P \vee Q) = (Q \vee P)$
- Luật kết hợp: $((P \wedge Q) \wedge R) = (P \wedge (Q \wedge R))$,
  $((P \vee Q) \vee R) = (P \vee (Q \vee R))$
- Luật phân phối: $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$,
  $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

Phép TínhVị Từ (1)

- **Ký hiệu vị từ** là tập hợp gồm các chữ cái, chữ số, ký hiệu "_", và được bắt đầu bằng chữ cái. VD: X3, tom_and_jerry
- Ký hiệu vị từ có thể là:
  - **ký hiệu chân lý**: true, false
  - **Hằng**: dùng để chỉ một đối tượng / thuộc tính trong thế giới.
    - Ký hiệu bắt đầu bằng chữ thường: VD: helen, yellow, rain
  - **Biến**: dùng để chỉ một lớp tổng quát các đối tượng / thuộc tính.
    - Ký hiệu bắt đầu bằng chữ hoa: VD: X, People, Students
  - **Hàm**: dùng để chỉ một hàm trên các đối tượng.
    - Ký hiệu bắt đầu bằng chữ thường: VD: father, plus
    - Mỗi ký hiệu hàm có một ngôi n, chỉ số lượng các đối số của hàm.
  - **Vị từ**: dùng để định nghĩa một mối quan hệ giữa không hoặc nhiều đối tượng.
    - Ký hiệu vị từ bắt đầu bằng chữ thường. VD: likes, equals, part_of

Phép TínhVị Từ (2)

- **Biểu thức hàm**: là một ký hiệu hàm theo sau bởi n đối số.
  VD: father(david) price(bananas) like(tom, football)
- **Mục** (term): là một hằng, một biến hay một biểu thức hàm
- **Câu sơ cấp**: là một hằng vị từ với n ngôi theo sau bởi n thành phần (mỗi thành phần là một mục) đặt trong dấu (), cách nhau bởi dấu ‘,’ và kết thúc với dấu ‘.’
  - Trị chân lý true, false là các câu sơ cấp.
  - Câu sơ cấp còn được gọi là: biểu thức sơ cấp (atomic expression), nguyên tử (atom) hay mệnh đề (proposition)

  VD: friends(helen, marry). likes(hellen, mary).
  likes(helen, sister(mary)). likes(X, ice-cream).

  Ký hiệu vị từ trong các câu này là friends, likes.

Phép TínhVị Từ (3)

- **Câu**: được tạo ra bằng cách kết hợp các câu sơ cấp sử dụng:
  - **Các phép kết nối logic**: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, =
  - **Các lượng tử biến**:
    - **Lượng tử phổ biến** $\forall$: dùng để chỉ một câu là đúng với mọi giá trị của biến lượng giá VD: $\forall$ X likes(X, ice-cream).
    - **Lượng tử tồn tại** $\exists$: dùng để chỉ một câu là đúng với một số giá trị nào đó của biến lượng giá. VD: $\exists$Y friends(Y,tom).

  VD:

  likes(helen, chocolat) $\wedge$ $\neg$ likes(bart, chocolat).

  $\exists$ X foo(X,two,plus(two,three)) $\wedge$ equal(plus(three,two),five)

(foo(two, two,plus(two,three))) $\Rightarrow$ (equal(plus(three,two),five)= true).

Ngữ Nghĩa của Phép Tính Vị Từ

- **Sự thông dịch** của một tập hợp các câu phép tính vị từ: là một sự gán các thực thể trong miền của vấn đề đang đề cập cho mỗi ký hiệu hằng, biến, vị từ và hàm.
- Giá trị chân lý của một câu sơ cấp được xác định qua sự thông dịch. Đối với các câu không phải là câu sơ cấp, sử dụng bảng chân lý cho cho các phép nối kết, và:
  - Giá trị của câu ∀ **X** **<câu>** là true nếu <câu> là T cho tất cả các phép gán có thể được cho X.
  - Giá trị của câu ∃ **X** **<câu>** là true nếu tồn tại một phép gán cho X làm cho <câu> có giá trị T.

Phép Tính Vị Từ Bậc Nhất

- **Phép tính vị từ bậc nhất** cho phép các biến lượng giá tham chiếu đến các đối tượng trong miền của vấn đề đang đề cập nhưng KHÔNG được tham chiếu đến các vị từ và hàm.
- VD không hợp lệ: $\forall$(Likes) Likes(helen, ice-cream)
- VD hợp lệ:
  - *Nếu ngày mai trời không mưa, tom sẽ đi biển.*
    - $\neg$weather(rain, tomorrow) $\Rightarrow$ go(tom, sea)
  - *Tất cả các cầu thủ bóng rổ đều cao.*
    - $\forall$ X (basketball_player(X) $\Rightarrow$ tall(X))
  - *Có người thích coca-cola*
    - $\exists$ X person(X) $\wedge$ likes(X, coca-cola)
  - *Không ai thích thuế*
    - $\neg$ $\exists$X likes(X, taxes)

Ví dụ về phép tính vị từ

- Cho trước:

  mother(eve,abel) mother(eve, cain)

  father(adam, abel) father(adam,cain)

  $\forall X\ \forall Y$ father(X,Y) $\vee$ mother(X,Y) $\Rightarrow$ parent(X,Y)

  $\forall X\ \forall Y\ \exists Z$ parent(Z,X) $\wedge$ parent(Z,Y) $\Rightarrow$ sibling(X,Y)

- Có thể suy luận:

  parent(eve,abel) parent(eve, cain)

  parent(adam,abel) parent(adam,cain)

  sibling(abel, cain) sibling(cain, abel)

  sibling(abel,abel) sibling(cain,cain) *!không có nghĩa*

Các luật suy diễn

- **Luật Modus Ponens** (MP):

$$\frac{P \Rightarrow Q \quad P}{Q}$$

- **Luật Modus Tolens** (MT):

$$\frac{P \Rightarrow Q \quad \neg Q}{\neg P}$$

- **Luật triển khai phổ biến** (Universal Instantiation):

$$\frac{\forall X\ P(X) \quad a \text{ thuộc miền xác định của } X}{P(a)}$$

Ví Dụ

"Tất cả mọi người đều chết và Socrates là người, do đó Socrates sẽ chết"

=> $\forall X\ man(X) \Rightarrow mortal(X)$ (1)

$man(socrates)$ (2)

Từ (1),(2) bằng luật UI, ta có:

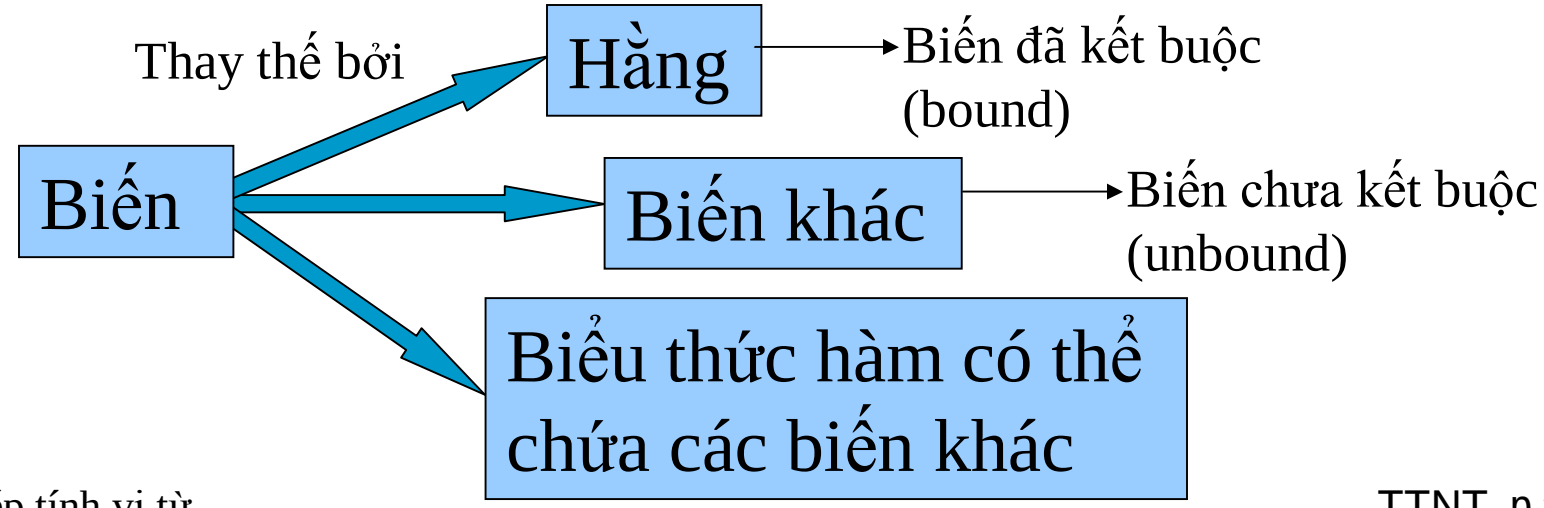
$man(socrates) \Rightarrow mortal(socrates)$ (3)

Từ (3) và (2) bằng luật MP, ta có:

$mortal(bill)$ (4)

Đối sánh mẫu và phép hợp nhất

- Để áp dụng các luật như MP, một hệ suy diễn phải có khả năng xác định khi nào thì hai biểu thức là ***một*** hay còn gọi là ***đối sánh*** (match).
- **Phép hợp nhất** là một giải thuật dùng để xác định những phép thế (substitution) cần thiết để làm cho hai biểu thức vị từ đối sánh nhau.
- Một biến có thể thay thế bởi một ***mục*** bất kỳ:



"Giải thuật" Đối Sánh Mẫu

1. **Hằng / hằng** đối sánh : chỉ khi chúng giống hệt nhau
   VD: tom không đối sánh với jerry
2. **Hằng a / biến X** đối sánh:
   a. Biến chưa kết buộc: biến trở thành kết buộc với hằng
      => Khi đó ta có phép thế {a/X}
   a. Biến đã kết buộc : xem (1)
3. **Biến X/ biến Y** đối sánh:
   a. Hai biến chưa kết buộc: luôn luôn đối sánh
      => Khi đó ta có phép thế {X/Y}
   a. Một biến kết buộc và một biến chưa kết buộc: xem (2)
   b. Hai biến kết buộc: xem (1)
4. **Biểu thức / biểu thức** đối sánh: chỉ khi các tên hàm hoặc vị từ, số ngôi giống nhau thì áp dụng đối sánh từng đối số một.
   VD: goo(X) - không đối sánh với foo(X) hay goo(X,Y)
   - đối sánh với goo(foo(Y)) với phép thế {foo(Y) / X}

Phạm vi của một biến

- Phạm vi của một biến là một câu.
- Một khi biến đã bị kết buộc, các phép hợp nhất theo sau và các suy luận kế tiếp phải giữ sự kết buộc này

VD:

man(X) => mortal(X)

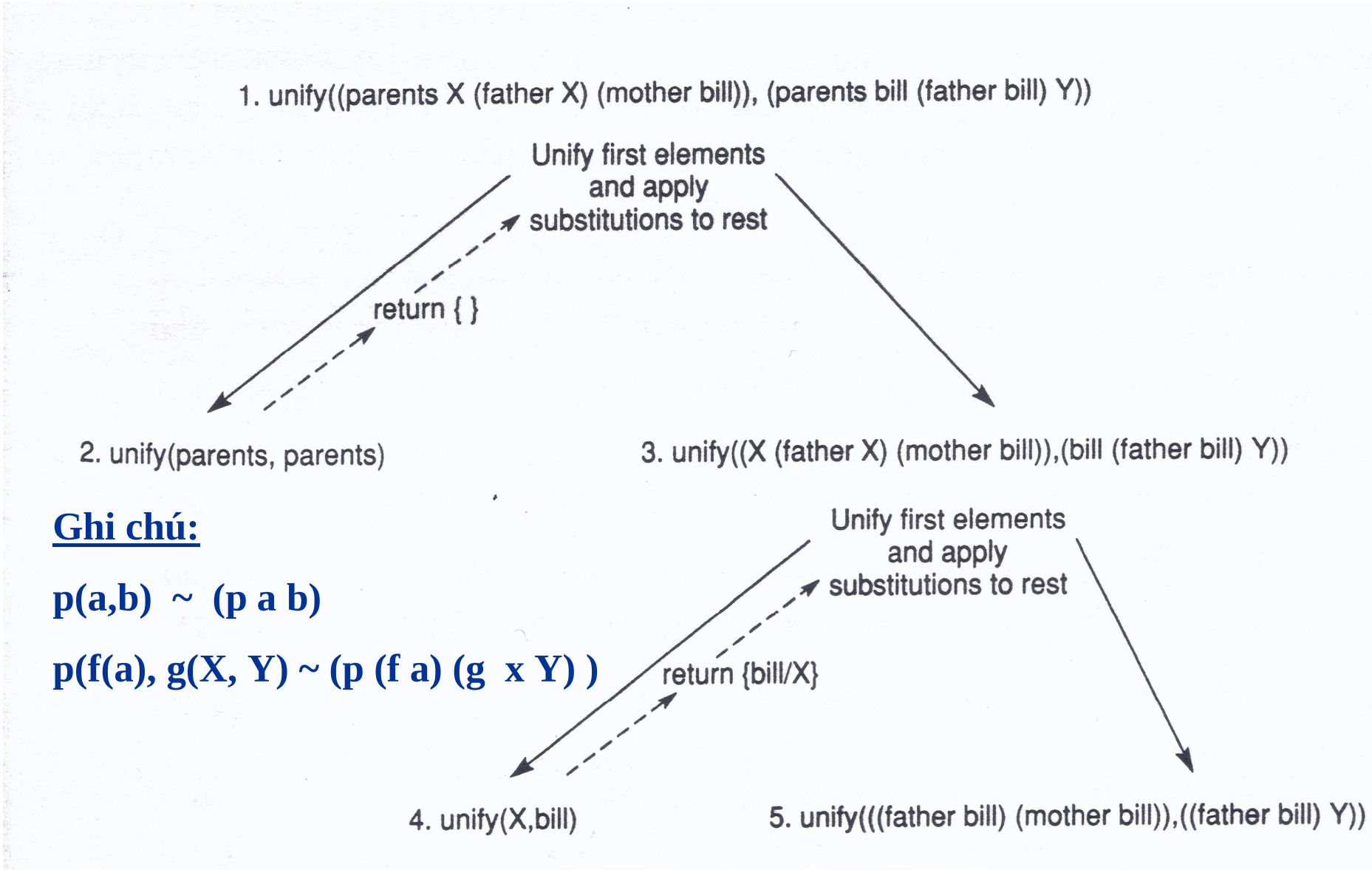
Nếu ta thế X bởi socrates thì ta được:

man(socrates) => mortal(socrates)

Ví dụ: Biểu thức đối sánh

- Hãy xác định xem foo(X,a,goo(Y)) có đối sánh với các biểu thức sau hay không? Nếu có thì cho biết phép thế tương ứng:
  - foo(X,b,foo(Y))
  - foo(fred, a, goo(Z))
  - foo(X,Y)
  - moo(X,a,goo(Y))
  - foo(Z,a,goo(moo(Z)))
  - foo(W,a,goo(jack))
- Cho biết kết quả có được khi hợp nhất p(a, X) với :
  - p(Y,Z) => q(Y,Z)
  - q(W,b) => r(W,b)

Thủ tục hợp nhất "Unify"



Ghi chú:

p(a,b) ~ (p a b)

p(f(a), g(X, Y) ~ (p (f a) (g x Y))

Tích các phép thế hợp nhất (Composition)

- Nếu S và S’ là hai tập hợp phép thế, thì tích của S và S’ được xác định bằng cách áp dụng S’ cho những phần tử của S và bổ sung kết quả này vào S.

  VD: {X/Y, W/X}, {V/X}, {a/V, f(b)/W}

  => {a/Y, f(b)/Z}

Hợp tử tổng quát nhất (Most General Unifier)

- Yêu cầu của giải thuật hợp nhất là hợp tử (unifier) càng tổng quát càng tốt: đó là hợp tử tổng quát nhất tìm thấy cho hai biểu thức.

  VD: Khi hợp nhất p(X) và p(Y):

  $\Rightarrow$ *không nên chọn* {fred/X, fred/Y} vì fred không phải là hợp tử tổng quát nhất

  $\Rightarrow$ *nên chọn* {Z/Y, Z/Y}

Ứng Dụng: Hệ tư vấn tài chính (1)

Hệ tư vấn tài chính hoạt động theo các nguyên tắc sau:

- Các cá nhân không đủ tiền tiết kiệm nên tăng tiền tiết kiệm, bất kể thu nhập là bao nhiêu.
- Các cá nhân có đủ tiền tiết kiệm và đủ thu nhập nên xem xét việc đầu tư vào chứng khoán.
- Các cá nhân với thu nhập thấp nhưng đủ tiền tiết kiệm có thể chia phần thu nhập thêm vào tiết kiệm và chứng khoán.

Với:

- tiết kiệm đủ là 5000$/ người phụ thuộc
- Thu nhập đủ 15000$ + (4000$ / người phụ thuộc)

Ứng Dụng: Hệ tư vấn tài chính (2)

Xâu dựng hệ thống logic với các câu vị từ như sau:

1. savings_account(inadequate) $\Rightarrow$investment(saving)
2. savings_account(adequate) $\wedge$income(adequate) $\Rightarrow$ investment(stocks)
3. savings_account(adequate) $\wedge$income(inadequate) $\Rightarrow$ investment(combination)
4. $\forall$X amount_saved(X) $\wedge\exists$Y(dependents(Y) $\wedge$ greater(X,minsavings(Y))) $\Rightarrow$ savings_account(adequate)
5. $\forall$X amount_saved(X) $\wedge\exists$Y(dependents(Y) $\wedge$ $\neg$greater(X,minsavings(Y))) $\Rightarrow$ savings_account(inadequate)
6. $\forall$X earning(X, steady) $\wedge\exists$Y(dependents(Y) $\wedge$ greater(X,minincome(Y))) $\Rightarrow$ income(adequate)
7. $\forall$X earning(X, steady) $\wedge\exists$Y(dependents(Y) $\wedge$ $\neg$greater(X,minincome(Y))) $\Rightarrow$ income(inadequate)
8. $\forall$X earning(X, unsteady) $\Rightarrow$ income(inadequate)

With: minavings(X) = 5000 * X minincome(X)=15000+(4000*X)

Ứng Dụng: Hệ tư vấn tài chính(3)

Một nhà đầu tư với tình trạng như sau:

9. amount_saved(22000)
10. earnings(25000,steady)
11. dependents(3)

$\Rightarrow$ investment (?)

Dùng phép hợp nhất và luật Modus Ponens, suy ra:

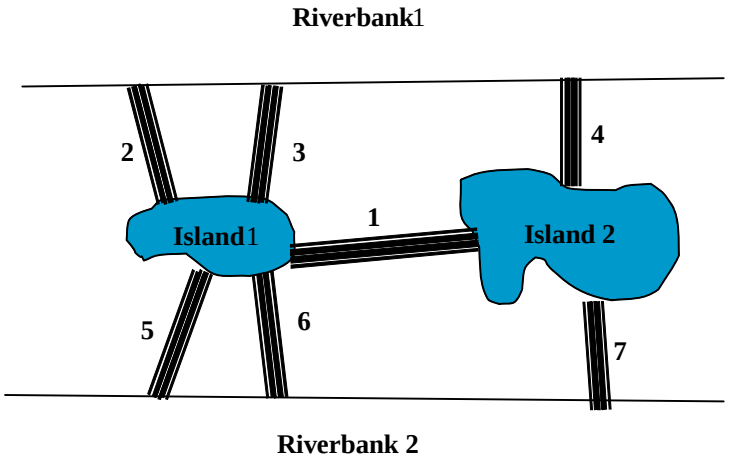
12. income(inadequate)
13. savings_account(adequate)

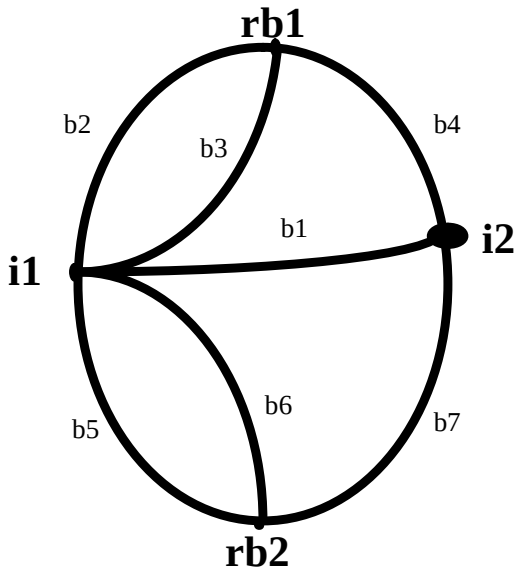
$\Rightarrow$ investment (combination)

Bài Tập Chương 2

Chương 3 - Cấu trúc và chiến lược cho TK - KGTT

- Khi biểu diễn một vấn đề như là một đồ thị không gian trạng thái, chúng ta có thể sử dụng lý thuyết đồ thị để phân tích cấu trúc và độ phức tạp của các vấn đề cũng như các thủ tục tìm kiếm.





Hệ thống cầu thành phố Konigsberg và biểu diễn đồ thị tương ứng

Nội dung chương 3

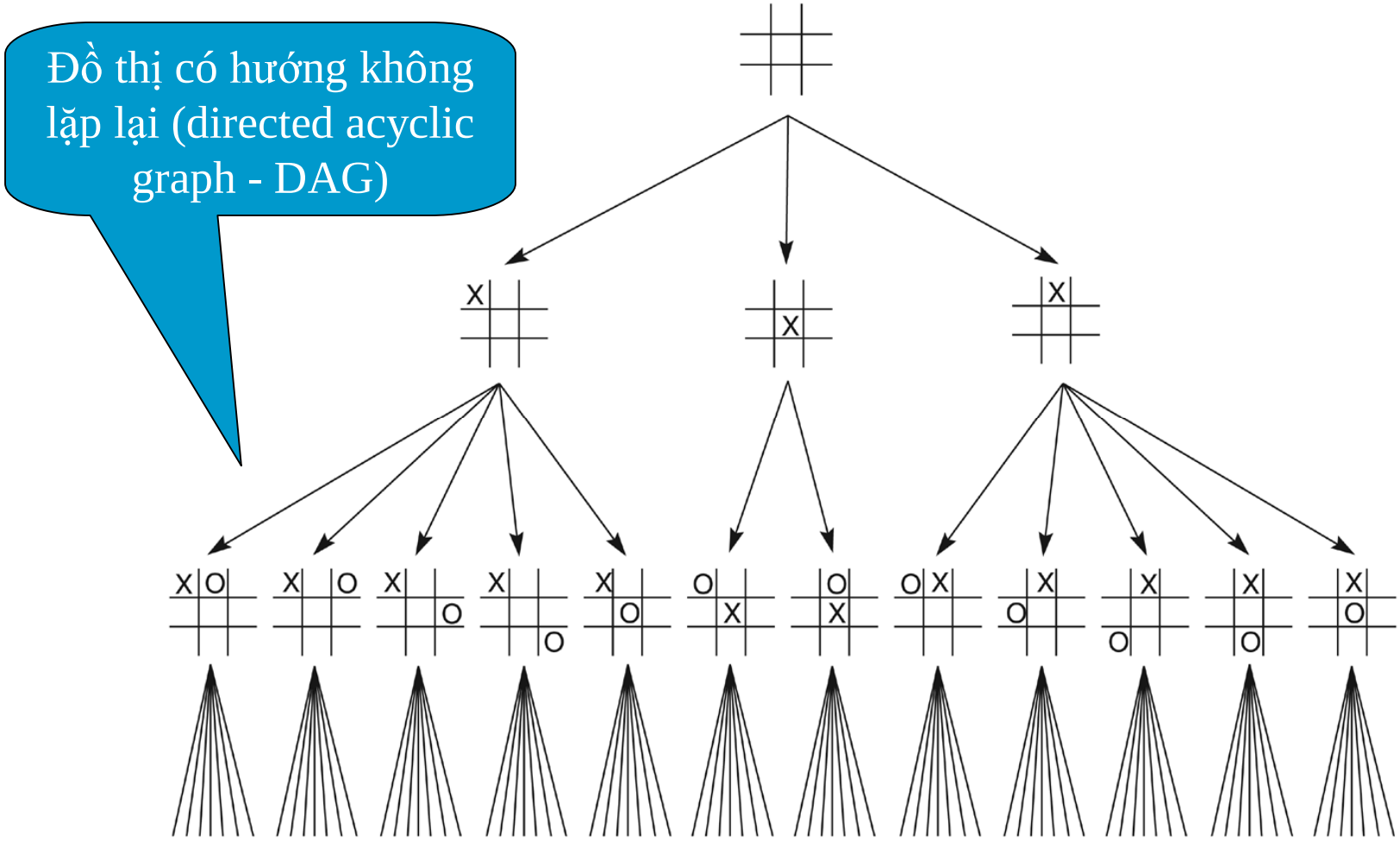
- Định nghĩa Không Gian Trạng Thái
- Các chiến lược tìm kiếm trên không gian trạng thái:
  - TK ***hướng từ dữ liệu*** (data – driven)
  - TK ***hướng từ mục tiêu*** (goal – driven).
- Tìm kiếm trên không gian trạng thái:
  - ***TK rộng*** (breath – first search)
  - ***TK sâu*** (depth – first search)
  - ***TK sâu bằng cách đào sâu nhiều lần*** (depth – first search with iterative deepening)
- Sử dụng không gian trạng thái để biểu diễn suy luận với phép tính vị từ: ***Đồ thị Và/Hoặc*** (And/Or Graph)

ĐN: KHÔNG GIAN TRẠNG THÁI

Một **KGTT** (state space) là 1 bộ [N, A, S, GD] trong đó:

- **N** (node) là các *nút* hay các *trạng thái* của đồ thị.
- **A** (arc) là tập các *cung* (hay các *liên kết*) giữa các nút.
- **S** (solution) là một tập chứa các *trạng thái đích* của bài toán.($S \subset N \wedge S \neq \varnothing$)
- Các trạng thái trong **GD** (Goal Description) được mô tả theo một trong hai đặc tính:
  - Đặc tính có thể đo lường được các trạng thái gặp trong quá trình tìm kiếm. VD: Tic-tac-toe, 8-puzzle,…
  - Đặc tính của đường đi được hình thành trong quá trình tìm kiếm. VD: TSP
- ***Đường đi của lời giải*** (solution path) là một con đường đi qua đồ thị này từ một nút thuộc S đến một nút thuộc GD.

Một phần KGTT triển khai trong Tic-tac-toe



Trò đố 8 ô hay 15 ô

Trạng thái ban đầu Trạng thái đích

- Trò đố 15 ô

11	14	4	7
10	6		5
1	2	13	15
9	12	8	3

1	2	3	4
12	13	14	5
11		15	6
10	9	8	7

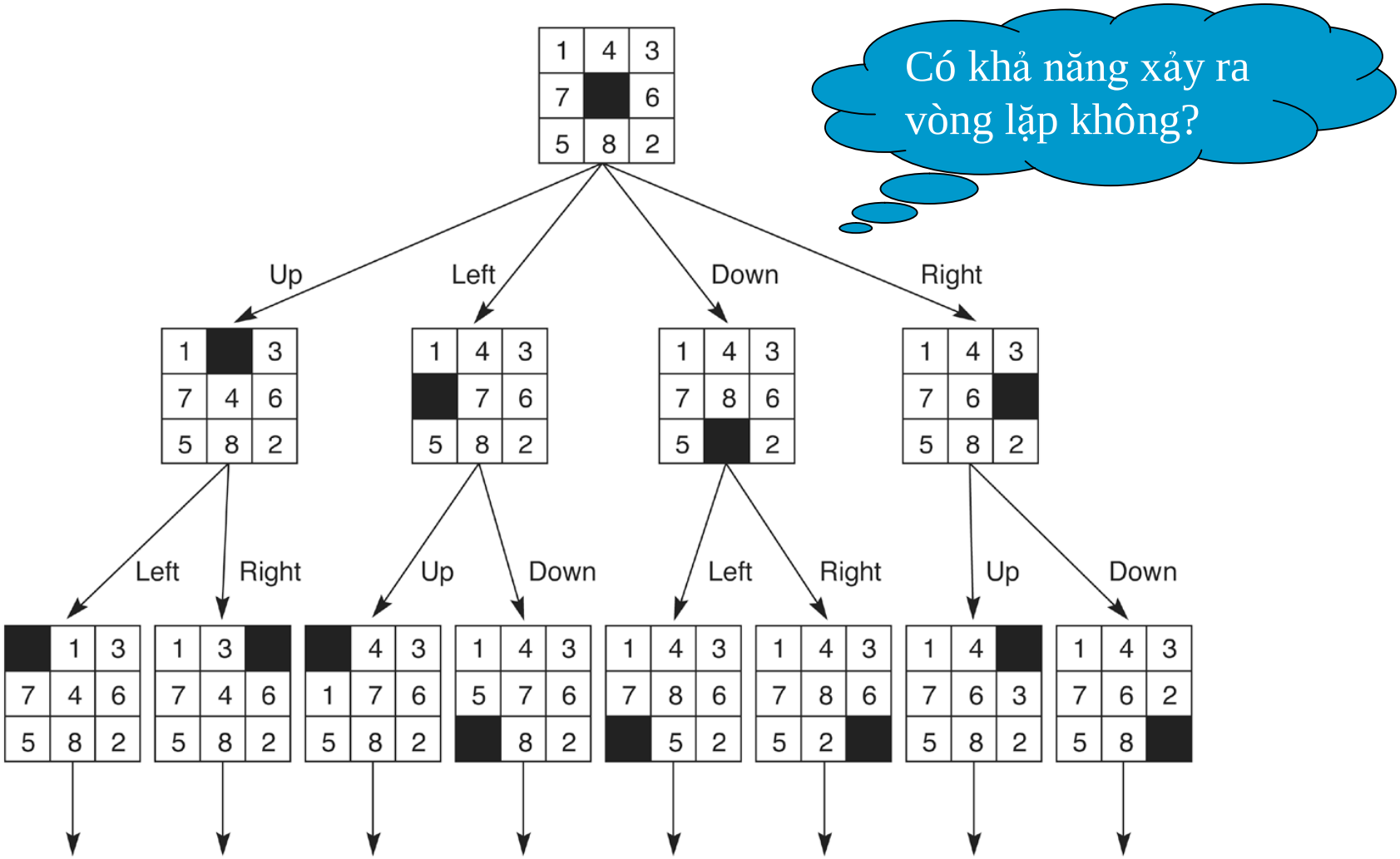
- Trò đố 8 ô

	2	8
3	5	7
6	2	1

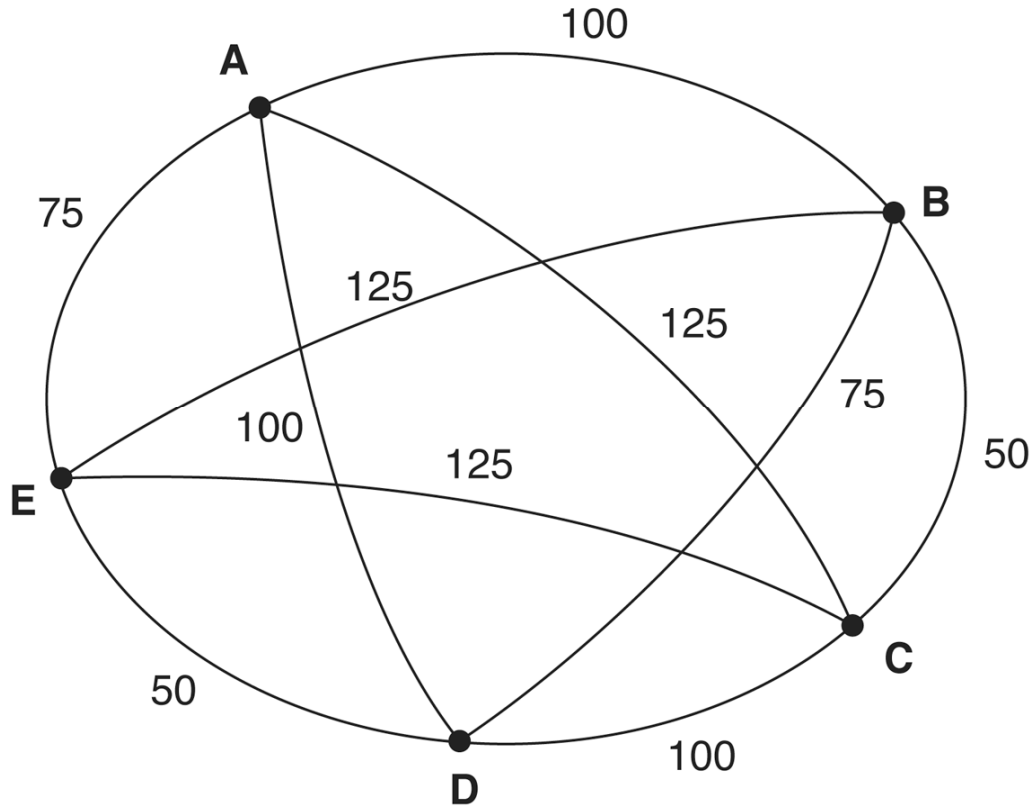
1	2	3
8		4
7	6	5

- Cần biểu diễn KGTT cho bài toán này như thế nào?

KGTT của 8-puzzle sinh ra bằng phép "di chuyển ô trống"

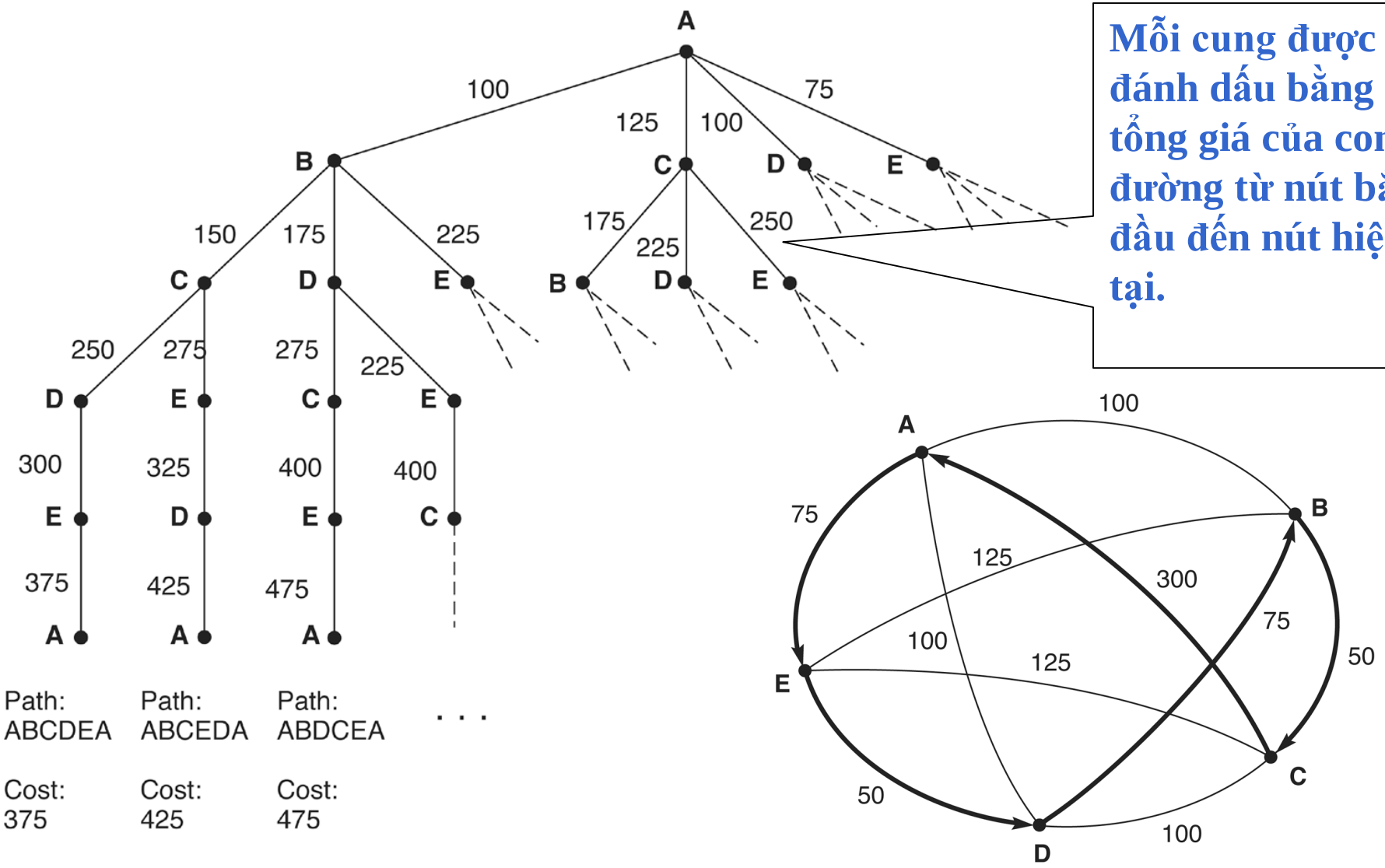


Một ví dụ của bài toán TSP



- Cần biểu diễn KGTT cho bài toán này như thế nào?

KGTT của bài toán TSP

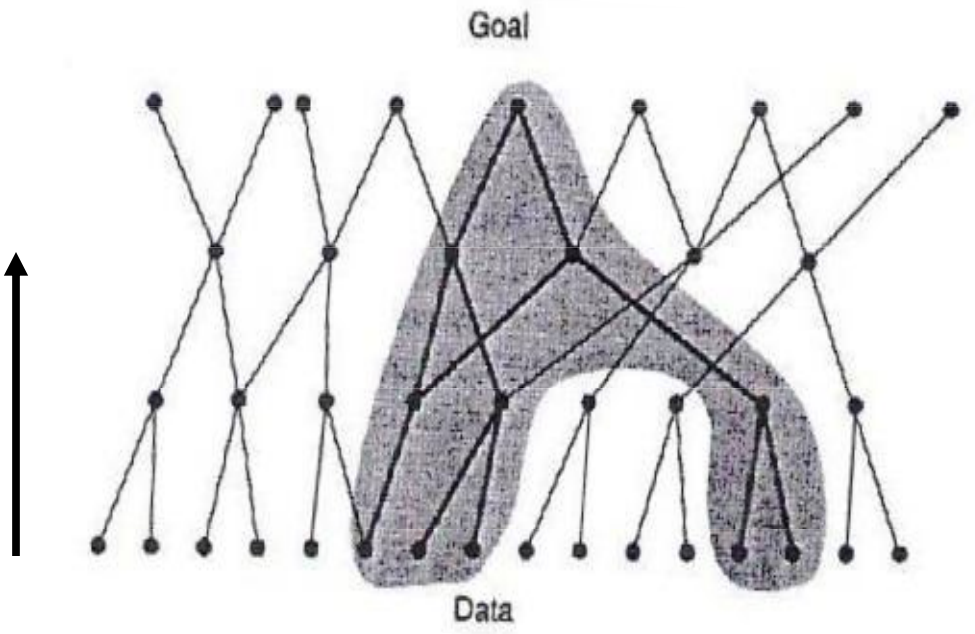


Các Chiến Lược cho TK-KGTT

- **TK hướng từ dữ liệu (Data-driven Search)**
  - Suy diễn tiến (forward chaining)
- **TK hướng từ mục tiêu (Goal-driven Search)**
  - Suy diễn lùi (backward chaining)

TK Hướng từ Dữ Liệu

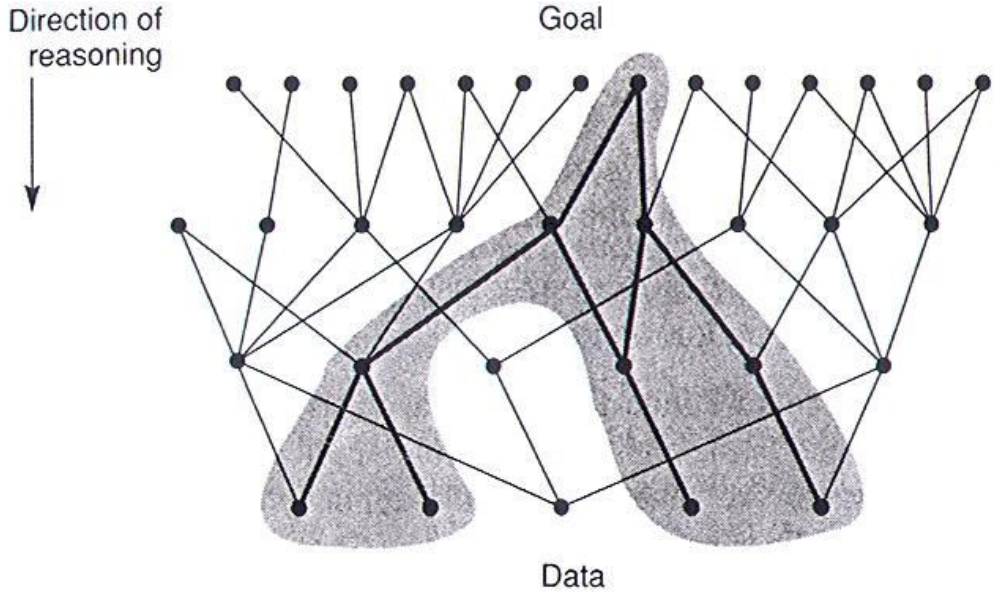
- Việc tìm kiếm đi từ dữ liệu đến mục tiêu



- Thích hợp khi:
  - Tất cả hoặc một phần dữ liệu được cho từ đầu.
  - Có nhiều mục tiêu, nhưng chỉ có một số ít các phép toán có thể áp dụng cho một trạng thái bài toán.
  - Rất khó đưa ra một mục tiêu hoặc giả thuyết ngay lúc đầu.

TK Hướng Từ Mục Tiêu

- Việc tìm kiếm đi từ mục tiêu trở về dữ liệu.



- Thích hợp khi:
  - Có thể đưa ra mục tiêu hoặc giả thuyết ngay lúc đầu.
  - Có nhiều phép toán có thể áp dụng trên 1 trạng thái của bài toán => sự bùng nổ số lượng các trạng thái.
  - Các dữ liệu của bài toán không được cho trước, nhưng hệ thống phải đạt được trong quá trình tìm kiếm.

Các phương pháp tìm kiếm trên đồ thị KGTT:

Phát triển từ giải thuật quay lui (**back – tracking):**

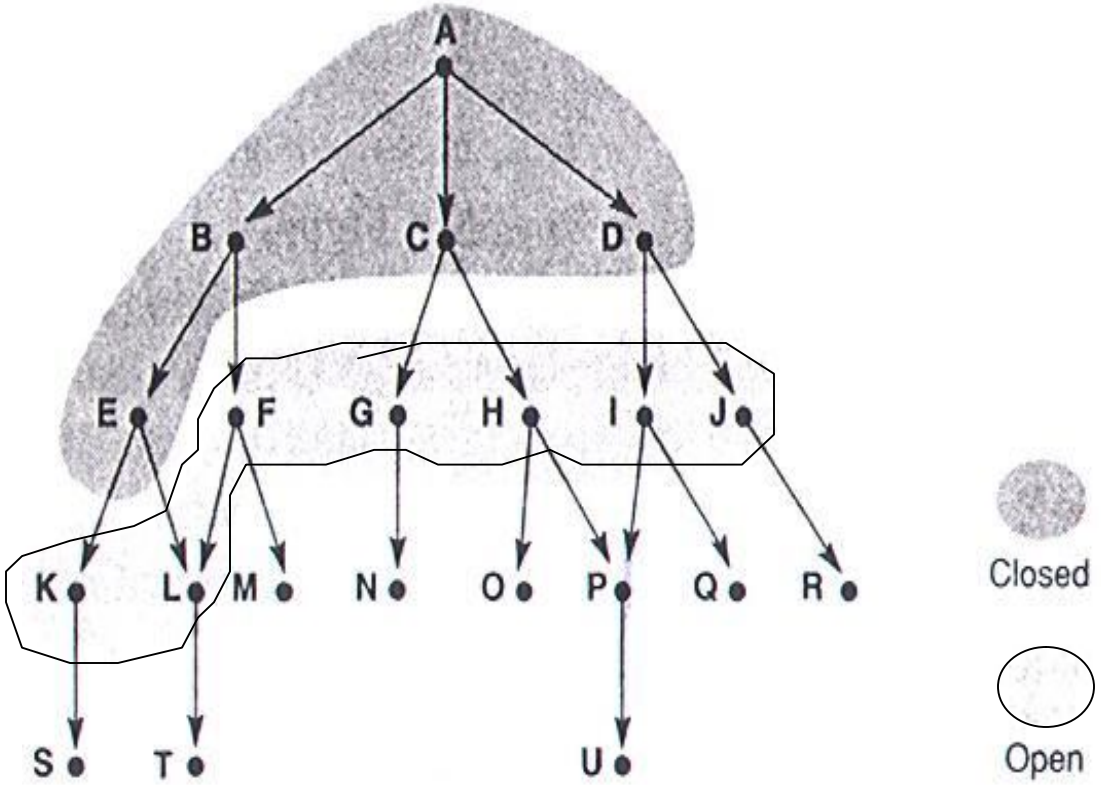
- ***Tìm kiếm rộng*** (breath-first search)
- ***Tìm kiếm sâu*** (depth-first search)
- ***TK sâu bằng cách đào sâu nhiều lần*** (depth-first search with iterative deepening)

Tìm Kiếm Rộng

1. Open = [A]; closed = []
2. Open = [B,C,D];
   closed = [A]
2. Open = [C,D,E,F];
   closed = [B,A]
3. Open = [D,E,F,G,H];
   closed = [C,B,A]
4. Open = [E,F,G,H,I,J];
   closed = [D,C,B,A]
5. Open = [F,G,H,I,J,K,L];
   closed = [E,D,C,B,A]
6. Open = [G,H,I,J,K,L,M];
   (vì L đã có trong open);

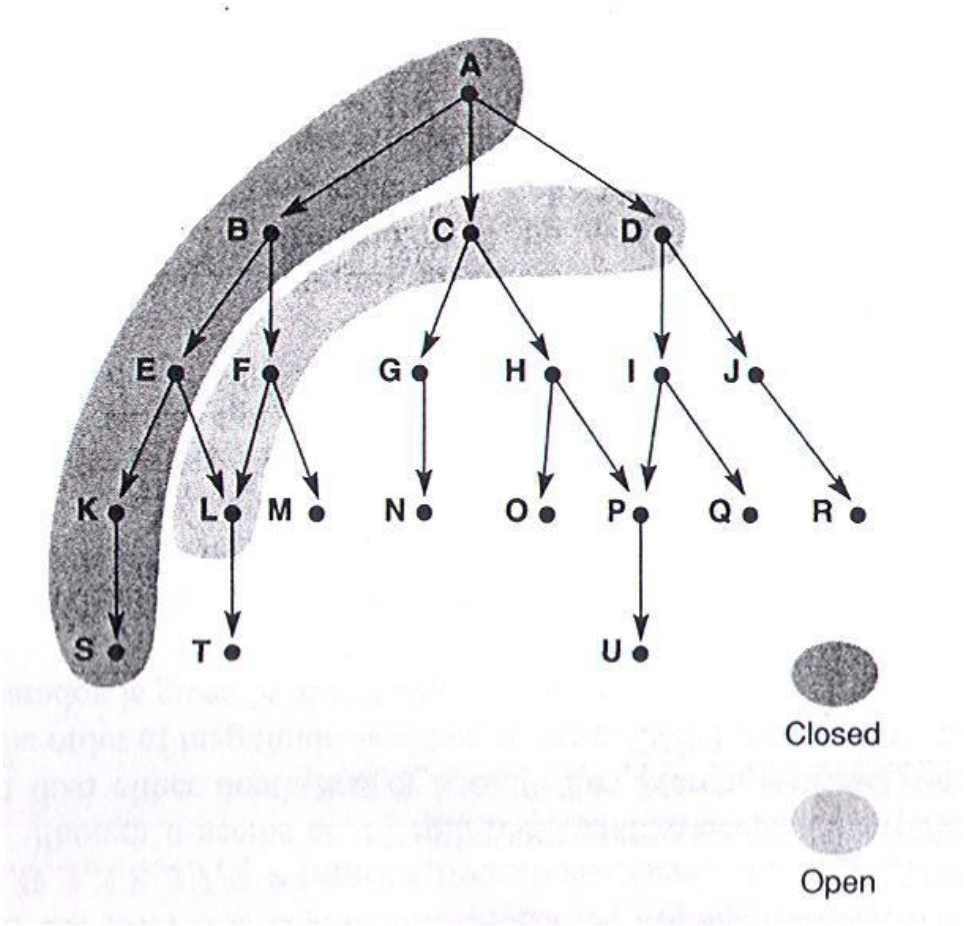
   closed = [F,E,D,C,B,A]

   …



Tìm kiếm Sâu

1. Open = [A]; closed = []
2. Open = [B,C,D]; closed = [A]
3. Open = [E,F,C,D];closed = [B,A]
4. Open = [K,L,F,C,D];
   closed = [E,B,A]
5. Open = [S,L,F,C,D];
   closed = [K,E,B,A]
6. Open = [L,F,C,D];
   closed = [S,K,E,B,A]
7. Open = [T,F,C,D];
   closed = [L,S,K,E,B,A]
8. Open = [F,C,D];
   closed = [T,L,S,K,E,B,A]
9. …



Tìm Kiếm Sâu hay Rộng? (1)

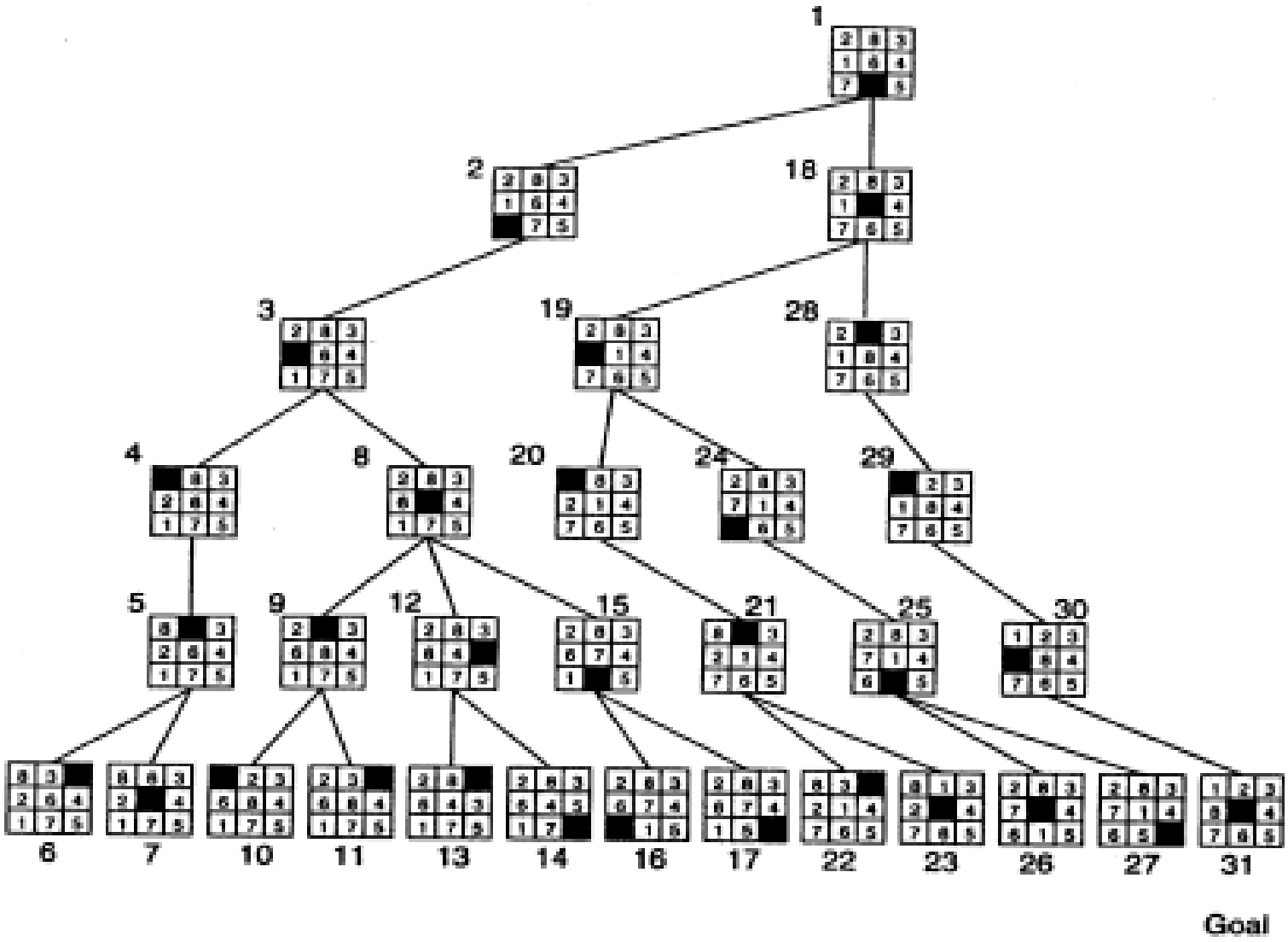
- Có cần thiết tìm một *đường đi ngắn nhất* đến mục tiêu hay không?
- Sự *phân nhánh* của không gian trạng thái
- Tài nguyên về *không gian* và *thời gian* sẵn có
- *Khoảng cách trung bình* của đường dẫn đến trạng thái mục tiêu.
- Yêu cầu đưa ra *tất cả các lời giải* hay chỉ là lời giải tìm được đầu tiên.

Tìm kiếm sâu bằng cách đào sâu nhiều lần (depth-first iterative deepening)

- Độ sâu giới hạn (depth bound): giải thuật TK sâu sẽ quay lui khi trạng thái đang xét đạt đến độ sâu giới hạn đã định.
- TK Sâu bằng cách đào sâu nhiều lần: TK sâu với độ sâu giới hạn là 1, nếu thất bại, nó sẽ lặp lại GT TK sâu với độ sâu là 2,… GT tiếp tục cho đến khi tìm được mục tiêu, mỗi lần lặp lại tăng độ sâu lên 1.
- GT này có độ phức tạp về thời gian cùng bậc với TK Rộng và TK Sâu.

Trò chơi ô đố 8-puzzle

The 8-puzzle searched by a production system with loop detection and depth bound 5



Đồ thị Và/Hoặc

- Sử dụng KGTT để biểu diễn suy luận với phép tính vị từ
- Là phương pháp *qui bài toán về các bài toán con.*
- Một tập hợp các mệnh đề / câu vị từ tạo thành một đồ thị Và/Hoặc (And/Or graph) hay siêu đồ thị (hypergraph).
- Trong đồ thị Và/Hoặc:
  - Các nút AND biểu thị sự phân chia bài toán, tất cả các bài toán con phải được chứng minh là đúng.
  - Các nút OR biểu thị các chiến lược giải quyết bài toán khác nhau, chỉ cần chứng minh một chiến lược đúng là đủ
- Có thể áp dụng TK theo kiểu hướng từ dữ liệu hay từ mục tiêu.
- Trong giải thuật cần ghi nhận diễn tiến của quá trình.

Ví dụ Đồ thị Và/Hoặc

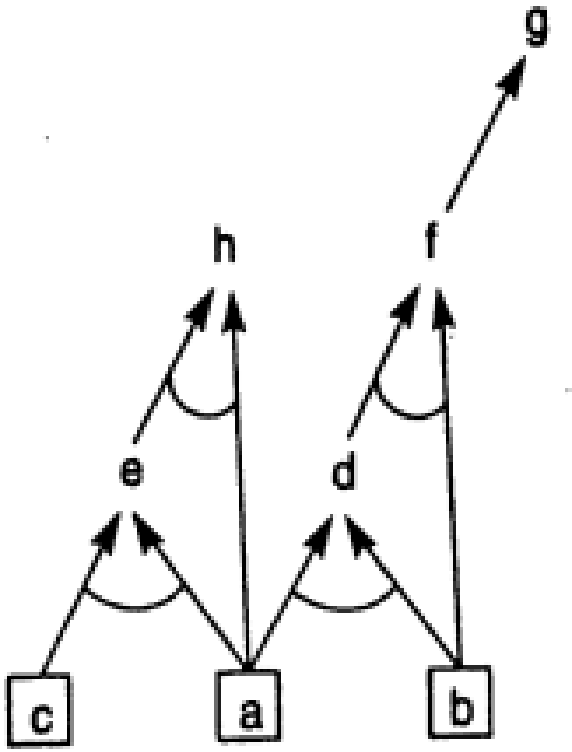
- Giả sử một tình huống với các mệnh đề sau:

  a b c

  $a \wedge b \Rightarrow d$ $a \wedge c \Rightarrow e$

  $b \wedge d \Rightarrow f$ $f \Rightarrow g$

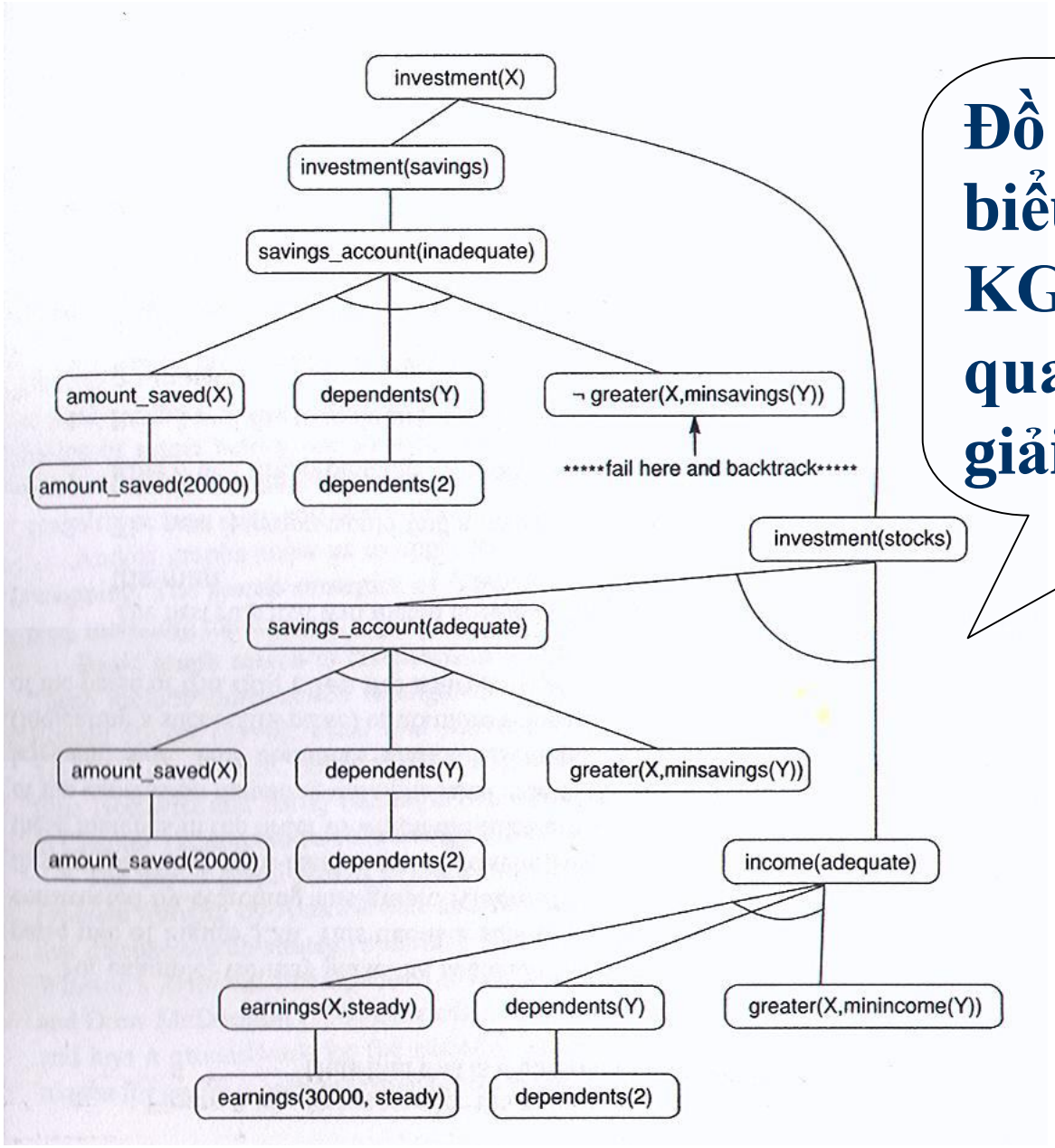
  $a \wedge e \Rightarrow h$



Hãy trả lời các câu hỏi sau:

1. h có đúng không?
2. h có cón đúng nêu b sai?

Ví dụ: Hệ Tư Vấn Tài Chính



Đồ Thị And/Or biểu diễn phần KGTT đã duyệt qua để đi đến lời giải

VÍ DỤ ĐỒ THỊ AND/OR:

Cho một bài toán được mô tả bằng các câu vị từ:

⇨Hãy vẽ đồ thị AND/OR biểu diễn phần KGTK để trả lời câu hỏi: "Fred đang ở đâu?" (Áp dụng suy diễn lùi)

1. Fred is a collie.
   collie(fred).
2. Sam is Fred's master.
   master(fred,sam).
3. The day is Saturday.
   day(saturday).
4. It is cold on Saturday.
   ¬ (warm(saturday)).
5. Fred is trained.
   trained(fred).
6. Spaniels are good dogs and so are trained collies.
   ∀ X[spaniel(X) ∨ (collie(X) ∧ trained(X)) → gooddog(X)]
7. If a dog is a good dog and has a master then he will be with his master.
   ∀ (X,Y,Z) [gooddog(X) ∧ master(X,Y) ∧ location(Y,Z) → location(X,Z)]
8. If it is Saturday and warm, then Sam is at the park.
   (day(saturday) ∧ warm(saturday)) → location(sam,park).
9. If it is Saturday and not warm, then Sam is at the museum.
   (day(saturday) ∧ ¬ (warm(saturday))) → location(sam,museum).

Bài Tập Chương 3

Chương 4 – Tìm kiếm heuristic

- **Heuristics**: là các phỏng đoán, ước chừng dựa trên kinh nghiệm, trực giác.
- Các hệ giải quyết AI **sử dụng heuristic trong hai tình huống cơ bản**:
  - Bài toán được định nghĩa chính xác nhưng *chi phí tìm lời giải bằng TK vét cạn là không thể chấp nhận*.

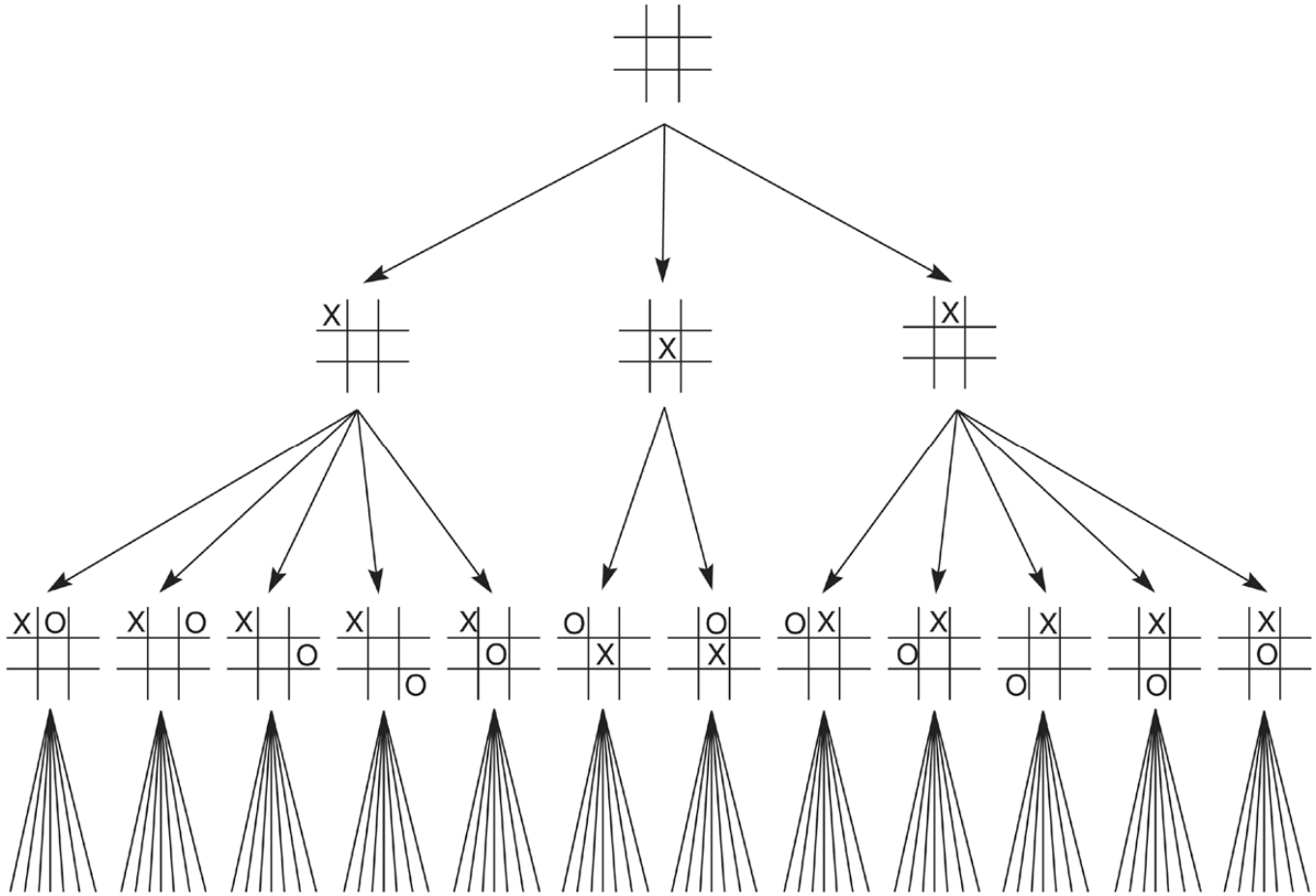
    VD: Sự bùng nổ KGTT trong trò chơi cờ vua.
  - *Vấn đề với nhiều sự mơ hồ* trong lời phát biểu bài toán hay dữ liệu cũng như tri thức sẵn có.

    VD: Chẩn đoán trong y học.

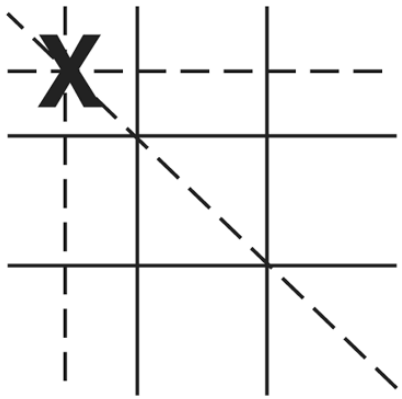
Giải Thuật Heuristic

- Một giải thuật heuristic có thể được xem gồm 2 phần:
  - Phép đo heuristic: thể hiện qua *hàm đánh giá heuristic (evaluation function)*, dùng để đánh giá các đặc điểm của một trạng thái trong KGTT.
  - Giải thuật tìm kiếm heuristic:
    - *Giải thuật leo núi (hill-climbing)*
    - *TK tốt nhất (best-first search)*

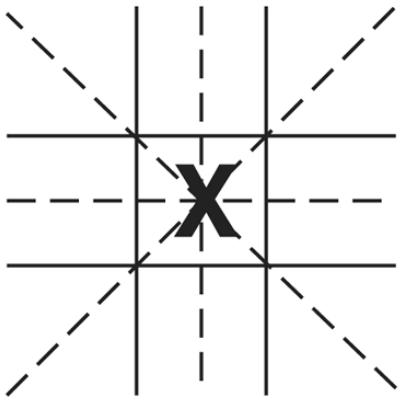
KGTT của tic-tac-toe được thu nhỏ nhờ tính đối xứng của các trạng thái.



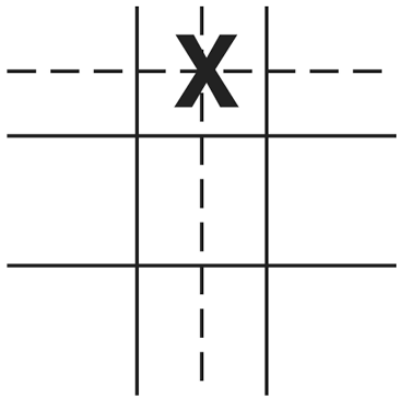
Phép đo heuristic (2)



Three wins through
a corner square



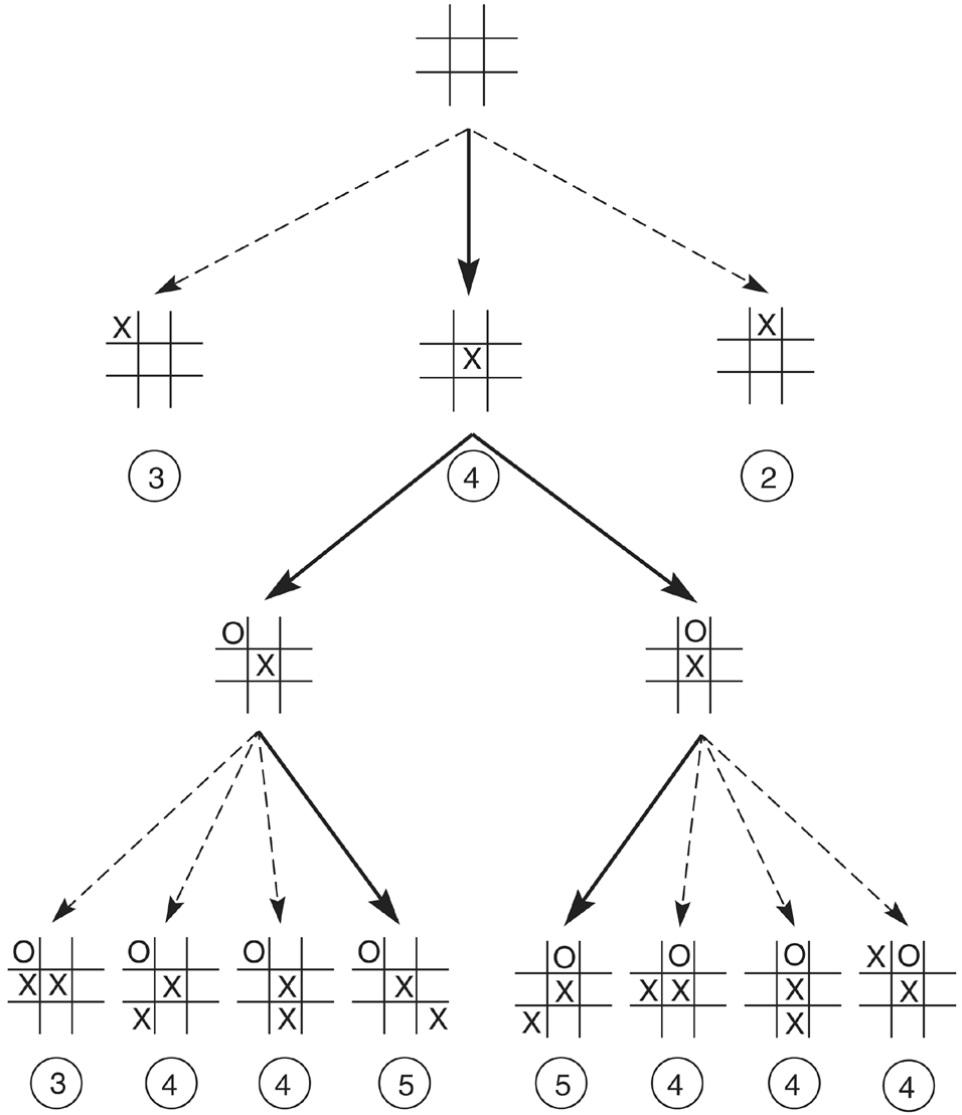
Four wins through
the center square



Two wins through
a side square

Heuristic "*Số đường thắng nhiều nhất*" áp dụng cho các nút con đầu tien trong tic-tac-toe.

KGTT càng thu nhỏ khi áp dụng heuristic

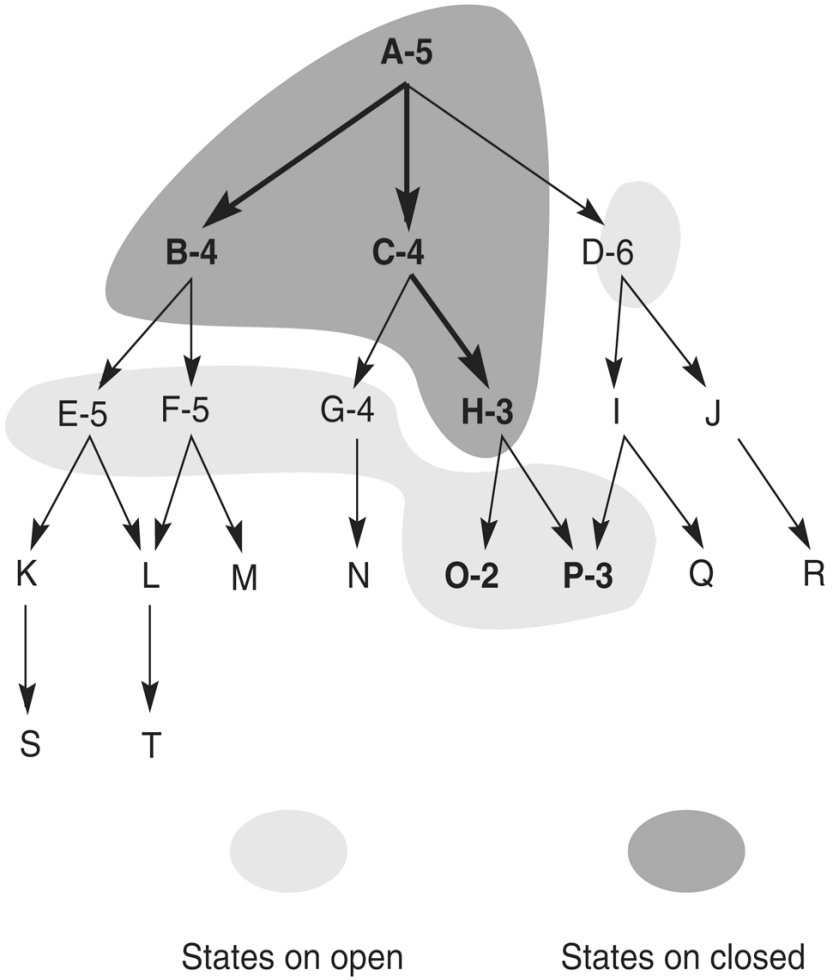


Giải thuật Leo Núi

- Giải thuật:
  - Mở rộng trạng thái hiện tại và đánh giá các trạng thái con của nó bằng hàm đánh giá heuristic.
  - Con "tốt nhất" sẽ được chọn để đi tiếp.
- Giới hạn:
  - Giải thuật có khuynh hướng bị sa lầy ở những cực đại cục bộ:
    - Lời giải tìm được không tối ưu
    - Không tìm được lời giải mặc dù có tồn tại lời giải
  - Giải thuật có thể gặp vòng lặp vô hạn do không lưu giữ thông tin về các trạng thái đã duyệt.

Giải thuật TK Tốt Nhất

1. open = [A5]; closed = []
2. Đánh giá A5; open = [B4,C4,D6]; closed = [A5]
3. Đánh giá B4;
   open = [C4,E5,F5,D6];
   closed = [B4,A5]
4. Đánh giá C4;
   open = [H3,G4,E5,F5,D6];
   closed = [C4,B4,A5]
5. Đánh giá H3;
   open = [O2,P3,G4,E5,F5,D6];
   closed = [H3,C4,B4,A5]
6. Đánh giá O2;
   open = [P3,G4,E5,F5,D6];
   closed = [O2,H3,C4,B4,A5]
7. Đánh giá P3; tìm được lời giải!



Cài Đặt Hàm Đánh Giá (Evaluation Function)

Xét trò chơi 8-puzzle. Cho mỗi trạng thái n một giá trị f(n):

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

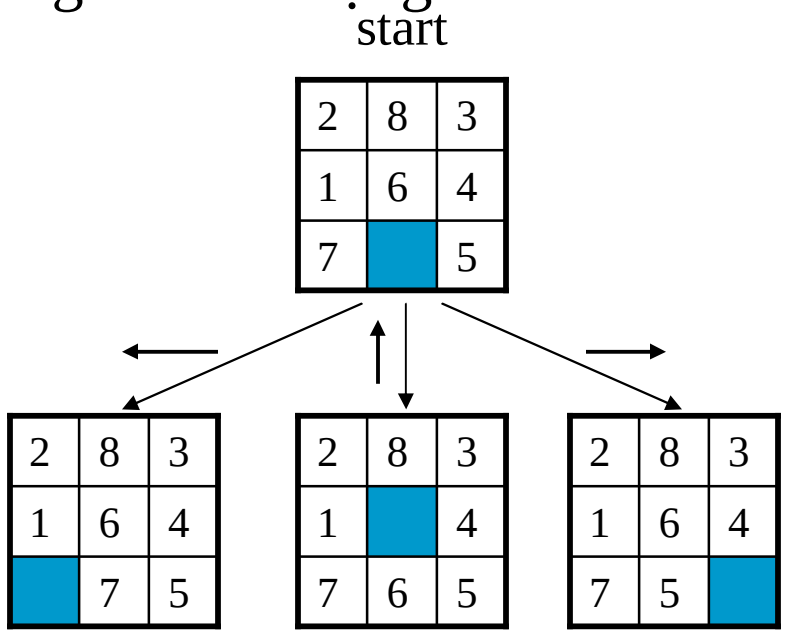
g(n) = khoảng cách thực sự từ n đến trạng thái bắt đầu

h(n) = hàm heuristic đánh giá khoảng cách từ trạng thái n đến mục tiêu.

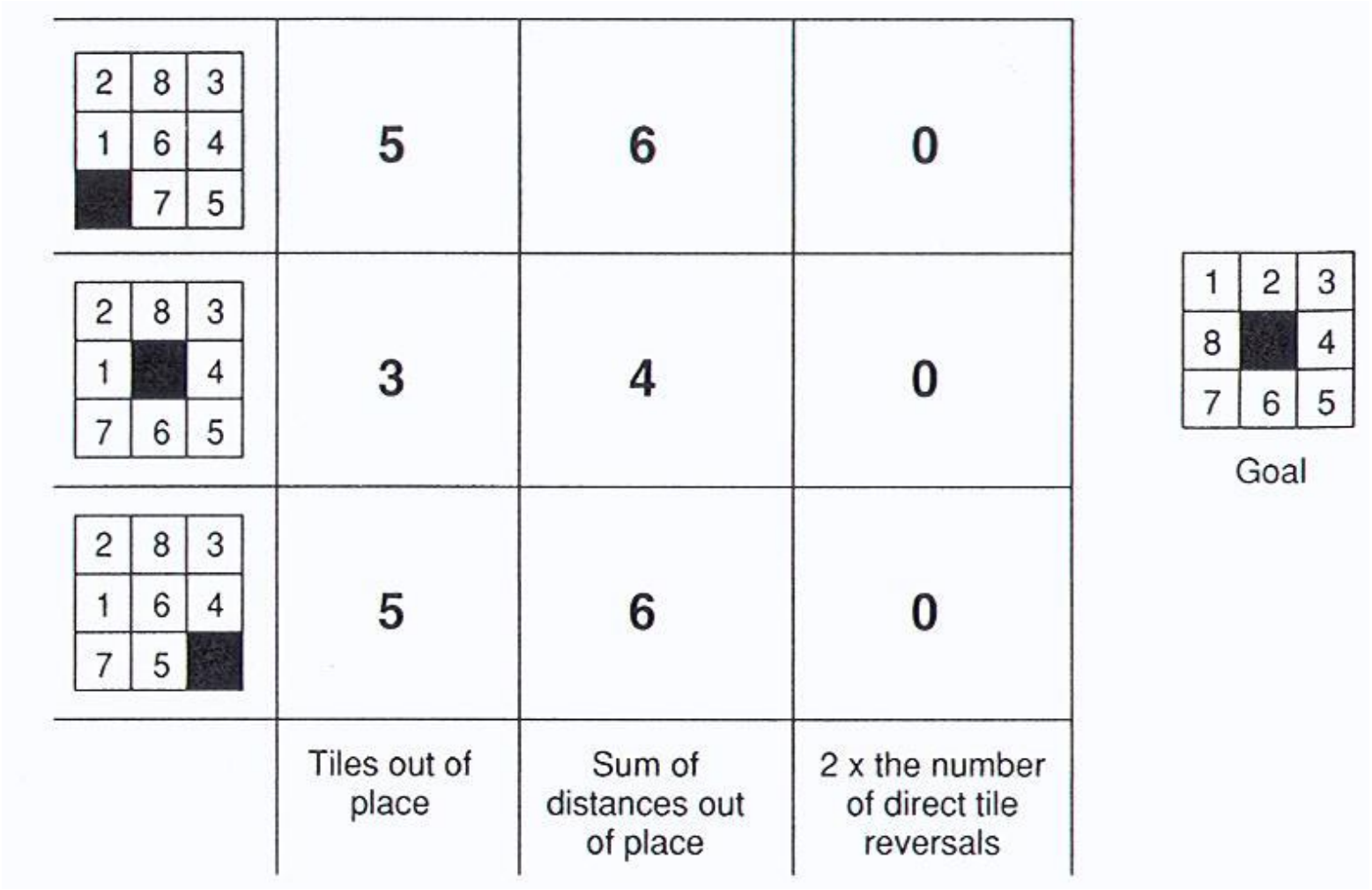
1	2	3
8		4
7	6	5

goal

h(n): số lượng các vị trí còn sai



Khó khăn trong thiết kế hàm heuristic



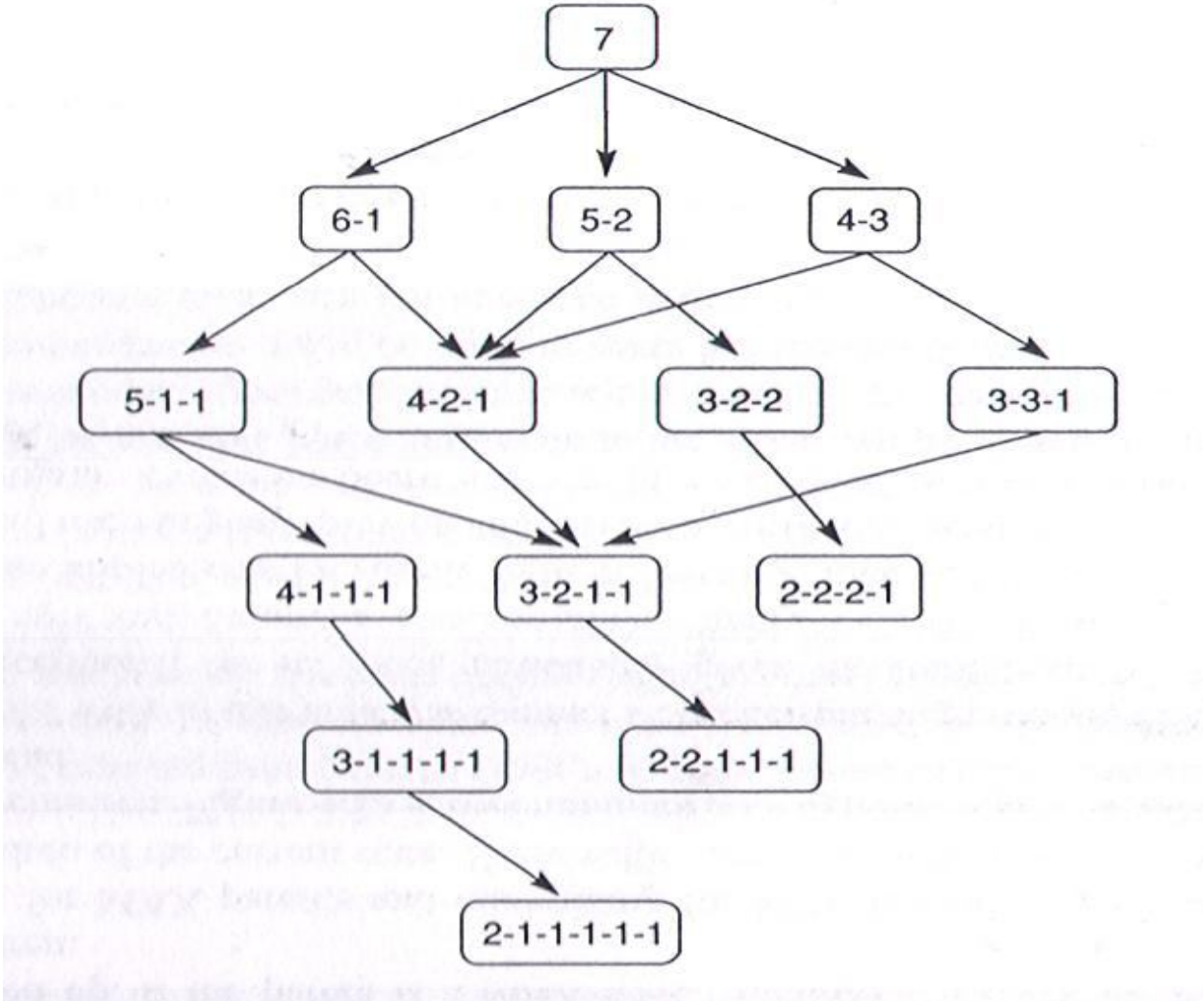
	Tiles out of place	Sum of distances out of place	2 x the number of direct tile reversals
2 8 3 1 6 4 ■ 7 5	5	6	0
2 8 3 1 ■ 4 7 6 5	3	4	0
2 8 3 1 6 4 7 5 ■	5	6	0

Ba heuristic áp dụng vào 3 trạng thái của trò chơi ô đố 8 số

Heuristic trong trò chơi đối kháng

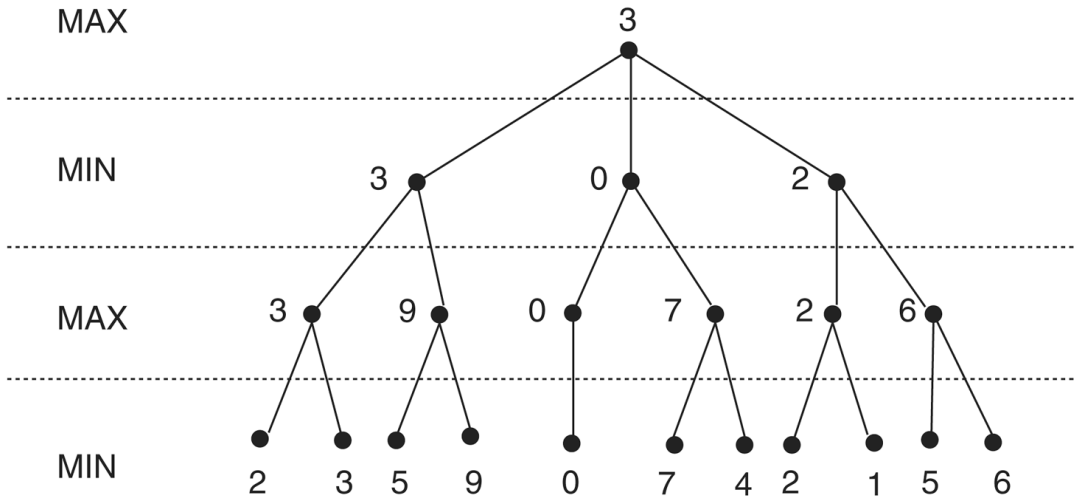
- **Giải thuật minimax**:
  - Hai đấu thủ trong trò chơi được gọi là **MIN** và **MAX**.
  - Mỗi nút lá có giá trị:
    - **1** nếu là MAX thắng,
    - **0** nếu là MIN thắng.
  - Minimax sẽ truyền các giá trị này lên cao dần trên đồ thị, qua các nút cha mẹ kế tiếp theo các luật sau:
    - Nếu trạng thái cha mẹ là **MAX**, gán cho nó giá trị **lớn nhất** có trong các trạng thái con.
    - Nếu trạng thái bố, mẹ là **MIN**, gán cho nó giá trị **nhỏ nhất** có trong các trạng thái con.

Hãy áp dụng GT Minimax vào Trò Chơi NIM



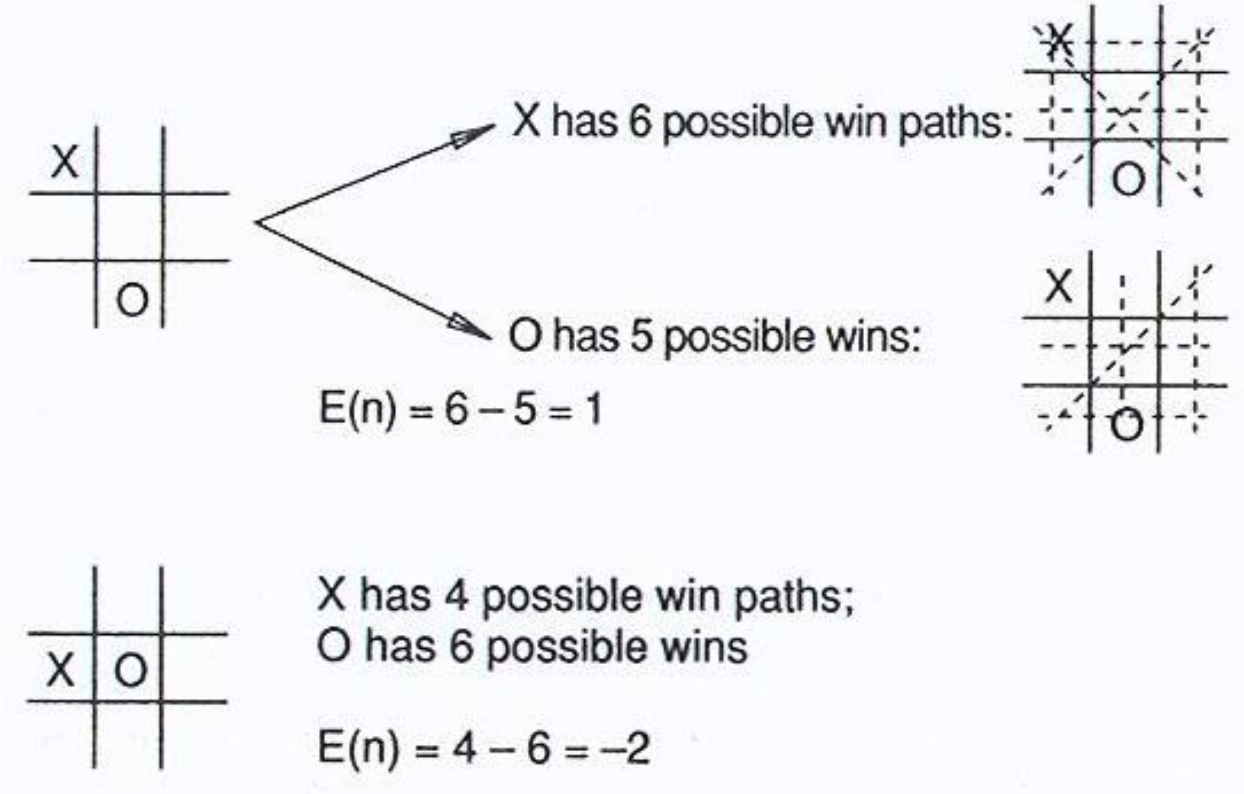
Minimax với độ sâu lớp cố định

- Minimax đối với một KGTT giả định.



- Các nút lá được gán các giá trị ***heuristic***
- Còn giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax

Heuristic trong trò chơi tic-tac-toe



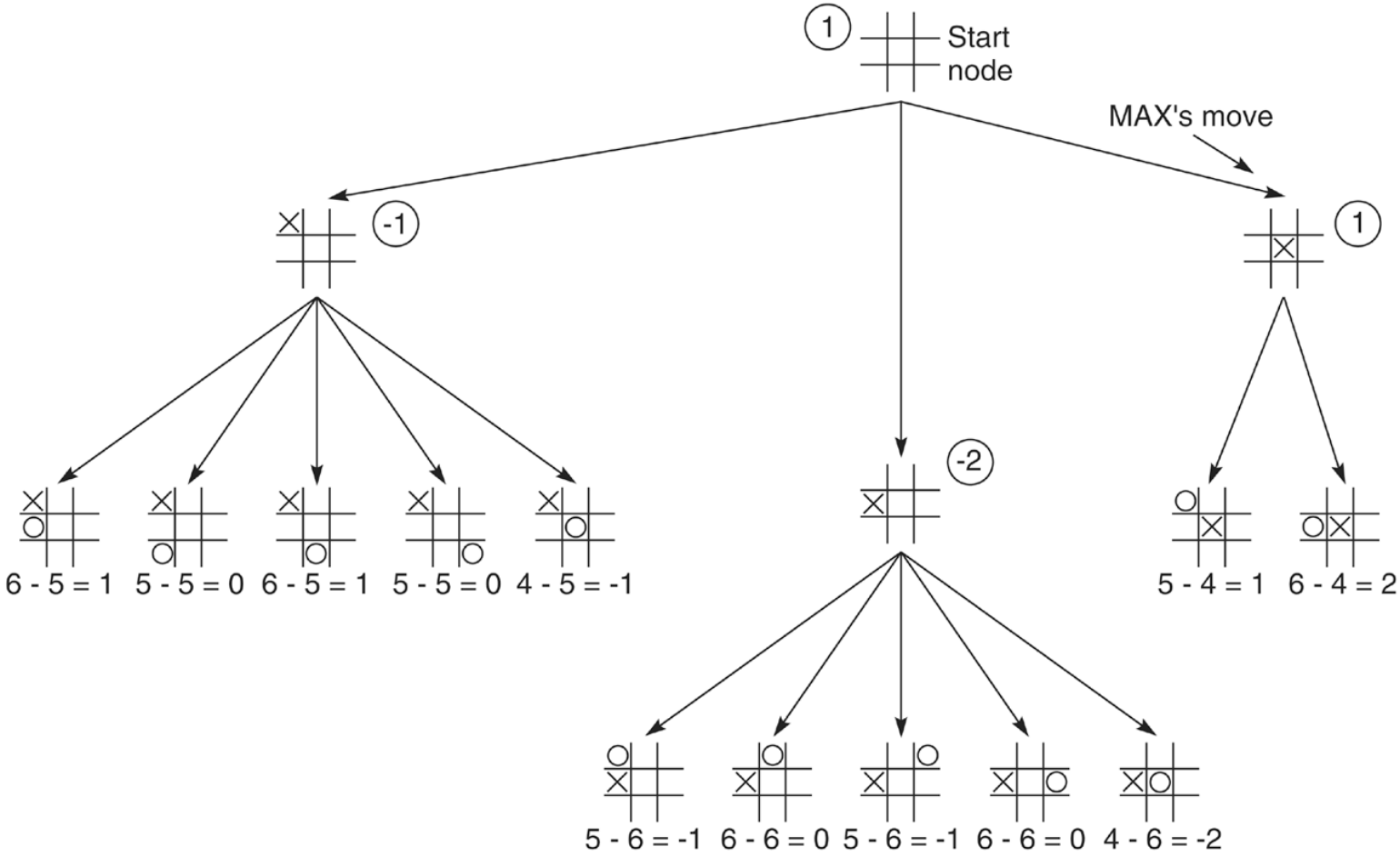
Hàm Heuristic: **E(n) = M(n) – O(n)**

Trong đó: M(n) là *tổng số đường thắng có thể của tôi*

O(n) là *tổng số đường thắng có thể của đối thủ*

E(n) là *trị số đánh giá tổng cộng cho trạng thái n*

Minimax 2 lớp được áp dụng vào nước đi mở đầu trong tic-tac-toe

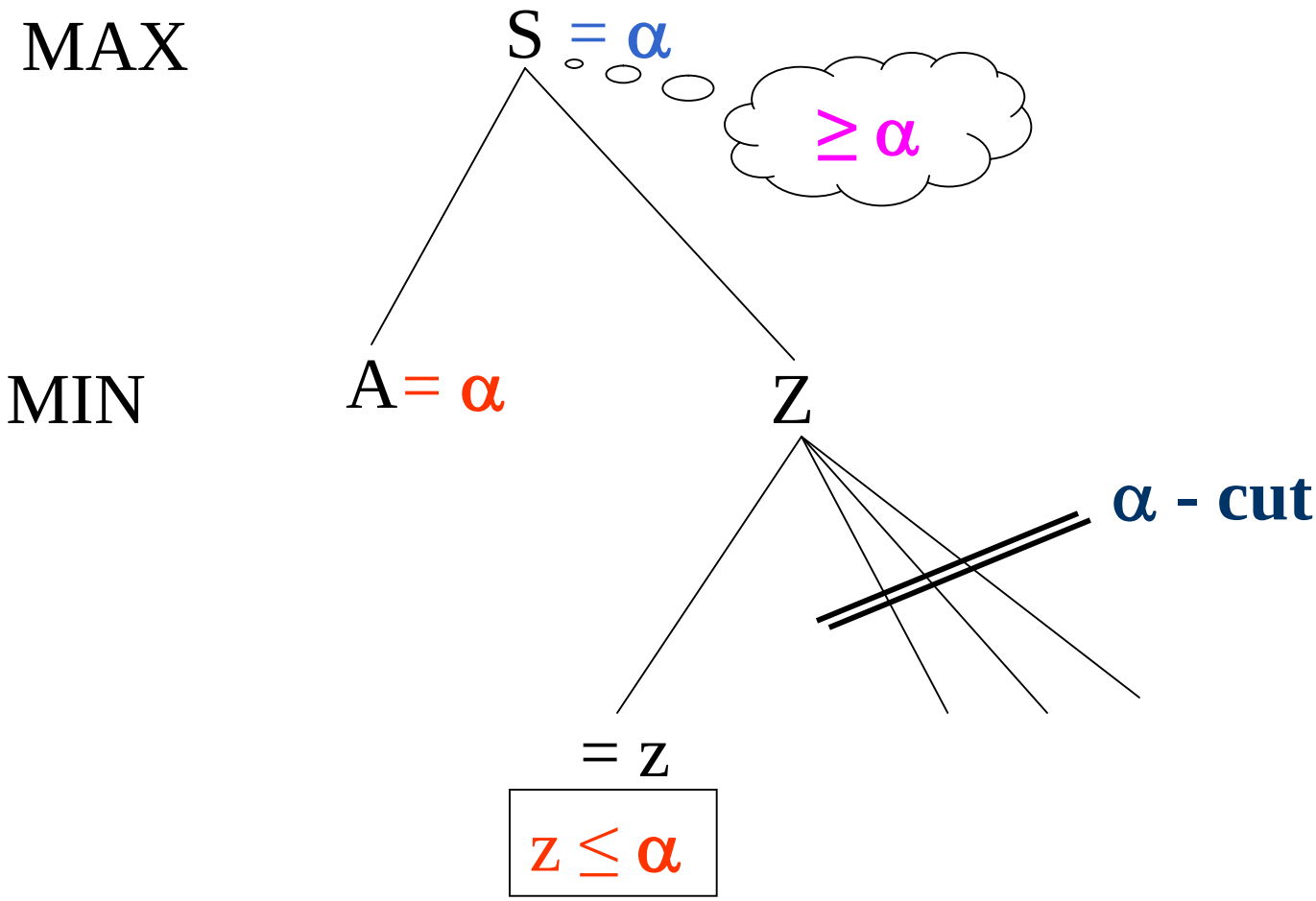


Trích từ Nilsson (1971).

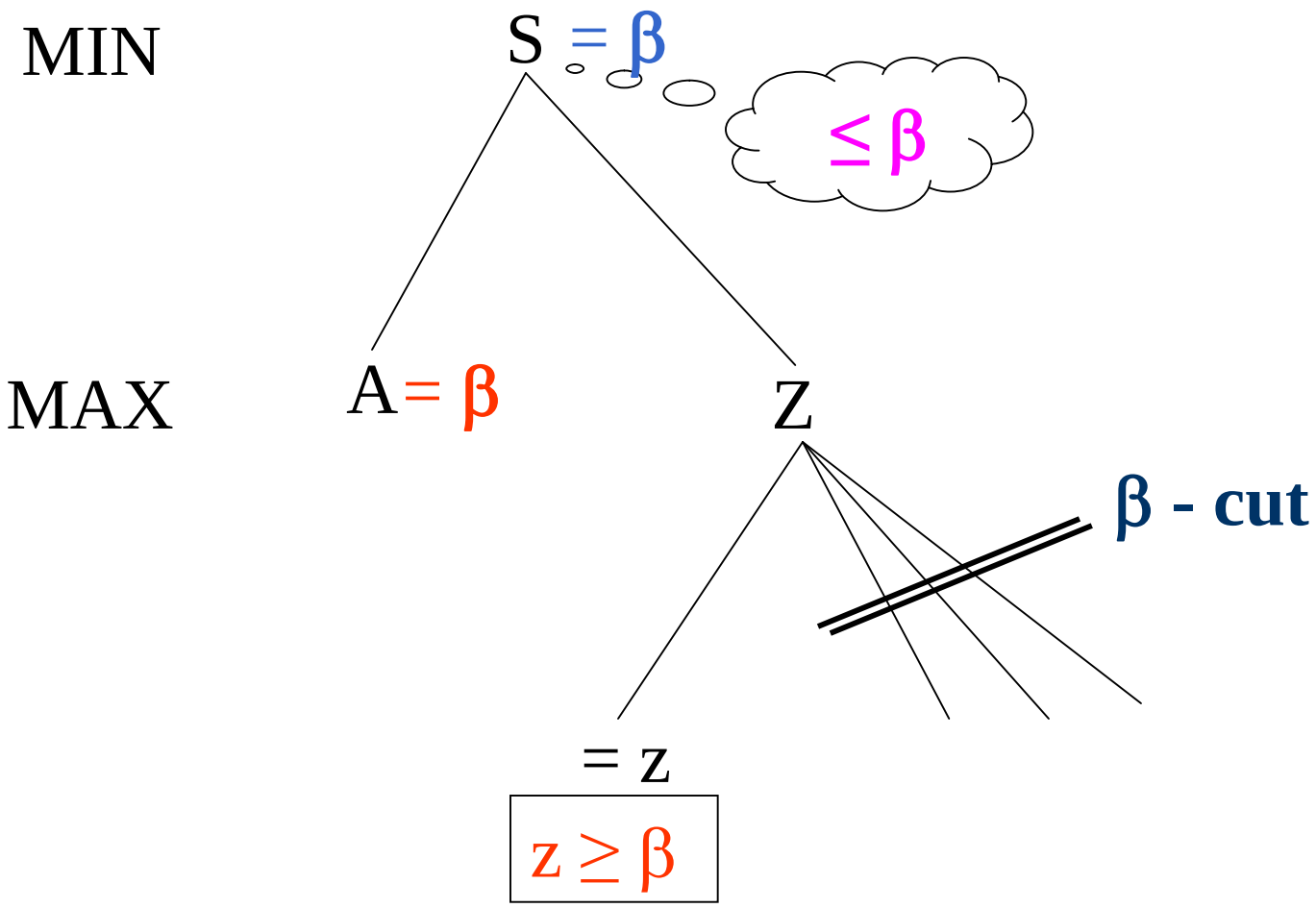
Giải thuật cắt tỉa α-β

- Tìm kiếm theo kiểu depth-first.
- Nút MAX có 1 giá trị **α** (luôn tăng)
- Nút MIN có 1 giá trị **β** (luôn giảm)
- **TK có thể kết thúc dưới bất kỳ**:
  - Nút MIN nào có $\beta \leq \alpha$ của bất kỳ nút cha MAX nào.
  - Nút MAX nào có $\alpha \geq \beta$ của bất kỳ nút cha MIN nào.
- Giải thuật cắt tỉa α-β thể hiện *mối quan hệ giữa các nút ở lớp n và n+2*, mà tại đó toàn bộ cây có gốc tại lớp n+1 có thể cắt bỏ.

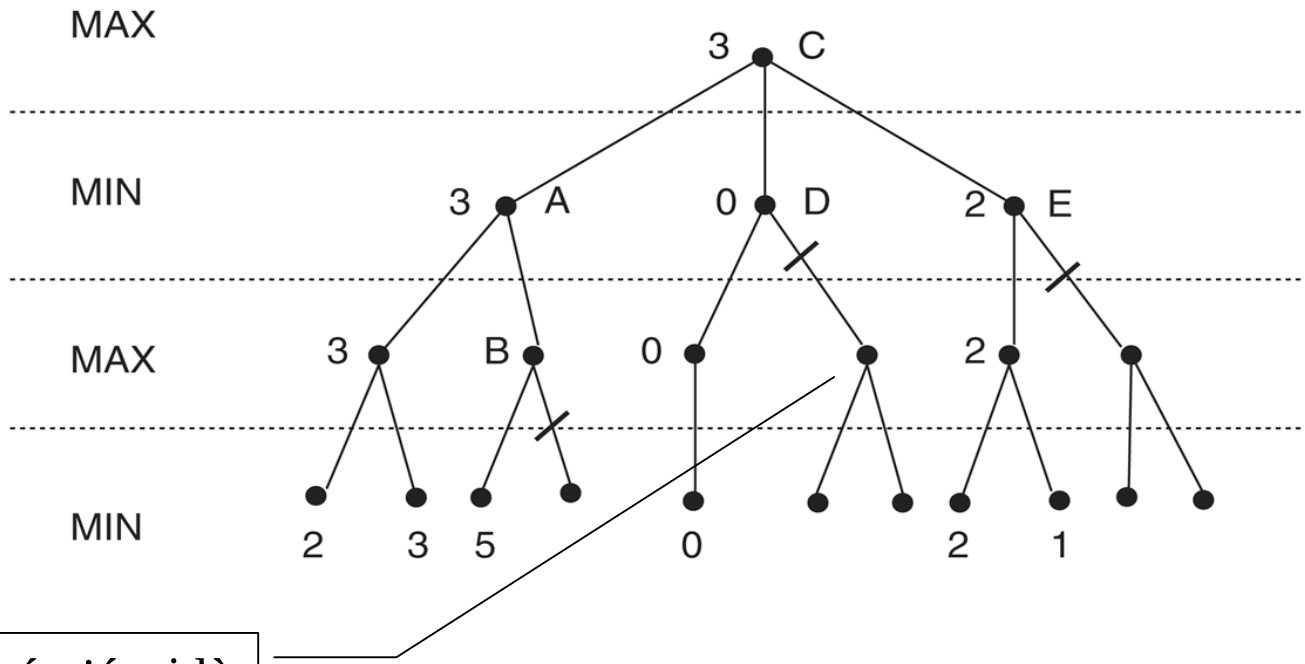
Cắt tỉa α



Cắt tỉa β



GT Cắt Tỉa α-β áp dụng cho KGTT giả định



Các nút không có giá trị là các nút không được duyệt qua

A has $\beta = 3$ (A will be no larger than 3)
B is β pruned, since 5 > 3
C has $\alpha = 3$ (C will be no smaller than 3)
D is α pruned, since 0 < 3
E is α pruned, since 2 < 3
C is 3

Bài Tập Chương 4

Chương 5 – Điều Khiển & Cài Đặt cho TK–KGTT

Giáo viên: Trần Ngân Bình

Nội Dung

- Giải thuật **tìm kiếm đệ qui (Recursive-based search)**
  - Có thể cài đặt tìm kiếm sâu với quay lui một cách đệ qui.
  - Kết hợp phép đồng nhất để tạo ra giải thuật TK hướng mẫu.
  - Là cơ sở của ngôn ngữ PROLOG.
- Giải thuật **tìm kiếm hướng mẫu (Pattern search)**
  - Cài đặt tìm kiếm trên đồ thị Và/Hoặc
  - Tách biệt tri thức giải quyết vấn đề khỏi việc điều khiển tìm kiếm.
- **Hệ thống luật sinh (Production system)**
  - Tìm kiếm được điều khiển theo kiểu hướng mẫu
  - Mô phỏng quá trình giải quyết vấn đề của con người
  - Tách biệt tri thức và điều khiển
  - Tách biệt tri thức giải quyết vấn đề khỏi các dữ kiện bài toán cụ thể trong bộ nhớ làm việc
- **Kiến trúc bảng đen (Blackboard architecture)**

Giải thuật Đệ Qui cho TK Sâu

```
function depthsearch (current_state);                    % closed is global

begin
   if current_state is a goal                            % Teminating condition
      then return SUCCESS;
   add current_state to closed;
   while current_state has unexamined children
      begin
         child := next unexamined child;
         if child not member of closed                   % Loop detection
            then if depthsearch(child) = SUCCESS         % Recusive call
               then return SUCCESS
      end;
   return FAIL                                           % search exhausted
end
```

Tìm Kiếm Hướng Mẫu

Function ***pattern_search***(*current_goal*);
Begin
if *current_goal* ∈ *closed* **then**
return fail
else add current_goal to closed;
while còn dữ kiện hoặc luật đồng nhất **do**
begin case
- current_goal đồng nhất với dữ kiện:
return tập phép thế;
- current goal là ¬p:
pattern_search(p);
if pattern_search thất bại
then return {}
else return fail

- current_goal đồng nhất với kết luận của luật (p ← q):
begin
áp dụng phép thế đồng nhất mục tiêu vào tiền đề (p);
pattern_search (p);
if pattern_search thành công
return hợp của tập phép thế của p và q;
else return fail;
end;

.

Tìm Kiếm Hướng Mẫu

- current_goal có dạng ($p_1 \wedge \ldots$):

begin
for mỗi thành phần p_i **do**
begin
pattern_search(p_i);
if pattern_search thất bại **then**
return fail;
else áp dụng các phép thế vào các p_i còn lại;
end;
if pattern_search thành công cho tất cả các p_i **then**
return hợp của các tập phép thế;
else return fail;
end;

- current_goal có dạng ($p_1 \vee \ldots$):

begin
repeat cho mỗi p_i
pattern_search(pi);
until không còn thành phần p_i nào hoặc thành công;
if pattern_search thành công
then return {phép thế};
else return fail;
end;
return fail;
End.

Một số vấn đề về biểu diễn luật

- $P \Rightarrow Q$ $Q \Leftarrow P$
- If giả thuyết then kết luận Q If P
- If điều kiện then hành động Q :- P
- If tiền đề then hệ quả Q when P

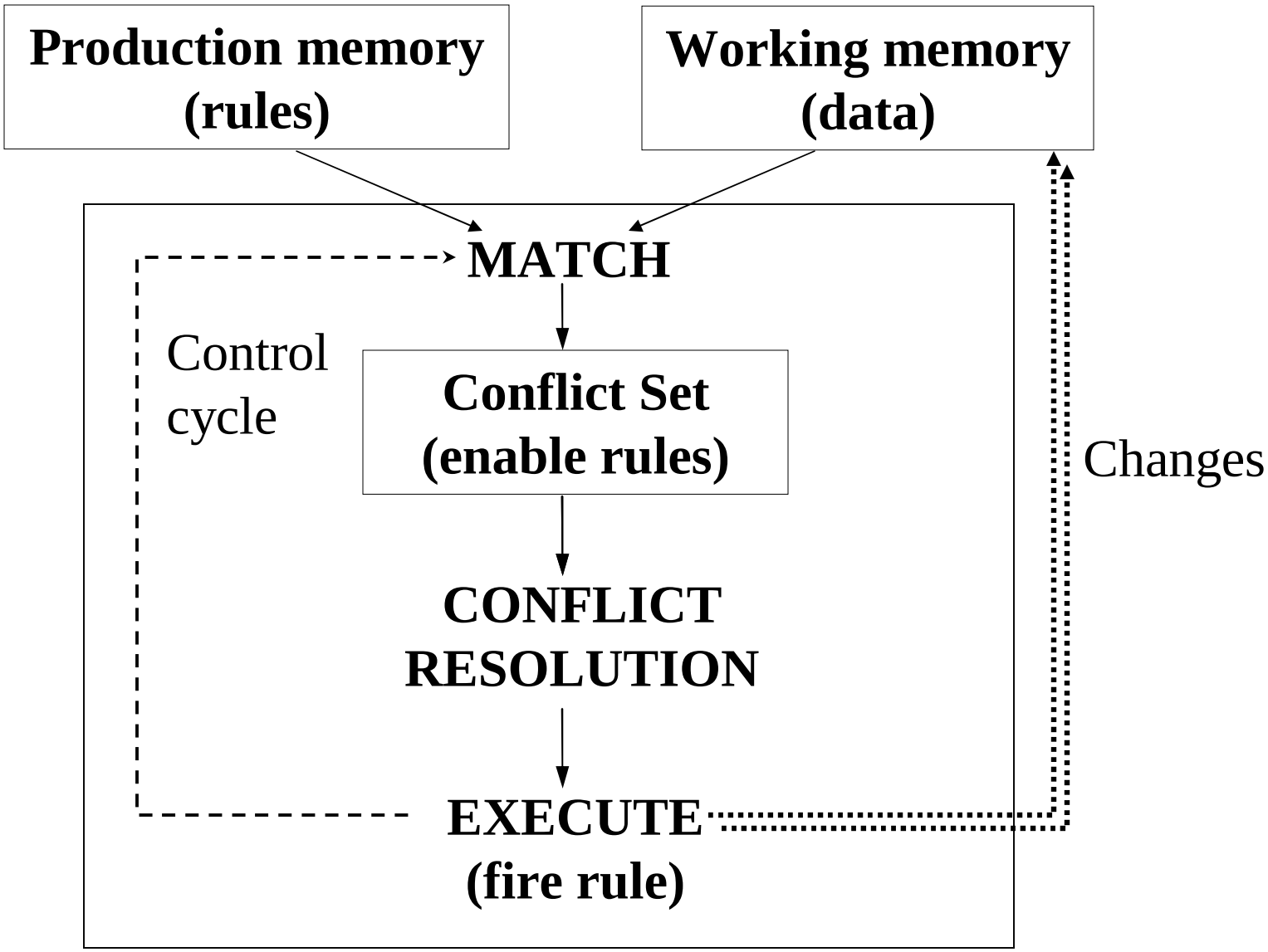
- Đôi khi có một số ràng buộc như:
  - Không cho phép: $p \vee q \Rightarrow r$
  - Không cho phép: $p \Rightarrow \neg q$
  - Không cho phép: $p \Rightarrow r \vee q$
- Các lượng tử biến được xóa bỏ khi:
  - Các biến xuất hiện ở cả hai vế của luật kết hợp với lượng tử phổ biến.
  - Các biến xuất hiện trong tiền đề của luật chỉ kết hợp với lượng tử tồn tại.

Định Nghĩa Hệ Thống Luật Sinh

- Một hệ thống luật sinh được định nghĩa bởi:
  - **Tập hợp các luật sinh (production rules):**
    mỗi luật sinh có dạng: condition → action
  - **Bộ nhớ làm việc (Working memory):** chứa các mô tả về trạng thái hiện hành của 'thế giới' trong quá trình suy luận
  - **Chu trình nhận dạng – hành động (recognize-act cycle):** là cấu trúc điều khiển của hệ sinh.
- Một số khái niệm:

- **Luật khả thi (enable rule):** là luật có các điều kiện đối sánh với các mẫu trong bộ nhớ làm việc.
- **Tập tranh chấp (conflict set):** là tập hợp tất cả các luật khả thi.
- **Giải quyết tranh chấp (conflict resolution):** chọn một luật trong tập tranh chấp để thi hành.

Kiến Trúc của Hệ Sinh



Một Hệ Thống Luật Sinh Đơn Giản

- **Cơ chế điều khiển:** lặp lại cho đến khi mẫu trong bộ nhớ làm việc không còn khớp với điều khiển của bất kỳ luật sinh nào.

Production set:

1. ba → ab
2. ca → ac
3. cb → bc

Iteration #	Working memory	Conflict set	Rule fired
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	∅	Halt

Figure 5.4: **Trace of a simple production system.**

Trò chơi ô đố 8-puzzle (1)

Start state:

2	8	3
1	6	4
7		5

Goal state:

1	2	3
8		4
7	6	5

Production set:

Condition		Action
goal state in working memory	→	halt
blank is not on the left edge	→	move the blank left
blank is not on the top edge	→	move the blank up
blank is not on the right edge	→	move the blank right
blank is not on the bottomedge	→	move the blank down

Working memory is the present board state and goal state.

Control regime:

1. Try each production in order.
2. Do not allow loops.
3. Stop when goal is found.

Figure 5.5: **The 8-puzzle as a production system.**

Trò chơi ô đố 8-puzzle (2)

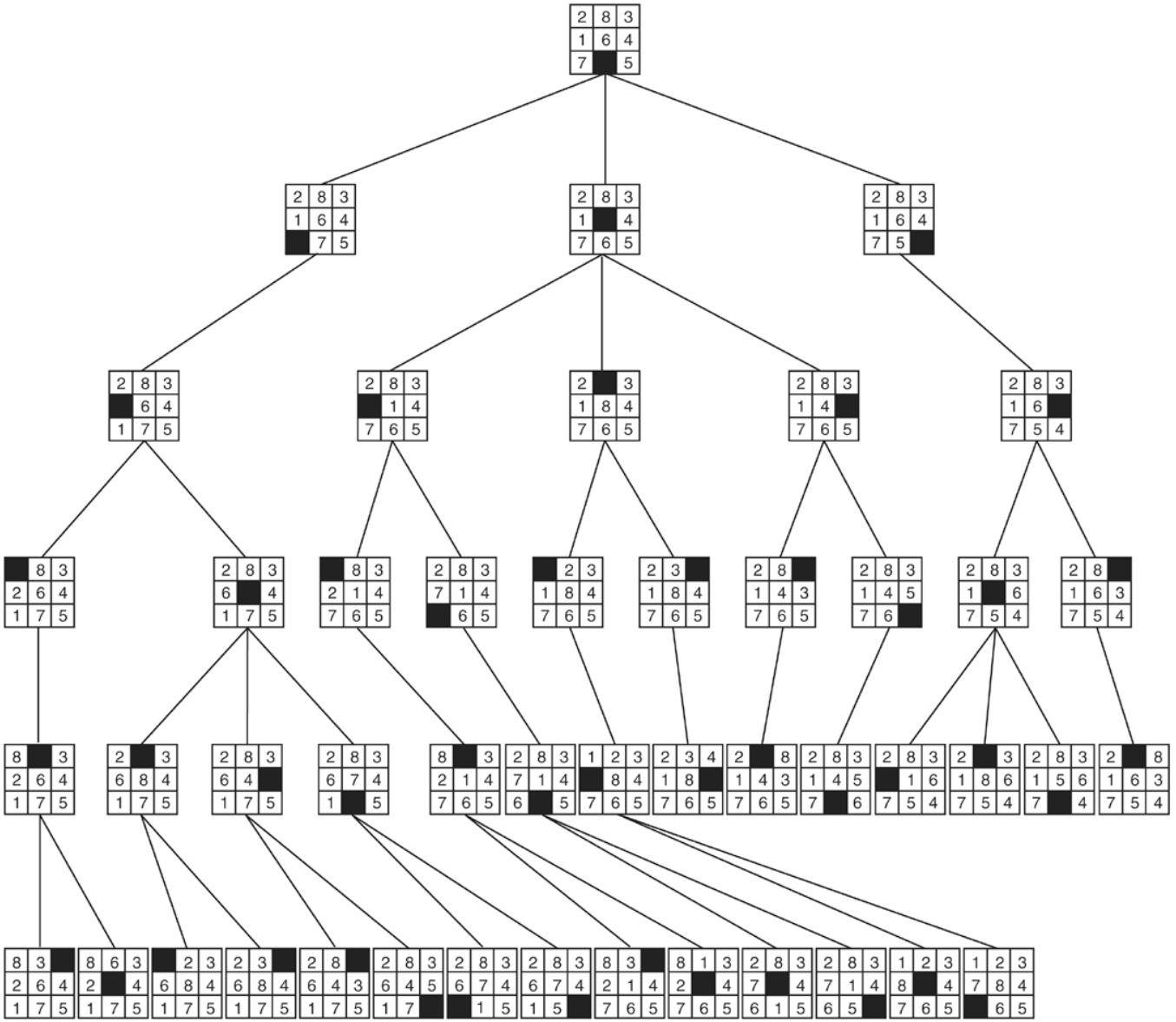
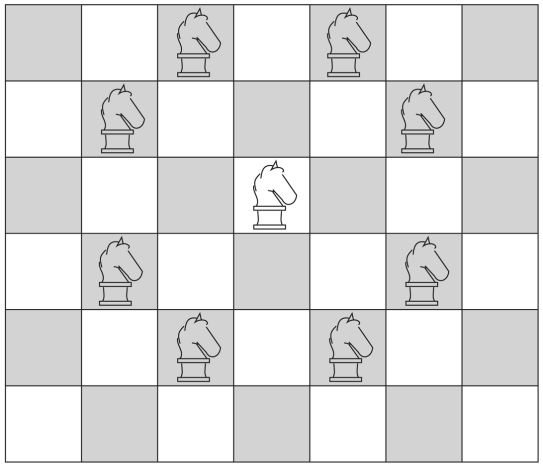


Figure 5.6: The 8-puzzle searched by a production system with loop detection and depth bound 5

Trò chơi đường đi quân mã (1)



Các bước đi hợp lệ
của quân mã

1	2	3
4	5	6
7	8	9

Đánh số các ô trong
bàn cờ 3x3

RULE#	CONDITION	ACTION
1	Quân mã ở ô 1	Di chuyển đến ô 8
2	Quân mã ở ô 1	Di chuyển đến ô 6
3	Quân mã ở ô 2	Di chuyển đến ô 9
4	Quân mã ở ô 2	Di chuyển đến ô 7
5	Quân mã ở ô 3	Di chuyển đến ô 4
6	Quân mã ở ô 3	Di chuyển đến ô 8
7	Quân mã ở ô 4	Di chuyển đến ô 9
8	Quân mã ở ô 4	Di chuyển đến ô 3
9	Quân mã ở ô 6	Di chuyển đến ô 1
10	Quân mã ở ô 6	Di chuyển đến ô 7
11	Quân mã ở ô 7	Di chuyển đến ô 2
12	Quân mã ở ô 7	Di chuyển đến ô 6
13	Quân mã ở ô 8	Di chuyển đến ô 3
14	Quân mã ở ô 8	Di chuyển đến ô 1
15	Quân mã ở ô 9	Di chuyển đến ô 2
16	Quân mã ở ô 9	Di chuyển đến ô 4

Trò chơi đường đi quân mã (2)

- **Cơ chế điều khiển:** Áp dụng luật đầu tiên tìm được mà không tạo vòng lặp (không đi lại ô đã đi qua).Cho phép quay lui

VD: Tìm 1 con đường để quân mã đi từ vị trí 1 đến vị trí 2

Iteration #	Working memory		Conflict set (rule #'s)	Fire rule
	Current square	Goal square		
0	1	2	1, 2	1
1	8	2	13, 14	13
2	3	2	5, 6	5
3	4	2	7, 8	7
4	9	2	15, 16	15
5	2	2		Halt

Figure 5.7: **A production system solution to the 3 × 3 knight's tour problem.**

Điều Khiển TK Trong Hệ Sinh

- Chọn lựa giữa tiếp cận hướng từ dữ liệu hay hướng từ mục tiêu.
- Điều khiển được mã hóa trong cấu trúc các luật (tạo khả năng sử dụng heuristic)

  if engine does not turn over

  And the lights don't come on

  then check the battery

  Lưu ý: thứ tự của các điều kiện

- Cơ chế điều khiển - là phương pháp chọn lựa luật để thi hành:
  - Khúc xạ (refraction)
  - Mới xảy ra (recency)
  - Tính chi tiết (specificity)
  - Thứ tự các luật được lưu trữ.

Tìm Kiếm Hướng Từ Dữ Liệu

Production set:

1. $p \wedge q \rightarrow goal$
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow q$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. $start \rightarrow v \wedge r \wedge q$

Trace of execution:

Iteration #	Working memory	Conflict set	Rule fired
0	start	6	6
1	start, v, r, q	6, 5	5
2	start, v, r, q, s	6, 5, 2	2
3	start, v, r, q, s, p	6, 5, 2, 1	1
4	start, v, r, q, s, p, goal	6, 5, 2, 1	halt

Space searched by execution:

Figure 5.9: **Data-driven search in a production system.**

Tìm Kiếm Hướng Từ Mục Tiêu

Production set:

1. p ∧ q → goal
2. r ∧ s → p
3. w ∧ r → p
4. t ∧ u → q
5. v → s
6. start → v ∧ r ∧ q

Trace of execution:

Iteration #	Working memory	Conflict set	Rule fired
0	goal	1	1
1	goal, p, q	1, 2, 3, 4	2
2	goal, p, q, r, s	1, 2, 3, 4, 5	3
3	goal, p, q, r, s, w	1, 2, 3, 4, 5	4
4	goal, p, q, r, s, w, t, u	1, 2, 3, 4, 5	5
5	goal, p, q, r, s, w, t, u, v	1, 2, 3, 4, 5, 6	6
6	goal, p, q, r, s, w, t, u, v, start	1, 2, 3, 4, 5, 6	halt

Space searched by execution:

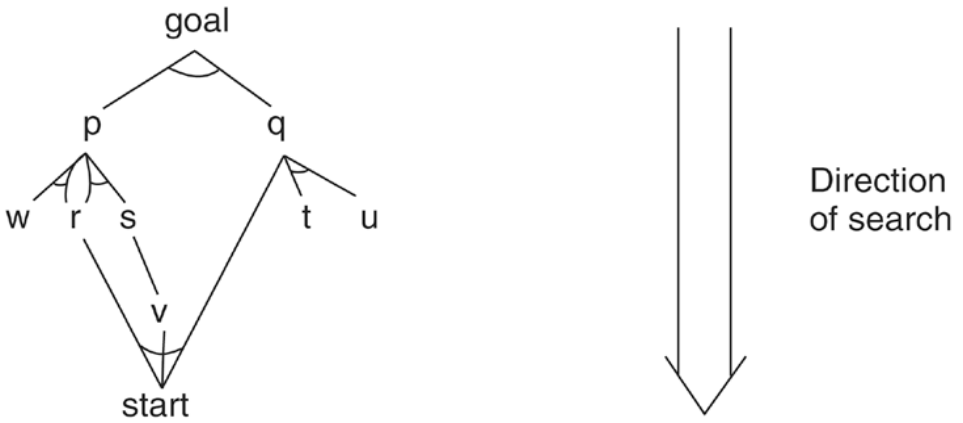


Figure 5.10: **Goal-driven search in a production system.**

Cơ Chế Điều khiển

- **Khúc xạ (refraction):** một khi luật được chọn, nó sẽ không được chọn lại cho đến khi các điều kiện của nó trong bộ nhớ làm việc được thay đổi.
- **Mới xảy ra (recency):** chọn những luật khớp với các mẫu được thêm vào gần đây nhất.
- **Tính chi tiết (specificity):** chọn luật có chứa nhiều điều kiện hơn.
- **Thứ tự các luật được lưu trữ:** chọn luật khả thi đầu tiên.

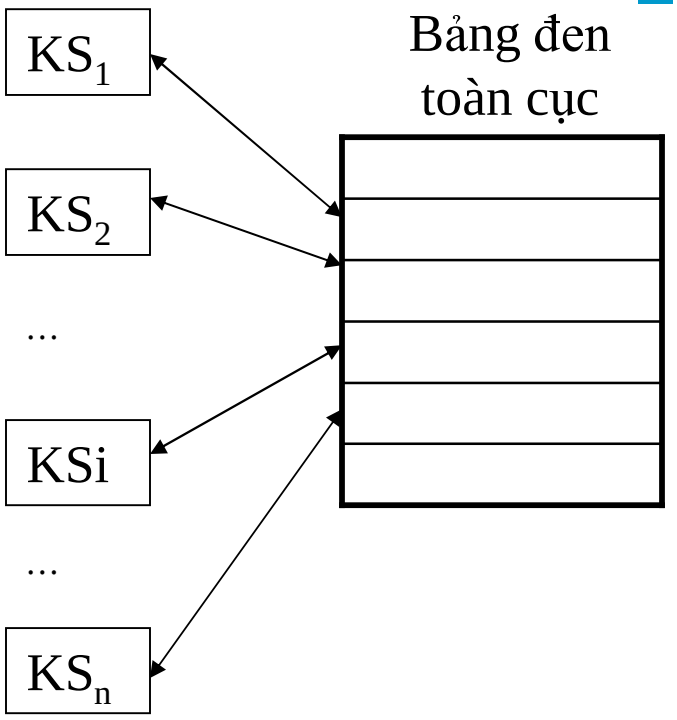
Ưu Điểm của Hệ Thống Luật Sinh

- Tách biệt rõ ràng giữa tri thức và điều khiển
- Cung cấp một Ánh xạ tự nhiên vào TK trên KGTT
- Mô-đun hóa các luật sinh
- Tìm kiếm được điều khiển theo kiểu hướng mẫu
- Tạo cơ hội sử dụng heuristic trong việc điều khiển TK
- Cho phép lần vết và diễn giải
- Độc lập ngôn ngữ
- Là một mô hình hợp lý mô tả việc giải quyết vấn đề của con người.

$\Rightarrow$ Là một công cụ quan trọng để xây dựng các hệ chuyên gia

Kiến Trúc Bảng Đen

- Một bảng đen là một cơ sở các dữ liệu *toàn cục tập trung* cho sự giao tiếp giữa các tài nguyên tri thức *không đồng bộ độc lập nhau* tập trung vào các khía cạnh liên quan của một vấn đề nào đó.



- Các thành phần:
  - **Bảng đen (blackboard):** là một cơ sở dữ liệu tập trung toàn cục
  - Các **tài nguyên tri thức (Knowledge Source):** độc lập không đồng bộ giao tiếp với nhau thông qua bảng đen.
  - **Bộ lập lịch trình (Scheduler):** tổ chức việc cấp tài nguyên và truy cập bảng đen: thể hiện cơ chế điều khiển (focus of attention).

Ví dụ Hearsay II

- Mục đích: Thông dịch các câu nói với một lượng từ vựng và văn phạm giới hạn, ± 1000 words, ±17 từ có thể theo sau bởi một từ khác
- Giải pháp: Tìm kiếm một không gian các phần lời giải được ghi nhận trong một cây phân cấp trừu tượng được thực hiện với một bảng đen.
- Tìm kiếm:
  - tìm kiếm có cơ hội
  - Trên xuống + dưới lên (đưa ra giả thuyết và kiểm tra)
  - Cơ chế điều khiển thông qua lập lịch cho các tài nguyên tri thức. (VD: Ban đầu các tài nguyên mức thấp có độ ưu tiên cao hơn, sau đó đến các tài nguyên ở mức cao có độ ưu tiên cao hơn.)

Ưu Điểm của Kiến Trúc Bảng đen

- Mở rộng của các hệ thống luật sinh: cho phép tổ chức bộ nhớ làm việc vào các module riêng, mỗi module tương ứng với các tập con luật sinh khác nhau.
- Cho phép tổ chức và phối hợp nhiều chương trình giải quyết vấn đề trong một cấu trúc toàn cục duy nhất.
- Thích hợp cho việc thực thi chương trình trong một môi trường tính toán phân tán, đa bộ xử lý.
- Hổ trợ mạnh mẽ cho những ứng dụng tri thức có cấu trúc cao và mang tính cơ hội.

Chương 6:
Hệ chuyên Gia

Giáo viên: Trần Ngân Bình

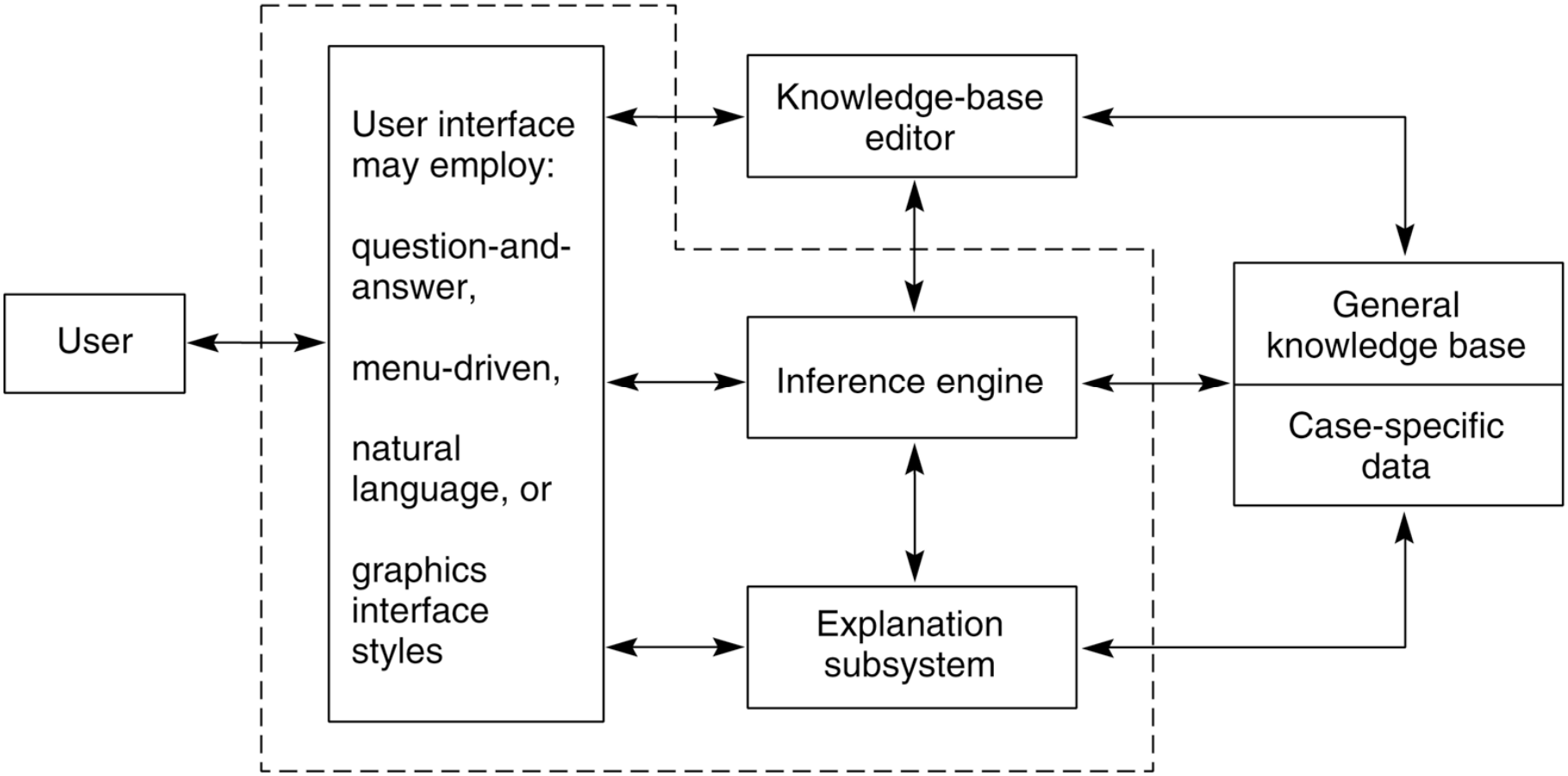
Nội Dung

- Hệ chuyên gia (Expert System – ES)
  - Tổng quát về các hệ chuyên gia
  - Công nghệ tri thức
- Hệ chuyên gia dựa trên luật (rule-based ES): là các hệ thống suy luận dựa trên luật.
- Hệ chuyên gia dựa trên mô hình (model-based ES): là các hệ thống suy luận dựa trên mô hình lý thuyết của tri thức chuyên ngành.
- Hệ chuyên gia dựa trên trường hợp (case-based ES): là các hệ thống suy luận dựa trên các ví dụ đã có.

Hệ chuyên gia (HCG)

- = là một nhánh của TTNT liên quan đến sự phát triển của các hệ thống dựa trên tri thức
- = là một chương trình dựa trên tri thức, cung cấp các giải pháp với "chất lượng chuyên gia" cho các vấn đề trong một lĩnh vực nào đó.
- HCG nói chung:
  - Cung cấp sự theo dõi quá trình suy luận.
  - Cho phép thay đổi cơ sở tri thức một cách dễ dàng.
  - Suy luận một cách heurisic, sử dụng tri thức để đưa ra lời giải

Kiến trúc của một HCG tiêu biểu



Các bài toán phù hợp với giải pháp HCG:

1. Sự cần thiết của một giải pháp biện minh cho chi phí và sức lực của việc xây dựng HCG.
2. Tri thức chuyên môn không sẵn sàng ở những nơi cần đến nó.
3. Vấn đề có thể được giải quyết bằng cách sử dụng các kỹ thuật suy luận ký hiệu
4. Vấn đề được cấu trúc tốt và không đòi hỏi sự suy luận theo lẽ thường.
5. Vấn đề có thể không giải quyết được bằng cách sử dụng các phương pháp tính toán truyền thống.
6. Có cơ sở hợp tác và hiểu ý nhau giữa các chuyên gia.
7. Vấn đề có kích cỡ và quy mô vừa phải.

Qui trình công nghệ tri thức (knowledge Engineering)

- Ba người liên quan:
  - Kỹ sư tri thức (knowledge engineer): là các chuyên gia về ngôn ngữ và biểu diễn trong TTNT.
  - Chuyên gia (domain expert): là những người làm việc trong lĩnh vực chuyên môn và hiểu các phương pháp giải quyết vấn đề trong lĩnh vực đó.
  - Người sử dụng (end user): là những người xác định các ràng buộc thiết kế chủ yếu.
- Quá trình xây dựng HCG đòi hỏi một chu trình phát triển theo kiểu không truyền thống dựa trên các bản mẫu ban đầu và sửa lại chương trình với mức độ tăng dần => phương pháp lập trình thăm dò

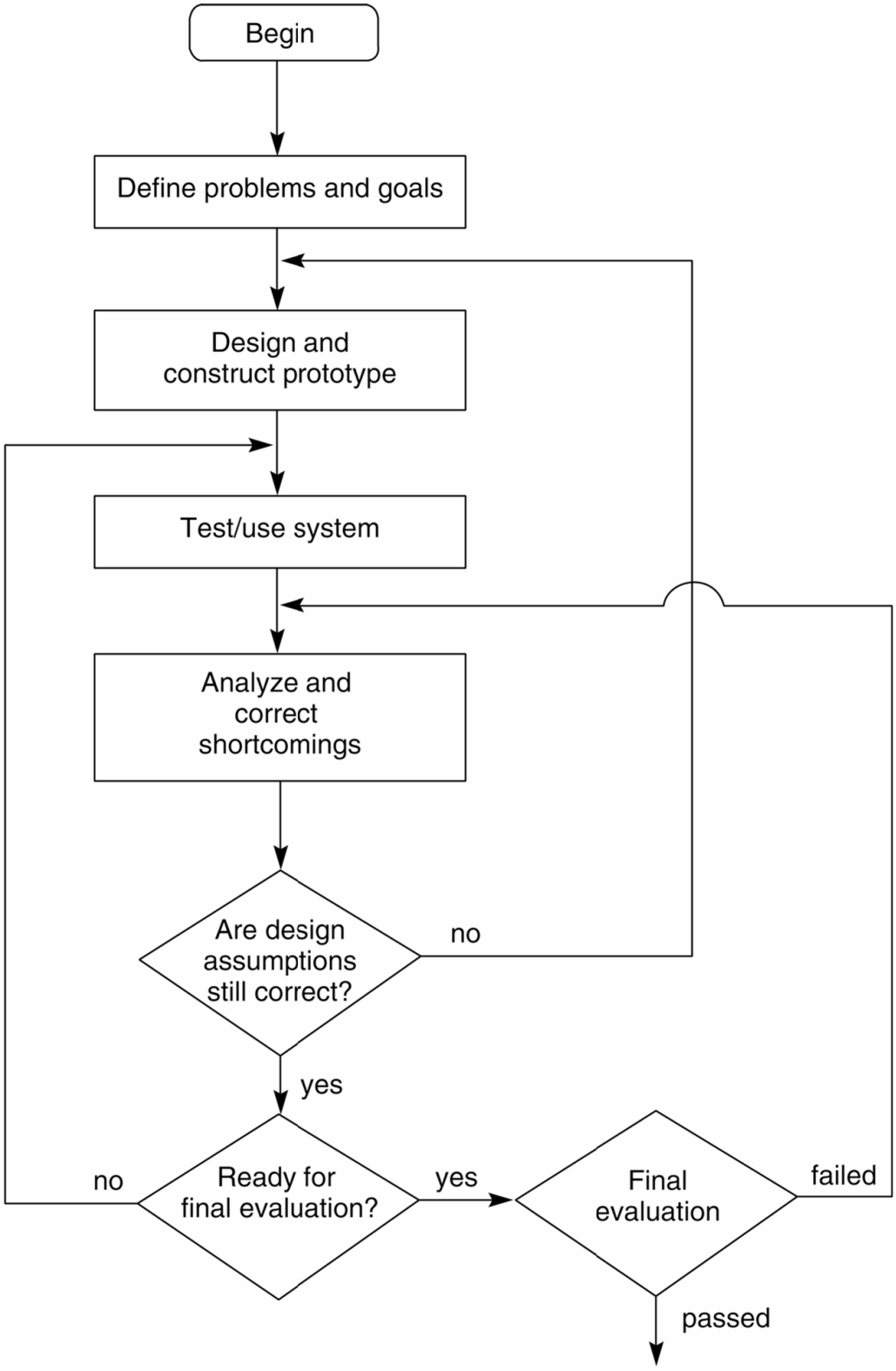
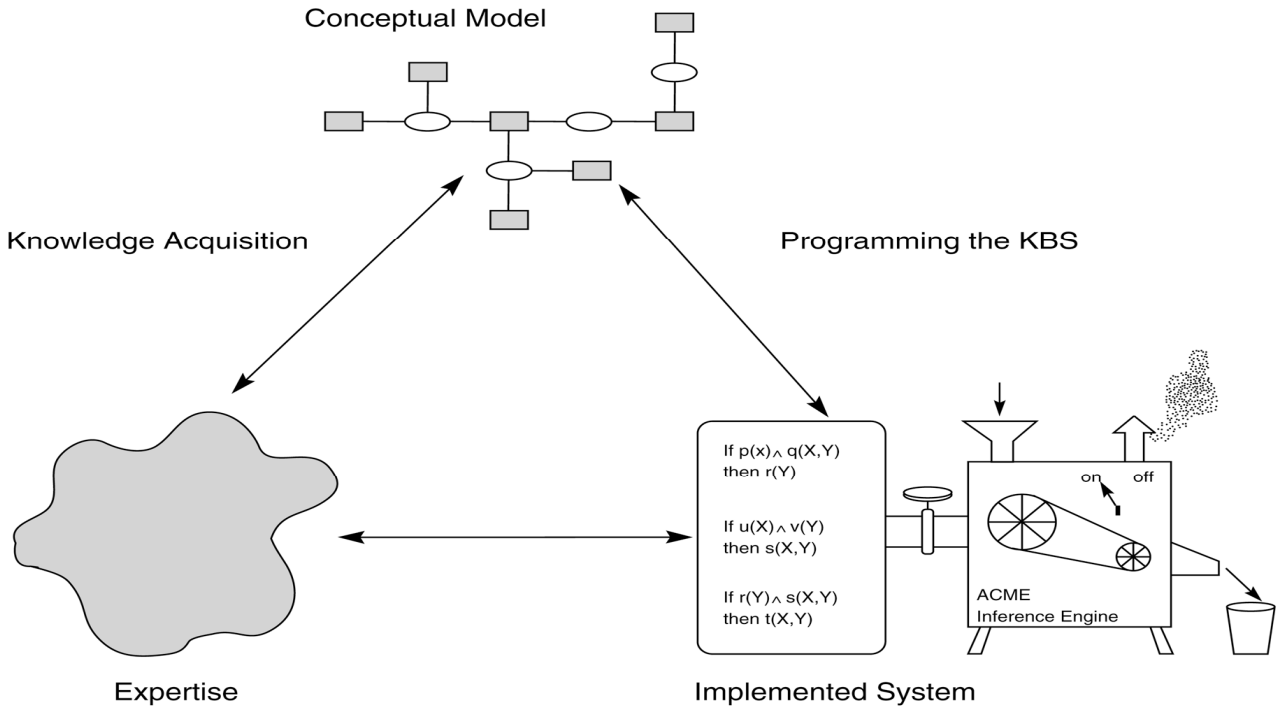


Figure 6.2 – Chu trình phát triển theo kiểu thăm dò

Mô hình khái niệm & việc tích lũy tri thức

- Các khó khăn trong việc tích lũy tri thức:
  - Các kỹ năng của con người thường dựa trên thực nghiệm.
  - Tri thức của con người là "biết làm thế nào"
  - Tri thức của con người không căn cứ theo sự thật.
  - Tri thức luôn luôn thay đổi.



Ví dụ một HCG dựa trên luật

Luật 1 IF động cơ nhận được xăng
AND động cơ khởi động được
THEN trục trặc là do bugi.

Luật 2 IF động cơ không khởi động được
AND đèn không sáng
THEN trục trặc là do ắcquy hoặc dây cáp

Luật 3 IF động cơ không khởi động được
AND đèn sáng
THEN trục trặc là do môtơ khởi động

Luật 4 IF còn xăng trong bình chứa nhiên liệu
AND còn xăng trong bộ chế hòa khí
THEN động cơ nhận được xăng

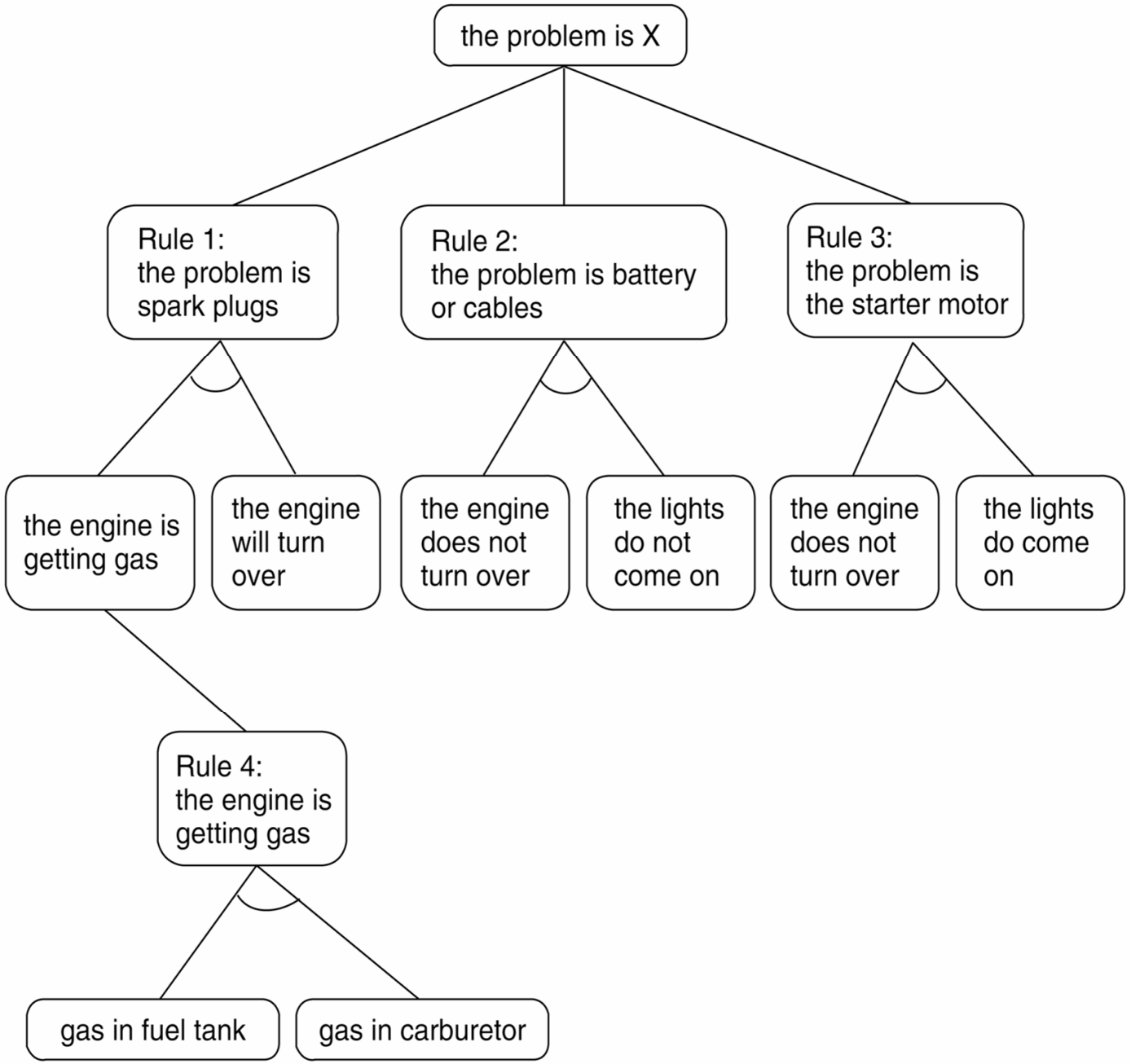


Figure 6.8 - Đồ thị Và/Hoặc tìm được trong ví dụ chẩn đoán trục trặc xe hơi

Khả năng giải thích và tính trong sáng

còn xăng trong bình chứa nhiên liệu? yes
còn xăng trong bộ chế hòa khí? yes
động cơ khởi động được không? why
Nó đã suy luận được:
1. động cơ nhận được xăng
Vì vậy nếu
2. động cơ khởi động được
Thì có thể kết luận: trục trặc là do bugi

động cơ nhận được xăng như thế nào?

Điều này có từ luật 4

IF còn xăng trong bình chứa nhiên liệu
AND còn xăng trong bộ chế hòa khí
THEN động cơ nhận được xăng

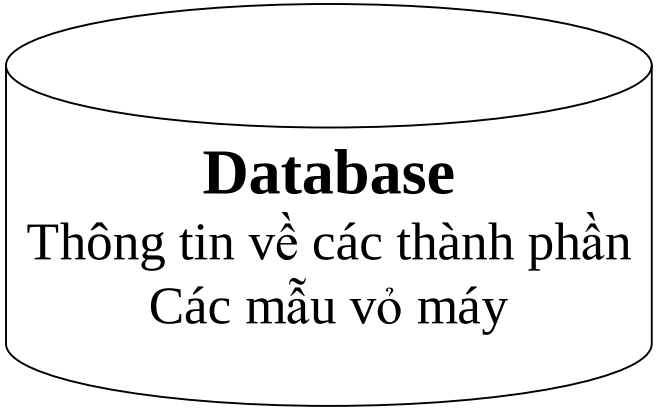
Người dùng cho biết: còn xăng trong bình chứa nhiên liệu
Người dùng cho biết: còn xăng trong bộ chế hòa khí

Hệ chuyên gia R1/XCON

- Mục đích: tạo cấu hình hệ thống VAX-11/780 của công ty DEC
- Đầu vào: Đơn đặt hàng = danh sách các thành phần cấu hình nên hệ thống.
- Đầu ra: Sơ đồ cấu hình
- Kết quả: cấu hình 97% các đơn đặt hàng của DEC

Công việc tạo cấu hình của R1/XCON có thể được xem như là một hệ thống phân cấp các công việc nhỏ hơn với *sự phụ thuộc thời gian (temporal dependency)* rất mạnh.

Kiến trúc của XCON



Cơ sở các luật
Các luật 'điều hành'
Các luật chuyển đổi theo tình huống

Bộ nhớ làm việc
Các ký hiệu thành phần
Các cấu hình chưa hoàn chỉnh
Các ký hiệu tình huống

OPS5
Động cơ suy diễn

Đặc biệt: chọn luật có điều kiện trùng khớp với yếu tố mới nhất trong bộ nhớ làm việc

Giới hạn của HCG dựa trên luật

- Các luật đạt được từ các chuyên gia mang tính heuristic rất cao (e.g. kết hợp trực tiếp các triệu chứng quan sát được và các chẩn đoán) , mà thiếu một sự hiểu biết lý thuyết sâu hơn về lĩnh vực chuyên ngành và quá trình giải quyết vấn đề.
- Các luật heuristic "dễ vỡ", không thể xử lý các trường hợp ngoài dự kiến.
- Có khả năng giải thích chứ không chứng minh.
- Các tri thức thường rất phụ thuộc vào công việc
- Khó bảo trì các cơ sở luật lớn.

Tri thức bề nổi

Các luật heuristic

Tri thức sâu

Lý thuyết chuyên ngành

+

Tri thức giải quyết vấn đề

HCG dựa trên mô hình

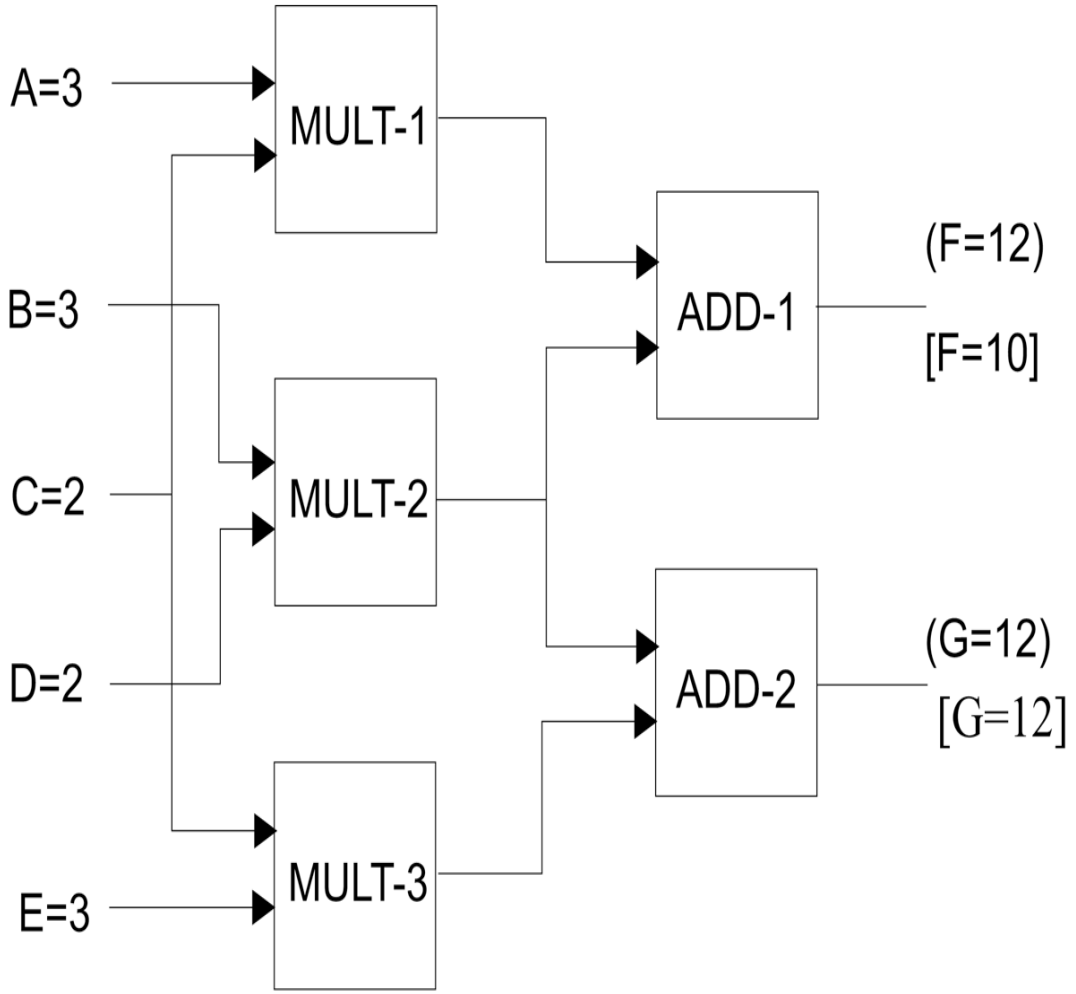
- = là một hệ thống mà sự phân tích căn cứ trực tiếp trên sự mô tả chi tiết và chức năng của một hệ thống vật lý.
- Ứng dụng: trong mục đích giảng dạy (mô hình của các thiết bị vật lý như mạch điện), các hệ thống tìm lỗi,…
- Một hệ thống chẩn đoán dựa trên mô hình đòi hỏi:
  - Mô tả từng thành phần của một thiết bị => khả năng mô phỏng chức năng của từng thành phần
  - Mô tả cấu trúc bên trong của một thiết bị, thường là các thành phần và sự liên kết bên trong của chúng => khả năng mô phỏng sự tương tác giữa các thành phần.
  - Sự quan sát của việc thực hiện thật sự của thiết bị, ví dụ do các thông số vào/ra

  => Sự xác định lỗi thông qua việc giải thích sự khác biệt giữa các hành vi thật sự và hành vi mong đợi của thiết bị

Ví dụ: định vị nơi gây lỗi

- Thực hiện 3 bước:
  - Tạo ra giả thuyết
  - Kiểm tra giả thuyết
  - Loại dần giả thuyết

Giới hạn: Chương trình hoạt động trên giả thuyết là hệ thống vật lý này chỉ có một lỗi



Ưu điểm của HCG dựa trên mô hình

- Tạo khả năng sử dụng hiểu biết về cấu trúc và chức năng của vấn đề để giải quyết vấn đề.
- Vượt qua hạn chế của HCG dựa trên luật, HCG này có khuynh hướng mạnh, "khó vỡ".
- Một số tri thức có thể chuyển tải cho các công việc khác.
- Có khả năng cung cấp các lời giải thích chỉ rõ nguyên nhân gây lỗi.

Khuyết điểm của HCG dựa trên mô hình

- Mô hình chỉ là một mô hình nghĩa là một sự trừu tượng của hệ thống, vì vậy ở một mức độ chi tiết nào đó có thể không đúng (vd: tình trạng của đầu vào dữ liệu).
- Có một sự giả thiết ngầm hiểu về thế giới đóng =>những gì không nằm trong mô hình có nghĩa là không tồn tại.
- Đòi hỏi một mô hình lý thuyết rõ ràng => việc tích lũy tri thức có thể gặp nhiều khó khăn, khó đạt được mô hình tốt, có khi là không tồn tại
- Hệ thống tạo ra có thể lớn và chậm

Tuy vậy, HCG dựa trên mô hình là một bổ sung quan trọng vào các gói phần mềm công nghệ tri thức, đặc biệt trong lĩnh vực chẩn đoán.

HCG dựa trên trường hợp

- HCG dựa trên trường hợp (Case-based Reasoning – CBR) sử dụng một CSDL riêng biệt chứa giải pháp của các trường hợp đã giải quyết, để dựa vào đó tìm kiếm giải pháp cho một trường hợp mới.
- Phương pháp này minh họa cách giải quyết vấn đề của các chuyên gia trong nhiều lĩnh vực: luật sư, lập trình viên, kiến trúc sư, sử gia …
- Để giải quyết một vấn đề, một CBR phải:

1. Truy vấn các trường hợp thích hợp từ bộ nhớ của nó, dựa vào *sự tương tự* của một số *đặc điểm nổi bật*.
2. Sửa đổi trường hợp đó để có thể áp dụng trong tính huống hiện tại.
3. Áp dụng trường hợp đã chuyển đổi vào bài toán mới.
4. Lưu lại lời giải và kết quả của nó (thành công hay thất bại).

Ưu điểm của HCG dựa trên trường hợp

- Khả năng lưu trữ một cách trực tiếp các tri thức có được => có thể loại bỏ việc tích lũy tri thức từ các chuyên gia.
- Cho phép rút ngắn thời gian suy luận.
- Tạo khả năng tự học của hệ thống: giúp hệ thống tránh lỗi cũ và tận dụng những thành công trong quá khứ
- Việc phân tích tri thức của lĩnh vực chỉ diễn ra một lần, đó là khi tìm kiếm một sự biểu diễn hợp lý cho các trường hợp.
- Việc tích lũy tri thức và lập trình là tương đối đơn giản.
- Các chiến lược sắp xếp (index) thích hợp làm tăng sức mạnh của phương pháp này.

Khuyết điểm của HCG dựa trên trường hợp

- Các trường hợp không thể hiện tri thức sâu về lĩnh vực bài toán => khó giải thích tại sao đưa ra lời giải như vậy, hoặc có thể đưa ra lời giải sai hoặc không tốt.
- Một cơ sở chứa các trường hợp lớn phải xem xét sự tương xứng giữa tính toán và lưu trữ.
- Khó đưa ra tiêu chuẩn đánh giá sự tương tự của các trường hợp, và sắp xếp chúng.

Chương 7
Suy luận với thông tin không chắc chắn hoặc không đầy đủ

Giáo viên: Trần Ngân Bình

Nội Dung

- Các nguyên nhân của sự không chắc chắn:
  - Dữ liệu/thông tin/tri thức có thể: không đủ, không đáng tin cậy, không đúng, không chính xác
  - Các phép suy luận có thể không hợp logic: suy luận ngược từ kết luận về điều kiện (abduction reasoning)
- Xử lý trường hợp không chắc chắn:
  - Tiếp cận thống kê: quan tâm đến mức độ tin tưởng (belief) của một khẳng định.
    - Lý thuyết xác suất Bayesian (Bayesian Probability Theory)
    - Đại số chắc chắn Stanford (The Stanford Certainty Algebra)
  - Suy luận theo Loggic mờ (Fuzzy Logic) quan tâm đến mức độ thật (truth) của một khẳng định.

Xác suất

- Hữu dụng để:
  - Mô tả một thế giới hoàn toàn ngẫu nhiên (chơi bài,…)
  - Mô tả một thế giới bình thường (mối tương quan thống kê,…)
  - Mô tả các ngoại lệ (tỉ lệ xuất hiện lỗi,…)
  - Làm cơ sở cho việc học của máy (quy nạp cây quyết định,…)
- Thường xác suất được dùng cho:
  - Sự kiện: xác suất của việc quan sát một chứng cớ nào đó.
  - Giả thuyết: xác suất để giả thuyết đúng.
- Theo xác suất truyền thống: tần số xuất hiện tương đối của một sự kiện trong một thời gian dài sẽ tiến đến xác suất của nó.

Lý thuyết xác suất

$P(e) \in [0,1]$

$P(e_1) + P(e_2) + \ldots + P(e_n) = 1$

Ví dụ: đồng xu tốt P(mặt_sấp) = P(mặt_ngửa) = 0.5

đồng xu không đều P(mặt_sấp) =0.7 P(mặt_ngửa) = 0.3

- *Nếu sự kiện e_1 và e_2 độc lập nhau:*

  $P(e_1 \text{ And } e_2) = P(e_1) * P(e_2)$

  $P(e_1 \text{ Or } e_2) = P(e_1) + P(e_2) - P(e_1) * P(e_2)$

  $P(\text{Not } e) = 1 - P(e)$

  Ví dụ: tung 2 đồng xu: các khả năng có thể xảy ra là SS SN NS NN, suy ra:

  P(S And N) = ¼ = 0.25 P(S Or S) = ¾ = 0.75

Xác suất có điều kiện

- Xác suất tiên nghiệm (prior probability) hay xs vô điều kiện (unconditional probability): là xs của một sự kiện trong điều kiện không có tri thức bổ sung cho sự có mặt hay vắng mặt của nó.
- Xác suất hậu nghiệm (posterior probability) hay xs có điều kiện(conditional probability): là xs của một sự kiện khi biết trước một hay nhiều sự kiện khác

$$P(e1|e2) = \frac{|e1 \text{ and } e2|}{|e2|}$$

- Ví dụ: P(cúm) = 0.001 P(sốt) = 0.003
  P(cúm And sốt) = 0.000003
  nhưng cúm và sốt là cá sự kiện không độc lập
  các chuyên gia cho biết: P(sốt | cúm) = 0.9

Suy luận Bayesian (1)

- P(h|e) là xác suất khẳng định *giả thuyết* h đúng cho trước *bằng chứng* e.

$$P(h|e) = \frac{P(e|h) * P(h)}{P(e)} \quad <= \text{luật Bayes}$$

Công thức này nói rằng xác suất đúng của giả thuyết h khi quan sát được bằng chứng e, bằng với xác xuất cho rằng chúng ta sẽ quan sát được bằng chứng e nếu giả thuyết h là đúng, nhân với xác suất tiên nghiệm của h, tất cả chia cho xác suất tiên nghiệm của việc quan sát được bằng chứng e.

Suy luận Bayesian (2)

Ví dụ: Bằng chứng (triệu chứng): bệnh nhân bị sốt

Giả thuyết (bệnh): bệnh nhân bị cảm cúm

$$P(\text{cúm}|\text{sốt}) = \frac{P(\text{cúm}) * P(\text{sốt}|\text{cúm})}{P(\text{sốt})} = \frac{0.001 * 0.9}{0.003} = 0.3$$

Các con số ở vế phải thì dễ đạt được hơn con số ở vế trái

- Khi nào bằng chứng e không làm tăng xác suất đúng của giả thuyết h?
  - Khi xác suất của giả thuyết h đã là 1.0
  - Khi bằng chứng e không liên quan gì đến giả thuyết h

Tại sao sử dụng luật Bayes?

Tri thức về nguyên nhân (knowledge of causes):

P (sốt | cúm)

thì dễ dàng có được hơn là tri thức về chẩn đoán (diagnostic knowledge):

P (cúm | sốt).

Luật Bayes cho phép chúng ta sử dụng tri thức về nguyên nhân để suy ra tri thức về chẩn đoán.

Các vấn đề trong suy luận Bayes

Việc tính toán các xác suất tiên nghiêm và hậu nghiệm liên quan đòi hỏi một sự thu thập dữ liệu rất lớn

- Trong thực tế phải xử lý nhiều triệu chứng
  - Chỉ có vài triệu chứng là độc lập nhau:

$$P(s_i|s_j) = P(s_i)$$

  - Nếu chúng không độc lập nhau:

$$P(d \mid s_1 \,\&\, s_2 \,\&\ldots\, s_n) = \frac{P(d) * P(s_1 \,\&\, s_2 \,\&\ldots\, s_n \mid d)}{P(s_1 \,\&\, s_2 \,\&\ldots\, s_n)}$$

- Đối với thông tin phủ định:

$$P(\text{not } s) = 1 - P(s) \quad \text{và} \quad P(\text{not } d \mid s) = 1 - P(d \mid s)$$

Sự độc lập của các điều kiện trong luật Bayes

- Trong thực tế có nhiều giả thuyết canh tranh nhau, vì vậy công thức Bayes tổng quát nhất là:

$$P(h_i \mid e) = \frac{P(e \mid h_i) * P(h_i)}{\Sigma_k (P(e \mid h_k) * P(h_k))}$$

Đòi hỏi tất cả các $P(e \mid h_k)$ phải **độc lập nhau**.

- Giả sử các chấm đỏ và sốt là độc lập về điều kiện khi cho trước bệnh sởi:

P(các chấm đỏ, sốt | sởi) = P(các chấm đỏ| sởi) P (sốt| sởi)

Khi đó ta có thể kết luận:

P(các chấm đỏ, sốt, sởi) = P(các chấm đỏ, sốt | sởi) P(sởi)

= P(các chấm đỏ | sởi) P(sốt | sởi) P(sởi)

Các yếu tố chắc chắn Stanford

Không phải là xác suất, mà là độ đo sự tự tin.
Lý thuyết chắc chắn là một cố gắng hình thức hóa tiếp cận heuristic vào suy luận với sự không chắc chắn

- Các chuyên gia đo sự tự tin trong các kết luận của họ và các bước suy luận bằng từ ‘không có lẻ’, ‘gần như chắc chắn’, ‘có khả năng cao’, ‘có thể’. Đây không phải là xác suất mà là heuristic có từ kinh nghiệm.
- Các chuyên gia có thể đặt sự tự tin vào các mối quan hệ mà không phải có cảm giác là nó không đúng.

MB(H | E) đo độ tin tưởng của giả thuyết H, cho trước E

MD(H | E) đo độ không tin tưởng

$0 < MB(H \mid E) < 1$ trong khi $MD(H \mid E) = 0$

$0 < MD(H \mid E) < 1$ trong khi $MB(H \mid E) = 0$

$$CF(H \mid E) = MB(H \mid E) - MD(H \mid E)$$

Đại số chắc chắn Stanford (1)

CF(fact) ∈[-1,1] : dữ liệu đã cho, dữ liệu suy luận được, giả thuyết

- Một CF tiến về 1 cho thấy sự tin tưởng dữ kiện là đúng
- Một CF tiến về -1 cho thấy sự tin tưởng dữ kiện là không đúng
- Một CF xung quanh 0 cho thấy tồn tại rất ít bằng cớ cho việc ủng hộ hay chống lại dữ kiện. => một giới hạn được đưa ra nhằm tránh việc suy luận với thông tin không chắc chắn như vậy (vd: 0.2)

CF(rule) ∈[-1,1] : thể hiện sự tin tưởng của các chuyên gia vào độ tin cậy của luật.

- Kết hợp các CF

  CF (A And B) = Min[CF(A), CF(B)]

  CF (A Or B) = Max[CF(A), CF(B)]

Ví dụ: CF(bệnh nhân bị sốt) = 0.9

CF(bệnh nhân bị hắc hơi) = 0.6

CF(bệnh nhân bị sốt And bệnh nhân bị hắc hơi) = 0.6

CF(bệnh nhân bị sốt Or bệnh nhân bị hắc hơi) = 0.9

Đại số chắc chắn Stanford (2)

- Truyền CF trên các luật:

  CF(Q) = CF(If P Then Q) * CF(P)

Ví dụ: CF(bệnh nhân bị sốt) = 0.8

CF(If bệnh nhân bị sốt Then bệnh nhân bị cúm) = 0.5

CF(bệnh nhân bị cúm) = 0.4

- Kết hợp nhiều CF từ nhiều luật

  If P Then Q -> $CF_1(Q)$

  If R Then Q -> $CF_2(Q)$

$$CF(Q) = CF_1(Q) + CF_2(Q) - CF_1(Q) * CF_2(Q)$$ Khi CF_1 & $CF_2 > 0$

$$= CF_1(Q) + CF_2(Q) + CF_1(Q) * CF_2(Q)$$ Khi CF1 & CF2 < 0

$$= \frac{CF_1(Q) + CF_2(Q)}{1 - Min\,(|CF_1(Q)|, |CF_2(Q)|)}$$ Ngoài ra

Đại số chắc chắn Stanford (3)

Ví dụ: CF(bệnh nhân bị sốt) = 1

CF(bệnh nhân bị hắc hơi) = 0.8

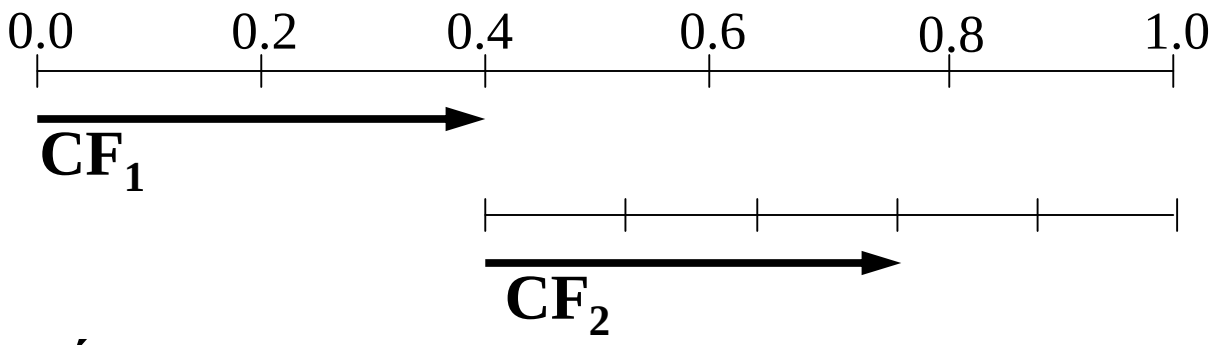
CF(If bệnh nhân bị hắc hơi Then bệnh nhân bị cúm) = 0.5

CF(If bệnh nhân bị sốt Then bệnh nhân bị cúm) = 0.6

CF_1(bệnh nhân bị cúm) = 0.4

CF_2(bệnh nhân bị cúm) = 0.6

CF(bệnh nhân bị cúm) = 0.4 + 0.6 – 0.24 = 0.76



Tính chất: kết quả CF phải nằm trong khoảng [-1,+1]

kết hợp các CF nghịch nhau sẽ xóa bớt lẫn nhau

Phép đo CF kết hợp phải mang tính tuyến tính

Mycin

- Mục đích: Giúp đỡ các bác sĩ trong việc chẩn đoán và điều trị các bệnh truyền nhiễm
  1. Nhận dạng các cơ quan bị nhiễm bệnh
  2. Chọn các loại thuốc khống chế các cơ quan này
- Giao diện người dùng: Đối thoại với bác sĩ để thu thập dữ liệu
  1. Dữ liệu tổng quát về bệnh nhân
  2. Các kết quả xét nghiệm
  3. Các triệu chứng của bệnh nhân

EMYCIN = MYCIN – Tri thức Y học
= Sườn hệ chuyên gia (ES shell)

Biểu diễn tri thức của Mycin

- Dữ kiện:

Thông số	Ngữ cảnh	Giá trị	CF
Nhận ra	Cơ_quan_1	Klebsiella	.25
Nhạy cảm	Cơ_quan_1	Penicillin	-1.0

- Luật: Luật + diễn giải của luật

IF (a) the infection is primary-bacteria, and
(b) the site of the culture is one of the serile sites, and
(c) the suspected portal of entry is gastrointestinal tract
THEN there is suggestive evidence **(.7)** that infection is bacteroid

IF: (AND (same_context infection primary_bacteria)
(membf_context site sterilesite)
(same_context portal GI))
THEN: (conclude context_ident bacteroid tally **.7**)

Suy luận của Mycin

- Ngữ cảnh: các đối tượng được thảo luận bởi Mycin
  - Các kiểu đối tượng khác nhau: bệnh nhân, thuốc, …
  - Được tổ chức trong một cây
- Động cơ suy diễn: tiếp cận hướng từ mục tiêu hay suy diễn lùi
  - Tìm kiếm sâu gần như là vét cạn
  - Có thể suy luận với thông tin không chắc chắn
  - Có thể suy luận với dữ liệu không đầy đủ
- Các tiện ích giải thích: Mô-đun ‘hỏi-trả lời’ với các câu hỏi tại sao, như thế nào.

Ví dụ Mycin

Chân của John đang bị đau (1.0). Khi tôi kiểm tra nó, thấy nó sưng tấy (0.6) and hơi đỏ (0.1). Tôi không có nhiệt kế nhưng tôi nghĩ anh ta có bị sốt (0.4). Tôi biết John là một vận động viên marathon, các khớp của anh ta thường xuyên làm việc quá tải (1.0). John có thể di chuyển chân của anh ấy.

Liệu chân của John bị gãy, quá mỏi, hay bị nhiễm trùng?

1. IF đau và sốt THEN bị nhiễm trùng 0.6
2. IF đau và sưng THEN bị chấn thương 0.8
3. IF quá tải THEN bị nhiễm trùng 0.5
4. IF bị chấn thương AND đỏ THEN bị gãy 0.8
5. IF bị chấn thương AND di chuyển được THEN quá mỏi 1.0

Một luật heuristic của Mycin

IF tuổi bệnh nhân <7 THEN không nên cấp thuốc tetracyline

- Tri thức miền:
  - Tetracyline làm đổi màu xương đang phát triển
  - trẻ em dưới 7 tuổi thì đang mọc răng
- Tri thức giải quyết vấn đề:
  - Trước khi kê một loại thuốc phải kiểm tra các chống chỉ định
  - Có hai loại chống chỉ định: liên quan đến bệnh và liên quan đến bệnh nhân.
- Tri thức về thế giới:
  - Hàm răng màu nâu thì không đẹp

Luật heuristic biên dịch tất cả những thông tin này và vì vậy hỗ trợ một phương pháp giải quyết vấn đề hiệu quả

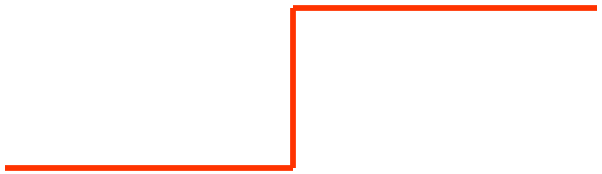
Điều khiển cài trong luật của Mycin

IF sự nhiễm trùng là bệnh viêm màng não
And sự nhiễm trùng là do vi khuẩn
And chỉ có chứng cớ gián tiếp
And tuổi của bệnh nhân > 16
And bệnh nhân là một người nghiện rượu
THEN chứng cớ cho viêm phổi song cầu khuẩn **0.7**

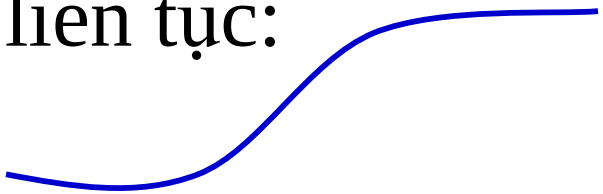
- Tri thức miền:
  - Các bệnh nhân bị nghiện rượu thì đáng nghi ngờ với vi khuẩn viêm phổi song cầu khuẩn
- Tri thức giải quyết vấn đề
  - Lọc sự chẩn đoán theo từng bước
- Tri thức về thế giới
  - Người nghiện rượu thì hiếm khi dưới 17 tuổi
  - Câu hỏi gây sốc cho cha mẹ của các trẻ nhỏ.

Logic Mờ (Fuzzy Logic)

- Một số phần của thế giới là nhị phân:
  - Con mimi của tôi là một con mèo
- Một số phần thì không:
  - An thì khá cao, Bảo thì thuộc loại cao, tôi thì hơi cao, Trân thì không cao lắm
- Nhị phân có thể biểu diễn bằng một đồ thị:



- Logic mờ cũng có thể biểu diễn bằng đồ thị, nhưng là đồ thị liên tục:



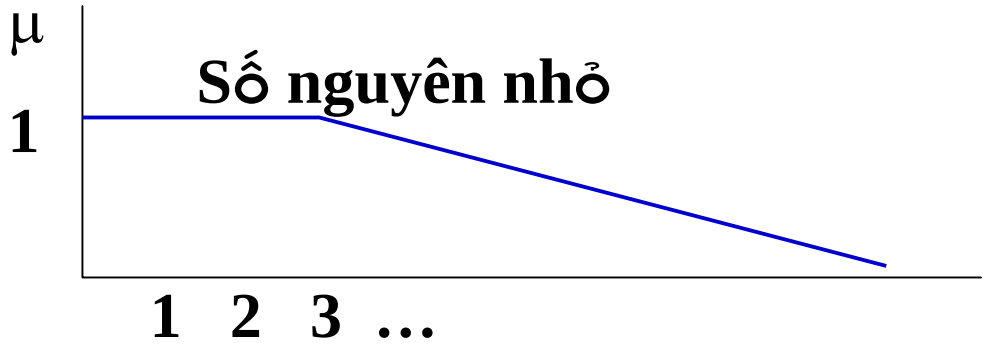
Tập Mờ

- Cho S là một tập hợp và x là một phần tử của tập hợp đó. Một tập con mờ F của S được định nghĩa bởi một ***hàm tư cách thành viên*** **μF(x)** đo "mức độ" mà theo đó x thuộc về tập F. Trong đó, $0 \leq \mu F(x) \leq 1$.
  - Khi $\mu F(x) = 0$ => $x \notin$ F hoàn toàn.
  - Khi $\mu F(x) = 1$ => $x \in$ F hoàn toàn.
- Nếu $\forall x, \mu F(x) = 0$ hoặc 1

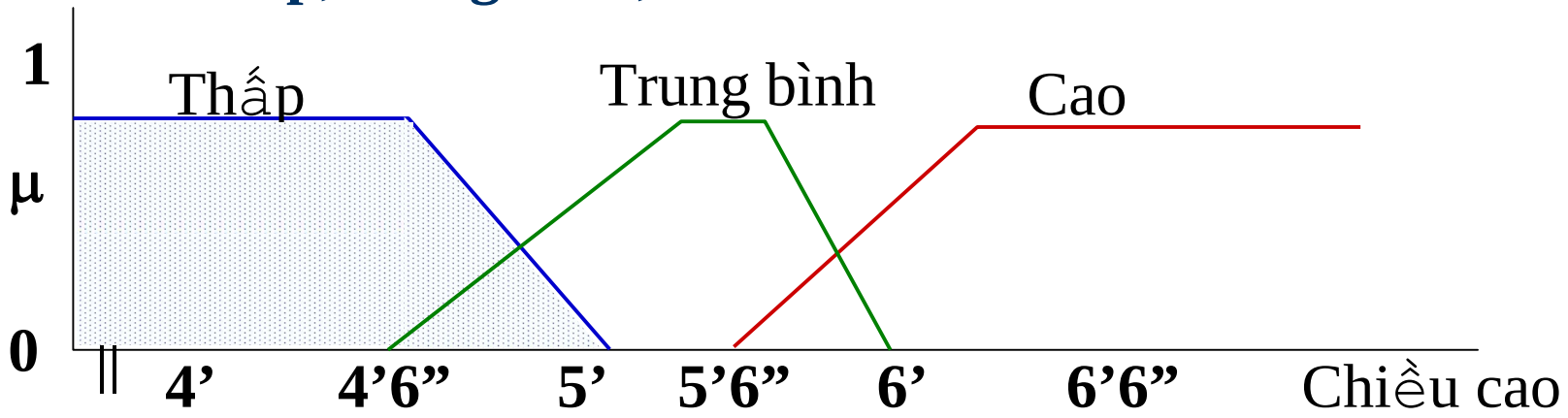
  thì F được xem là "giòn"
- Hàm thành viên μF(x) thường được biểu diễn dưới dạng đồ thị.

Ví dụ Tập Mờ

<u>Ví dụ 7.7:</u> S là tập hợp tất cả các *số nguyên dương* và F là tập con mờ của S được gọi là *"số nguyên nhỏ"*



Ví dụ: 7.8: Một sự biểu diễn tập mờ cho các tập người đàn ông thấp, trung bình, và cao.



Tính Chất của Tập Mờ

- *Hai tập mờ bằng nhau*:

  $A = B$ nếu $\forall x \in X, \quad \mu A(x) = \mu B(x)$

- *Tập con*: $A \subseteq B$ nếu $\forall x \in X, \quad \mu A(x) \leq \mu B(x)$

- *Một phần tử có thể thuộc về nhiều hơn một tập mờ.*

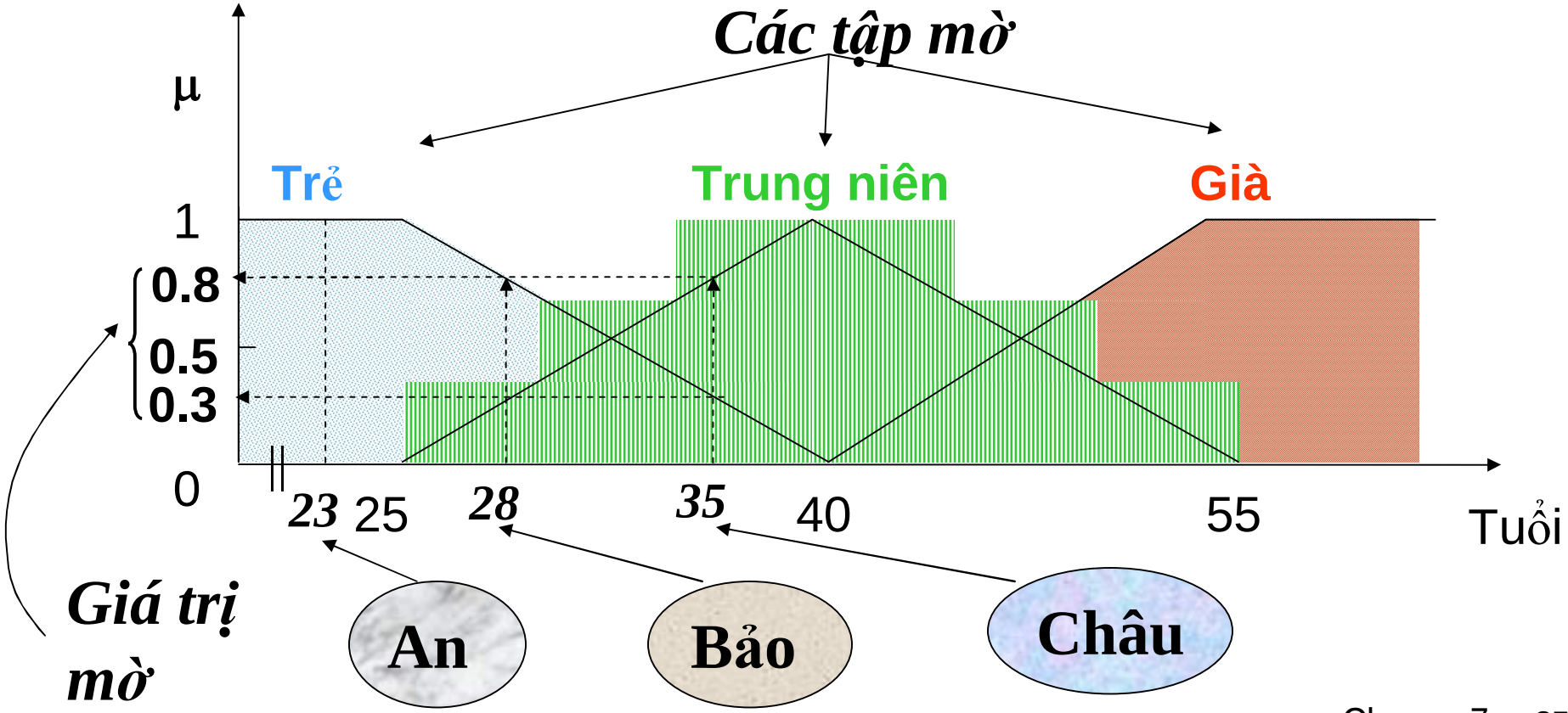
  Ví dụ: (hình 7.5) một người đàn ông cao 5’10” thuộc về cả hai tập “trung bình” và “cao”.

- *Tổng các giá trị mờ của một phần tử khác 1*:

  $\mu\text{Thấp}(x) + \mu\text{Trungbình}(x) + \mu\text{Cao}(x) \neq 1$

Mờ hóa (fuzzification)

- Từ hàm thành viên cho trước, ta có thể suy ra được mức độ một thành viên thuộc về một tập hợp, hay giá trị mờ của nó đối với một tập mờ.



Hợp của hai tập mờ

- ***Khái niệm***: Hợp của hai tập mờ (A∪B) thể hiện mức độ một phần tử thuộc về một trong hai tập là bao nhiêu.
- ***Công thức***: $\mu A \vee B(x) = \max(\mu A(x), \mu B(x))$
- *Thí dụ 7.10*:

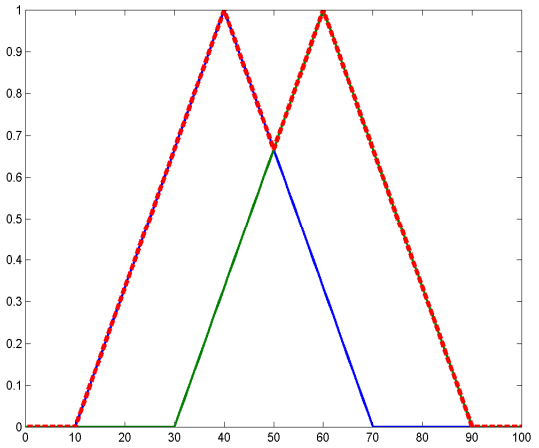
$\mu Tre(An) = 0.8$

và $\mu Trung\ niên(An) = 0.3$

$\Rightarrow \mu Tre \vee Trung\ Niên(An)$

$= \max(0.8, 0.3) = 0.8$

A ∪ B



Giao của hai tập mờ

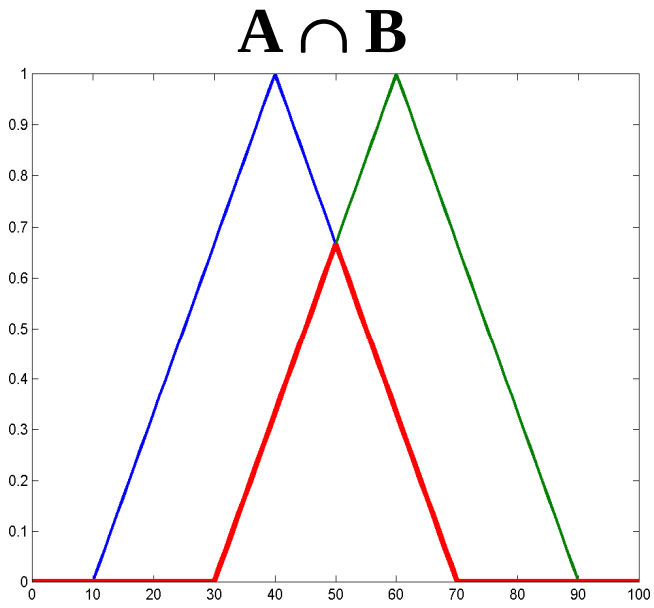
- ***Khái niệm***: Giao của hai tập mờ (A∩B) thể hiện mức độ một phần tử thuộc về cả hai tập là bao nhiêu.
- ***Công thức***: $\mu A \wedge B(x) = \min(\mu A(x), \mu B(x))$
- *Thí dụ 7.11*:

μTre(An) = 0.8

và μTrung niên(An) = 0.3

=> μTre $\wedge$ Trung Niên(An)

= min(0.8, 0.3) = 0.3



Bù của một tập mờ

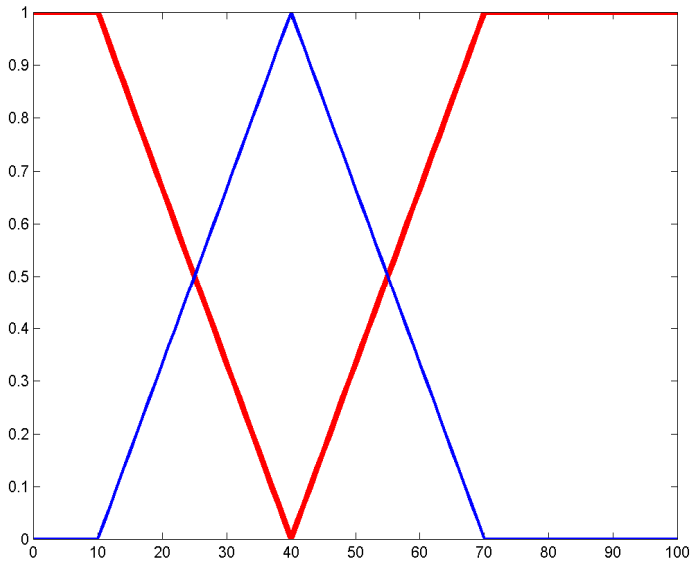
- Khái niệm: Bù của một tập mờ thể hiện mức độ một phần tử không thuộc về tập đó là bao nhiêu.
- Công thức: $\mu \neg A(x) = 1 - \mu A(x)$
- *Thí dụ 7.12*:

μTrẻ(An) = 0.8

=> $\mu \neg$Trẻ(An)

= 1 – 0.8 = 0.2

A’



Luật mờ

- Một luật mờ là một biểu thức if - then được phát biểu ở dạng ngôn ngữ tự nhiên thể hiện sự phụ thuộc nhân quả giữa các biến.
- *Thí dụ 7.14*:

  if ***nhiệt độ*** là *lạnh*

  và ***giá dầu*** là *rẻ*

  then ***sưởi ấm*** *nhiều*.

Biến

Giá trị của biến (hay tập mờ)

Hoặc:

if một người có ***chiều cao*** là *cao* và ***cơ bắp*** là *lực lưỡng* **then** ***chơi bóng rổ*** *hay*.

Nhận xét

- Logic mờ không tuân theo các luật về tính bù của logic truyền thống:

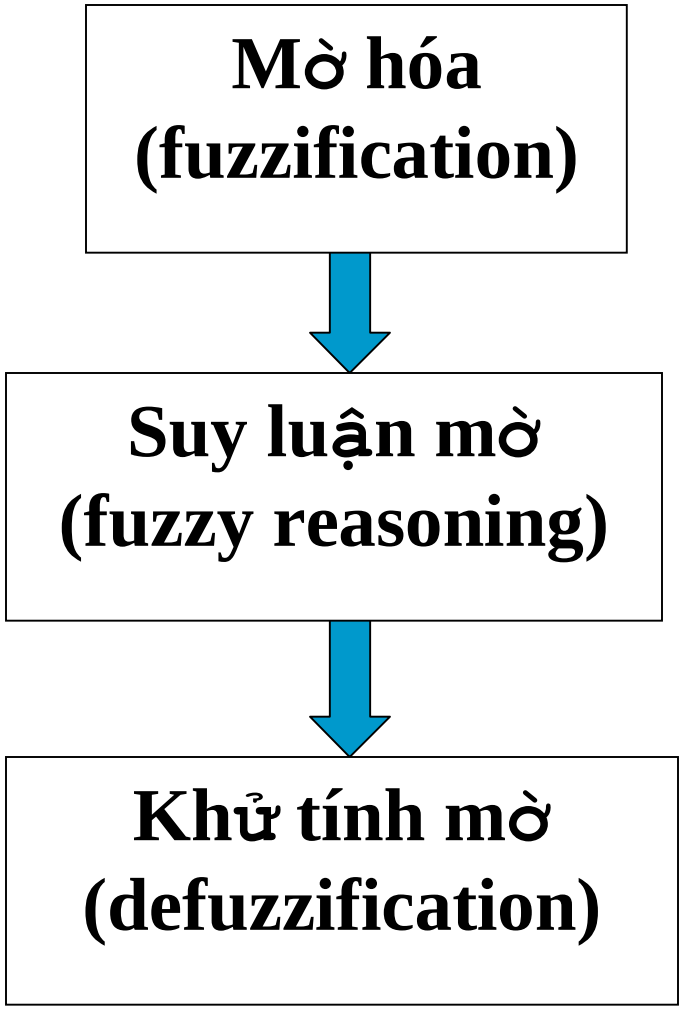
$$\mu_{\neg A \vee A}(x) \equiv 1 \qquad \text{và} \qquad \mu_{\neg A \wedge A}(x) \equiv 0$$

- *<u>Thí dụ 7.13</u>*:

$$\mu_{\neg A \vee A}(x) = \max(0.8, 0.2) = 0.8$$

$$\mu_{\neg A \wedge A}(x) = \min(0.8, 0.2) = 0.2$$

Thủ tục ra quyết định mờ (fuzzy decision making procedure)



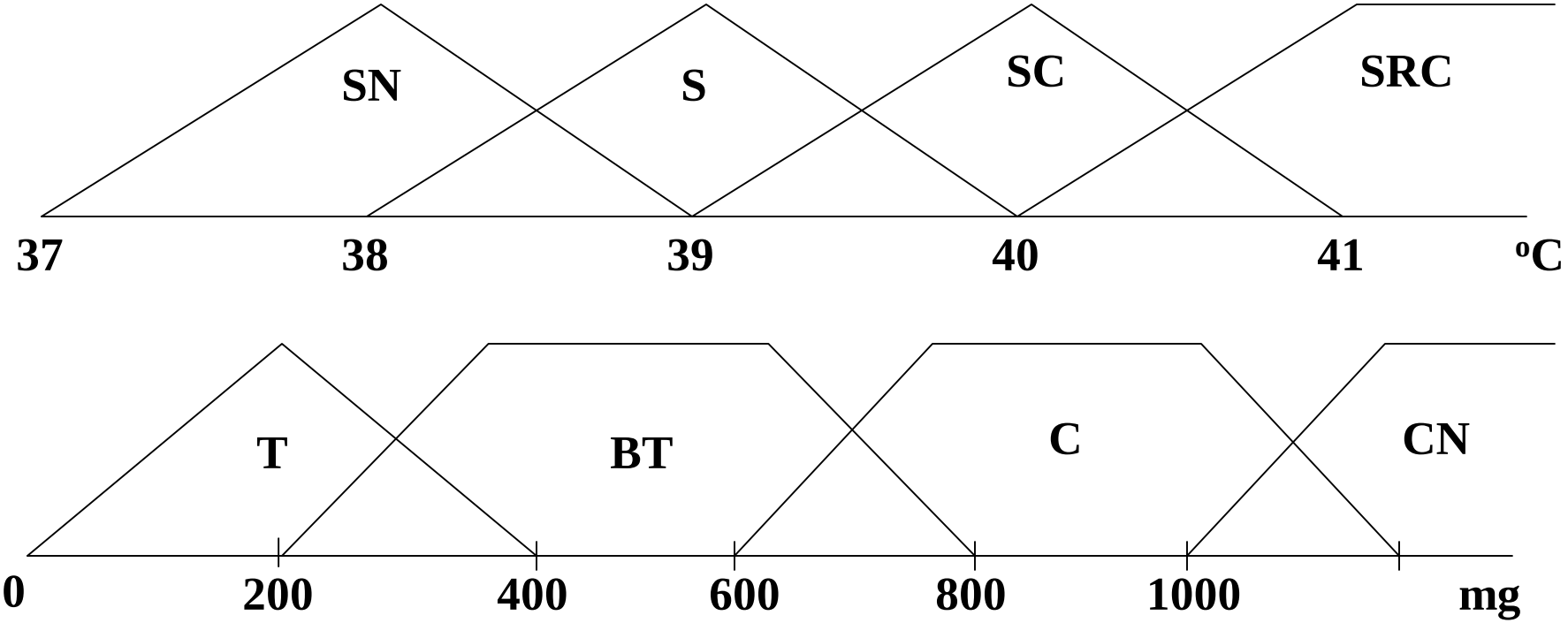
Chuyển các giá trị của dữ liệu thực tế về dạng mờ

Thực hiện tất cả các luật khả thi, các kết quả sẽ được kết hợp lại

Chuyển kết quả ở dạng mờ về dạng dữ liệu thực tế

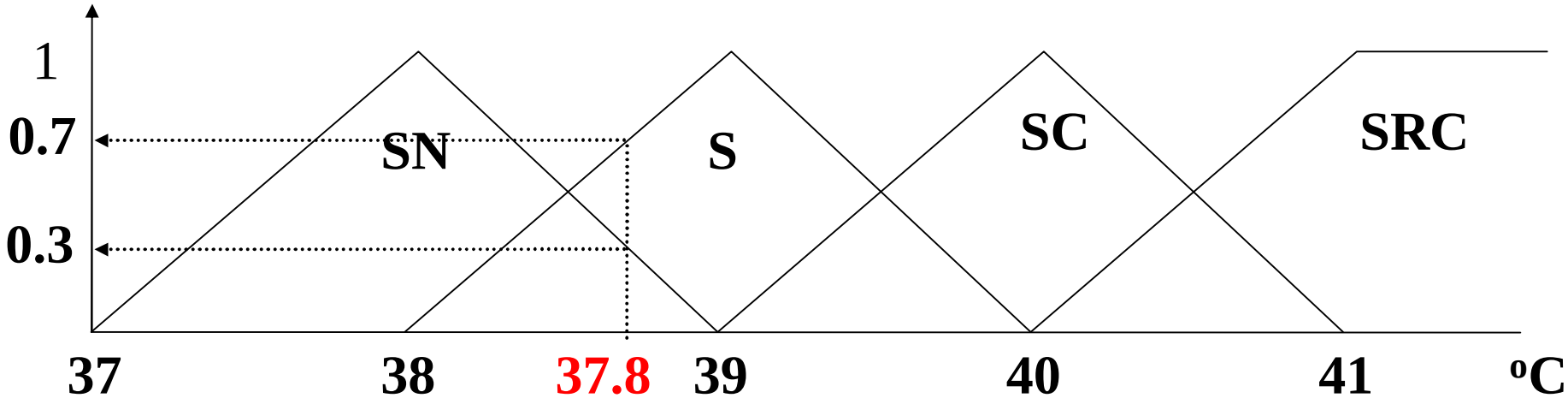
Hệ thống mờ dùng trong điều trị bệnh

- IF sốt nhẹ THEN liều lượng asperine thấp
- IF sốt THEN liều lượng asperine bình thường
- IF sốt cao THEN liều lượng asperine cao
- IF sốt rất cao THEN liều lượng asperine cao nhất



Ví dụ: Một bệnh nhân sốt ở 38.7 độ. Hãy xác định liều lượng asperince cần thiết để cấp cho bệnh nhân

- **Bước 1**: Mờ hóa giá trị x = 37.8 đã cho ta thấy 37.8 thuộc về các tập mờ như sau:



μSốt nhẹ (x) = 0.3 μSốt (x) = 0.7

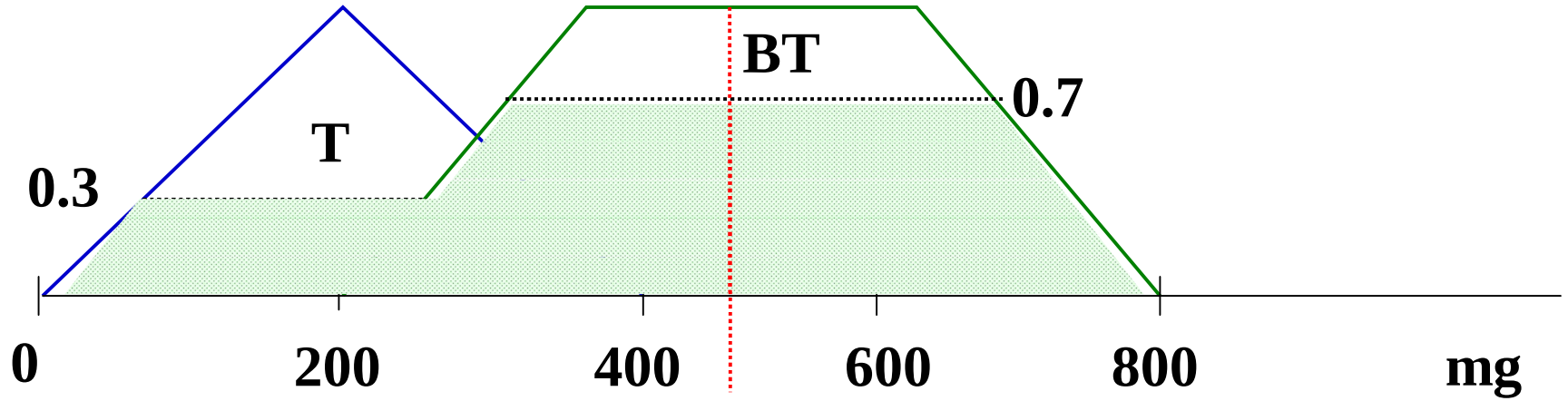
μSốt cao (x) = 0 μSốt rất cao (x) = 0

Ví dụ (tt.)

- **Bước 2:** Ta thấy có 2 luật 1 và 2 có thể áp dụng cho ra hai liều lượng aspirine:

  μThấp (x) = 0.3 μBình thường (x) = 0.7

- Kết hợp các giá trị mờ này lại ta được vùng được tô màu sau đây:



Ví dụ (tt.)

- **Bước 3:** Phi mờ hóa kết quả bằng cách tính trọng tâm của diện tích được tô trong hình trên:
  - Chiếu xuống trục hoành ta được giá trị ±480mg
- **Kết luận**: liều lượng aspirine cần cấp cho bệnh nhân là 480mg.

Tóm Tắt

- Vận dụng công thức Bayes để tính xác suất của một giả thuyết.
- Hiểu nguyên tắc hoạt động của HCG MYCIN
- Vận dụng đại số hệ số chắc chắn Stanford vào hệ chuyên gia MYCIN.
- Hiểu lý thuyết về logic mờ & ứng dụng của nó vào các HCG mờ.
- Biết lựa chọn phương pháp suy luận phù hợp với vấn đề cần giải quyết.

Chương 12 -

SUY LUẬN TỰ ĐỘNG
(AUTOMATIC REASONING)

Giáo viên: Trần Ngân Bình

Ví dụ 1: Xét ví dụ với tập hợp các câu biểu diễn trong Phép tính vị từ như sau:

1) man (marcus)
2) pompeian (marcus)
3) $\forall X$ pompeian (X) $\rightarrow$ roman(X)
4) ruler (caesar)
5) $\forall X$ roman(X) $\rightarrow$ loyalto(X, caesar) $\vee$ hate(X, caesar)
6) $\forall X, \exists Y$ loyalto (X,Y)
7) $\forall X, \forall Y$ person(X) $\wedge$ ruler(Y) $\wedge$ trytoassasinate(X) $\rightarrow$ $\neg$loyalto (X,Y)
8) trytoassasinate (marcus, caesar)

Chứng minh $\neg$loyalto(marcus, caesar)

Hình bên dưới minh họa một cách chứng minh cho mục tiêu trên:

$\neg$loyalto (marcus, caesar)

↑ *(câu 7, {marcus/X, ceasar/Y})*

person(marcus) $\wedge$ ruler(caesar) $\wedge$

trytoassasinate (marcus, caesar)

↑ *(câu 4)*

person(marcus) $\wedge$ trytoassasinate(marcus, caesar)

↑ *(câu 8)*

person (marcus)

Marcus là một người đàn ông (man)

? => Marcus là một người (person).

Vì vậy, ta phải thêm

9) $\forall X$ man(X) $\vee$ woman(X) $\rightarrow$ person(X)

Thủ tục hợp giải (Resolution)

1. **Chuyển về dạng mệnh đề (Clause form):**
2. **Cơ sở của Hợp giải (Resolution):**
3. **Giải thuật hợp giải dùng cho Logic mệnh đề:**
4. **Giải thuật hợp giải dùng cho Logic vị từ**

<u>Một số vấn đề liên quan đến giải thuật Hợp giải:</u>

- **Hợp giải có thể phát hiện trường hợp không tồn tại sự mâu thuẩn**
- **Sử dụng hàm tính toán, vị từ tính toán, và mối quan hệ bằng**
- **Trả lời câu hỏi**
- **Nhận xét về Hợp giải**

Chuyển về dạng mệnh đề (Clause form)

* Nhu cầu chuyển câu về dạng mệnh đề:

Xét câu: $\forall X$ [roman (X) $\wedge$ know (X,marcus) $\rightarrow$

[hate(X,Caesar) $\vee$ ($\forall Y$, $\exists Z$ hate(Y,Z) $\rightarrow$ thinkcrazy(X,Y))]

Công thức sẽ dễ dàng thao tác hơn nếu chúng nó:

– Phẳng hơn, nghĩa là có ít thành phần được nhúng vào.

– Các lượng tử biến ($\forall$,$\exists$) được tách khỏi phần còn lại.

Conjunctive Normal Form (CNF) của câu trên:

$\neg$roman (X) $\vee$ $\neg$know (X,Marcus) $\vee$ hate(X,Caesar) $\vee$ $\neg$hate(Y,Z) $\vee$ thinkcrazy(X,Y)

Để có thể sử dụng thủ tục Robinson, ta phải chuyển các wff thành một tập hợp các *mệnh đề (clauses).*

Mệnh đề: (clause) là một wff ở dạng CNF nhưng không có sự hiện diện của phép kết nối And ($\wedge$), hay nói khác hơn mỗi mệnh đề là tuyển ($\vee$) của các ***biến mệnh đề*** *(literal).*

Giải thuật chuyển câu về mệnh đề:

1. **Loại bỏ dấu →** $a \rightarrow b = \neg a \vee b$
2. **Thu hẹp phạm vi của toán tử ¬**
   a. $\neg(\neg p) = p$
   b. Luật De Morgan
   $\neg(a \wedge b) = \neg a \vee \neg b$ hay $\neg(a \vee b) = \neg a \wedge \neg b$
   c. $\neg\forall X\ p(X) = \exists X\ \neg p(X)$ hay $\neg\exists X\ p(X) = \forall X\ \neg p(X)$
3. **Chuẩn hóa các biến** sao cho mỗi lượng tử chỉ kết nối với một biến duy nhất.
   Ví dụ: $\forall X\ p(X) \vee \forall X\ q(X)$ ⇨ $\forall X\ p(X) \vee \forall Y\ q(Y)$
4. **Dịch chuyển tất cả các lượng tử về bên trái**.
5. **Xóa bỏ các lượng tử tồn tại (∃).**
   Ví dụ: $\exists Y$ president(Y) được chuyển thành president(S1)
   Với S1 là một hàm tạo ra giá trị thỏa mãn vị từ president.
   Ví dụ: $\forall X\ \exists Y$ father_of (Y,X) ⇨ $\forall X$ father_of (S2(X),X)
   S1, S2 được gọi là ***hàm Skolem***.
   S1 còn được gọi là ***hằng Skolem***.

6. **Bỏ đi các lượng tử phổ biến**
7. **Chuyển công thức về dạng hội của các tuyển:**
   $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$
   hay $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$
   Ví dụ: $(winter \wedge wearingboots) \vee (summer \wedge wearingsandals)$
   $\Rightarrow$ $[(winter \vee (summer \wedge wearingsandals)]$
   $\wedge [wearingboots \vee (summer \wedge wearingsandals)]$
   $\Rightarrow$ $(winter \vee summer) \wedge (winter \vee wearingsandals) \wedge$
   $(wearingboots \vee summer) \wedge (wearingboots \vee wearingsandals)$

8. **Tạo ra các mệnh đề tách biệt**
   Ví dụ: từ kết quả ở bước 7, ta có thể tách thành 4 mệnh đề.

9. **Chuẩn hoá các biến trong tập hợp các mệnh đề vừa tạo ở bước 8**, nghĩa là đặt lại tên cho các biến sao cho không có hai mệnh đề có cùng tên biến.

Ví dụ chuyển câu về dạng mệnh đề

$\forall X$ [roman (X) $\wedge$ know (X,marcus) $\rightarrow$
[hate(X,caesar) $\vee$ ($\forall Y$, $\exists Z$ hate(Y,Z) $\rightarrow$ thinkcrazy(X,Y))]

1. **Loại bỏ dấu $\rightarrow$** $\forall X \neg$ [roman (X) $\wedge$ know (X,marcus)] $\vee$
   [hate(X,caesar) $\vee$ ($\forall Y$, $\neg$($\exists Z$ hate(Y,Z)) $\vee$ thinkcrazy(X,Y))]
2. **Đưa $\neg$ vào trong** $\forall X$ [$\neg$roman (X) $\vee$ $\neg$know (X,marcus)] $\vee$
   [hate(X,caesar) $\vee$ ($\forall Y$, $\forall Z$ $\neg$hate(Y,Z) $\vee$ thinkcrazy(X,Y))]
3. **Chuẩn hoá các biến** ✓
4. **Dịch chuyển tất cả các lượng tử về bên trái**
   $\forall X$, $\forall Y$, $\forall Z$ [$\neg$roman (X) $\vee$ $\neg$know (X,marcus)] $\vee$
   [hate(X,caesar) $\vee$ ($\neg$hate(Y,Z) $\vee$ thinkcrazy(X,Y))]
5. **Xoá bỏ các lượng tử tồn tại** ✓
6. **Bỏ đi lượng tử phổ biến**
   [$\neg$roman (X) $\vee$ $\neg$know (X,marcus)] $\vee$
   [hate(X,caesar) $\vee$ ($\neg$hate(Y,Z) $\vee$ thinkcrazy(X,Y))]
7. **Chuyển thành hội của các tuyển:** vì công thức trên không còn toán tử And, nên ta chỉ bỏ đi các dấu ngoặc là có được mệnh đề như sau:
   $\neg$roman (X) $\vee$ $\neg$know (X,marcus) $\vee$
   hate(X,caesar) $\vee$ $\neg$hate(Y,Z) $\vee$ thinkcrazy(X,Y)

Cơ sở của Hợp giải (Resolution)

Thủ tục hợp giải là một quá trình lặp đơn giản: ở mỗi lần lặp, hai mệnh đề, gọi là mệnh đề cha, được so sánh (hay *giải quyết - resolved*), để tạo ra mệnh đề mới.

Giả sử trong hệ thống có hai mệnh đề:

$winter \vee summer$ và $\neg winter \vee cold$

có thể dẫn xuất thành:

$summer \vee cold$

Nếu mệnh đề kết quả là rỗng thì xem như đã tìm được *sự mâu thuẩn* (contradiction), nghĩa là mục tiêu đã được chứng minh.

Giải thuật hợp giải cho Logic mệnh đề

Cho trước: Tập hợp các tiên đề F là các câu trong phép tính mệnh đề.

Yêu cầu: chứng minh P

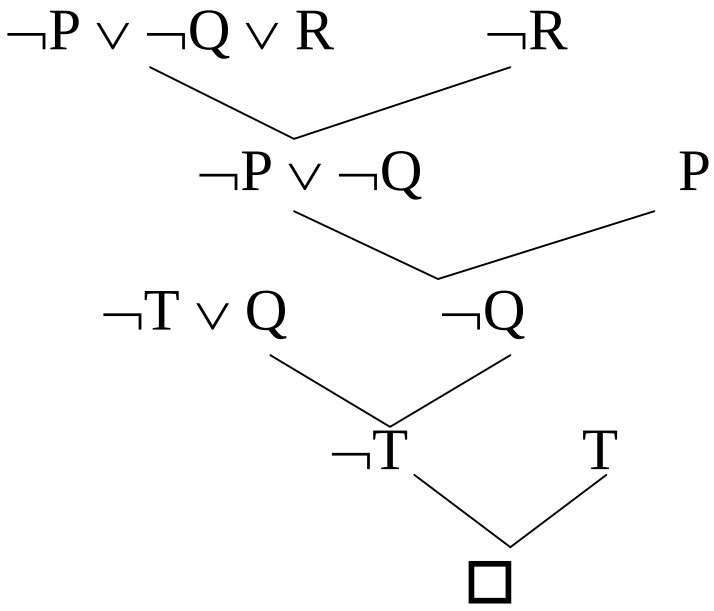
* Giải thuật Hợp giải cho Phép tính mệnh đề (Propositional Logic):

1) Chuyển tất cả các câu trong F về dạng mệnh đề (clause form)
2) Lấy phủ định P và chuyển về dạng mệnh đề. Thêm nó vào tập các mệnh đề vừa tạo ở bước 1.
3) Lặp lại cho đến khi tìm thấy sự mâu thuẩn hoặc không thể tiếp tục:
   a. Chọn hai mệnh đề. Gọi là các mệnh đề cha.
   b. Hợp giải chúng. Mệnh đề kết quả là tuyển của tất cả các biến mệnh đề trong các mệnh đề cha trừ: nếu có bất kỳ các cặp biến mệnh đề L và ¬L, một nằm trong mệnh đề cha này, một nằm trong mệnh đề cha kia, thì chọn một cặp và xóa cả hai L và ¬L ra khỏi mệnh đề kết quả.
   c. Nếu mệnh đề kết quả là rỗng, thì xem như đã tìm được sự mâu thuẩn.
      Nếu không, thêm mệnh đề kết quả đó vào trong tập hợp các mệnh đề hiện có.

Ví dụ hợp giải trong Logic mệnh đề

Các câu cho trước	Chuyển về dạng mệnh đề	
P	P	(1)
$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$	(2)
$(S \vee T) \rightarrow Q$	$\neg S \vee Q$	(3)
	$\neg T \vee Q$	(4)
T	T	(5)

Chứng minh R:



Đồ thị hợp giải (cây hợp giải)

Giải thuật hợp giải cho Logic vị từ

Cho trước: tập hợp các tiên đề F là các câu trong Phép tính vị từ.

Yêu cầu: chứng minh P

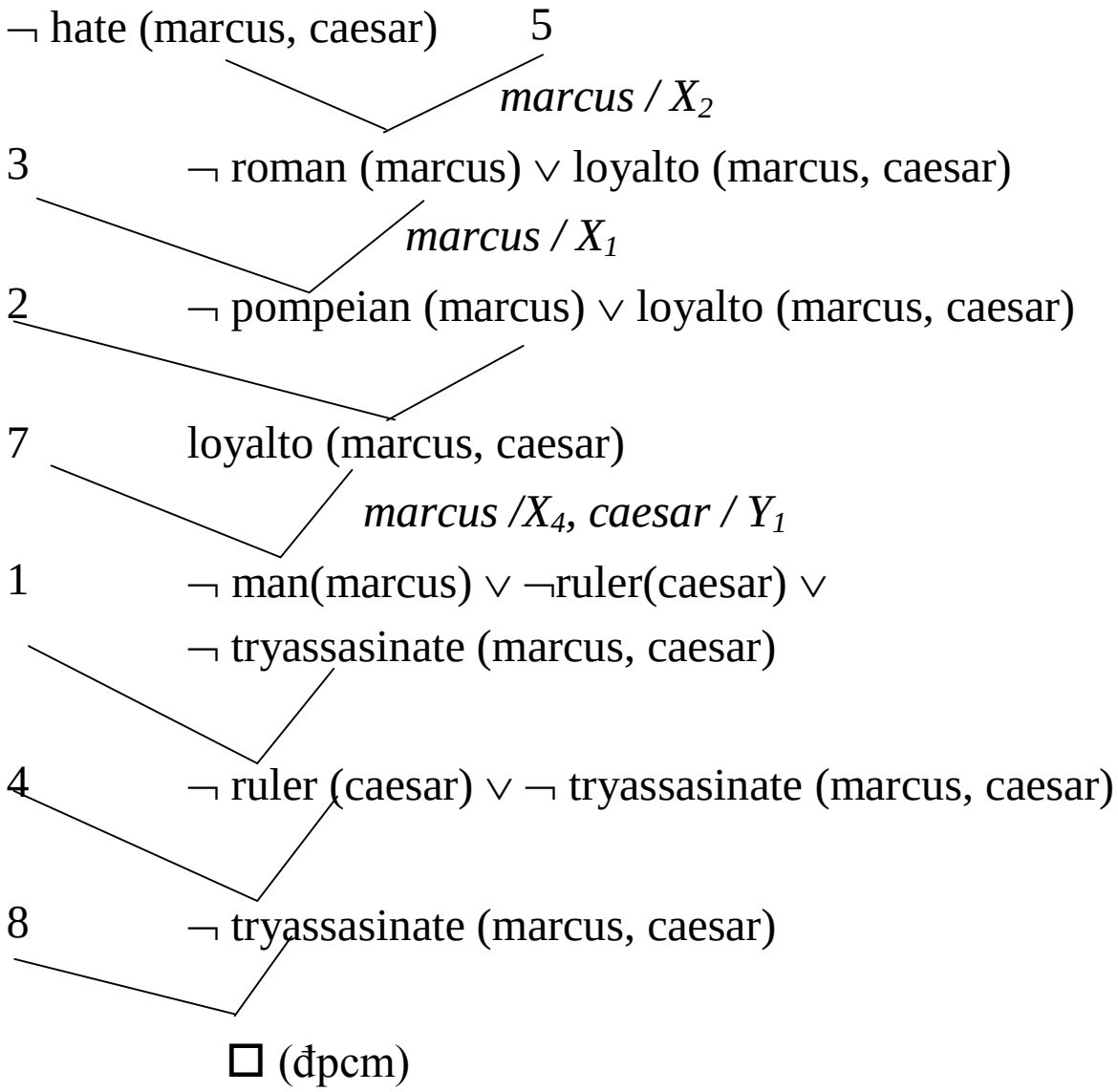
*** Giải thuật Hợp giải dùng cho Phép tính vị từ (Predicate Logic):**

1) Chuyển tất cả các câu trong F về dạng mệnh đề (clause form)
2) Lấy phủ định của P và chuyển về dạng mệnh đề. Thêm nó vào tập các mệnh đề vừa tạo ở bước 1.
3) Lặp lại cho đến khi tìm thấy sự mâu thuẩn hay không thể tiếp tục:
   a. Chọn hai mệnh đề. Gọi là các mệnh đề cha.
   b. Hợp giải chúng. Mệnh đề kết quả là tuyển của tất cả các biến mệnh đề trong các mệnh đề cha với các phép thế phù hợp và trừ đi: nếu có một cặp biến mệnh đề T1 và ¬T2, sao cho T1 nằm trong mệnh đề cha này, còn ¬T2 nằm trong mệnh đề cha kia, và nếu T1 và T2 là hai biến mệnh đề có thể đồng nhất *(unifiable)*, thì xóa cả hai T1 và ¬T2 ra khỏi mệnh đề kết quả. T1 và T2 là các *biến mệnh đề bù nhau* (complementary literals).
   Sử dụng tập phép thế trả ra bởi giải thuật đồng nhất để tạo ra mệnh đề kết quả.
   c. Nếu mệnh đề kết quả là rỗng, thì xem như đã tìm được sự mâu thuẩn. Nếu không, thêm mệnh đề kết quả đó vào trong tập hợp các mệnh đề hiện có.

Ví dụ hợp giải trong Logic Vị từ

1. man (marcus)
2. pompeian (marcus)
3. ¬pompeian (X_1) ∨ Roman (X_1)
4. ruler (caesar)
5. ¬roman (X_2) ∨ loyalto (X_2, caesar) ∨ hate (X_2, caesar)
6. loyato (X_3, fl(X_3))
7. ¬man (X_4) ∨ ¬ ruler (Y_1) ∨ ¬ tryassasinate (X_4, Y_1) ∨ loyalto (X_4, Y_1)
8. tryassasinate (marcus, caesar)

Chứng minh: hate (marcus, caesar)



□ (đpcm)

Một số chiến lược (heuristic) hỗ trợ cho việc lựa chọn mệnh đề

- Chỉ hợp giải những cặp mệnh đề có chứa các biến *mệnh đề bù nhau*.
- Loại bỏ các mệnh đề ngay khi chúng vừa được tạo ra trong quá trình hợp giải: - mệnh đề luôn luôn đúng (tautology),
  - mệnh đề được tạo thành từ các mệnh đề khác
  (ví dụ: $P \vee Q$ được tạo thành từ P)
- Chiến lược ***set-of-support***: hợp giải với một trong những mệnh đề là một phần của câu mà ta cần phản chứng hoặc với một mệnh đề được sinh ra do hợp giải với mệnh đề như vậy.
- Chiến lược ***unit-preference***: hợp giải với mệnh đề chỉ có một biến mệnh đề.

Phát hiện các trường hợp không tồn tại sự mâu thuẩn

Với câu hỏi ‘Did Marcus hate Caesar?’.

⇨ Câu hỏi: ¬ hate (marcus, caesar).

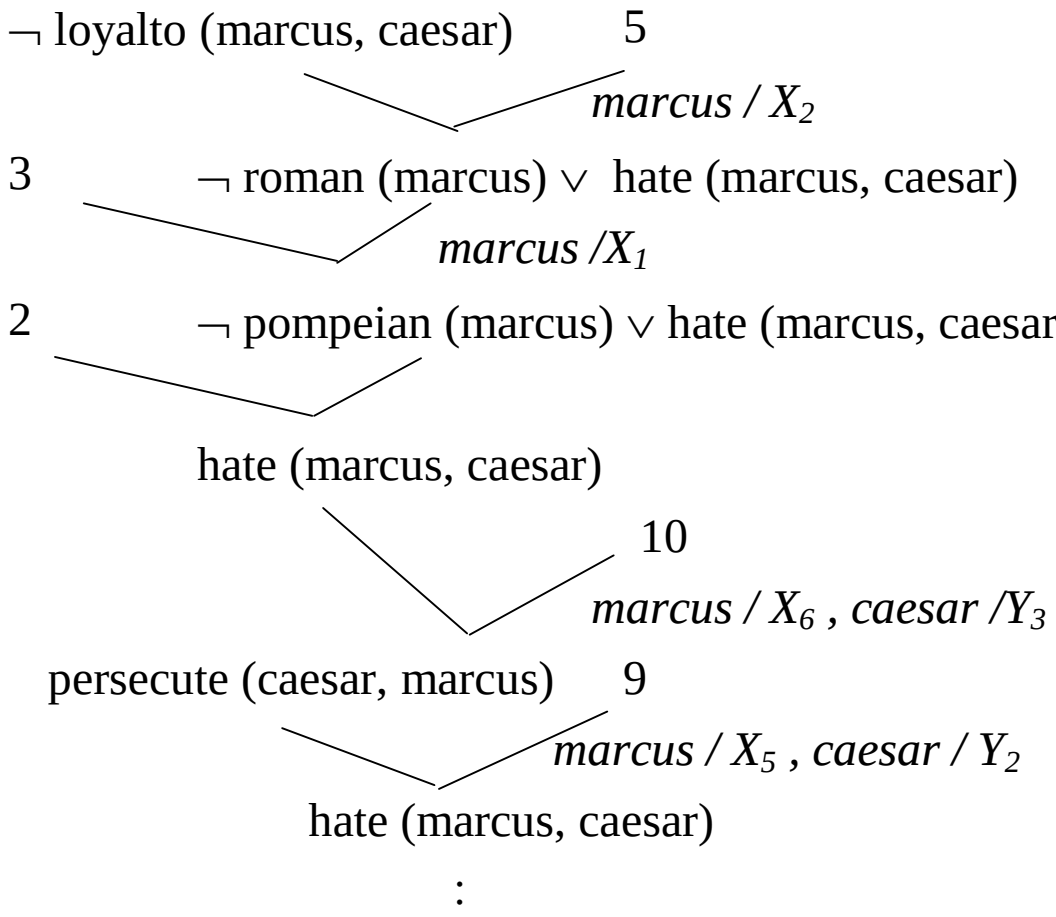
⇨ Phủ định: hate (marcus, caesar).

☹ Không có mệnh đề nào chứa biến mệnh đề ¬hate!!!

<u>Hoặc:</u> Giả sử ta có thêm hai câu:

9. persecute(X,Y) → hate(Y,X) ¬persecute(X_5,Y_2) ∨ hate(Y_2,X_5)
10. hate(X,Y) → persecute(Y,X) ¬hate(X_6,Y_3) ∨ persecute(Y_3,X_6)

Chứng minh: loyalto (marcus, caesar)

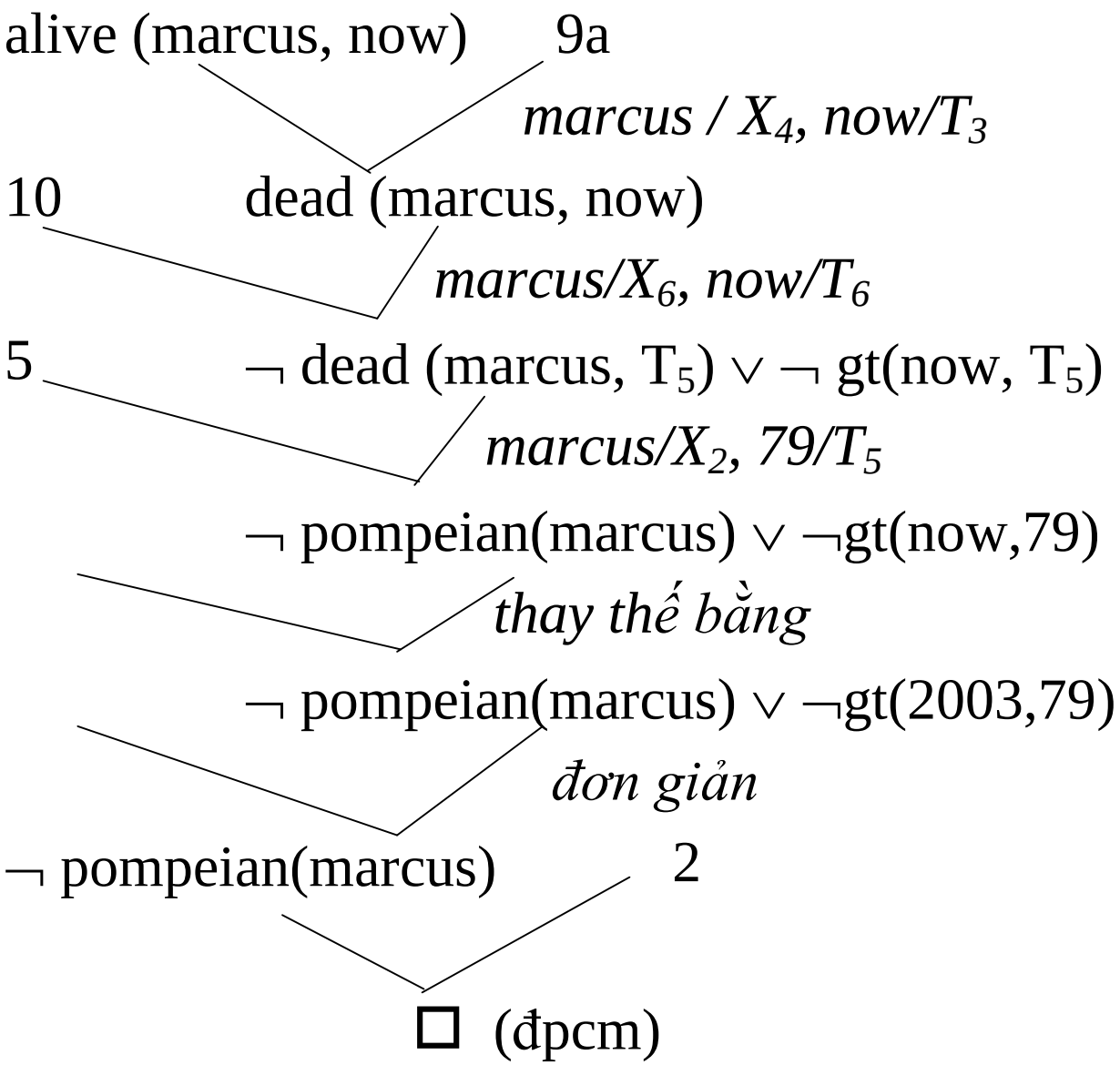


Sử dụng hàm tính toán, vị từ tính toán, và mối quan hệ bằng

Ví dụ 6: Giả sử ta có cơ sở tri thức đã được chuyển sang dạng mệnh đề như sau:

1. man (marcus)
2. pompeian (marcus)
3. born(marcus, 40)
4. $\neg man(X_1) \vee mortal\ (X_1)$
5. $\neg pompeian\ (X_2) \vee died\ (X_2,79)$
6. erupted (volcano,79)
7. $\neg mortal\ (X_3) \vee \neg born\ (X_3,\ T_1) \vee \neg gt(T_2 - T_1,150) \vee dead(X_3,T_2)$
8. now = 2003
9.
   a. $\neg alive(X_4,T_3) \vee \neg dead(X_4,T_3)$
   b. $dead(X_5,T_4) \vee alive(X_5,T_4)$
10. $\neg died\ (X_6,\ T_5) \vee \neg gt(T_6,\ T_5) \vee dead(X_6,T_6)$

Chứng minh: $\neg$ alive (marcus, now)


alive (marcus, now)
9a
marcus / X4, now/T3
10
dead (marcus, now)
marcus/X6, now/T6
5
¬ dead (marcus, T5) ∨ ¬ gt(now, T5)
marcus/X2, 79/T5
¬ pompeian(marcus) ∨ ¬gt(now,79)
thay thế bằng
¬ pompeian(marcus) ∨ ¬gt(2003,79)
đơn giản
¬ pompeian(marcus)
2
□ (đpcm)

Trả lời câu hỏi

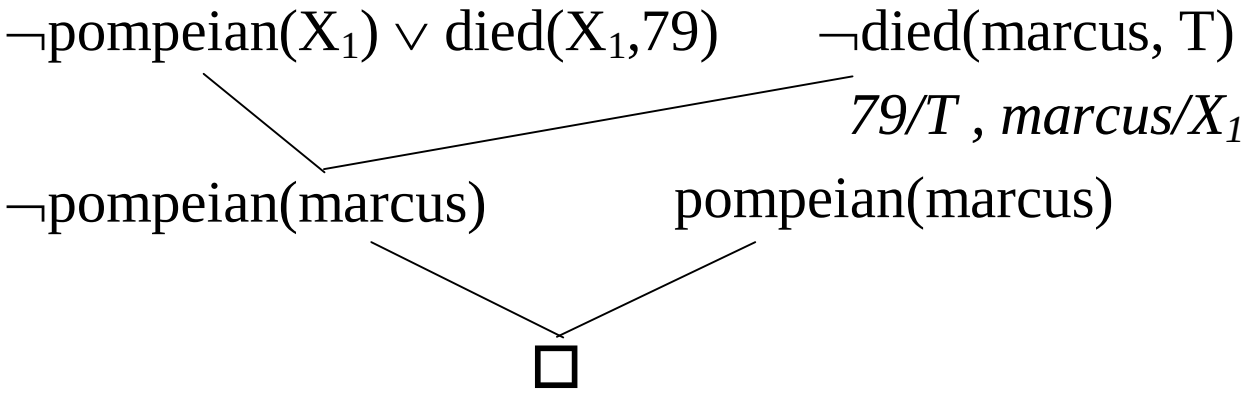
Câu hỏi dạng điền vào chổ trống:

"Marcus đã chết khi nào?' => *died(marcus, ??)*

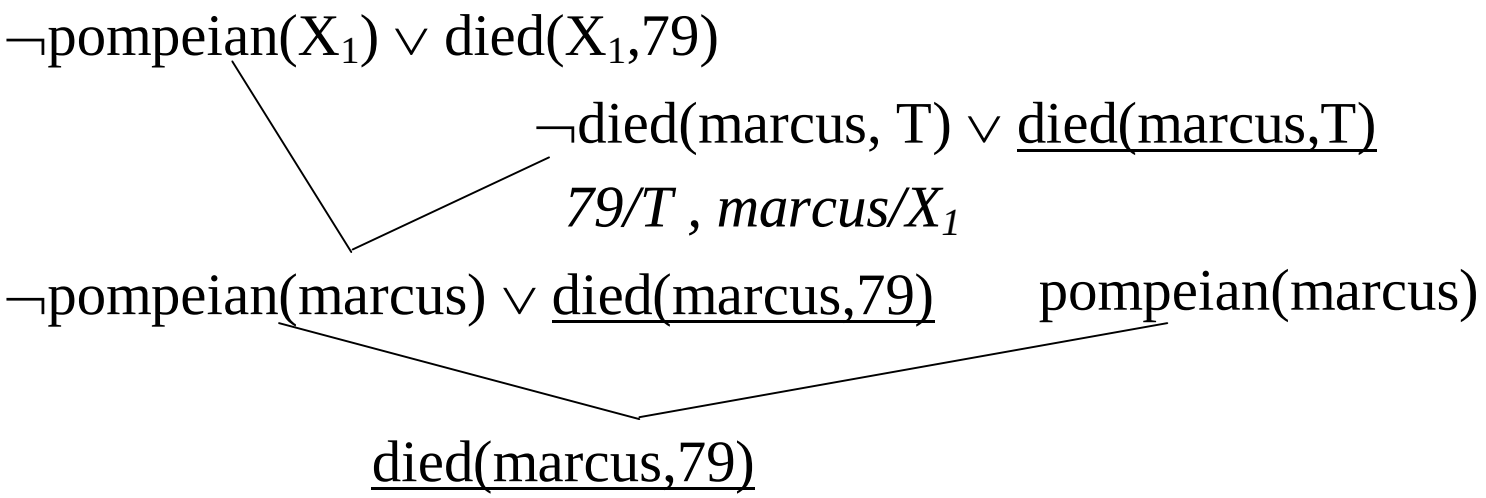
Câu hỏi: $\exists T$ *died(marcus, T)*

Phủ định: $\neg\exists T$ *died(marcus, T)*

Chuyển về dạng mệnh đề: $\neg$died(marcus,T)



Cây hợp giải



Cây chứng minh cải biên.

Nhận xét về Hợp giải

Ưu điểm: Tính tổng quát, tính phổ dụng, khả năng áp dụng tốt đối với các dạng câu Horn $A_1 \wedge \ \ldots \ A_n \rightarrow B$.

Nhược điểm:

- Con người không sử dụng chiến lược suy diễn theo kiểu hợp giải. Vì vậy một người khó có thể giao tiếp với chương trình chứng minh sử dụng phương pháp hợp giải, để có thể cho nó một lời khuyên hay nhận lời khuyên từ nó.
- Trong khi chuyển chuyển các câu về dạng mệnh đề, chúng ta đã đánh mất các thông tin kinh nghiệm có giá trị chứa trong các câu ban đầu.

Ví dụ: Với tri thức nhận định một người là có giáo dục tốt nếu am hiểu và không quanh co (thành thật):

$$\forall X \ judge(X) \wedge \neg crooked(X) \rightarrow educated(X)$$

Chuyển về dạng mệnh đề:

$$\neg judge(X) \vee crooked(X) \ \vee educated(X)$$

Câu này cũng có thể hiểu là dùng để xác định một người là am hiểu nếu như anh ta không quanh co và không được giáo dục tốt!

Chương 9
Học Máy

Giáo viên: Trần Ngân Bình

Học Máy (Machine Learning)

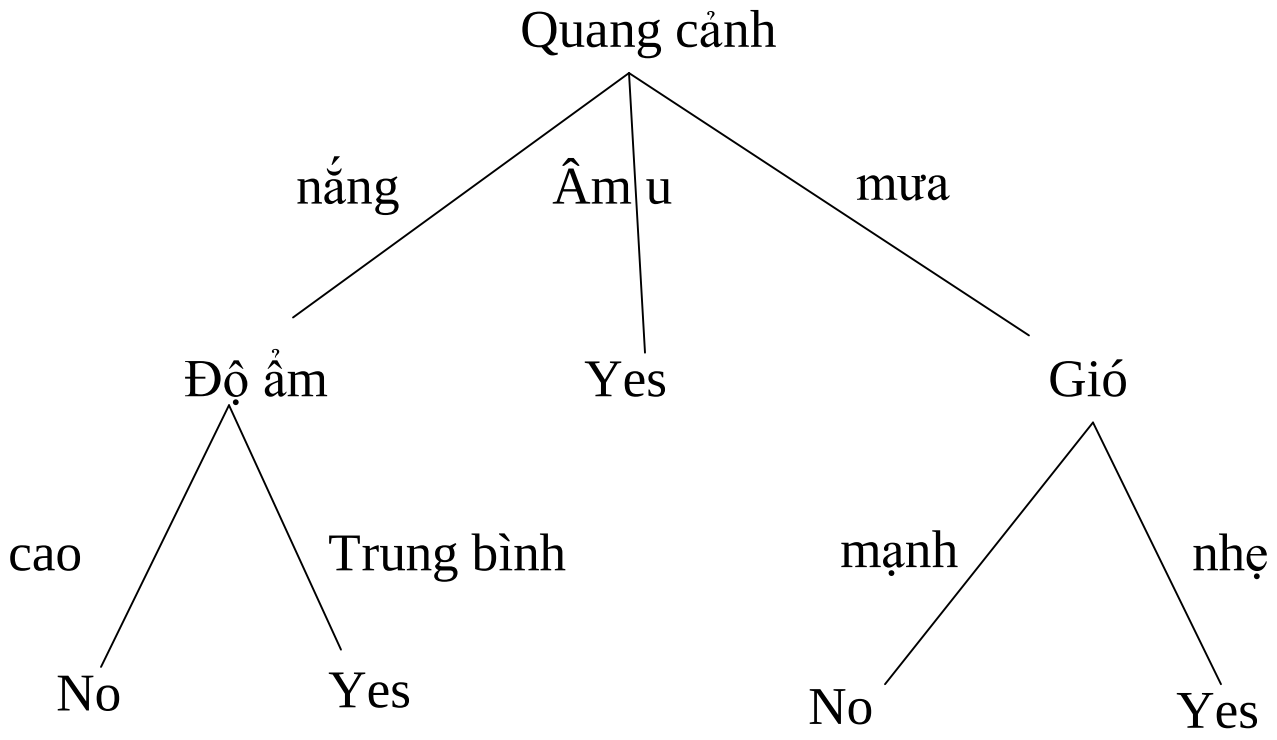
- Học (learning) là bất cứ sự thay đổi nào trong một hệ thống cho phép nó tiến hành tốt hơn trong lần thứ hai khi lặp lại cùng một nhiệm vụ hoặc với nhiệm vụ khác từ cùng một quần thể đó. (Herbert Simon)
- Học liên quan đến vấn đề khái quát hóa từ kinh nghiệm (dữ liệu rèn luyện) => bài toán *quy nạp* (induction)
- Vì dữ liệu rèn luyện thường hạn chế, nên thường khái quát hóa theo một số khía cạnh nào đó (heuristic) => tính *thiên lệch quy nạp* (inductive bias)
- Có ba tiếp cận học:
  - Các phương pháp học dựa trên ký hiệu (symbol-based): ID3
  - Tiếp cận kết nối: Các mạng neuron sinh học
  - Tiếp cận di truyền hay tiến hóa: giải thuật genetic

Cây quyết định (ID3)

- Là một giải thuật học đơn giản nhưng thành công
- Cây quyết định (QĐ) là một cách biểu diễn cho phép chúng ta xác định phân loại của một đối tượng bằng cách kiểm tra giá trị của một số thuộc tính.
- Giải thuật có:
  - **Đầu vào:** Một đối tượng hay một tập hợp các thuộc tính mô tả một tình huống
  - **Đầu ra:** thường là quyết định yes/no, hoặc các phân loại.
- Trong cây quyết định:
  - Mỗi nút trong biểu diễn một sự kiểm tra trên một thuộc tính nào đó, mỗi giá trị có thể của nó tương đương với một nhánh của cây
  - Các nút lá thể hiện sự phân loại.
- Kích cỡ của cây QĐ tùy thuộc vào thứ tự của các kiểm tra trên các thuộc tính.

Ví dụ Cây QĐ: Chơi Tennis

- Mục đích: học để xem có chơi Tennis không?
- Cây quyết định:



Quy nạp cây QĐ từ các ví dụ

- Ví dụ (hay dữ liệu rèn luyện cho hệ thống) gồm:

  Giá trị của các thuộc tính + Phân loại của ví dụ

Ngày	Quang cảnh	Nhiệt độ	Độ ẩm	Gió	Chơi Tennis
D1	Nắng	Nóng	Cao	nhẹ	Không
D2	Nắng	Nóng	Cao	Mạnh	Không
D3	Âm u	Nóng	Cao	Nhẹ	Có
D4	Mưa	ấm áp	Cao	nhẹ	Có
D5	Mưa	Mát	TB	nhẹ	Có
D6	Mưa	Mát	TB	Mạnh	Không
D7	Âm u	Mát	TB	Mạnh	Có
D8	Nắng	ấm áp	Cao	nhẹ	Không
D9	Nắng	Mát	TB	nhẹ	Có
D10	Mưa	ấm áp	TB	nhẹ	Có
D11	Nắng	ấm áp	TB	Mạnh	Có
D12	Âm u	ấm áp	Cao	Mạnh	Có
D13	Âm u	Nóng	TB	nhẹ	Có
D14	Mưa	ấm áp	Cao	Mạnh	không

Làm sao để học được cây QĐ

- Tiếp cận đơn giản
  - Học một cây mà có một lá cho mỗi ví dụ.
  - Học thuộc lòng một cách hoàn toàn các ví dụ.
  - Có thể sẽ không thực hiện tốt trong các trường hợp khác.
- Tiếp cận tốt hơn:
  - Học một cây nhỏ nhưng chính xác phù hợp với các ví dụ
  - Occam's razor – cái đơn giản thường là cái tốt nhất!

    Giả thuyết có khả năng nhất là giả thuyết đơn giản nhất thống nhất với tất cả các quan sát.

Xây dựng cây QĐ: Trên - xuống

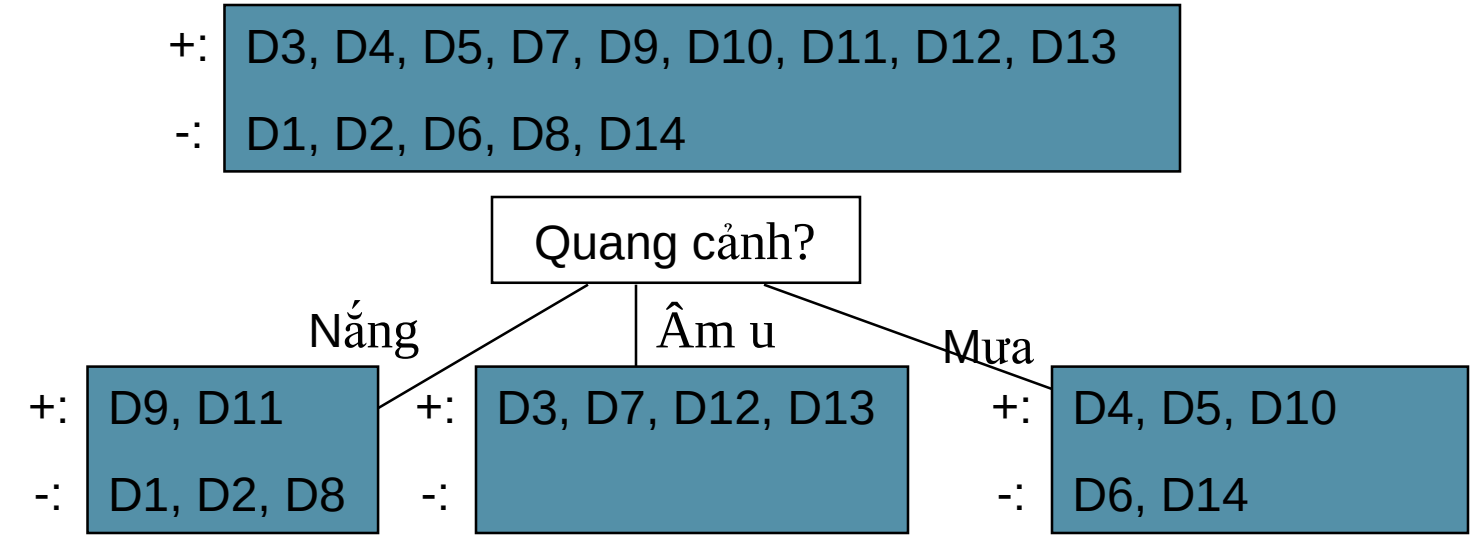
Vòng lặp chính:

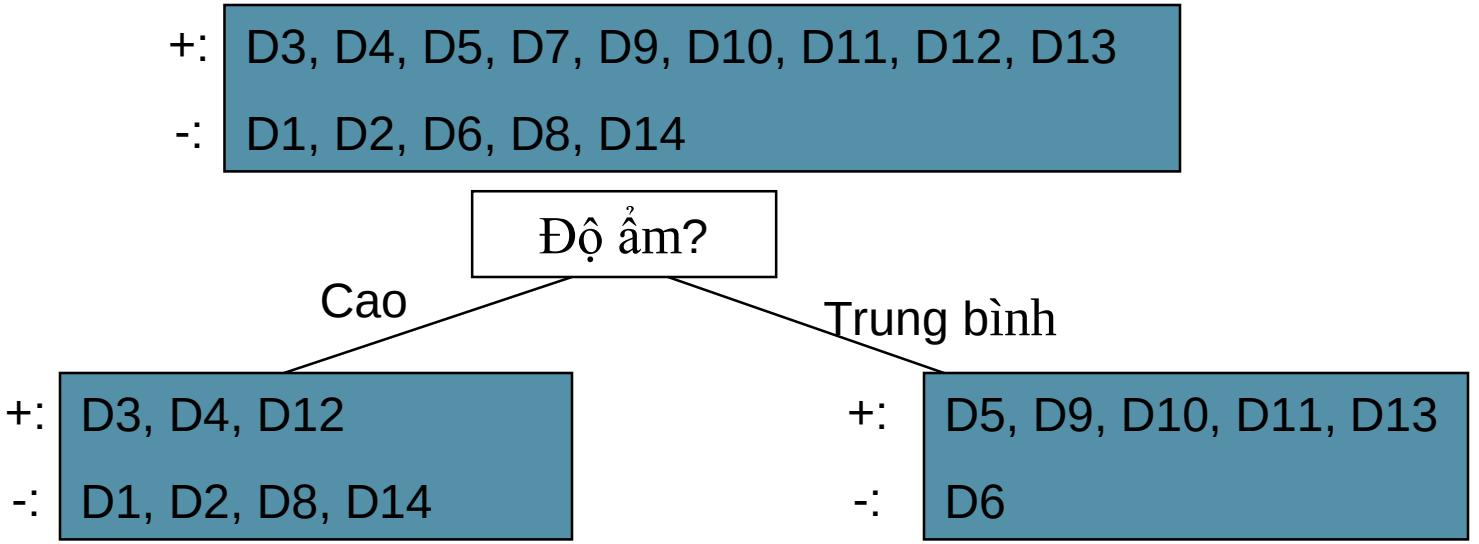
1. A <- thuộc tính quyết định tốt nhất cho nút kế
2. Gán A là thuộc tính quyết định cho nút
3. Với mỗi giá trị của A, tạo một nút con mới cho nút
4. Sắp xếp các ví dụ vào các nút lá
5. If các ví dụ đã được phân loại đúng, dừng ctr; Else lặp lại trên mỗi nút lá mới

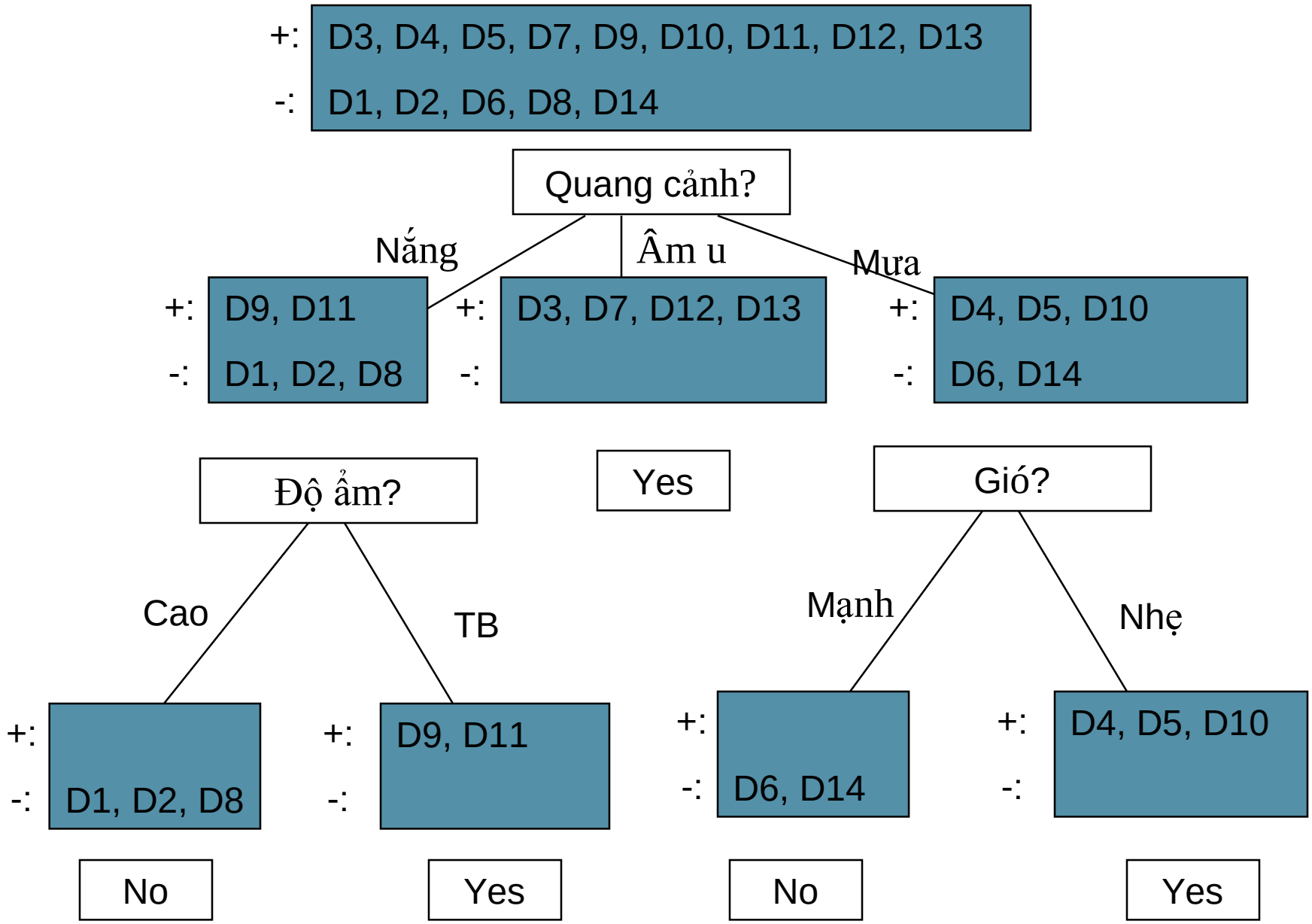
Để phân loại một trường hợp, có khi cây QĐ không cần sử dụng tất cả các thuộc tính đã cho, mặc dù nó vẫn phân loại đúng tất cả các ví dụ.

Các khả năng có thể của nút con

- Các ví dụ có cả âm và dương:
  - Tách một lần nữa
- Tất cả các ví dụ còn lại đều âm hoặc đều dương
  - trả về cây quyết định
- Không còn ví dụ nào
  - trả về mặc nhiên
- Không còn thuộc tính nào (nhiễu)
  - Quyết định dựa trên một luật nào đó (luật đa số)


+: D3, D4, D5, D7, D9, D10, D11, D12, D13
-: D1, D2, D6, D8, D14
Quang cảnh?
Nắng
Âm u
Mưa
+: D9, D11
-: D1, D2, D8
+: D3, D7, D12, D13
-:
+: D4, D5, D10
-: D6, D14


+: D3, D4, D5, D7, D9, D10, D11, D12, D13
-: D1, D2, D6, D8, D14
Độ ẩm?
Cao
Trung bình
+: D3, D4, D12
-: D1, D2, D8, D14
+: D5, D9, D10, D11, D13
-: D6


+: D3, D4, D5, D7, D9, D10, D11, D12, D13
-: D1, D2, D6, D8, D14
Quang cảnh?
Nắng
Âm u
Mưa
+: D9, D11
-: D1, D2, D8
+: D3, D7, D12, D13
-:
+: D4, D5, D10
-: D6, D14
Độ ẩm?
Yes
Gió?
Cao
TB
Mạnh
Nhẹ
+:
-: D1, D2, D8
+: D9, D11
-:
+:
-: D6, D14
+: D4, D5, D10
-:
No
Yes
No
Yes

ID3 xây dựng cây QĐ theo giải thuật sau:

```
function induce_tree (example_set, Properties)

begin
if all entries in example_set are in the same class
   then return a leaf node labeled with that class
   else if Properties is empty
      then return leaf node labeled with disjunction of all classes in example_set
      else begin
         select a property, P, and make it the root of the current tree;
         delete P from Properties;
            for each value, V, of P,
               begin
                  create a branch of the tree labeled with V;
                  let partition_V be elements of example_set with values V for property P;
                  call induce_tree(partition_V, Properties), attach result to branch V
               end
      end
end
```

Đánh giá hiệu suất

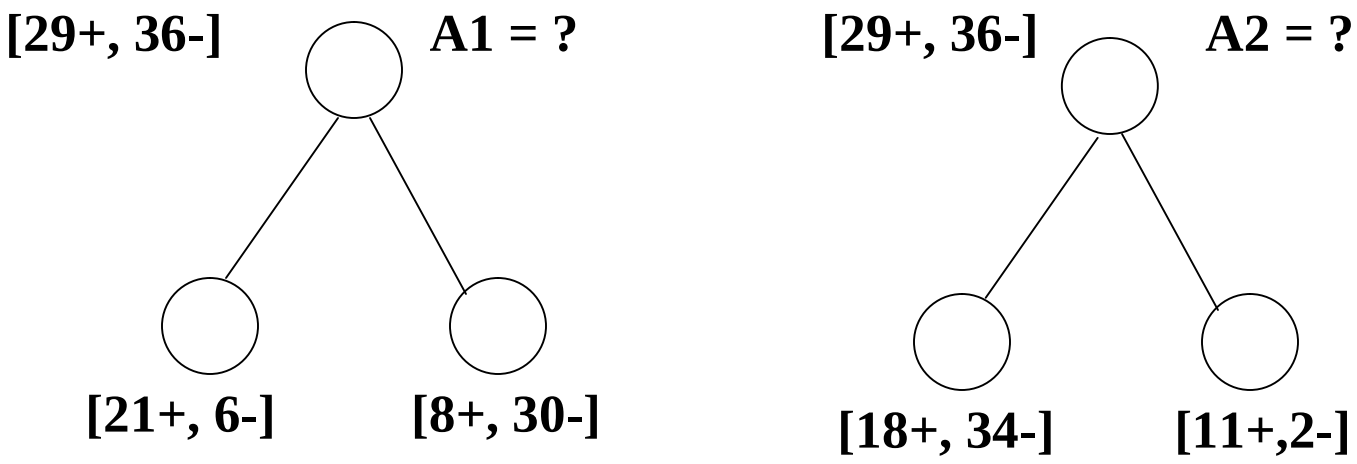
- Chúng ta muốn có một cây QĐ có thể phân loại đúng một ví dụ mà nó chưa từng thấy qua.
- Việc học sử dụng một "tập rèn luyện" (traning set), và
- Việc đánh giá hiệu suất sử dụng một "tập kiểm tra" (test set):

1. Thu thập một tập hợp lớn các ví dụ
2. Chia thành tập rèn luyện và tập kiểm tra
3. Sử dụng giải thuật và tập rèn luyện để xây dựng giả thuyết h (cây QĐ)
4. Đo phần trăm tập kiểm tra được phân loại đúng bởi h
5. Lặp lại bước 1 đến 4 cho các kích cỡ tập kiểm tra khác nhau được chọn một cách nhẫu nhiên.

Sử dụng lý thuyết thông tin

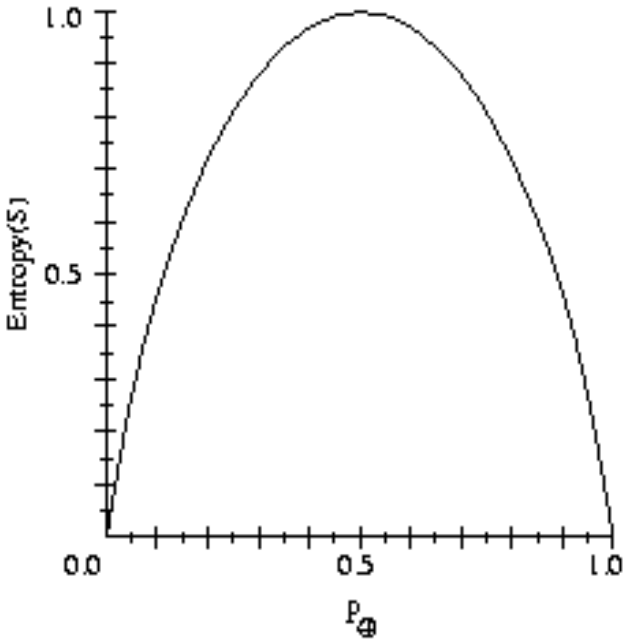
- Chúng ta muốn chọn các thuộc tính có thể giảm thiểu chiều sâu của cây QĐ.
- Thuộc tính tốt nhất: chia các ví dụ vào các tập hợp chứa toàn ví dụ âm hoặc ví dụ dương.
- Chúng ta cần một phép đo để xác định thuộc tính nào cho khả năng chia tốt hơn.

Thuộc tính nào tốt hơn?



Entropy

- Entropy(S) = số lượng mong đợi các bit cần thiết để mã hóa một lớp (+ hay –) của một thành viên rút ra một cách ngẫu nhiên từ S (trong trường hợp tối ưu, mã có độ dài ngắn nhất).
- Theo lý thuyết thông tin: mã có độ dài tối ưu là mã gán $-\log_2 p$ bits cho thông điệp có xác suất là p.



- S là một tập rèn luyện
- $p_\oplus$ là phần các ví dụ dương trong tập S
- p_Θ là phần các ví dụ âm trong tập S
- Entropy đo độ pha trộn của tập S:

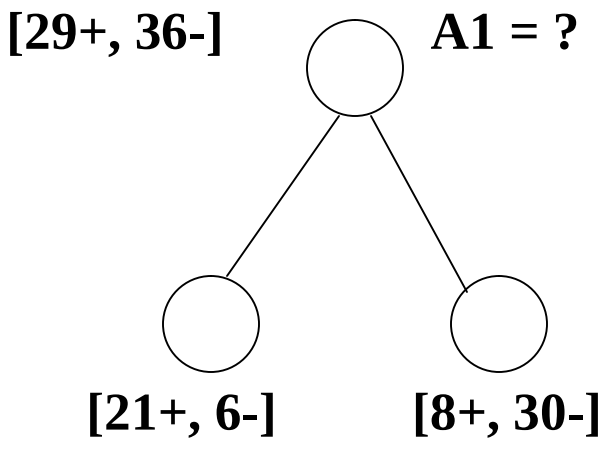
$$Entropy(S) = -p_\oplus \log_2 p_\oplus - p_\Theta \log_2 p_\Theta$$

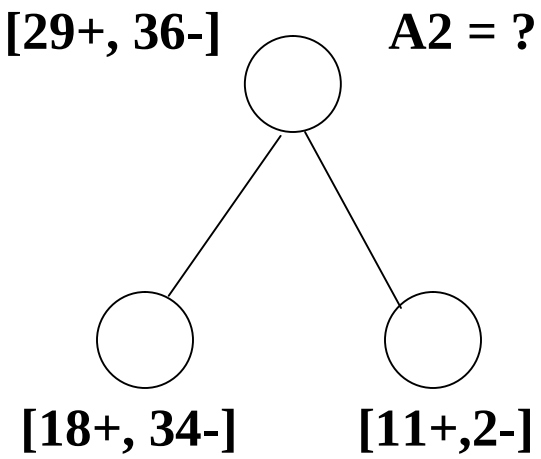
$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Lượng thông tin thu được
Information Gain

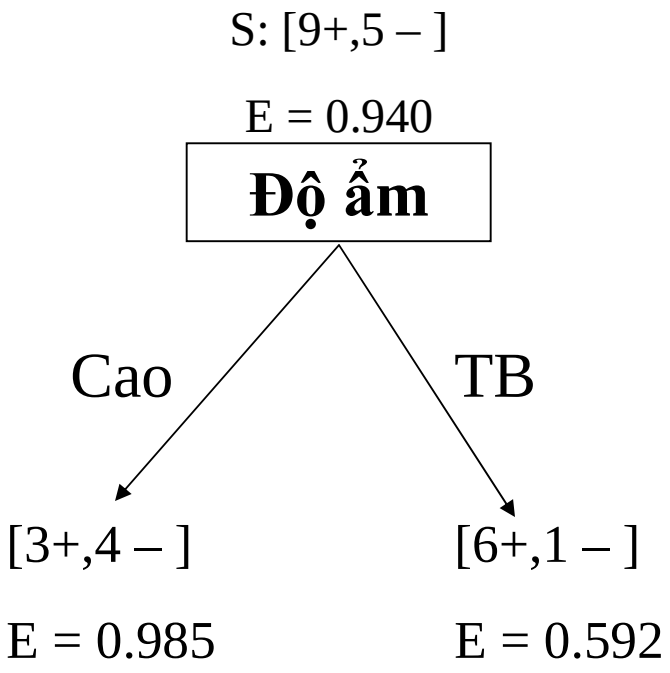
- Gain(S, A) = Lượng giảm entropy mong đợi qua việc chia các ví dụ theo thuộc tính A

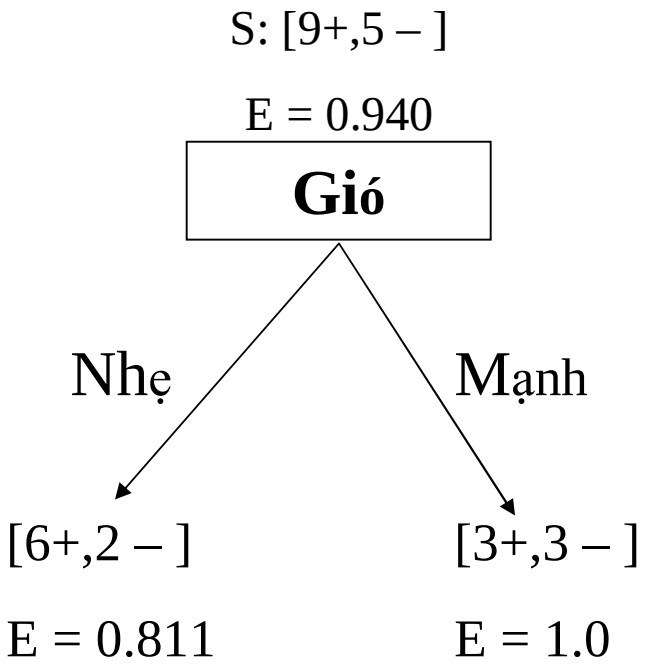
$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$





Chọn thuộc tính kế tiếp

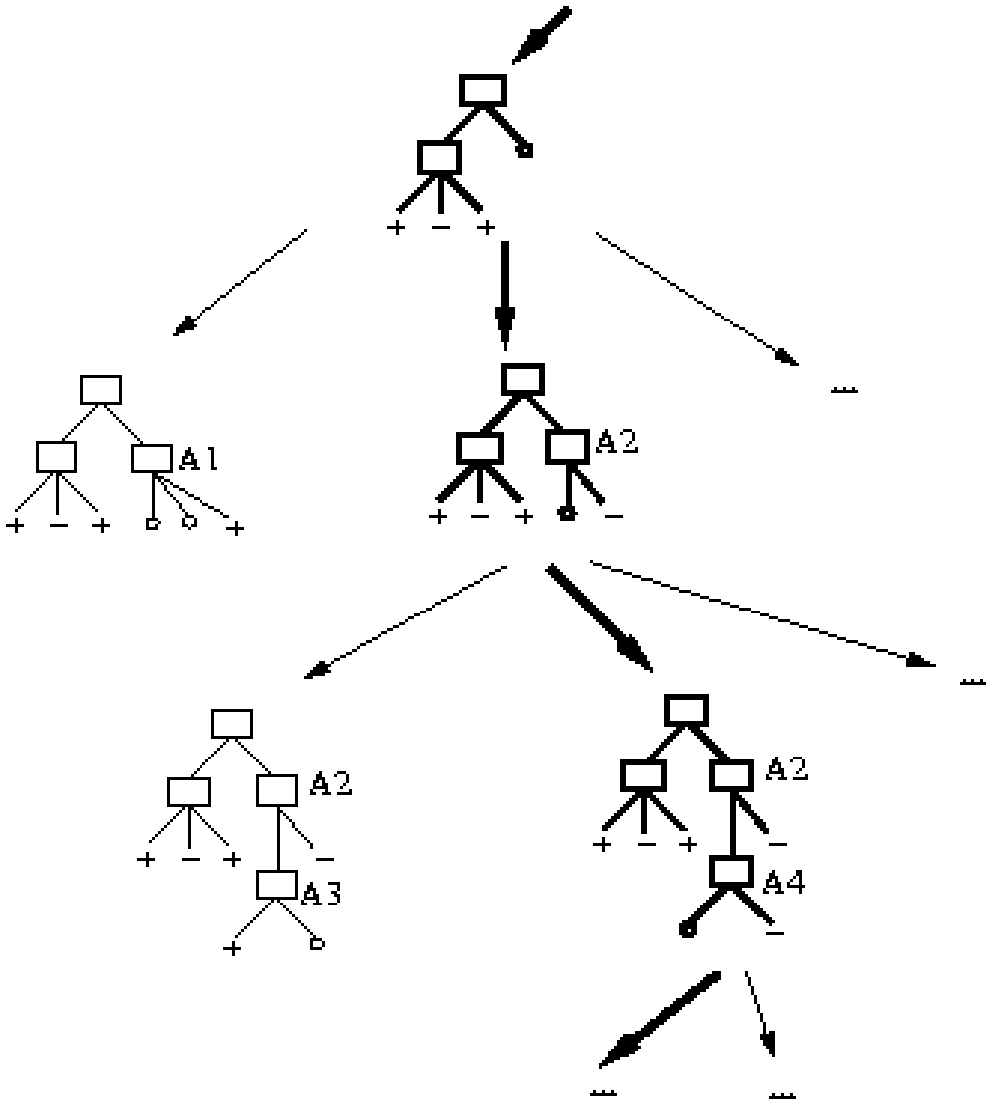




Gain(S, Độ ẩm)
= .940 – (7/14).985 – (7/14).592
= .151

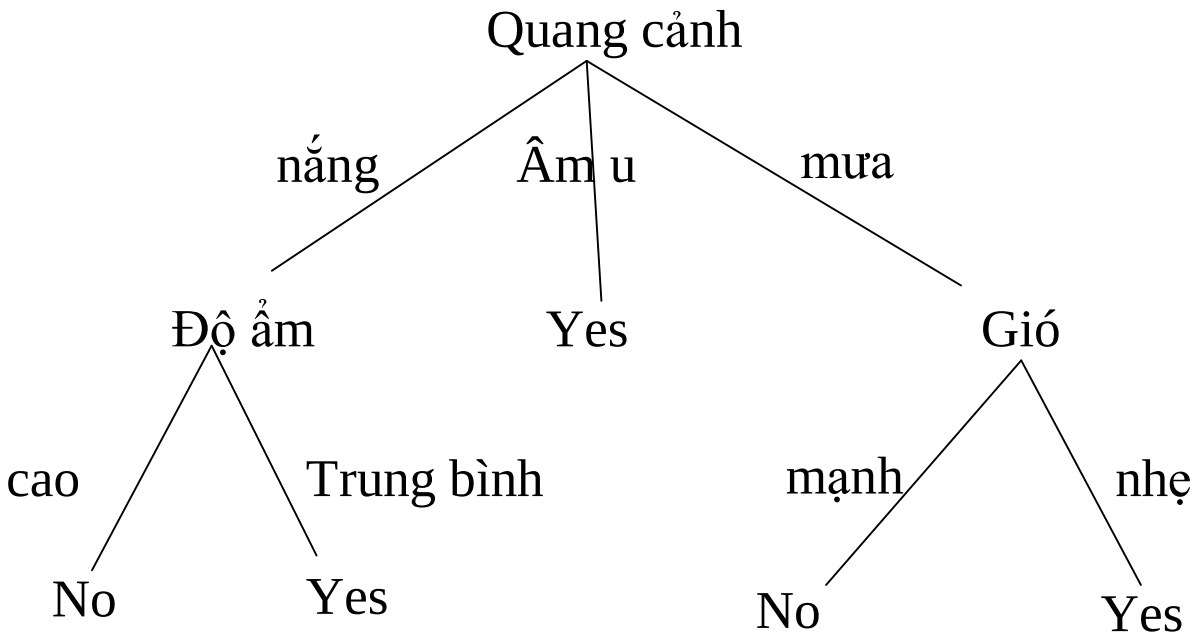
Gain(S, Gió)
= .940 – (8/14).811 – (6/14)1.0
= .048

Tìm kiếm KG giả thuyết trong ID3 (1)



- KG giả thuyết đầy đủ =>giả thuyết chắc chắn thuộc KG này
- Đầu ra là một giả thuyết (cây QĐ) =>Cây nào? Không thể chọn cây với 20 câu hỏi
- Không quay lui => cực tiểu địa phương
- Lựa chọn tìm kiếm dựa trên thống kê => chịu được dữ liệu nhiễu
- Thiên lệch quy nạp: thích cây ngắn hơn.

Chuyển cây về thành các luật



If (Quang-cảnh =nắng) $\wedge$ (Độ ẩm = Cao) Then Chơi-Tennis = No

If (Quang-cảnh =nắng) $\wedge$ (Độ ẩm = TB) Then Chơi-Tennis = Yes

If (Quang-cảnh =Âm u) Then Chơi-Tennis = Yes

…

Khi nào nên sử dụng cây QĐ

- Các ví dụ được mô tả bằng các cặp "thuộc tính – giá trị", vd: Gió - mạnh, Gió - nhẹ
- Kết quả phân loại là các giá trị rời rạc, vd: Yes, No
- Dữ liệu rèn luyện có thể chứa lỗi (bị nhiễu)
- Dữ liệu rèn luyện có thể thiếu giá trị thuộc tính

Ví dụ:

- Phân loại bệnh nhân theo các bệnh của họ
- Phân loại hỏng hóc thiết bị theo nguyên nhân
- Phân loại người vay tiền theo khả năng chi trả

Ví dụ: ước lượng độ an toàn của một tài khoản tín dụng

Table 13.1: **Data from credit history of loan applications.**

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

Figure 13.13: **Một cây QĐ cho bài toán đánh giá độ an toàn của tín dụng.**

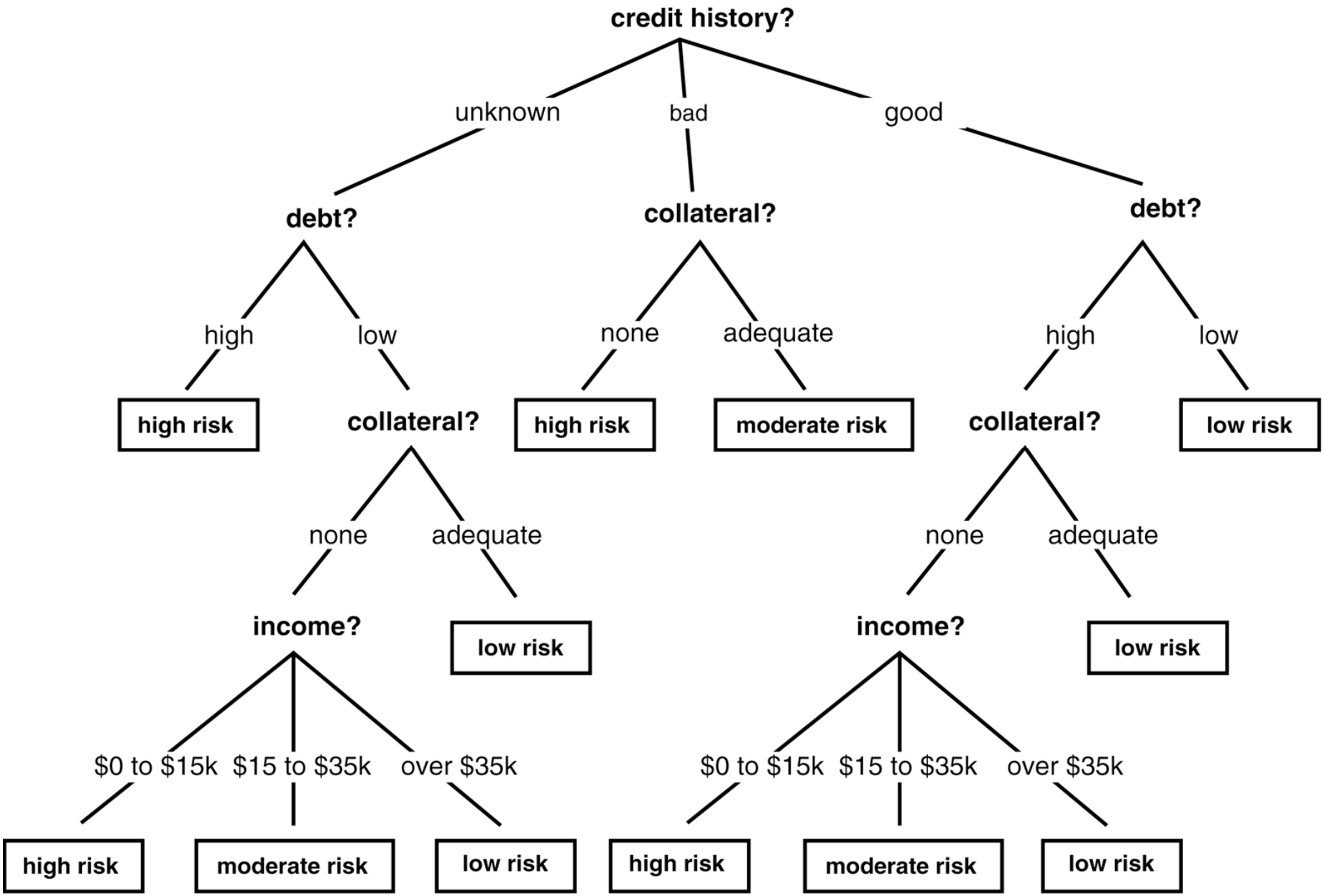


Figure 13.14: **Một cây QĐ đơn giản hơn.**

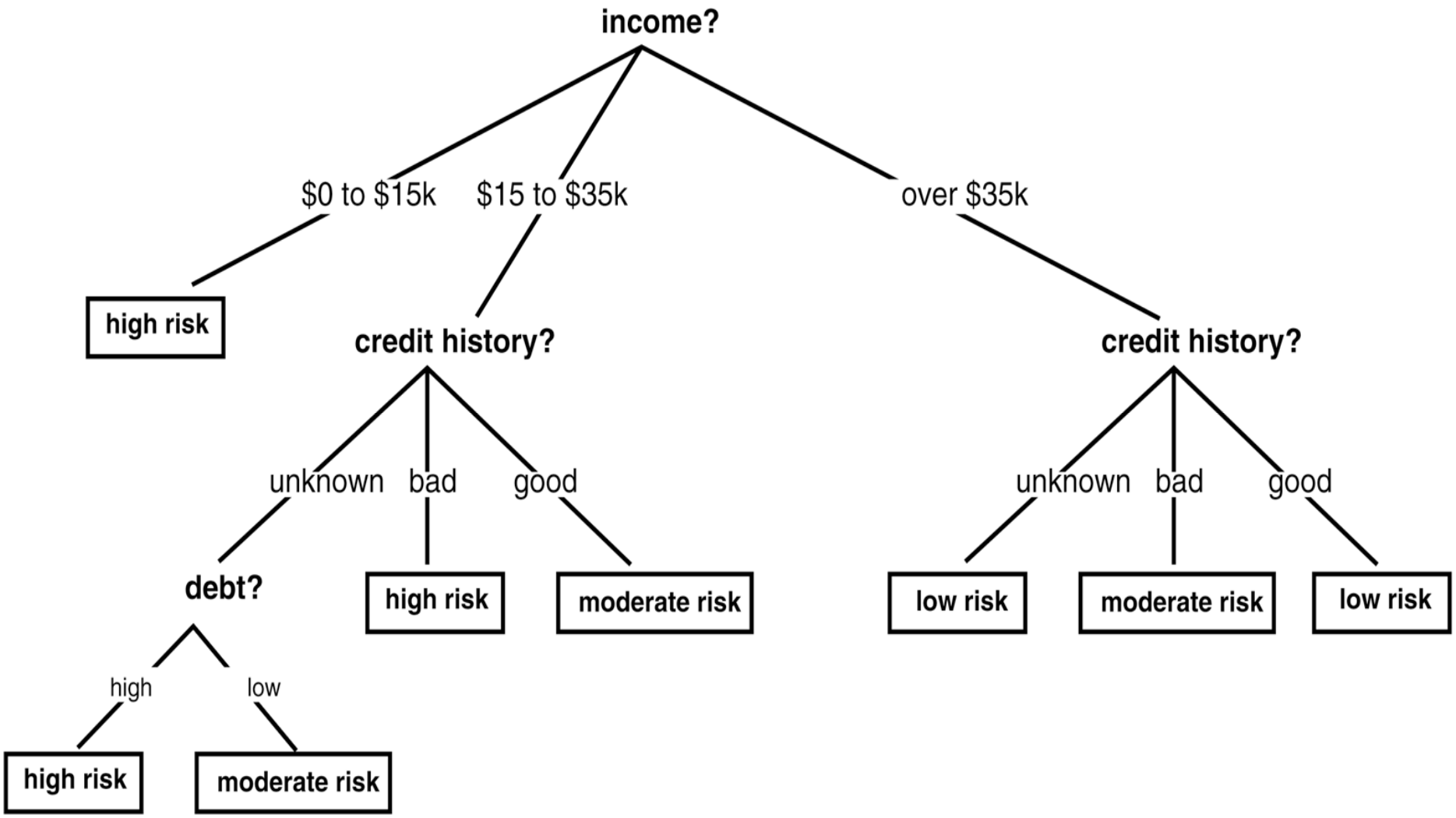


Figure 13.15: **Một cây QĐ đang xây dựng.**

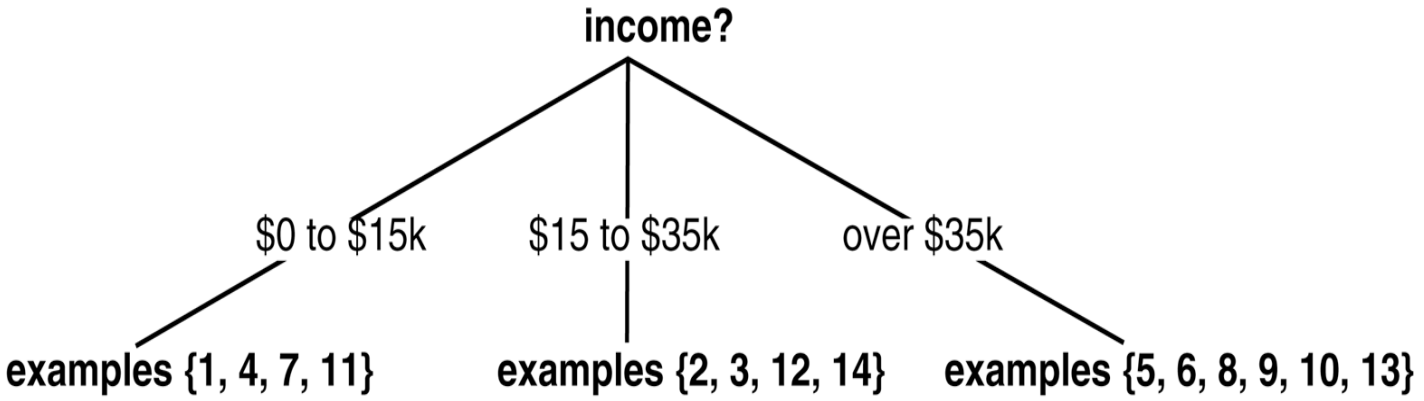
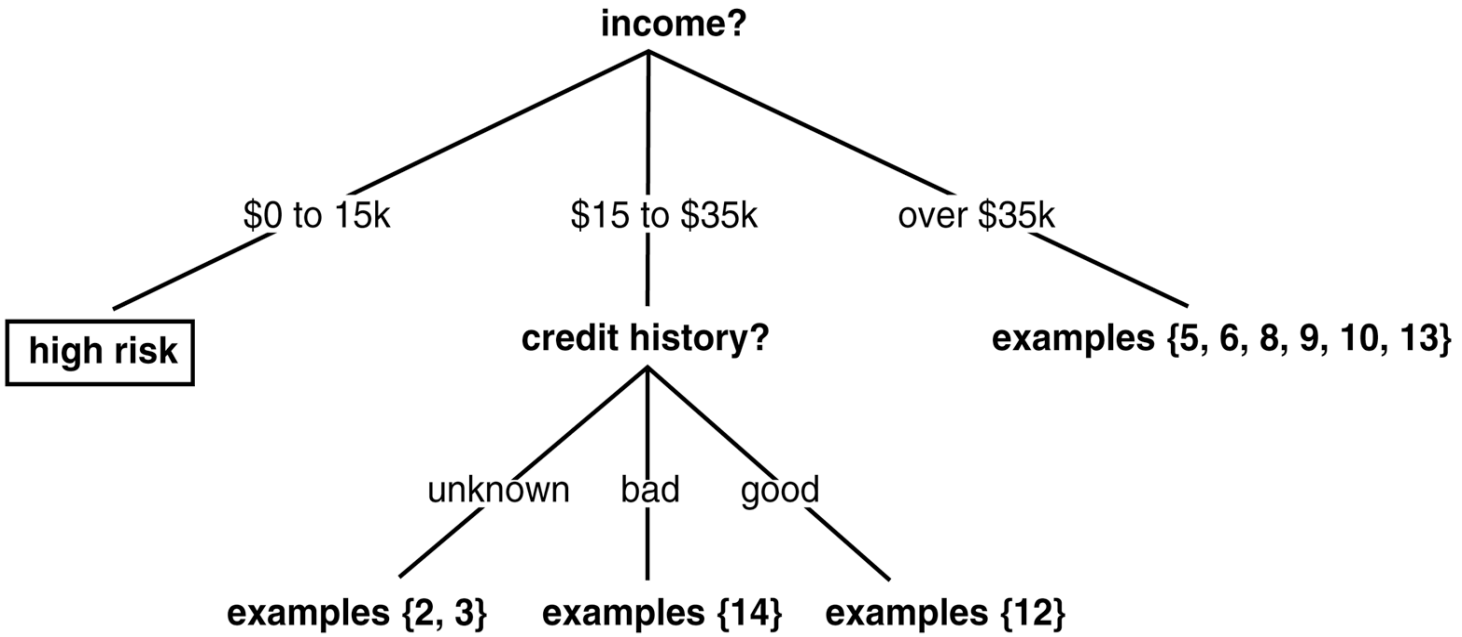


Figure 13.16: **Một cây QĐ khác đang xây dựng.**

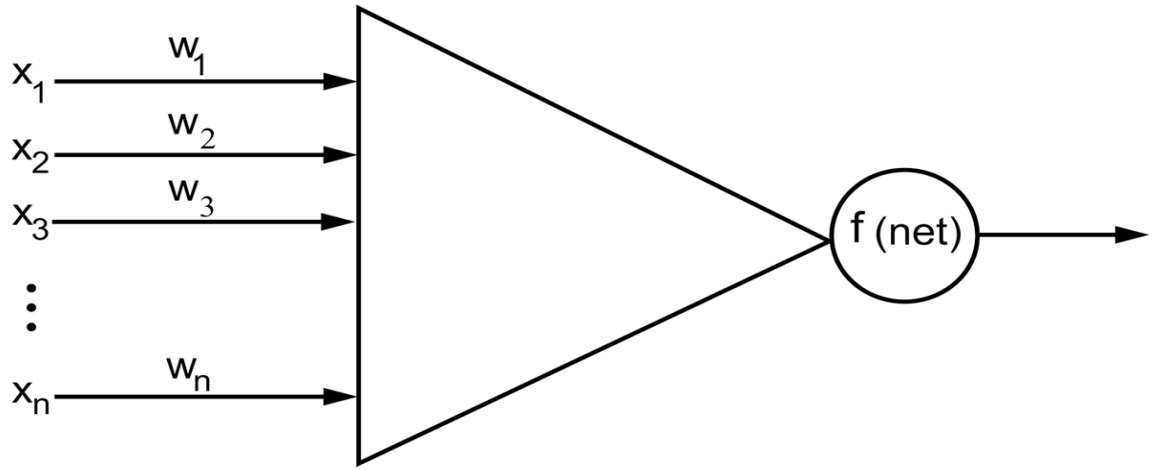


Neural Networks

- Ngược lại với các mô hình dựa trên ký hiệu: Không chú trọng việc sử dụng các ký hiệu một cách tường minh để giải quyết vấn đề.
- Ý tưởng dựa trên các hệ não: Xem trí tuệ là sự phát sinh từ các hệ thống gồm những thành phần đơn giản (neuron), tương tác với nhau thông qua một quá trình học hoặc thích nghi mà ở đó các kết nối giữa các thành phần được điều chỉnh.
- Gặt hái rất nhiều thành công trong những năm gần đây.
- Từ đồng nghĩa:
  - Tính toán neural (neural computing)
  - Các mạng neural (neural networks)
  - Các hệ kết nối (connectionist system)
  - Các hệ xử lý phân tán song song (parallel distributed processing)

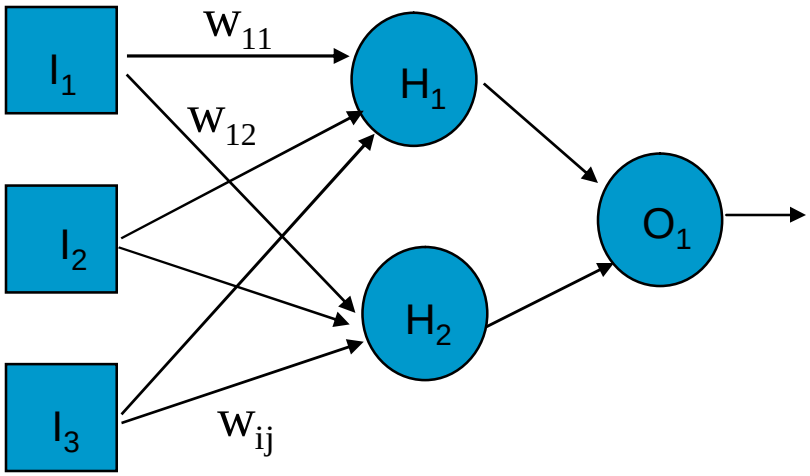
Neuron nhân tạo

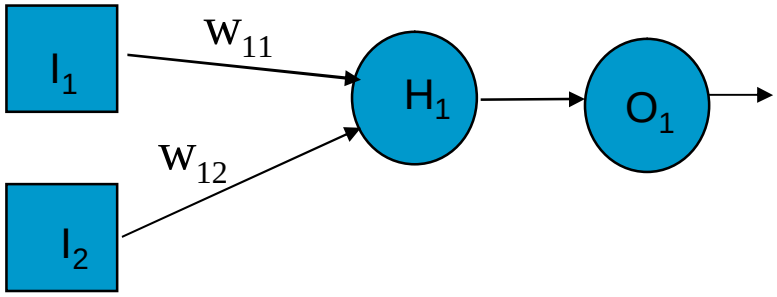
- Thành phần cơ bản của mạng neuron là một neuron nhân tạo.
- Các thành phần của một neuron nhân tạo:
  - Các tín hiệu vào x_i {0,1} {1,-1} real
  - Các trọng số w_i real
  - Một mức kích hoạt $\sum_i w_i x_i$
  - Một hàm ngưỡng $f : \sum_i w_i x_i \rightarrow$ tín hiệu ra



Neural Networks

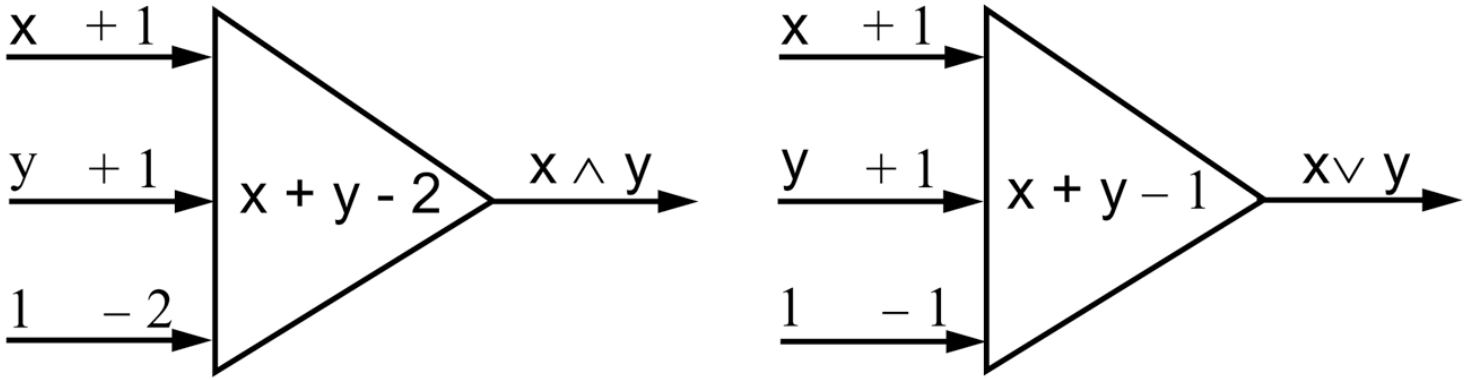
- Các thuộc tính tổng quát của một mạng là:
  - Hình thái mạng: mẫu kết nối giữa (các tầng của) các neuron.
  - Giải thuật học: cách điều chỉnh các trọng số trong quá trình xử lý tập dữ liệu rèn luyện
  - Cơ chế mã hóa: sự thông dịch của các tín hiệu vào và tín hiệu ra





Ví dụ: Neuron McCulloch-Pitts

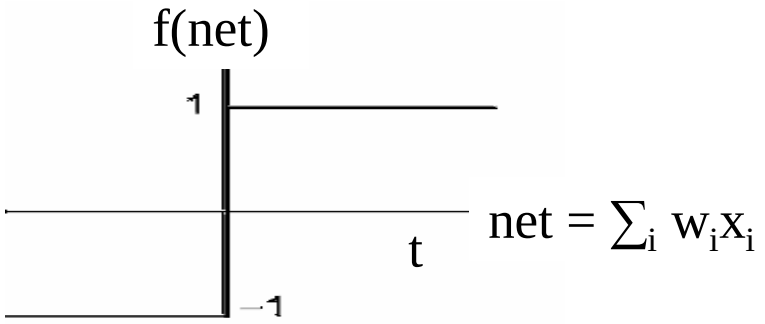
x	y	x + y – 2	Output
1	1	0	1
1	0	–1	–1
0	1	–1	–1
0	0	–2	–1



Các neurron dùng để tính các hàm logic and và or

Học Perceptron

- Mạng neuron đơn tầng
- Các giá trị vào 1 hoặc -1
- Các trọng số kiểu thực
- Mức kích hoạt $\sum_i w_i x_i$
- Hàm ngưỡng giới hạn cứng f :



$$1 \text{ if } \sum_i w_i x_i >= t$$
$$-1 \text{ if } \sum_i w_i x_i < t$$

- Điều chỉnh trọng số: $\Delta w_i = c(d - f(\sum_i w_i x_i))\, x_i$

c: hằng số chỉ tốc độ học

d: đầu ra mong muốn

Nếu kết quả thực và kết quả mong muốn giống nhau, không làm gì

Nếu kết quả thực là -1 và kết quả mong muốn là 1, tăng trọng số của đường thứ i lên $2cx_i$

Nếu kết quả thực là 1 và kết quả mong muốn là -1, giảm trọng số của đường thứ i xuống $2cx_i$

Phân loại của các hệ thống Học

- Học có sự hướng dẫn (Supervised learning)
  - Cho hệ thống một tập các ví dụ và một câu trả lời cho mỗi ví dụ.
  - Rèn luyện hệ thống cho đến khi nó có thể đưa ra câu trả lời đúng cho các ví dụ này.
- Học không có sự hướng dẫn (Unsupervised learning)
  - Cho hệ thống một tập hợp các ví dụ và cho nó tự khám phá các mẫu thích hợp trong các ví dụ.

Mạng neuron sử dụng một hình thức học có sự hướng dẫn

Sử dụng perceptron trong bài toán phân loại

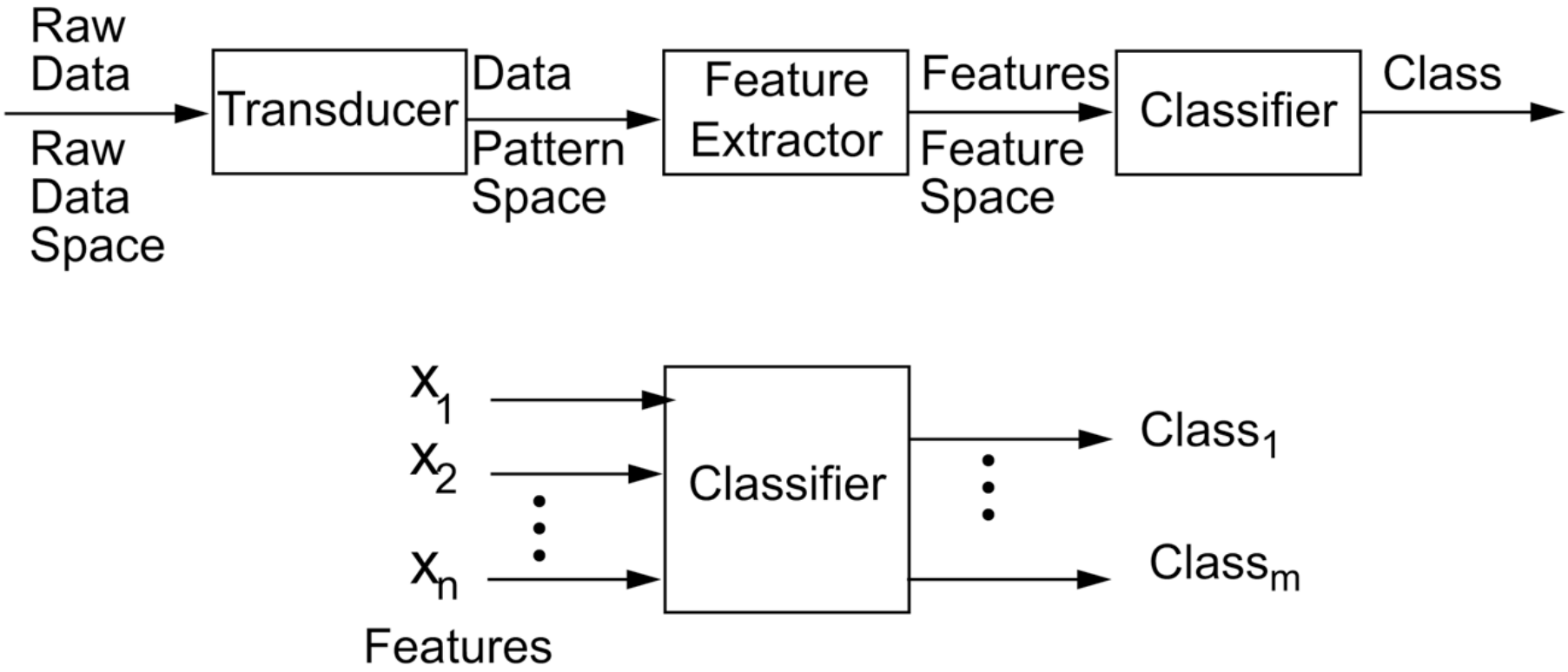
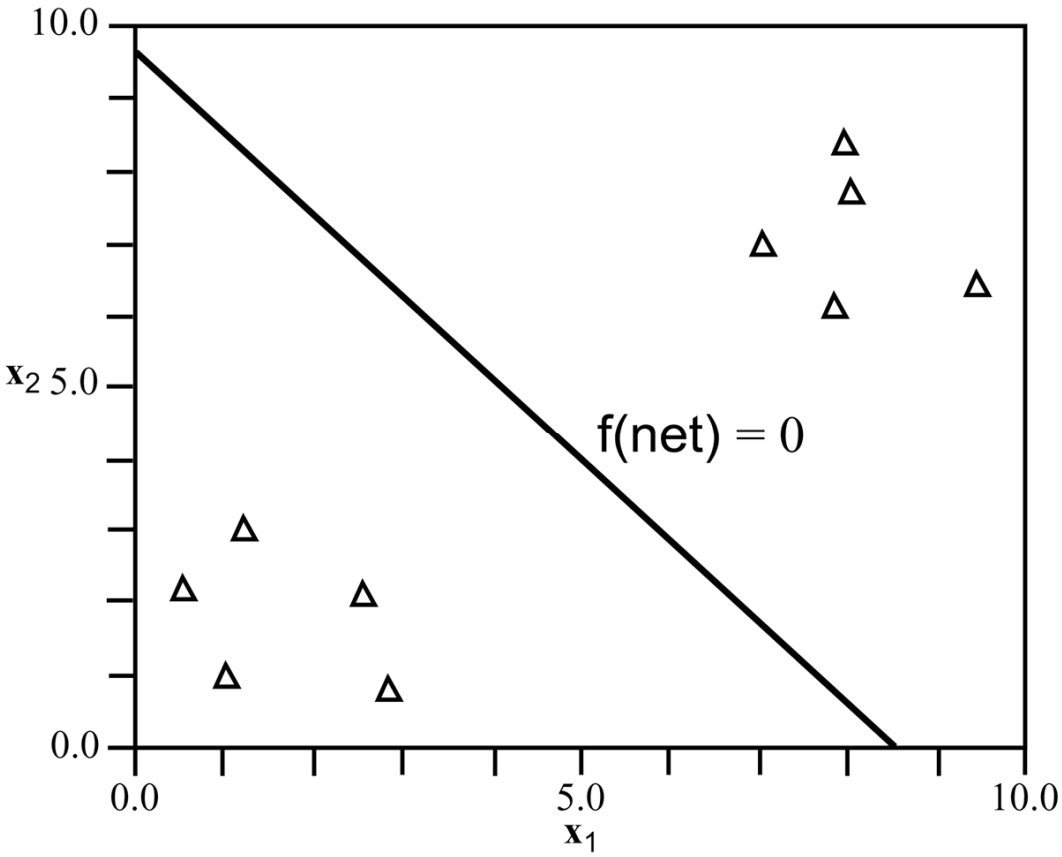


Fig 14-4: Một hệ thống phân loại đầy đủ

Ví dụ Perceptron

- Cho trước: một tập các dữ liệu vào
- Yêu cầu: rèn luyện perceptron sao cho nó phân loại các đầu vào một cách đúng đắn

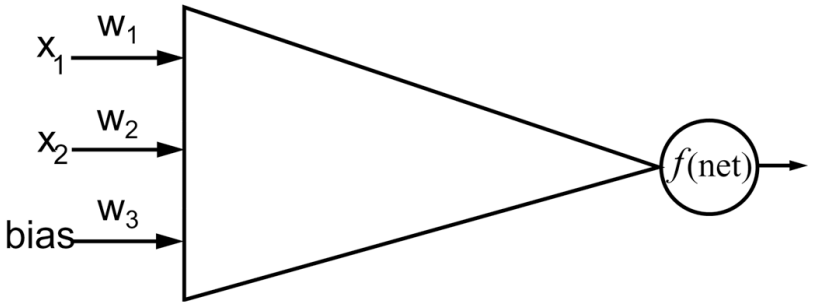
x_1	x_2	Output
1.0	1.0	1
9.4	6.4	−1
2.5	2.1	1
8.0	7.7	−1
0.5	2.2	1
7.9	8.4	−1
7.0	7.0	−1
2.8	0.8	1
1.2	3.0	1
7.8	6.1	−1



Ví dụ Perceptron: giải pháp

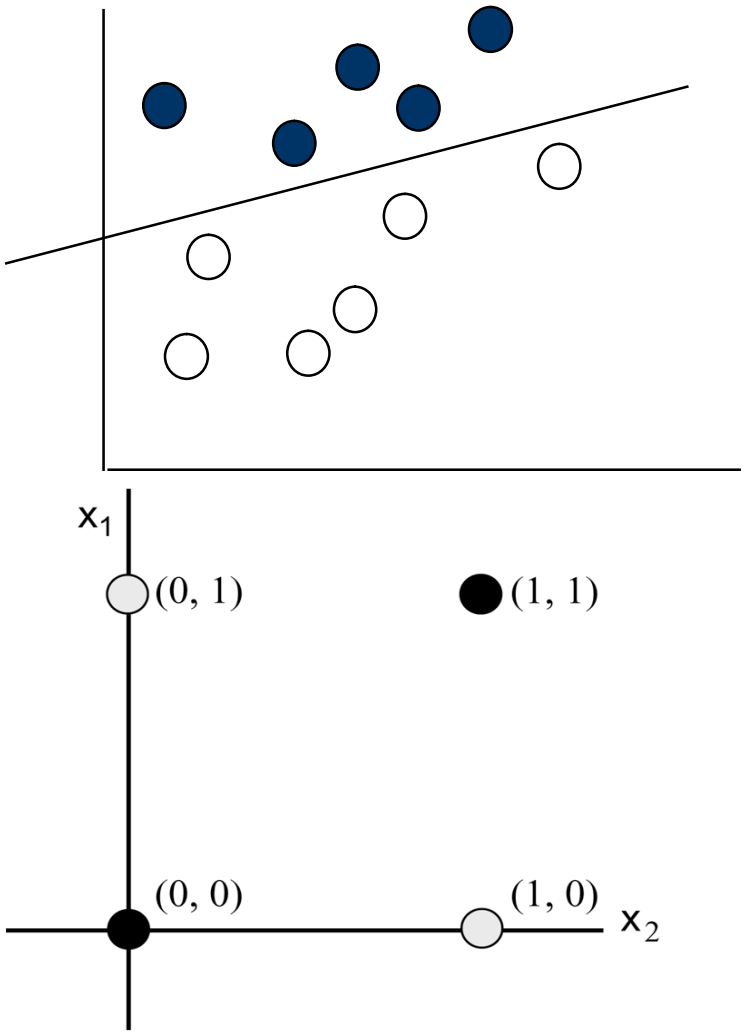
- 2 tín hiệu vào x_1 x_2
- Một tín hiệu vào thứ ba được sử dụng như một thiên vị và có giá trị cố định bằng 1, cho phép dịch chuyển đường phân cách
- Mức kích hoạt: $w_1x_1 + w_2x_2 + w_3$
- Hàm ngưỡng: hàm dấu, >0 = +1, <0 = -1

  đây là ngưỡng giới hạn cứng tuyến tính hai cực
- Các trọng số: được khởi tạo ngẫu nhiên, cập nhật 10 lần, với tốc độ học là 0.2
- Kết quả: **$-1.3x_1 + -1.1x_2 + 10.9 = 0$**



Tính tách rời tuyến tính (linearly seperatable)

- Trong một không gian n chiều, một sự phân loại mang tính tuyến tính nếu các lớp của nó có thể được tách rời bởi một mặt n-1 chiều.
- Perceptron không thể giải quyết các bài toán phân loại không tách rời tuyến tính.
  - Ví dụ: bài toán X-OR



Luật Delta

Tổng quát hóa perceptron bằng cách:

1. Thay thế hàm ngưỡng giới hạn cứng bằng các hàm kích hoạt khác có khả năng lấy vi phân

   Ví dụ: một hàm kích hoạt sigmoidal

   $f(net) = 1/(1 + e^{-\lambda * net})$ với $net = \sum_i w_i x_i$

   $f\,'(net) = f(net) * (1- f(net))$

2. Sử dụng luật delta để điều chỉnh trọng số trên đầu vào thứ k của nút thứ i

   $\Delta w = c(d_i - O_i)\, f'(net_i) x_k$

   $= c(d_i - O_i)\, O_i\, (1 - O_i)\, x_k$

f': đạo hàm bậc nhất
c: tốc độ học
d_i: đầu ra mong muốn
O_i: đầu ra thật sự

Lan truyền ngược (backpropagation)

- Tại các nút của các mạng đa tầng, lỗi mà một nút phải chịu trách nhiệm cũng phải được chia phần cho các nút ở tầng ẩn và các trọng số phải được điều chỉnh một cách phù hợp.
- *Giải thuật lan truyền ngược* bắt đầu tại tầng ra và truyền các lỗi ngược về xuyên qua các tầng ẩn.

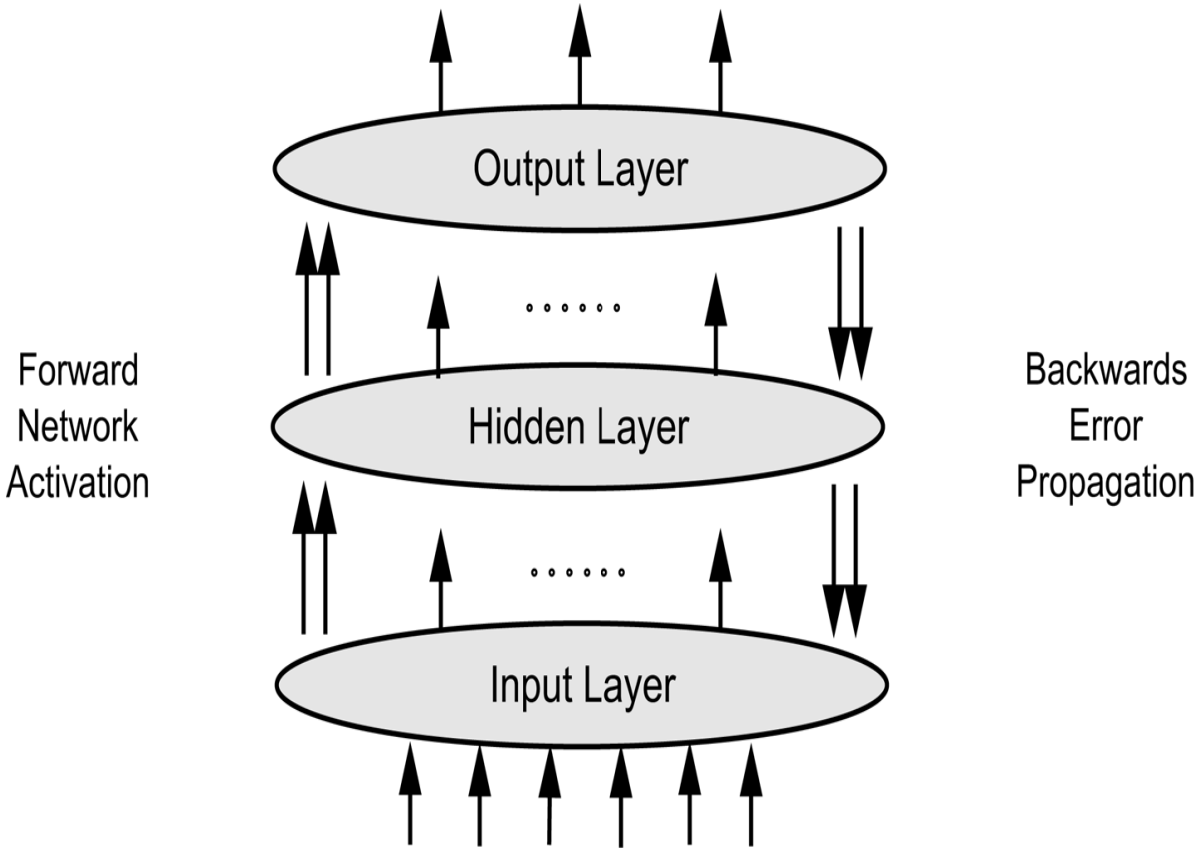
Luật delta tổng quát để điều chỉnh trọng số của đầu vào thứ k của nút thứ i:

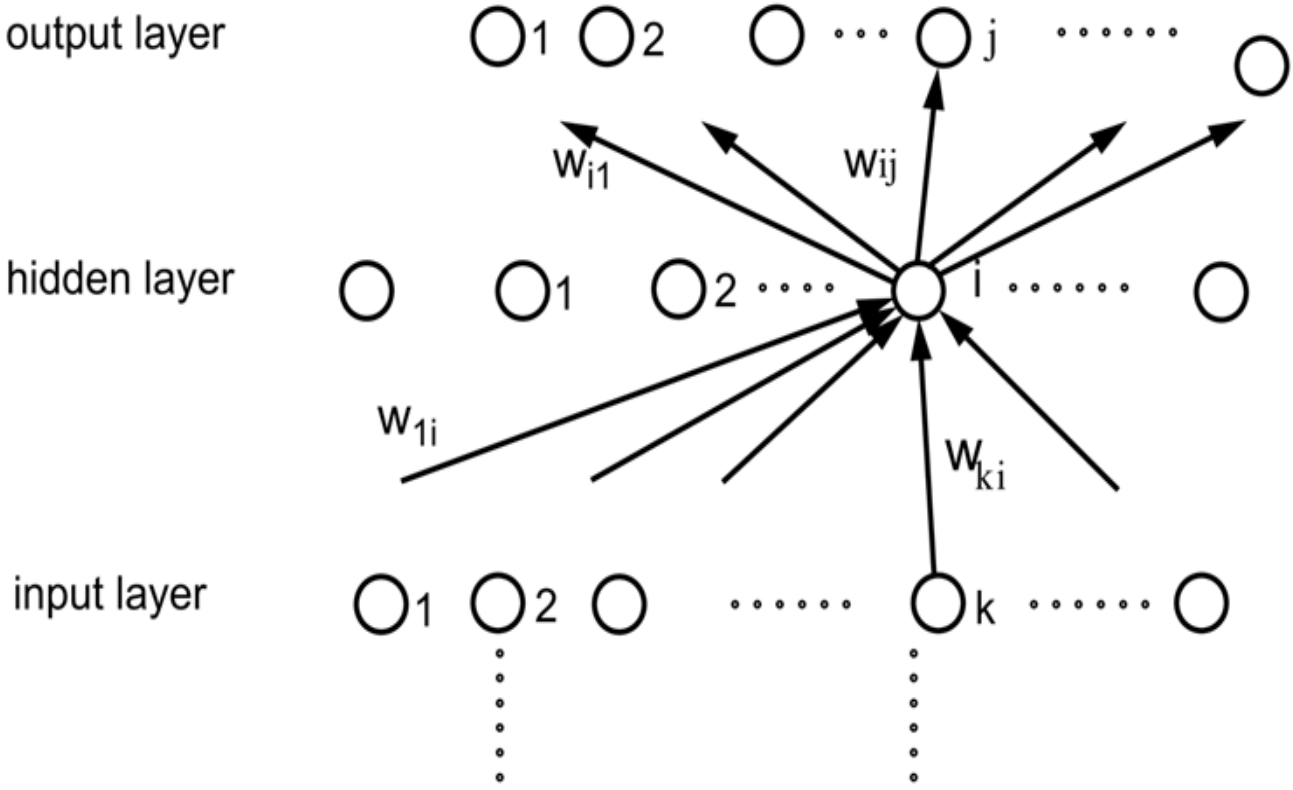
$\Delta w_k = c(d_i - O_i)\, O_i\, (1 - O_i)\, x_k$ cho nút ở tầng ra

$\Delta w_k = c \sum_j (delta_j\, w_{ij})\, O_i\, (1 - O_i)\, x_k$ cho nút ở tầng ẩn

với $delta_j = (d_j - O_j)\, O_j\, (1 - O_j)$

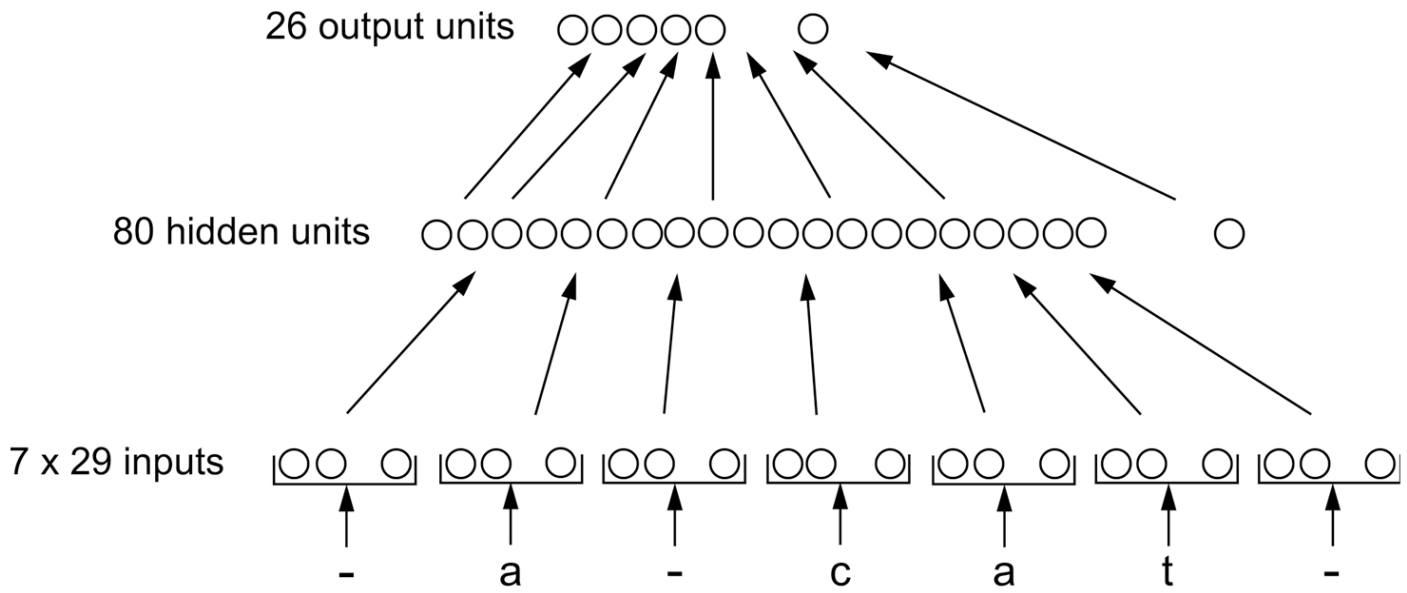
j chạy trên các nút của tầng kế tiếp mà tại đó nút i truyền các đầu ra của nó.


Output Layer
Hidden Layer
Input Layer
Forward
Network
Activation
Backwards
Error
Propagation


output layer
hidden layer
input layer
1
2
j
i
k
w_{i1}
w_{ij}
w_{1i}
w_{ki}

Ví dụ mạng Neuron: NETtalk

- Vấn đề: phát âm văn bản tiếng Anh đúng
- Đầu vào: một chuỗi
- Đầu ra: âm vị và trọng âm kèm theo cho mỗi ký tự
- Giải pháp:

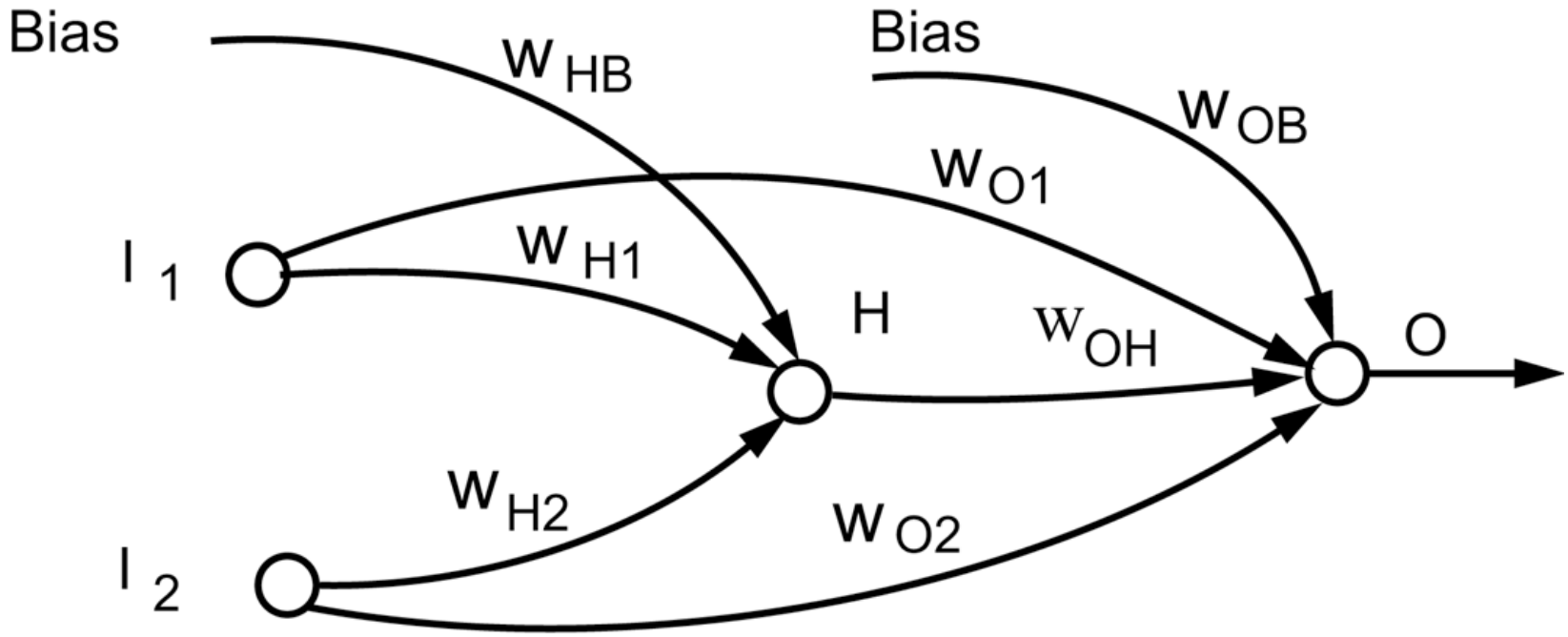


- Kết quả thực nghiệm:

  đúng 60% sau khi rèn luyện với 500 ví dụ (100 lượt)

  càng nhiều ví dụ rèn luyện => kết quả càng tốt

Figure 10.12: **A backpropagation net to solve the exclusive-or problem. The W_{ij} are the weights and H is the hidden node.**



Sử dụng 4 mẫu ví dụ để luyện tập:

(0,0) -> 0; (1,0) ->1; (0,1) -> 1; (1,1) ->0

Sau 1400 lượt: $W_{H1} = -7.0$ $W_{HB} = 2.6$ $W_{O1} = -5.0$

$W_{H2} = -7.0$ $W_{OB} = 7.0$ $W_{O2} = -4.0$

$W_{HO} = -11.0$

Các vấn đề liên quan khi sử dụng Neural Networks

- Các mạng đa tầng là đầy đủ về mặt tính toán, tuy nhiên:
  - Làm sao để chọn số nút ẩn và số tầng ẩn
  - Khi nào sử dụng các nút thiên lệch
  - Cách chọn một tập rèn luyện
  - Điều chỉnh các trọng số hay tốc độ học nên n.t.n?
  - …

Giải thuật Genetic

- Nắm bắt ý tưởng từ thuyết tiến hóa
- Học được xem như là sự cạnh tranh giữa các quần thể các giải pháp khả dĩ đang tiến hóa của bài toán
- Thành phần:
  - Quần thể các giải pháp khả dĩ
  - Hàm đánh giá
  - Các phép toán tạo con mới:
    - giao nhau (crossover)
    - Đột biến (mutation)
- Giải thuật:
  - Điều kiện kết thúc: #vònglặp, Trung bình 'độ tốt' của quần thể