Mục lục

[**#THUẬT TOÁN TÌM KIẾM** 2](#_Toc93213727)

[##TÌM KIẾM MÙ (UNINFORMED SEARCH) 2](#_Toc93213728)

[##TÌM KIẾM CÓ HEURISTIC 4](#_Toc93213729)

[#CSP: BÀI TOÁN THỎA MÃN RÀNG BUỘC 5](#_Toc93213730)

[#PP BIỂU DIỄN TRI THỨC 8](#_Toc93213731)

[##LOGIC 8](#_Toc93213732)

[a) định nghĩa **mô hình**: 8](#_Toc93213733)

[b) định nghĩa **câu** (sentence) 8](#_Toc93213734)

[c) định nghĩa “**nghĩa của câu**”. VD: X = true là 1 thể hiện (note: riêng lẻ X chưa dc gọi là 1 thể hiện) 9](#_Toc93213735)

[d) định nghĩa “**suy dẫn**” 9](#_Toc93213736)

[e) định nghĩa “**chứng minh**” 10](#_Toc93213737)

[**##LOGIC MỆNH ĐỀ** 10](#_Toc93213738)

[a) Mệnh đề horn 11](#_Toc93213739)

[b) Suy diễn tiến (forward chaining): giống bt xác định PK của bt dạng chuẩn CSDL 11](#_Toc93213740)

[c) Suy diễn lùi: quay lui từ q (goal): ktra xem q đã biết hay chưa, nếu chưa thì suy diễn lùi tất cả tiền đề (VT) của 1 luật nào đó rút ra q..nếu VT chưa biết cái nào thì coi là sub-goal rồi quay lùi tiếp…VD: 12](#_Toc93213741)

[d) Hợp giải mệnh đề 12](#_Toc93213742)

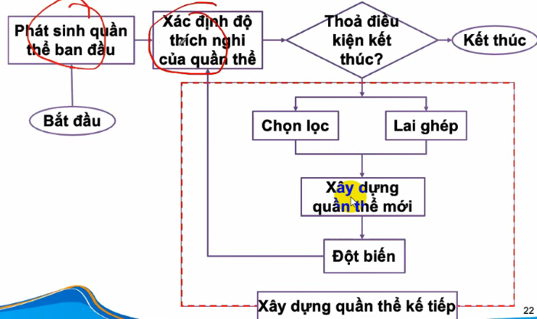
[**##LOGIC BẬC NHẤT** 15](#_Toc93213743)

[#MẠNG NGỮ NGHĨA 20](#_Toc93213744)

[#MẠNG NEURAL 21](#_Toc93213745)

[#ND THI 24](#_Toc93213746)

**#THUẬT TOÁN TIẾN HÓA**



# **#THUẬT TOÁN TÌM KIẾM**

##KHÁI NIỆM

- agent: là thực thể có khả năng quan sát môi trường & có hành động tương ứng

- 1 bài toán tìm kiếm trong ngữ cảnh AI bao gồm:

+ **không gian trạng thái** (state space) = **graph** (tree, network,..): trạng thái bắt đầu, hành động & mô hình di chuyển định nghĩa 1 ko gian trạng thái của bài toán. 3 khái niệm dc nói tới trong vid buổi 3

+ trạng thái (state) = nodes

+ hành động (actions) = edges: là hành động giữa các trạng thái, mỗi cạnh tương ứng với 1 hành động

- fringe: <https://ai.stackexchange.com/questions/5949/what-is-the-fringe-in-the-context-of-search-algorithms>

(tạm hiểu là CTDL để lưu trữ node?)

- 1 thuật toán tìm kiếm AI phải thỏa:

+ đường đi chi phí thấp nhất

+ time complexity

+ space complexity

+ complete: ?

- trạng thái và nút là 2 khái niệm khác nhau: trạng thái có thể lặp lại trong cây nhưng nút thì ko dc lặp lại trong cây vì sẽ tạo chu trình. 2 nút có thể cùng trạng thái nhưng vẫn phải khác nhau (khác dựa vào node cha của nút đó: node.parent)

## ##TÌM KIẾM MÙ (UNINFORMED SEARCH)

Các thuật toán mù dc đánh giá đơn giản bằng hàm f(n) = g(n), với g(n) là trọng số cạnh

**\*BFS**: tìm đường đi ngắn nhất (với ý nghĩa số bước di chuyển = số node ít nhất, NOT chi phí thấp nhất)

- implemetation note:

+ CTDL: queue

**\*DFS**: có xu hướng find ‘leftmost’ solution

- DFS có lợi hơn về mặt space complexity nhưng ko thực sự optimal vì có xu hướng đi theo ‘leftmost’ path nếu nghiệm nó nằm nhánh giữa

- implementation note:

+ CTDL: stack 🡪 cũng chính vì vậy về mặt implementation, DFS = **rightmost search** 🡪 đi phải trước. Coi lại video ‘[bt] DFS, UCS, Greedy’ để rõ

**\*ID** (iterative deepening): thừa hưởng ưu thể linear space của DFS và time complexity của BFS. Đây là thuật toán **vừa duyệt ngang vừa duyệt dọc**

- instruction: <https://www.youtube.com/watch?v=7QcoJjSVT38>

**\*UCS** (uniform cost search): đảm bảo tìm dc nghiệm mà tìm kiếm nghiệm đó có chi phí thấp nhất

- gần giống Dijkstra, sự khác biệt là: Dijkstra tìm shortest path từ 1 node ban đầu tới all nodes còn lại. Còn UCS là tìm shortest path từ 1 node đầu tới MỘT node goal

- thuật toán

+ B1: cho đỉnh xuất phát vào tập **open**

+ B2: open rỗng? 🡪 ko tìm thấy goal hoặc goal ko có trong graph

+ B3: đặt O = đỉnh đầu của open. Check (O == goal)? Nếu đúng thì dừng tìm kiếm vì O sẽ là nghiệm rẻ nhất; nếu sai thì thì add O vào tập **close**

+ B4: find all O’s **unvisited** **adjacent** nodes và cho vào open **theo TT tăng dần k/c từ điểm xuất phát**. Lưu ý unvisited ở đây là chưa có trong tập close chứ có thể có trong tập open

+ B5: trở lại B2

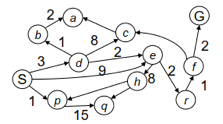
- implementation notes:

+ open 🡪 CTDL priority queue

+ tập close 🡪 visited = [] dùng lưu các node đã ‘chọn để mở’

+ cần có thuộc tính par để lưu node parent. Lưu ý 1 node x có thể có nhiều node par, việc cập nhật lại x.par để tìm đường đi tối ưu khi và chỉ khi node x nằm đầu priority queue

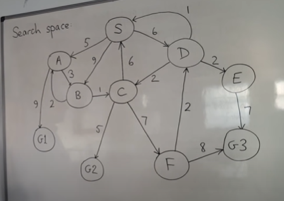
Ex1:



|  |  |  |
| --- | --- | --- |
| Step | Open (sắp theo chiều tăng dần k/c từ S🡪đỉnh đang xét) | Close = visited |
| 1 | (S,0) | RỖNG |
| 2 | (p,1), (d,3), (e,9) | S (do S ko phải đích) |
| 3 | (d,3), (e,9), **(q,16)** | S,p (do p ko phải đích) |
| 4 | **(b,4)**, **(e,5)**, (e,9), **(c,11)**, (q,16) 🡪 ko gộp e nếu e chưa có trong close | S,p,d |
| 5 | (e,5), **(a,6)**, (e,9), (c,11), (q,16) | S,p,d,b |
| 6 | (a,6), **(r,7)**, ~~(e,9)~~, (c,11), **(h,13)**, (q,16) 🡪 loại e vì có e trong tập close | S,p,d,b,e |
| 7 | (r,7), (c,11), (h,13), (q,16) | S,p,d,b,e,a |
| 8 | **(f,8)**, (c,11), (h,13), (q,16) | S,p,d,b,e,a,r |
| 9 | **(G,10)**, (c,11), (h,13), (q,16) 🡪 nếu goal xhien ngay head của tập open thì nó là nghiệm rẻ nhất, nếu muốn coi các nghiệm còn lại thì làm tiếp các bước thuật toán | S,p,d,b,e,a,r,f |

**\*Tip**: để tìm đường đi, vẽ lại hình và đánh dấu node đã thăm trên hình, kết thúc thuật toán, nhìn vào hình sẽ tìm lại dc đường đi

Ex2: <https://www.youtube.com/watch?v=dRMvK76xQJI>



|  |  |  |
| --- | --- | --- |
| step | Open | Close |
|  | (s,0) |  |
|  | (a,5), (d,6), (b,9) | s |
|  | (d,6), **(b,8**), (b,9), **(g1,14)** | S,a |
|  | (b,8), **(c,8)**, **(e,8)**, (b,9), (g1,14) | sad |
|  | (c,8), (e,8), **(c,9)**, ~~(b,9)~~, (g1,14) 🡪 loại luôn (b,9) vì đã có (b,8) ngắn hơn dc visit | sadb |
|  | (e,8), ~~(c,9)~~, **(g2,13)**, (g1,14), **(f,15)** 🡪 loại luôn (c,9) | sadbc |
|  | (g2,13), (g1,14), (f,15), **(g3,15)** | sadbce |

🡺 stop, vì chọn dc goal rẻ nhất là (g2,13)

## ##TÌM KIẾM CÓ HEURISTIC

- heuristic: là 1 hàm ước lượng k/c trạng thái htai với **trạng thái đích**; nó như là 1 thực thề nhận biết ‘gần xa’ – còn gọi là ‘trí khôn của thuật toán’

h(n): chi phí ước tính **k/c** từ n **đến đích** (vd: euclid/manhattan)

+ mỗi trạng thái đích có 1 heuristic riêng, ko dùng heuristic của trạng thái đích này cho trạng thái đích khác

- trọng số cạnh c(s,a,s’): chi phí ước tính **di chuyển (trọng số)** tới s’ (s 🡪 s’ thông qua hành động a)

- **f(n)**: hàm đánh giá, tùy thuật toán mà hàm này sẽ có CT khác nhau để ra hiệu có mở rộng theo node đó ko

\*DK cho 1 heuristic **hợp lý** & **nhất quán**: slide 4 thầy Đức

**\*greedy search**: f(n) = h(n)

- thực thi như UCS nhưng priority queue ưu tiên f(n) bé 🡪 lớn

- ít tốn chi phí hơn UCS??

- ko đảm bảo tối ưu (i.e. hên thì mới ra tối ưu)

**\*thuật giải A\***: kết hợp UCS + greedy 🡪 f(n) = h(n) + g(n)

+ **g(n)**: chi phí đường đi tới n (trọng số cạnh) - cost of path

+ **h(n)**: ước tính k/c tới đích (đánh giá độ gần của trạng thái htai tới đích)

+ **f(n)**: **ước tính** chi phí đến đích

- thực thi như UCS nhưng priority queue ưu tiên f(n) bé 🡪 lớn

- để A\* tối ưu thì heuristic phải hợp lý & nhất quán

+ hợp lý: h(n) ≤ h\*(n)

với h\*(n) là **chi phí thấp nhất** đến đích trong thực tế

+ nhất quán: với mỗi successor n’ của n thì h(n) ≤ c(n,a,n’) + h(n’)

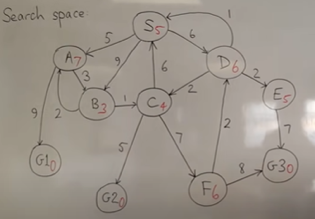
- thiết kế heuristic: sdung của btoan nới lỏng (relaxed problem)

- A\* cố gắng thu hẹp ko gian tìm kiếm & mở đường đi tới đích

- <https://en.wikipedia.org/wiki/Admissible_heuristic>

- time complexity & space complexity: hàm mũ ! 🡪 giảm space complexity dc bằng cách sdung thuật toán biến thể bên dưới để giới hạn ko gian tìm kiếm của A\*

VD:



**\*Tip:** nếu đề cho graph như trên, 1 cách trình bày đó là vẽ bảng như sau

Chart, radar chart

Description automatically generated

**\*thuật giải IDA\*** (iterative deepening A\*): ?

**\*thuật giải RBFS** (recursive best first search): ?

# #CSP: BÀI TOÁN THỎA MÃN RÀNG BUỘC

* Cần xác định:
  + Tập biến: X = {X1,…,Xn}
  + Tập miền gtri D = {D1,…,Dn} với Di = {v1,…, vk}: miền gtri của biến Xi. **Tip**: tưởng tượng D = {RGB}
  + C = Constraints: tập DK ràng buộc
* Vẽ đồ thị ràng buộc cần xác định:
  + Node: các biến
  + Cung: ràng buộc

VD: tô màu RGB cho các mảnh đất sau biết rằng 2 vùng lân cận phải khác màu nhau



X = {WA; NT; Q; NSW; V; SA; T}

Di = {R,G,B}

C = {SA != WA; SA != NT; SA != Q; SA != NSW; SA != V; WA != NT; NT != Q; Q != NSW; NSW != V}

A picture containing text, clock, watch

Description automatically generated

**\*1 số pp heuristic giải btoan CSP**

* MRV (minimum remaining value): chọn biến có **tập giá trị nhỏ nhất** 🡪 tạo lỗi sớm để loại lỗi sớm (tỉa nhánh)
* DH (degree heuristic): chọn biến có nhiều ràng buộc nhất vs các biến còn lại để gán gtri 🡪 giảm SL nhánh con
* LCV (Least-constraining value): chọn giá trị có ảnh hưởng tối thiểu đến các giá trị khác. Lưu ý là chọn ‘giá trị’ chứ ko phải chọn ‘biến’ như 2 cách trên. Trong VD này, ‘biến’ là tên các lãnh thổ, ‘giá trị’ là các màu

Diagram

Description automatically generated

* GH (greedy heuristic): mỗi lần chọn 1 cái tốt nhất

**\*1 số thuật toán có sdung 1 trong 3 heuristic trên để giải CSP**

* Kiểm tra tiến + cạnh hợp lệ (chưa xài heuristic)

Chart, application

Description automatically generated

* + i=0: trạng thái đầu, tất cả biến X (lãnh thỗ) đều có khả năng nhận 3 giá trị (3 màu) R,G,B
  + i=1: chọn đại WA là đỏ, khi đó các biến (lãnh thổ lân cận) WA phải loại đỏ
  + i=2: chọn đại Q, làm tương tự i=1. Thấy ngay lỗi 🡪 khi đó cần lan truyền ràng buộc bằng **pp cạnh hợp lệ**

|  |
| --- |
| Một cạnh X 🡪 Y là hợp lệ (arc-consistency) khi ∀x thuộc DX, ∃y lân cận x thuộc DY ko vi phạm ràng buộc. Nếu ∃y thuộc DY vi phạm ràng buộc thì cần loại gtri nào đó của y |

**Note**: chỉ check cạnh hợp lệ khi tồn tại 1 biến với đúng 1 gtri còn sót lại (VD: SA)

A picture containing diagram

Description automatically generated



VD: x = SA thuộc{Blue} có y = NSW lân cận x = SA cũng trùng màu Blue với x 🡪 vi phạm ràng buộc nên ta loại bỏ gtri Blue của NSW. Mà khi NSW thay đổi thì biến (lãnh thổ) lân cận nó cũng cần xét lại, tức xét V kề NSW…(làm tương tự).

* Quay lui:
  + B1

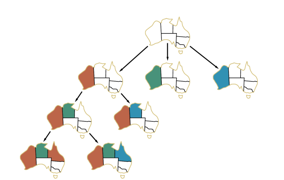
A picture containing map, text

Description automatically generated

* + B2



* + B3



* + Expand tương tự

# #PP BIỂU DIỄN TRI THỨC

## ##LOGIC

Text

Description automatically generated

- 1 số ký hiệu

+ I(x): ngữ nghĩa/ý nghĩa diễn giải của x. VD:

Text

Description automatically generated

+ A |= B: A “bao hàm” B hay B được chứa bởi A. **DK bao hàm**: nếu A đúng thì B cũng phải đúng

\*CÁC ĐỊNH NGHĨA CẦN NHỚ

1. định nghĩa **mô hình**:

Diagram, text

Description automatically generated

1. định nghĩa **câu** (sentence)

Text

Description automatically generated

+ T/c của câu

Text

Description automatically generated

**Bsung**: phản chứng thì luôn hợp lệ

A picture containing text, sky, roof

Description automatically generated

VD:

A picture containing diagram

Description automatically generated

1. định nghĩa “**nghĩa của câu**”. VD: X = true là 1 thể hiện (note: riêng lẻ X chưa dc gọi là 1 thể hiện)

Text

Description automatically generated

Text

Description automatically generated with medium confidence

1. định nghĩa “**suy dẫn**”

Diagram

Description automatically generatedđại khái ta phải tìm toàn bộ thể hiện làm cho KB đúng 🡪 rất nhiều thể hiện 🡪 tốn time duyệt

1. định nghĩa “**chứng minh**”

Text

Description automatically generatedText

Description automatically generatedDiagram

Description automatically generated

+ luật suy diễn

Table

Description automatically generated with medium confidence

VD: chứng minh S, biết 3 dòng đầu cho trước

Table

Description automatically generated

Ta có thể gọi 3 dòng đầu là: KB

Kết luận: 3 dòng đầu **suy dẫn** dc S

## **##LOGIC MỆNH ĐỀ**

- liên quan tới câu (sentence) và t/c của câu dc định nghĩa phía trên

- độ ưu tiên toán tử: not, giao, hợp, kéo theo, tương đương

- 2 phép tương đương cần nhớ:

Text

Description automatically generated

1. Mệnh đề horn

- literal: là các biến mệnh đề. VD: A, !A, B, !B

tối đa

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

1. Suy diễn tiến (forward chaining): giống bt xác định PK của bt dạng chuẩn CSDL

Text

Description automatically generated

Text

Description automatically generated

1. Suy diễn lùi: quay lui từ q (goal): ktra xem q đã biết hay chưa, nếu chưa thì suy diễn lùi tất cả tiền đề (VT) của 1 luật nào đó rút ra q..nếu VT chưa biết cái nào thì coi là sub-goal rồi quay lùi tiếp…VD:

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

1. Hợp giải mệnh đề

- đòi hỏi các câu (sentence) phải chuyển về dạng hội (^) chuẩn (CNF): là dạng chỉ có 3 dấu **hội, giao, not**

Text

Description automatically generated

#### **\*hợp giải Robinson** (c/m phản chứng): muốn c/m KB => a đúng thì c/m điều ngược lại là sai. Các bước:

Graphical user interface, text

Description automatically generated

VD 1:

Table

Description automatically generatedcó thể kết hợp bước 3 & 5 để tiết kiệm bước

#### **\*Thủ tục Davis Putnam**:

- Hợp giải xong thành mệnh đề mới thì bỏ 2 mệnh đề cũ đã hợp giải (khác với Robinson là vẫn giữ lại 2 mệnh đề cũ)

- 1 lần có thể hợp giải nhiều mệnh đề cùng lúc để tạo nhiều mệnh đề mới (VD dưới), khác với Robinson là mỗi lần chỉ hợp giải dc 2 mệnh đề

Text

Description automatically generated

#### **\*Thuật giải Vương Hạo**: dùng pp chia để trị (NOTE: ko cần làm theo thứ tự các bước)

- B1: Phát biểu lại giả thuyết và kết luận của bài toán dưới dạng chuẩn sau

GT1, GT2, …, GTn → KL1, KL2, … KLm

- B2: **Chuyển vế** các GTi và KLj (phải ở dạng mệnh đề, not biến mệnh đề) có dạng phủ định. Khi chuyển vế thì **mất dấu NOT**

+ Note: -(r ^ s) chuyển sang VP là (r ^ s)

-(r ^ -s) chuyển sang VP là (r ^ -s). Vì ‘s’ là [biến mệnh đề] nên không khử dấu của s

- B3: Thay dấu “∧” ở trong GTi và dấu “∨” ở trong KLj bằng dấu “,”

- B4: Nếu dòng hiện hành có một trong hai dạng sau..thì thay bằng:

|  |  |  |
| --- | --- | --- |
| Dạng |  | …thì thay bằng |
| 1 |  |  |
| 2 |  |  |

- B5: **1 dòng dc c/m** nếu **tồn tại chung** một mệnh đề ở **cả 2 vế** thì coi như đúng. VD: p,q 🡪 p

- B6:

+ Nếu một dòng không còn dấu ‘∨’ và ‘∧’ mà cả ở hai vế đều không có chung **biến mệnh đề** nào thì dòng đó không được chứng minh.

TRAP !!! gs: p, !p 🡪 q thì theo B2: p 🡪 p,q. Theo bước 5 thì mệnh đề dc c/m 🡪 nhớ thuật toán Vương Hạo ko nhất thiết theo thứ tự các bước

+ mọi nhánh được chứng minh (DCM) 🡺 bài toán dc c/m.

+ tồn tại 1 nhánh ko dc c/m (KCM) 🡺 dừng thuật toán và bài toán ko dc c/m

VD: r, !p OR s 🡪 q, !r AND s

Phân thành 2 dòng:

1. r, !p 🡪 q, !r AND s
2. r, s 🡪 q, !r AND s

(1) tách thành:

(1.1) r, !p 🡪 q, !r

= r, r 🡪 p, q

= Ko dc c/m

(1.2) r, !p 🡪 q, s

= r 🡪 p,q,s

= Ko dc c/m

(2) tách thành

(2.1) r, s 🡪 q, !r

= Ko dc c/m

(2.2) r, s 🡪 q, s

= Dc c/m

Kết luận: bài toán ko dc c/m

##BT Vương Hạo

<https://sinhvientot.net/giai-thuat-vuong-hao-bai-tap-2/>

## **##LOGIC BẬC NHẤT**

- tên riêng được coi là hằng (const). VD: Lan, John

- biểu diễn: verb(verb’s main subject, O).

VD: Cháu(x,y): x là cháu của y

- lượng từ ‘với mọi’ (thường đi kèm với dấu kéo theo): ∀x. P

Ex: Sinh viên CNTT thì thông minh 🡪 ∀x. Sinh-viên(x,CNTT) 🡺 thông-minh(x)

- lượng từ ‘tồn tại’ (thường đi kèm với dấu giao “^”): ∃x. P. chỉ cần có tồn tại 1 cái gì đó thì xài dc

\*Hợp giải logic bậc nhất: tương tự như “chứng minh” của Logic (mục e)

**Table

Description automatically generatedthế A vào x(\*)** xong hợp giải, VD như hợp giải !P(x) V P(A) thì sẽ thành 1

**🡪 Thế nào là 1 phép thế đúng đắn?**

VD 1:

Table

Description automatically generated

VD 2:

A picture containing table

Description automatically generated

VD 3:

a) Art là cha của Bob và Bud

Bob là cha của Cal và Coe

Ông nội là cha của cha 🡪 ∀ x,y,z. F(x,y) ^ F(y,z) => G(x,z) 🡪 lấy phủ định

Hỏi: Art có là ông của Coe?

🡪 giả sử Art KHÔNG là ông Coe

🡪 !G(Art, Coe)

Tip: thường hội chuẩn với kết luận trước

|  |  |  |
| --- | --- | --- |
| 1 | F(Art, Bob) | Tiền đề |
| 2 | F(Art, Bud) | Tiền đề |
| 3 | F(Bob, Cal) | Tiền đề |
| 4 | F(Bob, Coe) | Tiền đề |
| 5 | !F(x,y) V !F(y,z) V G(x,z) | Tiền đề |
| 6 | !G(Art, Coe) | Kết luận |
| 7 | !F(Art,y) V !F(y, Coe) | 5,6; theta={x/Art, z/Coe} |
| 8 | !F(Bob, Coe) | 1,7; theta={x/Art, z/Coe, y/Bob} |
| 9 | False (dpcm) | 4,8; theta={x/Art, z/Coe, y/Bob} |

b) Art là cha của Bob và Bud

Bob là cha của Cal và Coe

Ông nội là cha của cha 🡪 ∀ x,y,z. F(x,y) ^ F(y,z) => G(x,z)

Ai là cháu của Art?

🡪 gs ko ai là cháu của Art = Art ko là ông của bất kỳ ai

🡪 ∀ t. !G(Art, t)

|  |  |  |
| --- | --- | --- |
| 1 | F(Art, Bob) | Tiền đề |
| 2 | F(Art, Bud) | Tiền đề |
| 3 | F(Bob, Cal) | Tiền đề |
| 4 | F(Bob, Coe) | Tiền đề |
| 5 | !F(x,y) V !F(y,z) V G(x,z) | Tiền đề |
| 6 | !G(Art, t) | Kết luận |
| 7 | !F(Art,y) V !F(y,t) | 5,6; {x/Art, z/t} |
| 8 | !F(Bob,t) | 1,7; {y/Bob, x/Art, z/t} |
| 9 | !F(Bud,t) | 2,7; {y/Bud, x/Art, z/t} 🡪 có thể xài lại 7 (bỏ bước này củng dc) |
| 10 | false | 3,8; {t/Cal, y/Bob, x/Art, z/t} |
| * Cal là cháu của Art | | |

b) Art là cha của Bob và Bud

Bob là cha của Cal và Coe

Ông nội là cha của cha 🡪 ∀ x,y,z. F(x,y) ^ F(y,z) => G(x,z)

Hỏi: Các cặp ông cháu?

🡪 ∀ x,z. G(x,z) (lưu ý về ngữ nghĩa mà dùng lại x và z, chứ ko thêm biến mới)

|  |  |  |
| --- | --- | --- |
| 1 | F(Art, Bob) | Tiền đề |
| 2 | F(Art, Bud) | Tiền đề |
| 3 | F(Bob, Cal) | Tiền đề |
| 4 | F(Bob, Coe) | Tiền đề |
| 5 | !F(x,y) V !F(y,z) V G(x,z) | Tiền đề |
| 6 | ! G(x,z) | Kết luận |
| 7 | !F(x,y) V !F(y,z) | 5,6 |
| 8 | !F(Bob,z) | 1,7; {x/Art, y/Bob} |
| 9 | !F(Bud,z) | 2,7; {x/Art, y/Bud} |
| 10 | false | 3,8; {x/Art, y/Bob, z/Cal} |
| 11 | false | 4,8; {x/Art, y/Bob, z/Coe} |
| * 2 cặp ông cháu Art-Cal và Art-Coe | | |

//BT:

Text

Description automatically generated

1. ∃x. D(x) ∧ O(Jack,x)

= D(A) ∧ O(Jack,A) (Skolem - thay tên mới cho tất cả lượng từ ∃)

\*Lưu ý: khi đi vào bảng phải tách D(A) và O(Jack,A) riêng vì chúng có dấu “∧”

1. ∀x. (∃y. D(y) ∧ O(x, y)) → L(x)

= ∀x. ¬(∃y. D(y) ∧ O(x, y)) ∨ L(x)

= ∀x. ∀y. ¬D(y) ∨ ¬O(x, y) ∨ L(x)

= ¬D(y) ∨ ¬O(x, y) ∨ L(x) (luật 5 - bỏ ∀)

1. ∀x. L(x) → (∀y. A(y) **→** ¬K(x,y)) 🡪 vì là lượng từ với mọi nên dùng kéo theo

= ∀x. ¬L(x) ∨ (∀y. ¬A(y) ∨ ¬K(x,y))

= ¬L(x) ∨ ¬A(y) ∨ ¬K(x,y) (luật 5 - bỏ ∀)

1. K(Jack,Tuna) ∨ K(Curiosity,Tuna)
2. C(Tuna)
3. ∀x. C(x) → A(x)

= ¬C(x) ∨ A(x)

GOAL

1. K(Curiosity,Tuna)

|  |  |  |
| --- | --- | --- |
| 1 | D(A) | Tiền đề |
| 2 | O(Jack,A) | Tiền đề |
| 3 | ¬D(y) ∨ ¬O(x, y) ∨ L(x) | Tiền đề |
| 4 | ¬L(x) ∨ ¬A(y) ∨ ¬K(x,y) | Tiền đề |
| 5 | K(Jack,Tuna) ∨ K(Curiosity,Tuna) | Tiền đề |
| 6 | C(Tuna) | Tiền đề |
| 7 | ¬C(x) ∨ A(x) | Tiền đề |
| 8 | ¬K(Curiosity,Tuna) | Kết luận |
|  | | |

\*CÁC KIẾN THỨC CẦN DÙNG ĐỂ HỢP GIẢI LOGIC BẬC NHẤT PHÍA TRÊN

\*\*Cần biến đổi các logic bậc nhất thành **mệnh đề** **CNF** (clausal form). Kết hợp các cách sau

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

\*\*Phép thế

Table

Description automatically generated

\*\*Phép đồng nhất: 1 phép thế gọi là phép đồng nhất khi thế nó vào 2 biểu thức thì 2 biểu thức giống y chang nhau

Table

Description automatically generated

+ w1s: thay s vào biểu thức w1

+ y/x: thay y **THÀNH** x

\*\*Phép đồng nhất TQ nhất (most general unifier - MGU): <https://www.youtube.com/watch?v=zeyjeGDxrWc>

* Có thể thay 1 hàm f thành 1 hàm f, miễn **cùng SL input**: f(t1,…,t**n**)/f(u1,…,u**n**). Nếu khác SL input hoặc khác hàm (VD: f và g thay vì f và f) 🡪 ko có MGU
* Có thể thay x THÀNH 1 hàm và ngược lại, miễn là x **ko nằm** trong input của hàm:   
  x/f(t1,…,tn) = f(t1,…,tn)/x. Nếu x nằm trong input của f(t1,..x..,tn) 🡪 ko có MGU

Table

Description automatically generated

VD:

Table

Description automatically generated

# #MẠNG NGỮ NGHĨA

Diagram

Description automatically generated

- Sau khi dựng xong MNN, đề cho biến nào thì **kích hoạt** biến đó trong đồ thị. Biến dc kích hoạt sẽ truyền động ra mọi nhánh nối vs nó đến các đỉnh neighbor

- Khi truyền động tới 1 đỉnh (ở đây là CT) bất kỳ, nếu CT có n-1 biến dc xác định (hay ‘kích hoạt’)

🡪 biến còn lại trong CT dc **auto kích hoạt**

🡪 CT dc kích hoạt

- tiếp tục từ những đỉnh dc kích hoạt truyền động ra các đỉnh neighbor

VD: những đỉnh viền đỏ là những đỉnh dc kích hoạt; cạnh đỏ thể hiện sự lan truyền

Cho trước alpha, beta, c 🡪 tính diện tích tam giác

Diagram

Description automatically generated

Text

Description automatically generated with low confidencelời giải bài toán là những suy luận có VT là 1 CT

# #MẠNG NEURAL

Diagram

Description automatically generated with low confidence



(1): gọi là **hàm kích hoạt** = tổng của(tích giữa các input xi & trọng số wi)

(2): ngưỡng, thuộc [0,1], sẽ dc cho trước bởi 1 hàm

- input X & output Y chỉ nhận giá trị 0 or 1

- nếu hàm kích hoạt > ngưỡng thì output Y=1; ngược lại output Y=0

- khái niệm ‘lớp’:

+ lớp nhập: là W, ko tính X

+ lớp ẩn: có nhiều lớp

+ lớp xuất: là Y

VD 1: cho neural có trọng số như hình, ta muốn output=1 thì input phải là mấy

A picture containing diagram

Description automatically generated

VD 2: cho mạng neural sau, biết output=1. Tìm input

Diagram

Description automatically generated

\*Nhận dạng chữ C & T

Diagram

Description automatically generated

#BT ÔN

Dùng ID3 cho bảng sau:

Table

Description automatically generated

Diagram

Description automatically generated

Text

Description automatically generated

Đây là kq 🡪 phải rút luật từ cây

Giải

**LẦN 1**

S = [3 á, 5 âu] 🡪 E(S) = -3/8.log(3/8) - 5/8.log(5/8) = 0.9544

\*G(S, h.dáng) = E(S) - 3/8.E(Sto) - 5/8.E(Snhỏ) = **3.16\*10-3**

+ Sto = [1 á, 2 âu] 🡪 E(Sto) = -1/3.log(1/3) -2/3.log(2/3) = 0.9183

+ Snhỏ = [2 á, 3 âu] 🡪 E(Snhỏ) = -2/5.log(2/5) - 3/5.log(3/5) = 0.971

\*G(S, c.cao) = E(S) - 4/8.E(STB) - 1/8.E(Sthấp) -3/8.E(Scao) = **0.4544**

+ STB = [2 á, 2 âu] 🡪 E(STB) = 1

+ E(Sthấp) = 0

+ Scao = [0 á, 3 âu] 🡪 E(Scao) = 0

\*G(S, giới) = E(S) - 5/8.E(Snam) - 3/8.E(Snữ) = **0.3475**

+ Snam = [3 á, 2 âu] 🡪 E(Snam) = -3/5.log(3/5) -2/5log(2/5) = 0.971

+ Snữ = [0 á, 3 âu] 🡪 E(Snữ) = 0

🡺 chọn chiều cao

Chiều cao

|\_thấp: á

|\_cao: âu

|\_TB: ?

**LẦN 2**

S = STB = [2 á, 2 âu] 🡪 E(STB) = 1

\*G(S, h.dáng) = 1 - 2/4.E(Sto) - 2/4.E(Snhỏ) = 0

+ Sto = [1 á, 1 âu] 🡪 E(Sto) = 1

+ Snhỏ = [1 á, 1 âu] 🡪 E(Snhỏ) = 1

🡺 G(S, h.dáng) = 0, mà chỉ còn cột giới tính nên chọn cột giới tính là node tiếp theo

+ Snam = [2 á, 0 âu]

+ Snữ = [0 á, 2 âu]

Chiều cao

|\_thấp: á

|\_cao: âu

|\_TB

|\_nam: á

|\_nữ: âu

\*Rút luật (thực chất là miêu tả cây):

# #ND THI

1. tìm kiếm heuristic

2. CSP

3. c/m logic: davis putnam, vương hạo, robinson

4. pp học máy: dectree, cây định danh, ILA, Quinland