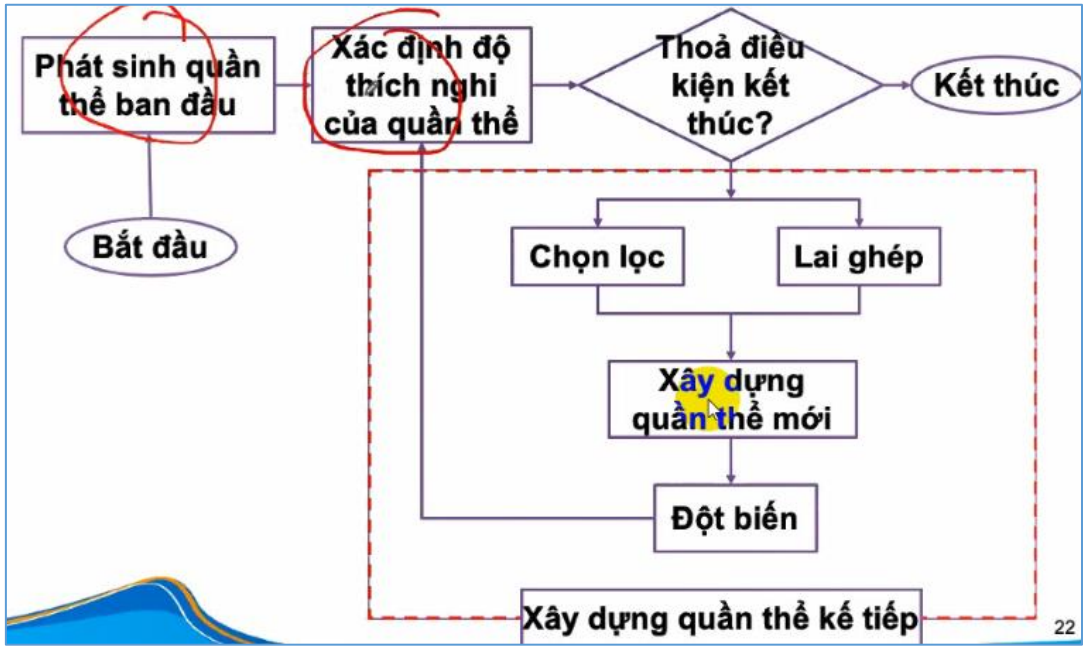
Mục lục

#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 thế 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

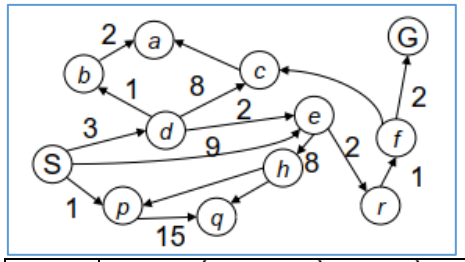
- implementation notes:

+ 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

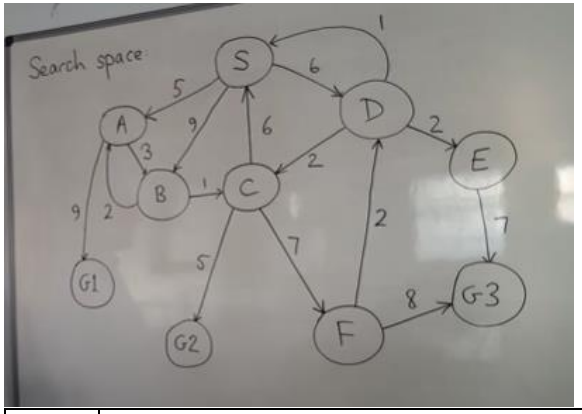
Ex1:



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 thể 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)

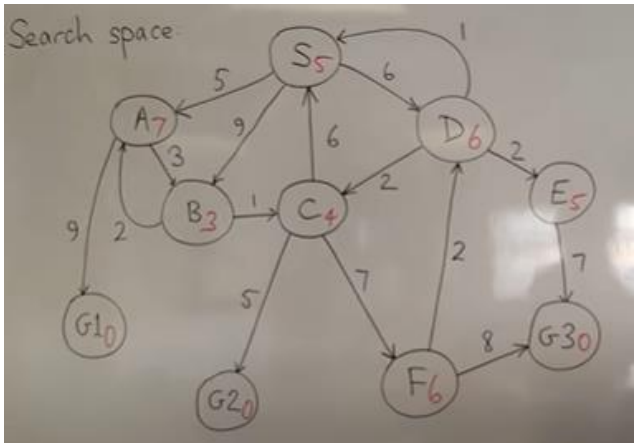
thuật giải A: kết hợp UCS + greedy → f(n) = h(n) + g(n)

+ **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: $h(n) \leq h^*(n)$

        với h*(n) là **chi phí thấp nhất** đến đích trong thực tế
    + nhất quán: với mỗi successor n' của n thì $h(n) \leq c(n,a,n') + h(n')$
- 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*

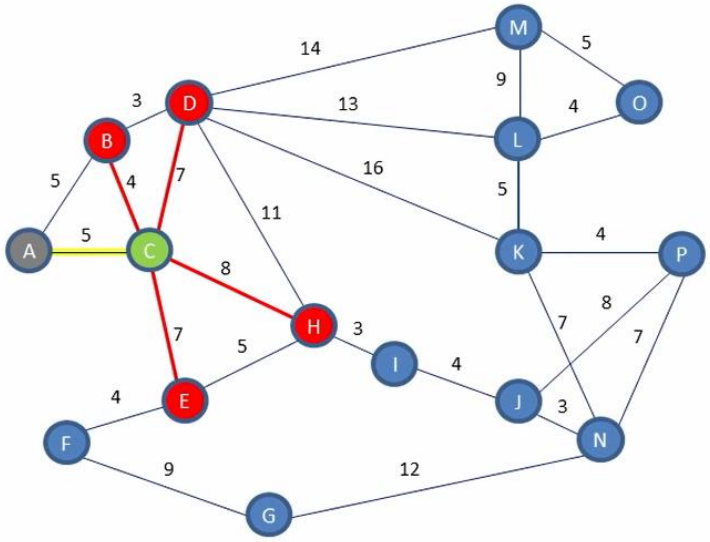
VD:



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

Open B C D H E

Closed A



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5 9	17	22 26	A C
C	5	13	18	A
D	12	16	28	C
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

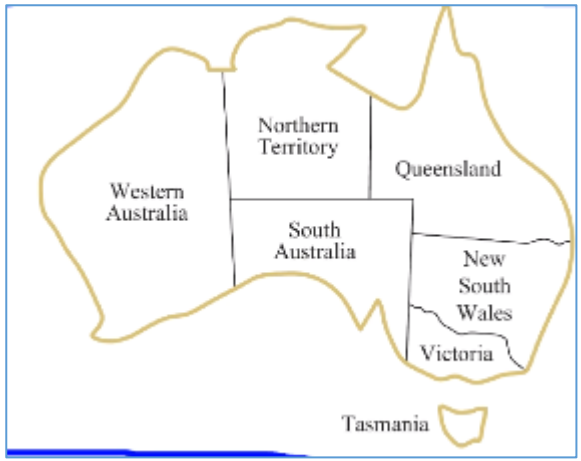
thuật giải IDA (iterative deepening A*): ?

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

#CSP: BÀI TOÁN THỎA MÃN RÀNG BUỘC

- Cần xác định:
  - Tập biến: $X = \{X_1,\ldots,X_n\}$
  - Tập miền gtri $D = \{D_1,\ldots,D_n\}$ với $D_i = \{v_1,\ldots, v_k\}$: miền gtri của biến X_i. **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

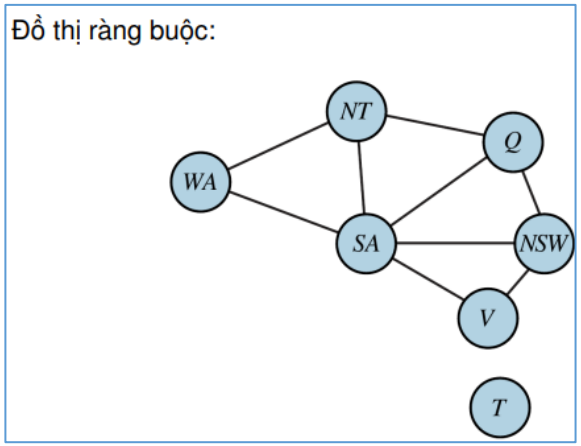


X = {WA; NT; Q; NSW; V; SA; T}

D_i = {R,G,B}

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

Đồ thị ràng buộc:



***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

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

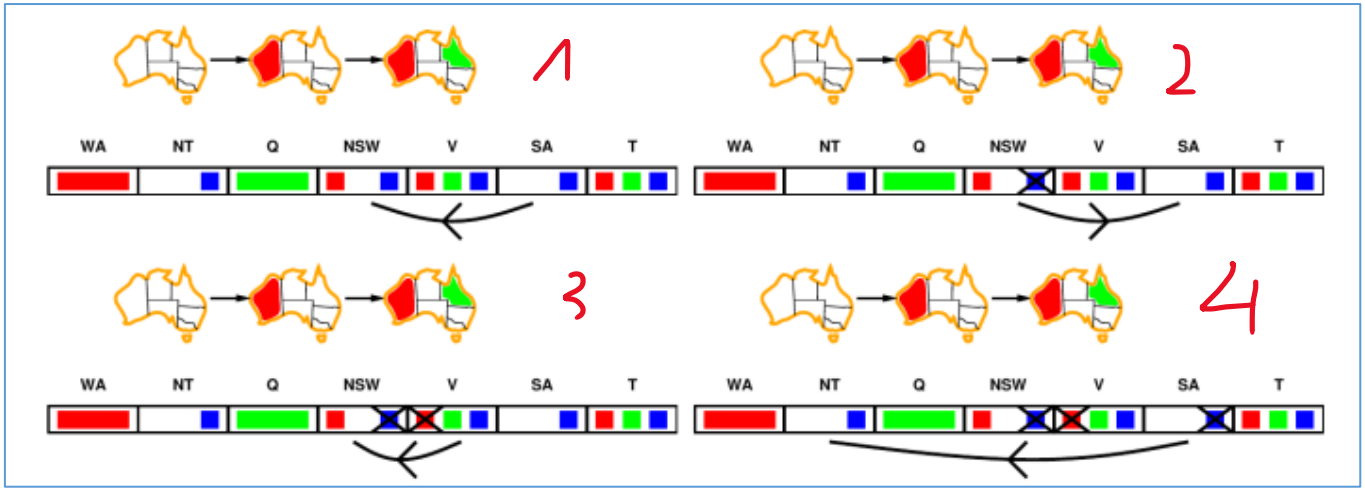
***1 số thuật toán có sdung 1 trong 3 heuristic trên để giải CSP**

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

  - i=0: trạng thái đầu, tất cả biến X (lãnh thổ) 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**

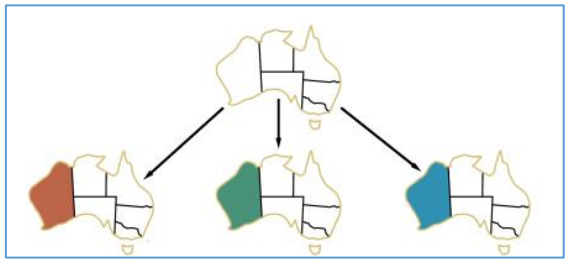
Một cạnh X → Y là hợp lệ (arc-consistency) khi ∀x thuộc D_X, ∃y lân cận x thuộc D_Y ko vi phạm ràng buộc. Nếu ∃y thuộc D_Y vi phạm ràng buộc thì cần loại gtri nào đó của y

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

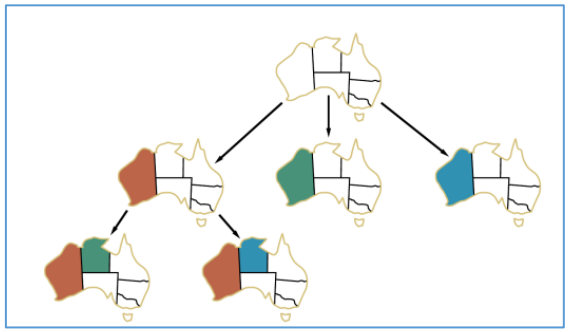


VD: x = SA thuộc{Blue} có y = NSW lân cận x = SA cũng trùng màu Blue với x → vi phạm ràng buộc nên ta loại bỏ gtri Blue của NSW. Mà khi NSW thay đổi thì biến (lãnh thổ) lân cận nó cũng cần xét lại, tức xét V kề NSW…(làm tương tự).

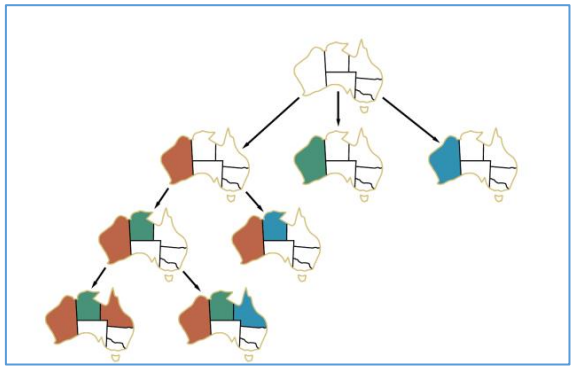
- Quay lui:
  - B1



  - B2



  - B3



  - Expand tương tự

#PP BIỂU DIỄN TRI THỨC

##LOGIC

Logic = Syntax + Semantics

- **Cú pháp (syntax)**: để xác định các mệnh đề (sentences) trong một ngôn ngữ
- **Ngữ nghĩa (semantics)**: để xác định "ý nghĩa" của các mệnh đề trong một ngôn ngữ
  - Xác định sự đúng đắn của một mệnh đề

Ví dụ: Trong ngôn ngữ của toán học

- (x+2 ≥ y) là một mệnh đề;
- (x+y > {}) không phải là một mệnh đề

- 1 số ký hiệu

+ I(x): ngữ nghĩa/ý nghĩa diễn giải của x. VD:
  - *I*(*one*) nghĩa là 1 (∈ N)
  - *I*(*plus*) nghĩa là phép cộng +: N x N → N
  - *I*(*equal*) nghĩa là phép so sánh bằng =: N x N → {*true*, *false*}
  - *I*(*one plus one equal two*) nghĩa là *true*

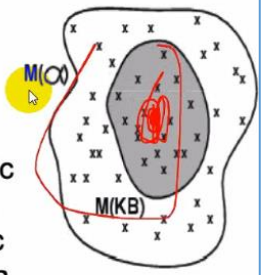
+ A |= B: A "bao hàm" B hay B được chứa bởi A. **DK bao hàm**: nếu A đúng thì B cũng phải đúng

*CÁC ĐỊNH NGHĨA CẦN NHỚ

a) định nghĩa **mô hình**:

CÁC MÔ HÌNH

- Các nhà logic học thường hay xem xét các sự việc theo các mô hình (models)
- Các mô hình là các không gian (thế giới) có cấu trúc, mà trong các không gian đó tính đúng đắn (của các sự việc) có thể đánh giá được
- ***Định nghĩa***: *m* là một mô hình của mệnh đề α nếu α là đúng trong *m*
- *M(α)* là tập hợp tất cả các mô hình của α
- KB ⊨ α nếu và chỉ nếu *M(KB)* ⊆ *M(α)*
  - Ví dụ: *KB* = "*Đội Brasil đã thắng*" và "*Đội Argentina đã thắng*", *α* = "*Đội Brasil đã thắng.*



b) định nghĩa **câu** (sentence)

Câu (sentence) (còn gọi là **well-formed formulas - WFF**)

- true và false là các câu
- Các biến mệnh đề là các câu: P, Q, R, Z...
- Nếu α, β là các câu thì ¬α, α ∧ β, α ∨ β, α ⇒ β, α ⇔ β cũng là các câu
- Ngoài ra, không có một câu nào nữa.

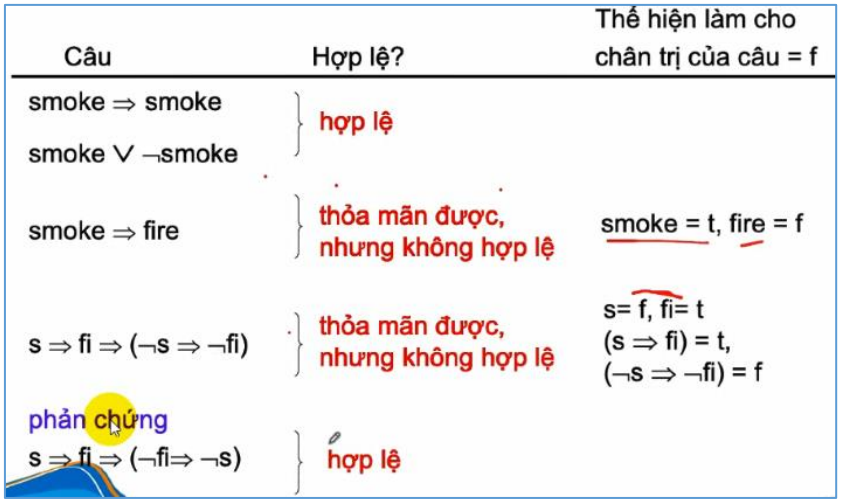
+ T/c của câu

- Một câu là **hợp lệ** nếu và chỉ nếu chân trị của nó là **t** trong tất cả thể hiện
  - Câu hợp lệ: true, ¬false, P ∨ ¬P
- Một câu là **thỏa mãn được** nếu và chỉ nếu chân trị của nó là ***t*** trong ít nhất một thể hiện
  - Câu thỏa mãn được: P, true, ¬P
- Một câu là **không thỏa mãn được** nếu và chỉ nếu chân trị của nó là ***f*** trong tất cả thể hiện
  - Câu không thỏa mãn được: P ∧ ¬P, false, ¬true
- Tất cả các câu trong logic mệnh đề đều quyết định được.

Bsung: phản chứng thì luôn hợp lệ

P	Q	¬P	P∧Q	P∨Q	P⇒Q	Q⇒P	P⇔Q
f	f	t	f	f	t	t	t
f	t	t	f	t	t	f	f
t	f	f	f	t	f	t	f
t	t	f	t	t	t	t	t

VD:



Câu	Hợp lệ?	Thế hiện làm cho chân trị của câu = f
smoke ⇒ smoke smoke ∨ ¬smoke	hợp lệ	
smoke ⇒ fire	thỏa mãn được, nhưng không hợp lệ	smoke = t, fire = f
s ⇒ fi ⇒ (¬s ⇒ ¬fi)	thỏa mãn được, nhưng không hợp lệ	s= f, fi= t (s ⇒ fi) = t, (¬s ⇒ ¬fi) = f
phản chứng s ⇒ fi ⇒ (¬fi⇒ ¬s)	hợp lệ	

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

Nghĩa của một câu là một chân trị {t, f}.

Thế hiện là việc gán chân trị cho các biến mệnh đề

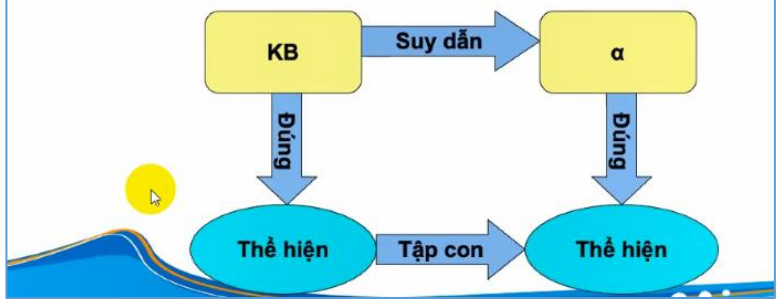
- *holds*(α,*i*) [câu α là **t** trong thể hiện *i*]
  [câu α đúng trong thể hiện *i*]
- *fails*(α,*i*) [câu α là **f** trong thể hiện *i*]
  [câu α sai trong thể hiện *i*]

Thể hiện *i* dưới dạng bảng tra, P là biến mệnh đề:

- holds(P, *i*) *iff* *i*(P) = t
- fails(P, *i*) *iff* *i*(P) = f

d) định nghĩa "**suy dẫn**"

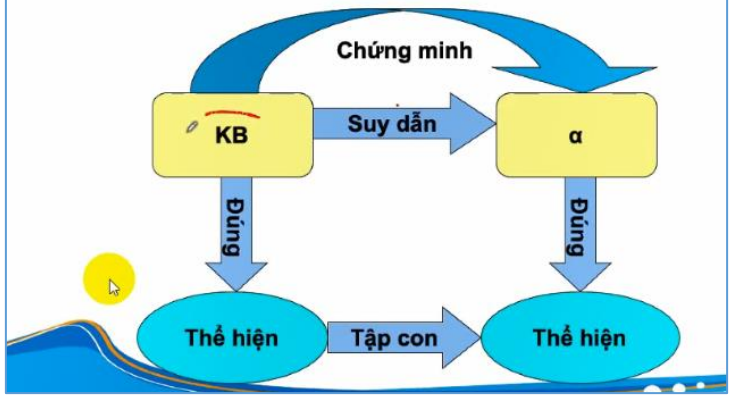
- Một cơ sở tri thức (KB) **suy dẫn** (entails) một câu α nếu và chỉ nếu mọi thể hiện làm cho KB đúng cũng làm cho α đúng.
- Ký hiệu: KB ⊨ α



đại khái ta phải tìm toàn bộ thể hiện làm cho KB đúng → rất nhiều thể hiện → tốn time duyệt

e) định nghĩa "**chứng minh**"

Chứng minh là cách kiểm tra xem một KB có suy dẫn một câu α hay không mà không cần liệt kê tất cả các thể hiện có thể.



- Một chứng minh là một chuỗi các câu
- Các câu đầu tiên là các tiền đề (KB)
- Sau đó, ta có thể viết được dòng kế tiếp là kết quả của việc áp dụng một luật suy diễn lên dòng trước.
- Khi *α* xuất hiện trên dòng, ta đã chứng minh *α* từ KB

- Nếu các luật suy diễn là đúng, thì bất kỳ α có thể chứng minh từ KB cũng suy dẫn được bởi KB
- Nếu các luật suy diễn là đủ, thì bất kỳ α nào có thể được suy dẫn bởi KB cũng có thể được chứng minh từ KB

+ luật suy diễn

Modus Ponens	Modus Tolens	And-Introduction	And-Elimination
$\alpha \Rightarrow \beta$, α / β	$\alpha \Rightarrow \beta$, $\neg\beta$ / $\neg\alpha$	α, β / $\alpha \wedge \beta$	$\alpha \wedge \beta$ / α

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

Bước	Công thức	Nguồn gốc
1	$P \wedge Q$	Cho trước
2	$P \Rightarrow R$	Cho trước
3	$Q \wedge R \Rightarrow S$	Cho trước
4	P	1 And-Elim
5	R	4,2 Modus Ponens
6	Q	1 And-Elim
7	$Q \wedge R$	5,6 And-Intro
8	S	3,7 Modus Ponens

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:

$\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$ (điều kiện, kéo theo)

tiền đề $\Rightarrow$ kết luận

$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ (tương đương)

a) Mệnh đề horn

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

- Mệnh đề Horn là nối rời của các literal sao cho có tối đa một literal là khẳng định.
  - $\neg A \vee \neg B \vee C$ là mệnh đề horn,
  - $\neg A \vee B \vee C$ không phải là mệnh đề horn.
- Mệnh đề Horn thường được biểu diễn thành dạng luật có tiền đề là nối liền các literal dương và kết quả là một literal dương đơn.
  - $\neg A \vee \neg B \vee C \equiv A \wedge B \Rightarrow C$
- Khả năng biểu diễn của mệnh đề Horn bị giới hạn.

- KB dạng Horn = nối liền các mệnh đề Horn
  - Mệnh đề Horn = (biến mệnh đề), hay, (nối liền các biến) ⇒ biến. Ví dụ: $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow D)$

- Qui tắc suy diễn: Modus Ponens – đầy đủ đối với KB dạng Horn

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

  - Suy diễn trên mệnh đề Horn được thực hiện bằng phương pháp suy diễn tiến và suy diễn lùi.
  - Các thuật toán này rất tự nhiên và chạy với thời gian tuyến tính.

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

- Suy diễn tiến (Forward chaining) và suy diễn lùi (Backward chaining) được áp dụng lên các biểu thức dạng Horn
- Biểu thức dạng Horn: trong biểu thức có nhiều nhất một literal khẳng định

$$p_1 \vee \neg p_2 \vee \neg p_3 \vee \ldots \vee \neg p_n$$

- Hay dạng luật (luật sinh)

$$p_2 \wedge p_3 \wedge \ldots \wedge p_n \Rightarrow p_1$$

```
FOL-FC-Ask(KB,α) {
  repeat until new là rỗng
    new ← {}
    for each câu r trong KB  // r ở dạng chuẩn hóa (p1 ∧... ∧ pn ⇒ q)
      for each phép thế θ sao cho (p1 ∧... ∧ pn)θ = (p'1 ∧... ∧ p'n)θ
      với p'1,..., p'n nào đó trong KB
            q' ← Subst(θ,q)
        if q' không phải là một câu đã có trong KB hay new then
            thêm q' vào new
            ϕ ← Unify(q', α)
            if ϕ thành công then return ϕ
    thêm new vào KB
  return false
}
```

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:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

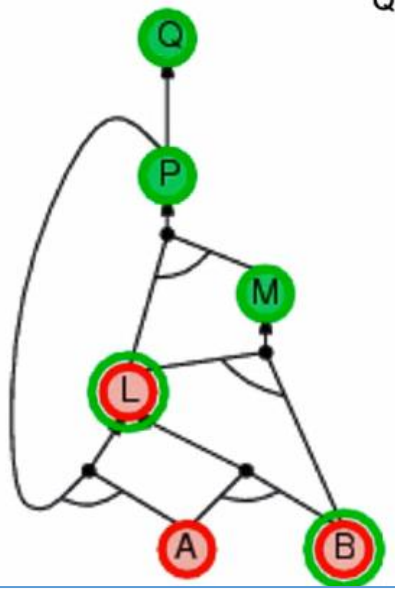
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Q?
P?
L? ✓
A? ✓
B? ✓
M? L ∧ B ⇒ M
L?
B?

P ⇒ Q
L ∧ M ⇒ P

ĐẶC ĐIỂM CỦA SUY DIỄN LÙI

- Tìm kiếm chứng minh bằng cách đệ qui theo chiều sâu: không gian tuyến tính theo kích thước của chứng minh
- Không đầy đủ do lặp vô tận
  - Giải pháp: Kiểm tra trạng thái hiện tại với mọi trạng thái đang có trong stack
- Không hiệu quả do các mục tiêu con bị lặp lại (cả khi thất bại cũng như thành công)
  - Giải pháp: dùng bộ nhớ tạm lưu các mục tiêu con đã duyệt
- Được dùng nhiều trong lập trình logic (ngôn ngữ Prolog)

```
FOL-BC-ASK(KB, goals, θ) {
    Inputs: KB, cơ sở tri thức
        goals, danh sách dưới dạng nối liền của một câu truy vấn
        θ, phép thế hiện tại, được khởi tạo rỗng {}
    biến cục bộ: ans, một tập các phép thế, được khởi tạo rỗng

    if goals rỗng then return {θ}
    q' ← SUBST(θ, first(goals))
    for each r trong KB mà r có dạng chuẩn (p1 ∧ ... ∧ pn ⇒ q)
          và θ' ← UNIFY(q, q') thành công
        ans ← FOL-BC-ASK(KB, [p1,...,pn| REST(goals)], θ ∪ θ')) ∪ ans
    return ans
}
```

d) 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**

- Biểu thức Dạng hội Chuẩn (CNF) có dạng:

  $(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$

  - $(A \vee B \vee \neg C)$ là một mệnh đề
  - A, B, $\neg C$ là literal – biến hay phủ định của biến
  - Mỗi mệnh đề phải được thoả và có thể được thoả theo nhiều cách
  - Mọi câu trong logic mệnh đề đều có thể viết dưới dạng CNF

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

1. Biến đổi tất cả các câu thành dạng CNF
2. Lấy phủ định kết luận, đưa vào KB
3. Lặp
   a. Nếu trong KB có chứa hai mệnh đề phủ định nhau (p và ¬p) thì trả về ***false***
   b. Nếu có hai mệnh đề chứa các literal phủ định nhau thì áp dụng hợp giải.
   c. Lặp cho đến khi không thể áp dụng tiếp luật hợp giải.
4. Trả về ***true***

VD 1:

Chứng minh R

1	$P \vee Q$
2	$P \Rightarrow R$
3	$Q \Rightarrow R$

Bước	Công thức	Suy dẫn
1	$P \vee Q$	Cho trước
2	$\neg P \vee R$	Cho trước
3	$\neg Q \vee R$	Cho trước
4	$\neg R$	Phủ định kết luận
5	$Q \vee R$	1, 2
6	$\neg P$	2, 4
7	$\neg Q$	3, 4
8	R	5, 7
9	•	4, 8

có 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

Chứng minh $A \Rightarrow F$

1	$A \Rightarrow (B \vee C)$
2	$B \Rightarrow (D \vee F)$
3	$A \wedge D \Rightarrow F$
4	$C \Rightarrow F$

- ☐ $\neg A \vee B \vee C, \neg B \vee D \vee F, \neg A \vee \neg D \vee F, \neg C \vee F, A, \neg F$
- ☐ $B \vee C, \neg B \vee D \vee F, \neg D \vee F, \neg C \vee F, \neg F$ (Hợp giải biến A)
- ☐ $C \vee D \vee F, \neg D \vee F, \neg C \vee F, \neg F$ (Hợp giải biến B)
- ☐ $D \vee F, \neg D \vee F, \neg F$ (Hợp giải biến C)
- ☐ $F, \neg F$ (Hợp giải biến D)

Chứng minh được vì có chứa hai mệnh đề phủ định nhau

**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

  $GT1, GT2, ..., GTn \rightarrow KL1, KL2, ... KL_m$

- B2: **Chuyển vế** các GT_i và KL_j (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	$GT1,...,a \vee b,...,GTn \rightarrow KL1, KL2,...,KLm$	$\begin{cases} GT1,...,a,...,GTn \rightarrow KL1, KL2,...,KLm \\ GT1,...,b,...,GTn \rightarrow KL1, KL2,...,KLm \end{cases}$
2	$GT1,...,GTn \rightarrow KL1, KL2,... \ a \wedge b,...,KLm$	$\begin{cases} GT1,...,GTn \rightarrow KL1, KL2,... \ a,...,KLm \\ GT1,...,GTn \rightarrow KL1, KL2,... \ b,...,KLm \end{cases}$

- 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 'V' 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

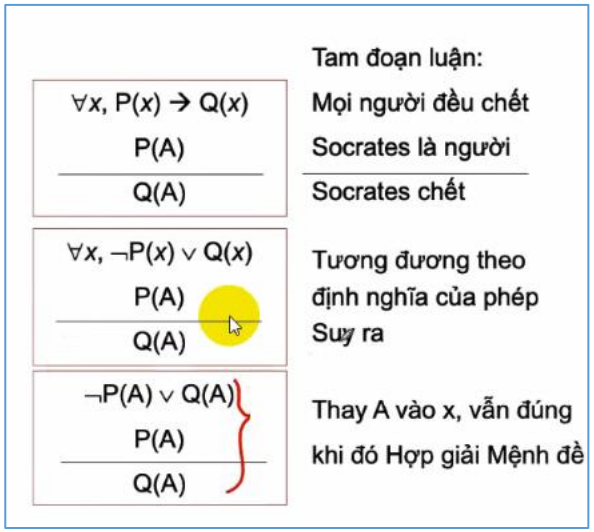
= Ko dc c/m

(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/

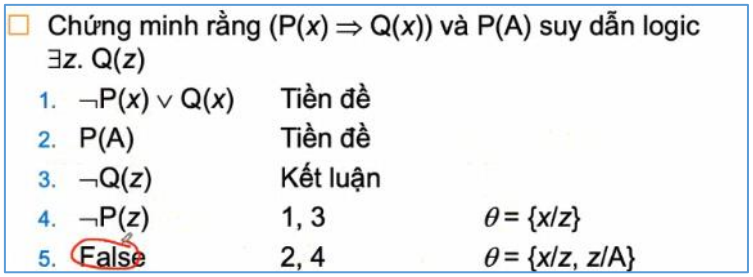
##LOGIC BẬC NHẤT
- tên riêng được coi là hằng (const). VD: Lan, John
- biểu diễn: verb(verb's main subject, O).
  VD: Cháu(x,y): x là cháu của y
- lượng từ 'với mọi' (thường đi kèm với dấu kéo theo): ∀x. P
  Ex: Sinh viên CNTT thì thông minh → ∀x. Sinh-viên(x,CNTT) ➔ thông-minh(x)
- lượng từ 'tồn tại' (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)

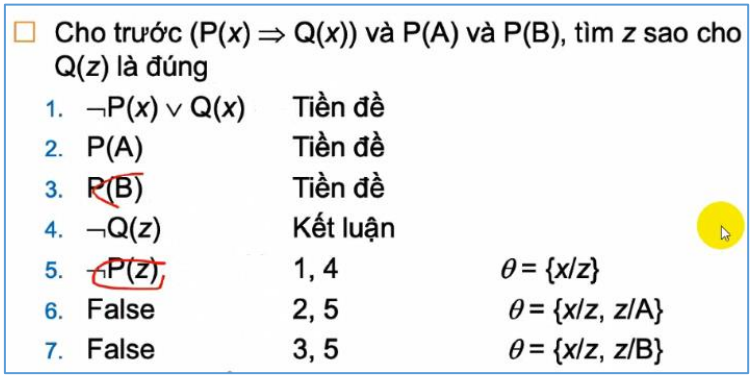


thế A vào x(*) xong hợp giải, VD như hợp giải !P(x) V P(A) thì sẽ thành 1
→ Thế nào là 1 phép thế đúng đắn?

VD 1:



VD 2:



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)

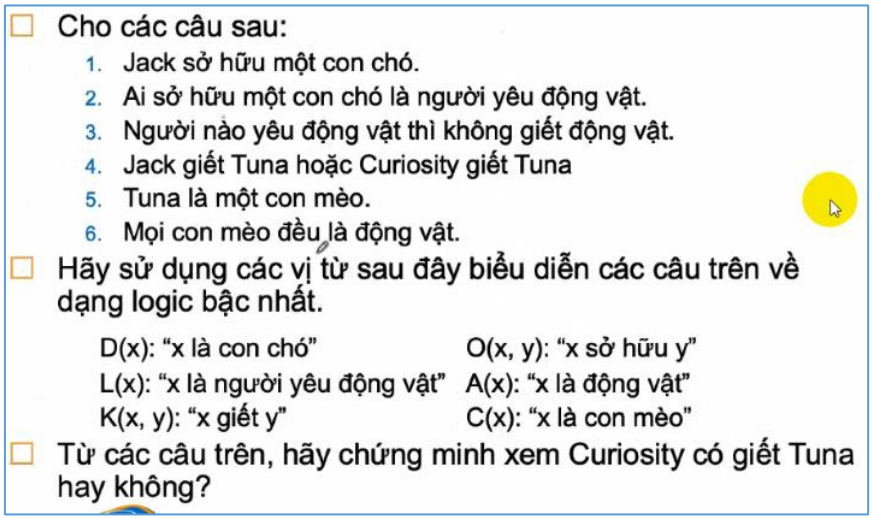
Hỏi: Các cặp ông cháu?

→ ∀ 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:



1. $\exists x.\ D(x) \wedge O(Jack,x)$
   $= D(A) \wedge O(Jack,A)$ (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 "∧"
2. $\forall x.\ (\exists y.\ D(y) \wedge O(x, y)) \rightarrow L(x)$
   $= \forall x.\ \neg(\exists y.\ D(y) \wedge O(x, y)) \vee L(x)$
   $= \forall x.\ \forall y.\ \neg D(y) \vee \neg O(x, y) \vee L(x)$
   $= \neg D(y) \vee \neg O(x, y) \vee L(x)$ (luật 5 - bỏ ∀)
3. $\forall x.\ L(x) \rightarrow (\forall y.\ A(y) \rightarrow \neg K(x,y))$ ➔ vì là lượng từ với mọi nên dùng kéo theo
   $= \forall x.\ \neg L(x) \vee\ (\forall y.\ \neg A(y) \vee \neg K(x,y))$
   $= \neg L(x) \vee\ \neg A(y) \vee \neg K(x,y)$ (luật 5 - bỏ ∀)
4. $K(Jack,Tuna) \vee K(Curiosity,Tuna)$
5. $C(Tuna)$
6. $\forall x.\ C(x) \rightarrow A(x)$
   $= \neg C(x) \vee A(x)$

GOAL

7. 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
9	*K(Jack, Tuna)*	*5, 8*
10	*A(Tuna)*	*6, 7 {x/Tuna}*
11	*¬L(Jack) ∨ ¬A(Tuna)*	*4, 9 {x/Jack, y/Tuna}*
12	*¬L(Jack)*	*10, 11*
13	*¬D(y) ∨ ¬O(Jack, y)*	*3, 12 {x/Jack}*
14	*¬D(A)*	*2, 13 {y/A}*
15	•	*14, 1*

*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

1. Loại bỏ các dấu mũi tên

$$\alpha \leftrightarrow \beta \Rightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$
$$\alpha \rightarrow \beta \Rightarrow \neg\alpha \vee \beta$$

2. Phân phối phủ định

$$\neg\neg\alpha \Rightarrow \alpha$$
$$\neg(\alpha \vee \beta) \Rightarrow \neg\alpha \wedge \neg\beta$$
$$\neg(\alpha \wedge \beta) \Rightarrow \neg\alpha \vee \neg\beta$$
$$\neg\forall x.\ \alpha \Rightarrow \exists x.\neg\alpha$$
$$\neg\exists x.\ \alpha \Rightarrow \forall x.\neg\alpha$$

3. Đổi tên các biến thành phần

$$\forall x.\exists y.(\neg P(x) \vee \exists x.\ Q(x,y)) \Rightarrow \forall x_1.\exists y_2.(\neg P(x_1) \vee \exists x_3.\ Q(x_3,y_2))$$

4. Skolem hoá (Skolemization)
   - thay tên mới cho tất cả lượng từ tồn tại

$$\exists x.\ P(x) \Rightarrow P(Lan)$$
$$\exists x,y.\ R(x,y) \Rightarrow R(Thing1, Thing2)$$
$$\exists x.\ P(x) \wedge Q(x) \Rightarrow P(Fleep) \wedge Q(Fleep)$$
$$\exists x.\ P(x) \wedge \exists x.\ Q(x) \Rightarrow P(Frog) \wedge Q(Grog)$$
$$\exists y, \forall x.\ Loves(x,y) \Rightarrow \forall x.\ Loves(x, Englebert)$$

   - thay hàm mới cho tất cả các lượng từ tồn tại ở tầm vực với mọi

$$\forall x\ \exists y.\ Loves(x,y) \Rightarrow \forall x.Loves(x, beloved(x))$$

5. Bỏ các lượng từ với mọi

$$\forall x.\ \exists y\ Loves(x,y) \Rightarrow Loves(x, beloved(x))$$

6. Phân phối *or* vào *and;* trả về các mệnh đề

$$P(z) \vee (Q(z,w) \wedge R(w,z)) \Rightarrow \{P(z) \vee Q(z,w),\ P(z) \vee Q(w,z)\}$$

7. Đổi tên các biến trong từng mệnh đề

$$\{P(z) \vee Q(z,w),\ P(z) \vee Q(w,z)\} \Rightarrow$$
$$\{P(z_1) \vee Q(z_1,w_1),\ P(z_2) \vee Q(w_2,z_2)\}$$

**Phép thế

- P(x, f(y), B): một câu nguyên tố

Các thể hiện	Phép thế $\{v_1/t_1, v_2/t_2 \ldots\}$	Ghi chú
P(z, f(w), B)	{x/z, y/w}	Đổi tên biến
P(x, f(A), B)	{y/A}	
P(g(z), f(A), B)	{x/g(z), y/A}	
P(C, f(A), B)	{x/C, y/A}	Phép thế cơ sở

- Áp dụng một phép thế

P(x, f(y), B) {y/A} = P(x, f(A), B)

P(x, f(y), B) {y/A, x/y} = P(A, f(A), B)

**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

- Hai biểu thức ω_1 và ω_2 là đồng nhất được (unifiable) khi vào chỉ khi tồn tại thế s sao cho

  $\omega_1 s = \omega_2 s$

- Gọi ω1 = *x* và ω2 = *y*, đây là các phép đồng nhất:

s	$\omega_1 s$	$\omega_2 s$
{*y* / *x*}	*x*	*x*
{*x* / *y*}	*y*	*y*
{*x* / f(f(A)), *y* / f(f(A))}	f(f(A))	f(f(A))
{*x*/A, *y*/A}	A	A

+ $w_1 s$: thay s vào biểu thức w_1
+ 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**: $f(t_1,\ldots,t_n)/f(u_1,\ldots,u_n)$. 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: $x/f(t_1,\ldots,t_n) = f(t_1,\ldots,t_n)/x$. Nếu x nằm trong input của $f(t_1,..x..,t_n)$ → ko có MGU

- g là phép đồng nhất tổng quát nhất (most general unifier - MGU) của ω_1 và ω_2 khi và chỉ khi với mọi phép đồng nhất s, tồn tại s' sao cho $\omega_1.s = (\omega_1.g)s'$

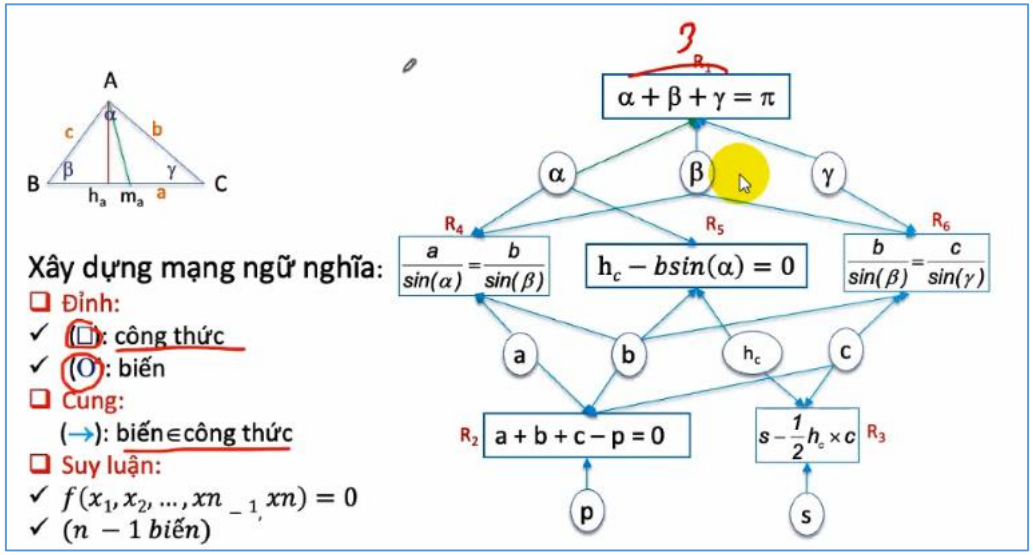
ω_1	ω_2	MGU
P(*x*)	P(A)	{*x*/A}
P(f(*x*), *y*, g(*x*))	P(f(*x*), *x*, g(*x*))	{*y*/*x*} hay {*x*/*y*}
P(f(*x*), *y*, g(*y*))	P(f(*x*), *z*, g(*x*))	{*y*/*x*, *z*/*x*}
P(*x*, B, B)	P(A, *y*, *z*)	{*x*/A, *y*/B, *z*/B}
P(g(f(*v*)), g(*u*))	P(*x*, *x*)	{*x*/g(f(*v*)), *u*/f(*v*)}
P(*x*, f(*x*))	P(*x*, *x*)	Không có MGU!

VD:

☐ Hãy tìm MGU cho các cặp câu sau

ω_1	ω_2	MGU
A(B,C)	A(*x*, *y*)	{*x*/B, *y*/C)
A(*x*, f(D,*x*))	A(E, f(D, *y*))	{*x*/E, *y*/E}
A(*x*, *y*)	A(f(C, *y*), *z*)	{*x*/f(C,*y*), *y*/*z*}
P(A, *x*, f(g(*y*)))	P(*y*, f(*z*), f(*z*))	{*y*/A, x/f(z), z/g(y)}
P(*x*, g(f(A)), f(*x*))	P(f(*y*), *z*, *y*)	Không có MGU
P(*x*, f(*y*))	P(*z*, g(*w*))	Không có MGU

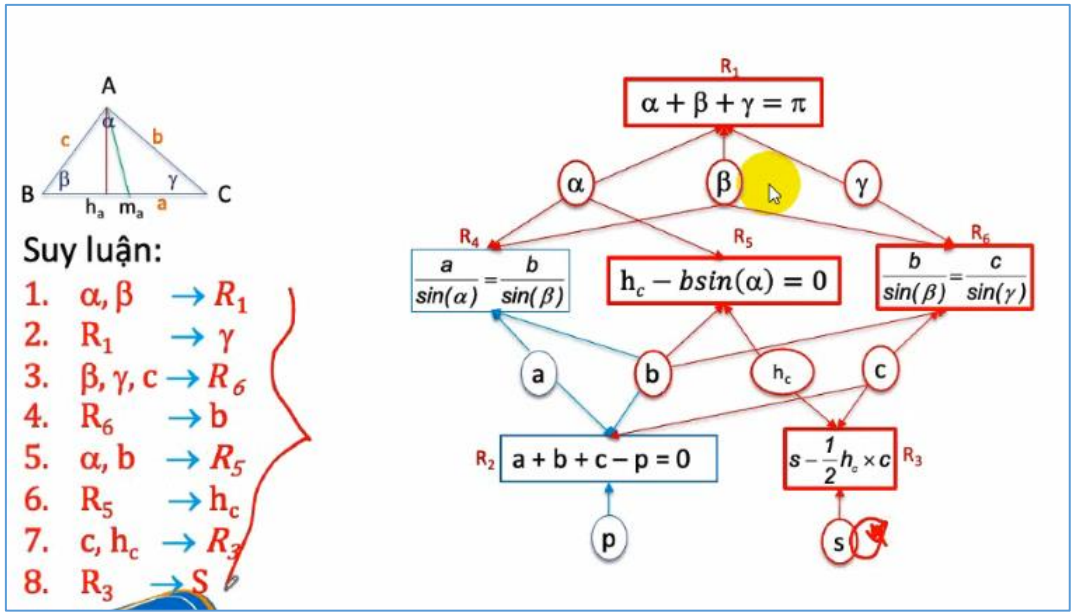
#MẠNG NGỮ NGHĨA

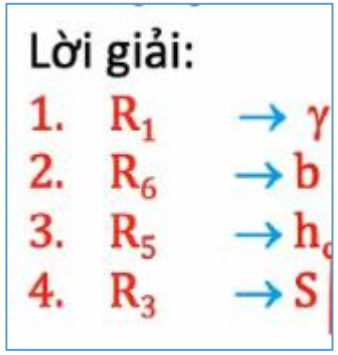


- Sau khi dựng xong MNN, đề cho biến nào thì **kích hoạt** biến đó trong đồ thị. Biến dc kích hoạt sẽ truyền động ra mọi nhánh nối vs nó đến các đỉnh neighbor
- Khi truyền động tới 1 đỉnh (ở đây là CT) bất kỳ, nếu CT có n-1 biến dc xác định (hay 'kích hoạt')
  → biến còn lại trong CT dc **auto kích hoạt**
  → 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

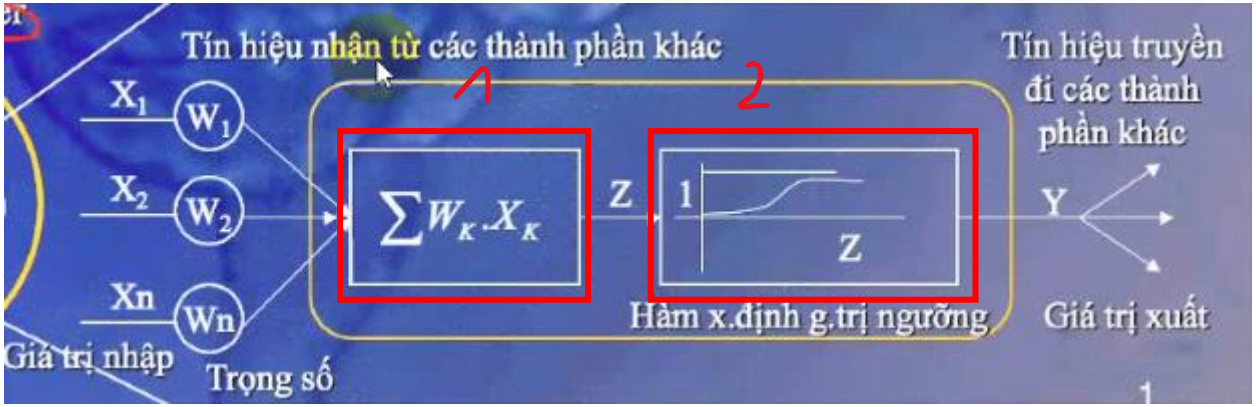
Cho trước alpha, beta, c → tính diện tích tam giác



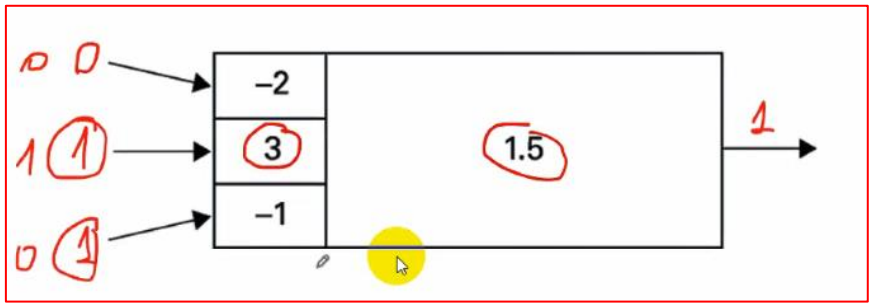


lời giải bài toán là những suy luận có VT là 1 CT

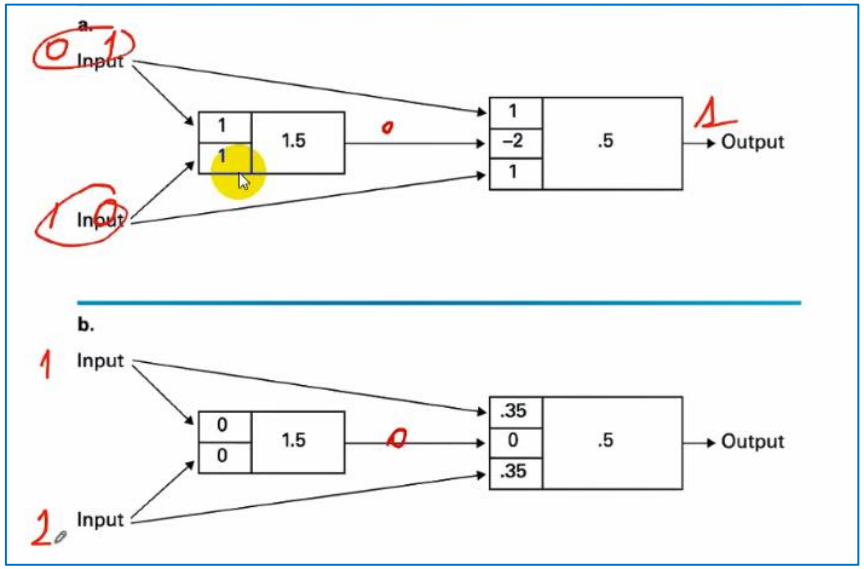
#MẠNG NEURAL



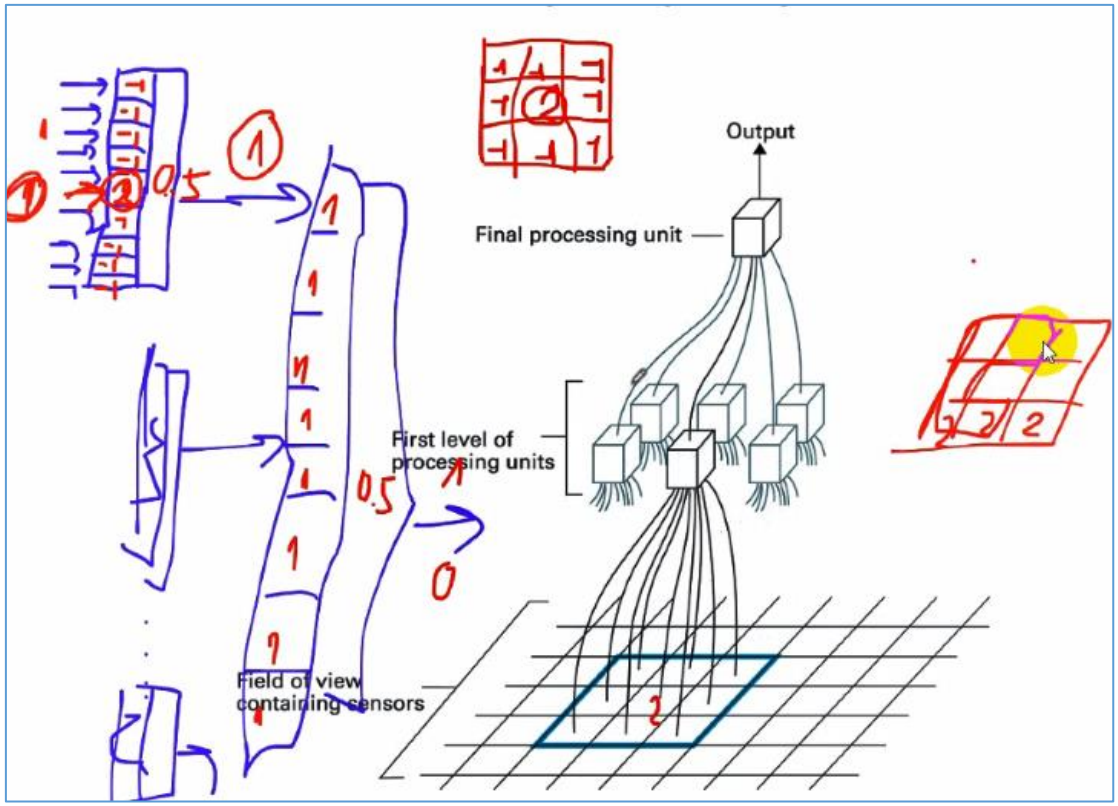
(1): gọi là **hàm kích hoạt** = tổng của(tích giữa các input x_i & trọng số w_i)
(2): ngưỡng, thuộc [0,1], sẽ dc cho trước bởi 1 hàm
- input X & output Y chỉ nhận giá trị 0 or 1
- nếu hàm kích hoạt > ngưỡng thì output Y=1; ngược lại output Y=0
- khái niệm 'lớp':
+ lớp nhập: là W, ko tính X
+ lớp ẩn: 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



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



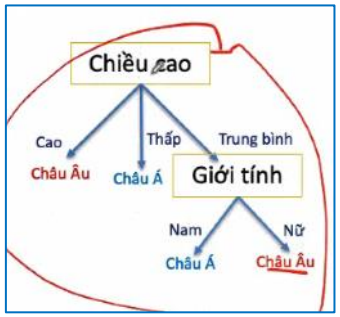
*Nhận dạng chữ C & T



#BT ÔN
Dùng ID3 cho bảng sau:

- Ví dụ:
  - Xác định là người Châu Á hay Châu Âu khi xem xét một nhóm người căn cứ trên: hình dáng, chiều và giới tính
  - Giả sử có Bảng quan sát sau:

STT	HÌNH DÁNG	CHIỀU CAO	GIỚI TÍNH	QUAN SÁT
1	To	Trung bình	Nam	Châu Á
2	Nhỏ	Thấp	Nam	Châu Á
3	Nhỏ	Trung bình	Nam	Châu Á
4	To	Cao	Nam	Châu Âu
5	Nhỏ	Trung bình	Nữ	Châu Âu
6	Nhỏ	Cao	Nam	Châu Âu
7	Nhỏ	Cao	Nữ	Châu Âu
8	To	Trung bình	Nữ	Châu Âu



Nếu Chiều cao = Cao Thì Châu Âu
Nếu Chiều cao = Thấp Thì Châu Á
Nếu Chiều cao = T. bình và G. tính = Nam Thì Châu Á
Nếu Chiều cao = T. bình và G. tính = Nữ Thì Châu Âu

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(S_{to}) - 5/8.E($S_{nhỏ}$) = **$3.16*10^{-3}$**

+ S_{to} = [1 á, 2 âu] → E(S_{to}) = -1/3.log(1/3) -2/3.log(2/3) = 0.9183

+ $S_{nhỏ}$ = [2 á, 3 âu] → E($S_{nhỏ}$) = -2/5.log(2/5) - 3/5.log(3/5) = 0.971

*G(S, c.cao) = E(S) - 4/8.E(S_{TB}) - 1/8.E($S_{thấp}$) -3/8.E(S_{cao}) = **0.4544**

+ S_{TB} = [2 á, 2 âu] → E(S_{TB}) = 1

+ E($S_{thấp}$) = 0

+ S_{cao} = [0 á, 3 âu] → E(S_{cao}) = 0

*G(S, giới) = E(S) - 5/8.E(S_{nam}) - 3/8.E($S_{nữ}$) = **0.3475**

+ S_{nam} = [3 á, 2 âu] → E(S_{nam}) = -3/5.log(3/5) -2/5log(2/5) = 0.971

+ $S_{nữ}$ = [0 á, 3 âu] → $E(S_{nữ}) = 0$
→ chọn chiều cao

Chiều cao
|_thấp: á
|_cao: âu
|_TB: ?

LẦN 2
S = S_{TB} = [2 á, 2 âu] → $E(S_{TB}) = 1$
*G(S, h.dáng) = 1 - 2/4.$E(S_{to})$ - 2/4.$E(S_{nhỏ})$ = 0
+ S_{to} = [1 á, 1 âu] → $E(S_{to}) = 1$
+ $S_{nhỏ}$ = [1 á, 1 âu] → $E(S_{nhỏ}) = 1$
→ G(S, h.dáng) = 0, mà chỉ còn cột giới tính nên chọn cột giới tính là node tiếp theo
+ S_{nam} = [2 á, 0 âu]
+ $S_{nữ}$ = [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