



Môn học

Trí Tuệ Nhân Tạo

Giảng Viên:

Phạm Minh Tuấn
(Lưu hành nội bộ)



Giới thiệu

- ❖ Phạm Minh Tuấn
- ❖ E-mail: pmtuan@dut.udn.vn
- ❖ Tel: 0913230910
- ❖ Khoa Công nghệ thông tin – Trường ĐHBK – ĐHĐN



Thỏa thuận

❖ Đối với giáo viên:

- Dạy đủ tất cả nội dung của môn học.
- Trả lời các câu hỏi của học sinh trong và ngoài giờ giảng liên quan tới môn học.
- Ra bài tập cho học sinh
- Lên lớp đúng giờ



Thỏa thuận (tiếp)

❖ Đối với học sinh:

- Tham gia trên 80% số tiết học.
- Tham gia đóng góp tiết học như phát biểu, trả lời hay đặt câu hỏi cho giáo viên (không nói chuyên riêng)
- Làm bài tập đầy đủ.
- Lên lớp đúng giờ (không được đi trễ hơn giáo viên quá 5 phút)



Nội dung môn học

❖ Chương 1: Giới thiệu

- Trí tuệ nhân tạo là gì?
- Nền tảng của ngành Trí tuệ nhân tạo
- Lịch sử AI
- Ứng dụng

❖ Chương 2: Suy luận Logic

- Logic mệnh đề
- Logic vị từ



Nội dung môn học

❖ Chương 3:Tìm kiếm trên không gian trạng thái

(State Space Search)

- AI : Biểu diễn và tìm kiếm
- Các giải thuật tìm kiếm trên không gian trạng thái
- Depth first search (DFS) - Breath first search (BFS)

❖ Chương 4:Tìm kiếm theo Heuristic

- Heuristic là gì?
- Tìm kiếm theo heuristic
- Các giải thuật Best first search (BFS), Giải thuật A*
- Chiến lược Minimax, Alpha Beta



Nội dung môn học

❖ Chương 5:Hệ luật sinh

- Tìm kiếm đệ qui
- Hệ luật sinh: Định nghĩa và ứng dụng
- Tìm kiếm trên hệ luật sinh

❖ Chương 6:Hệ chuyên gia

- Giới thiệu về hệ chuyên gia
- Mô hình hệ chuyên gia: dựa trên luật, dựa trên frame
- Phát triển một hệ chuyên gia

❖ Chương 7:Biểu diễn tri thức

- Biểu diễn tri thức trong AI: vai trò và ứng dụng
- Các kỹ thuật biểu diễn tri thức: semantic network, lưu đồ phụ thuộc khái niệm, frame, script



Nội dung môn học

❖ Chương 8: Thuật toán di truyền

- Thuyết tiến hóa
- Di truyền
- Thuật toán di truyền
- Ví dụ minh họa

❖ Chương 9: Mạng nơ ron nhân tạo

- Thuật toán rơi dốc nhanh nhất
- Bình phương nhỏ nhất
- Perceptron



I T F

Chương 1: GIỚI THIỆU

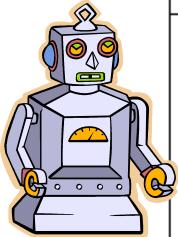
Trí tuệ nhân tạo là gì?



Thinking Humanly (suy nghĩ như người)

“The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense.” (Haugeland, 1985)

“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)



Acting Humanly (hành động như người)

“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)

“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)

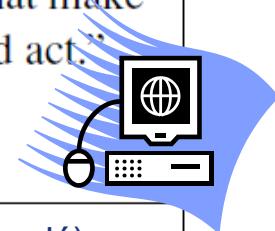
Thinking Rationally (suy nghĩ hợp lý)

“The study of mental faculties through the use of computational models.”

(Charniak and McDermott, 1985)

“The study of the computations that make it possible to perceive, reason, and act.”

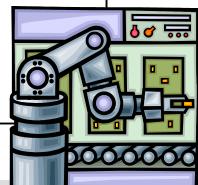
(Winston, 1992)



Acting Rationally (hành động hợp lý)

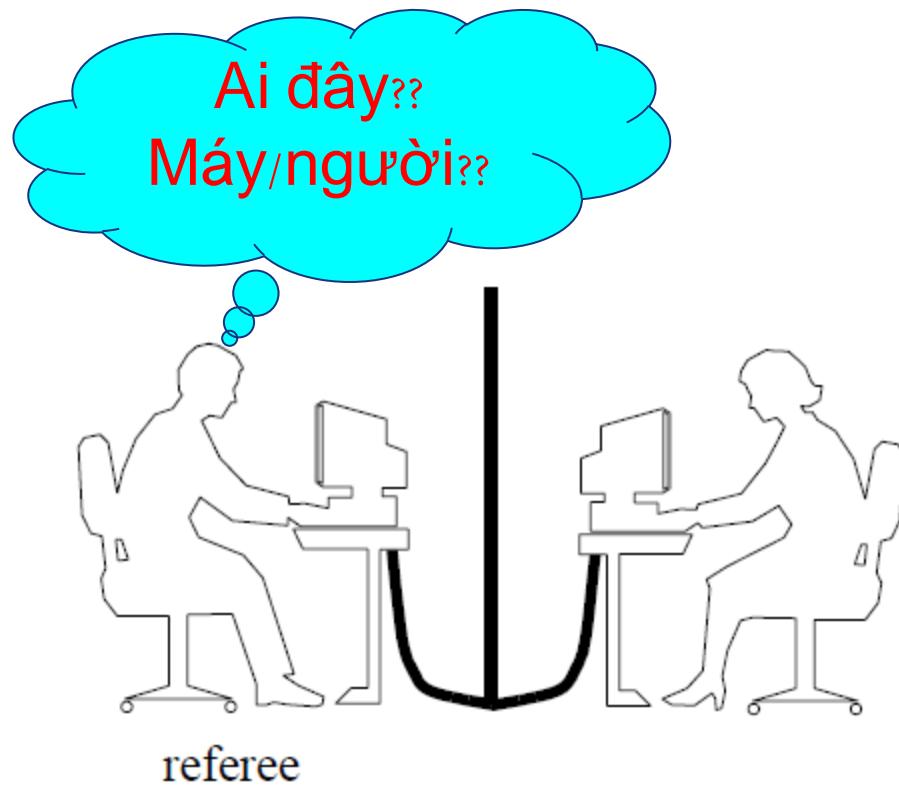
“Computational Intelligence is the study of the design of intelligent agents.” (Poole *et al.*, 1998)

“AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)





Acting humanly: The Turing Test approach



Turing Test is proposed by Alan Turing (1950)



Acting humanly

- ❖ **Natural language processing**
- ❖ **Knowledge representation**
- ❖ **Automated reasoning**
- ❖ **Machine learning**
- ❖ **Computer vision**
- ❖ **Robotics**



Thinking humanly

- ❖ **The cognitive modeling approach**
 - General Problem Solver (regression planning system)
 - (Newell and Simon, 1961)
 - Cognitive science (Khoa học nhận thức)
 - (Wilson and Keil, 1999)



Thinking rationally

- ❖ The “laws of thought” approach
 - Suy luận Logic
- ❖ Two main obstacles (trở ngại)
 - Khó biểu diễn các vấn đề trong thực tế
 - Máy tính không thể tính toán các bài toán lớn



Acting rationally

❖ The rational agent approach

- Situations
- Intelligent Agents
- Probabilistic Reasoning
- Making Complex Decisions



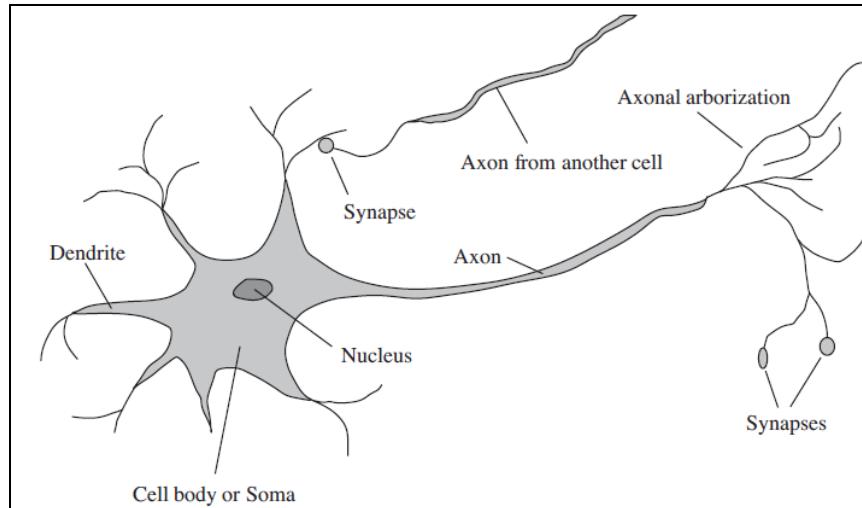
Nền tảng của ngành Trí tuệ nhân tạo

- ❖ **Philosophy (Triết học)**
- ❖ **Mathematics (toán học)**
 - Logic
 - Computation (Algorithm)
 - Probability
- ❖ **Economics (Kinh tế)**
 - How should we make decisions so as to maximize payoff?



Nền tảng của ngành Trí tuệ nhân tạo

❖ Neuroscience(Khoa học thần kinh)



	Supercomputer	Personal Computer	Human Brain
Computational units	10^4 CPUs, 10^{12} transistors	4 CPUs, 10^9 transistors	10^{11} neurons
Storage units	10^{14} bits RAM 10^{15} bits disk	10^{11} bits RAM 10^{13} bits disk	10^{11} neurons 10^{14} synapses
Cycle time	10^{-9} sec	10^{-9} sec	10^{-3} sec
Operations/sec	10^{15}	10^{10}	10^{17}
Memory updates/sec	10^{14}	10^{10}	10^{14}



Nền tảng của ngành Trí tuệ nhân tạo

❖ Psychology (Tâm lý học)

- How do humans and animals think and act?

❖ Computer engineering

- How can we build an efficient computer?



Lịch sử AI : Giai đoạn cổ điển

❖ Giai đoạn cổ điển (1950 – 1965)

▪ Game Playing (Trò chơi) :

- Dựa trên kỹ thuật State Space Search với trạng thái (State) là các tình huống của trò chơi. Đáp án cần tìm là trạng thái thắng hay con đường dẫn tới trạng thái thắng. áp dụng với các trò chơi loại đối kháng.
- Ví dụ: Trò chơi đánh cờ vua.
- Có 2 kỹ thuật tìm kiếm cơ bản:
 - Kỹ thuật **generate and test** : chỉ tìm được 1 đáp án/ chưa chắc tối ưu.
 - Kỹ thuật **Exhaustive search** (vét cạn): Tìm tất cả các nghiệm, chọn lựa phương án tốt nhất.



Lịch sử AI : Giai đoạn cổ điển

❖ Giai đoạn cổ điển (1950 – 1965)

- **Theorem Proving (Chứng minh định lý) :**

- Dựa trên tập tiên đề cho trước, chương trình sẽ thực hiện chuỗi các suy diễn để đạt tới biểu thức cần chứng minh.
- Ví dụ: Chứng minh các định lý tự động, giải toán,...
- Vẫn dựa trên kỹ thuật state space search nhưng khó khăn hơn do mức độ và quan hệ của các phép suy luận: song song, đồng thời, bắc cầu,..



Lịch sử AI- **Giai đoạn viễn vông**

❖ Giai đoạn viễn vông (1965 – 1975)

- Giai đoạn: tham vọng làm cho máy hiểu được con người qua ngôn ngữ tự nhiên.
- Biểu diễn tri thức và phương thức giao tiếp giữa người & máy bằng ngôn ngữ tự nhiên.
 - Semantic Network (mạng ngữ nghĩa)
 - Conceptual graph (đồ thị khái niệm)
 - Frame (khung)
 - Script (kịch bản)
- Kết quả không mấy khả quan



Lịch sử phát triển của AI- Giai đoạn hiện đại

❖ Giai đoạn hiện đại (từ 1975)

- Xác định lại mục tiêu mang tính thực tiễn :
 - Tìm ra **lời giải tốt nhất** trong thời gian chấp nhận.
 - Không nhất thiết tìm ra **lời giải tối ưu**
- Ví dụ:
 - neural networks
 - Hidden Markov models
 - machine learning



Các lĩnh vực ứng dụng

- Game Playing: Tìm kiếm / Heuristic
- Automatic reasoning & Theorem proving: Tìm kiếm / Heuristic
- Expert System: là hướng phát triển mạnh mẽ nhất và có giá trị ứng dụng cao nhất.
- Planning & Robotic: Các hệ thống dự báo, tự động hóa
- Machine learning: Trang bị khả năng học tập để giải quyết vấn đề kho tri thức:
 - Supervised : Học có hướng dẫn.
 - UnSupervised:Tự học



Các lĩnh vực ứng dụng (tt)

- Natural Language Understanding & Semantic modelling:
- Modeling Human performance: Nghiên cứu cơ chế tổ chức trí tuệ của con người để áp dụng cho máy.
- Language and Environment for AI: Phát triển công cụ và môi trường để xây dựng các ứng dụng AI.
- Neural network / Parallel Distributed processing: giải quyết vấn đề năng lực tính toán và tốc độ tính toán bằng kỹ thuật song song và mô phỏng mạng thần kinh của con người.



Câu đố tuần này

1. Tên 4 định nghĩa về AI

2. So sánh tốc độ của máy tính và bộ não người

3. Tên 6 ví dụ về Acting humanly

4. Ké tên 3 ứng dụng AI

Họ tên: Số SV:



Bài tập lớn

- ❖ Chủ đề: Viết AI cho Game Bóng Đá.
- ❖ Cách đánh giá:
 - Thuật toán: 30%
 - Điểm số trong game: 70% (Xếp theo thứ tự của lớp)



I T F

Chương 1: Suy luận Logic



Đặt vấn đề

❖ Con người:

- Nhận thức được thế giới nhờ các giác quan
- Sử dụng các **tri thức** tích lũy để đưa ra hành động hợp lý thông qua **lập luận, suy diễn**.

❖ Mục tiêu AI:

- Làm những công việc như con người.



Ngôn ngữ biểu diễn tri thức

Ngôn ngữ biểu diễn tri thức
=Cú pháp
+Ngữ nghĩa
+Luật suy diễn



Logic mệnh đề

- ❖ Cú pháp
- ❖ Ngữ nghĩa
- ❖ Dạng chuẩn tắc
- ❖ Luật suy diễn



Ví dụ

Thực tế

- ❖ “Nếu trời mưa thì bầu trời có mây”
 - ❖ Trời đang mưa
- Vậy → Bầu trời có mây

Mệnh đề logic

- ❖ $P = \text{“Trời mưa”}$
 - ❖ $Q = \text{“Bầu trời có mây”}$
- Ta có hai phát biểu sau đúng:
- ❖ $P \rightarrow Q$
 - ❖ P

Theo luật Modus Ponens:
Q là đúng.

Nghĩa là: “**Bầu trời có mây**”



Ví dụ

Thực tế

- ❖ “Nếu Tuấn có nhiều tiền thì Tuấn đi mua sắm”
 - ❖ “Tuấn KHÔNG đi mua sắm”
- Vậy → Tuấn KHÔNG có nhiều tiền

Mệnh đề logic

- ❖ $P = \text{“Tuấn có nhiều tiền”}$
- ❖ $Q = \text{“Tuấn đi mua sắm”}$

Ta có hai phát biểu sau đúng:

- ❖ $P \rightarrow Q$
- ❖ $\neg Q$

Theo luật Modus Tollens:

$\neg P$ là đúng.

Nghĩa là: “**Tuấn KHÔNG có nhiều tiền**”



Cú pháp - Các ký hiệu

- ❖ Cú pháp logic mệnh đề gồm tập **các ký hiệu** và tập **các luật xây dựng công thức**.
- ❖ Các ký hiệu
 - Hai hằng logic: True & False
 - Các ký hiệu mệnh đề (biến mệnh đề): P, Q, ...
 - Các kết nối logic: \wedge , \vee , \neg , \Rightarrow , \Leftrightarrow
 - Các dấu mở ngoặc(, đóng ngoặc)



Cú pháp - Luật xây dựng công thức

❖ Luật xây dựng công thức

- Các biến mệnh đề là công thức
- Nếu A và B là công thức thì:
 - $(A \wedge B)$ ("A hội B" hoặc "A và B")
 - $(A \vee B)$ ("A tuyển B" hoặc "A hoặc B")
 - $(\neg A)$ ("phủ định A")
 - $(A \Rightarrow B)$ ("A kéo theo B" hoặc "nếu A thì B")
 - $(A \Leftrightarrow B)$ ("A và B kéo theo nhau")

là các công thức.

Ngữ nghĩa

- ❖ Ngữ nghĩa của logic mệnh đề cho phép ta **xác định ý nghĩa của các** công thức trong thế giới hiện thực nào đó.
- ❖ Một sự kết hợp các kí hiệu mệnh đề với các sự kiện trong thế giới thực được gọi là một minh họa (interpretation).
- ❖ Một sự kiện chỉ có thể **đúng** hoặc **sai**.
- ❖ Ví dụ:
 - P= “Paris là thủ đô nước Pháp”



Bảng chân lý các kết nối logic

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE



Bài tập

Điền các giá trị TRUE hoặc FALSE vào chỗ trống

P	Q	S	$P \Rightarrow Q$	$(P \Rightarrow Q) \wedge S$
FALSE	FALSE	FALSE		
FALSE	FALSE	TRUE		
FALSE	TRUE	FALSE		
FALSE	TRUE	TRUE		
TRUE	FALSE	FALSE		
TRUE	FALSE	TRUE		
TRUE	TRUE	FALSE		
TRUE	TRUE	TRUE		



Dạng chuẩn tắc - Tương đương

- ❖ A và B được xem là **tương đương nếu chung có cùng** một giá trị chân lý.
- ❖ Ký hiệu: $A \equiv B$
- ❖ Các tương đương:
 - Trong các tương đương sau A,B,C là các mệnh đề bất kỳ.

Luật thông dụng

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

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

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



Dạng chuẩn tắc - Tương đương

Luật De Morgan

$$\begin{aligned}\neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B\end{aligned}$$

Luật giao hoán

$$\begin{aligned}A \vee B &\equiv B \vee A \\ A \wedge B &\equiv B \wedge A\end{aligned}$$

Luật kết hợp

$$\begin{aligned}(A \vee B) \vee C &\equiv A \vee (B \vee C) \\ (A \wedge B) \wedge C &\equiv A \wedge (B \wedge C)\end{aligned}$$

Luật phân phối

$$\begin{aligned}A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C)\end{aligned}$$



Dạng chuẩn tắc

- **Dạng chuẩn:** là kết xuất chuẩn của các giải thuật làm việc với phép toán mệnh đề.
- **Tuyễn cơ bản:** là thành phần cơ bản hay sự kết hợp của các thành phần cơ bản bằng phép tuyễn (\vee)
- **Hội cơ bản:** là thành phần cơ bản hay sự kết hợp của các thành phần cơ bản bằng phép hội (\wedge).
- **Dạng chuẩn hội – CNF:** là thành phần tuyễn cơ bản hay các tuyễn cơ bản kết hợp bởi phép hội.
- **Dạng chuẩn tuyễn – DNF:** là thành phần hội cơ bản hay các hội cơ bản kết hợp bởi phép tuyễn.

Dạng chuẩn hội

- ❖ Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta chuẩn hóa các công thức, đưa chúng về **dạng chuẩn hội**.
- ❖ Các phương pháp chuyển đổi công thức về dạng chuẩn hội:
 - Bỏ các dấu kéo theo (\Rightarrow) bằng cách thay ($A \Rightarrow B$) bởi ($\neg A \vee B$).
 - Chuyển các dấu phủ định (\neg) vào sát các ký hiệu mệnh đề (luật De Morgan)
 - Áp dụng luật phân phối, thay các công thức có dạng $A \vee(B \wedge C)$ bởi $(A \vee B) \wedge (A \vee C)$.



Dạng chuẩn hội – Ví dụ

Chuẩn hóa công thức sau:

$$(P \Rightarrow Q) \vee \neg(R \vee \neg S)$$

Kết quả:

$$\begin{aligned} & (P \Rightarrow Q) \vee \neg(R \vee \neg S) \\ & \equiv (\neg P \vee Q) \vee (\neg R \wedge S) \\ & \equiv ((\neg P \vee Q) \vee \neg R) \wedge ((\neg P \vee Q) \vee S) \\ & \equiv (\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee S). \end{aligned}$$



Luật suy diễn

❖ Luật suy diễn được áp dụng để phát triển các ứng dụng có khả năng suy luận. Suy luận là hoạt động thường xuyên của con người để hiểu các lý lẽ, kiểm chứng, phán đoán các vấn đề.

❖ Luật Modus Ponens (MP)

$$A, A \Rightarrow B \quad \therefore \quad B$$

❖ Luật Modus Tollens (MT)

$$A \Rightarrow B, \neg B \quad \therefore \quad \neg A$$

❖ Luật Hội

$$A, B \quad \therefore \quad A \wedge B$$

❖ Luật đơn giản

$$A \wedge B \quad \therefore \quad A$$

❖ Luật Cộng

$$A \quad \therefore \quad A \vee B$$

❖ Luật tam đoạn luận tuyễn

$$A \vee B, \neg A \quad \therefore \quad B$$

❖ Luật tam đoạn luận giả thiết

$$A \Rightarrow B, B \Rightarrow C \therefore A \Rightarrow C$$

❖ Luật phân giải

$$A \vee B, \neg B \vee C \therefore A \vee C$$



Phương pháp chứng minh bác bỏ

- ❖ Là phương pháp thường xuyên được sử dụng trong các chứng minh toán học.
- ❖ Để chứng minh P đúng, ta giả sử P sai ($\neg P$ vào các giả thiết) và dẫn tới một mâu thuẫn. (Dùng luật phân giải để sinh ra công thức rỗng $[]$)



Chứng minh bắc bỏ bằng luật phân giải

Luật phân giải trên các câu tuyễn

$$\frac{A_1 \vee \dots \vee A_m \vee C}{\neg C \vee B_1 \vee \dots \vee B_n}$$
$$A_1 \vee \dots \vee A_m \vee B_1 \vee \dots \vee B_n$$

Luật phân giải trên các câu Horn

$$\frac{P_1 \wedge \dots \wedge P_m \wedge S \Rightarrow Q}{R_1 \wedge \dots \wedge R_n \Rightarrow S}$$
$$P_1 \wedge \dots \wedge P_m \wedge R_1 \wedge \dots \wedge R_n \Rightarrow Q$$



Ví dụ 1:

Cho tập các câu tuyên sau:

$$\neg A \vee \neg B \vee P \quad (1)$$

$$\neg C \vee \neg D \vee P \quad (2)$$

$$\neg E \vee C \quad (3)$$

$$A \quad (4)$$

$$E \quad (5)$$

$$D \quad (6)$$

Chứng minh P.

Chứng minh:

Giả sử P sai:

$$\neg P \quad (7)$$

Từ (2) và (7), theo luật phân giải:

$$\neg C \vee \neg D \quad (8)$$

Từ (6) và (8), theo luật phân giải:

$$\neg C \quad (9)$$

Từ (3) và (9), theo luật phân giải:

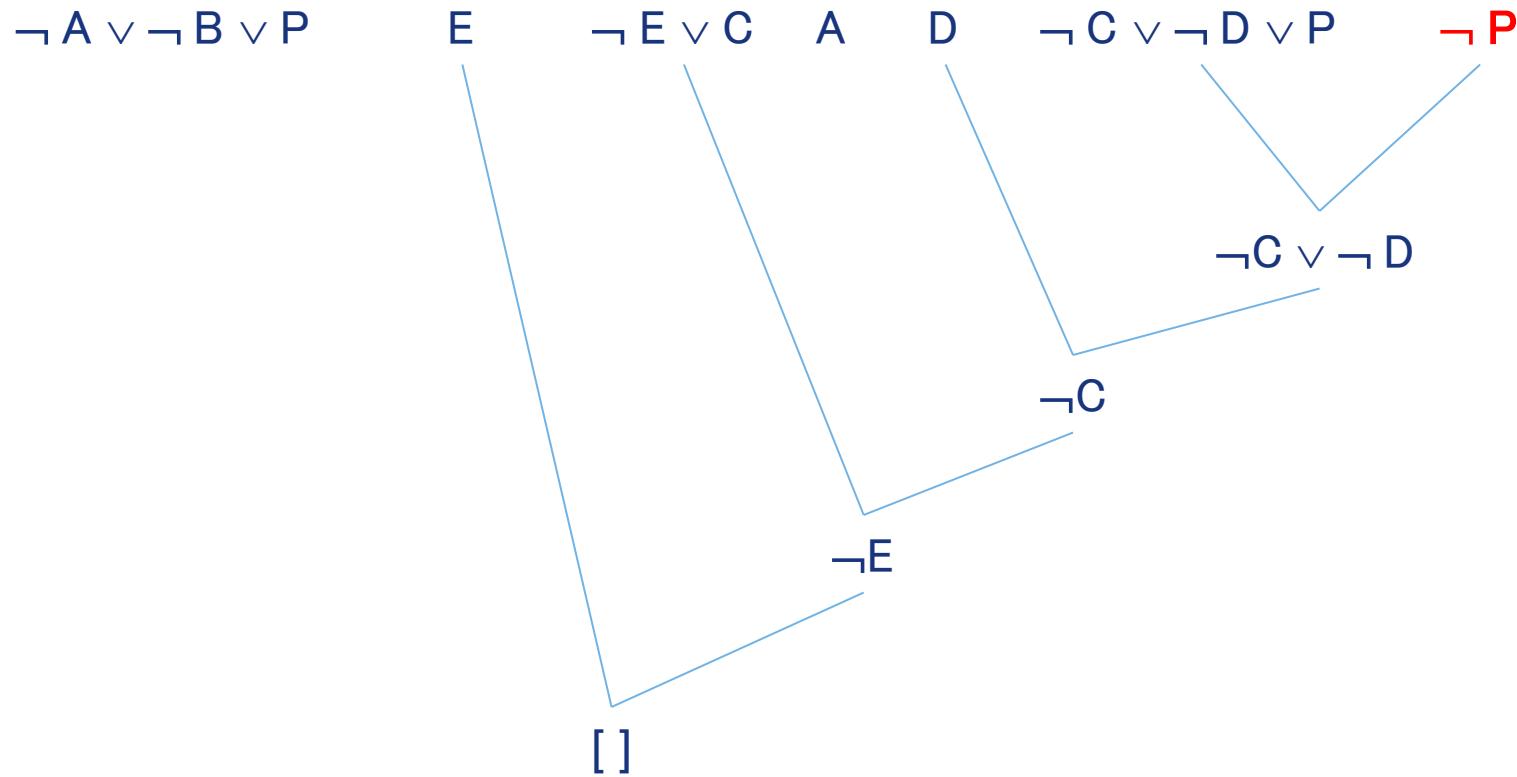
$$\neg E \quad (10)$$

Từ (5) và (10), theo luật phân giải:

[] (Mâu thuẫn) \rightarrow P đúng



Chứng minh bằng đồ thị





Ví dụ 2:

❖ Cho đoạn sau:

“ Nam đẹp trai, giàu có. Do vậy, Nam hoặc là phung phí hoặc là (nhân từ và giúp người). Thực tế, Nam không phung phí và cũng không kêu cǎng.”

Suy luận : “Do vậy, có thể nói Nam là người nhân từ”

❖ Kiểm chứng kết quả suy luận trên, bằng luật phân giải.

Ví dụ 2(tt)

❖ (i) Chuyển sang mệnh đề:

- P1 = “Nam đẹp trai.”
- P2 = “Nam giàu có.”
- P3 = “Nam phung phí.”
- P4 = “Nam kiêu căng.”
- P5 = “Nam nhân từ.”
- P6 = “Nam giúp người.”

Các biểu thức thành lập được từ đoạn trên:

- Wff1 = P1 \wedge P2
- Wff2 = $(P1 \wedge P2) \Rightarrow (P3 \wedge \neg(P5 \wedge P6)) \vee (\neg P3 \wedge (P5 \wedge P6))$
- Wff3 = $\neg P3 \wedge \neg P4$

Wff4 = P5 Biểu thức cần chứng minh.



Cách Chứng Minh 1- Nguồn từ BK HCM

❖ (ii) Đưa về dạng clause:

- ❖ Wff₁, sinh ra hai clause: $C_1 = P_1$ $C_2 = P_2$
- ❖ Wff₂ = $\neg(P_1 \wedge P_2) \vee ((P_3 \wedge \neg(P_5 \wedge P_6)) \vee (\neg P_3 \wedge (P_5 \wedge P_6)))$
.....
 $= (\neg P_1 \vee \neg P_2 \vee P_3 \vee \neg P_3 \vee \neg P_6) \wedge (\neg P_1 \vee \neg P_2 \vee P_5 \vee \neg P_3 \vee \neg P_6) \wedge (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_3 \vee P_5) \wedge (\neg P_1 \vee \neg P_2 \vee P_5 \vee P_3 \vee P_5) \wedge (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5 \vee \neg P_6) \wedge (\neg P_1 \vee \neg P_2 \vee P_5 \vee P_5 \vee \neg P_6) \wedge (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_3 \vee P_6) \wedge (\neg P_1 \vee \neg P_2 \vee P_5 \vee P_3 \vee P_6)$

Sinh ra các clause:

$$C_3 = (\neg P_1 \vee \neg P_2 \vee \neg P_6)$$

$$C_5 = (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5)$$

$$C_7 = (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5 \vee \neg P_6)$$

$$C_9 = (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_6)$$

$$C_4 = (\neg P_1 \vee \neg P_2 \vee P_5 \vee \neg P_3 \vee \neg P_6)$$

$$C_6 = (\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5)$$

$$C_8 = (\neg P_1 \vee \neg P_2 \vee P_5 \vee \neg P_6)$$

$$C_{10} = (\neg P_1 \vee \neg P_2 \vee P_5 \vee P_3 \vee P_6)$$

- ❖ Wff₃ sinh ra các clause: $C_{11} = \neg P_3$ $C_{12} = \neg P_4$ $C_{13} = \neg P_5$

(gồm cả bước lấy phủ định kết luận)



Cách Chứng Minh 1(tt)

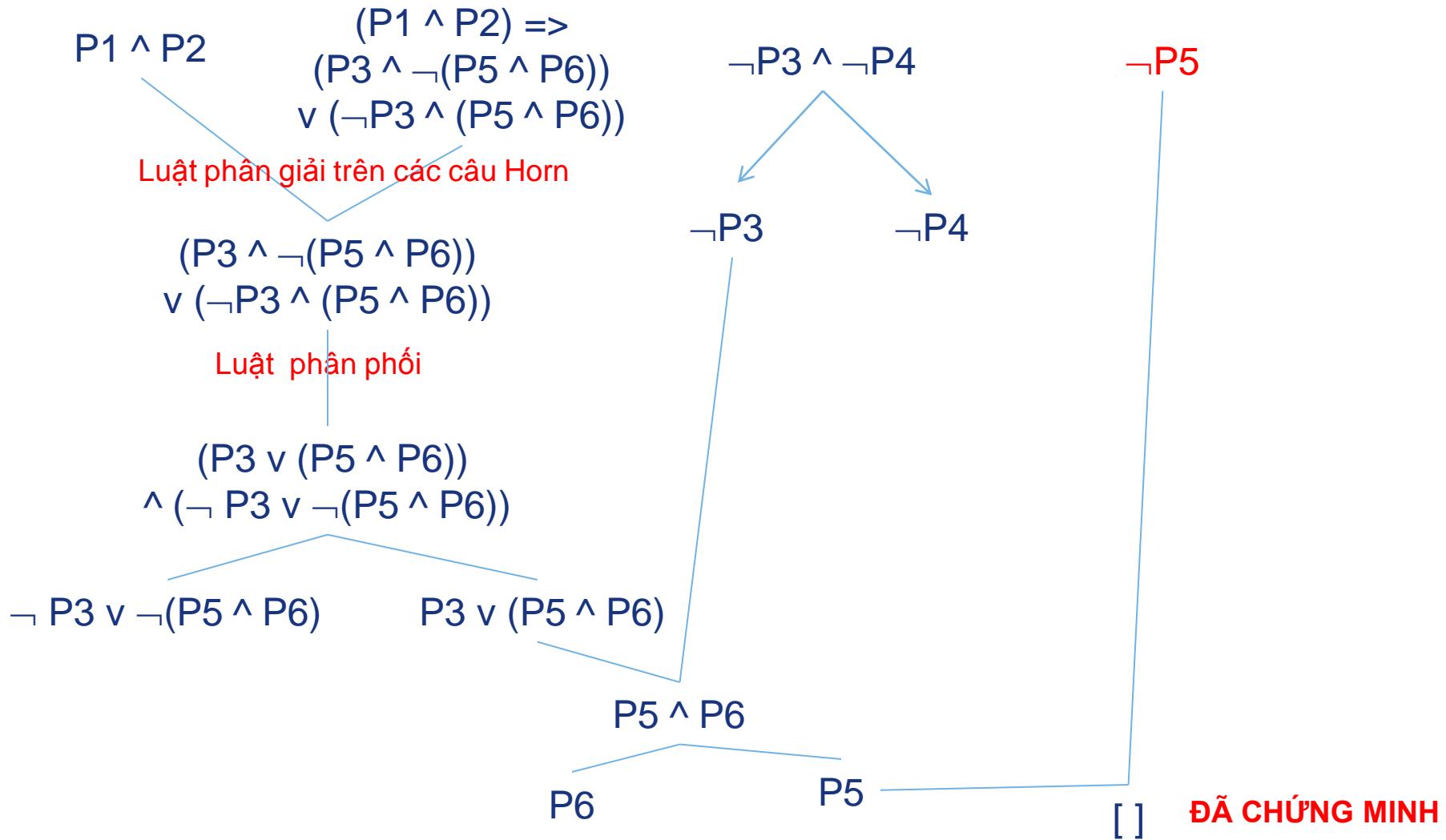
- iii) áp dụng luật phân giải trên các clause:

TT	Clauses	Luật áp dụng	TT	Clauses	Luật áp dụng
1	P_1	P	12.	$\neg P_4$	P
2	P_2	P	13	$\neg P_5$	P
3	$\neg P_1 \vee \neg P_2 \vee \neg P_6$	P		
4	$\neg P_1 \vee \neg P_2 \vee P_5 \vee \neg P_3 \vee \neg P_6$	P	14	$\neg P_2 \vee \neg P_6$	1,2, R
5	$\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5$	P	15	$\neg P_6$	2, 14, R
6	$\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5$	P	16	$\neg P_1 \vee \neg P_2 \vee P_5 \vee P_3$	10,15,R
7	$\neg P_1 \vee \neg P_2 \vee P_3 \vee P_5 \vee \neg P_6$	P	17	$\neg P_2 \vee P_5 \vee P_3$	1,16,R
8	$\neg P_1 \vee \neg P_2 \vee P_5 \vee \neg P_6$	P	18	$P_5 \vee P_3$	2,17, R
9	$\neg P_1 \vee \neg P_2 \vee P_3 \vee P_6$	P	19	P_3	13, 18, R
10	$\neg P_1 \vee \neg P_2 \vee P_5 \vee P_3 \vee P_6$	P	20	□	11, 19, R
11.	$\neg P_3$	P			

ĐÃ CHỨNG MINH



Cách Chứng minh 2





Logic vị từ

- ❖ Đặt vấn đề
- ❖ Định nghĩa
- ❖ Cú pháp
- ❖ Ngữ nghĩa
- ❖ Dạng chuẩn tắc
- ❖ Luật suy diễn

Đặt vấn đề

- ❖ Phép toán mệnh đề → suy diễn tự động nhưng **chưa đủ** khi cần phải truy cập vào thành phần nhỏ trong câu, dùng biến số trong câu.

Ví dụ:

**“Mọi sinh viên trường ĐHBK đều có bằng tú tài. Lan không có bằng tú tài.
Do vậy, Lan không là sinh viên trường ĐHBK”**

“Lan” là một đối tượng cụ thể của “SV trường ĐHBK” – không thể đặc tả được “quan hệ” này trong mệnh đề được mà chỉ có thể là:

“LAN là sinh viên trường ĐHBK thì Lan có bằng tú tài. Lan không có bằng tú tài. Do vậy, Lan không là sinh viên trường ĐHBK”

- ❖ Mệnh đề phải giải quyết bằng cách liệt kê tất cả các trường hợp
→ Không khả thi
- ❖ Do đó, chúng ta cần một Logic khác hơn là phép toán mệnh đề:

PHÉP TOÁN VỊ TỪ

Định nghĩa

- ❖ **Vị từ là một phát biểu nói lên quan hệ giữa một đối tượng với các thuộc tính của nó hay quan hệ giữa các đối tượng với nhau.**

Vị từ được **biểu diễn** bởi một tên được gọi là **tên vị từ**, sau nó là một danh sách các thông số.

Ví dụ:

- + Phát biểu: “**Lan là sinh viên trường ĐHBK**”
- + Biểu diễn: **sv_bk(“Lan”)**

Ý nghĩa: đối tượng tên là “Lan” có thuộc tính là “sinh viên trường ĐHBK”.



Cú pháp - Các ký hiệu

- ❖ Các ký hiệu hằng: a, b, c, An, Ba, John,...
- ❖ Các ký hiệu biến: x, y, z, u, v, w,...
- ❖ Các ký hiệu vị từ: P, Q, R, S, Like, Havecolor, ...
 - Mỗi ký hiệu vị từ là vị từ của n biến ($n \geq 0$). Chẳng hạn
 - Like là vị từ của hai biến,
 - Các ký hiệu vị từ không biến là các ký hiệu mệnh đề.



Cú pháp - Các ký hiệu(tt)

- ❖ Các ký hiệu hàm: f, g, cos, sin, mother, husband, distance,...
- ❖ Các ký hiệu kết nối logic: \wedge (hội), \vee (tuyễn), \neg (phủ định), \Rightarrow (kéo theo), \Leftrightarrow (kéo theo nhau).
- ❖ Các ký hiệu lượng tử: \forall (với mọi), \exists (tồn tại).
- ❖ Các ký hiệu ngăn cách: dấu phẩy, dấu mở ngoặc và dấu đóng ngoặc.



Cú pháp - Công thức phân tử

Công thức phân tử :

tên_vị_tù(term₁, term₂, ..., term_n)

- ❖ **Tên vị tù:** [a..z](a..z| A..Z| 0..9|_)*
Bắt đầu bởi một ký tự chữ thường.
Ví dụ: ban_than, banThan,bAN_THAN,...
- ❖ **Term có thể là:** Hằng,Biến, Biểu thức hàm.



Cú pháp - Các công thức

- ❖ Các công thức được xác định đệ quy như sau:
 - Các công thức phân tử là công thức.
 - Nếu G và H là các công thức, thì
 - các biểu thức $(G \wedge H)$, $(G \vee H)$, $(\neg G)$, $(G \Rightarrow H)$, $(G \Leftrightarrow H)$ là công thức.
 - Nếu G là một công thức và x là biến thì
 - các biểu thức $(\forall x G)$, $(\exists x G)$ là công thức.

- ❖ Các công thức không chứa biến sẽ được gọi là **công thức cụ thể**.



Vị từ: Biểu diễn thế giới thực

❖ Chuyển các câu sau sang biểu thức vị từ:

“Mọi sinh viên trường DHBK đều có bằng tú tài.

Lan không có bằng tú tài.

Do vậy, Lan không là sinh viên trường DHBK”

❖ Với $sv_bk(X)$ cho biết: “ X là sinh viên trường DHBK”

$tu_tai(X)$ cho biết: “ X có bằng tú tài”

❖ Các câu trên được chuyển qua vị từ là:

$\forall X(sv_bk(X) \Rightarrow tu_tai(X))$.

$\neg tu_tai(\text{“Lan”})$.

Do vậy, $\neg sv_bk(\text{“Lan”})$.



Vị từ: Biểu diễn thế giới thực (tt)

❖ “Chỉ vài sinh viên CNTT lập trình tốt.”

với sv_mt(X) : “X là sinh viên CNTT”
 laptrinh_tot(X) : “X lập trình tốt”

Câu trên chuyển sang vị từ là: $\exists X (sv_mt(X) \wedge laptrinh_tot(X))$

❖ “Không một sinh viên CNTT nào không cần cù.”

với: sv_mt(X) : “X là sinh viên CNTT”
 can_cu(X) : “X cần cù”

Câu trên chuyển sang là: $\forall X (sv_mt(X) \Rightarrow can_cu(X))$

❖ “Không phải tất cả các sinh viên CNTT đều thông minh”

với thong_minh(X) : “X thông minh”

Câu trên chuyển sang là: $\exists X (sv_mt(X) \wedge \neg thong_minh(X))$



Vị từ: Ngữ nghĩa

❖ Vấn đề: Nếu chúng ta có biểu thức sau:

$$\forall X \exists Y p(X, Y)$$

Chúng ta hiểu như thế nào ????!

-> Cần sự diễn dịch.

+ Cách hiểu 1:

X, Y : là con người.

$p(X, Y)$ cho biết : “X là cha của Y”

Do vậy:

$\forall X \exists Y p(X, Y)$ có thể hiểu là:

“Mọi người X, tồn tại người Y để X là cha của Y”

-> wff = $\forall X \exists Y p(X, Y)$ có trị là F (sai)



Vị từ: Ngữ nghĩa (tt)

+ Cách hiểu 2:

X, Y : là con người.

p(X,Y) cho biết : “Y là cha của X”

Do vậy:

$\forall X \exists Y p(X,Y)$ có thể hiểu là:

“Mọi người X, tồn tại người Y là cha của X”

-> wff = $\forall X \exists Y p(X,Y)$ có trị là T (đúng)

+ Cách hiểu 3:

X, Y : là số nguyên.

p(X,Y) cho biết : “Y bằng bình phương của X”

-> wff = $\forall X \exists Y p(X,Y)$ có trị là T (đúng)

Các công thức tương đương

- ❖ Giả sử G là một công thức, cách viết $G(x)$ thể hiện G có chứa các xuất hiện của biến x .
- ❖ Ta có các tương đương sau
 - Đặt tên lại biến
 - $\forall x \ G(x) \equiv \forall y \ G(y)$
 - $\exists x \ G(x) \equiv \exists y \ G(y)$
 - Phủ định
 - $\neg (\forall x \ G(x)) \equiv \exists x (\neg G(x))$
 - $\neg (\exists x \ G(x)) \equiv \forall x (\neg G(x))$
 - Phân giải
 - $\forall x (G(x) \wedge H(x)) \equiv \forall x G(x) \wedge \forall x H(x)$
 - $\exists x (G(x) \vee H(x)) \equiv \exists x G(x) \vee \exists x H(x)$



Chuẩn hóa công thức

❖ Thủ tục chuẩn hóa các công thức gồm các bước sau:

- Loại bỏ các ký hiệu kéo theo

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

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

- Chuyển các dấu phủ định (\neg) vào sát phần tử

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

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

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

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



Chuẩn hóa công thức

- ❖ Thay đổi đại lượng tồn tại
 - ❖ $\exists y (G(y)) \equiv (G(Y))$
 - ❖ $\forall x (\exists y (P(x,y))) \equiv \forall x (P(x,f(x)))$
 - ❖ Thay thế đại lượng tồn tại bằng một công thức hoặc hằng
- ❖ Xóa đại lượng với mọi
 - ❖ $\forall x (P(x,f(x))) \equiv P(x,f(x))$
- ❖ Chuyển về dạng chuẩn hội



Luật suy diễn

Luật phân giải trên các câu tuyễn

$$\frac{A_1 \vee \dots \vee A_m \vee C}{\neg C \vee B_1 \vee \dots \vee B_n}$$
$$A_1 \vee \dots \vee A_m \vee B_1 \vee \dots \vee B_n$$

Luật phân giải trên các câu Horn

$$\frac{P_1 \wedge \dots \wedge P_m \wedge S \Rightarrow Q}{R_1 \wedge \dots \wedge R_n \Rightarrow S}$$
$$P_1 \wedge \dots \wedge P_m \wedge R_1 \wedge \dots \wedge R_n \Rightarrow Q$$



Bài tập ví dụ

❖ Ta biết các thông tin sau:

- Ba nuôi một con chó
- Ba hoặc Am đã giết con mèo Bibi
- Mọi người nuôi chó đều yêu quý động vật
- Ai yêu quý động vật cũng không giết động vật
- Chó mèo là động vật

❖ Hỏi

- Ai giết mèo Bibi



Lời giải

❖ Định nghĩa

- Hằng: D, Ba, Am, Bibi,
- Các vị từ:
 - Dog(x): x là chó
 - Cat(x): x là mèo
 - Rear(x,y): x nuôi y
 - AnimalLover(x): x yêu động vật
 - Kill(x,y): x giết y
 - Animal(x): y là động vật



Chuyển các câu thành logic vị từ

- ❖ Ba nuôi một con chó
- ❖ Ba hoặc Am đã giết con mèo Bibi
- ❖ Mọi người nuôi chó đều yêu quý động vật
- ❖ Ai yêu quý động vật cũng không giết động vật
- ❖ Chó mèo là động vật
- ❖ Hỏi: Ai giết mèo Bibi

$\text{Dog}(D) \wedge \text{Rear}(\text{Ba}, D)$

$\text{Cat}(\text{Bibi}) \wedge (\text{Kill}(\text{Ba}, \text{Bibi}) \vee \text{Kill}(\text{Am}, \text{Bibi}))$

$\forall x (\forall y (\text{Dog}(y) \wedge \text{Rear}(x, y)) \Rightarrow \text{AnimalLover}(x))$

$\forall x \forall y (\text{AnimalLover}(x) \wedge (\text{Animal}(y)) \Rightarrow \neg \text{Kill}(x, y))$

$\forall x (\text{Dog}(x) \Rightarrow \text{Animal}(x)) \wedge \forall y (\text{Cat}(y) \Rightarrow \text{Animal}(y))$

Hỏi: $\text{Kill}(\text{?????}, \text{Bibi})$

Chuẩn hóa công thức

❖ Dog(D) \wedge Rear(Ba,D)

- Dog(D),
- Rear(Ba,D)

❖ Cat(Bibi) \wedge (Kill(Ba,Bibi) \vee Kill(Am,Bibi))

- Cat(Bibi),
- Kill(Ba,Bibi) \vee Kill(Am,Bibi)

❖ $\forall x (\forall y (Dog(y) \wedge Rear(x,y)) \Rightarrow AnimalLover(x))$

- $\neg Dog(y) \vee \neg Rear(x,y) \vee AnimalLover(x)$

❖ $\forall x \forall y (AnimalLover(x) \wedge (Animal(y)) \Rightarrow \neg Kill(x,y))$

- $\neg AnimalLover(x) \vee \neg (Animal(y)) \vee \neg Kill(x,y)$

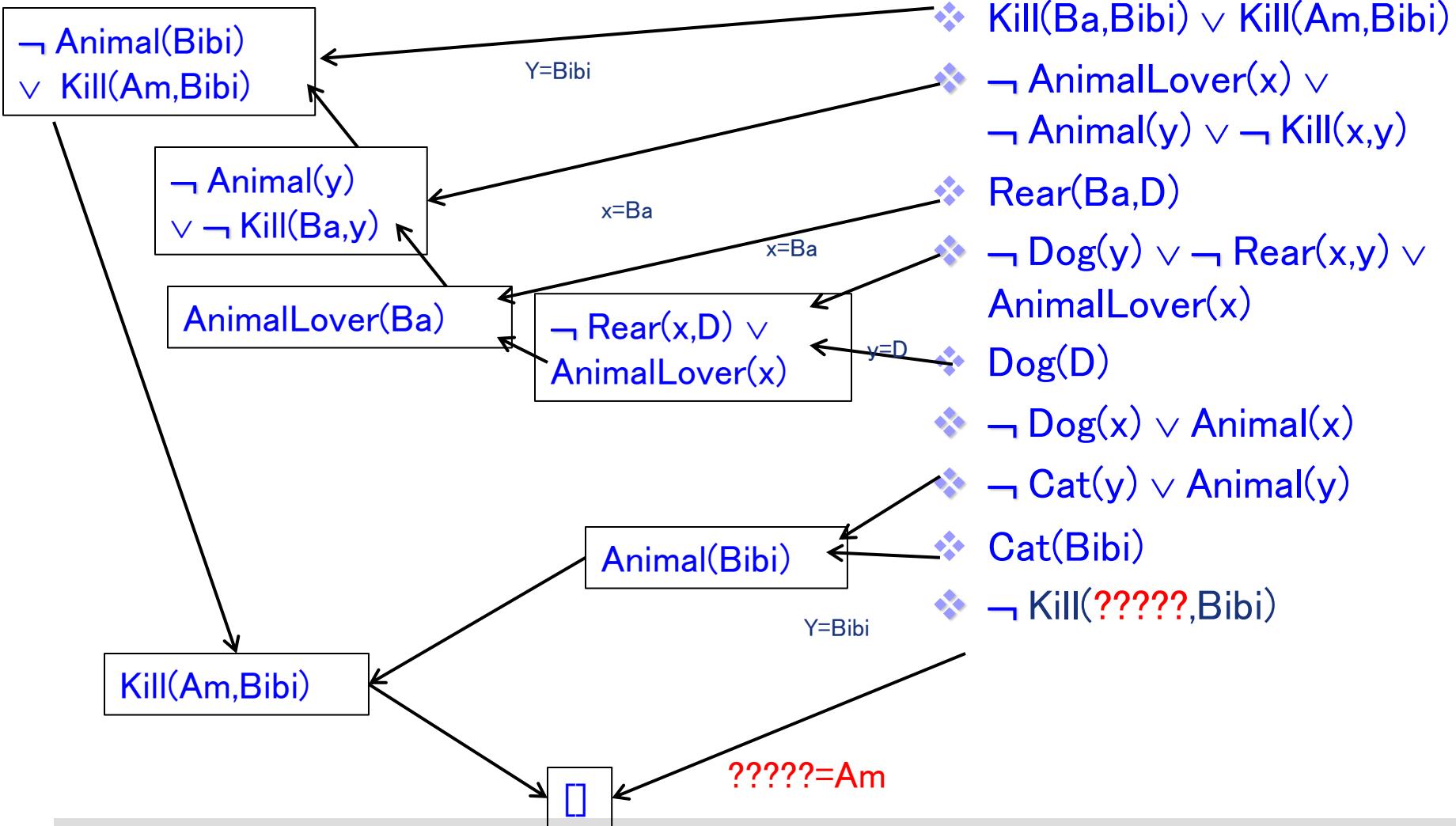
❖ $\forall x (Dog(x) \Rightarrow Animal(x)) \wedge \forall y (Cat(y) \Rightarrow Animal(y))$

- $\neg Dog(x) \vee Animal(x)$
- $\neg Cat(y) \vee Animal(y)$

❖ Hỏi: Kill(?????,Bibi)



Đáp án



Biểu diễn bài toán và tìm kiếm

Problem formulation and search

Phạm Minh Tuấn, PhD

Nội dung

- Tổng quan về tìm kiếm
- Biểu diễn bài toán trong không gian trạng thái
- Đồ thị tìm kiếm
- Cây tìm kiếm
- Tìm kiếm không có thông tin
 - Tìm kiếm rộng
 - Tìm kiếm sâu

Tổng quan

- ✓ **Tìm kiếm** là thuật toán lấy đầu vào là một bài toán và trả về kết quả là một lời giải cho bài toán đó, thường là sau khi cân nhắc giữa một loạt các lời giải có thể.
- ✓ **Không gian trạng thái** trong tìm kiếm
 - ✓ Là tập các trạng thái có thể có của bài toán và các toán tử di chuyển trạng thái
 - ✓ Được biểu diễn bằng cây, đồ thị,...
- ✓ Các phương pháp tìm kiếm
 - ✓ Tìm kiếm không có thông tin
 - ✓ Tìm kiếm có thông tin
 - ✓ Tìm kiếm đối kháng

Biểu diễn trong không gian trạng thái

Trò chơi 8 số

- ✓ Trạng thái đầu: {1, null, 4, 6, 3, 2, 7, 8, 5}
- ✓ Trạng thái cuối: {1, 2, 3, 4, 5, 6, 7, 8, null}
- ✓ Toán tử:
 - ✓ **Up:** Ô ở dưới ô trống di chuyển lên phía trên
 - ✓ **Down:** Ô ở trên ô trống di chuyển xuống phía dưới
 - ✓ **Left:** Ô ở bên phải ô trống di chuyển sang trái
 - ✓ **Right:** Ô ở bên trái ô trống di chuyển sang phải
- ✓ Trạng thái sẽ thay đổi phụ thuộc và toán tử
- ✓ Tùy thuộc vào trạng thái mà sẽ có toán tử không thực hiện được

Trạng thái đầu

1		4
6	3	2
7	8	5

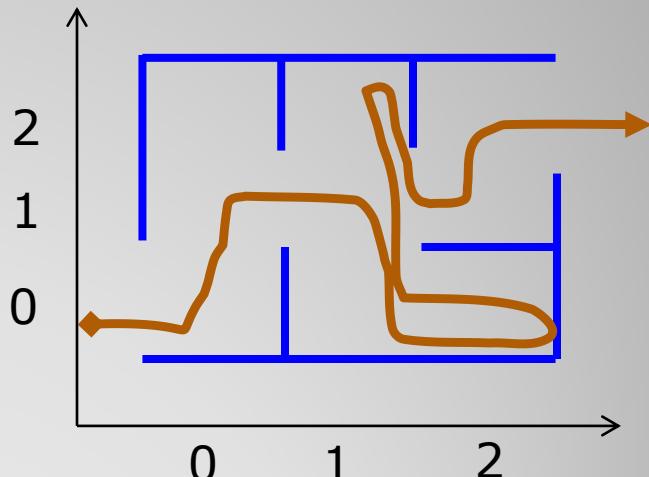
Trạng thái cuối

1	2	3
4	5	6
7	8	

Biểu diễn trong không gian trạng thái

Tìm đường trong Mê cung

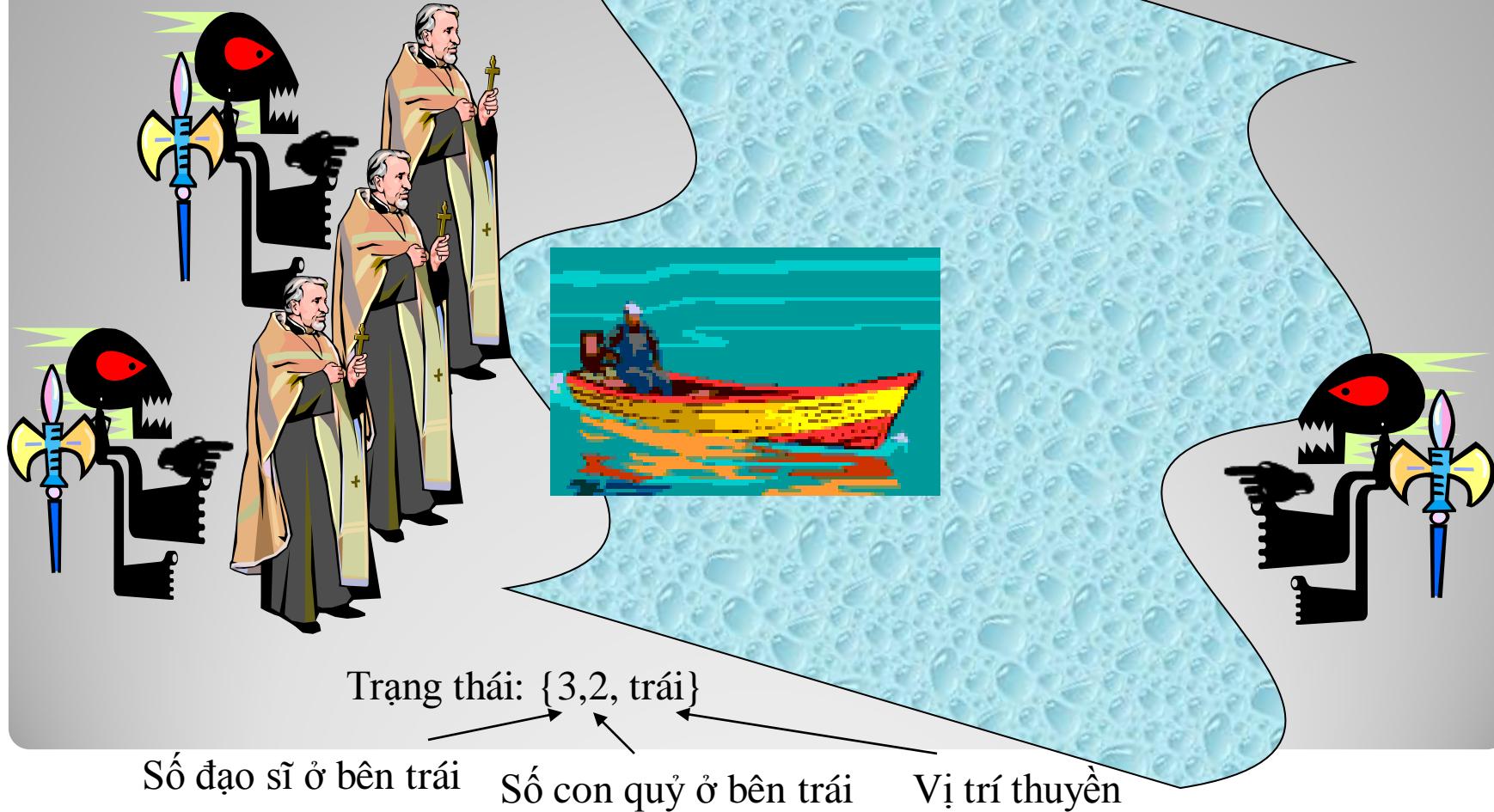
- ✓ Trạng thái đầu: {0,0}
- ✓ Trạng thái cuối: {2,2}
- ✓ Toán tử:
 - ✓ Up: Di chuyển lên trên
 - ✓ Down: Di chuyển xuống dưới
 - ✓ Left: Di chuyển sang trái
 - ✓ Right: Di chuyển sang phải
- ✓ Trạng thái sẽ thay đổi phụ thuộc và toán tử
- ✓ Tùy thuộc vào trạng thái mà sẽ có toán tử không thực hiện được



Biểu diễn trong không gian trạng thái

Trạng thái đầu: {3,3, trái}

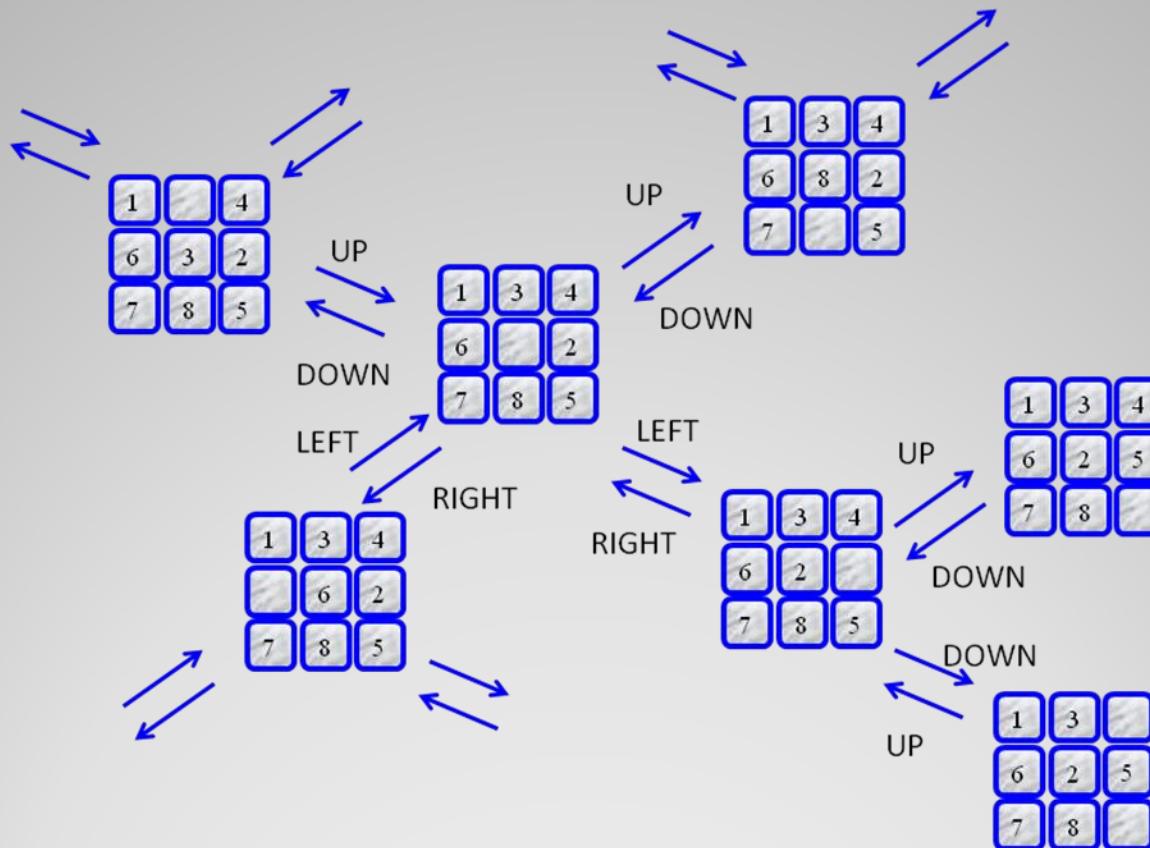
Trạng thái cuối: {0,0, phải}



Các bài toán khác

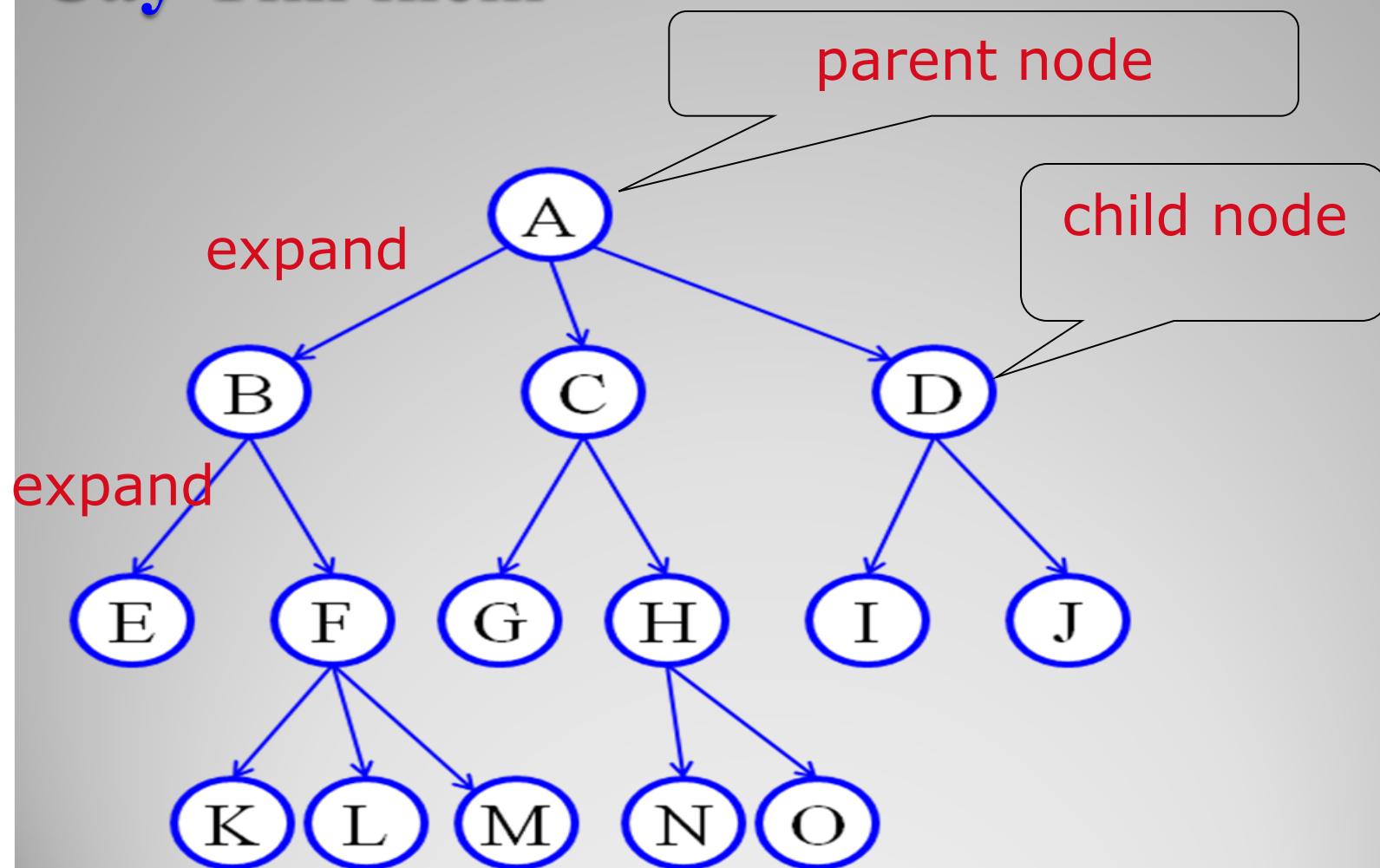
- ✓ Bài toán tháp hanoi
- ✓ Bài toán 8 con hậu
- ✓ Cờ vua, cờ tướng
- ✓ Tìm đường
- ✓ Tìm kiếm thông tin trên internet
- ✓ Game AI (PES, Mario,...)

Đồ thị tìm kiếm



Đồ thị có hướng của trò chơi 8 số

Cây Tìm kiếm

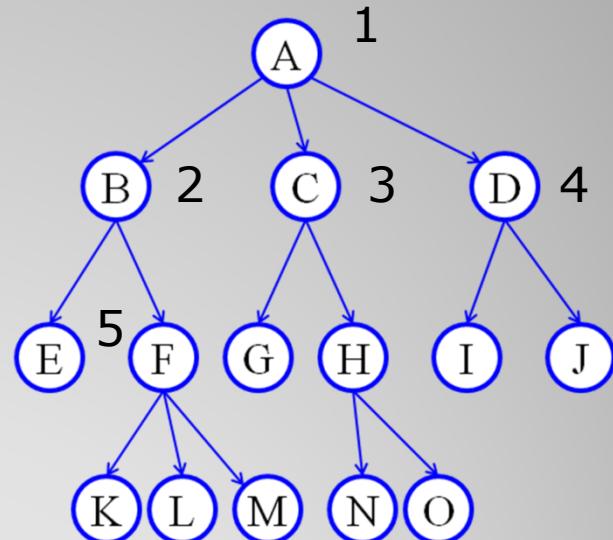


Tìm kiếm không có thông tin

- ✓ Tìm kiếm rộng
- ✓ Tìm kiếm sâu
- ✓ Tìm kiếm sâu dần

Tìm kiếm rộng

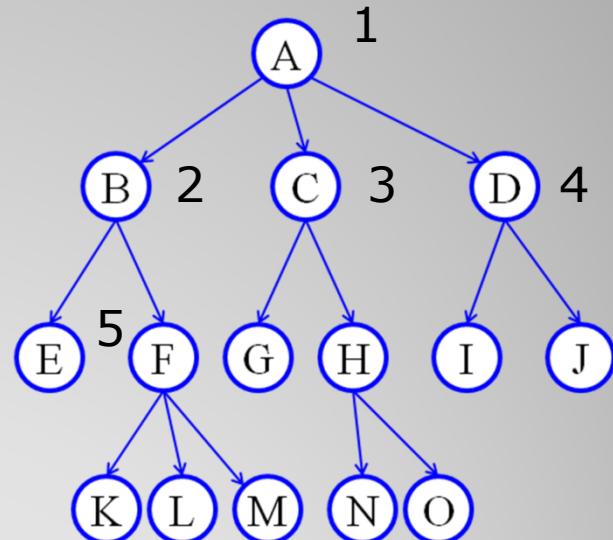
- ✓ Tìm kiếm rộng là phương pháp tìm kiếm mà ưu tiên những đỉnh gần với đỉnh xuất phát.
- ✓ Thứ tự tìm kiếm: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I \rightarrow J \rightarrow K \rightarrow L \rightarrow M \rightarrow N \rightarrow O$.



Tìm kiếm rộng

Thuật toán tìm kiếm rộng:

1. Cho đỉnh xuất phát vào **open**.
2. Nếu **open** rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh **đầu** trong **open** ra và gọi đó là Θ .
4. Nếu Θ là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của Θ và cho vào **cuối** của **open**
6. Trở lại bước 2.



*Open hoạt động theo quy tắc FIFO

Tìm kiếm rỗng – Ví dụ 1

Quá trình biến đổi của open:

open= {A}

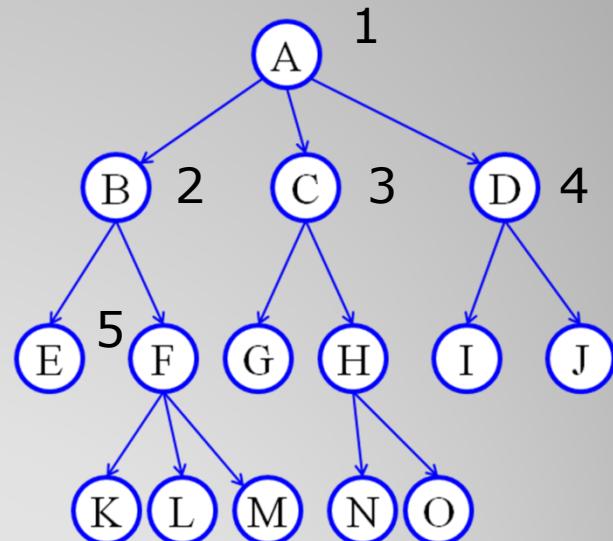
open= {B, C, D}

open= {C, D, E, F}

open= { D, E, F, G, H }

open= { E, F, G, H, I, J}

...



Tìm kiếm rỗng - Ví dụ 2

Quá trình biến đổi của open:

open= {A}

open= {B, **C**}

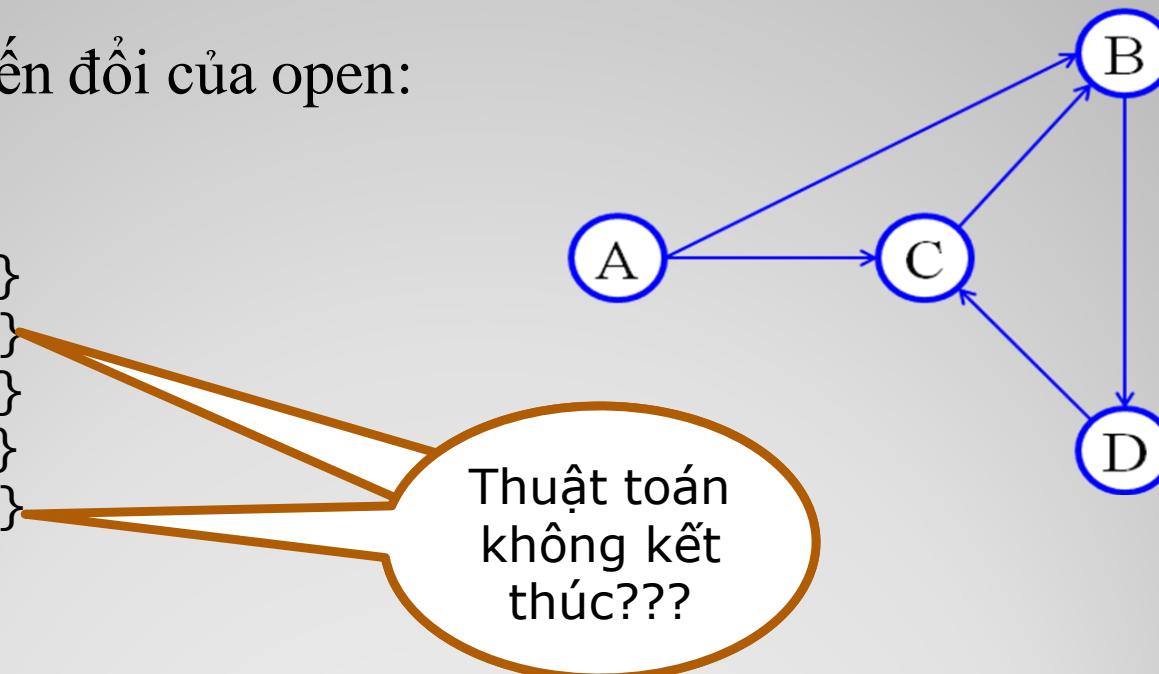
open= {C, **D**}

open= {D, **B**}

open= {B, **C**}

open= {C, **D**}

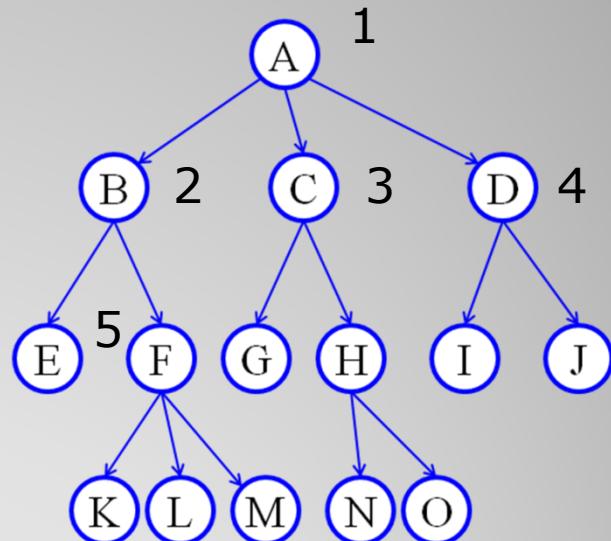
...



Tìm kiếm rỗng – Thuật toán 2

Thuật toán tìm kiếm rỗng:

1. Cho đỉnh xuất phát vào open.
2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh đầu trong open ra và gọi đó là Θ . Cho Θ vào closed
4. Nếu Θ là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của Θ không thuộc open và closed cho vào cuối của open
6. Trở lại bước 2.



Tìm kiếm rỗng – Đánh giá thuật toán

Tổng số đỉnh cần xét tối đa là:

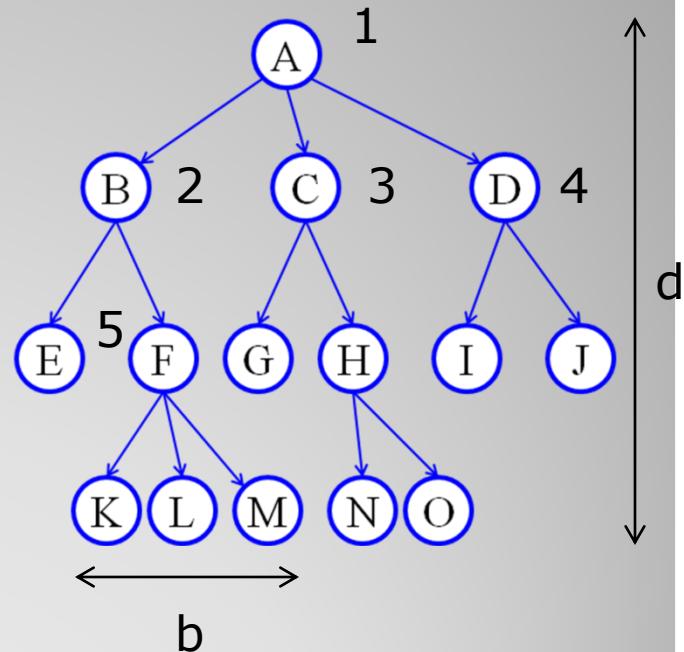
$$1 + b + b^2 + \dots + b^d.$$

Trong đó:

b : số nhánh tối đa của 1 trạng thái

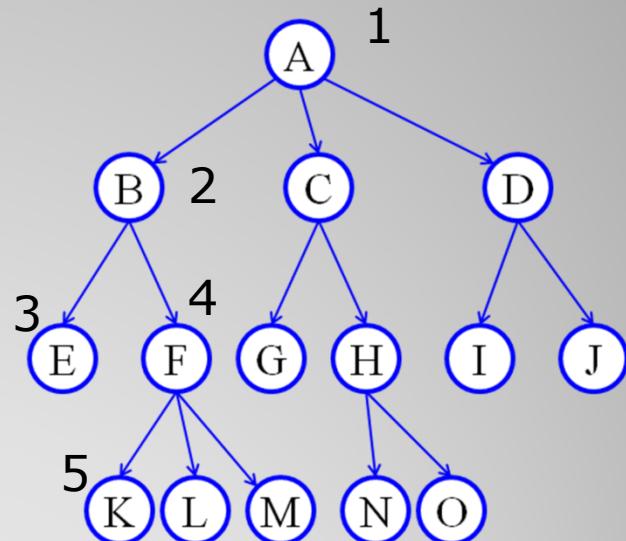
d : độ sâu

Như vậy độ phức tạp về thời gian và không gian (bộ nhớ) của thuật toán là $O(b^d)$.



Tìm kiếm sâu

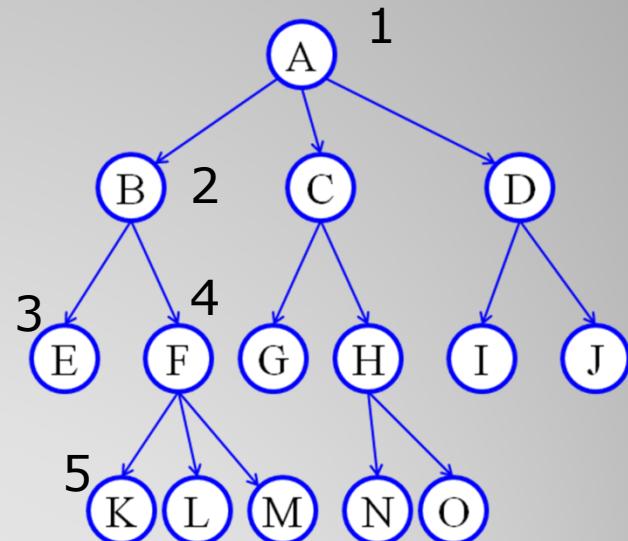
- ✓ Tìm kiếm sâu là phương pháp tìm kiếm mà ưu tiên tìm kiếm những đỉnh xa với đỉnh xuất phát.
- ✓ Thứ tự tìm kiếm: $A \rightarrow B \rightarrow E \rightarrow F \rightarrow K \rightarrow L \rightarrow M \rightarrow C \rightarrow G \rightarrow H \rightarrow N \rightarrow O \rightarrow D \rightarrow I \rightarrow J$.



Tìm kiếm sâu

Thuật toán tìm kiếm sâu:

1. Cho đỉnh xuất phát vào open.
2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh đầu trong open ra và gọi đó là O. Cho O vào closed
4. Nếu O là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của O không thuộc open và closed **cho vào đầu của open**
6. Trở lại bước 2.



*Open hoạt động theo quy tắc LIFO

Tìm kiếm sâu – Ví dụ

Quá trình biến đổi của open:

open= {A}

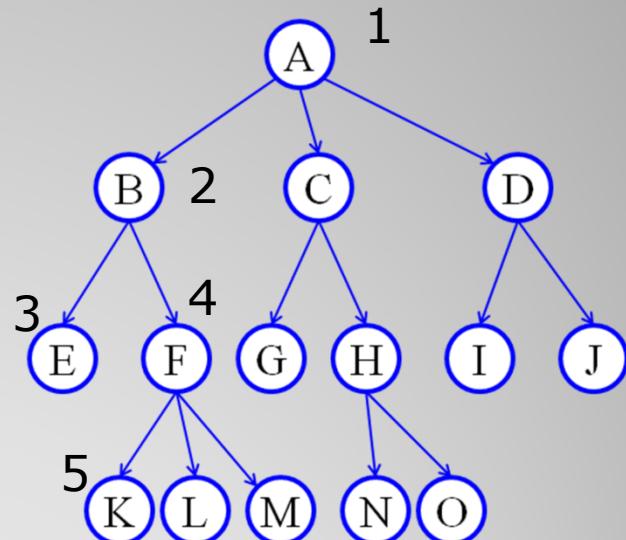
open= {B, C, D}

open= { E, F, C, D }

open= { F, C, D }

open= { K, L, M, C, D }

...



Tìm kiếm sâu – Đánh giá thuật toán

Tổng số đỉnh cần xét tối đa là:

$$1 + b + b^2 + \dots + b^d.$$

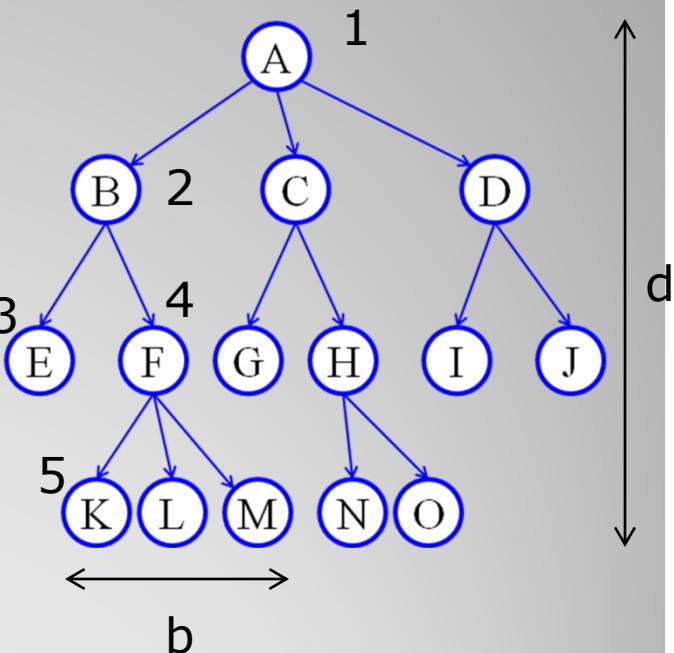
Tổng số đỉnh tối đa trong open là: b^d

Trong đó:

b : số nhánh tối đa của 1 trạng thái

d : độ sâu

Như vậy độ phức tạp về thời gian của thuật toán là $O(b^d)$ và không gian (bộ nhớ) của thuật toán là $O(bd)$.



Kết luận

- Giới thiệu một số bài toán và cách biểu diễn bài toán trong không gian trạng thái
- Biểu diễn trên
 - Đồ thị tìm kiếm
 - Cây tìm kiếm
- Tìm kiếm không có thông tin
 - Tìm kiếm rộng
 - Tìm kiếm sâu

Tìm kiếm có thông tin

- ✓ Tìm kiếm Heuristic
 - ✓ Tìm kiếm chi phí cực tiểu (Uniform cost search)
 - ✓ Tìm kiếm ước lượng (Best-first search)
 - ✓ A* algorithm
- ✓ Kiếm trên cây trò chơi (Game tree search)

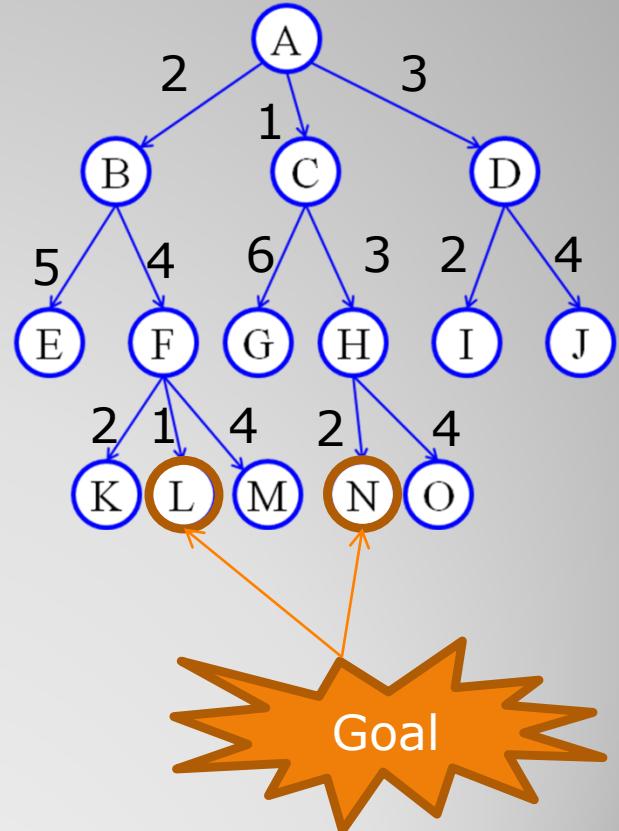
Uniform cost search

- ✓ Đặt vấn đề
 - ✓ Khi một bài toán có nhiều nghiệm thì thường chọn những phương án có “**chi phí thấp nhất.**”
- ✓ Ví dụ:
 - ✓ Tìm đường đi từ TP. A đến TP. B sao cho:
 - ✓ Nhanh nhất
 - ✓ Rẻ nhất
 - ✓ Các phương pháp tìm kiếm sâu và tìm kiếm rộng không đảm bảo việc tối ưu cho bài toán “**chi phí thấp nhất.**”
 - ✓ Uniform cost search
 - ✓ Đảm bảo tìm được “**chi phí thấp nhất.**”

Uniform cost search

Thuật toán tìm kiếm chi phí cực tiểu:

1. Cho đỉnh xuất phát vào open.
2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh đầu trong open ra và gọi đó là O. Cho O vào closed
4. Nếu O là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của O không thuộc open và closed cho vào open **theo thứ tự tăng dần về khoảng cách từ đỉnh xuất phát**.
6. Trở lại bước 2.



Uniform cost search – Ví dụ

Quá trình biến đổi của open:

open= {A(0)}

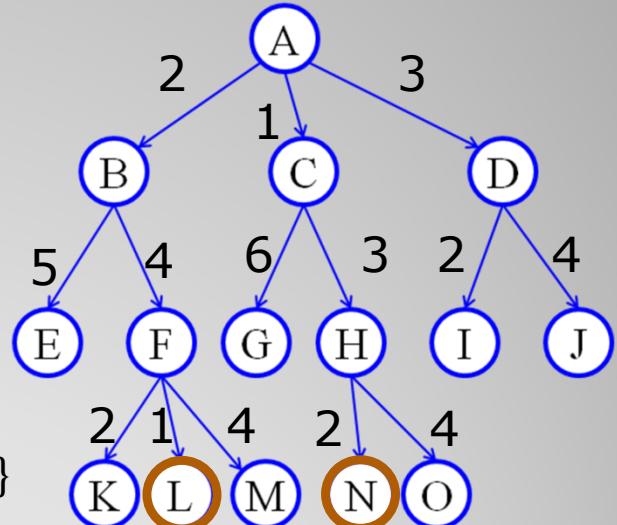
open= {C(1), **B**(2), **D**(3)}

open= {B(2), D(3), **H**(1+3=4), **G**(1+6=7)}

open= {D(3), H(4), **F**(2+4=6), G(7), E(2+5=7)}

open= {H(4), I(3+2=5), **F**(6), G(7), **E**(7), J(3+4=7)}

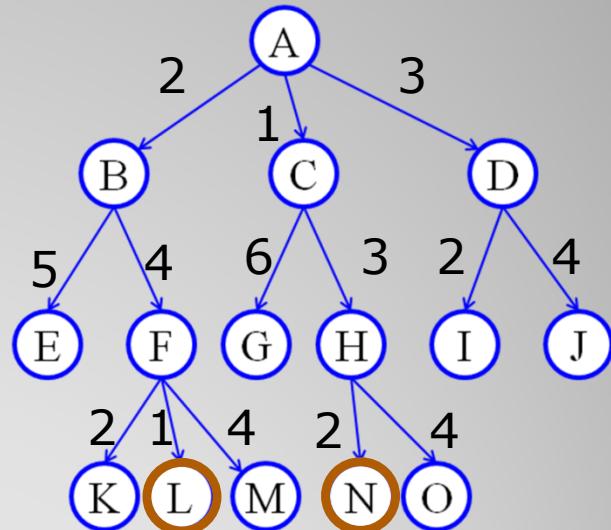
...



(Bài tập: Tiếp tục quá trình trên đến khi thuật toán kết thúc)

Uniform cost search – Tính chất

- ✓ Lời giải được phát hiện đầu tiên cũng là nghiệm có “chi phí thấp nhất”
- ✓ Nếu bài toán có lời giải, thì đảm bảo thuật toán sẽ dừng
- ✓ Tất cả các nhánh có chi phí là 1 thì thuật toán trở thành tìm kiếm rộng



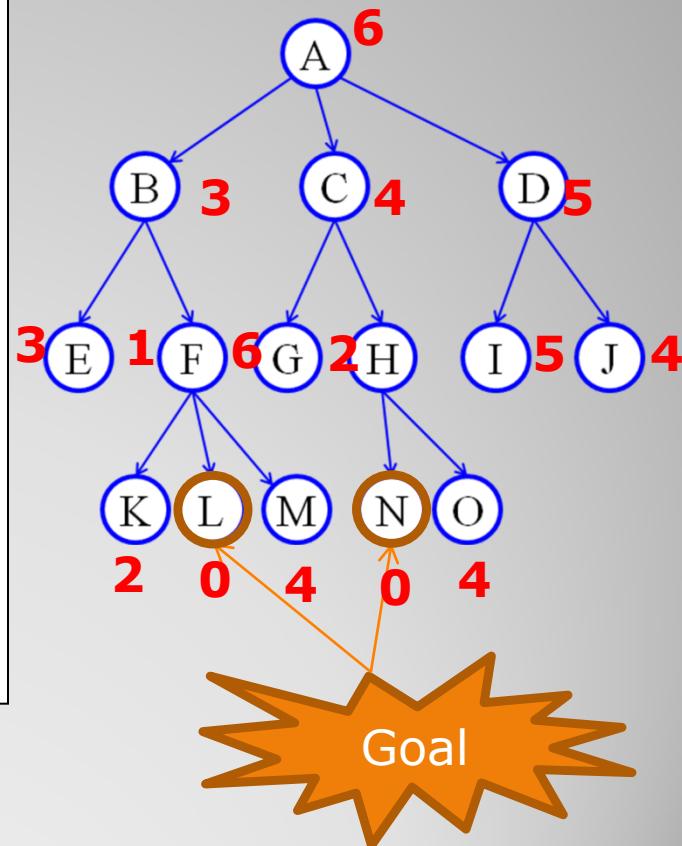
Tại sao lại tìm kiếm Heuristic?

- ✓ Heuristic là gì?
 - ✓ Theo kinh nghiệm
- ✓ Ví dụ:
 - ✓ Khi không rõ đường đi, thường thì mọi người nhắm về hướng cần tới để đi. (Ưu tiên đường đi mà có khoảng cách đường chim bay ngắn nhất)
 - ✓ Tìm kiếm Heuristic là tìm kiếm sử dụng kiến thức phỏng đoán chi phí từ đỉnh đang xét đến đích.
 - ✓ Thuật toán

Best-first search

Thuật toán tìm kiếm ước lượng:

1. Cho đỉnh xuất phát vào open.
2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh đầu trong open ra và gọi đó là O. Cho O vào closed
4. Nếu O là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của O không thuộc open và closed cho vào open **theo thứ tự tăng dần về khoảng cách ước lượng đến đích.**
6. Trở lại bước 2.



Best-first search – Ví dụ

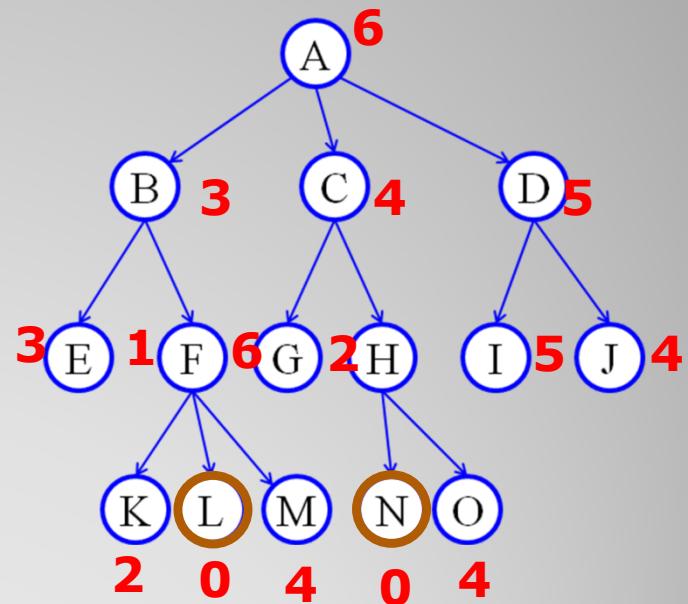
Quá trình biến đổi của open:

open= {A(6)}

open= {B(3), C(4), D(5)}

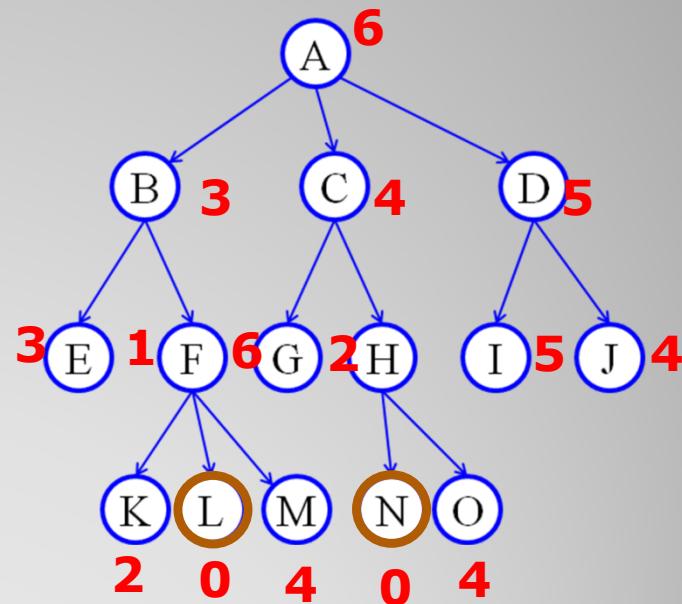
open= {F(1), E(3), C(4), D(5)}

open= {L(0), K(2), E(3), C(4), M(4), D(5)}



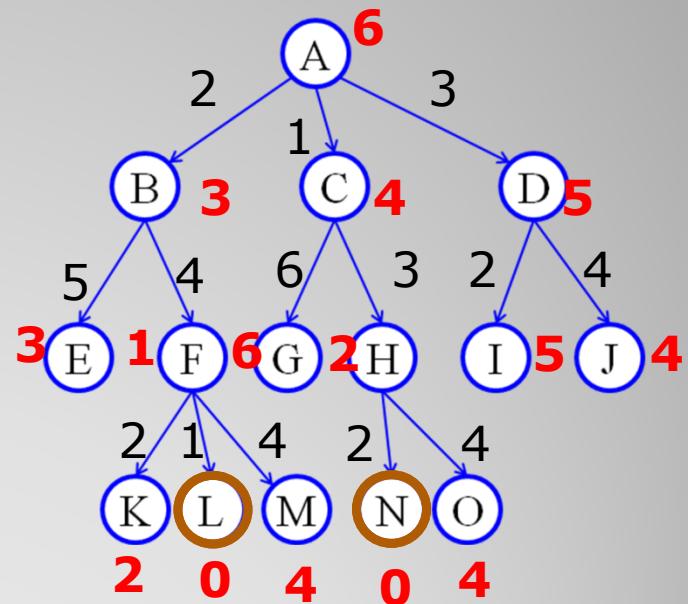
Best-first search – Tính chất

- ✓ Lời giải được phát hiện đầu tiên **không đảm bảo** “chi phí thấp nhất”
- ✓ So với **Uniform cost search** thì tiết kiệm về mặt không gian hơn



Thuật toán A* (A* Algorithm)

- ✓ Thuật toán A* kết hợp **Uniform cost search** và **Best-first search** nhằm tăng hiệu quả tìm kiếm
- ✓ Sử dụng khoảng cách từ đỉnh xuất phát đến đỉnh đang xét: $g(X)$
- ✓ Sử dụng khoảng cách ước lượng từ đỉnh đang xét đến đích $h'(X)$



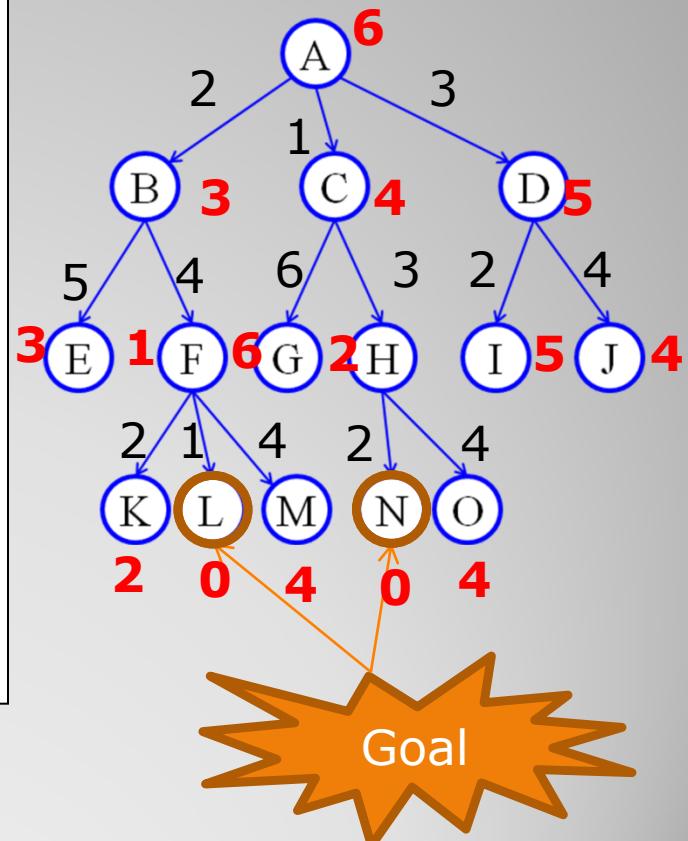
Thuật toán A* (A* Algorithm)

Thuật toán A*:

1. Cho đỉnh xuất phát vào open.
2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm.
3. Lấy đỉnh đầu trong open ra và gọi đó là O. Cho O vào closed
4. Nếu O là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.
5. Tìm tất cả các đỉnh con của O không thuộc open và closed cho vào open **theo thứ tự tăng dần đối với hàm $f(X)=g(X)+h'(X)$** .
6. Trở lại bước 2.

$g(X)$: Khoảng cách từ đỉnh xuất phát đến X

$h'(X)$: Khoảng cách ước lượng từ X đến đích



Thuật toán A* (A* Algorithm) – Ví dụ

Quá trình biến đổi của open:

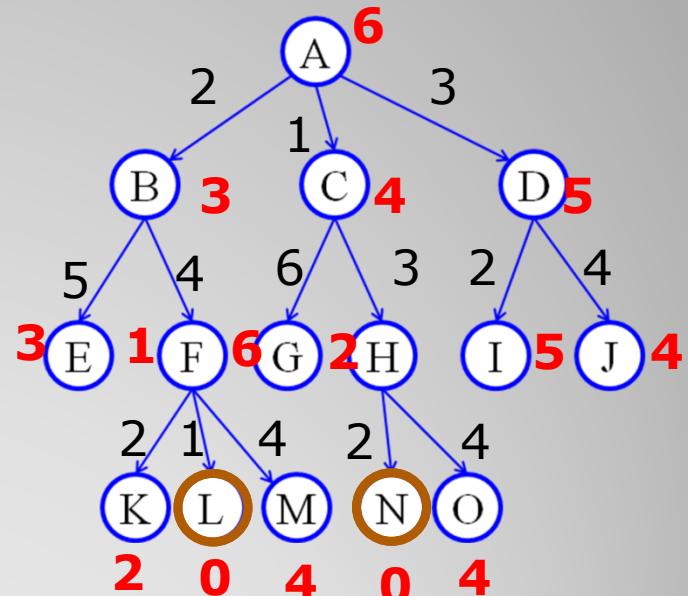
open= {A(0+6)}

open= {B(2+3), C(1+4), D(3+5)}

open= {C(1+4), F(6+1), D(3+5) , E(7+3)}

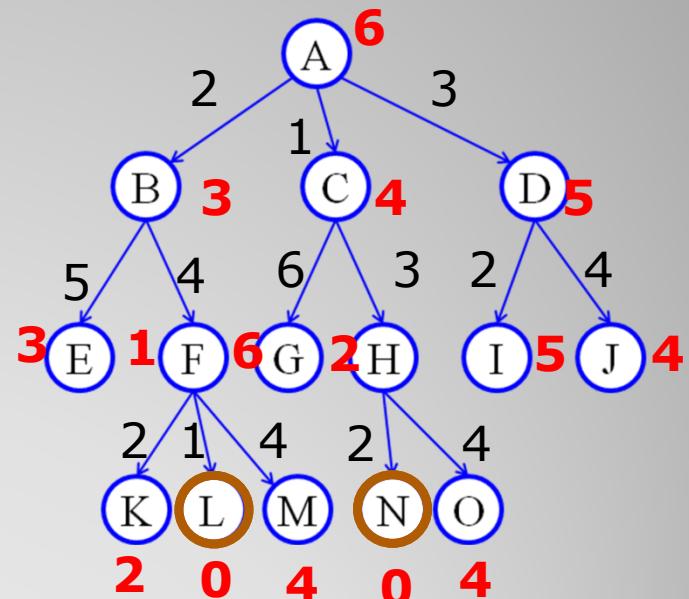
open= {H(4+2), F(6+1), D(3+5) , E(7+3), G(7+6)}

open= {N(6+0), F(6+1), D(3+5) , E(7+3), O(8+4), G(7+6)}



Thuật toán A* – Tính chất

- ✓ Lời giải được phát hiện đầu tiên **đảm bảo** “chi phí thấp nhất” khi tất cả khoảng cách ước lượng từ đỉnh đang xét đến đích nhỏ hơn khoảng cách thực. (*)
- ✓ So với **Uniform cost search** thì tiết kiệm về mặt không gian
- ✓ Nhanh hơn so với thuật toán khác (Từ kết quả thực nghiệm)



- ✓ BT: Chứng minh (*)?

- ✓ Gợi ý: Hãy chứng minh với mọi x là các đỉnh thuộc đường đi có chi phí thấp nhất thì $f(x) < f(G)$, G là đỉnh đích không phải chi phí thấp nhất.

Kiếm trên cây trò chơi

Phạm Minh Tuấn, PhD

Kiếm trên cây trò chơi (Game tree search)

✓ Đặt vấn đề

- ✓ Thuật toán A* chỉ thích hợp với các bài toán không có tính đối kháng
 - ✓ VD: Puzzle, Mê cung, n con Hậu
- ✓ Các bài toán có tính đối kháng như: Caro, cờ vua,... thì mọi người chơi đều muốn mình đến được đích và **cản phá đối phương** không tới được đích. → thuật toán A* không phát huy được tác dụng

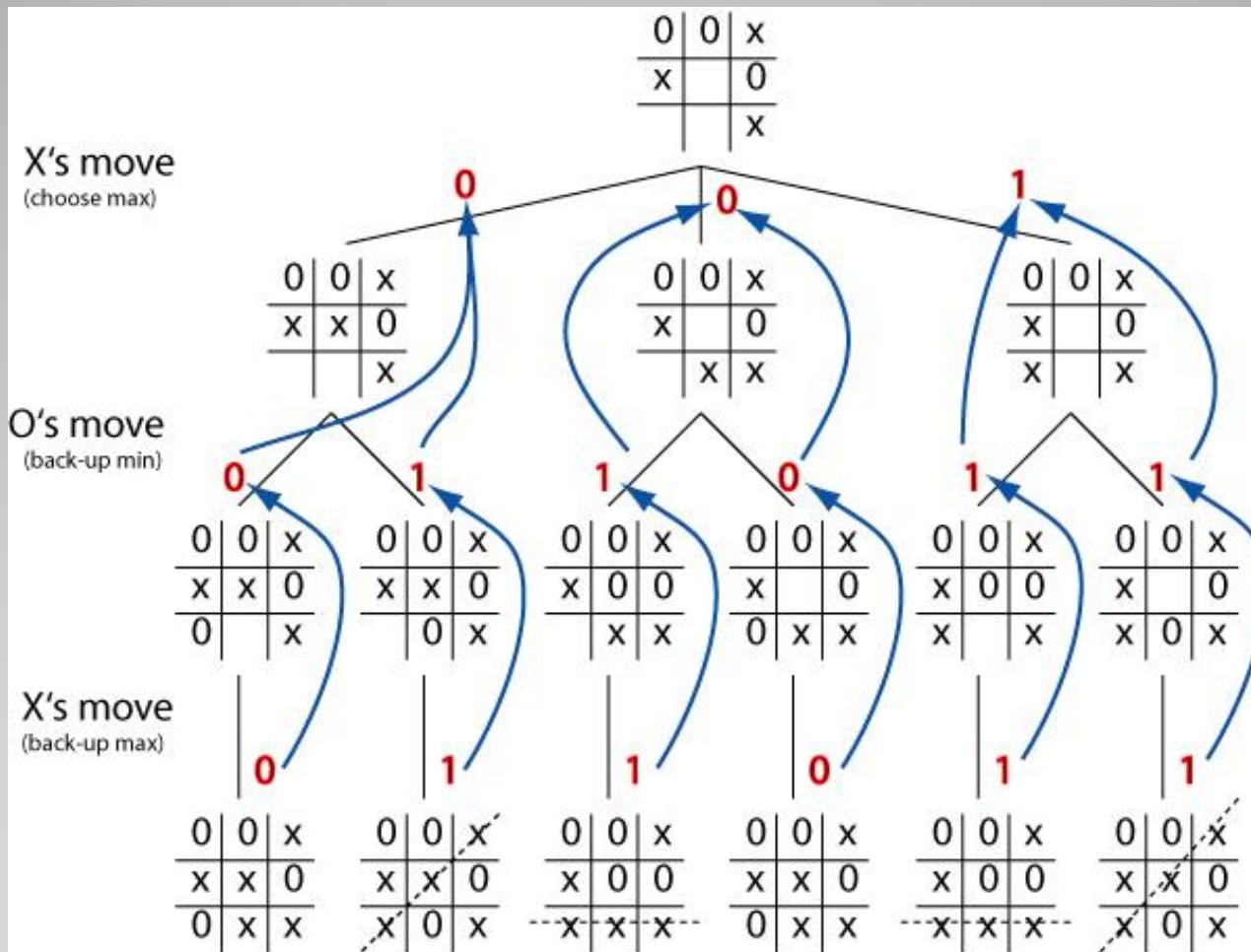
Giải thuật MiniMax

- ✓ Giải thuật MiniMax được xây dựng trên 2 giả thiết:
 - ✓ Cả 2 đối thủ có cùng kiến thức và không gian trạng thái của trò chơi
 - ✓ Cả 2 đối thủ có cùng mức cố gắng như nhau.

Giải thuật MiniMax

- ✓ Trong trò chơi đối kháng 2 người
 - ✓ Mỗi bên thay phiên nhau chơi
 - ✓ Bên địch: Làm nhỏ tối đa lợi thế của ta
 - ✓ Bên ta: Làm lớn tối đa lợi thế của ta
- ✓ Cây trò chơi
 - ✓ Phân thành các lớp Min-Max xen kẽ nhau
 - ✓ Lớp Min: lấy giá trị min của các node con
 - ✓ Lớp Max: lấy giá trị max của các node con

Giải thuật MiniMax - Ví dụ



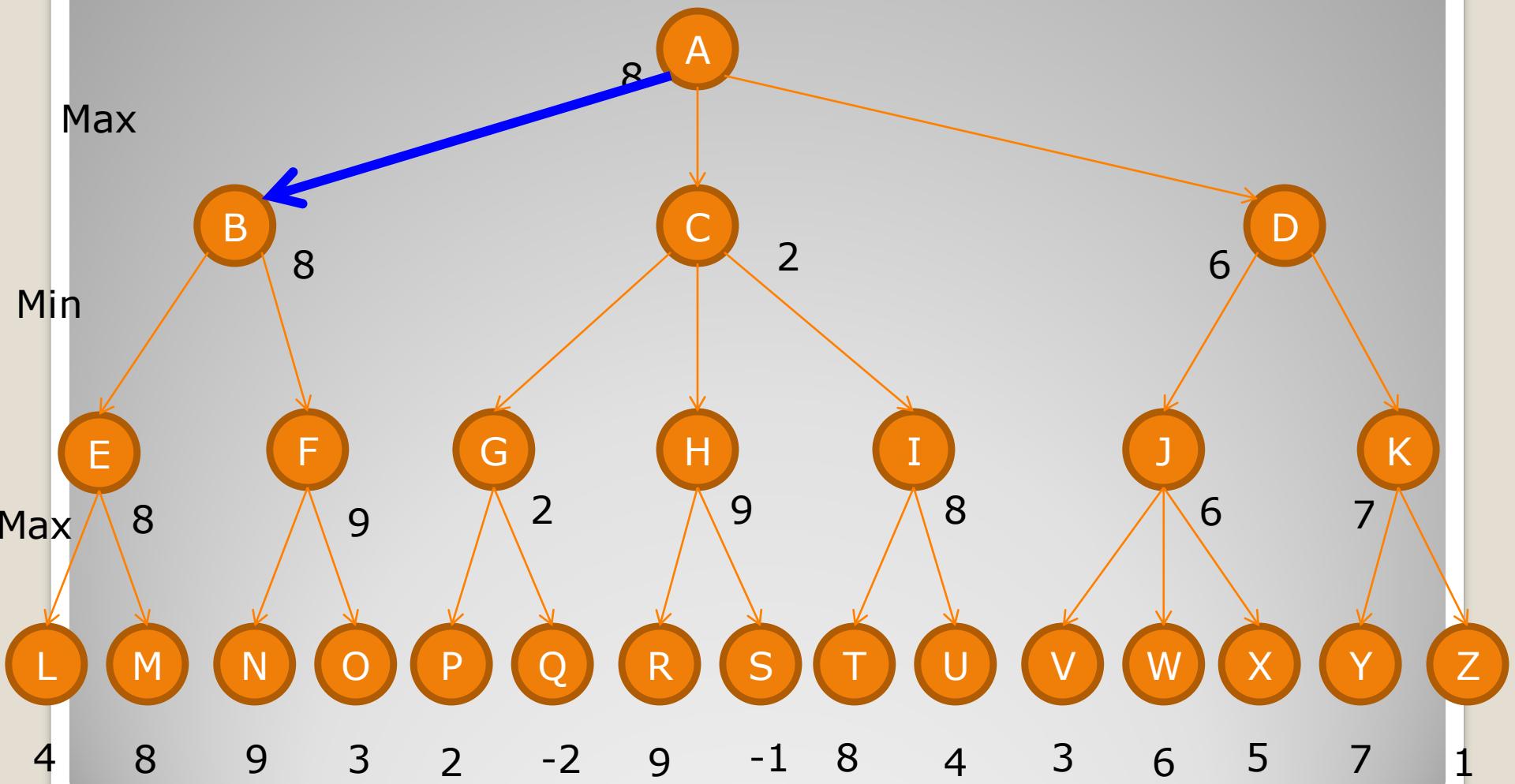
Minimax với độ sâu giới hạn

- ✓ Minimax như đã xét buộc phải có toàn bộ không gian trạng thái đã được triển khai để có thể gán trị cho các nút lá và tính ngược lại
→ Không khả thi vì không gian bài toán lớn
- ✓ Hướng giải quyết: **Giới hạn** không gian trạng thái theo nột **độ sâu** nào đó và giới hạn các **node con** theo một quy tắc nào đó

Thuật toán MiniMax

```
function minimax (node, depth, maximizingPlayer)
    if node is "End Node" or depth = 0
        return value(node)
    if maximizingPlayer
        Max=-∞
        foreach child of node
            Max= max(Max, minimax(child, depth-1, False))
        return Max
    else
        Min=∞
        foreach child of node
            Min := min(Min, minimax(child, depth-1, True))
        return Min
```

Minimax (Ví dụ)



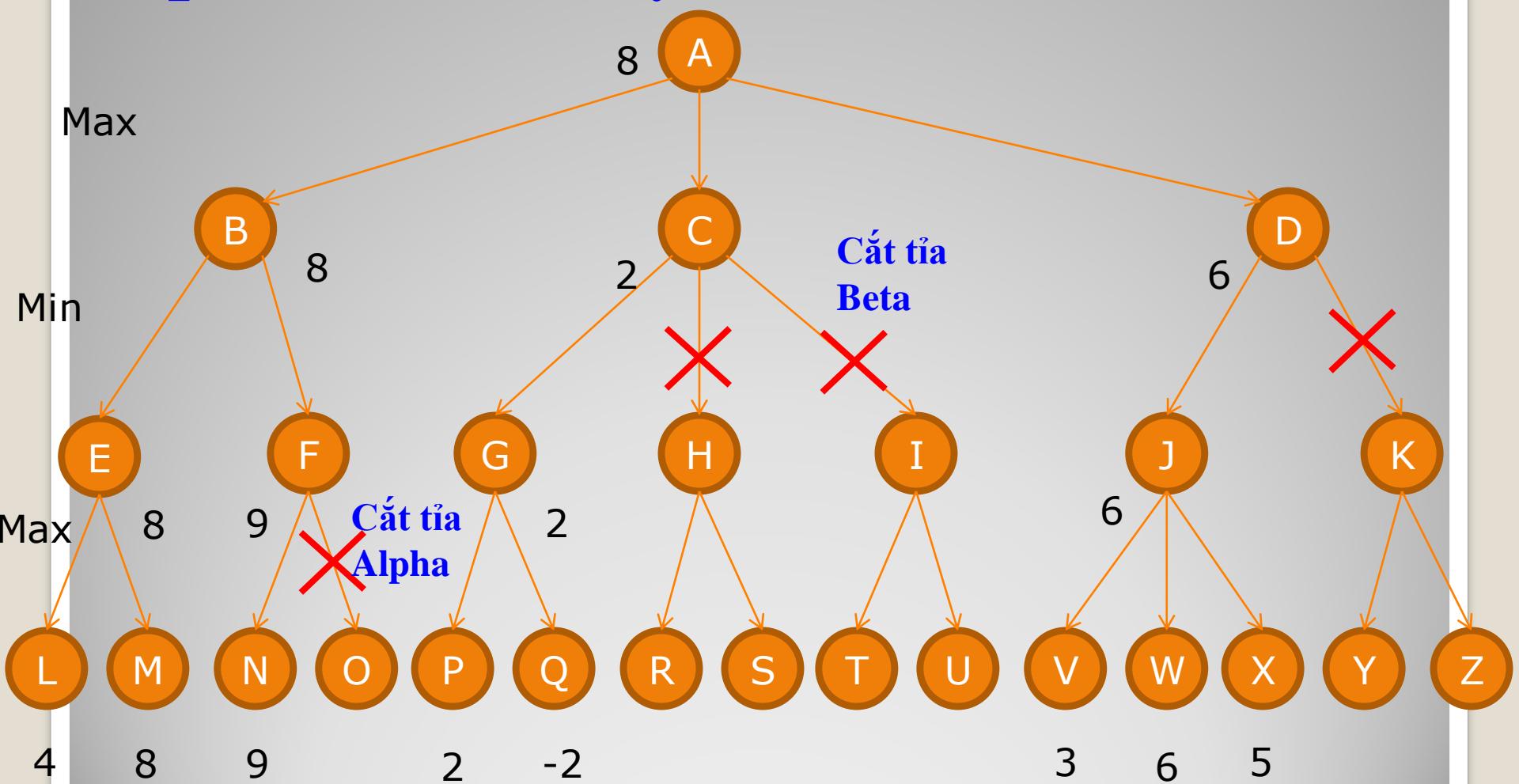
Minimax với độ sâu giới hạn

- ✓ Trong việc tìm kiếm Minimax, cho dù giới hạn độ sâu thì số đỉnh con của một đỉnh bất kỳ vẫn rất lớn trong thực tế
- ✓ Ví dụ:
 - ✓ Cờ vua: số nhánh trung bình khoảng 35.
 - ✓ Vậy nếu cần suy nghĩ độ sâu là 4 thì phải mất ít nhất $35^4 = 1.5 \cdot 10^6$ số phép đánh giá.
 - ✓ → Cần một phương pháp **làm giảm số nhánh**

Cắt tỉa alpha-beta

- ✓ Mục đích làm giảm số nhánh trong cây tìm kiếm và,
- ✓ Không làm ảnh hưởng đến sự đánh giá của đỉnh đang xét.
- ✓ → Phương pháp cắt tỉa alpha-beta cho phép cắt bỏ các nhánh không cần thiết cho sự đánh giá của đỉnh đang xét.

Alpha-beta (Ví dụ)



Cắt tỉa Alpha beta- Thuật Toán 1

```
function minimax (node, depth, maximizingPlayer)
    return alphabeta(node, depth, - ∞, ∞,maximizingPlayer)
```

```
function alphabeta (node, depth, a, b, maximizingPlayer)
    if node is "End Node" or depth = 0
        return value(node)
    if maximizingPlayer
        foreach child of node
            a= max(a, alphabeta (child, depth-1, a, b, False))
            if a>=b break;
        return a
    else
        foreach child of node
            b := min(b, alphabeta (child, depth-1, a, b, True))
            if a>=b break;
        return b
```

Cắt tỉa Alpha beta – Thuật Toán 2

```
function minimax (node, depth)
    return alphabeta(node, depth, - ∞, ∞)
```

```
function alphabeta (node, depth, a, b)
    if node is "End Node" or depth = 0
        return value(node)
    foreach child of node
        a= max(a, -alphabeta (child, depth-1, -b, -a))
        if a>=b break;
    return a
```

Demo

Lecture 2 of Artificial
Intelligence

Chương 5: HỆ LUẬT SINH

Production system

- ❖ Tìm kiếm đệ qui
- ❖ Hệ luật sinh: Định nghĩa và ứng dụng
- ❖ Tìm kiếm trên hệ luật sinh

Phạm Minh Tuấn

Đặc tính dữ liệu và điều khiển của SSS

- Các đặc tính của giải thuật SSS:

- Lời giải là một PATH từ điểm START đến điểm GOAL
 - Tìm kiếm là sự kiểm tra có hệ thống các đường dẫn đến GOAL
 - Backtracking cho phép giải thuật “phục hồi” khi đi vào một nhánh không có đáp án.
 - Các danh sách sẽ giữ các trạng thái đang xem xét:
 - Danh sách Open: cho phép hệ thống backtrack về các trạng thái chưa được xét.
 - Danh sách Close: cho phép hệ thống kiểm tra sự quay vòng tránh lặp vô tận
 - Dùng STACK cho DFS, QUEUE cho BFS và dùng PRIORITY QUEUE cho BFS.

Tìm kiếm đệ qui – Tại sao???

- Tìm kiếm đệ qui là giải thuật tìm kiếm trên SSS với các đặc tính:
 - Ngắn gọn xúc tích hơn
 - Tiếp cận của giải thuật tự nhiên hơn
 - Hợp nhất với phương thức hiện thực của Logic vị từ
- Các giải thuật tìm kiếm đệ qui chính:
 - Tìm kiếm đệ qui – Recursive Search (RS)
 - Tìm kiếm theo mẫu – Pattern Directed Search (PDS)
- Các giải thuật tìm kiếm đệ qui được sử dụng rộng rãi trong các shell của các Hệ chuyên gia (Expert System).
- Pattern Directed Search là nền tảng của PROLOG

Thê nào là đệ qui?

- Đệ qui là sự định nghĩa một đối tượng bằng cách sử dụng chính đối tượng đó – Toán học
- Đệ qui được dùng để định nghĩa và phân tích các cấu trúc dữ liệu cũng như các thủ tục xử lý trong ngành máy tính.
- Một thủ tục đệ qui bao gồm:
 - Thành phần đệ qui, trong đó thủ tục gọi chính nó để lặp lại chuỗi các thao tác.
 - Thành phần dừng dùng để dừng quá trình đệ qui vô tận. (lặp vô tận)
- Hai thành phần này tồn tại đồng thời trong tất cả các định nghĩa đệ qui cũng như giải thuật đệ qui.
- Đệ qui là một cấu trúc điều khiển dữ liệu tự nhiên cho những cấu trúc không xác định số phần tử cố định: list, tree, và đồ thị.

Thủ tục đệ qui – ví dụ

```
Function Member(item, list);
begin
if List rỗng then return (Fail)
else
if Item = phần tử đầu của list then
return (succes)
else
begin
Tail:= List \ phần tử đầu;
member (item, Tail);
end
end;
```

- Đệ qui có đầy đủ tính năng của các cấu trúc điều khiển truyền thống như Loop và rẽ nhánh → mọi chương trình viết được bằng cấu trúc truyền thống đều có thể viết đệ qui.
- Đệ qui thích hợp biểu diễn các cấu trúc toán học → thuận tiện trong việc kiểm tra tính đúng đắn của giải thuật đệ qui.
- Công thức đệ qui cũng thường được dùng trong việc sinh và kiểm tra chương trình tự động.
- Đệ qui là công cụ tự nhiên và mạnh mẽ cho hiện thực các chiến lược giải quyết vấn đề của AI.

Tại sao?

Giải thuật DFS đệ qui - DFS

Function Depth_First_Search;

Begin

if Open rỗng then return (fail);

Current_state := phần tử đầu tiên của open;

If (current_state là mục tiêu) then return (Success)

else begin

open:=phần đuôi của open;

Closed := Closed + current_state;

for mỗi phần tử con Y của current_state do

if not (Y in close) and not (Y in open) then thêm Y vào đầu của Open;

End;

depth_first_search;

End;

Nhược điểm của recursive ?

Pattern-Directed Search (PDS)

- Các giải thuật Search đã tìm hiểu và Recursive Seach không trình bày cách biểu diễn một trạng thái trong không gian trạng thái cũng như cách sinh các trạng thái mới.
- Pattern-Directed Search là một giải thuật search đệ quy dùng Logic Vị từ để hiện thực việc sinh các trạng thái mới.
- Pattern-Directed Search xuất phát từ goal và các modus ponens dạng $q(x) \rightarrow p(x)$ để chuyển trạng thái. Các modus ponens này gọi là các luật sinh.
- Giải thuật: Xuất phát từ goal P, áp dụng một giải thuật để tìm các rule với P ở vế phải, sau đó xem vế trái Q là subgoal. Đệ quy với Q cho đến khi Qx là một sự kiện trong kho tri thức.

Sự kiện (FACT) trong kho tri thức?????

Giải thuật PDS

```
Function Pattern_search(current_goal);
Begin
If current_goal có trong closed then return
fail
else thêm current_goal vào trong closed;
while còn trong database các rule hay fact
chưa xét
begin case
current_goal trùng với fact:
    return(success);
current_goal là một phép hội:
begin
    for mỗi thành phần hội Pi do
        pattern_search(Pi);
```

If tất cả các hội đều success
then return success else return fail.
end;

Current_goal là vé phải một rule:
begin
 áp dụng các thành phần vào vé trái Q.
 if pattern_search(Q) then return success
 else return fail;
end;
end; /* case
Return fail;
End;

Giải thuật PDS

- PDS dùng các rule và thành phần hội để sinh các trạng thái con.
- Tách bạch quá trình điều khiển của giải thuật và dữ liệu của giải thuật
 - → Cùng giải thuật chỉ cần thay đổi database : Fact & Rule ta sẽ áp dụng cho các bài toán khác nhau.
 - → Xây dựng các shell và có thể vận hành cho các hệ thống khác nhau bằng cách thay đổi Database.
- Để đơn giản hóa giải thuật chưa giải quyết ở mức độ có các biến trong các rule. Ví dụ $P(x) \wedge Q(x)$ chỉ thỏa khi P và Q cùng thỏa với cùng giá trị X .
- Các phép \neg , \vee , ... cũng chưa giải quyết trong giải thuật này.

Ví dụ: Bài toán mã đi tuần

Đặc tả bài toán: tìm đường đi cho con mã qua tất cả các ô trên bàn cờ. Ví dụ với bàn cờ 3x3.

move(1,8)	move(1,6)	move(2,9)	move(2,7)
move(3,4)	move(3,8)	move(4,9)	move(4,3)
move(6,1)	move(6,7)	move(7,2)	move(7,6)
move(8,3)	move(8,1)	move(9,2)	move(9,4)

1	2	3
4	5	6
7	8	9

$\forall X \text{ path}(X,X);$

$\forall X, Y [\text{path}(X,Y) \leftarrow \exists Z[\text{move}(X,Z) \wedge \text{path}(Z,Y)]]$

Giải thuật PDS đầy đủ

```
Function Pattern_search(current_goal);  
Begin  
If current_goal có trong Closed then return fail  
else thêm current_goal vào Closed;  
while còn các rule hay fact chưa xét  
begin  
case  
current_goal trùng với fact:    return(success);  
current_goal là negated( $\neg p$ ):  
begin  
if pattern_search(p) then return fail  
else return{ }  
end;  
current_goal là một phép hội:  
begin  
for mỗi thành phần hội Pi do  
if not (pattern_search(Pi)) the return fail  
else thay thế tất cả các biến cho các thành  
phần hội khác.
```

If tất cả các hội đều success then return các thành phần hội else return fail.
end;
current_goal là phép tuyển:
begin
repeat cho mọi thành phần tuyển Vi;
until (pattern_search(Vi) or (hết thành phần hội))
if pattern_search (Vi) then return các thay thế else return fail;
end;
Current_goal là vế phải một rule:
begin
áp dụng các thành phần vào vế trái Q.
if pattern_search(Q) then return kết hợp của Current_goal và các thay thế của Q
else return fail;
end; end; /* case */
return fail; End;

Hệ Luật Sinh – Production System

- **Khái niệm:** Hệ luật sinh là một mô hình tính toán quan trọng trong các bài toán tìm kiếm cũng như mô phỏng cách giải quyết vấn đề của con người trong lĩnh vực ứng dụng AI.
- **Định nghĩa:** Hệ luật sinh là một mô hình tính toán cung cấp cơ chế điều khiển Pattern-directed trong quá trình giải quyết vấn đề (Proplem solving process).
- **Cấu trúc hệ luật sinh bao gồm 3 thành phần:**
 1. Production rules (Tập luật sản sinh)
 2. Working memory (Vùng nhớ làm việc)
 3. Recognize-action control (Bộ điều khiển nhận dạng và hành động)

Định nghĩa (tt) – Production Rule

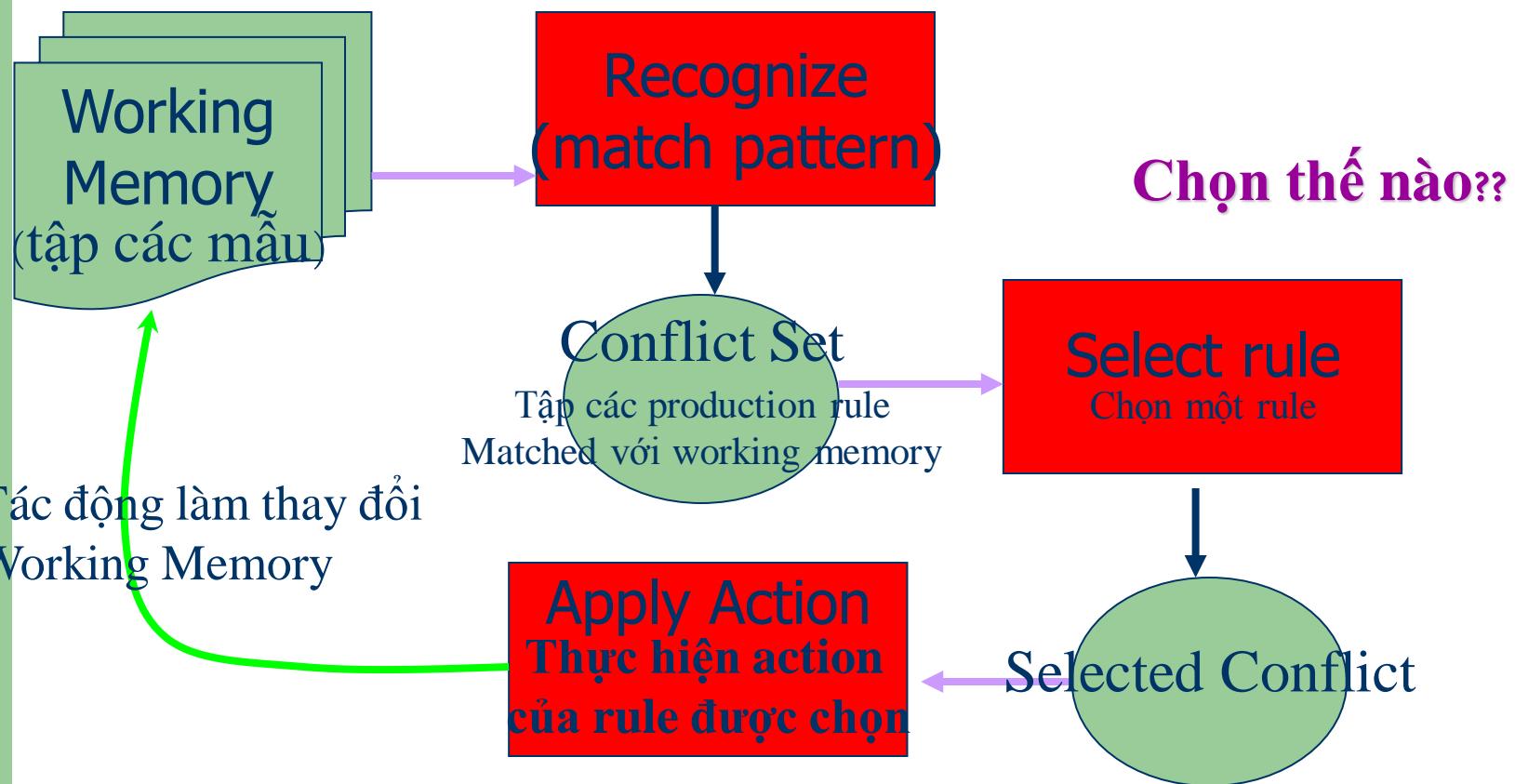
- **Production rules:** là một tập các luật sản sinh được đặc tả dạng: **Condition – Action** (điều kiện – hành động)
- Một luật là một mắt xích của kho tri thức giải quyết vấn đề. Kho tri thức là một database của các production rules.
- Thành phần Condition: là một mẫu (pattern) dùng xác định điều kiện áp dụng của rule cho một vấn đề tương ứng.
- Thành phần action: mô tả bước giải quyết vấn đề tương ứng sẽ được thực hiện. Đây là phần sẽ tác động lên hiện trạng của không gian tìm kiếm.

Định nghĩa (tt) – Working memory

- **Working memory** chứa những đặc tả trạng thái hiện tại của quá trình suy luận. Chúng được lưu trữ như là tập các mẫu.
- Những đặc tả này là các mẫu để so trùng với các condition của các production rules.
- Khi một production rule được so trùng phần condition thì phần action của nó có thể được áp dụng, và phần action này được xây dựng đặc thù để tác động trực tiếp lên working memory.
- Working memory được khởi tạo bằng trạng thái bắt đầu của vấn đề cần giải quyết.
- Working memory diễn tả hiện trạng của vấn đề cần suy luận

Định nghĩa (tt) – Recognize-action

- Đây là cấu trúc điều khiển dùng trong production system. Quy trình hoạt động của Recognize -Action



Các vấn đề khác

- Chọn lựa Conflict để thực hiện

Có thể chọn bằng cách đơn giản hay áp dụng các chiến lược lựa chọn heuristic → Ứng dụng meta knowledge.

Nếu gọi chiến lược heuristic chọn conflict là meta knowledge thì knowledge “thường” ở đâu trong hệ thống?

- Các hệ luật sinh đơn thuần không cung cấp cơ chế để quay lui khi việc áp dụng các action làm cho working memory thay đổi dẫn đến lúc không còn production rule nào có thể áp dụng được → Hệ thống sẽ dừng.
- Cần cung cấp cơ chế backtracking (Thường dùng UNDO). Tuy nhiên cần chú ý để tránh lặp vòng.

Ví dụ: bài toán 8 Puzzle

- Working Memory
- Production Rules

Goal state

Blank is not on top edge

Blank is not on the right edge

Blank is not on the bottom edge

Blank is not on the left edge

- Recognize-Action

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

2	8	3
1	6	4
7		5

Current state

1	2	3
8		4
7	6	5

Goal state

→ Halt

→ Move the blank up

→ Move the blank right

→ Move the blank down

→ Move the left down

Điều khiển tìm kiếm trong hệ luật sinh

- **Chiến lược Data-Driven / Goal-Driven:**

- Data-Driven: bắt đầu với problem trong working memory, matching các condition trong production rule → conflict → áp dụng các action → thay đổi working memory. Lặp lại cho đến khi đạt được goal state.
- Goal-Driven: Bắt đầu với mô tả goal trong working memory, matching với các kết quả của Action → sinh tập các condition → đưa các condition vào trong working memory. Lặp lại cho đến khi working memory chứa FACT.

- **Điều khiển qua cấu trúc rule:** Dùng các biến đổi tương đương của các biểu thức trong rule để điều khiển quá trình tìm kiếm.

- **Điều khiển bằng sự phân tích các Conflict:** Các conflict có thể được chọn lựa thông qua các Heuristic. áp dụng các meta knowledge trong việc chọn conflict. Ví dụ:

- 1) **Refraction:** Khi một rule đã được áp dụng, nó sẽ không được dùng nữa cho đến khi thành phần trùng lặp với rule cũ trong working memory được thay đổi.
- 2) **Recency:** Chọn rule có phần condition match với phần mới thêm vào working memory. Theo đuổi một hướng triển khai.
- 3) **Specificity:** Rule nào càng được đặc tả chi tiết càng được ưu tiên cao.

Các ưu điểm của Hệ luật sinh

- Hệ luật sinh là khung làm việc tổng quát để thực thi các giải thuật tìm kiếm. Với đặc tính đơn giản, dễ sửa đổi, và linh động, hệ luật sinh được dùng như một công cụ quan trọng để xây dựng các hệ chuyên gia và các ứng dụng AI khác
- Các ưu điểm của Hệ luật sinh:
 - Tách bạch giữa Tri thức & Điều khiển:
 - Điều khiển: nằm trong chu trình Recognize-Action
 - Tri thức: được chứa đựng trong bản thân các luật sinh.
→ Cung cấp khả năng cập nhật tri thức mà không cần điều chỉnh chương trình. Thay đổi mã chương trình mà không ảnh hưởng đến tập luật sinh.
 - Dễ dàng áp dụng trong tìm kiếm trên không gian trạng thái. Các state của working memory là các node. Các production rule là các chuyển đổi giữa các trạng thái (cơ chế sinh các trạng thái mới)
 - Tính độc lập của các luật sinh.
 - Khả năng áp dụng heuristic cho việc điều khiển quá trình hoạt động.
 - Theo dõi và giải thích quá trình hoạt động
 - Độc lập với ngôn ngữ & có thể dùng như kỹ thuật mô phỏng giải pháp của người.

Chương 6: HỆ CHUYÊN GIA (ES)

- ❖ Giới thiệu về hệ chuyên gia
- ❖ Mô hình hệ chuyên gia: dự trên luật, dựa trên frame
- ❖ Phát triển một hệ chuyên gia

Phạm Minh Tuấn

Nội dung.

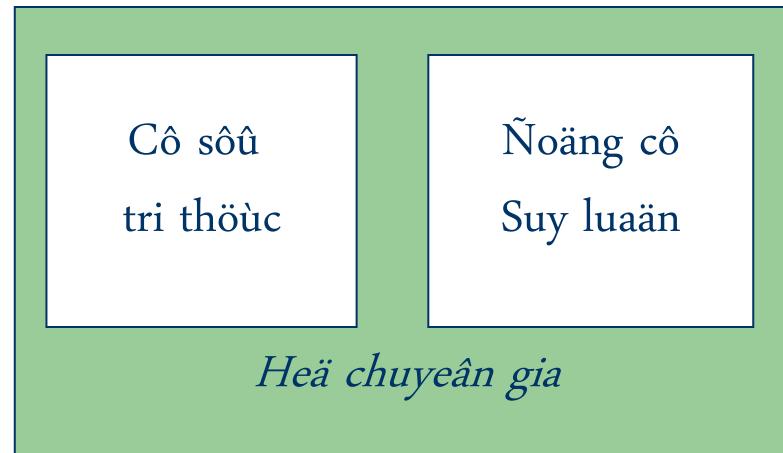
- Giới thiệu về hệ chuyên gia.
 - Định nghĩa, khả năng ứng dụng.
 - Cấu trúc, các đặc trưng cơ bản của ES.
 - Biểu diễn tri thức.
 - Các kỹ thuật suy luận.
- Khảo sát một vài hệ chuyên gia đã có.
 - XCON: ES trợ giúp cấu hình hệ thống máy tính của DEC
 - MYCIN: ES chuẩn đoán bệnh nhiễm trùng máu.
- Hệ chuyên gia dựa trên luật.
 - Kiến trúc, thiết kế.
 - Ưu - nhược điểm.
- Hệ chuyên gia dựa vào Frame.
 - Kiến trúc, thiết kế.
 - Ưu - nhược điểm.

Giới thiệu về hệ chuyên gia.

- Định nghĩa:

Hệ chuyên gia là một chương trình được thiết kế để theo mô hình có khả năng giải quyết vấn đề của chuyên gia con người.

- Sơ đồ khái cơ bản:



Giới thiệu về hệ chuyên gia.

- Cơ sở tri thức:

- ♣ Dùng để chứa tri thức trong một lĩnh vực nào đó, tri thức này do chuyên gia con người chuyển giao.

- ♣ Nó bao gồm: các khái niệm cơ bản, các sự kiện, các luật và quan hệ giữa chúng.

Ví dụ:

- Tri thức về *bệnh nhiễm trùng máu* do các bác sĩ chuyên khoa này chuyển giao.

- Tri thức về *chiến lược đầu tư* do các nhà cố vấn đầu tư chuyển giao.

- Tri thức về *sự diễn dịch dữ liệu khảo sát địa vật lý* do các kỹ sư địa chất chuyên giao.

- ...?

Giới thiệu về hệ chuyên gia.

- Động cơ suy luận:

Là bộ xử lý cho tri thức, được mô hình sao cho giống với việc suy luận của chuyên gia con người. Bộ xử lý này làm việc dựa trên thông tin mà người dùng mô tả về vấn đề, kết hợp với CSTT, cho ra kết luận hay đề nghị.

- Tạo sao phải xây dựng ES ?

Chuyên gia con người là tài nguyên quý giá cho nhiều tổ chức. Họ có thể giải quyết những vấn đề khó, hiệu quả,... Vậy có giá trị không khi chúng ta xây dựng một chương trình có khả năng như chuyên gia con người ? Một số mặt nào đó còn có thể hơn hẳn. Xem bảng so sánh sau:

Giới thiệu về hệ chuyên gia.

♣ Bảng so sánh:

Tiêu chí	CG con người	ES.
1. Sáng dùng	TG. hành chính	Mọi lúc.
2. Vị trí	Cục bộ	Mọi nơi.
3. An toàn	không thể thay thế	Có thể thay thế.
4. Có thể chết	Có	Không.
5. Hiệu suất	Thay đổi	Hàng số.
6. Tốc độ	Thay đổi	Hàng số <small>(thường nhanh hơn)</small>
7. Chi phí	Cao	Có thể cố gắng.

Giới thiệu về hệ chuyên gia.

- ♣ Vài lý do để phát triển ES thay cho chuyên gia con người:
 - ◆ Tạo cho tính chuyên gia sẵn dùng ở mọi nơi, mọi lúc.
 - ◆ Tự động hóa các công việc đòi hỏi chuyên gia.
 - ◆ Các chuyên gia đang nghỉ hưu hay chuyển đến nơi khác – Cần thay thế.
 - ◆ Thuê chuyên gia với chi phí quá lớn.
 - ◆ Tính chuyên gia cần thiết trong các môi trường làm việc không thân thiện, ở đó hỏi một ES sẽ nhanh hơn một chuyên gia con người.
 - ◆ Phát triển ES để trợ giúp cho chuyên gia con người.

Giới thiệu về hệ chuyên gia.

- Các kiểu vấn đề thường được giải quyết bởi ES:

 Điều khiển:

 Thiết kế:

 Chuẩn đoán:

 Dạy học:

 Diễn dịch:

 Giám sát:

 Hoạch định:

 Dự đoán:

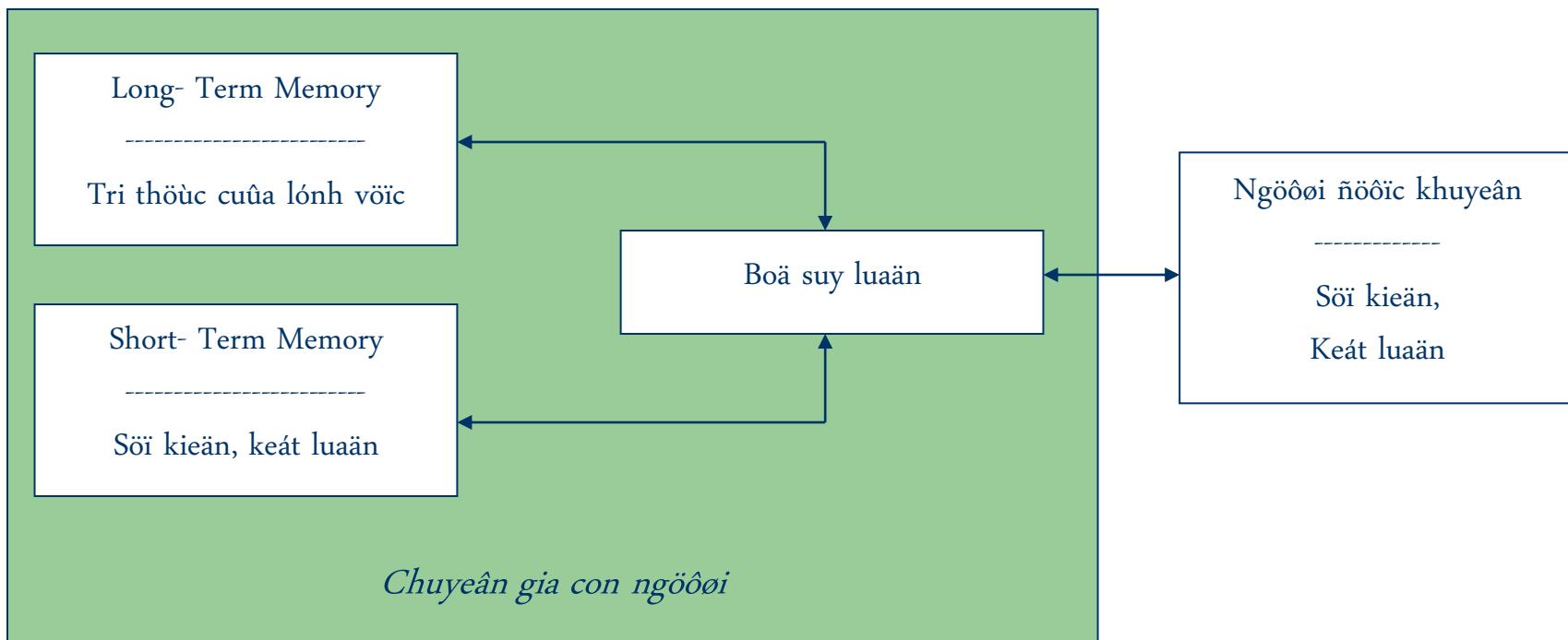
 Lựa chọn:

 Mô phỏng:

Cấu trúc của ES.

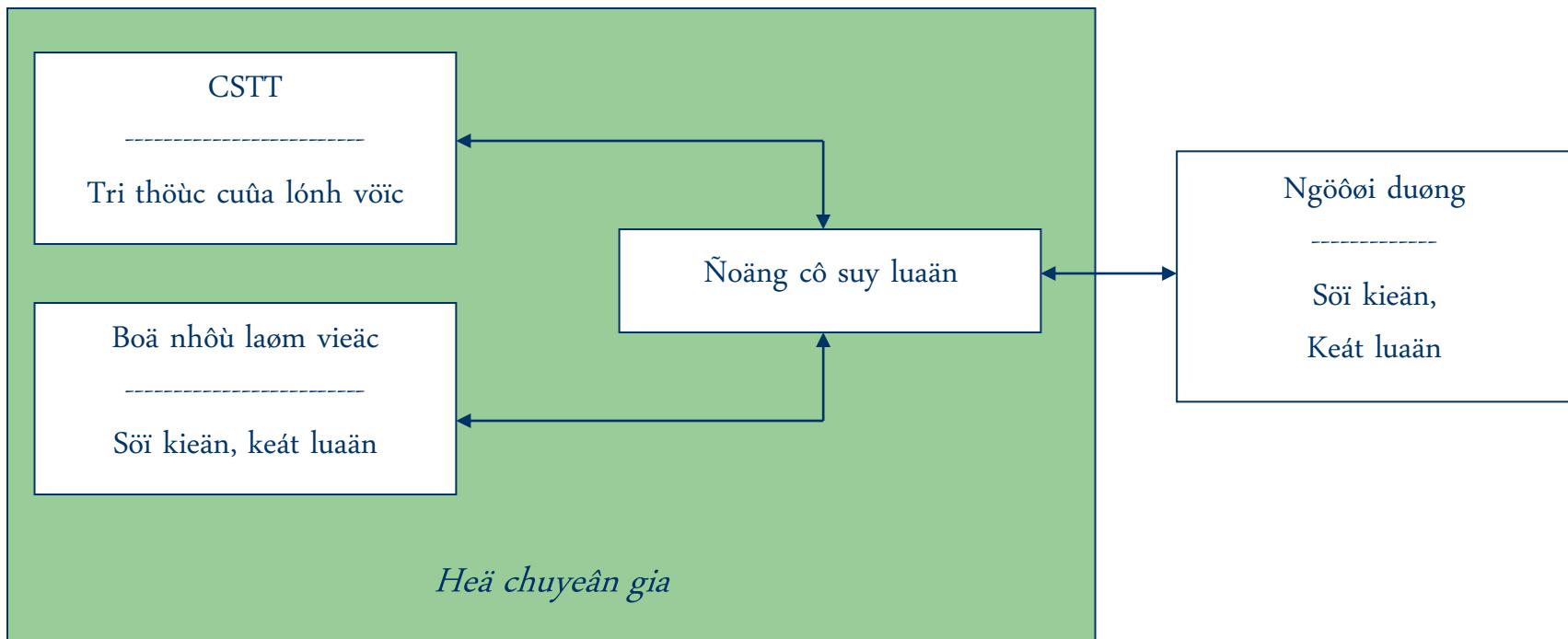
- Cấu trúc của ES:

ES mô phỏng khả năng giải quyết vấn đề của chuyên gia con người. Do vậy, chúng ta cần xem xét cách thức giải quyết của chuyên gia con người, để từ đó mô phỏng.



Cấu trúc của ES.

- ♣ Cách giải quyết vấn đề ở ES:



Cấu trúc của ES.

- CSTT:

Là một bộ phận của ES nhằm chứa tri thức của lĩnh vực.

♣ ES chứa tri thức của chuyên gia con người trong một bộ phận được gọi là CSTT. Để có tri thức này, người kỹ sư tri thức phải thu thập tri thức từ chuyên gia con người rồi mã hoá vào CSTT – cách thức mã hoá sẽ được đề cập trong phần kỹ thuật biểu diễn tri thức.

♣ Một trong các cách tiêu biểu để biểu diễn là dùng luật, như sau:

RULE 1:

IF “Xe car không thể khởi động được”

THEN “Vấn đề trong hệ thống điện”

RULE 2:

IF “Vấn đề trong hệ thống điện”

AND “Điện thế AC-quy nhỏ hơn 10Volt”

THEN lỗi tại bộ AC-quy”

Cấu trúc của ES.

- Bộ nhớ làm việc:

Là bộ phận của ES dùng để chứa các sự kiện của vấn đề. Các sự kiện này có thể do người dùng nhập vào lúc đầu hay do ES sinh ra trong quá trình làm việc.

- Với ES dùng cho nhiều người dùng cùng thì bộ nhớ làm việc thường phân nhóm theo phiên làm việc (session) của người dùng. Đó là trường hợp một ES chung cho nhiều người dùng từ xa.
- Nhiều ES cũng tận dụng các thông tin được chứa trong các nguồn ngoài như: CSDL, bảng tính, sensor,...ES sẽ tải thông tin này vào bộ nhớ làm việc đầu mỗi session hay khi cần thiết.

Cấu trúc của ES.

- **Động cơ suy luận:**

Là bộ xử lý trong hệ chuyên gia, là nhiệm vụ *so trùng các sự kiện được chứa trong bộ nhớ làm việc với tri thức được chứa trong CSTT nhằm dẫn ra kết luận cho vấn đề*.

♣ Tiêu biểu, nếu CSTT có chứa luật, ES sẽ tìm ra luật mà các tiên đề của luật so trùng với các sự kiện được chứa trong bộ nhớ làm việc, lúc đó ES sẽ thêm các kết luận của luật đó vào bộ nhớ làm việc, rồi tiếp tục tìm ra sự so trùng khác – giống như nguyên lý hoạt động của hệ luật sinh.

♣ Ví dụ: Giả sử CSTT chỉ với hai luật nêu trên

Bước 1:

ES:

Có phai xe car không khởi động được ?

Người dùng:

Đúng.

Cấu trúc của ES.

Chú thích: Người dùng trả lời “Đúng”, nên ES thêm vào bộ nhớ làm việc sự kiện để mô tả:

“Xe car không thể khởi động được”

Động cơ suy diễn của ES làm nhiệm vụ so trùng, nhận thấy RULE 1 có thể so trùng được, nên nó thêm vào bộ nhớ làm việc phân kết luận của RULE 1, đó là:

“Vấn đề trong hệ thống điện”

Bước 2:

ES: Có phai điện Ac-quy dưới 10 Volt?

Người dùng: Đúng.

Chú thích: Người dùng trả lời “Đúng”, nên ES thêm vào bộ nhớ làm việc sự kiện để mô tả:

“Điện thế Ac-quy nhỏ hơn 10Volt”

Động cơ suy diễn của ES làm nhiệm vụ so trùng, nhận thấy RULE 2 có thể so trùng được, nên nó thêm vào bộ nhớ làm việc phân kết luận của RULE 2, đó là:

“lỗi tại bộ Ac-quy” – phiên làm việc cũng kết thúc vì CSTT chỉ gồm hai luật trên.

Cấu trúc của ES.

- Tiện ích giải thích.

Một trong các điểm nổi bật của ES là khả năng giải thích về suy luận của nó. ES còn có một khối cơ bản nữa trong cấu trúc của nó đó là: *khối tiện ích giải thích*. Với khối này ES có thể cung cấp cho người dùng các khả năng giải thích:

- Tại sao ES lại hỏi câu hỏi nào đó. (WHY)
- Bằng cách nào ES có thể suy ra kết luận nào đó. (HOW)

Khối tiện ích giải thích thuận tiện cho cả người phát triển ES và người dùng. Người phát triển có thể nhờ đó khám phá các lỗi trong tri thức của ES. Người thì có thể yên tâm hơn khi nhận một kết luận nào đó, không cần thiết phải quan tâm với cấu trúc tri thức của ES.

♣ Giải thích bằng cách nào (HOW)

Ngoài chức năng cung cấp cho người dùng kết quả suy luận cuối cùng, ES càng có thể cung cấp nó đạt đến kết luận bằng cách nào.

Cấu trúc của ES.

Khả năng giải thích bằng cách nào rất quan trọng với ES, nó làm cho ES khác với các chương trình truyền thống. Các chương trình truyền thống làm việc trên các bài toán được định nghĩa tốt, kết quả của chương trình đã được kiểm nghiệm trước khi chuyển giao. ES làm việc trên các bài toán thiếu nhiều thông tin, thậm chí tri thức còn có thể cập nhật khi ES đã được triển khai – chính vì vậy mà kết luận của nó phải được giải thích bằng cách nào để cho kết quả cuối cùng có giá trị hơn.

Ví dụ: trong phiên làm việc của ES nói trên.

ES: lối tại bộ Ac-quy”.

Người dùng: HOW

ES: “Bởi vì, khi xe của bạn không khởi động được, tôi cho rằng đã có vấn đề trong hệ thống điện. Mỗi lần, tôi thấy điện thế của ac-quy dưới 10 volt, tôi biết rằng ac-quy đã hư.”

- Khả năng giải thích HOW của ES có thể thực hiện được bằng cách cho phép nó theo vết các luật mà nó đã sử dụng vào suy luận.

Cấu trúc của ES.

♣ Giải thích tại sao (WHY)

khả năng của ES cung cấp cách giải thích: “tạo sao nó lại hỏi một câu hỏi nào đó.” Với khả năng này, người dùng cũng có thể theo vết được luồng suy luận của ES và yên tâm hơn với kết quả.

Ví dụ: trong phiên làm việc với ES nói trên.

ES: Có phải xe car không khởi động được ?

Người dùng: WHY

ES: “Nếu tôi biết rằng xe car không khởi động được, thì tôi thường cho rằng có vấn đề trong hệ thống điện.”

Khi chúng ta hỏi WHY, ES thường đáp trả bằng cách mô tả cái gì mà nó có thể kết luận từ câu trả lời. Hầu hết các ES thường đáp trả bằng cách hiện luật mà nó đang quan tâm.

Cấu trúc của ES.

- Giao diện người dùng:

Giao diện cũng là một thành phần quan trọng của ES, nó giúp cho ES có thể đặt câu hỏi với người dùng và nhận về câu trả lời chính xác. Yêu cầu cao nhất cho giao diện là có khả năng cung cấp cách hỏi đáp tương tự như giữa người - với - người.

Khi hiện thực hệ thống, vì những hạn chế của kỹ thuật hiện tại nên người thiết kế phải nghĩ đến những hình thức giao tiếp sao cho tiện lợi, tuy chưa thật giống với “người- người”. Cụ thể, có thể dùng giao diện đồ họa , dạng menu chọn, phát âm câu hỏi, ... *cũng cần phải tính đến khả năng dùng web như môi trường tương tác.*

Các đặc trưng cơ bản của ES.

- Phân tách tri thức và điều khiển.

Đã đề cập trong hệ luật sinh. Đây cũng là đặc điểm phân biệt giữa chương trình truyền thống và ES.

Hãy so sánh khả năng thay đổi tri thức của vấn đề giữa hai loại chương trình nói trên.

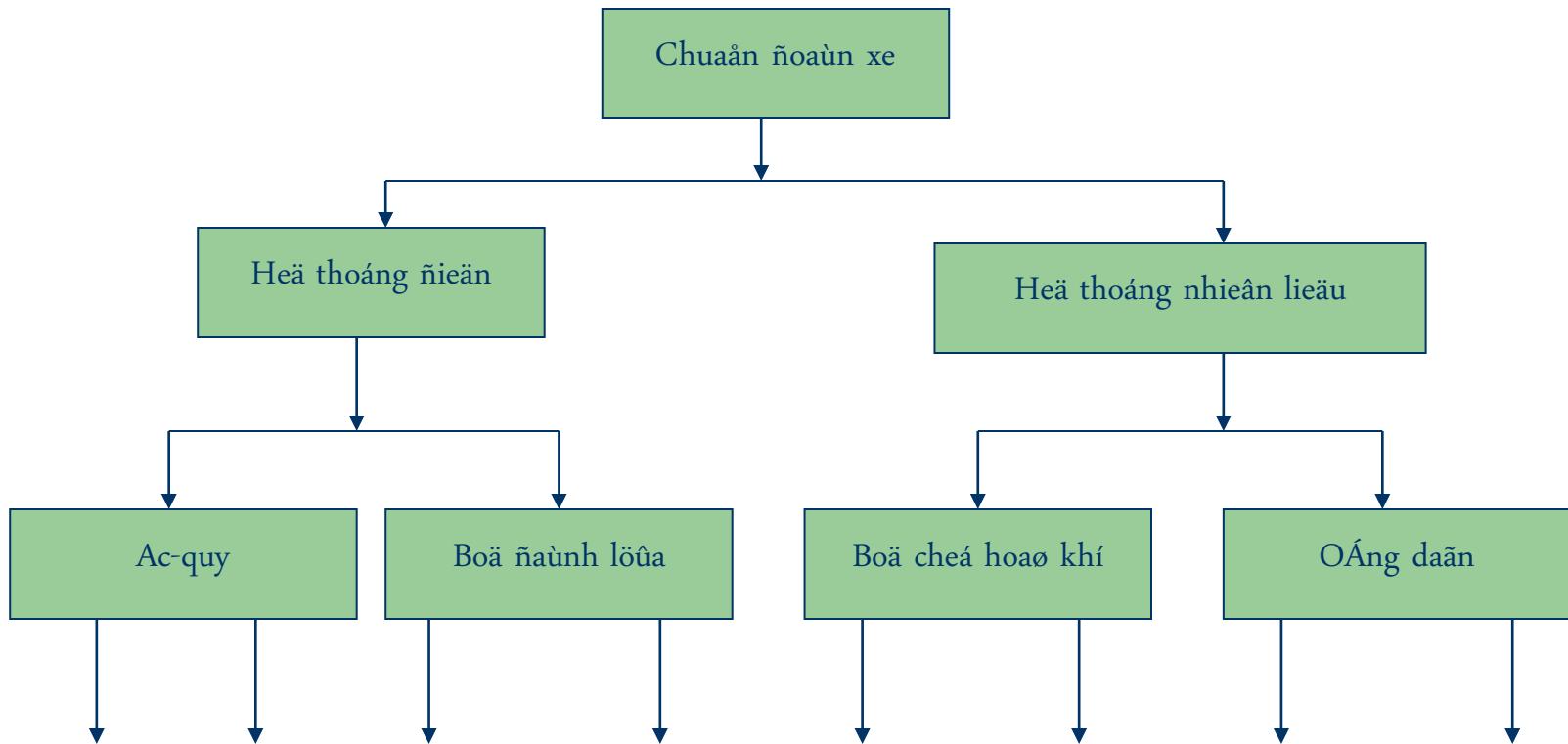
- Sỡ hữu tri thức chuyên gia.

ES có chứa tri thức của lĩnh vực trong CSTT. Nhờ có tri thức mà nó có giá trị. Đặc biệt là tri thức này có thể được nhân ra thành nhiều bản, có thể cập nhật trong khi hệ thống đã được triển khai.

- Tính chuyên gia trong lĩnh vực hẹp.

Cũng giống như chuyên gia con người, ES được phát triển nhằm vào một lĩnh vực hẹp. Điều này cũng dễ hiểu, vì lý do: trong lĩnh vực hẹp đó số lượng tri thức cũng nhỏ hơn, và giúp cho người thiết kế dễ dàng quản lý hơn, dễ dàng thử nghiệm chiến lược điều khiển trong động cơ suy diễn. Người thiết thường chia tri thức theo từng mảng như hình sau để quản lý nó.

Các đặc trưng cơ bản của ES.



Các đặc trưng cơ bản của ES.

- Suy luận trên ký hiệu.

Chúng ta có thể dùng ký hiệu để thể hiện tri thức cho ES. Chính vì vậy mà có thể tận dụng được các giải thuật trên ký hiệu để tri thức thức, như các giải thuật đã đề cập trong chương 2 – phần phép toán vị từ.

- Suy luận có heuristic

Chuyên gia con người có thể từ kinh nghiệm của mình để dẫn ra cách giải quyết vấn đề hiệu quả hơn, ví dụ:

Khi chuẩn đoán xe, họ có thể giả thiết cách làm:

- Luôn luôn kiểm tra luật về hệ thống điện trước các luật khác.

Hay một bác sĩ chuyên khoa có thể giả thiết:

- Nếu nghi ngờ bị ung thư, thì kiểm tra dòng họ trước.

Để có thể hiện thực trong ES, người thiết kế cần phải có cách đánh giá thứ tự ưu tiên của các luật, để từ một ngữ cảnh nào đó có thể chọn một luật có lý nhất để bắt đầu.

Các đặc trưng cơ bản của ES.

- Cho phép suy luận không chính xác.

ES có một khả năng rất mạnh đó là: nó có thể làm việc với các vấn đề đang thiếu thông tin, hay có nhưng hồn tạp, không rõ ràng. Cũng giống như trường hợp: một ekip bác sĩ đang phải cứu một bệnh nhân hấp hối, lúc đó họ không còn kịp thời gian để làm tất cả các xét nghiệm cần thiết. Khi thiếu thông tin như vậy họ đành tiến hành những cách có lý nhất theo họ. Chúng ta cũng có thể hiện thực ES có tính chất đó bằng cách đưa vào những luật tương ứng với tình huống thiếu thông tin để động cơ suy diễn vận dụng.

- Bị giới hạn vào vấn đề giải quyết.

Không phải mọi vấn đề đều có thể giải quyết bởi ES. Cụ thể, nếu lĩnh vực chúng muốn xây dựng ES hiện tại chưa có, chưa cần một chuyên gia con người thì việc xây dựng ES khó mà thành công.

- Giải quyết các vấn đề có độ phức tạp vừa phải.

Nếu vấn đề quá khó, yêu cầu chuyên gia con người đến vài giờ, cần thiết nghĩ đến khả năng chia thành nhiều bài toán con tương ứng mỗi ES.

Các đặc trưng cơ bản của ES.

- Có khả năng bị lỗi.

Giống như chuyên gia con người ES có khả năng bị lỗi. Chính vì vậy, cần thiết đưa vào khả năng phục hồi lại lỗi cho ES – ES có khả năng lưu vết quá trình suy luận, nếu nó đưa ra một kết luận mà người dùng kiểm nghiệm với thực tế có sai và báo cho ES, lúc đó nó phải có khả năng ghi nhận và theo đuổi một hướng suy luận khác.

đặc điểm này không xuất hiện trong các chương trình truyền thống, nhưng dùng vội kết luận loại chương trình đó tốt hơn. Mỗi loại có những đặc điểm riêng như bảng so sánh sau:

CT truyền thông

Xử lý số

Giải thuật

Tích hợp thông tin+ điều khiển

Khó thay đổi

Thông tin chính xác

Giao diện lệnh điều khiển

Kết quả cuối cùng

Tối ưu

ES

Xử lý ký hiệu.

Heuristic

Tách bạch thông tin+ điều khiển
dễ thay đổi.

Thông tin không chắc chắn.

Hội thoại + giải thích.

đề nghị + giải thích
Có thể chấp nhận.

Công nghệ tri thức.



- Quá trình gồm các giai đoạn như hình bên.
- Một số định nghĩa:
 - ♣ *Công nghệ tri thức: Là quá trình xây dựng ES.*
 - ♣ *Thu thập tri thức: Là quá trình thu thập, tổ chức và nghiên cứu tri thức.*

Các nhân tố trong một dự án ES

- **Các nhân tố chính:**
 - ✉ Chuyên gia lĩnh vực.
 - ✉ Kỹ sư tri thức
 - ✉ Người dùng sản phẩm
- **Các yêu cầu cho mỗi nhân tố:**
 - Chuyên gia lĩnh vực:**
 - ✉ Có tri thức chuyên gia
 - ✉ Có kỹ năng giải quyết vấn đề hiệu quả
 - ✉ Có thể chuyển giao tri thức
 - ✉ Không chống đối (thân thiện).
 - Kỹ sư tri thức:**
 - ✉ Có kỹ năng về công nghệ tri thức
 - ✉ Có kỹ năng giao tiếp tốt.
 - ✉ Có thể làm cho vấn đề được giải quyết bởi phần mềm.
 - ✉ Có kỹ năng lập trình hệ chuyên gia.
 - Người dùng sản phẩm:**
 - ✉ Có thể trợ giúp thiết kế giao diện cho ES.
 - ✉ Có thể trợ giúp việc thu thập tri thức.
 - ✉ Có thể trợ giúp trong quá trình phát triển ES.

Các kỹ thuật suy luận

- Suy luận: là quá trình làm việc với tri thức, sự kiện, chiến lược giải toán để dẫn ra kết luận.
- Bạn suy luận như thế nào?

Các hình thức cơ bản:

- ♣ ♣ Suy luận diễn dịch.
- ♣ ♣ Suy luận quy nạp.
- ♣ ♣ Suy luận tương tự.
- ♣ ♣ Suy luận khả sai.
- ♣ ♣ Suy luận common-sense.
- ♣ Suy luận đơn điệu
- ♣ Suy luận không đơn điệu.

Các kỹ thuật cơ bản:

- ♣ Suy luận tiến (forward-chaining)
- ♣ Suy luận lùi (backward-chaining)

Ưu – nhược điểm của mĩ kỹ thuật

Suy luận tiến – forward chaining

- **Ưu điểm:**

- ♣ Làm việc tốt với bài toán có bản chất: gồm thông tin và sau đó tìm xem có thể suy ra cái gì từ thông tin đó.
- ♣ Có thể dẫn ra rất nhiều thông tin chỉ từ một ít sự kiện ban đầu.
- ♣ Thích hợp cho một số vấn đề như: *hoạch định, giám sát, điều khiển, diễn dịch.*

- **Nhược điểm:**

- ♣ Không có cách để nhận thấy tính quan trọng của từng sự kiện. Hỏi nhiều câu hỏi thừa, vì đôi lúc chỉ cần một vài sự kiện là cho ra kết luận.
- ♣ Có thể hỏi những câu hỏi không liên quan gì nhau – chuỗi câu hỏi không ăn nhập nhau.

VD:

- Bạn có thân nhiệt cao ?
- Bạn đến VN đã lâu rồi ?
- ...

Ưu – nhược điểm của mỗi kỹ thuật

Suy luận lùi – backward chaining

- **Ưu điểm:**

- ♣ Làm việc tốt với bài toán có bản chất: thành lập giả thiết , sau đó tìm xem có thể chứng minh được không.
- ♣ Hướng đến một goal nào, nên hỏi những câu hỏi có liên quan nhau.
- ♣ Chỉ khảo sát CSTT trên nhánh vấn đề đang quan tâm.
- ♣ Tốt cho các vấn đề: *chuẩn đoán, kê toa, gỡ rối*.

- **Nhược điểm:**

- ♣ Luôn hướng theo dòng suy luận định trước thậm chí có thể dừng và rẽ sang một goal khác.
 - Giải quyết: dùng meta-rule để khắc phục.
- Meta-rule: dùng để hướng không gian tri thức được khảo sát sang một vùng khác.

Khảo sát ES: MYCIN

● Giới thiệu:

- ♣ Là ES về lĩnh vực chuẩn đoán bệnh nhiễm trùng nhau.
- ♣ Được phát triển vào khoảng năm 1970, tại đại học Standford
- ♣ Là thước đo cho nhiều ES dựa trên luật sau này.
- ♣ Được viết trên ngôn ngữ INTERLISP, một nhánh của LISP.
- ♣ Người ta từ MYCIN, tước bỏ phần CSTT của nó để được một hệ thống EMYCIN (empty MYCIN), dùng để phát triển các ES khác.

● Các đặc điểm chính:

- ♣ Sử dụng kỹ thuật suy luận lùi.
- ♣ Có khả năng phân tích tri thức và điều khiển.
- ♣ Có tích hợp Meta-rule.
- ♣ Có thể dùng khi thiếu thông tin hay thông tin không chính xác.
- ♣ Có khả năng nhớ được các thông tin trong một phiên làm việc với người bệnh cụ thể.
- ♣ Dễ sử dụng – giao tiếp với bác sĩ qua tiếng anh.
- ♣ Cung cấp các chức năng giải thích: WHY, HOW.

Khảo sát ES: MYCIN

- Một phiên làm việc cụ thể với MYCIN

(“Expert System”. Author: John Durkin)
--- đã được Việt hóa.

? Một vài thông tin về bệnh nhân:

1. MYCIN: Họ tên ?

USER: Jack Durkin.

2. MYCIN: Tuổi ?

USER: 24

3. MYCIN: Giới tính ?

USER: Nam

4. MYCIN: Chủng tộc?

USER: CAUCASIAN

? MYCIN cố gắng xác định bản chất lây nhiễm.

5. MYCIN: Có bất kỳ phép cấy vi khuẩn cho Jack Durkin liên qua đến bệnh hiện tại?

USER: Có

6. MYCIN: Mẫu vật lấy từ đâu?

USER: CSF

7. MYCIN: Ngày, giờ thực hiện cấy?

USER: 7- July-7:00

....

MYCIN: Gần đây Jack Durkin có các triệu chứng như: choáng váng, hôn mê không?

USER: Có.

....

Khảo sát ES: XCON

- Giới thiệu:
 - ♣ Là ES về lĩnh vực hướng dẫn cấu hình các hệ thống máy tính của DEC trước khi xuất xưởng.
 - ♣ Được phát triển vào khoảng năm 1980, tại đại học CMU
 - ♣ Được công bố giảm thời gian cấu hình cho mỗi hệ thống xuống còn 2 phút (so với 25 phút bằng tay.). Tiết kiệm vào khoảng 25 triệu \$ cho mỗi năm.

(Theo “Expert System” – John Durkin)

Hệ chuyên gia dựa trên luật

- **Định nghĩa:**

Là một chương trình máy tính, xử lý các thông tin cụ thể của bài toán được chứa trong bộ nhớ làm việc và tập các luật được chứa trong CSTT, sử dụng động cơ suy luận để suy ra thông tin mới.

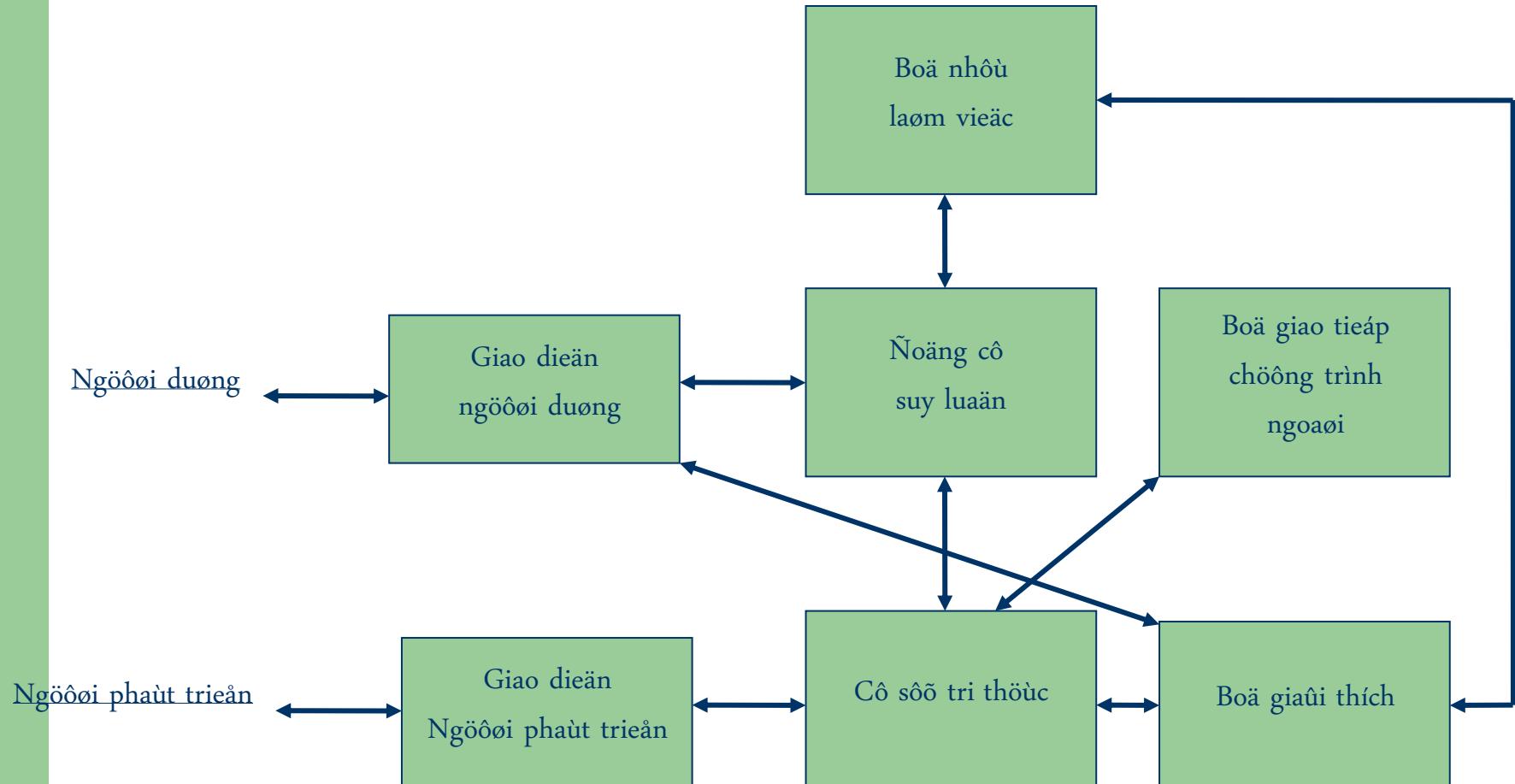
ES dựa trên luật: có nền tảng xây dựng lá hệ luật sinh – chương trước.

ES dựa trên luật cũng có những đặc trưng cơ bản như đã nêu trong phần trước cho các ES tổng quát, một vài đặc điểm:

- ♣ Có CSTT chứa các luật.
- ♣ Có bộ nhớ làm việc tạm thời.
- ♣ Có động cơ suy luận.
- ♣ Có một giao diện để giao tiếp với người dùng, người phát triển.
- ♣ Có tiện ích giải thích.
- ♣ Có khả năng giao tiếp với chương trình ngoài như: các DBMS, xử lý bảng tính,...

Hệ chuyên gia dựa trên luật

- Kiến trúc: (như hình sau)
- Nguyên lý hoạt động tương tự hệ luật sinh đã giới thiệu.



Hệ chuyên gia dựa trên luật

• Ưu điểm

- ♣ Biểu diễn tri thức tự nhiên: IF... THEN.
- ♣ Phân tách tri thức – điều khiển.
- ♣ Tri thức là tập các luật có tính độc lập cao -> dễ thay đổi, chỉnh sửa.
- ♣ Dễ mở rộng.
- ♣ Tận dụng được tri thức heuristic.
- ♣ Có thể dùng biến trong luật, tri xuất chương trình ngoài.

• Nhược điểm

- ♣ Các fact muốn đồng nhất nhau, phải khớp nhau hoàn toàn → Các facts cùng một ý nghĩa phải giống nhau về cú pháp, ngôn ngữ tự nhiên không như vậy.
- ♣ khó tìm mối qua hệ giữa các luật trong một chuỗi suy luận, vì chúng có thể nằm rải rác trong CSTT.
- ♣ Có thể hoạt động chậm.
- ♣ Làm cho nhà phát triển phải hình chung mọi cái ở dạng luật -> không phải bài toán nào cũng có thể làm được như thế này.

Chương 7: BIỂU DIỄN TRI THỨC

❖ Biểu diễn tri thức trong AI: vai trò và ứng dụng

❖ Các kỹ thuật biểu diễn tri thức:

- Semantic network

- Lưu đồ phụ thuộc khái niệm

- Frame

- Script

Phạm Minh Tuấn

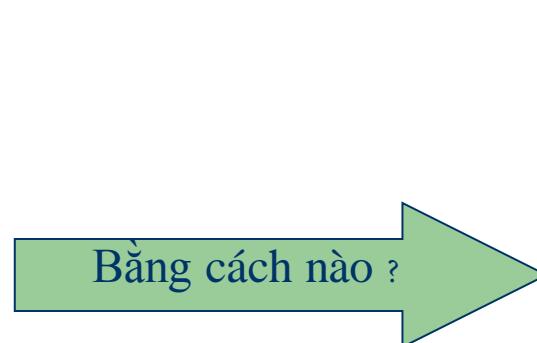
Các lược đồ biểu diễn tri thức.

- Định nghĩa:

Biểu diễn tri thức là phương pháp để mã hoá tri thức, nhằm thành lập cơ sở tri thức cho các hệ thống dựa trên tri thức (knowledge-based system).



Gồm: đối tượng và các quan hệ giữa chúng trong lĩnh vực.



Bằng cách: dùng các lược đồ biểu diễn (scheme).
→ Chọn dùng lược đồ cho loại tri thức là vấn đề quan trọng.



Gồm: Bảng ánh xạ giữa: Đối tượng thực → đối tượng tính toán.
Quan hệ thực → quan hệ tính toán.

Các lược đồ biểu diễn tri thức.

- Chú ý:
 - Cần phân biệt: Lược đồ biểu diễn (scheme) và môi trường hiện thực (medium), tương tự như việc phân biệt: cấu trúc dữ liệu (CTDL) và ngôn ngữ lập trình. Với một loại CTDL, ví dụ như: Bản ghi (record), chúng ta có hiện thực trong nhiều ngôn ngữ như: Pascal, C,...Tương tự, với một loại lược đồ nào đó chúng ta có thể chọn một trong các NNLT để hiện thực nó.
- Các loại lược đồ biểu diễn:
 - ♣ **Lược đồ logic.**
 - Dùng các biểu thức trong logic hình thức ,như phép toán vị từ, để biểu diễn tri thức.
 - Các luật suy diễn áp dụng cho loại lược đồ này rất rõ ràng, đã khảo sát trong chương 2 (như: MP, MT,...).
 - Ngôn ngữ lập trình hiện thực tốt nhất cho loại lược đồ này là: PROLOG.
 - ♣ **Lược đồ thủ tục:**
 - Biểu diễn tri thức như tập các chỉ thị lệnh để giải quyết vấn đề.

Các lược đồ biểu diễn tri thức.

♣ Lược đồ thủ tục:

Ngược lại với các lược đồ dạng khai báo, như logic và mạng, các chỉ thị lệnh trong lược đồ thủ tục chỉ ra bằng cách nào giải quyết vấn đề.

Các luật trong CSTT của ES dựa trên luật là ví dụ về thủ tục giải quyết vấn đề.

Hệ luật sinh là ví dụ điển hình của loại lược đồ này.

♣ Lược đồ mạng.

Biểu diễn tri thức như là đồ thị; các đỉnh như là các đối tượng hoặc khái niệm, các cung như là quan hệ giữa chúng.

Các ví dụ về loại lược đồ này gồm: mạng ngũ nghĩa, phụ thuộc khái niệm, đồ thị khái niệm → được khảo sát sau đây của chương này.

♣ Lược đồ cấu trúc:

Là một mở rộng của lược đồ mạng; bằng cách cho phép các node có thể là một CTDL phức tạp gồm các khe(slot) có tên và trị hay một thủ tục. Chính vì vậy nó tích hợp cả dạng khai báo và thủ tục.

Kịch bản(script), khung (frame), đối tượng (object) là ví dụ của lược đồ này → khảo sát sau.

Các chú ý về lược đồ.

- Khi xây dựng các lược đồ cần chú ý những vấn đề sau:
 - ♣ Các đối tượng và các quan hệ có thể biểu diễn cho cái gì trong lĩnh vực?

Ví dụ: để biểu diễn cho ý “Nam cao 1mét 70”, chúng ta có thể dùng: **chieucao(nam,170)**. Vậy thì để diễn tả “An cao hơn Nam” chúng ta làm như thế nào, vì chiều cao của An lúc này không là một trị cụ thể nữa!
 - ♣ Bằng cách nào phân biệt giữa “nội hàm” và “ngoại diện” của một khái niệm.
- ♣ Bằng cách nào thể hiện được meta-knowledge?
- ♣ Bằng cách nào thể hiện tính phân cấp của tri thức.

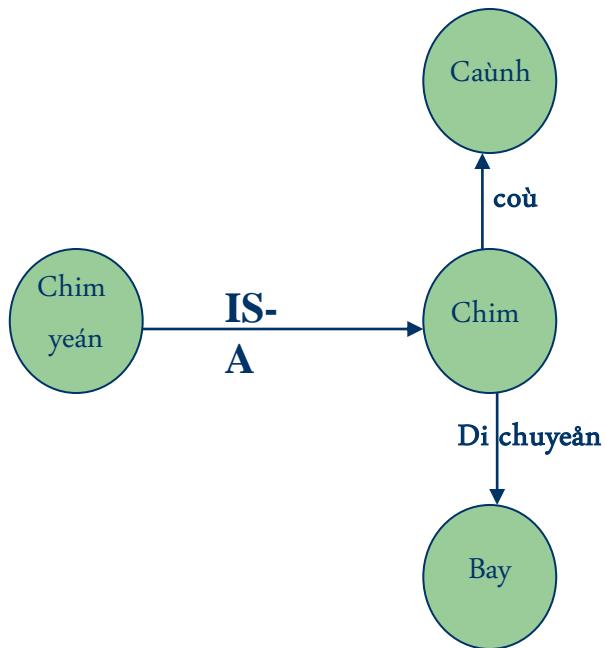
Lúc biểu diễn tính phân cấp thì các hình thức : ké thừa, ngoại lệ, trị mặc định, ngoại lệ, đa thừa kế phải đặc tả như thế nào
- ♣ Khi mô tả đối tượng, bằng cách nào có thể tích hợp một tri thức thủ tục vào bản thân mô tả, khi nào thủ tục được thực hiện,..

Mạng ngũ nghĩa

- Định nghĩa:

Là một lược đồ biểu diễn kiểu mạng, dùng đồ thị để biểu diễn tri thức. Các đỉnh biểu diễn đối tượng; các cung biểu diễn quan hệ giữa chúng.

- Ví dụ:



Xem mạng bên:

- Có hai đỉnh biểu diễn đối tượng, và hai đỉnh còn lại biểu diễn thuộc tính.
- đỉnh có nhãn: “Chim” nối với hai đỉnh thuộc tính có nhãn: “Cánh”, “Bay” nên có thể biểu diễn: “Một con chim thì có cánh và có hình thức di chuyển là bay”.
- Đỉnh có nhãn “Chim yến” nối với đỉnh “Chim” thông qua cung đặc biệt “IS-A” nói lên: “Chim yến là một loài chim”. Vì vậy chim yến có thể sở hữu các thuộc tính: có cánh, bay như một con chim thông thường.

Mạng ngũ nghĩa

- **Mở rộng mạng ngũ nghĩa:**

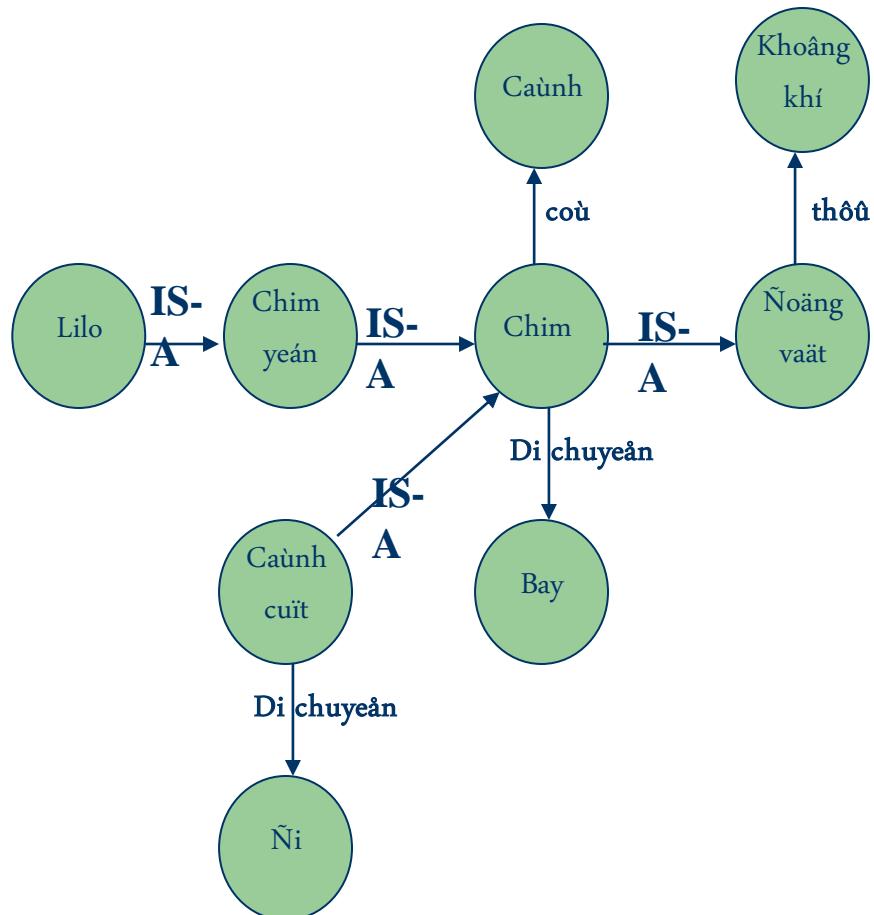
Để mở rộng mạng thật đơn giản; chúng ta chỉ việc thêm các đỉnh và các cung quan hệ với các đỉnh có sẵn. Các đỉnh được thêm vào mạng hoặc là biểu diễn đối tượng hoặc là biểu diễn thuộc tính như ví dụ trước. Xét ví dụ sau đây minh họa việc mở rộng mạng đã có.

- **Tính thừa kế:**

Là đặc điểm nổi bật của lược đồ mạng ngũ nghĩa. Mạng ngũ nghĩa định ra cung quan hệ đặc biệt “IS-A” để chỉ ra sự thừa kế. Ví dụ, nhờ tính thừa kế mà từ mạng bên chúng ta có thể suy ra: “Lilo là một động vật có thể bay và hít thở không khí.”

- **Tính ngoại lệ:**

Định nghĩa một cung quan hệ mới đến một đỉnh có trị khác.

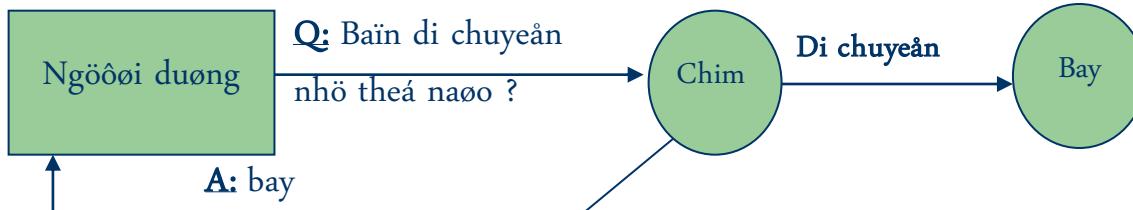


Mạng ngũ nghĩa

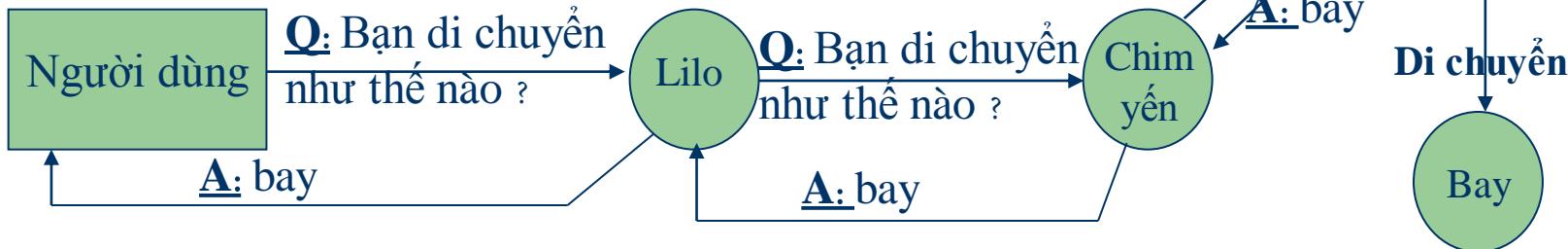
- **Phép toán trên mạng ngũ nghĩa:**

Giả sử chúng ta đã mã hoá mạng ở hình trước vào máy tính. Để dùng mạng, có thể đơn giản là chúng ta câu hỏi với một đỉnh nào đó. Ví dụ, với đỉnh “Chim” chúng ta đặt câu hỏi: “Bạn di chuyễn như thế nào?”. Để trả lời câu hỏi chúng ta có thể hiện thực cách trả lời sau cho đỉnh: tìm kiếm cung quan hệ có nhãn “di chuyễn” bắt đầu từ nó, như case 1,2 ở bên.

Case 1:



Case 2:



Lưu đồ về quan hệ phụ thuộc khái niệm.

- Trong quá trình nghiên cứu về cách hiểu ngôn ngữ tự nhiên, Schank và Rieger đã cố gắng thiết lập một tập các phần tử cơ bản để có thể biểu diễn cấu trúc ngữ nghĩa của các biểu thức ở ngôn ngữ tự nhiên theo một cách đồng nhất.

Lý thuyết về phụ thuộc khái niệm có đề ra 4 khái niệm cơ bản để từ đó ngữ nghĩa được xây dựng, chúng là:

♣ ACT (Action)

: các hành động.

: (các động từ trong câu)

♣ PP (Picture Producers) : các đối tượng.

: (các chủ từ, tên ngữ,...)

♣ AA (Action Adder)

: bổ nghĩa cho hành động.

: (trạng từ)

♣ PA (Picture Adder)

: bổ nghĩa cho đối tượng.

: (tính từ)

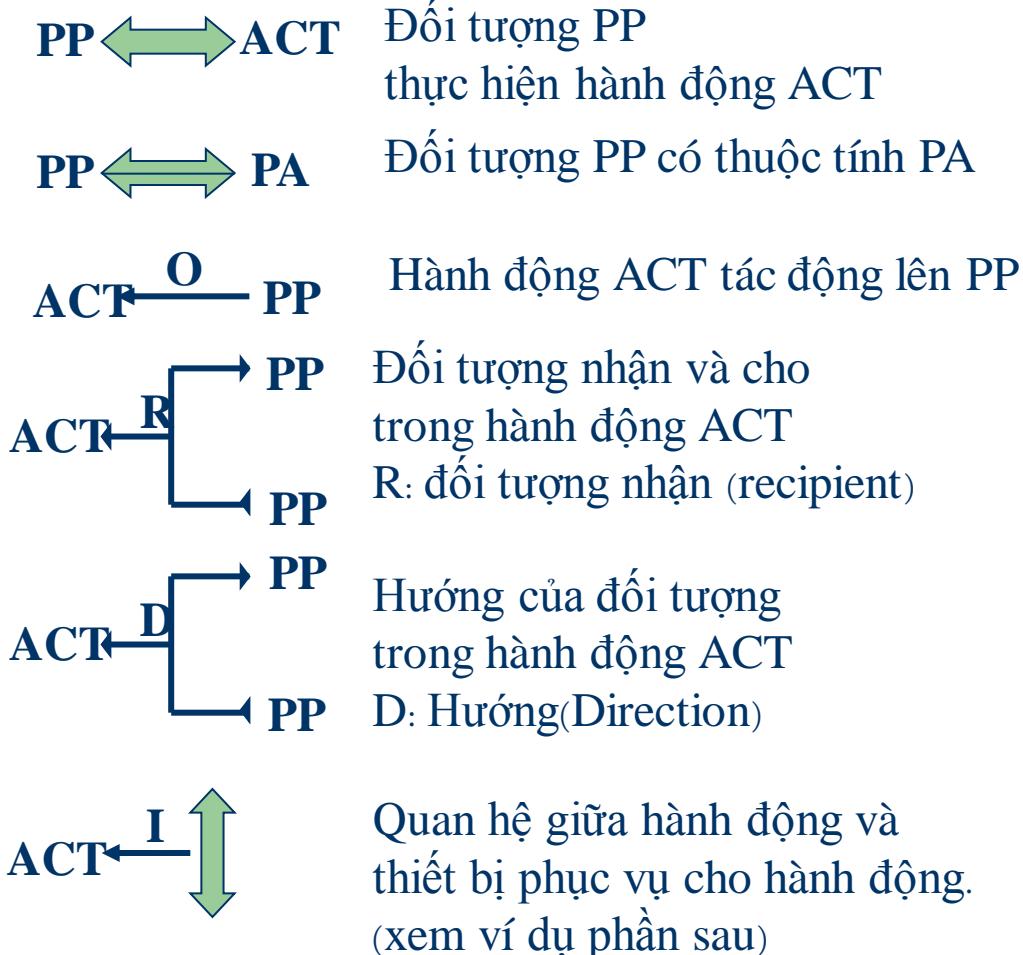
Lưu đồ về quan hệ phụ thuộc khái niệm.

- Tất cả các hành động được cho là có thể được mô tả bằng cách phân rã về một hoặc nhiều hành động như liệt kê sau đây:

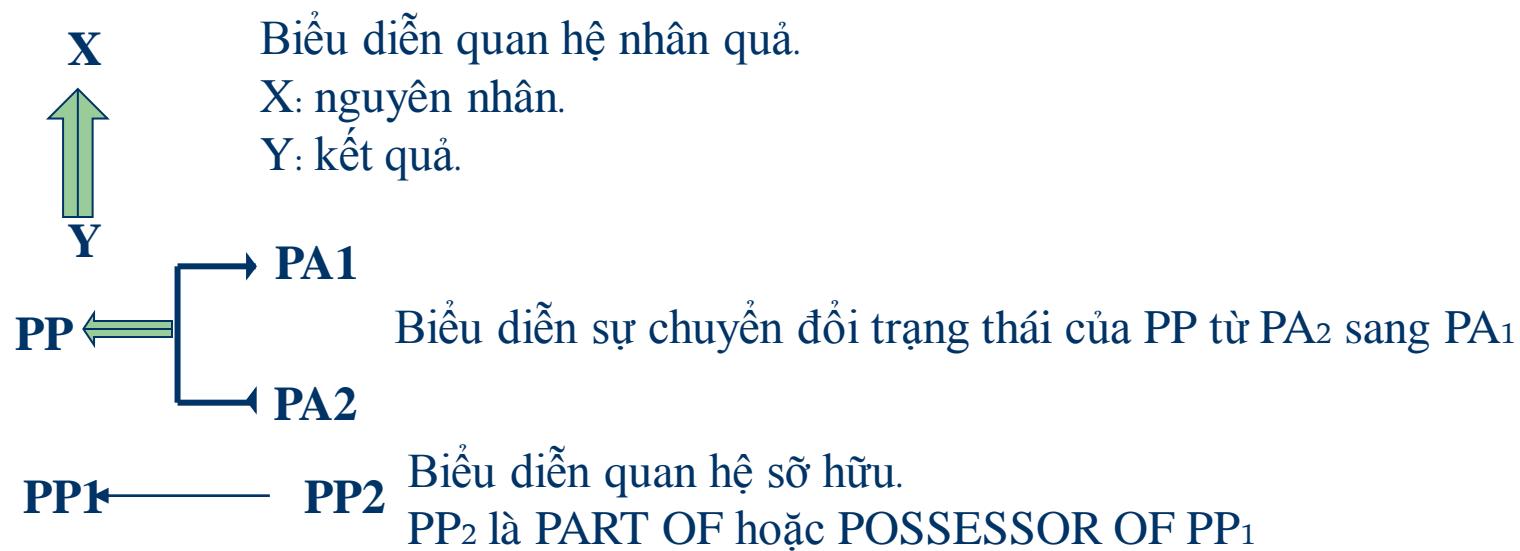
1. ATRANS : chuyển đổi một quan hệ – VD: động từ: cho, biểu,...
2. PTRANS : chuyển đổi vị trí vật lý – VD: đi, chạy, di chuyển,..
3. PROPEL : tác động một lực vật lý lên đối tượng – VD: đẩy, chải,...
4. MOVE : di chuyển một phần thân thể bởi đối tượng – VD: đá..
5. GRASP : nắm lấy đối tượng khác. – VD: cầm, nắm, giữ,...
6. INGEST : ăn vào bụng một đối tượng bởi đt khác – VD: ăn, nuốt,..
7. EXPEL : tống ra từ thân thể của một đối tượng – VD: khóc,..
8. MTRANS : chuyển đổi thông tin tinh thần – VD: nói, tiết lộ,..
9. MBUILD : tạo ra một thông tin tinh thần mới – VD: quyết định, ...
10. CONC : nghĩ về một ý kiến – VD: suy nghĩ, hình dung,...
11. SPEAK : tạo ra âm thanh – VD: nói, phát biểu,...
12. ATTEND: tập trung giác quan – VD: lắng nghe, nhìn,...

Lưu đồ về quan hệ phụ thuộc khái niệm.

- Quan hệ phụ thuộc khái niệm bao gồm một tập các luật cú pháp cho khái niệm, hình thành nên văn phạm về quan hệ ngữ nghĩa. Các quan hệ này sẽ được dùng vào việc biểu diễn bên trong cho câu trong ngôn ngữ tự nhiên. Danh sách các phụ thuộc khái niệm được liệt kê như bên.



Lưu đồ về quan hệ phụ thuộc khái niệm.



♣ Từ các phụ thuộc khái niệm cơ bản nêu trên. Chúng ta có thể kết hợp để có thể biểu diễn các câu trong NNTN, như ví dụ sau:



Ý nghĩa: “Nam đã tác dụng một lực vào quả bóng”

Lưu đồ về quan hệ phụ thuộc khái niệm.

- Các phụ thuộc khái niệm trên cho phép chúng biểu diễn quan hệ giữa: chủ từ với động từ (như phụ thuộc đầu tiên), hay giữa chủ từ và thuộc tính của nó,... Lược đồ về quan hệ phụ thuộc khái niệm càng đưa ra cách thừa để biểu diễn thì, điều kiện,..., như bên phải.

p : quá khứ – ACT đã xảy ra trong quá khứ
VD:

Ý Nam ←→^p PROPEL ←→^O Cái bàn
nghĩa: Nam đã tác dụng một lực (đẩy) vào cái bàn.

f : tương lai.

t : chuyển tiếp.

ts : bắt đầu chuyển tiếp.

tf : kết thúc chuyển tiếp.

k : đang diễn ra.

? : nghi vấn.

/ : phủ định.

C : điều kiện.

Nil: hiện tại. (không ghi chú gì)

Lưu đồ về quan hệ phụ thuộc khái niệm.

- Một số ví dụ về việc kết hợp các phụ thuộc khái niệm để biểu diễn câu:

PP \longleftrightarrow ACT VD: Nam \xleftarrow{P} PROPEL \xleftarrow{O} Lan : Nam đã đánh Lan

Nam \xleftarrow{K} ATTEND \xleftarrow{O} Bài giảng : Nam đang tập trung vào bài giảng.

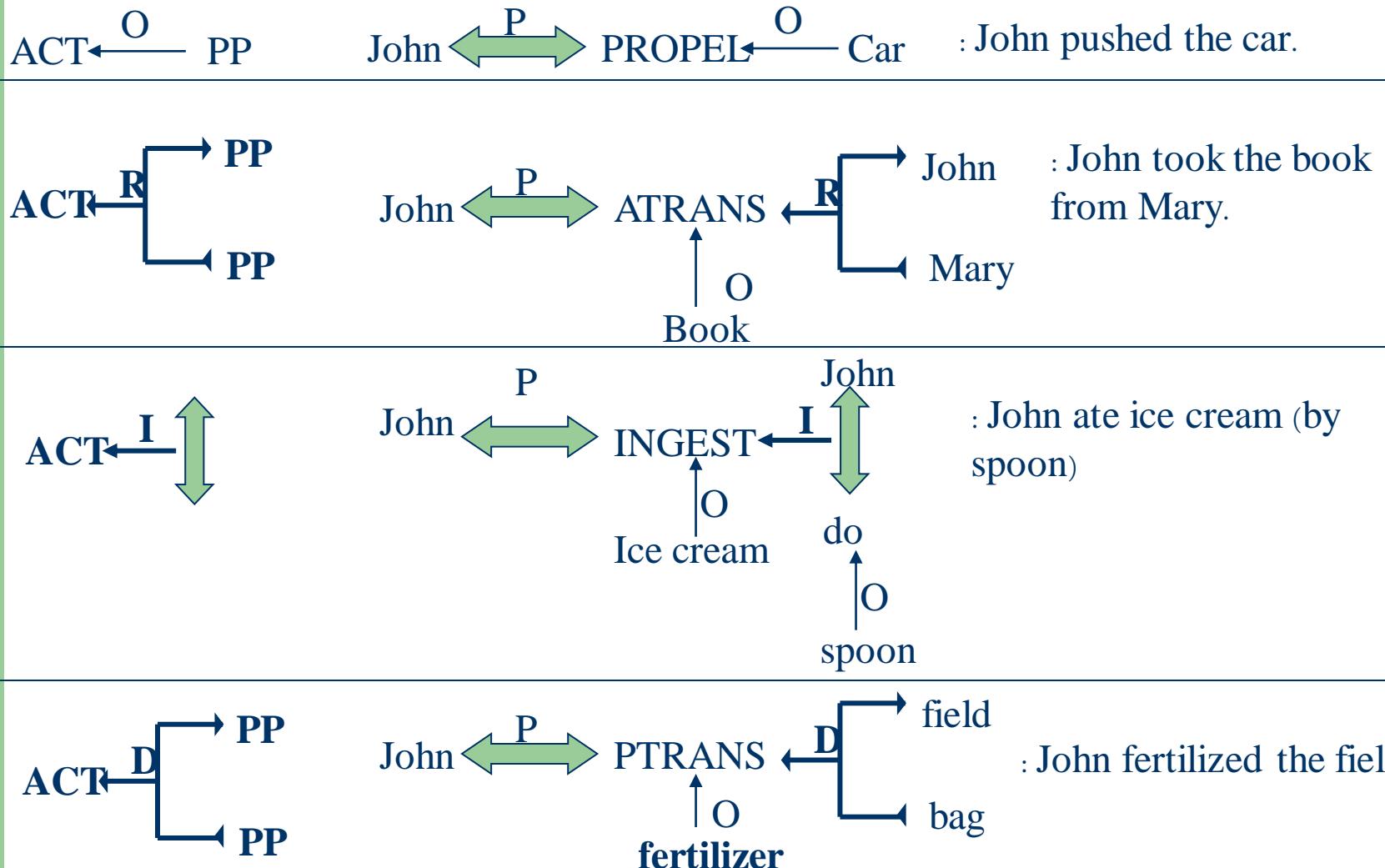
PP ↔ PA Nam ↔ Height (> average) : Nam is tall.

PP ↔ **PP** Nam ↔ doctor : Nam is a doctor.

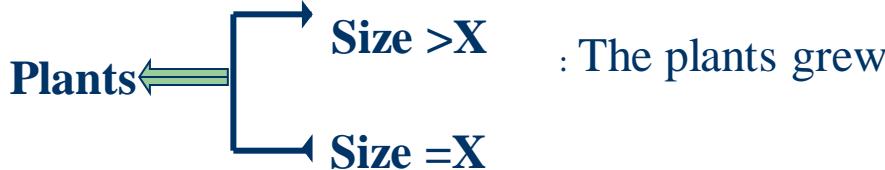
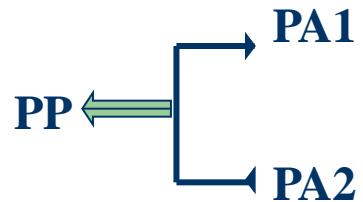
PP ← **PA** boy ← nice : A nice boy

The diagram illustrates the structure of the sentence "John's dog". It consists of two main parts: "John" and "John's dog". The word "John" is positioned below the first part. To its right, a green arrow points left towards the word "dog", which is written vertically. Above "dog", the word "Poss-
dog" is written, with another green arrow pointing left towards it. This indicates that "John" is the possessor of the noun "dog".

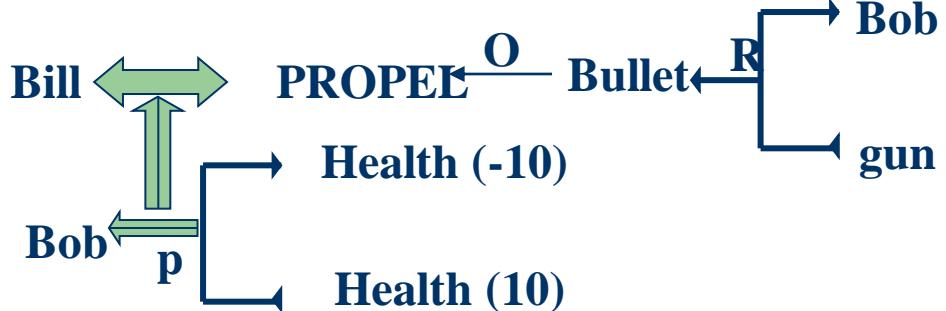
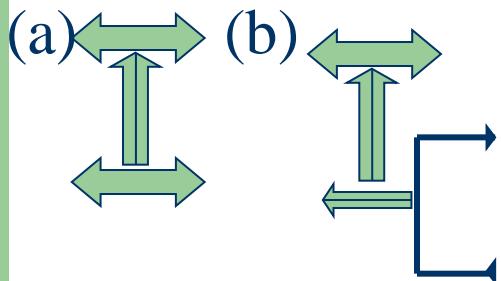
Lưu đồ về quan hệ phụ thuộc khái niệm.



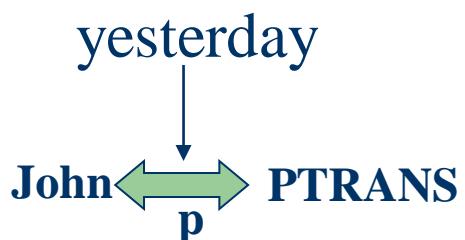
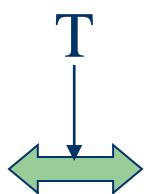
Lưu đồ về quan hệ phụ thuộc khái niệm.



: The plants grew



: Bill shot Bob



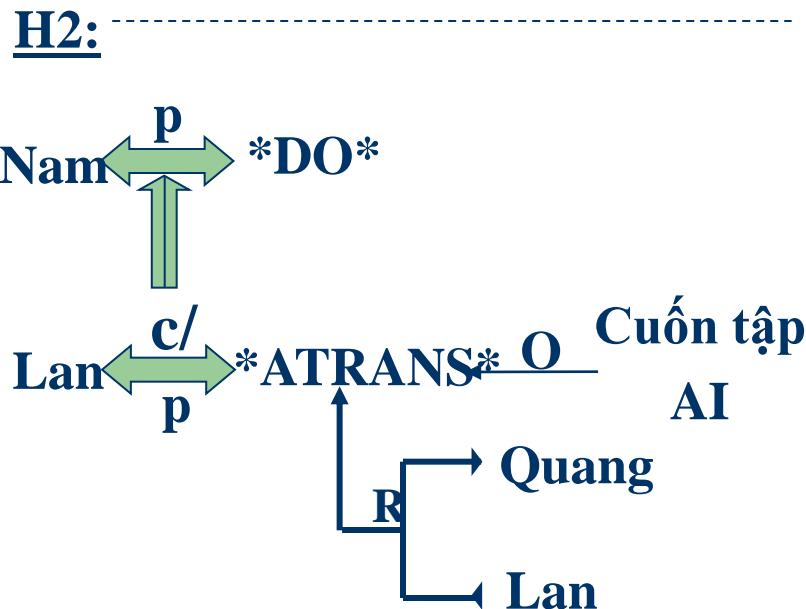
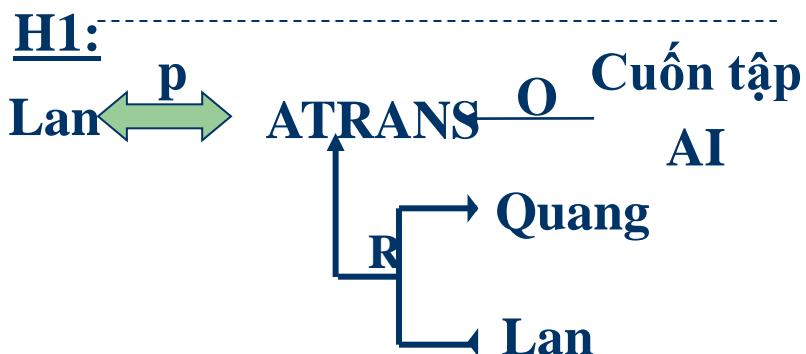
: John ran yesterday

Lưu đồ về quan hệ phụ thuộc khái niệm.

♣ Từ những kết hợp giữa các phụ thuộc khái niệm để biểu diễn các câu đơn giản ở trên, chúng ta có thể cũng có thể tạo ra biểu diễn cho các câu phức tạp hơn như ví dụ sau:
Câu: “*Nam đã cấm Lan gởi cuốn tập AI cho Quang*”

Nếu đặt C là mệnh đề: “*Lan gởi cuốn tập cho Quang*”, thì câu trên có thể hiểu là: Nam cấm cái mệnh đề vừa nêu xảy ra.

Mà mệnh đề C được biểu diễn như **H1**, nên toàn bộ câu là như **H2**:



Lưu đồ về quan hệ phụ thuộc khái niệm.

• Ưu điểm

- ♣ Cung cấp cách thức biểu diễn hình thức cho ngữ nghĩa của ngôn ngữ tự nhiên, ngữ nghĩa được biểu diễn theo dạng có quy tắc → giảm sự nhập nhằng.
- ♣ Chính bản thân dạng biểu diễn chứa đựng ngữ nghĩa → tính **đồng nghĩa** tương ứng là sự **đồng nhất** về cú pháp của lược đồ biểu diễn → chứng minh tính đồng nghĩa ⇔ so trùng hai đồ thị biểu diễn.

• Nhược điểm:

- ♣ Khó khăn trong việc phát triển chương trình để tự động thu giảm biểu diễn của câu bất kỳ về dạng quy tắc chuẩn.
- ♣ Trả giá cho việc phân rã mọi cái về các thành phần cơ bản: ACT, PP, ...
- ♣ Các thành phần cơ bản không thích hợp để miêu tả những khái niệm tinh tế của ngôn ngữ tự nhiên, như các từ có nghĩa định tính: cao, đẹp,...

Đồ thị khái niệm

- **Định nghĩa:**

Đồ thị khái niệm là một đồ thị hữu hạn, liên thông, các đỉnh được chia làm hai loại: đỉnh khái niệm và đỉnh quan hệ.

Đỉnh khái niệm: dùng để biểu diễn các khái niệm cụ thể (cái, điện thoại,...) hay trừu tượng (tình yêu, đẹp, văn hoá,...). Đỉnh khái niệm được biểu diễn bởi hình chữ nhật có gán nhãn là khái niệm.

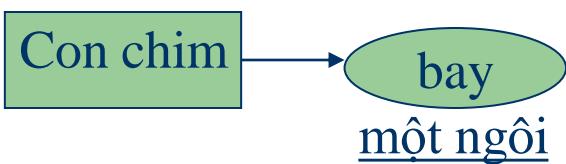
Đỉnh quan hệ: dùng để chỉ ra quan hệ giữa các khái niệm có nối đến nó.

Trong đồ thị khái niệm: chỉ có khái niệm mới nối được với nhau. Chính vì dùng đỉnh quan hệ nên các cung không cần phải được gán nhãn nữa.

- ♣ Mỗi đồ thị khái niệm biểu diễn một mệnh đề đơn.
- ♣ Cơ sở tri thức: chứa nhiều đồ thị khái niệm.

Đồ thị khái niệm

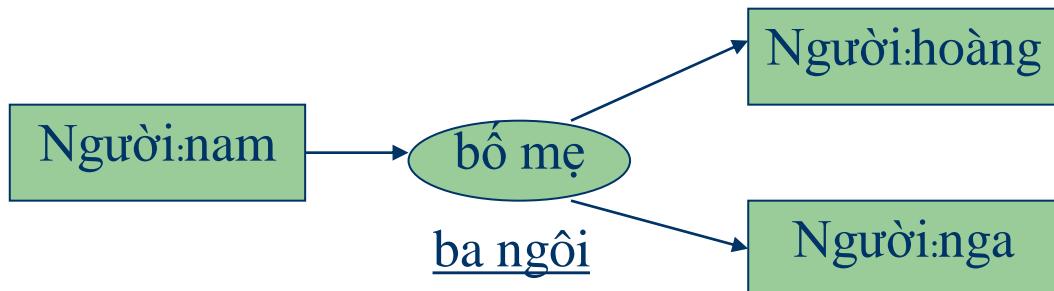
- Một số ví dụ:



Biểu diễn: “Con chim biết bay”



Biểu diễn: ”con chó có màu nâu”.

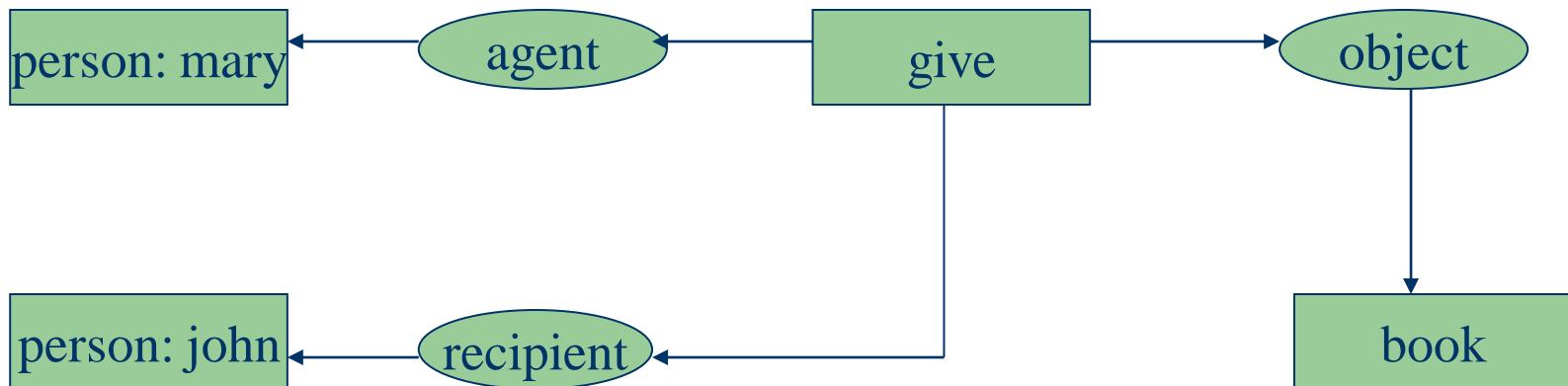


Biểu diễn: ”Nam có bố mẹ là ông Hoàng và bà Nga”.

* Một đỉnh quan hệ có thể là một hay nhiều ngôi.

Đồ thị khái niệm

- Một số ví dụ:



Represent: “Mary give John the book”

Trong ví dụ trên, chú ý: động từ “give” có chủ từ thông qua đỉnh quan hệ “agent”, tân ngữ trực tiếp thông qua đỉnh quan hệ “object”, tân ngữ gián tiếp cũng là người nhận thông qua đỉnh quan hệ “recipient”, hướng mũi tên cho các loại động từ tương tự có dạng như đồ thị trên.

Đồ thị khái niệm

● Loại, cá thể, tên:

Trong đồ thị khái niệm, mỗi đỉnh quan hệ biểu diễn cho một cá thể đơn lẻ thuộc một loại nào đó. Để nói lên quan hệ giữa “loại-cá thể”, nên mỗi đỉnh khái niệm được quy định cách gán nhãn là:

“loại: tên_cá_thể”

tên_cá_thể có thể là:

1. Một tên nào đó, như:

sinhviên: nam → một sinh viên có tên là Nam.

2. Một khoá để phân biệt, được viết theo cú pháp **#khoá**, như

sinhviên: #59701234 → một sinh viên có khoá là: 59701234.

3. Có thể dùng dấu sao (*) để chỉ ra một cá thể chưa xác định, như:

sinhviên: *, có tác dụng như sinhviên → chỉ ra một sinh viên bất kỳ

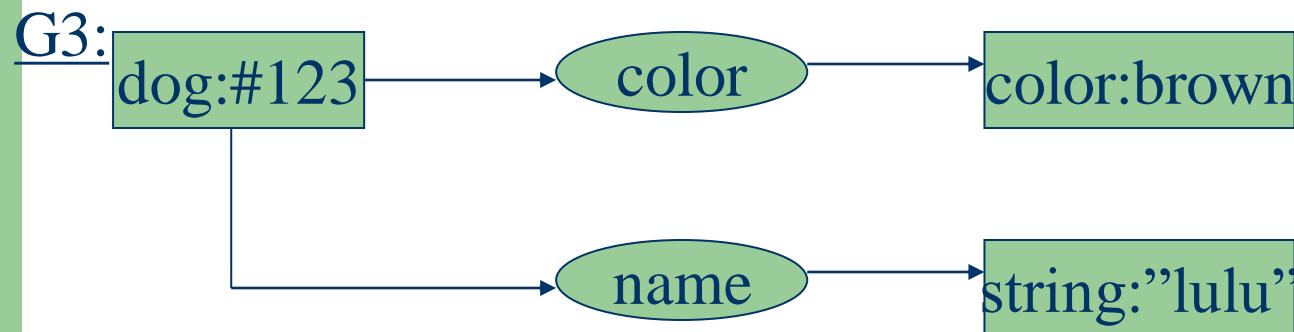
sinhviên:X → sinh viên bất kỳ, tên sinh viên đã được lấy qua biến X.

sinhviên:ng* → sinh viên có tên bắt đầu bởi “ng”

Trường hợp 1 và 2, khái niệm được gọi là khái niệm cá thể, trường hợp 3 ta có khái niệm tổng quát.

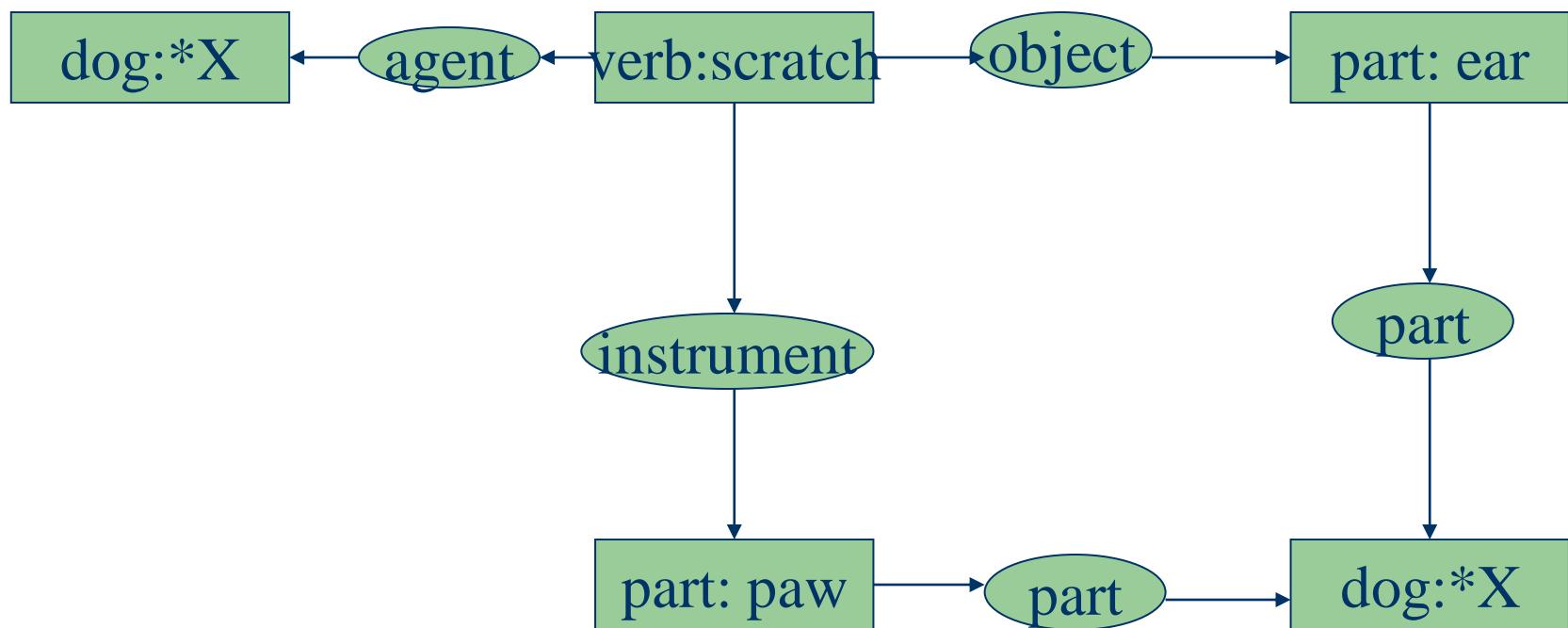
Đồ thị khái niệm

- Nếu dùng cách đặt tên như nói trên có thể nhìn thấy 3 đồ thị sau có tác dụng biểu diễn như nhau nếu con lulu có khoá là #123.



Đồ thị khái niệm

- Biến có thể được dùng khi cần chỉ ra nhiều định khái niệm đồng nhất nhau trong một đồ thị như trường hợp sau.



Represent: “The dog scratches its ear with its paw”

Đồ thị khái niệm

- Phân cấp loại (type)

Nếu có s và t là hai loại (type) thì:

s ≤ t : \rightarrow s: **subtype** của t
 \rightarrow t : **supertype** của s

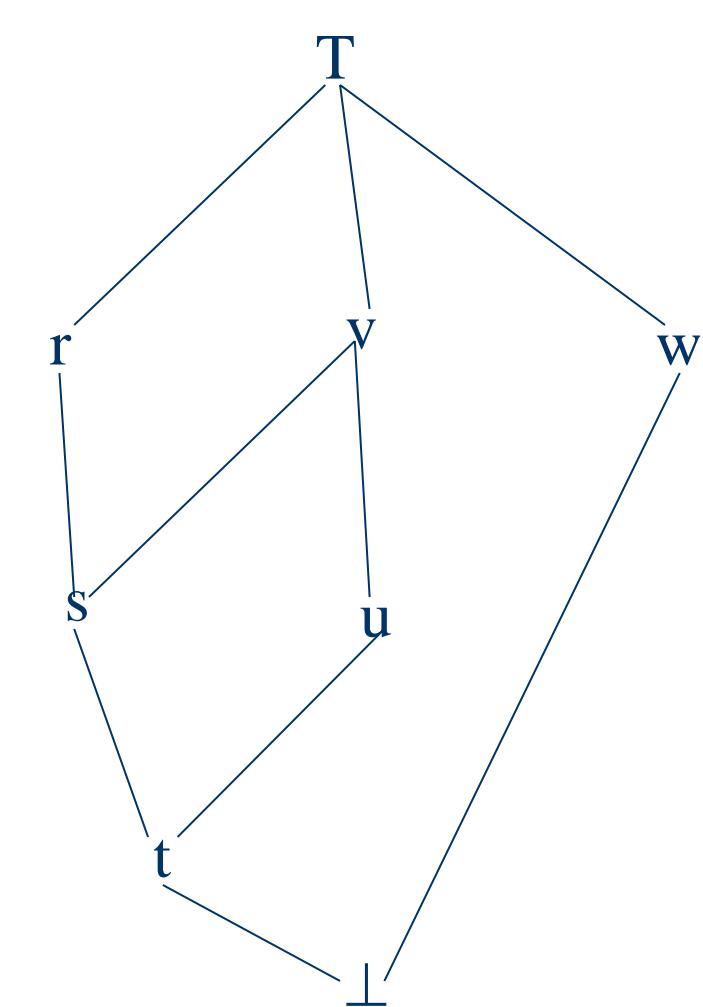
Ví dụ:

- sinhviên là subtype của người.
- người là super type của sinhviên.

\rightarrow nên viết: *sinhviên ≤ người*

Trong sơ đồ phân cấp bên,

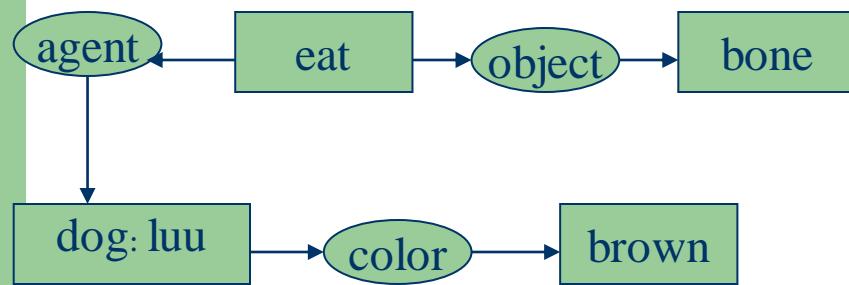
- s: được gọi là **common-subtype** của r và v.
- v : được gọi là **common-supertype** của s và u.
- T : supertype của mọi type
- ⊥ : subtype của mọi type



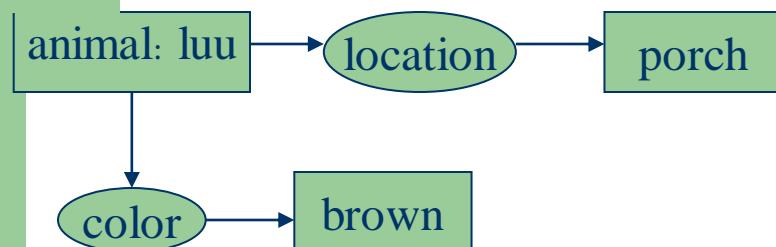
Đồ thị khái niệm

- Các phép toán trên đồ thị khái niệm.
Xét hai đồ thị sau:

G1:



G2:



♣ **Phép copy (nhân bản):** nhân bản một đồ thị.

♣ **Phép Restriction (giới hạn):** tạo ra đồ thị mới bằng cách: từ một đồ thị đã có, thay thế một đỉnh khái niệm bời một đỉnh khác cụ thể hơn, như hai trường hợp:

→ Một biến *, được thay thế bởi một khoá, hay một tên của cá thể.

VD: *dog:** → *dog:#123* hay *dog:luu*

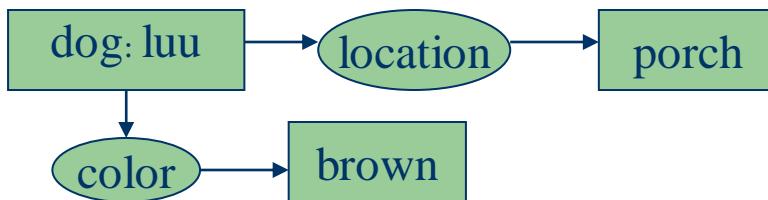
→ Một type được thay thế bởi subtype của nó.

VD: *người: nam* → *sinhviên:nam*

Đồ thị khái niệm

Aùp dụng phép restriction trên đồ thị G2, có thể dẫn ra G3 như sau:

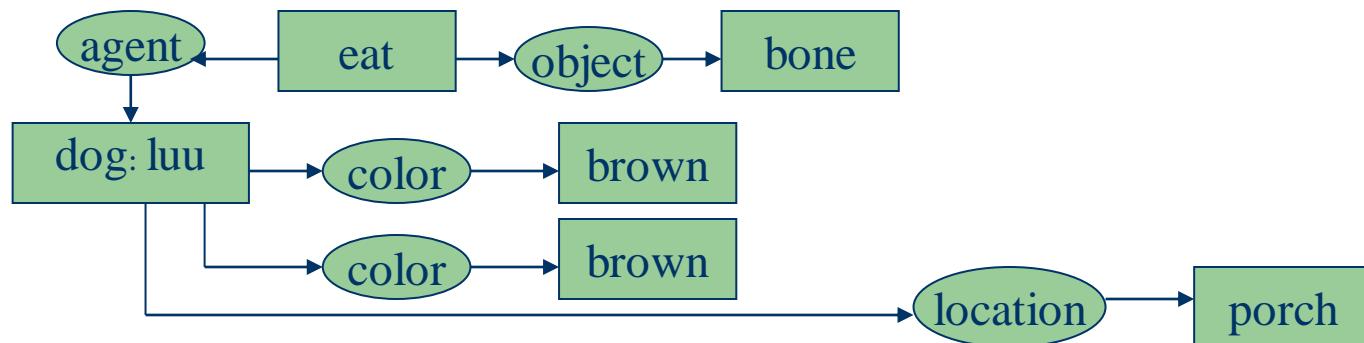
G3:



♣ **Phép Join (nối):** Nối hai đồ thị để được một đồ thị khác.

Nếu có đỉnh khái niệm C xuất hiện trên cả hai đồ thị X và Y, thì chúng ta có thể nối hai đồ thị trên đỉnh chung C nói trên, như từ G1 và G3 có thể tạo ra G4 như sau: (nối trên đỉnh chung là: *dog:luu*)

G4:

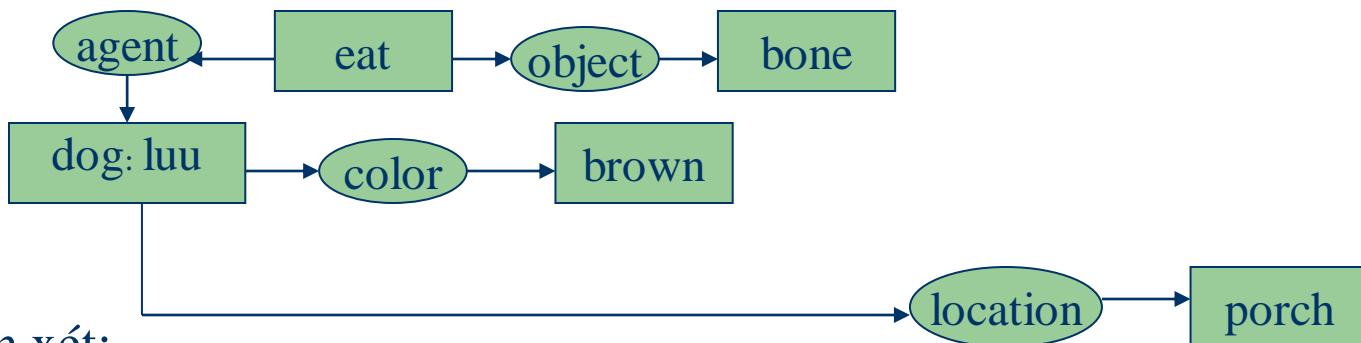


Đồ thị khái niệm

♣ Phép simplify: (rút gọn)

Nếu trên một đồ thị có hai đồ thị con giống nhau hoàn toàn thì chúng ta có thể bỏ đi một để tạo ra một đồ thị mới có khả năng biểu diễn không thay đổi. Từ G4 có thể sinh ra G5 cùng khả năng biểu diễn.

G5:



Nhận xét:

Phép Restriction và phép Join cho phép chúng ta thực hiện tính thừa kế trên đồ thị khái niệm. Khi thay một biến * bởi một cá thể cụ thể, lúc đó chúng ta cho phép cá thể thừa kế các tính chất từ loại(type) của nó, cũng tương tự khi ta thay thế một type bởi subtype của nó.

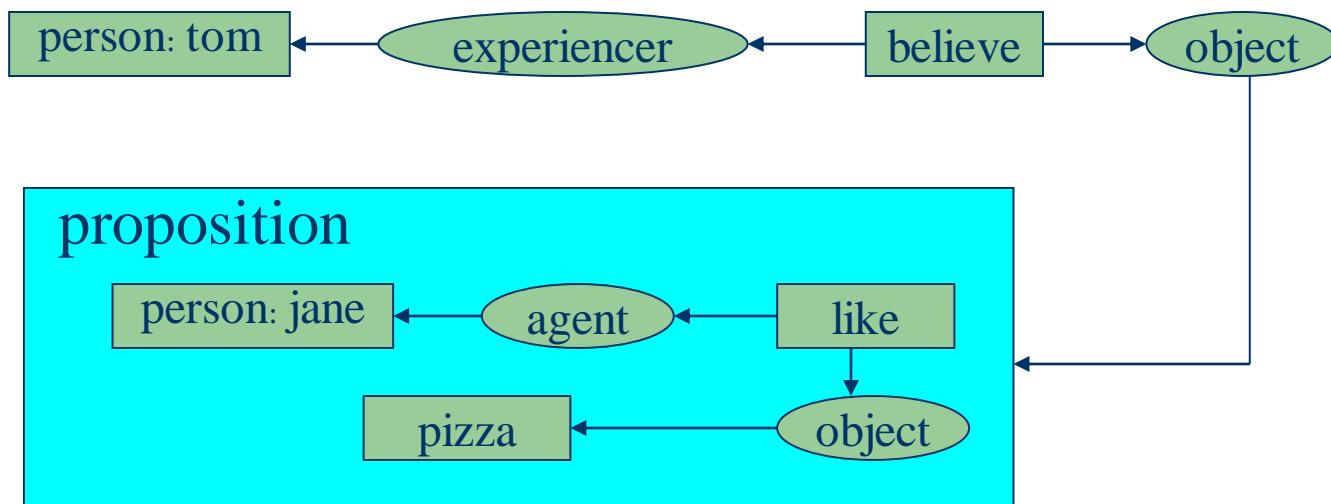
Đồ thị khái niệm

- **Đỉnh mệnh đề:**

Để thuận tiện biểu diễn cho các câu gồm nhiều mệnh đề, đồ thị khái niệm đã được mở rộng để có thể chứa cả một mệnh đề trong một đỉnh khái niệm, lúc đó chúng ta gọi là đỉnh mệnh đề.

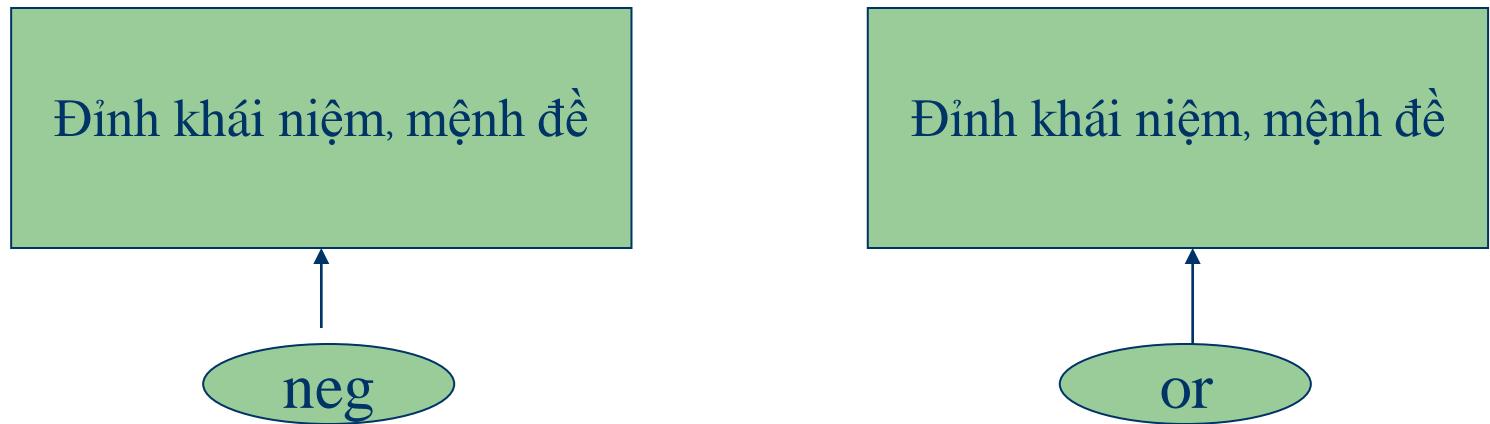
Vậy đỉnh mệnh đề là một đỉnh khái niệm có chứa một đồ thị khái niệm khác. Xét đồ thị khái niệm mở rộng biểu diễn cho câu:

“Tom believes that Jane likes pizza”.



Đồ thị khái niệm

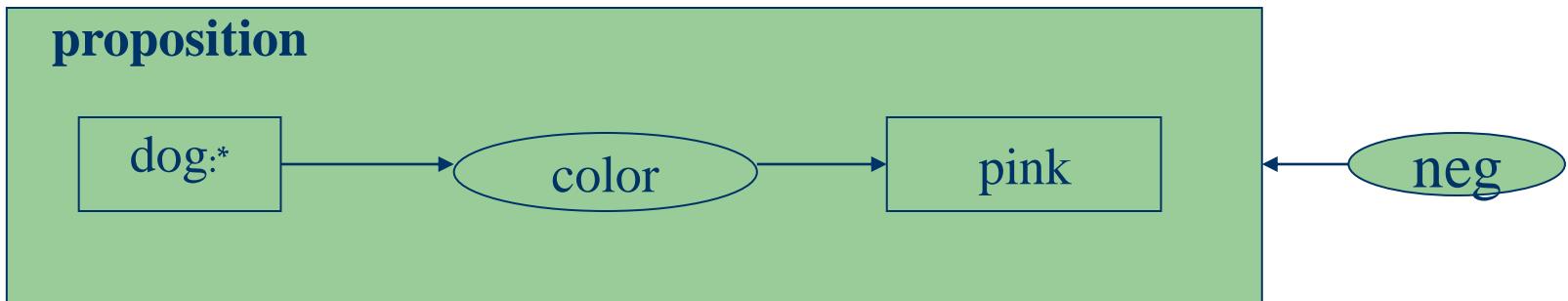
- Đồ thị khái niệm và logic.
 - Phép hội (and) của nhiều khái niệm, mệnh đề chúng ta có thể thực hiện dễ dàng cách cách nối nhiều đồ thị bởi phép toán join.
 - Phép phủ định(not) và phép tuyển(or) giữa các khái niệm hay mệnh đề cũng có thể được thể hiện bằng cách đưa vào đỉnh quan hệ có tên: neg(phủ định), or(tuyển) như dạng sau.



Đồ thị khái niệm

Ví dụ:

Câu: “There are no pink dogs”, được biểu diễn:



Trong đồ thị khái niệm, các khái niệm tổng quát (định dùng biến * - như **dog:***, hay chỉ có tên loại - như **dog**) được xem như có lượng từ tồn tại (\exists). Do vậy, mệnh đề trong ví dụ trên có biểu diễn vị từ là:

$$\exists X \exists Y (dog(X) \wedge color(X, Y) \wedge pink(Y)).$$

Và toàn bộ đồ thị (bao gồm định quan hệ :neg), có biểu diễn vị từ:

$$\neg \exists X \exists Y (dog(X) \wedge color(X, Y) \wedge pink(Y)).$$

$$= \forall X \forall Y (\neg (dog(X) \wedge color(X, Y) \wedge pink(Y))).$$

Đồ thị khái niệm

- Giải thuật để chuyển một đồ thị khái niệm sang biểu diễn vị từ:
 1. Gán một biến riêng biệt (X_1, X_2, \dots) cho mỗi khái niệm tổng quát.
 2. Gán một hằng cho mỗi khái niệm cá thể trong đồ thị. Hằng này có thể là tên cá thể hay khoá của nó.
 3. Biểu diễn một đỉnh khái niệm bởi một vị từ một ngôi; có tên là tên loại (type), đối số là biến hay hằng vừa gán trên.
 4. Biểu diễn mỗi đỉnh quan hệ bởi một vị từ n ngôi; có tên là tên của đỉnh quan hệ, các thông số là biến hay hằng được gán cho các đỉnh khái niệm nối đến nó.
- 5. Hội của tất cả các câu trong bước 3 và 4. Tất cả các biến trong biểu thức thu được đều đính kèm lượng từ tồn tại.

Ví dụ: có đồ thị như sau



Được chuyển sang là:

$\exists X_1(\text{dog}(X_1) \wedge \text{color}(X_1) \wedge \text{brown}(X_1))$

Lược đồ có cấu trúc - Frame

- Frame – khung.

Là một cấu trúc dữ liệu cho phép biểu diễn tri thức ở dạng khái niệm hay đối tượng.

Một khung có cấu trúc như hình vẽ bên.

- Cấu trúc của frame:

Đặc tả cho một frame gồm các thành phần cơ bản sau:

1. Frame name: tên của frame.

- Nếu frame biểu diễn cho một cá thể nào đó, thì đây là tên của cá thể.

Ví dụ: an, nam, lulu,..

Frame
name:

Class:

Properties:

Object1

Object2

Property 1	Value1
------------	--------

Property 2	Value2
------------	--------

...	...
-----	-----

...	...
-----	-----

Lược đồ có cấu trúc – Frame

● Cấu trúc của frame (tt):

- Nếu Frame biểu diễn cho một lớp, thì đây là tên lớp. Ví dụ: chim, động vật, ...

2. Class: Tên loại.

- Nếu thành phần này xuất hiện, nó cho biết rằng frame mà chúng ta đang biểu diễn có loại là giá trị trường class. → Cho phép thành lập quan hệ thừa kế IS-A.

Như ví dụ trên, chúng ta có:

Object1 IS-A Object2

3. Các thuộc tính (property): Khi biểu diễn một frame chúng ta có thể thiết lập một hay nhiều thuộc tính cho nó, như ví dụ sau:

Frame name:	chim
Properties:	
maøu	Chöa bieåt
aên	Coân truong
Soá caùnh	2
bay	true
ñoùi	Chöa bieåt
Hoaït ñoäng	Chöa bieåt

Lược đồ có cấu trúc – Frame

- Cấu trúc của frame (tt):
 - Khi chúng ta đặt tả thuộc tính cho một lớp; nếu chúng ta biết được giá trị chung cho tất cả các đối tượng thuộc lớp mà chúng ta đang biểu diễn thì điền vào trị cho thuộc tính đó, giá trị đó chúng ta gọi là **giá trị mặc nhiên, như: ăn, số cánh ở trên**; nếu chúng ta chưa biết trị cụ thể (nhưng biết là có thuộc tính đó) thì chúng ta có thể bỏ trống (**chưa biết**) – **như màu, hoạt động,...**.

Các thuộc tính của frame nằm ở hai dạng cơ bản:

→ *Dạng tĩnh(static)*: giá trị của nó không thay đổi trong quá trình hệ thống tri thức hoạt động.

→ *Dạng động(dynamic)*: giá trị có thể chuyển đổi.

Khi phải tìm kiếm một frame, chúng ta có thể dựa vào **frame name**, cũng có thể dựa vào các thuộc tính được đặt tả cho frame.

Lược đồ có cấu trúc – Frame

- Cấu trúc của frame (tt):

4. Các thủ tục: Lược đồ frame càng cho phép tích hợp cách thức đặt như các thuộc tính như trên và các thủ tục vào một frame. Về hình thức, một thủ tục sẽ chiếm một khe tương tự như khe thuộc tính nói trên.

Thủ tục được dùng để: biểu diễn một hành động nào đó của đối tượng, điều khiển giá trị của thuộc tính như: kiểm tra ràng buộc về trị, kiểu, của thuộc tính mỗi khi cần trích, hay thay đổi nó.

Hai thủ tục phổ biến được đính kèm với một thuộc tính là:

***IF_NEEDED* và
IF_CHANGED.**

→ *IF_NEEDED*:

Thủ tục này được thực thi mỗi khi chúng ta cần đến giá trị của thuộc tính (giống thủ tục GET trong VB).

Ví dụ: thủ tục sau (dạng if_needed) cho thuộc tính ***bay*** của frame ***chim*** nói trên.

If self:số_cánh < 2

Then self:bay = false

If self:số_cánh = 2

Then self:bay = true

Lược đồ có cấu trúc – Frame

● Cấu trúc của frame (tt):

→ IF_CHANGED:

Thủ tục này được thực thi mỗi khi giá trị của thuộc tính mà if_changed này được gắn vào thay đổi. (giống như SET, LET trong VB)

Ví dụ: gắn thủ tục sau cho thuộc tính *đói* của lớp *chim* nói trên.

If Seft:đói = true

Then Seft:hànhđộng =
 eating # seft:ăn

4. Các thông tin khác:

Một số khe khác của frame có thể chứa frame khác, link đến frame, mang ngữ nghĩa, rules, hay các loại thông tin khác.

Chú ý: các ví dụ trên mô phỏng theo ngôn ngữ *Kappa PC*, trong đó, “*Expert System -DurKin*”:

- Seft: từ khoá chỉ chính bản thân frame đang mô tả (như Me của VB, this của VC)

- # : dấu nối chuỗi(như & của VB, + của VC)

- Lược đồ frame cũng giống như các hệ thống hướng đối tượng. Chúng ta:

 → Có thể đặt tả frame lớp hay cá thể.

 → Có thể đặt tả tính thừa kế.

 → Mỗi khi tạo ra frame cá thể, có thể copy các thuộc tính, thủ tục của frame lớp; đồng thời có thể mở rộng thêm, hay định nghĩa lại một số thuộc tính, thủ tục.

Lược đồ có cấu trúc – Script

- Script – Kịch bản:

Là một lược đồ biểu diễn có cấu trúc, dùng để biểu diễn một chuỗi các sự kiện trong một ngữ cảnh cụ thể. Nó như một phương tiện để tổ chức các phụ thuộc khái niệm – (đã giới thiệu trước) để mô tả một tình huống cụ thể.

Script được dùng trong các hệ thống hiểu NNTN, tổ chức tri thức trong thành phần các tình huống mà hệ thống phải tìm hiểu.

- Cấu trúc của Script:

1. Entry conditions:

Các điều kiện phải true để script được gọi. Ví dụ: một cá nhân bị bệnh thì script nhập viện được gọi.

2. Results:

Kết quả thu được từ script khi nó hoàn thành.

3. Props:

Các đồ vật tham gia vào script, như: xe cứu thương, cán, bình oxy,...

4. Roles:

Các cá nhân tham gia vào script, như: bệnh nhân, bác sĩ, y tá, người nhà,...

5. Scenes:

Các cảnh chính trong script, như: di chuyển, cấp cứu, hồi sức,..

→ Một ví dụ về kịch bản đi nhà hàng như ví dụ sau:

Lược đồ có cấu trúc – Script

Script: RESTAURENT

Track: Coffe Shop

Entry conditions:

S is hungry

S has money

Results:

S has less money

O has more money

S is not hungry

S is pleased (optional)

Props: Tables

Menu

Food (F)

Check

Money

Roles: Custumer (S)

Waiter(W)

Cook(C)

Cashier(M)

Owner(O)

Scene 1: (Entering)

S PTRANS S into restaurent.

S ATTEND eyes to tables

S MBUILD where to sit

S PTRANS S to table

S MOVE S to sitting position

Scene 2: (Ordering)

(Menu on table)

S PTRANS menu to S

(S ask for menu)

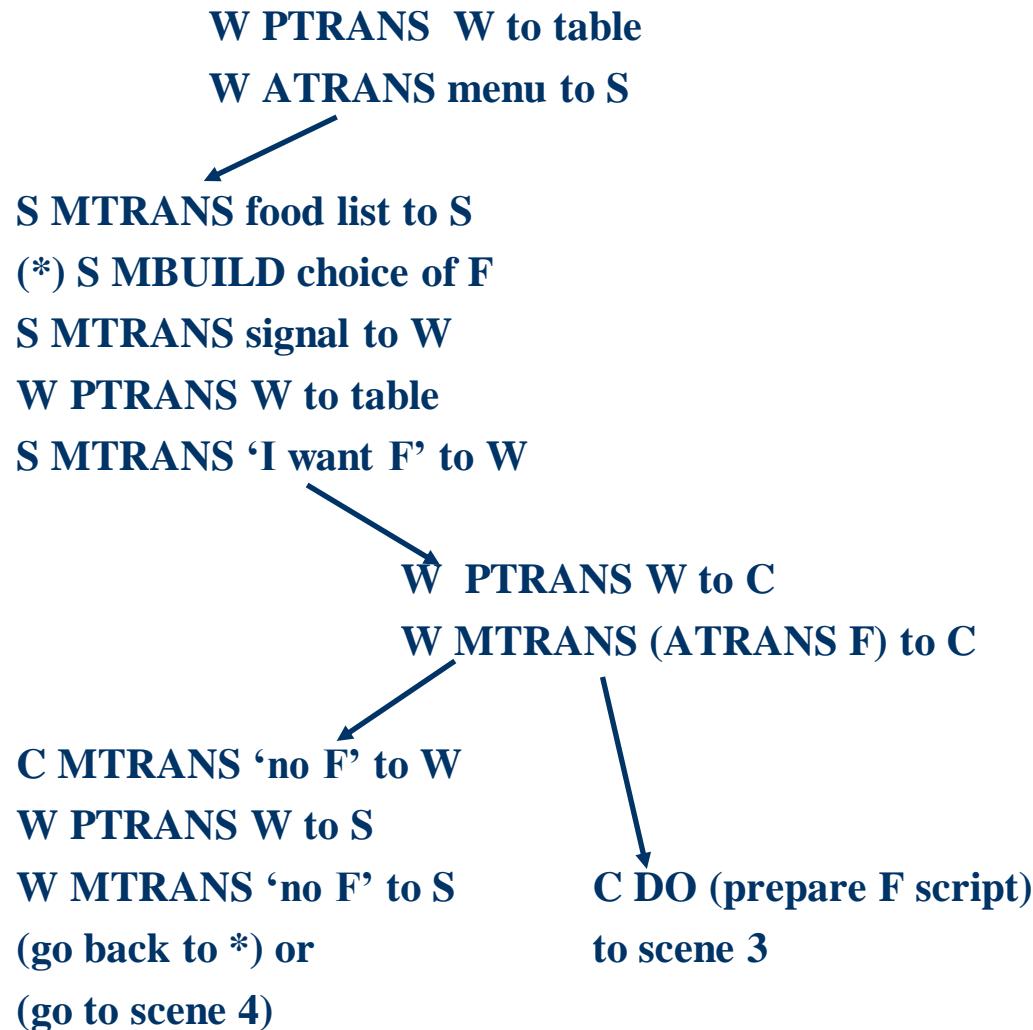
S MTRANS signal to W

W PTRANS W to table

S MTRANS ‘need menu’ to W

W PTRANS W to menu

Lược đồ có cấu trúc – Script



Lược đồ có cấu trúc – Script

Scene 3: (Eating)

C ATRANS F to W

W ATRANS F to S

S INGEST F

(Option: return to scene 2 to order more;

Scene 4: (exiting)



S MTRANS to W

W ATRANS check to S



otherwise: goto scene 4



W MOVE (write check) ;

W PTRANS W to S

W ATRANS check to S ;

S ATRANS tip to W

S PTRANS S to M;

S ATRANS money to M;

S PTRANS S to out of restaurent.

Chương 8: Thuật toán Di truyền



Thuật toán di truyền

- Thuật toán di truyền
- Genetic Algorithm
- GA

John Henry Holland đề xuất
vào năm 1975

Là gì ?

- Thuật toán di truyền là

Các bước học tập mô phỏng theo
cơ chế di truyền của sinh vật

Cơ chế di truyền của sinh vật

- Luật di truyền của Mendel
- Lý thuyết tiến hóa của Darwin



Luật di truyền của Mendel

1

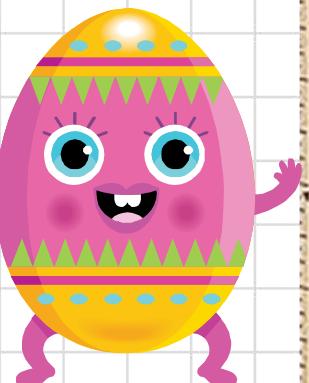


Luật di truyền của Mendel

1



2

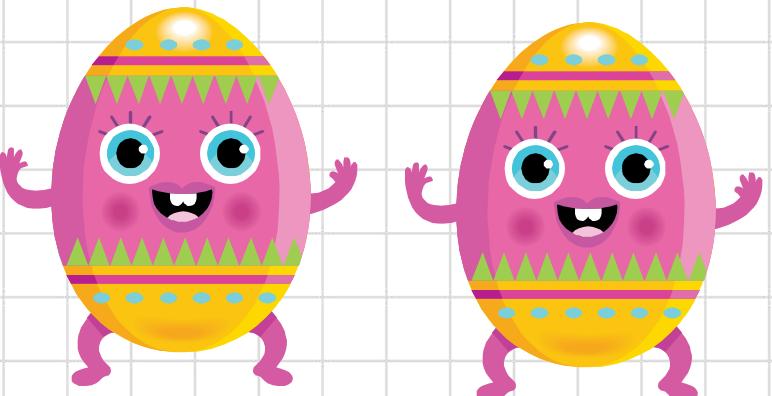


Định luật đồng tính F₁

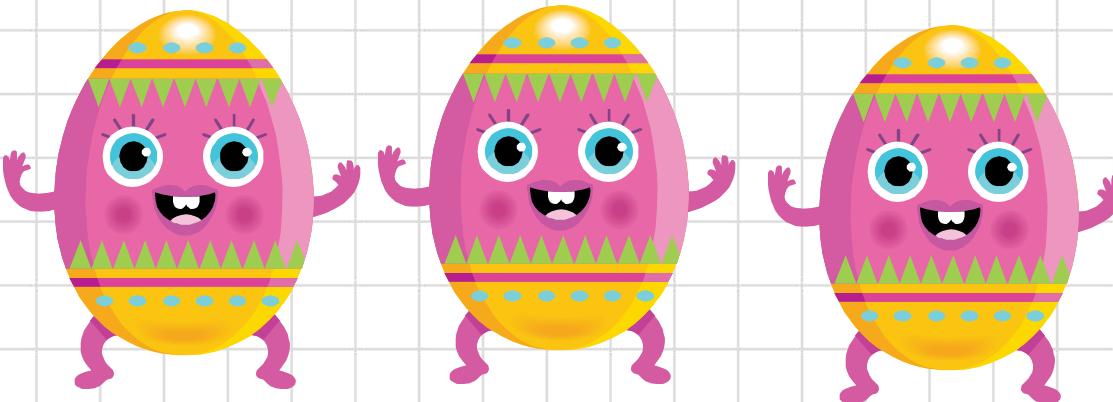


Luật di truyền của Mendel

②



③

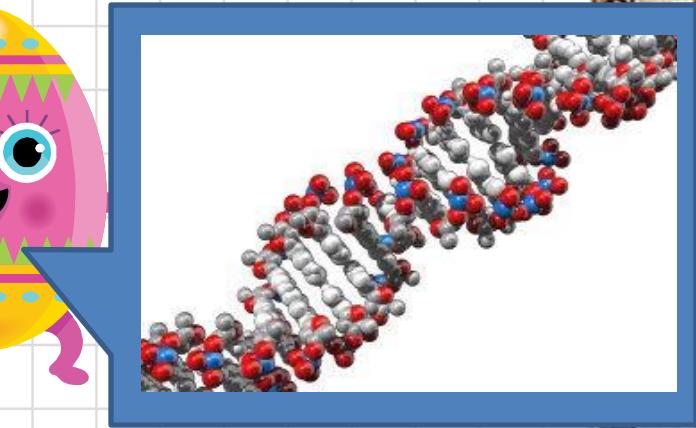


Định luật phân tinh F₂



Luật di truyền của Mendel

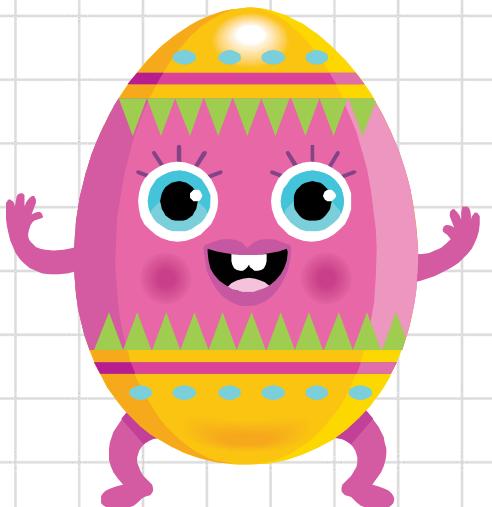
2



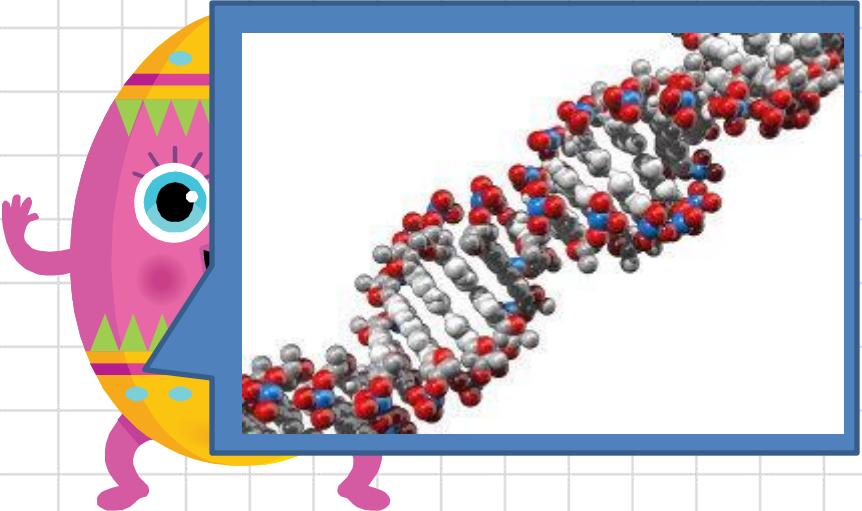
3



Ngoại hình và Gen di truyền



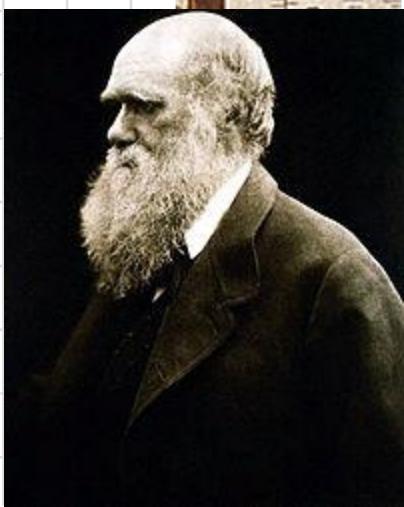
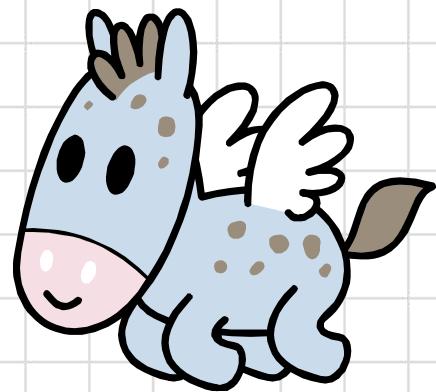
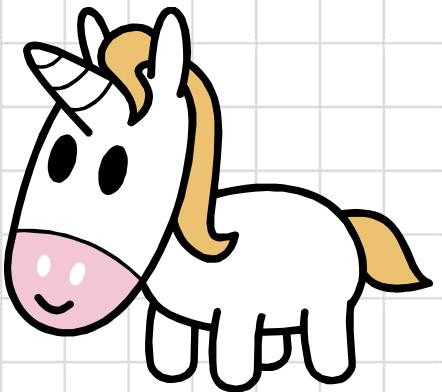
Ngoại hình
Pheno
Type



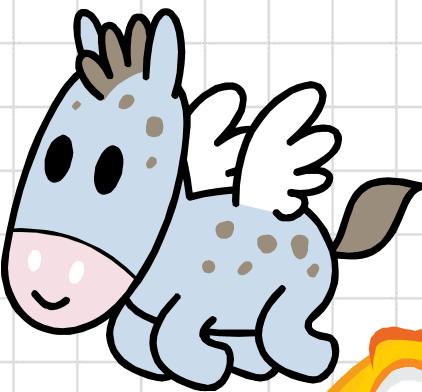
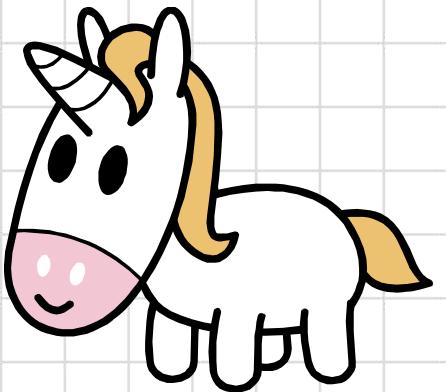
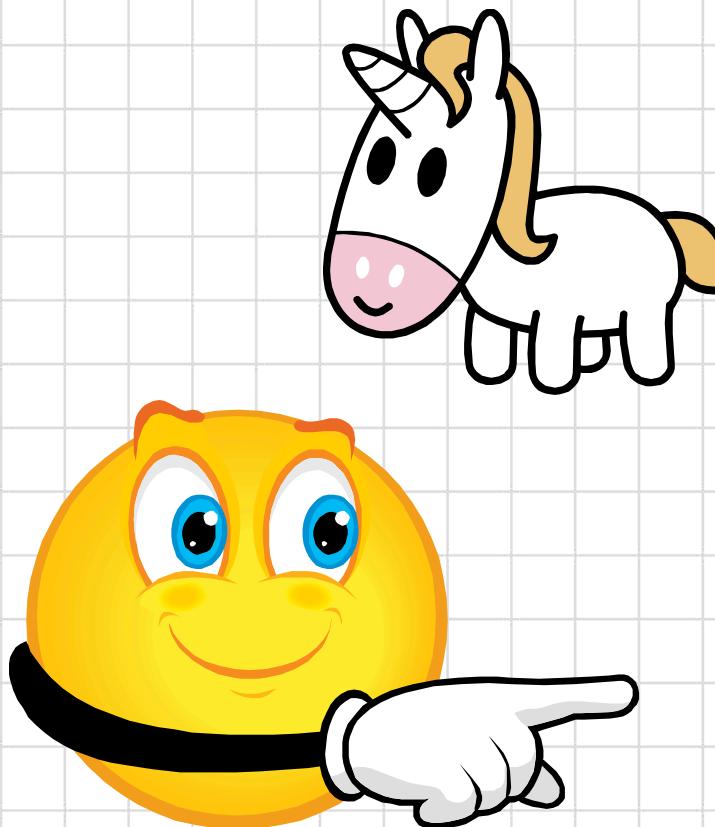
Gen di truyền
Geno
Type



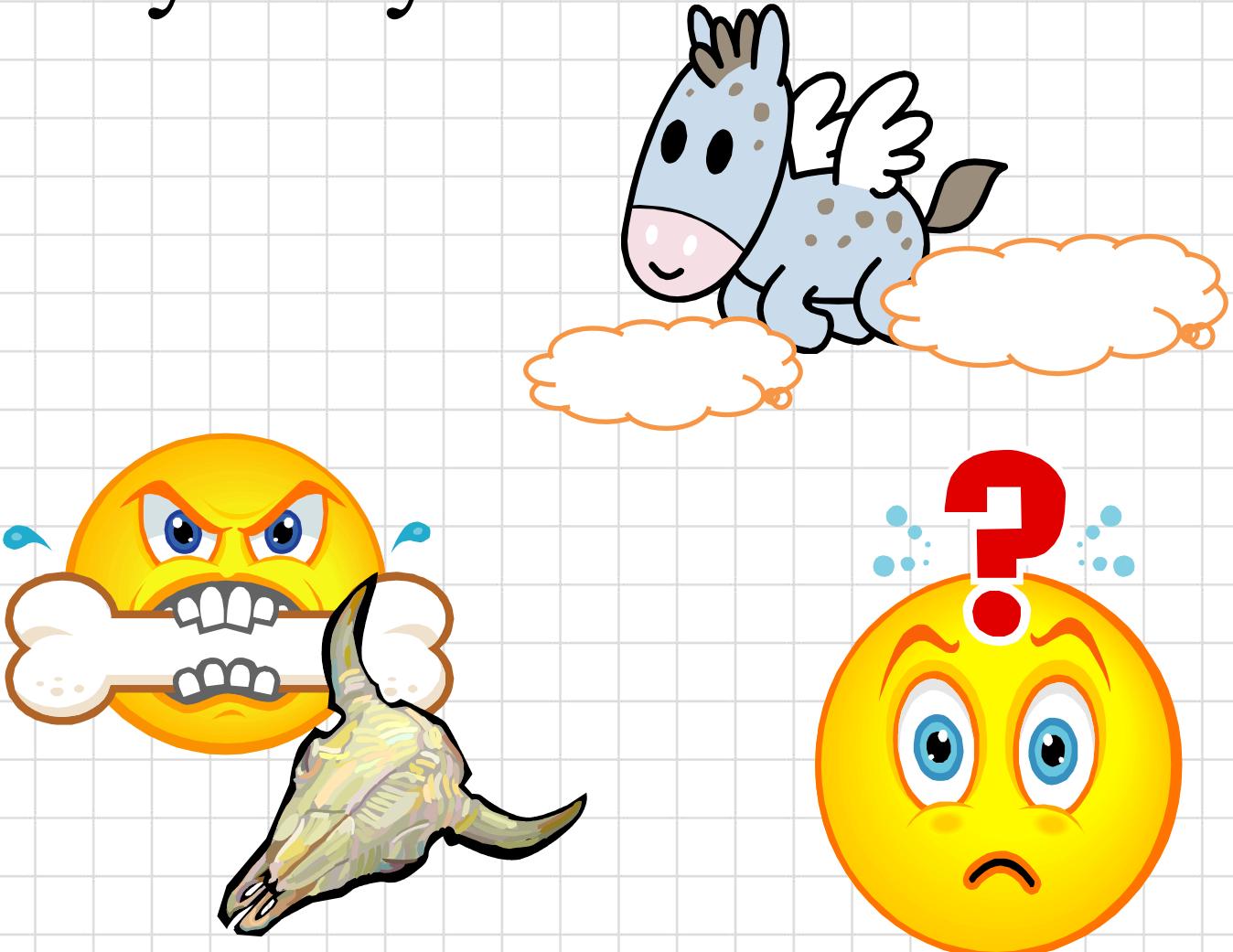
Lý thuyết tiến hóa Darwin



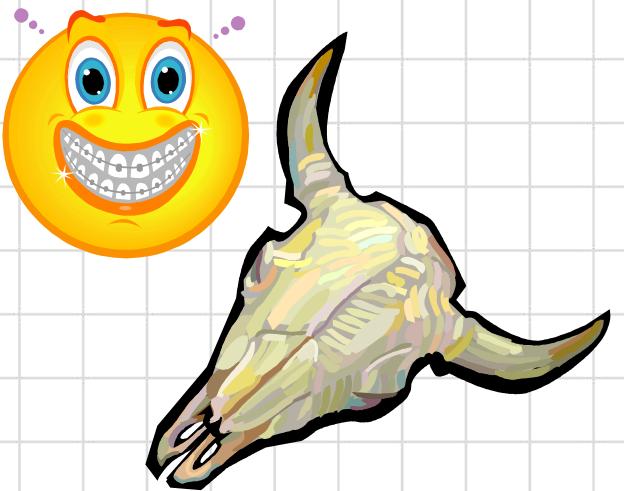
Lý thuyết tiến hóa Darwin



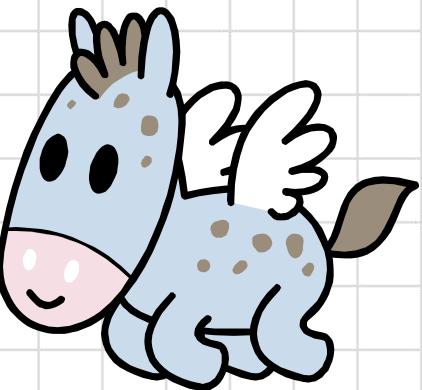
Lý thuyết tiến hóa Darwin



Quá trình chọn lọc



Không
thích nghi

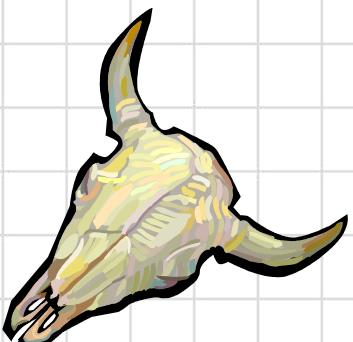


Thích
Nghi

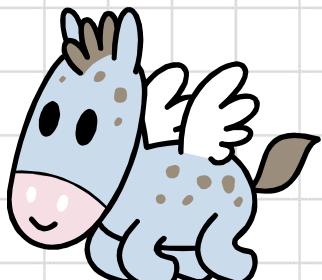


Thuật toán di truyền và Sự tiến hóa của sinh vật

- Mỗi cá thể đều có mức độ thích nghi với môi trường khác nhau
- Mỗi cá thể đều có gen di truyền



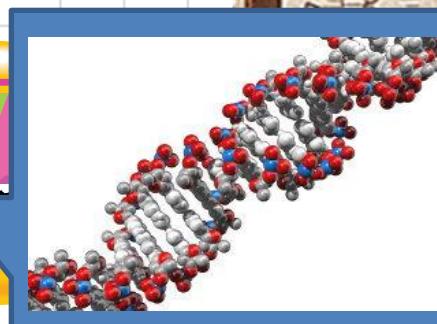
Không
thích nghi



Thích nghi



Ngoại hình

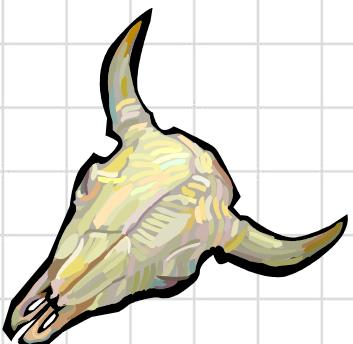


Gen di
truyền

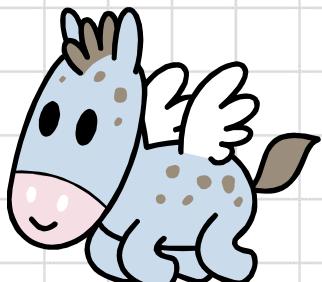


Thuật toán di truyền và Sự tiến hóa của sinh vật

- Cá thể có độ thích nghi cao thì sinh tồn
- Gen di truyền có độ thích nghi cao thì có thể để lại cho các thế hệ tiếp theo



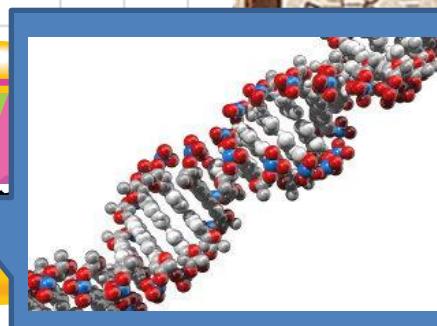
Không
thích nghi



Thích nghi



Ngoại hình



Gen di
truyền

Panta rhei:- Hērakleitos

Everything flows

Là gì ?

- Thuật toán di truyền là

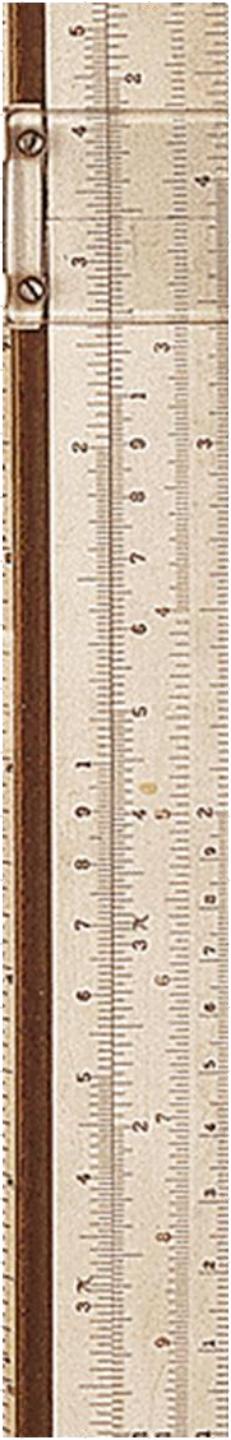
Các bước học tập mô phỏng theo
cơ chế di truyền của sinh vật

Dùng làm gì ?

- Thuật toán di truyền

Được ứng dụng rộng rãi

Trong các bài toán tối ưu



Ví dụ

- Bài toán cái túi
- Hamiltonian path problem
- Bài toán người bán hàng
- Facility location problem
- Matching problem
- Bin packing problem
- Generalized assignment problem

Trước tiên hãy xem cái
đã!!!

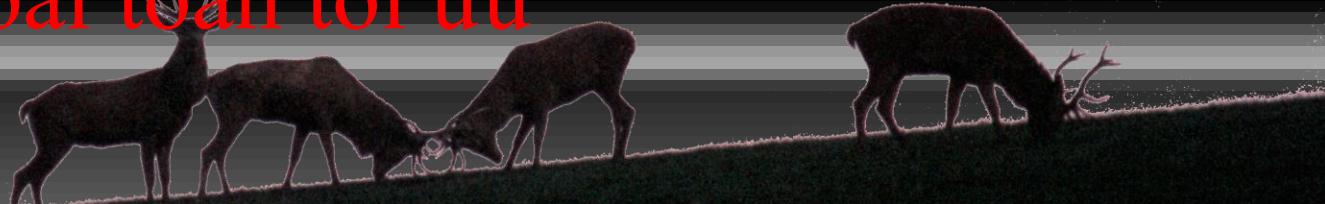


Genetic Algorithm Demo

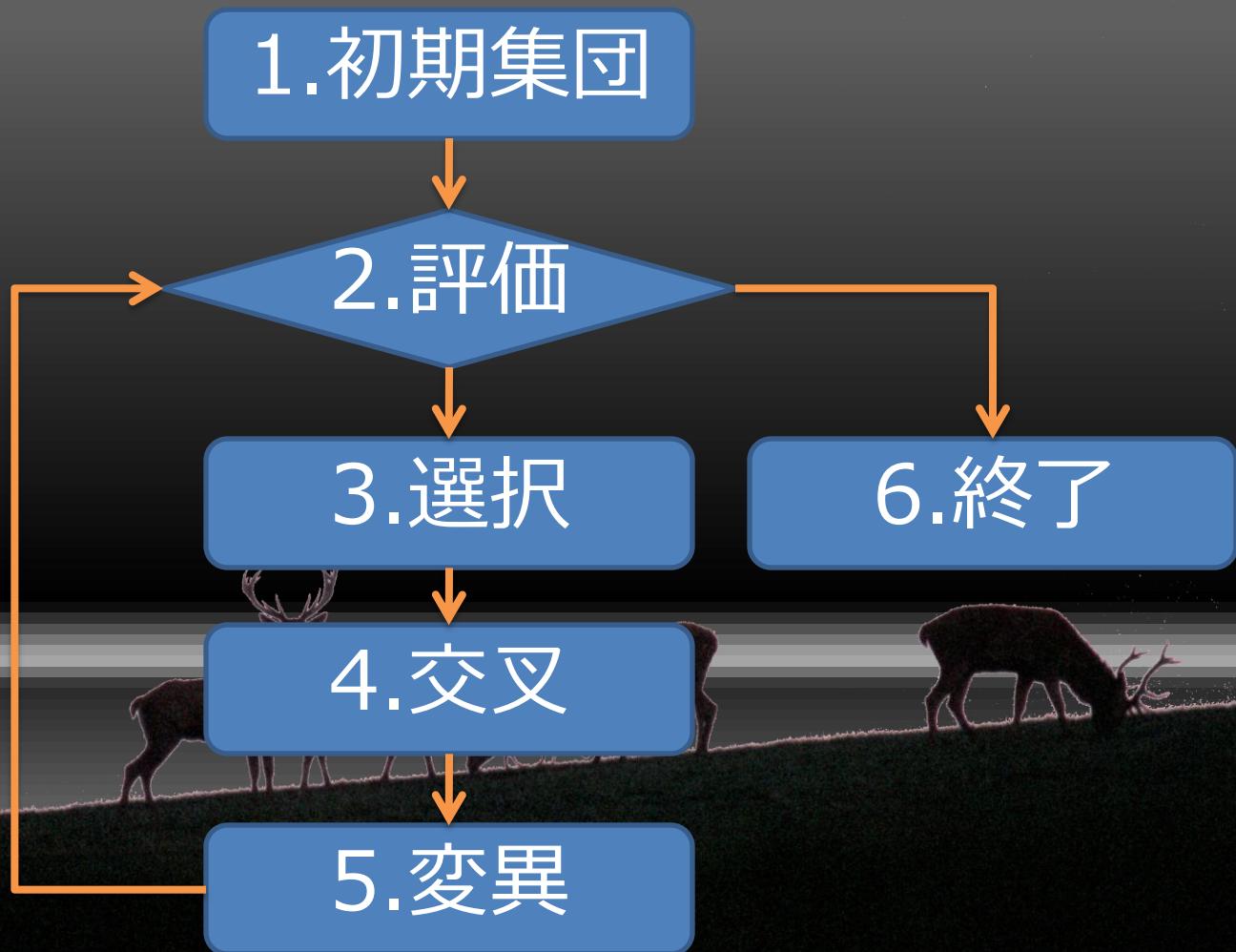
©2008 Jim Frazee

Thuật toán di truyền

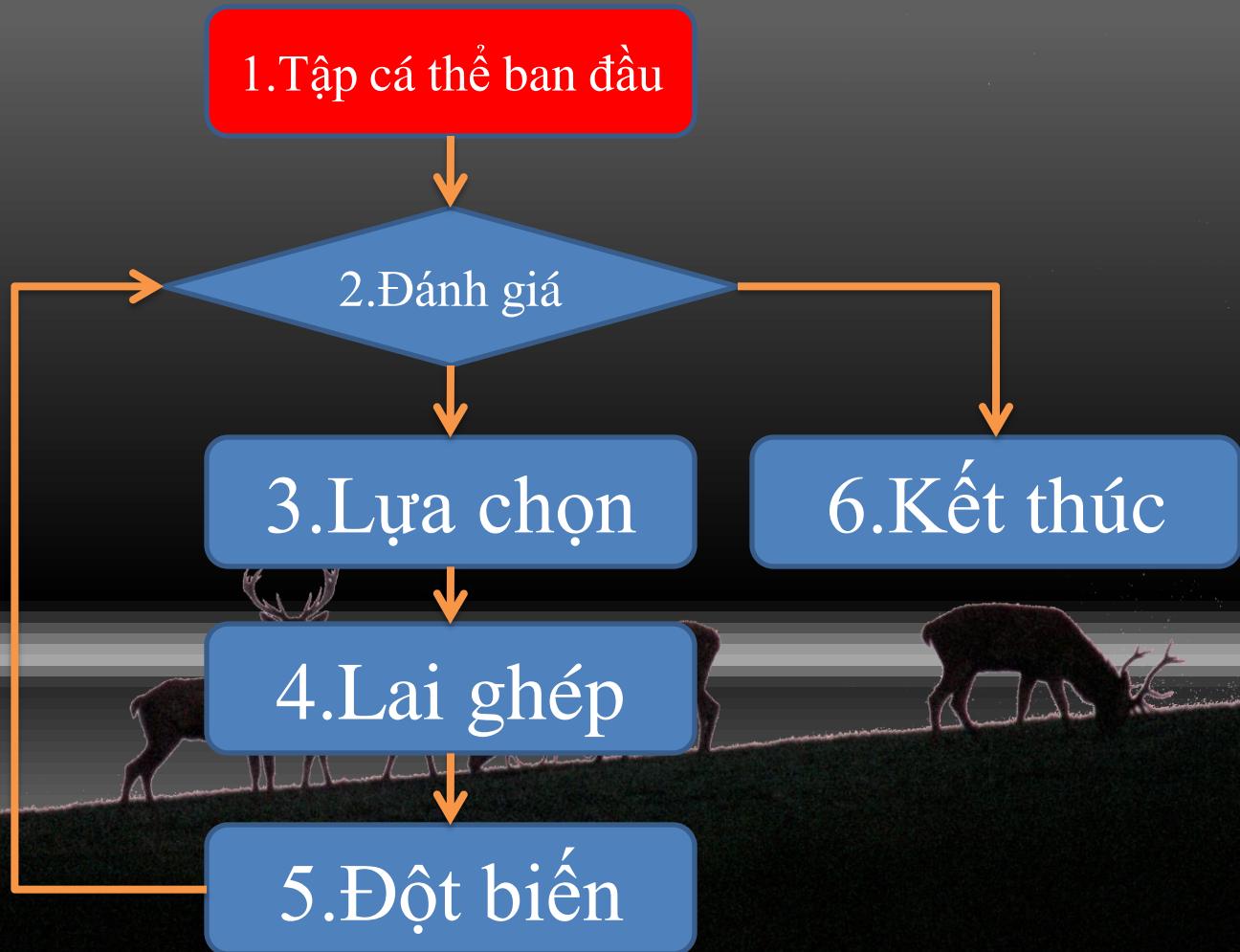
- Các bước học tập mô phỏng theo cơ chế di truyền của sinh vật
- Được ứng dụng rộng rãi trong các bài toán tối ưu



Làm thế nào ?



Bài toán người bán hàng



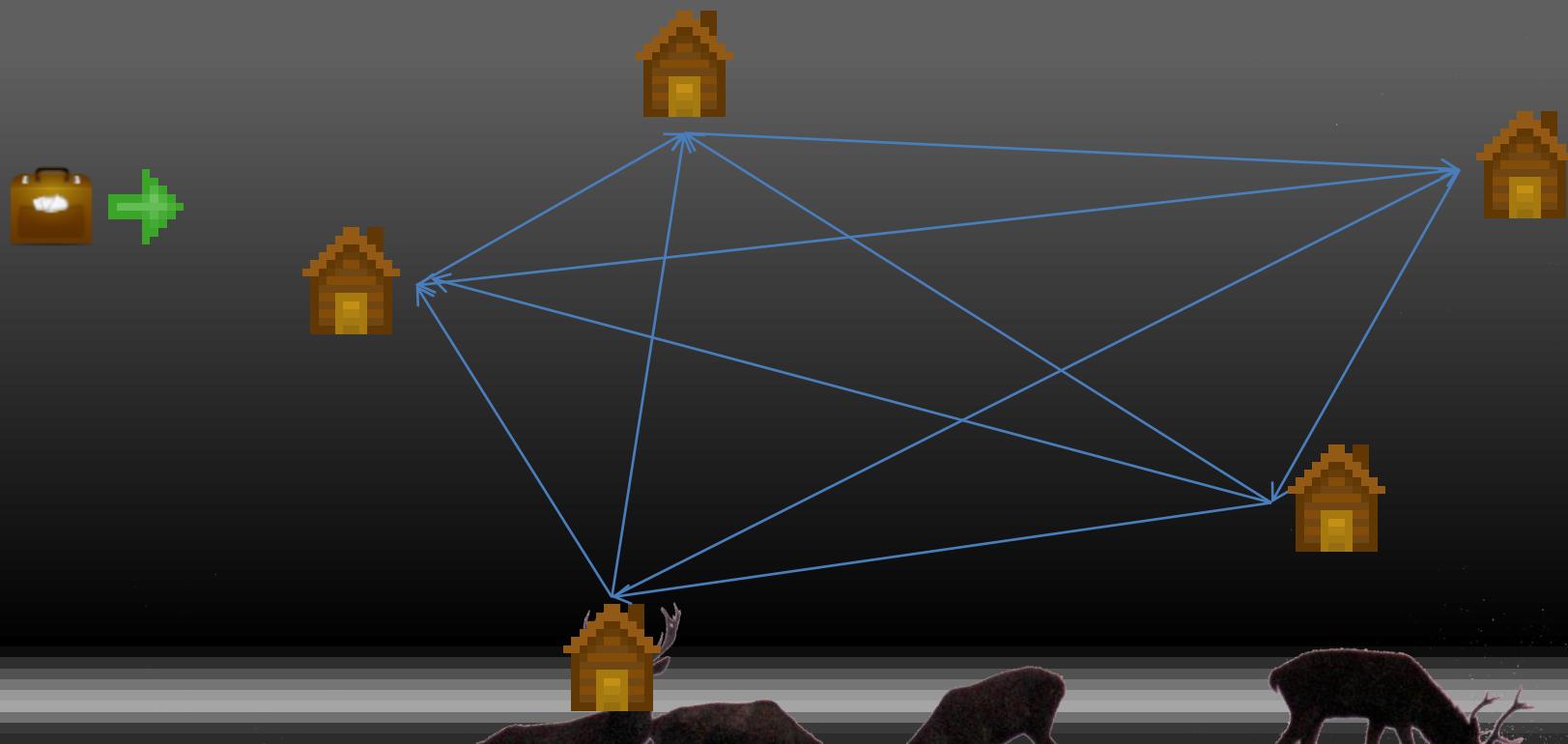
Bài toán người bán hàng

- Bài toán người bán hàng
- Traveling Salesman Problem
- TSP

“Cho trước một danh sách các thành phố và khoảng cách giữa chúng, tìm chu trình ngắn nhất thăm mỗi thành phố đúng một lần”.



Bài toán người bán hàng



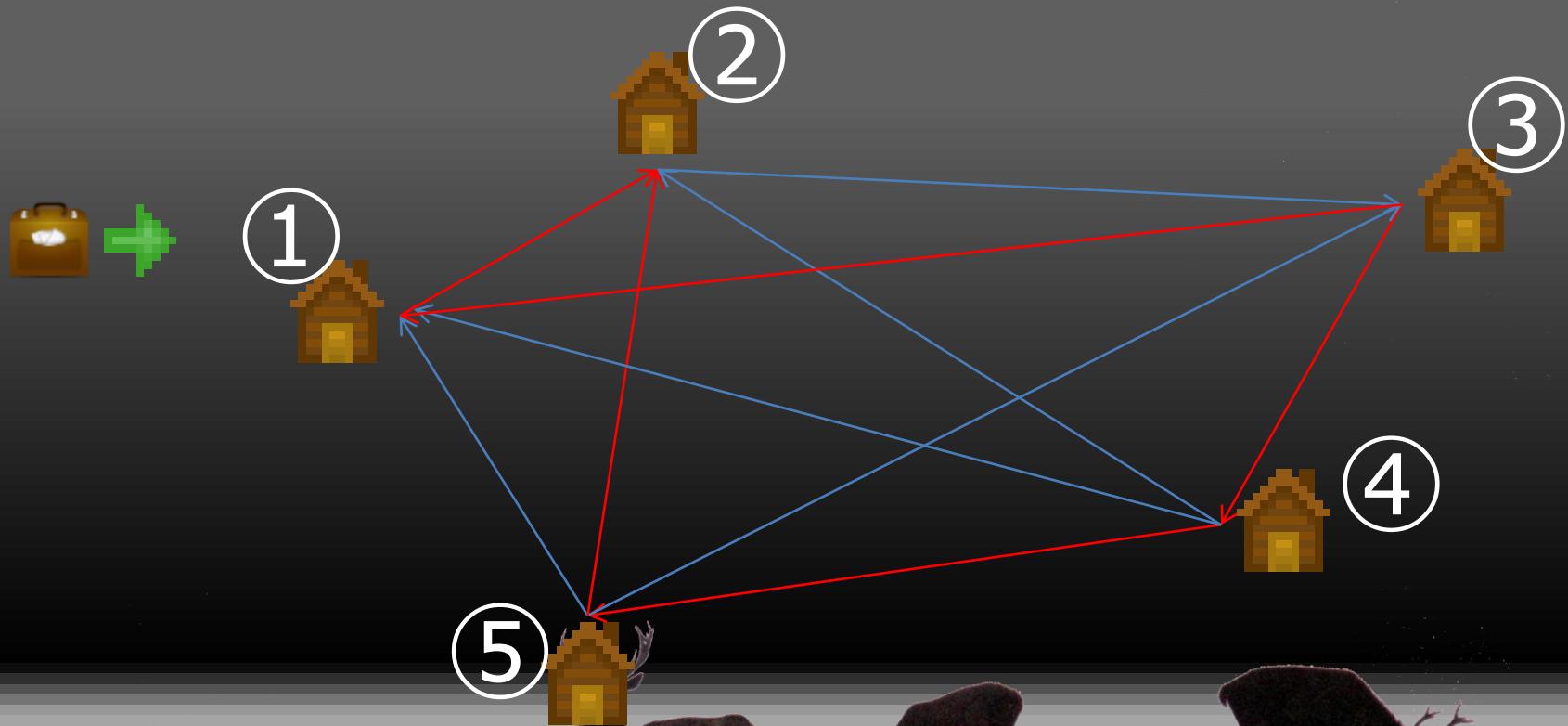
Giả sử có 5 thành phố thì có:
 $5 \times 4 \times 3 \times 2 \times 1$ cách tổ hợp,

Suy ra có 120 cách đi

Bài toán người bán hàng



Bài toán người bán hàng



Trước tiên đánh số thứ tự cho các thành phố

Bài toán người bán hàng



Tập cá thể ban đầu

Tạo tập cá thể ban đầu

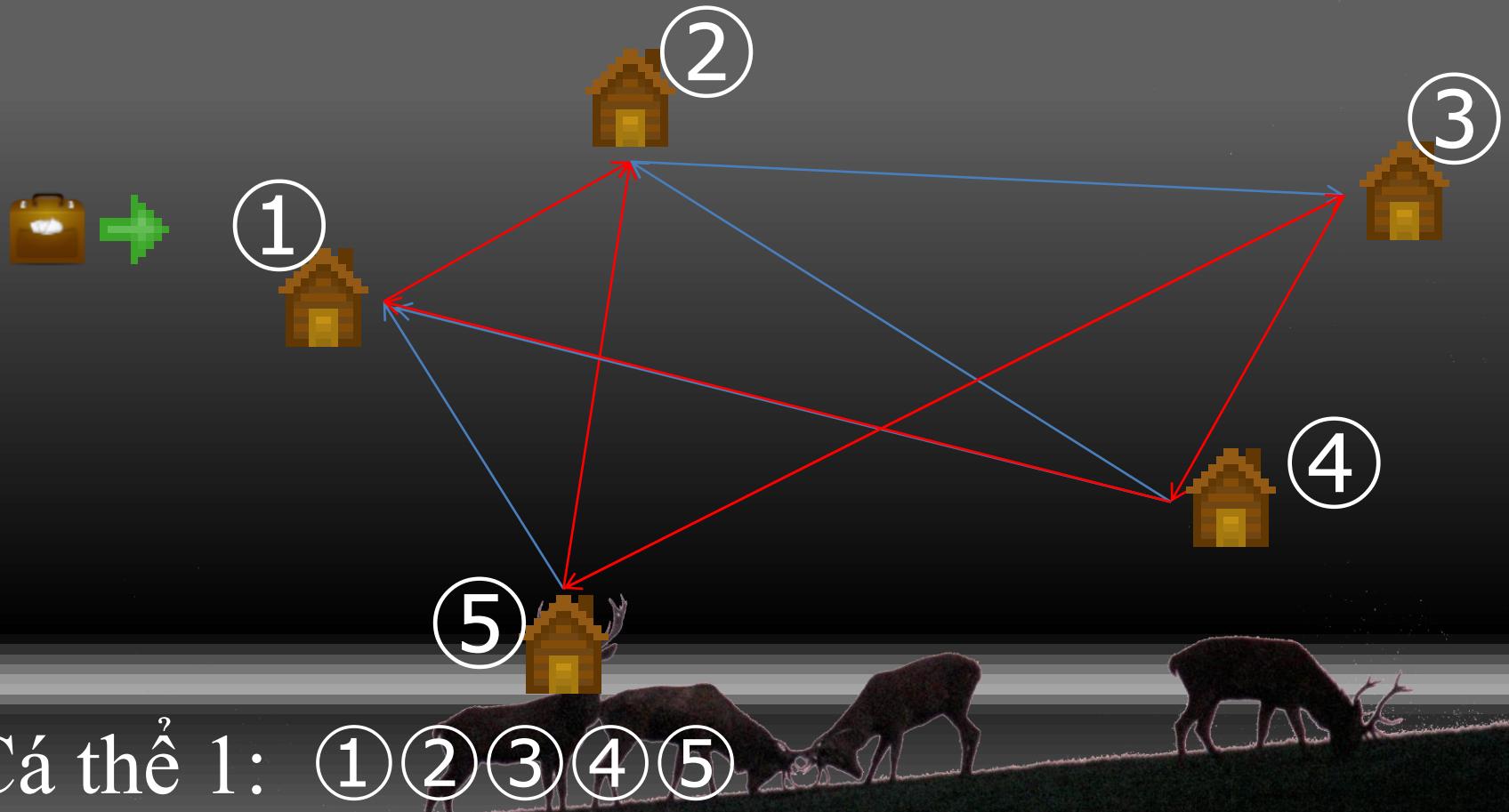
Giả sử có 3 cá thể ban đầu ngẫu nhiên
như sau:

Cá thể 1: ① ② ③ ④ ⑤

Cá thể 2: ① ⑤ ② ④ ③

Cá thể 3: ① ② ⑤ ④ ③

Tập cá thẻ ban đầu



Cá thẻ 1: ① ② ③ ④ ⑤

Cá thẻ 2: ① ⑤ ② ④ ③

Cá thẻ 3: ① ② ⑤ ③ ④

Tập cá thể ban đầu

Tạo gen cho các cá thể

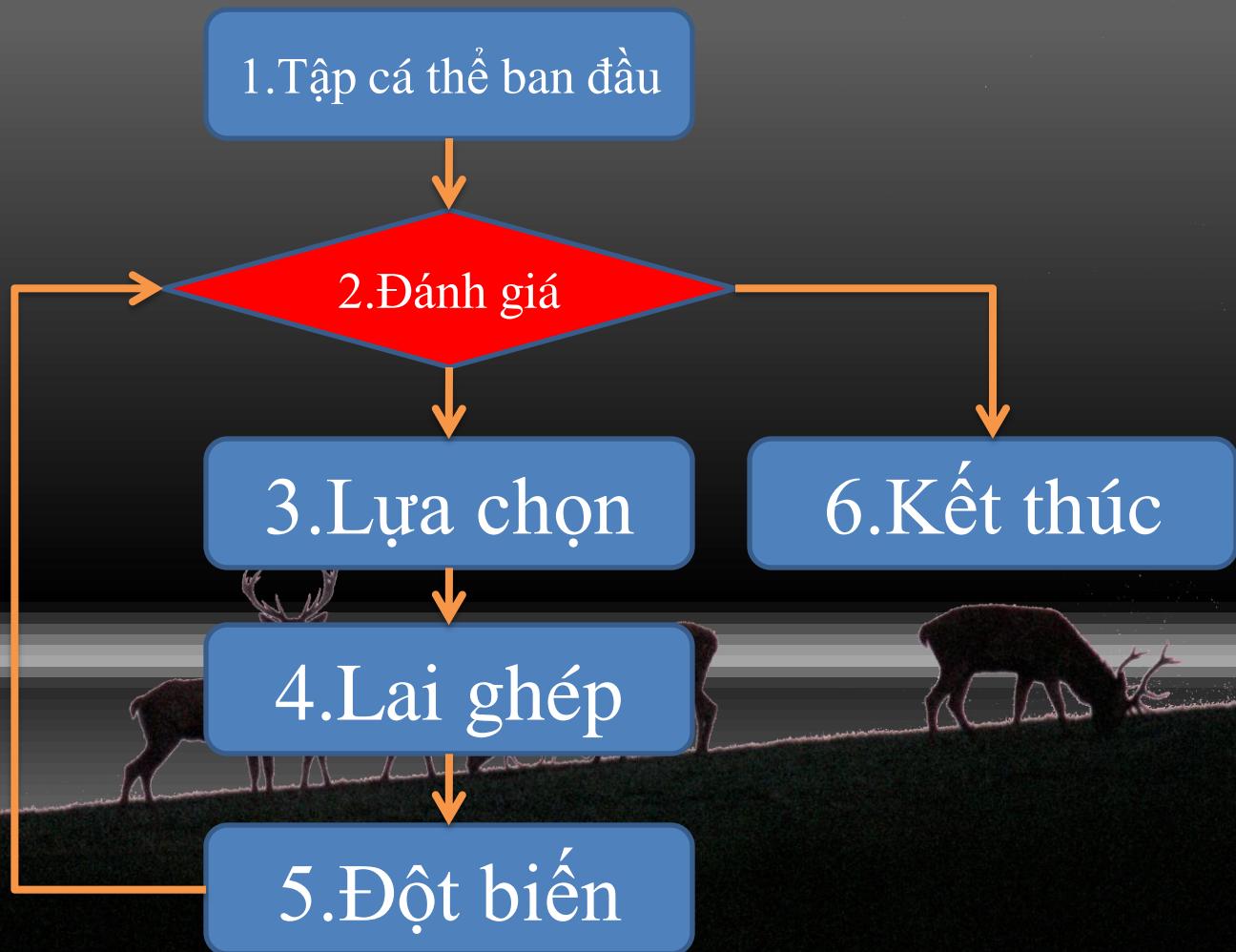
Cá thể 1: ①②③④⑤ → 00000

Cá thể 2: ①⑤②④③ → 03010

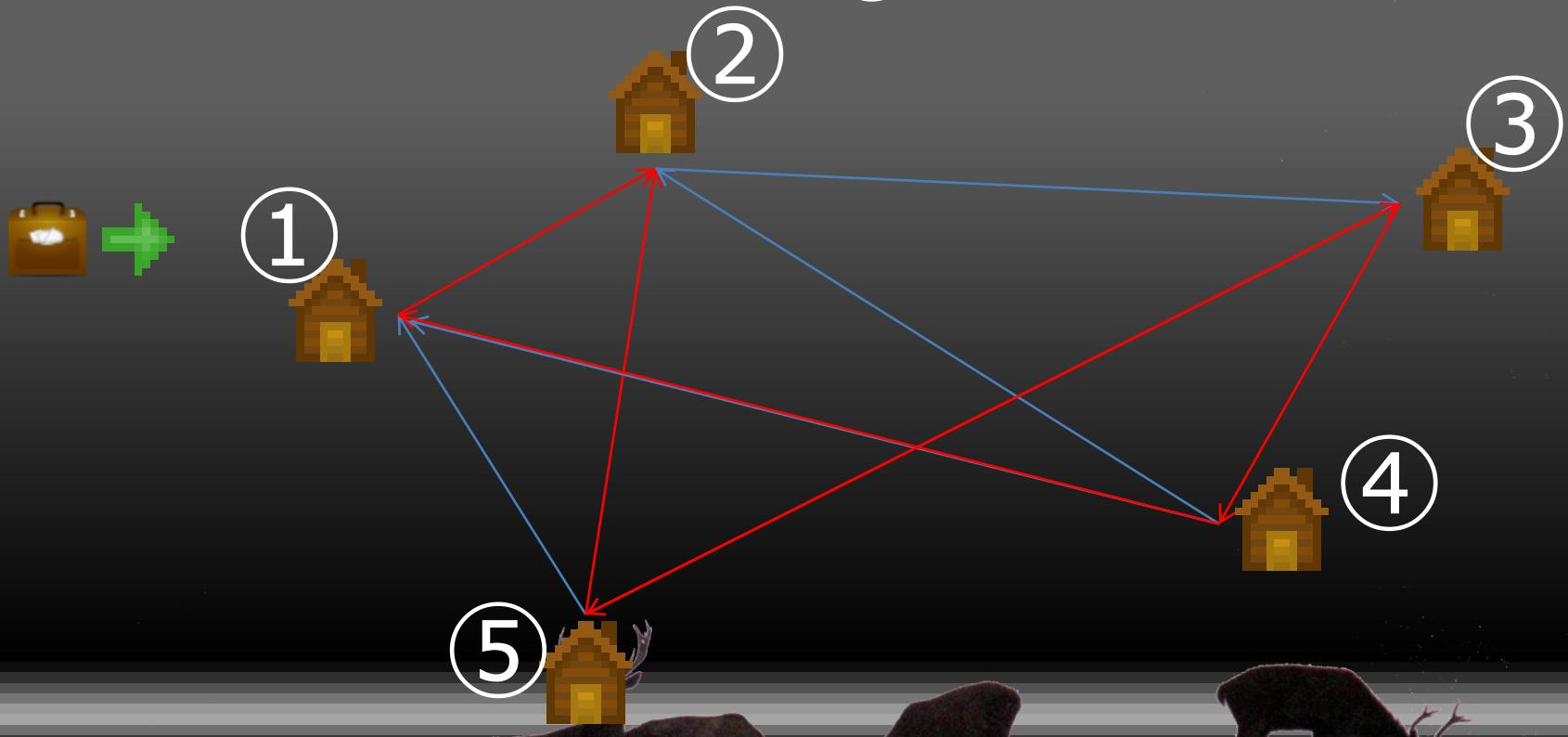
Cá thể 3: ①②⑤③④ → 00200

Lưu ý: Có nhiều cách tạo gen, đây chỉ là 1 ví dụ.

Bài toán người bán hàng

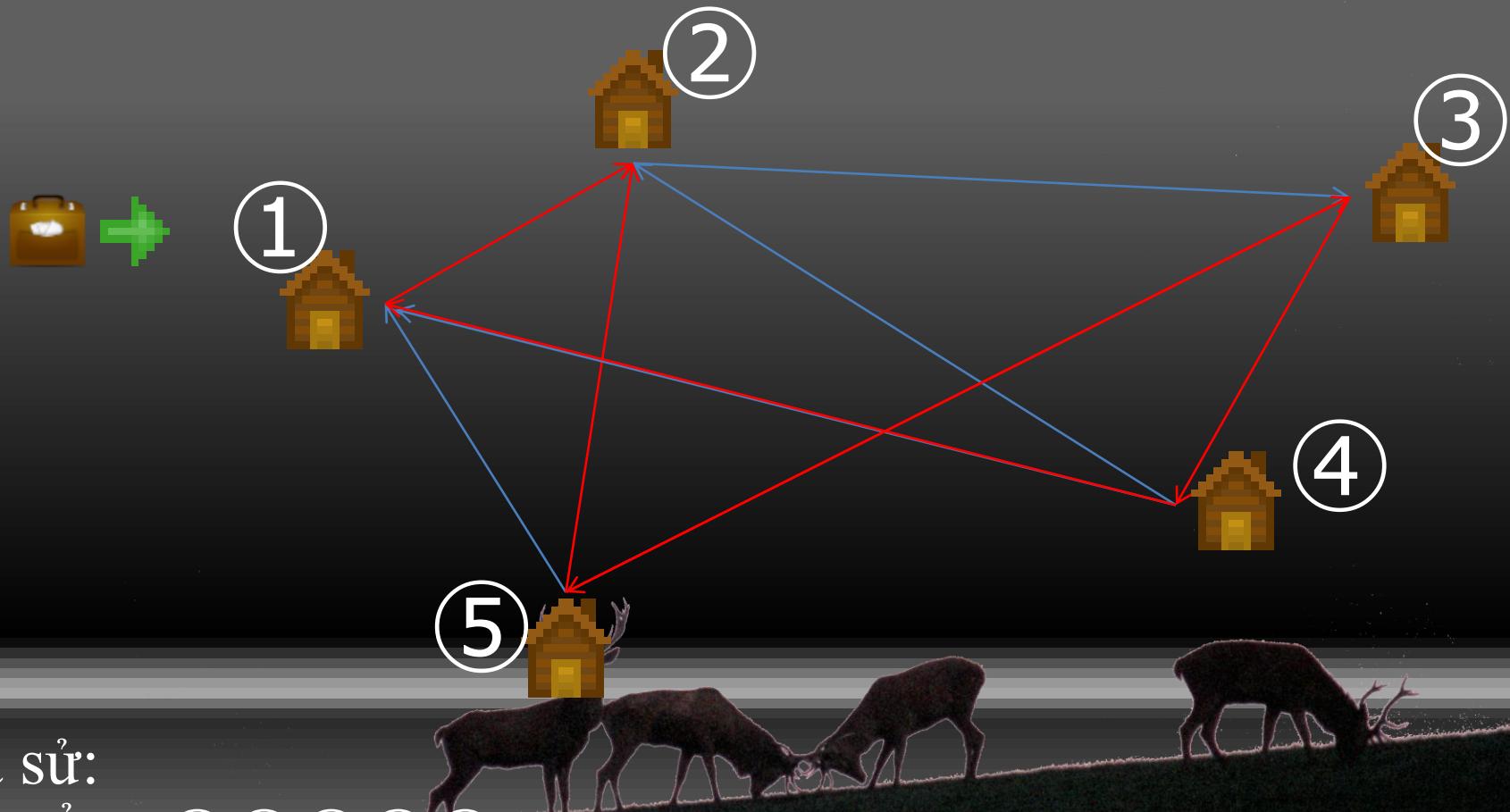


Đánh giá



Tiếp theo, đánh giá độ thích nghi của các cá thể
Cự ly ngắn → độ thích nghi cao
Cự ly dài → độ thích nghi thấp

Đánh giá



Giả sử:

Cá thể 1: ①②③④⑤ → 115km

Cá thể 2: ①⑤②④③ → 105km

Cá thể 3: ①②⑤③④ → 110km

Đánh giá

Cá thể 1: ①②③④⑤ → 115km

Cá thể 2: ①⑤②④③ → 105km

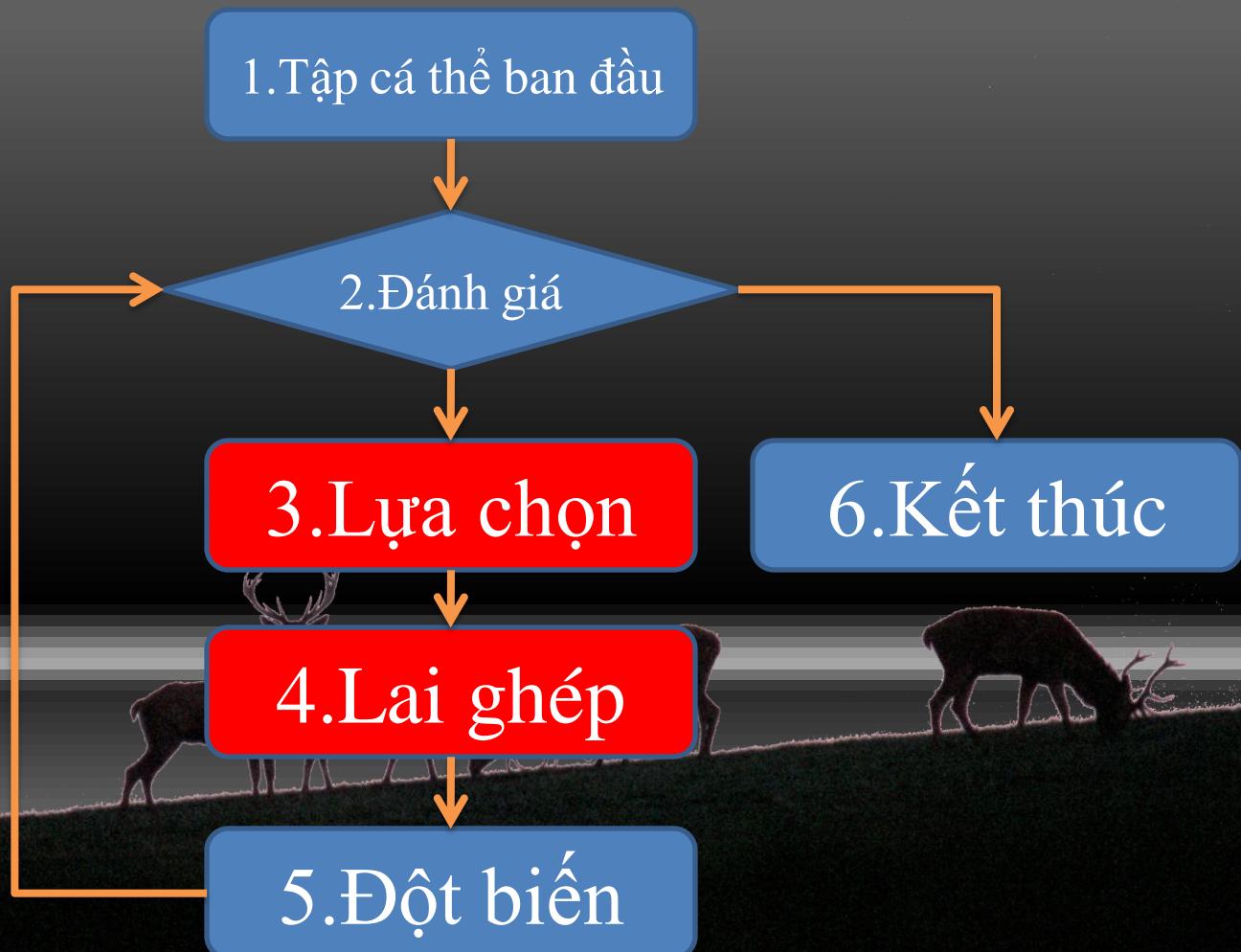
Cá thể 3: ①②⑤③④ → 110km

Vì vậy, cá thể 2 có độ thích nghi cao nhất

Genetic Algorithm



Bài toán người bán hàng



Lựa chọn và lai ghép

Thao tác trên gen của các cá thể

Cá thể 1: 00000 → 115km

Cá thể 1: 03010 → 105km

Cá thể 1: 00200 → 110km

Từ kết quả đánh giá

Ta **lựa chọn** các cá thể để **lai ghép**



Lựa chọn và lai ghép

Thao tác trên gen của các cá thể

Cá thể 1: 00000 → 115km

Cá thể 1: 03010 → 105km

Cá thể 1: 00200 → 110km

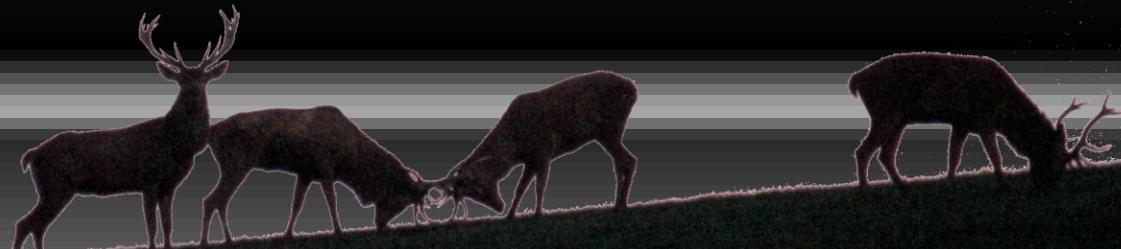
Vì có 3 cá thể cha nên ta tạo 3 cá thể con

Cá thể 2 có độ thích nghi cao nhất nên ưu tiên chọn làm cá thể để lai ghép

Lựa chọn

Có thể dùng hàm ngẫu nhiên để **lựa chọn** cá thể theo tiêu chí sau:

- + Cá thể có độ thích nghi cao thì có **xác suất** lựa chọn cao
- + Lựa chọn theo ranking



Lựa chọn

Cá thể 1: 00000 → 115km → $1/115 * 100 \rightarrow 0.87$

Cá thể 2: 03010 → 105km →
 $1/105 * 100 \rightarrow 0.95$

Cá thể 3: 00200 → 110km →
 $1/110 * 100 \rightarrow 0.91$

Lựa chọn

Cá thể 1: 00000 → 115km → 0.87

Cá thể 2: 03010 → 105km → 0.95

Cá thể 3: 00200 → 110km → 0.91

Tổng: $0.87 + 0.95 + 0.91 = 2.73$



Lựa chọn

Cá thể 1: 00000 → $0.87/2.73 = 31.9\%$

Cá thể 2: 03010 → $0.95/2.73 = 34.8\%$

Cá thể 3: 00200 → $0.91/2.73 = 33.3\%$

Ví dụ cá thể 2 và 3 được chọn để lai ghép

Lai ghép

Từ

Cá thể 1: 03010

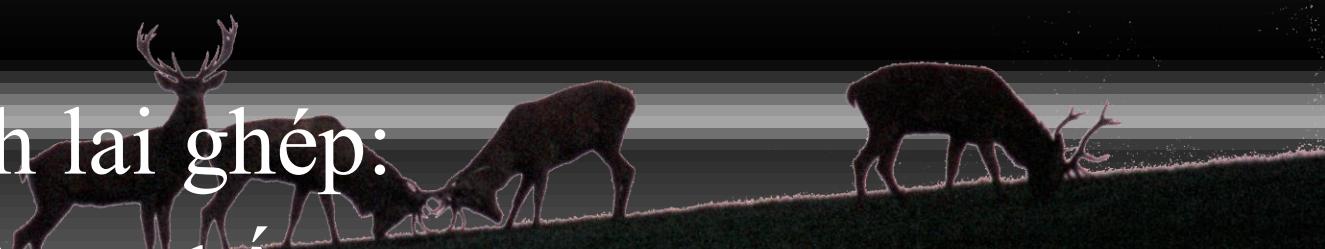
Cá thể 2: 00200

Ta tạo cá thể con bằng cách lai ghép

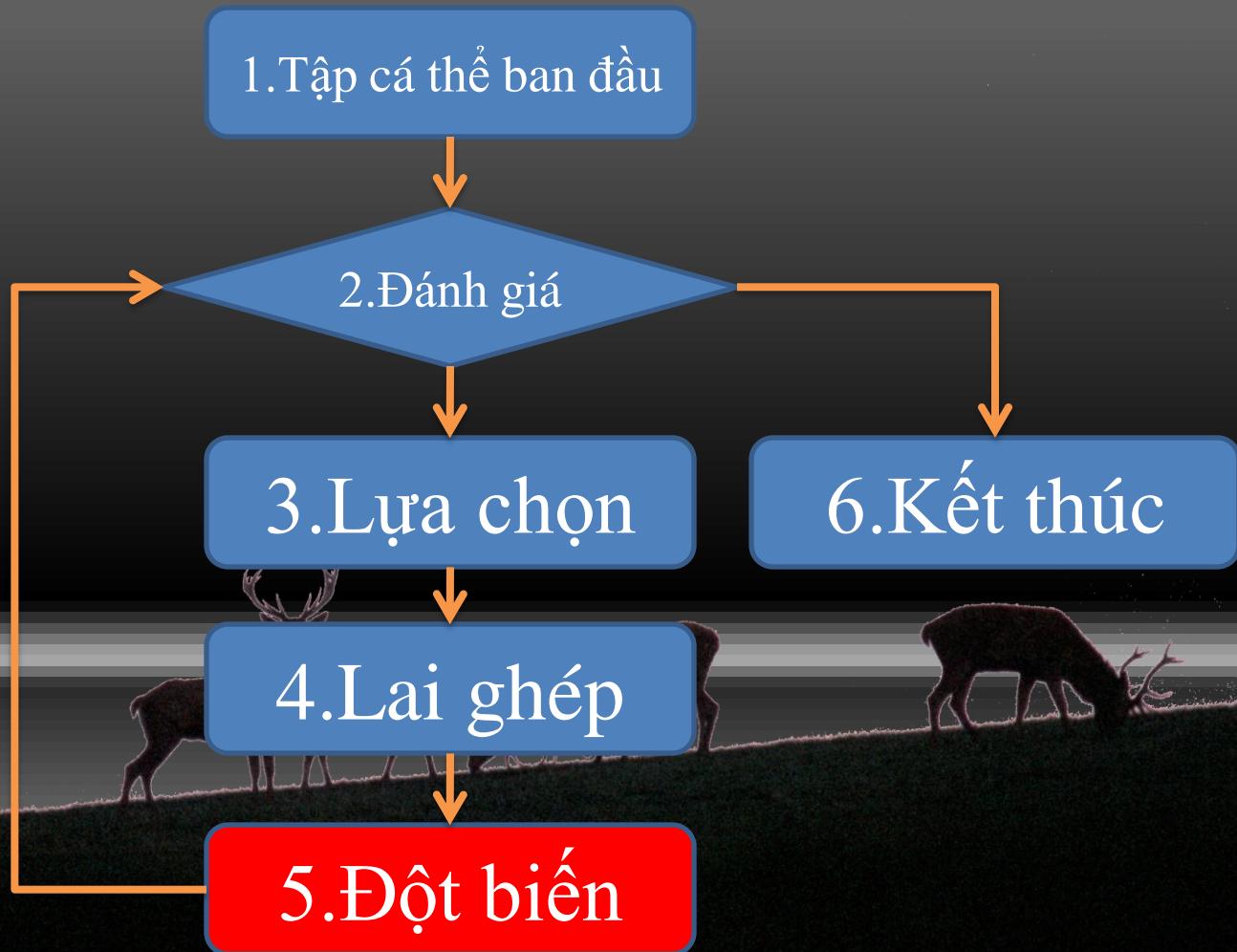
Có nhiều cách lai ghép:

Lai ghép đồng nhất,

Lai ghép 1 điểm...



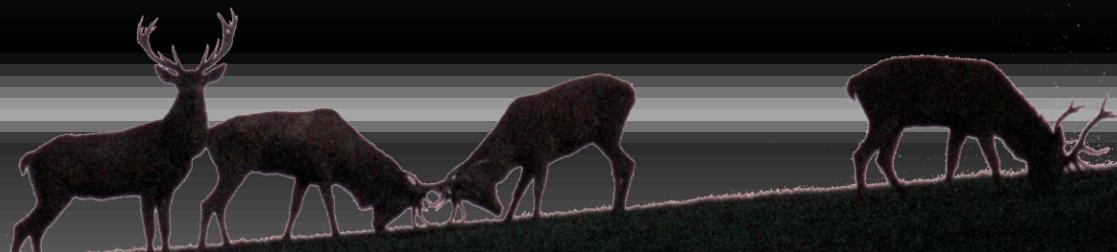
Bài toán người bán hàng



Đột biến

Khi thế hệ con được sinh ra, có những tính chất con khác với mẹ do **đột biến**

Có thể tạo đột biến cho gen bằng ngẫu nhiên



Đột biến

Giả sử xác suất đột biến của phần tử là 5%:

00210

03000

03000

Tổng phần tử của gen trong quần thể là 15 nên:

Xác suất đột biến của quần thể là

$$5 * 15 / 100 = 0.75$$

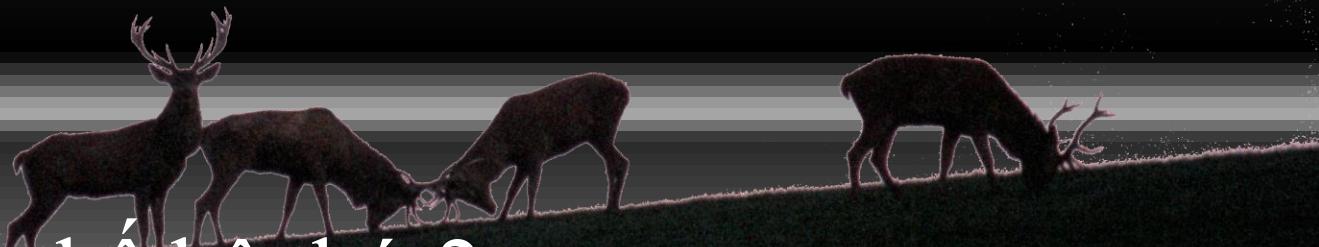
Đột biến

Ví dụ về đột biến:

00210

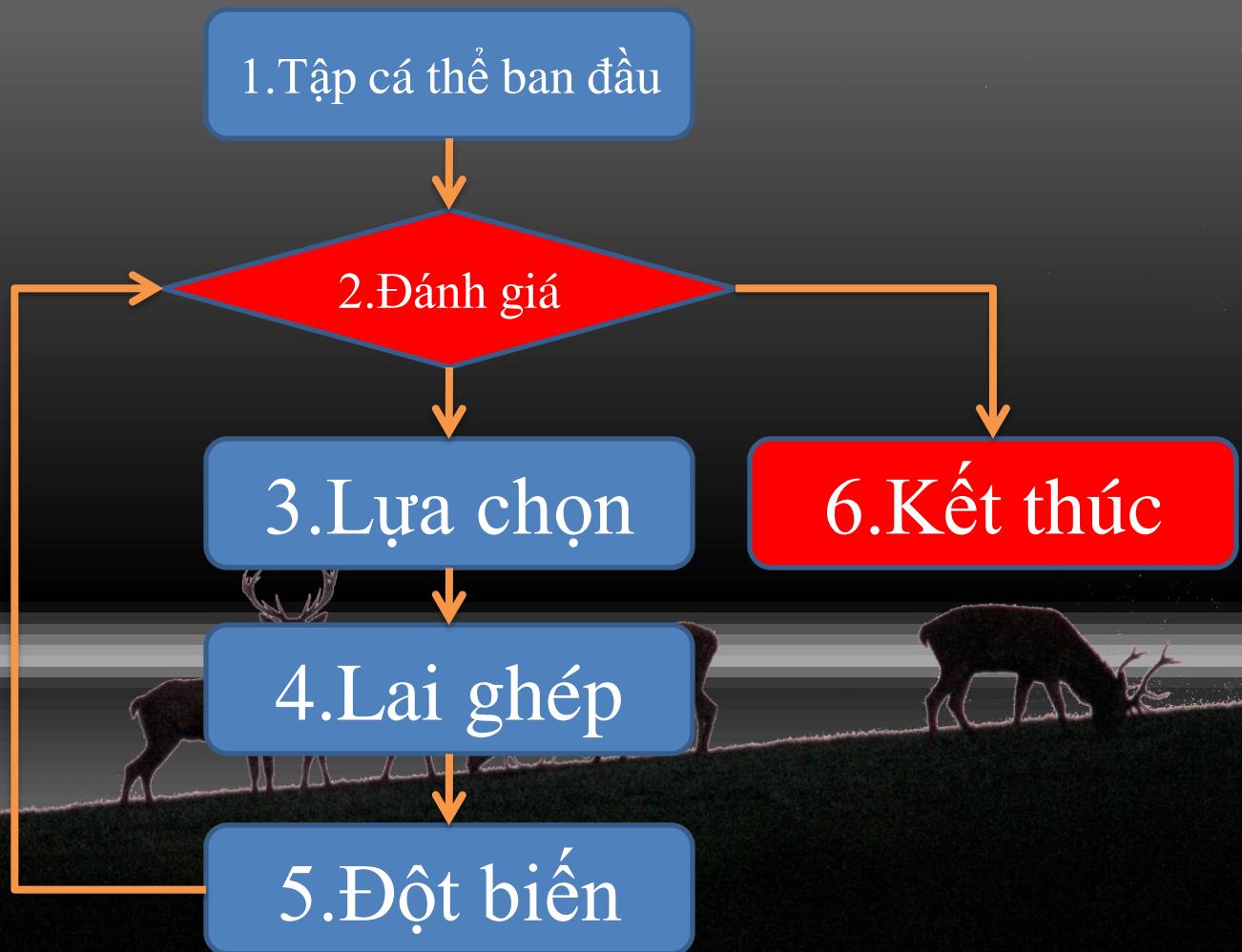
03000

030**1**0



Có đột biến ở thế hệ thứ 2.

Bài toán người bán hàng



Chẳng hiểu gì cả...
Hic hic ?

Vậy thì,
Demo thử nào!!!

Thuật toán di truyền

- Các bước học tập mô phỏng theo **cơ chế di truyền của sinh vật**
- Được ứng dụng rộng rãi
trong các bài toán tối ưu

Bài toán người bán hàng(32 TP)

Có tất cả

$31! / 2$ cách tổ hợp.

$$31! = (31 \times 30 \times 29 \times \dots \dots \times 4 \times 3 \times 2 \times 1) / 2$$

$\approx 4.11 \times 10^{33}$ cách.

Sử dụng

Super Computer

cũng cần tới

1.300.000.000.000.000

Năm

Genetic Algorithm



Môn học

Trí Tuệ Nhân Tạo

Giảng Viên:

Phạm Minh Tuấn



Chương 9:

Mạng Nơ Ron nhân tạo

(Artificial neural network)



Nội dung

- ❖ Introduction (Giới thiệu)
- ❖ Steepest descent method (Rơi dốc nhanh nhất)
- ❖ Least squares method (Bình phương tối thiểu)
- ❖ Perceptron
- ❖ Adaptive Linear Neuron
- ❖ Logistic regression
- ❖ Multi-Layer Perceptron



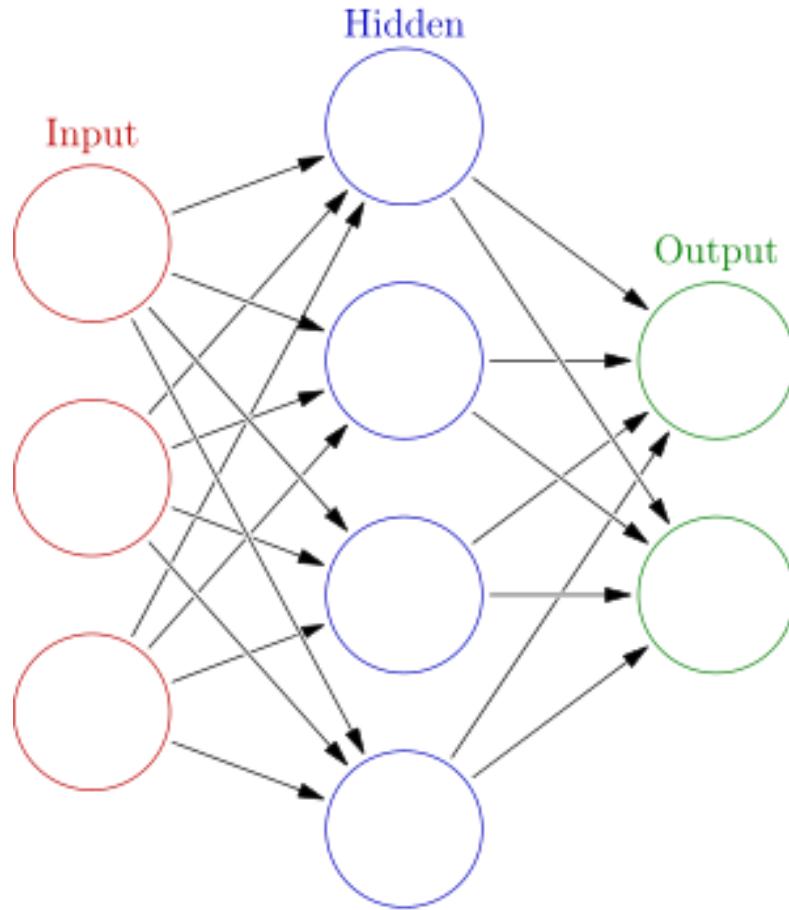
Introduction

❖ Artificial neural networks:

- Computational models inspired by animal central nervous systems (in particular the brain) that are capable of machine learning and pattern recognition.
- presented as systems of interconnected "neurons" that can compute values from inputs by feeding information through the network.



Example



- + An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain.
- + Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.



Real-life applications

- ❖ Function approximation, or regression analysis
- ❖ Classification
- ❖ Data processing
- ❖ Robotics
- ❖ Control



Artificial neural network

- ❖ An ANN is typically defined by three types of parameters:
 - **The interconnection pattern** between different layers of neurons
 - **The learning process** for updating the weights of the interconnections
 - **The activation function** that converts a neuron's weighted input to its output activation.



The biggest problem is how to learn?



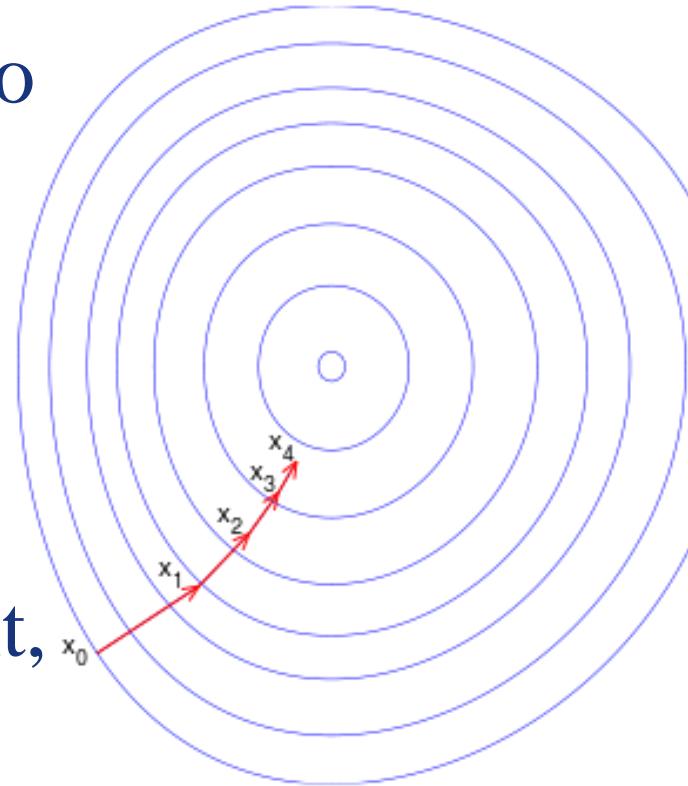
Rơi dốc nhanh nhất

Steepest descent method



Steepest descent method

- ❖ Steepest descent method is also known as Gradient descent
- ❖ It is a first-order optimization algorithm
- ❖ To find a local minimum of a function using gradient descent, one takes steps proportional to the **negative** of the gradient of the function at the current point.

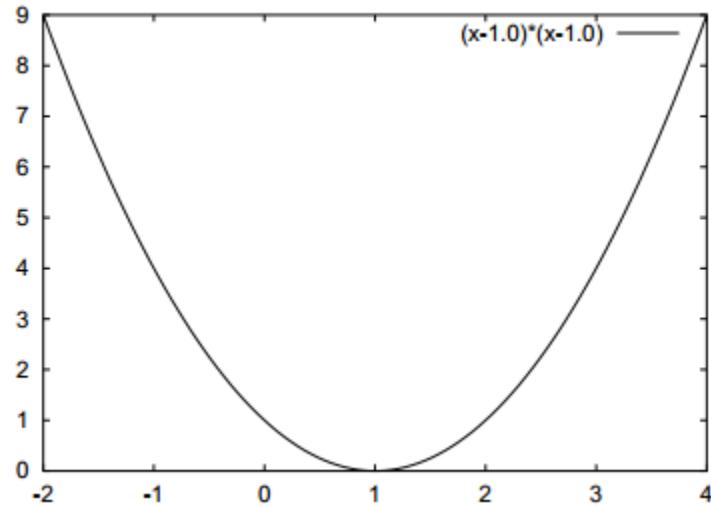




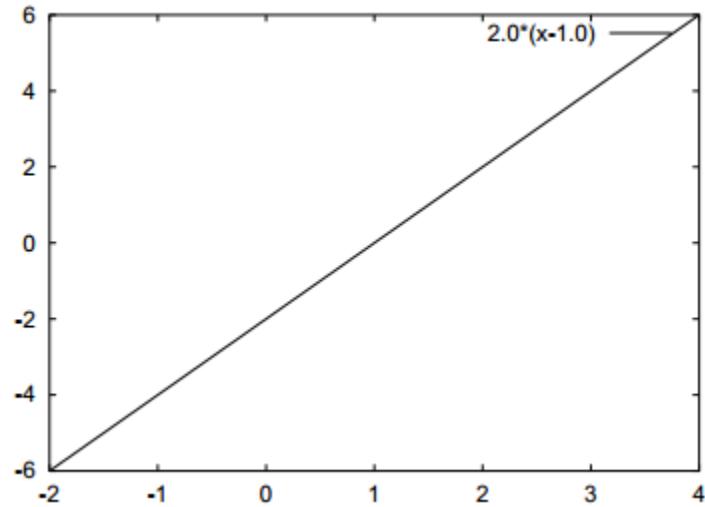
Example

- ❖ Gradient descent has problems such as a function shown here.

$$f(a) = (a - 1.0)^2$$



$$\frac{\partial f(a)}{\partial a} = 2(a - 1.0)$$





Algorithm

- ❖ To find a local minimum of a function using gradient descent, one takes steps proportional to the **negative** of the gradient of the function at the current point.
- ❖ Update parameter:

$$a^{(k+1)} = a^{(k)} - \alpha \frac{\partial f(a)}{\partial a} \Big|_{a=a^{(k)}}$$

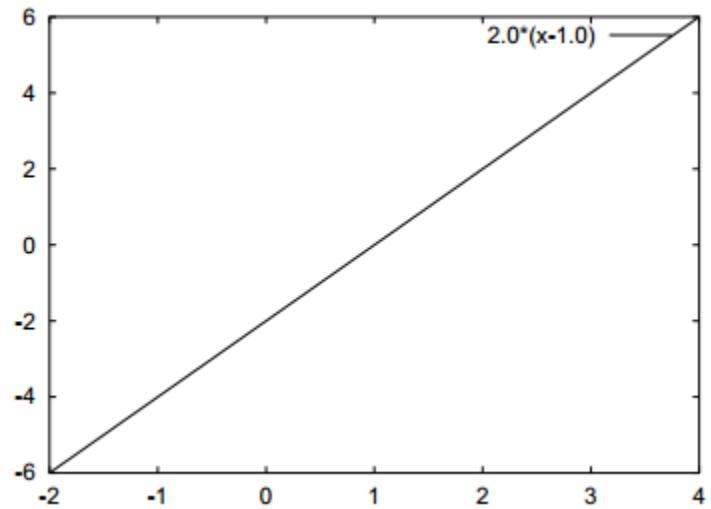
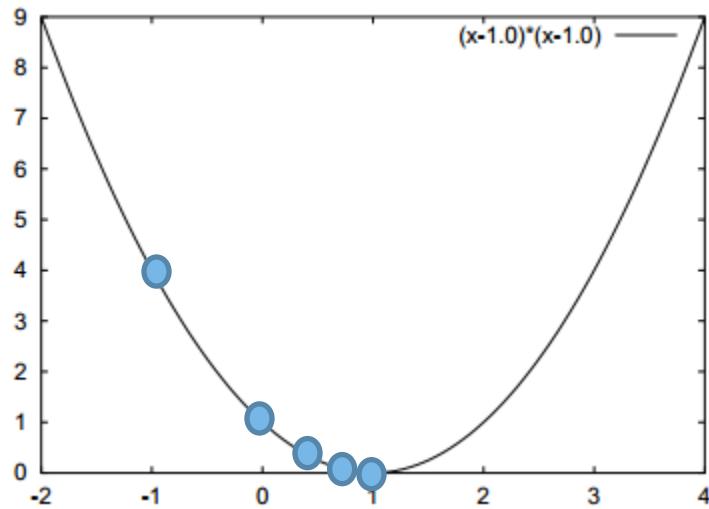


Example

$$a^{(k+1)} = a^{(k)} - 2\alpha(a^{(k)} - 1.0)$$

0.25







Program to find the optimum value

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double f(double a) {return((a-1.0)*(a-1.0));}
double df(double a) {return(2.0*(a-1.0));}
main() {
    double a;
    int i;
    double alpha = 0.1; /* Learning Rate */
    /* set the initial value of a by random number within [-50.0:50.0] */
    a = 100.0 * (drand48() - 0.5);
    printf("Value of a at Step 0 is %f, ", a);
    printf("Value of f(a) is %f\n", f(a));
    for (i = 1; i < 100; i++) {
        /* update theta by steepest decent method */
        a = a - alpha * df(a);
        printf("Value of a at Step %d is %f, ", i, a);
        printf("Value of f(a) is %f\n", f(a)); }
}
```



Results

Value of a at Step 0 is -50.000000, Value of f(a) is 2601.000000
Value of a at Step 1 is -39.800000, Value of f(a) is 1664.640000
Value of a at Step 2 is -31.640000, Value of f(a) is 1065.369600
Value of a at Step 3 is -25.112000, Value of f(a) is 681.836544
Value of a at Step 4 is -19.889600, Value of f(a) is 436.375388
Value of a at Step 5 is -15.711680, Value of f(a) is 279.280248
Value of a at Step 6 is -12.369344, Value of f(a) is 178.739359
Value of a at Step 7 is -9.695475, Value of f(a) is 114.393190
Value of a at Step 8 is -7.556380, Value of f(a) is 73.211641
Value of a at Step 9 is -5.845104, Value of f(a) is 46.855451
Value of a at Step 10 is -4.476083, Value of f(a) is 29.987488
Value of a at Step 11 is -3.380867, Value of f(a) is 19.191993
Value of a at Step 12 is -2.504693, Value of f(a) is 12.282875
Value of a at Step 13 is -1.803755, Value of f(a) is 7.861040
Value of a at Step 14 is -1.243004, Value of f(a) is 5.031066
Value of a at Step 15 is -0.794403, Value of f(a) is 3.219882
Value of a at Step 16 is -0.435522, Value of f(a) is 2.060725
Value of a at Step 17 is -0.148418, Value of f(a) is 1.318864



Results

Value of a at Step 76 is 0.999998, Value of f(a) is 0.000000

Value of a at Step 77 is 0.999998, Value of f(a) is 0.000000

Value of a at Step 78 is 0.999999, Value of f(a) is 0.000000

Value of a at Step 79 is 0.999999, Value of f(a) is 0.000000

Value of a at Step 80 is 0.999999, Value of f(a) is 0.000000

Value of a at Step 81 is 0.999999, Value of f(a) is 0.000000

Value of a at Step 82 is 0.999999, Value of f(a) is 0.000000

Value of a at Step 83 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 84 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 85 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 86 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 87 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 88 is 1.000000, Value of f(a) is 0.000000

Value of a at Step 89 is 1.000000, Value of f(a) is 0.000000



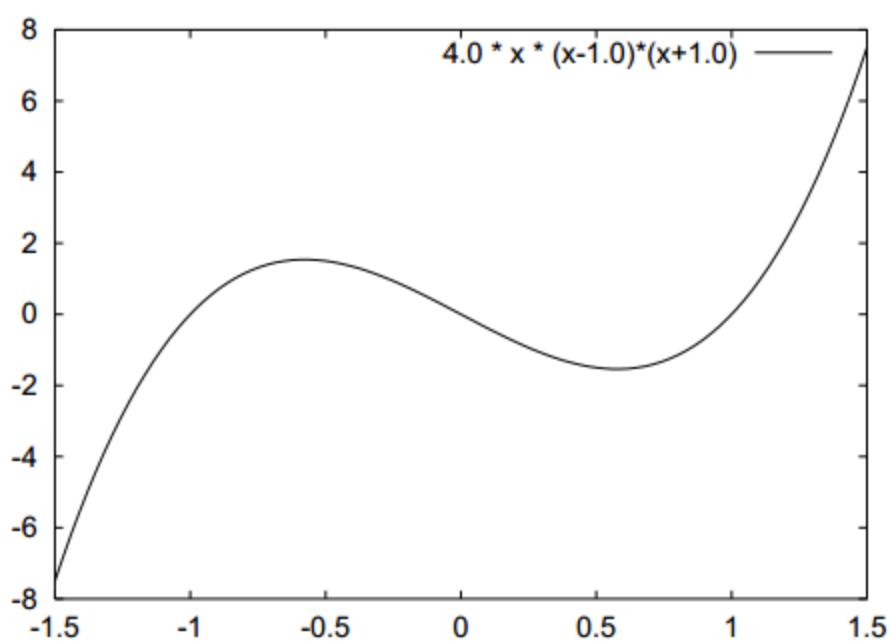
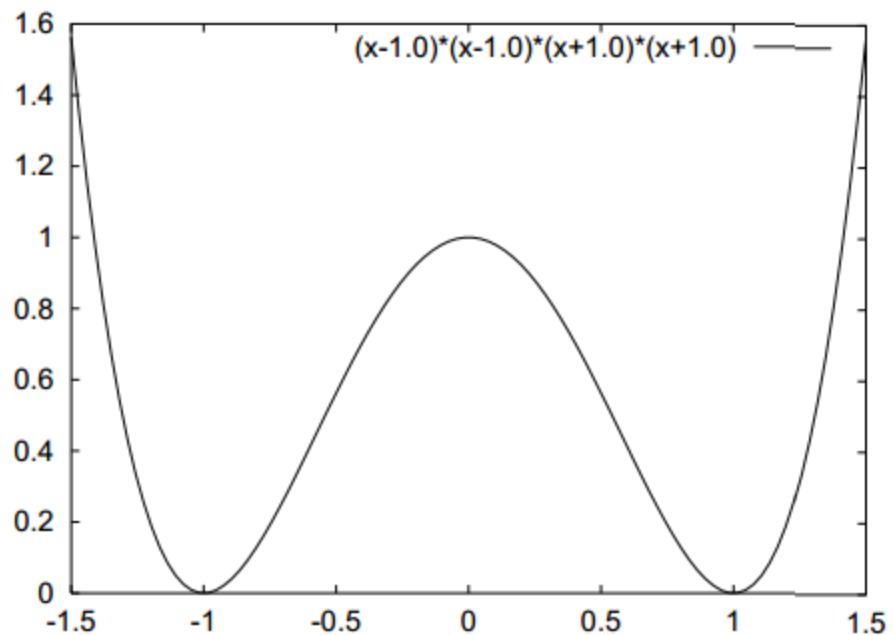
Problem 1.

- ❖ Optimize the next function:

$$f(a) = (a - 1.0)^2(a + 1.0)^2$$



I T F





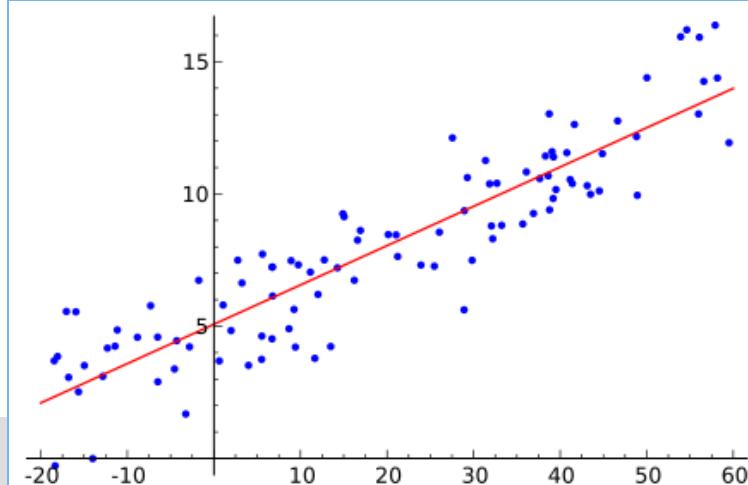
Bình phương tối thiểu

Least squares method



Least squares method

- ❖ The method of least squares is a standard approach to the approximate solution of over determined systems.
- ❖ "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation.





Data

STT	Kỷ lục ném xa (m)	Lực nắm	Chiều cao	Cân nặng
1	22	28	146	34
2	36	46	169	57
3	24	39	160	48
4	22	25	156	38
5	27	34	161	47
6	29	29	168	50
7	26	38	154	54
8	23	23	153	40
9	31	42	160	62
10	24	27	152	39
11	23	35	155	46
12	27	39	154	54
13	31	38	157	57
14	25	32	162	53
15	23	25	142	32



Problem 2.

- ❖ How to predict the record(t) based on grip strength (x1), height(x2) and weight(x3)?
- ❖ Prediction Equation:

$$y(x_1, x_2, x_3) = a_0 + a_1x_1 + a_2x_2 + a_3x_3$$



Least squares method

- ❖ The least squares method finds its optimum when the sum of squared residuals is a minimum:

$$\begin{aligned}\varepsilon^2(a_0, a_1, a_2, a_3) &= \frac{1}{15} \sum_{l=1}^{15} \varepsilon_l^2 = \frac{1}{15} \sum_{l=1}^{15} (t_l - y_l)^2 \\ &= \frac{1}{15} \sum_{l=1}^{15} \{t_l - (a_0 + a_1 x_{1l} + a_2 x_{2l} + a_3 x_{3l})\}^2\end{aligned}$$



❖ How to optimize?

- By Least squares method
- By Steepest descent method

❖ Look at black board!!!!



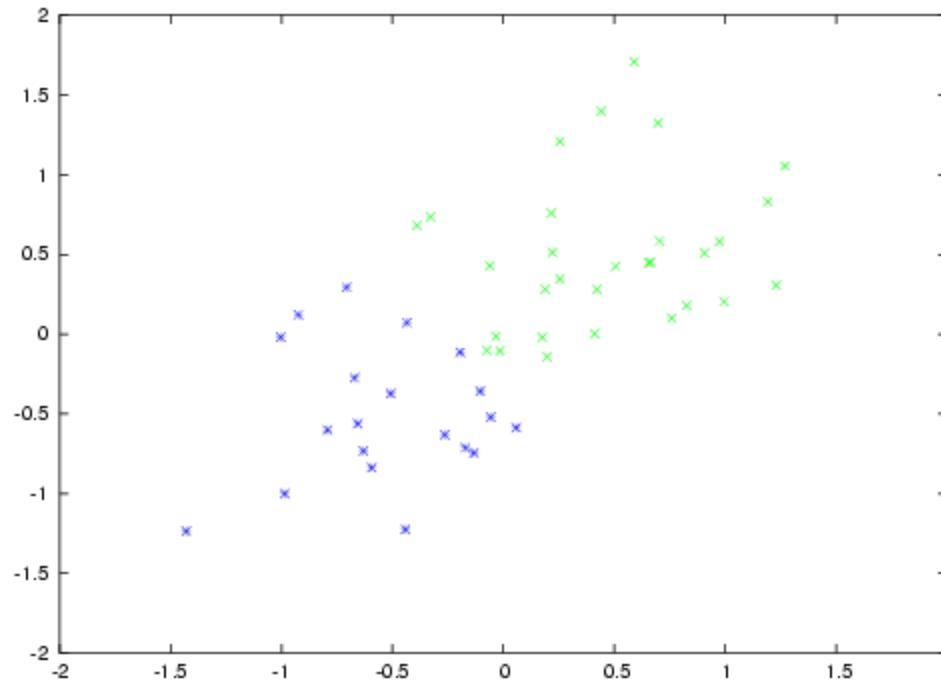
I T F

Perceptron



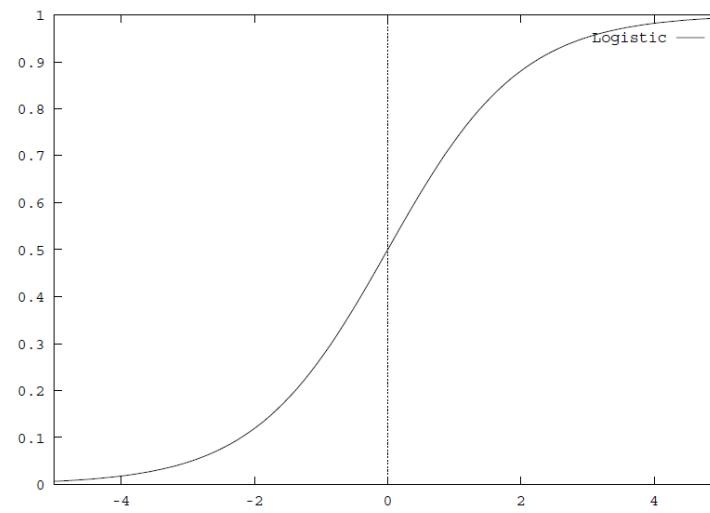
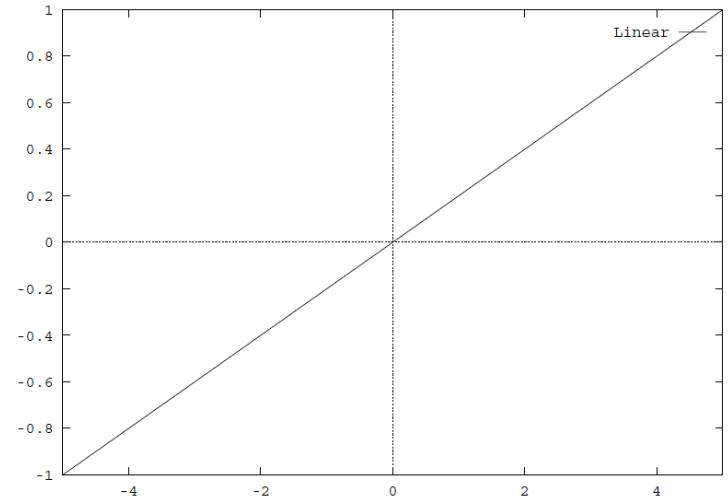
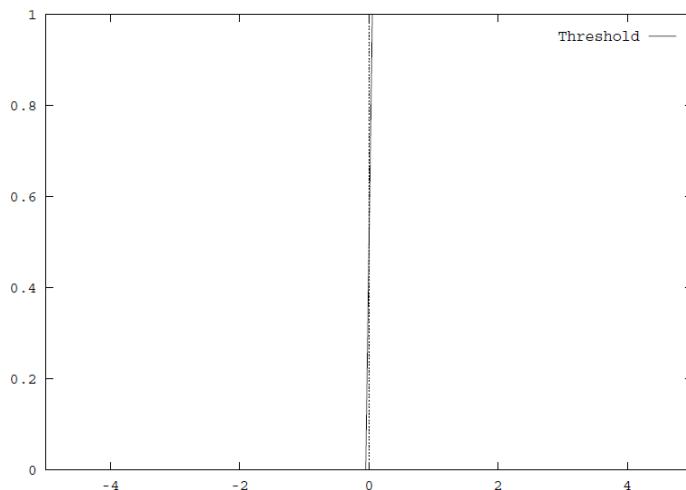
Perceptron

- ❖ The perceptron is an algorithm for supervised classification of an input into one of several possible non-binary outputs.





Non-binary output function





Definition

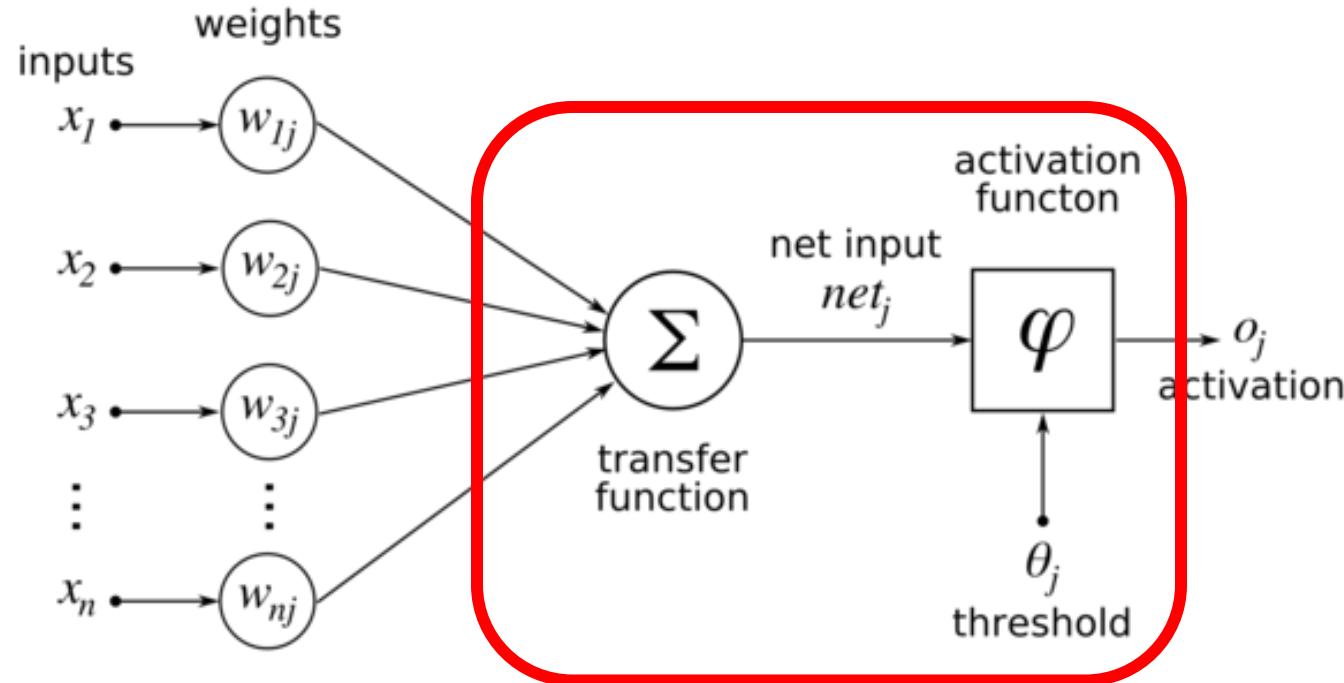
- ❖ The perceptron is a binary classifier which maps its input x (a real-valued vector) to an output value $f(x)$ (a single binary value):

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- ❖ w is a vector of real-valued weights,
- ❖ b is the “bias”



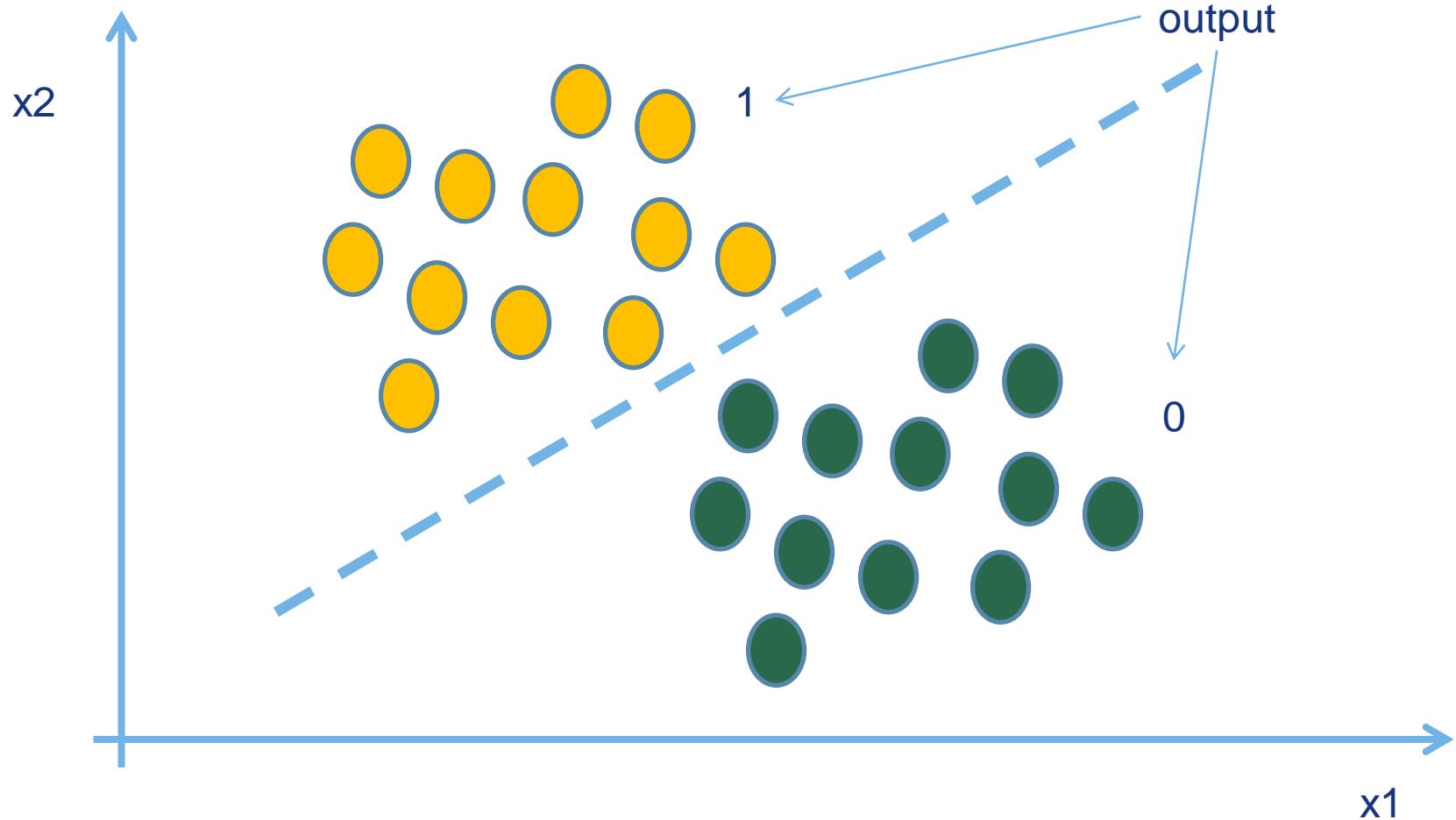
Example



$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



Classification





I T F

Adaptive Linear Neuron

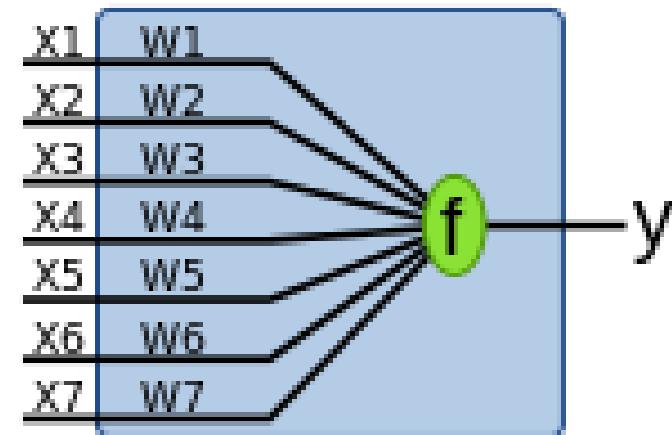
Learning Algorithm

1. Initialize the weights and the threshold.
2. For each example, perform the following steps over the input, and desired output:
 1. Calculate the actual output:

$$o = \mathbf{w} \cdot \mathbf{x}_i + b$$

2. Update the weights:

$$\mathbf{w}' = \mathbf{w} + \alpha(y_i - o)\mathbf{x}_i$$



3. Repeat Step 2



The algorithm based on Steepest descent method.

Why?



Optimization problem

- ❖ Optimization problem is:

$$E = (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 \rightarrow \min$$

- ❖ Steepest descent method:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \frac{dE}{d\mathbf{w}}(\mathbf{w}^k)$$

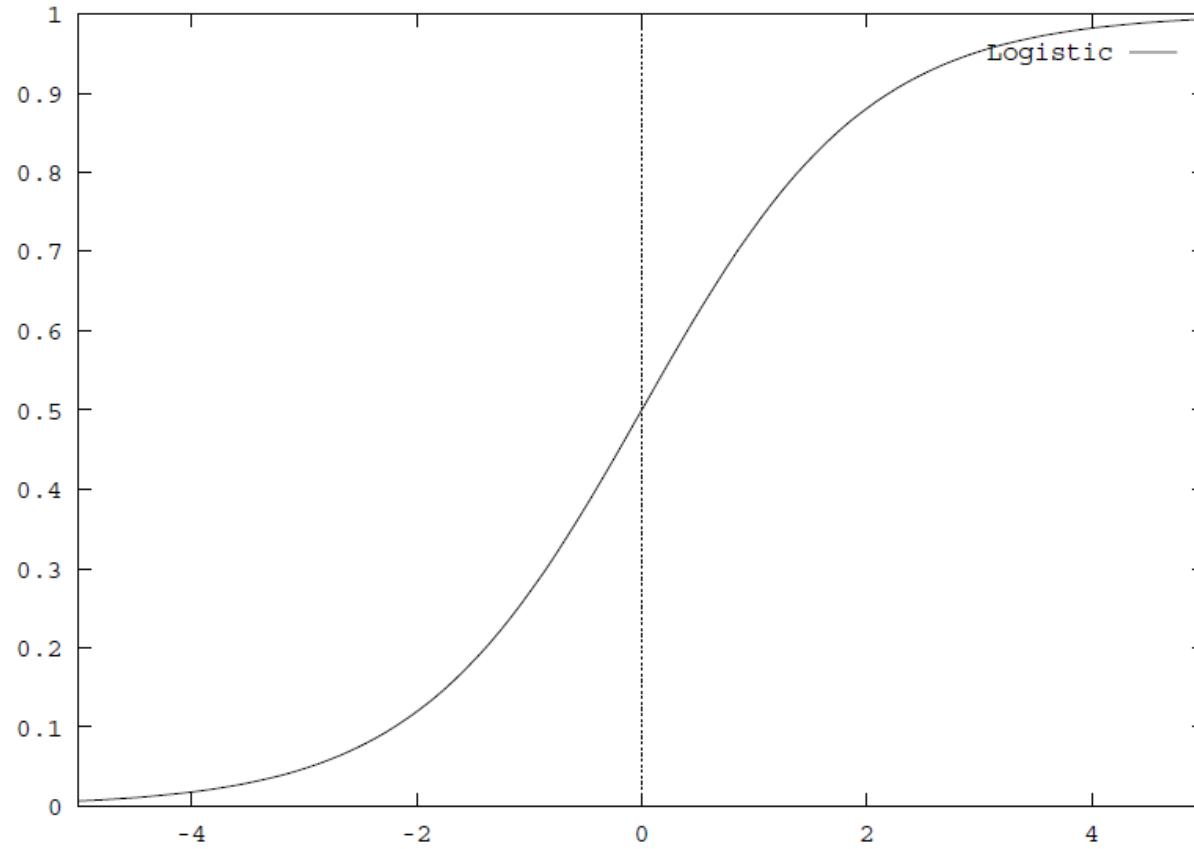


I T F

Logistic regression



Logistic function





Logistic function

- ❖ Logistic function:

$$f(\text{net}) = \frac{\exp(\text{net})}{1 + \exp(\text{net})}$$

- ❖ Output function:

$$f(\mathbf{x}) = \frac{\exp(\mathbf{w} \cdot \mathbf{x} + b)}{1 + \exp(\mathbf{w} \cdot \mathbf{x} + b)}$$



Optimization problem

- ❖ Optimization problem is:

$$E = \left(y_i - \frac{\exp(\mathbf{w} \cdot \mathbf{x}_i + b)}{1 + \exp(\mathbf{w} \cdot \mathbf{x}_i + b)} \right)^2 \rightarrow \min$$

- ❖ Steepest descent method:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \frac{dE}{d\mathbf{w}}(\mathbf{w}^k)$$



Algorithm

1. Calculate the actual output:

$$o = f(\mathbf{x}_i) = \frac{\exp(\mathbf{w} \cdot \mathbf{x}_i + b)}{1 + \exp(\mathbf{w} \cdot \mathbf{x}_i + b)}$$

2. Update the weights:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \alpha o(1 - o)(y_i - o)\mathbf{x}_i$$



Classification Program

2

2

1

1

2-classes classification problem

Perceptron with Java Program

```
public class Perceptron {  
    int w,h;  
    double[] weight;  
    double alpha=0.5;  
    double loop=100;  
  
    public Perceptron(int w,int h){  
        this.w=w;  
        this.h=h;  
        this.weight=new double[w*h+1];  
        Random rand=new Random();  
        for (int i=0;i<weight.length;i++)  
            weight[i]=rand.nextDouble()-0.5;  
    }  
}
```



Output

```
public double output(BufferedImage image)
{
    double sum=weight[w*h];
    for (int i=0;i<w;i++)
    for (int j=0;j<h;j++)
        if (image.getRGB(i, j)==Color.BLACK.getRGB())
            sum+=weight[i*h+j];
    return 1.0/(1.0+Math.exp(-sum/(1000)));
}//output
```



Learning

```
public void learning(BufferedImage image, int y){  
    for(int l=0;l<loop;l++){  
        double o=output(image);  
        double d=alpha*o*(1-o)*(y-o);  
        for (int i=0;i<w;i++)  
            for (int j=0;j<h;j++)  
                if (image.getRGB(i, j)==Color.BLACK.getRGB())  
                    weight[i*h+j]=weight[i*h+j]+d;  
        weight[w*h]=weight[w*h]+d;  
    }//learning  
  
}//Perceptron
```



Experiments

