Hệ chuyên gia (Expert System)

PGS.TS. Phan Huy Khánh

khanhph@vnn.vn

Chương 1

Mở đầu về hệ chuyên gia

1.2

Hệ chuyên gia (HCG) là gì ?

- Hệ chuyên gia là lĩnh vực ứng dụng của *trí tuệ nhân tạo*
- Có nhiều cách định nghĩa HCG :
  - E. Feigenbaum : «*Hệ chuyên gia (Expert System) là một chương trình máy tính thông minh sử dụng tri thức (knowledge) và các thủ tục suy luận (inference procedures) để giải những bài toán tương đối khó khăn đòi hỏi những chuyên gia mới giải được*»
  - Hệ chuyên gia là một hệ thống tin học có thể mô phỏng (emulates) năng lực quyết đoán (decision) và hành động (making abilily) của một chuyên gia (con người)
  - Một hệ chuyên gia sử dụng tri thức của một lĩnh vực cụ thể để cung cấp việc giải quyết vấn đề với "chất lượng chuyên gia" trong lĩnh vực đó.

What is an Expert System?

- Expert System:
  A computer program that:
  - Attempts to code the knowledge of human experts in the form of heuristics (i.E. A rule of thumb)
  - Emulates the decision-making ability of a human expert in a restricted domain
  - Edward Feigenbaum: "An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solutions"
- A computer program which:
  - Encapsulates knowledge from some domain, normally obtained from a human expert in that domain

Khái niệm chuyên gia trong cuộc sống

- Trong cuộc sống, các chuyên gia có thể giải quyết vấn đề ở một mức độ cao vì họ có rất nhiều tri thức về lĩnh vực họ hoạt động
- Những tri thức này bao gồm lý thuyết đến cả các kinh nghiệm, kỹ xảo, phương pháp làm tắt, chiến lược heuristic đã tích lũy được của các chuyên gia con người qua quá trình làm việc của họ trong một lĩnh vực chuyên môn
- Từ tri thức này, người ta cố gắng cài đặt chúng vào hệ thống để hệ thống có thể mô phỏng theo cách thức các chuyên gia làm việc
- Tuy nhiên, không giống với con người, các chương trình hiện tại không tự học lấy kinh nghiệm :
  - Tri thức phải được lấy từ con người
  - Được mã hóa trong một ngôn ngữ hình thức để khai thác
- Đây là nhiệm vụ chính mà các nhà thiết kế HCG phải đương đầu

Bách khoa toàn thư mở Wikipedia

- **HCG**, còn gọi là **hệ thống dựa tri trức**, là một chương trình máy tính chứa một số tri thức đặc thù của một hoặc nhiều chuyên gia con người về một chủ đề cụ thể nào đó
- Các chương trình thuộc loại này đã được phát triển từ các thập kỷ 1960 và 1970, và trở thành ứng dụng thương mại từ thập kỷ 1980
- Nhiều HCG đã được thiết kế và xây dựng để phục vụ các lĩnh vực kế toán, y học, điều khiển tiến trình (*process control*), dịch vụ tư vấn tài chính (*finalcial service*), tài nguyên con người (*human resources*), v.v...

Bách khoa toàn thư mở Wikipedia

- Dạng phổ biến nhất của HCG là :
  - Một chương trình gồm một tập luật phân tích thông tin (thường được cung cấp bởi NSD hệ thống) về một lớp vấn đề cụ thể
  - Những phân tích về các vấn đề đã xác định
  - Tùy theo thiết kế chương trình mà đưa lời khuyên về trình tự các hành động cần thực hiện để giải quyết vấn đề
- HCG sử dụng các tri thức của những chuyên gia để giải quyết các vấn đề (bài toán) khác nhau thuộc mọi lĩnh vực
- Là một hệ thống sử dụng các khả năng lập luận để đạt tới các kết luận
- Các thuật ngữ có cùng nghĩa :
  - Hệ chuyên gia
  - Hệ thống dựa trên tri thức (knowledge–based system)
  - HCG dựa trên tri thức (knowledge–based expert system)

Lớp vấn đề của HCG_1

1. **Diễn giải (interpretation)**
   Đưa ra kết luận hay mô tả dễ hiểu từ những tập hợp dữ liệu thô
2. **Dự đoán (prediction)**
   Đưa ra những hậu quả có thể xảy ra khi cho trước một tình huống
3. **Chẩn đoán (diagnosis)**
   Xác định nguyên nhân của những sự cố trong các tình huống phức tạp dựa trên các triệu chứng có thể quan sát được
4. **Thiết kế (design)**
   tìm ra cấu hình cho các thành phần hệ thống, đáp ứng được các mục tiêu trong khi vẫn thỏa mãn một tập hợp các ràng buộc về thiết kế
5. **Lập kế hoạch (planning)**
   Tìm ra một chuỗi các hành động để đạt được một tập hợp các mục tiêu, khi được cho trước các điều kiện khởi đầu và những ràng buộc trong thời gian chạy (run-time)

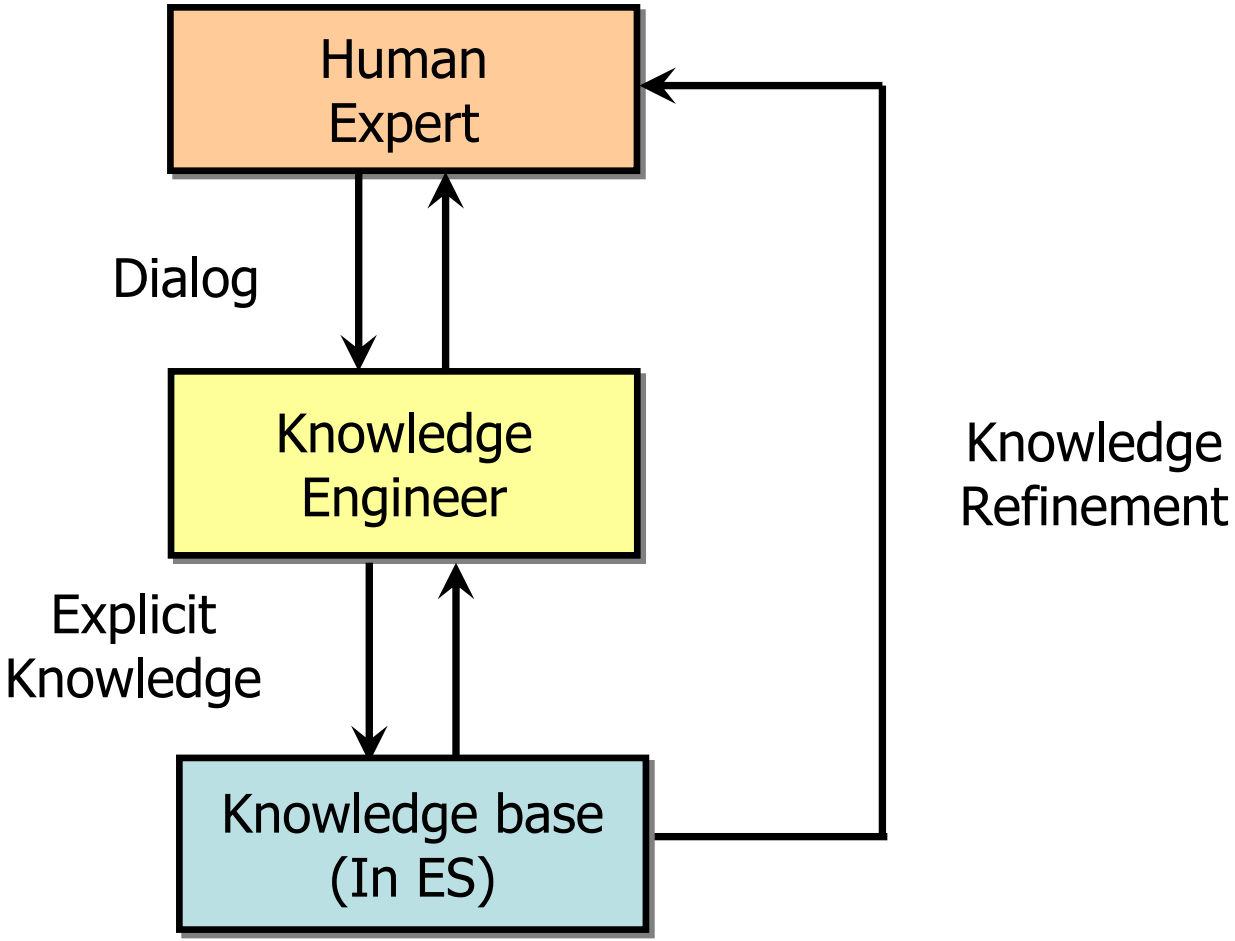
Lớp vấn đề của HCG_2

6. Theo dõi (monitoring)
   So sánh những hành vi quan sát được của hệ thống với hành vi mong đợi
7. Bắt lỗi và sửa chữa (debuging and repair)
   Chỉ định và cài đặt phương pháp chữa trị cho các sự cố, rủi ro
8. Hướng dẫn (instruction)
   Phát hiện và sửa chữa những thiếu sót trong quan niệm của học viên về một chủ đề lĩnh vực nào đó
9. Điều khiển (control)
   Chỉ đạo hành vi trong một môi trường phức tạp

Tổ chức hoạt động của HCG

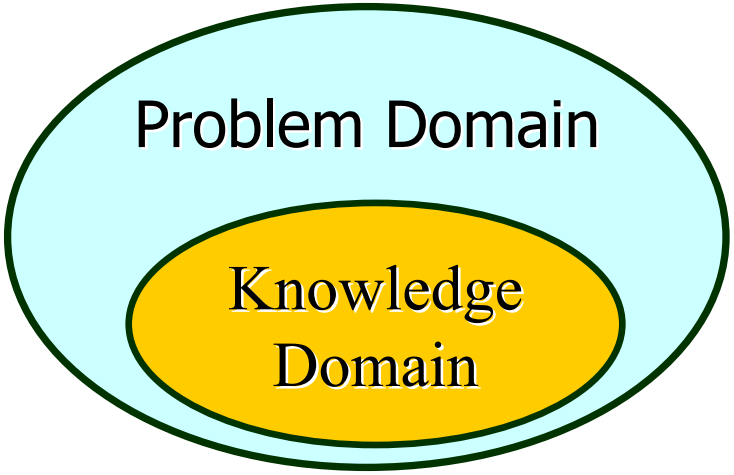
- Một hệ chuyên gia gồm ba thành phần chính :
  - *Cơ sở tri thức* (knowledge base) nơi chứa tri thức của HCG
  - *Máy suy diễn* hay *môtơ suy diễn* (inference engine)
  - *Hệ thống giao tiếp với người sử dụng* (user interface)
- Khai thác HCG : Người sử dụng (User) đặt câu hỏi (truy vấn) HCG bằng cách :
  - Cung cấp *sự kiện* (facts) là những gì đã biết, đã có thật hay những tri thức có ích (luật-rules) cho hệ chuyên gia, và nhận được những câu trả lời
- Hoạt động của HCG :
  - Máy suy diễn khai thác cơ sở tri thức để tạo ra câu trả lời là những lời khuyên hay những gợi ý đúng đắn (expertise) cho người sử dụng qua hệ thống giao tiếp

Knowledge Engineering in a Nutshell



SE overviews

- Relies on internally represented knowledge to perform tasks
- Utilizes reasoning methods to derive appropriate new knowledge
- Usually restricted to a specific *problem domain*
- The term knowledge-based system is often used synonymously
- Two distinctions from Decision Support System (DSS):
  1. has the potential to extend the manager's problem-solving ability beyond his or her normal capabilities
  2. the ability to explain how the solution was reached



Nhiều cách nhìn nhận khác nhau về HCG

Loại người sử dụng	*Vấn đề đặt ra*
Người quản trị	Tôi có thể dùng nó để làm gì ?
Kỹ thuật viên	Làm cách nào để tôi vận hành nó tốt nhất ?
Nhà nghiên cứu	Làm sao để tôi có thể mở rộng nó ?
Người sử dụng cuối	Nó sẽ giúp tôi cái gì đây ? Nó có rắc rối và tốn kém không ? Nó có đáng tin cậy không ?

Đặc trưng của hệ chuyên gia

- Có bốn đặc trưng cơ bản của một hệ chuyên gia :
  - *Hiệu quả cao* (high performance). Khả năng trả lời với mức độ tinh thông bằng hoặc cao hơn so với chuyên gia (người) trong cùng lĩnh vực.
  - *Thời gian trả lời thoả đáng* (adequate response time). Thời gian trả lời hợp lý, bằng hoặc nhanh hơn so với chuyên gia (người) để đi đến cùng một quyết định. Hệ chuyên gia là một hệ thống thời gian thực (real time system).
  - *Độ tin cậy cao* (good reliability). Không thể xảy ra sự cố hoặc giảm sút độ tin cậy khi sử dụng.
  - *Dễ hiểu* (understandable). Hệ chuyên gia giải thích các bước suy luận một cách dễ hiểu và nhất quán, không giống như cách trả lời bí ẩn của các hộp đen (black box)

Ưu điểm của hệ chuyên gia_1

- Những ưu điểm của hệ chuyên gia :
  - *Phổ cập* (increased availability). Là sản phẩm chuyên gia, được phát triển không ngừng với hiệu quả sử dụng không thể phủ nhận.
  - *Giảm giá thành* (reduced cost).
  - *Giảm rủi ro* (reduced dangers). Giúp con người tránh được trong các môi trường rủi ro, nguy hiểm.
  - *Tính thường trực* (Permanance). Bất kể lúc nào cũng có thể khai thác sử dụng, trong khi con người có thể mệt mỏi, nghỉ ngơi hay vắng mặt.
  - *Đa lĩnh vực* (multiple expertise). chuyên gia về nhiều lĩnh vực khác nhau và được khai thác đồng thời bất kể thời gian sử dụng

Ưu điểm của hệ chuyên gia_2

- Những ưu điểm của hệ chuyên gia :
  - *Độ tin cậy* (increased relialility)
    Luôn đảm bảo độ tin cậy khi khai thác.
  - *Khả năng giảng giải* (explanation). Câu trả lời với mức độ tinh thông được giảng giải rõ ràng chi tiết, dễ hiểu.
  - *Khả năng trả lời* (fast reponse). Trả lời theo thời gian thực, khách quan.
  - *Tính ổn định, suy luận có lý và đầy đủ mọi lúc mọi nơi* (steady, une motional, and complete response at all times).
  - *Trợ giúp thông minh như một người hướng dẫn* (intelligent -tutor).
  - *Có thể truy cập như là một cơ sở dữ liệu thông minh* (intelligent database)

Lịch sử phát triển của HCG 1

Năm	*Các sự kiện*
1943	Dịch vụ bưu điện ; mô hình Neuron của (Mc Culloch and Pitts Model)
1954	Thuật toán Markov (Markov Algorithm) điều khiển thực thi các luật
1956	Hội thảo Dartmouth ; lý luận logic ; tìm kiếm nghiệm suy (heuristic search) ; thống nhất thuật *ngữ trí tuệ nhân tạo* (AI: Artificial Intelligence)
1957	Rosenblatt phát minh khả năng nhận thức ; Newell, Shaw và Simon đề xuất giải bài toán tổng quát (GPS: General Problem Solver)
1958	Mc Carthy đề xuất ngôn ngữ trí tuệ nhân tạo LISA (LISA AI language)
1962	Nguyên lý Rosenblatt's về chức năng thần kinh trong nhận thức (Rosenblatt's *Principles of Neurodynamicdynamics* on Perceptions)
1965	Phương pháp hợp giải Robinson. Ứng dụng logic mờ (fuzzy logic) trong suy luận về các đối tượng mờ (fuzzy object) của Zadeh. Xây dựng hệ chuyên gia đầu tiên về nha khoa DENDRAL (Feigenbaum , Buchanan , et.al)
1968	Mạng ngữ nghĩa (semantic nets), mô hình bộ nhớ kết hợp (associative memory model) của Quillian

Lịch sử phát triển của HCG 2

Năm	*Các sự kiện*
1969	Hệ chuyên gia về Toán học MACSYMA (Martin and Moses)
1970	Ứng dụng ngôn ngữ PROLOG (Colmerauer, Roussell, et, al.)
1971	Hệ chuyên gia HEARSAY I về nhận dạng tiếng nói (speech recognition). Xây dựng các luật giải bài toán con người (*Human Problem Solving* popularizes rules (Newell and Simon)
1973	Hệ chuyên gia MYCIN về chẩn trị y học (Shortliffe, et,al.)
1975	Lý thuyết khung (frames), biểu diễn tri thức (knowledge representation) (Minsky)
1976	Toán nhân tạo (AM: Artificial Mathematician) (Lenat). Lý thuyết Dempster–Shafer về tính hiển nhiên của lập luận không chắc chắn (Dempster–Shafer theory of Evidence for reason under uncertainty). Ứng dụng hệ chuyên gia PROSPECTOR trong khai thác hầm mỏ (Duda, Har)
1977	Sử dụng ngôn ngữ chuyên gia OPS (OPS expert system shell) trong hệ chuyên gia XCON/R1 (Forgy)

Lịch sử phát triển của HCG 3

Năm	*Các sự kiện*
1978	Hệ chuyên gia XCON/R1 (McDermott, DEC) để bảo trì hệ thống máy tính DEC (DEC computer systems)
1979	Thuật toán mạng về so khớp nhanh (rete algorithm for fast pattern matching) của Forgy ; thương mại hoá các ứng dụng về trí tuệ nhân tạo
1980	Ký hiệu học (symbolics), xây dựng các máy LISP (LISP machines) từ LMI.
1982	Hệ chuyên gia về Toán học (SMP math expert system) ; mạng nơ-ron Hopfield (Hopfield Neural Net) ; Dự án xây dựng máy tính thông minh thế hệ 5 ở Nhật bản (Japanese Fifth Generation Project to develop intelligent computers)
1983	Bộ công cụ phục vụ hệ chuyên gia KEE (KEE expert system tool) (intelli Corp)
1985	Bộ công cụ phục vụ hệ chuyên gia CLIPS (CLIPS expert system tool (NASA)

Các lĩnh vực ứng dụng của hệ chuyên gia

- Cho đến nay, hàng trăm hệ chuyên gia đã được xây dựng và đã được báo cáo thường xuyên trong các tạp chí, sách, báo và hội thảo khoa học
- Ngoài ra còn các hệ chuyên gia được sử dụng trong các công ty, các tổ chức quân sự mà không được công bố vì lý do bảo mật

Một số lĩnh vực ứng dụng

Lĩnh vực	*Ứng dụng diện rộng*
Cấu hình (Configuration)	Tập hợp thích đáng những thành phần của một hệ thống theo cách riêng
Chẩn đoán (Diagnosis)	Lập luận dựa trên những chứng cứ quan sát được
Truyền đạt (Instruction)	Dạy học kiểu thông minh sao cho sinh viên có thể hỏi vì sao (why?), như thế nào (how?) và cái gì nếu (what if?) giống như hỏi một người thầy giáo
Giải thích (Interpretation)	Giải thích những dữ liệu thu nhận được
Kiểm tra (Monitoring)	So sánh dữ liệu thu lượm được với dữ liệu chuyên môn để đánh giá hiệu quả
Lập kế hoạch (Planning)	Lập kế hoạch sản xuất theo yêu cầu
Dự đoán (Prognosis)	Dự đoán hậu quả từ một tình huống xảy ra
Chữa trị (Remedy)	Chỉ định cách thụ lý một vấn đề
Điều khiển (Control)	Điều khiển một quá trình, đòi hỏi diễn giải, chẩn đoán, kiểm tra, lập kế hoạch, dự đoán và chữa trị

Một số hệ chuyên gia 1

Bảng 1 Ngành hoá học (Chemistry)	
CRYSALIS	Interpret a protein'n 3-D structure
DENDRAL	Interpret molecular structure
TQMSTUNE	Remedy Triple Quadruple Mass Spectrometer (keep it tuned)
CLONER	Design new biological molecules
MOLGEN	Design gene - cloning experiments
SECS	Design complex organic molecules
SPEX	Plan molecular biology experiments

Một số hệ chuyên gia 2

Bảng 2 Ngành điện tử (Electronics)	
ACE	Diagnosis telephone network faults
IN -ATE	Diagnosis oscilloscope faults
NDS	Diagnosis national communication net
EURISKO	Design 3-D micro-electronics
PALLADIO	Design and test new VLSI cicuits
REDESIGN	Redesign digital circuits to new
CADHELP	Instruct for computer aided design
SOPHIE	Instruct circuit fault diagnosis

Một số hệ chuyên gia 3

Bảng 3 Ngành địa chất (Geology)	
DIPMETER	Interpret dipmeter logs
LITHO	Interpret oil well log data
MUD	Diagnosis / remedy drilling problems
PROSPECTOR	Interpret geologic data for minerals
Bảng 4 Công nghệ (Engineering)	
REACTOR	Diagnosis / remedy reactor accidents
DELTA	Diagnosis / remedy GE locomotives
STEAMER	Instruct operation - steam power-plant

Một số hệ chuyên gia 5

Bảng 5 Ngành y học (Medicine)	
PUFF	Diagnosis lung disease
VM	Monitors intensive - care patients
ABEL	Diagnosis acid - base / electrolytes
AI/COAG	Dianosis blood disease
AI/ RHEUM	Diagnosis rheumatoid disease
CADUCEUS	Diagnosis internal medicine disease
ANNA	Monitor digitalis therapy
BLUE BOX	Diagnosis / remedy depression
MYCIN	Diagnosis / remedy bacterial infections
ONCOCIN	Remedy / manage chemotherapy patient
ATTENDING	Instruct in anesthetic manegement
GUIDON	Instruct in bacterial infections

Một số hệ chuyên gia 6

Bảng 6 Máy tính điện tử (Computer systems)	
PTRANS	Prognosis for managing DEC computers
BDS	Diagnosis bad parts in switching net
XCON	Configune DEC computer systems
XSEL	Configure DEC computer sales order
XSITE	Configure customer site for DEC computers
YES/MVS	Monitor / control IBM MVS opeating system
TIMM	Diagnosis DEC computer

Example of ES

The end-user usually sees an expert system through an interactive dialog:

Q. Do you know to which restaurant you want to go?

A. No

Q. Is there any kind of food you would particularly like?

A. Unknown

Q. Do you like spicy food?

A. No

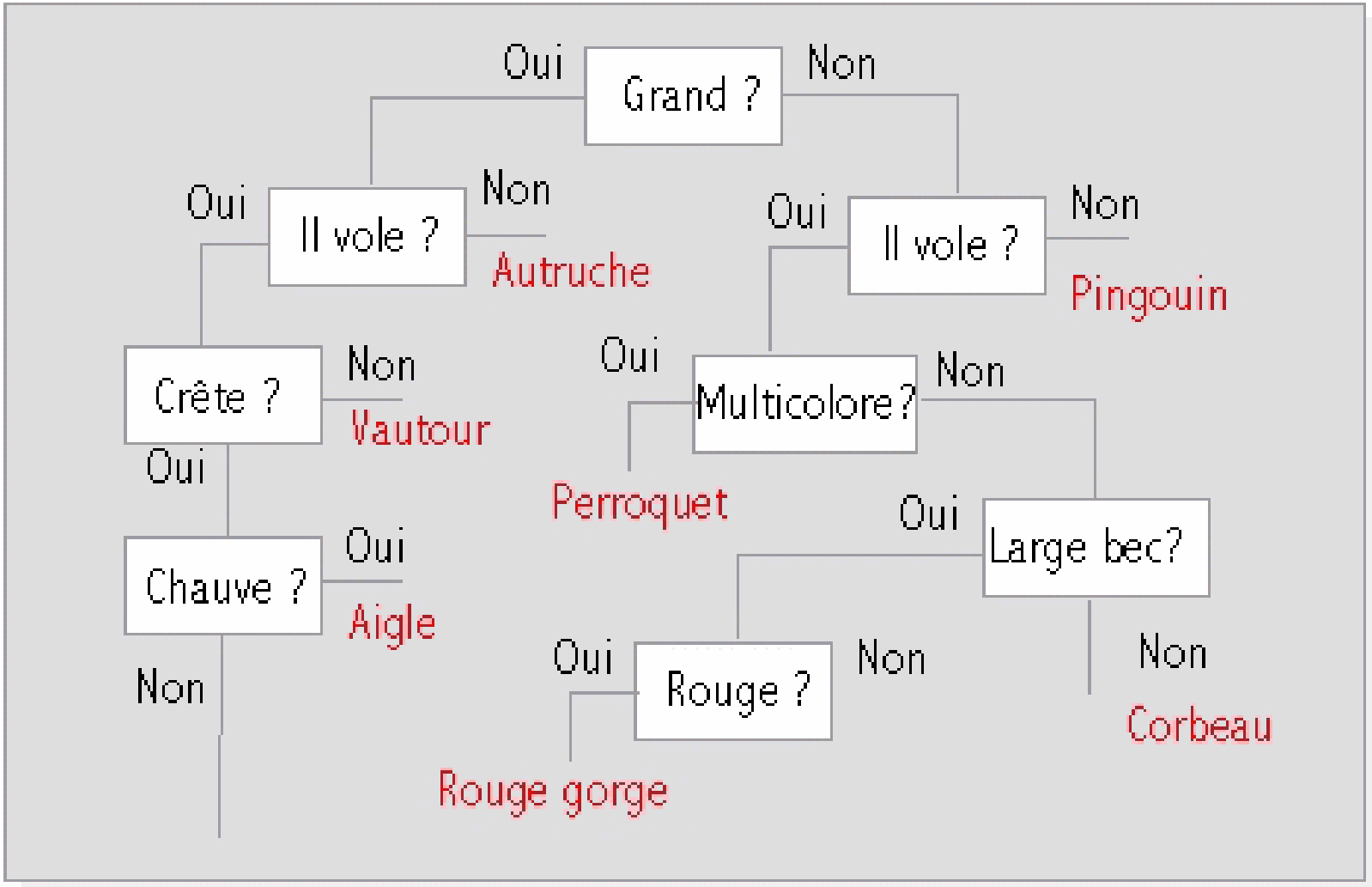
Q. Do you usually drink wine with meals?

A. Yes

Q. When you drink wine, is it French wine?

A. Why

Example of ES



ES Problems

- Limited knowledge
  - "Shallow" knowledge
    - no "deep" understanding of the concepts and their relationships
  - no "common-sense" knowledge
  - no knowledge from possibly relevant related domains
  - "closed world"
    - the ES knows only what it has been explicitly "told"
    - it doesn't know what it doesn't know
- Mechanical reasoning
  - may not have or select the most appropriate method for a particular problem
  - some "easy" problems are computationally very expensive
- Lack of trust
  - users may not want to leave critical decisions to machines

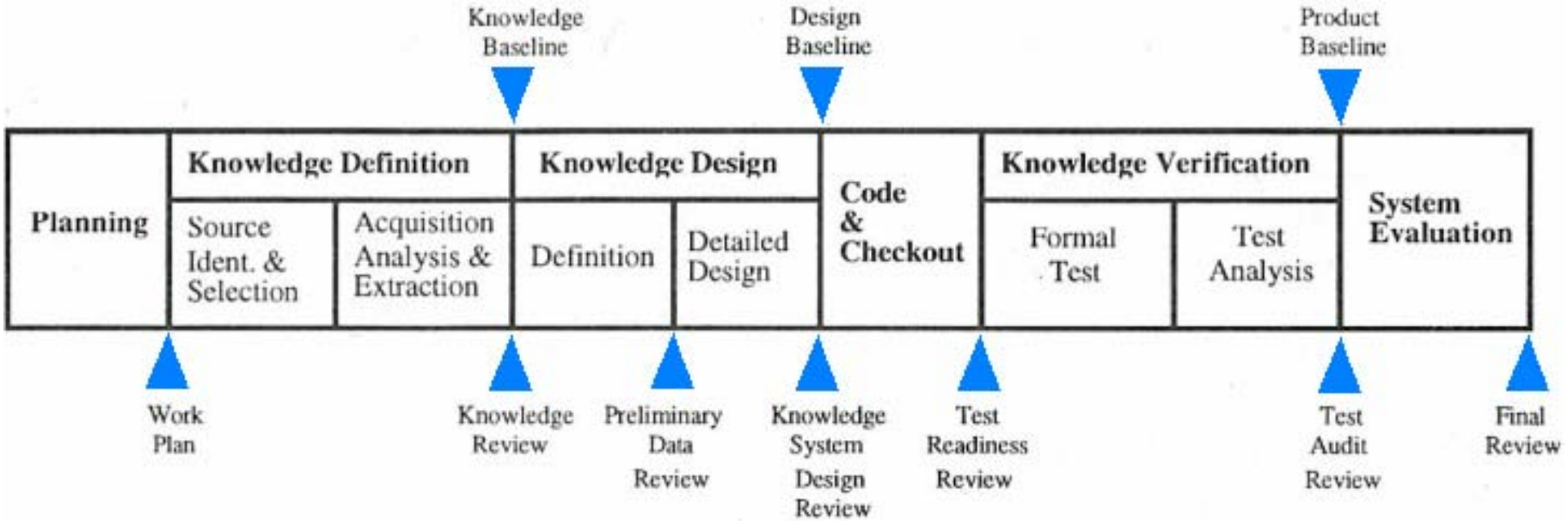
ES Design + Development

- Which are the major steps in the Linear Model of ES Development?
- What do you have to take into account when you plan an ES project?
- What can you say about the delivery phase of an ES project?
- What kind of errors can occur (during the development) of an ES?

ES Design + Development

- Major basic steps in ES development
  - Identification:
    Determine problem and project requirements
  - Conceptualization: Extract knowledge and construct a conceptual model for solving problems using the knowledge
  - Formalization: Map the conceptual model to available representation and control paradigms
  - Implementation: Construct/extend the implementation of the formalized model
  - Testing: Evaluate the system
- Software Engineering models
- Project phases
- Linear Model of ES Development
- Sources of error in ES

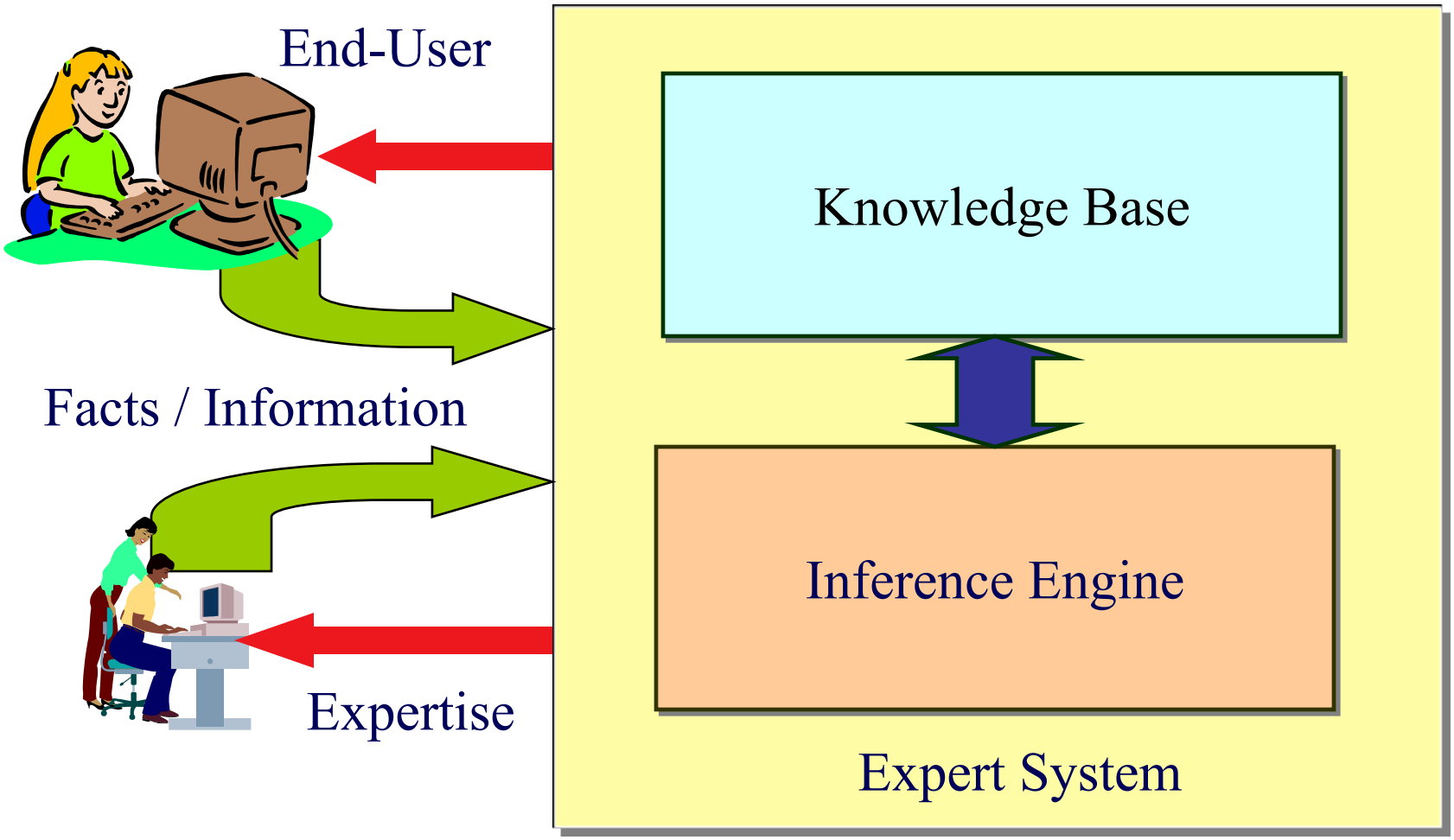
The linear model of ES development life cycle



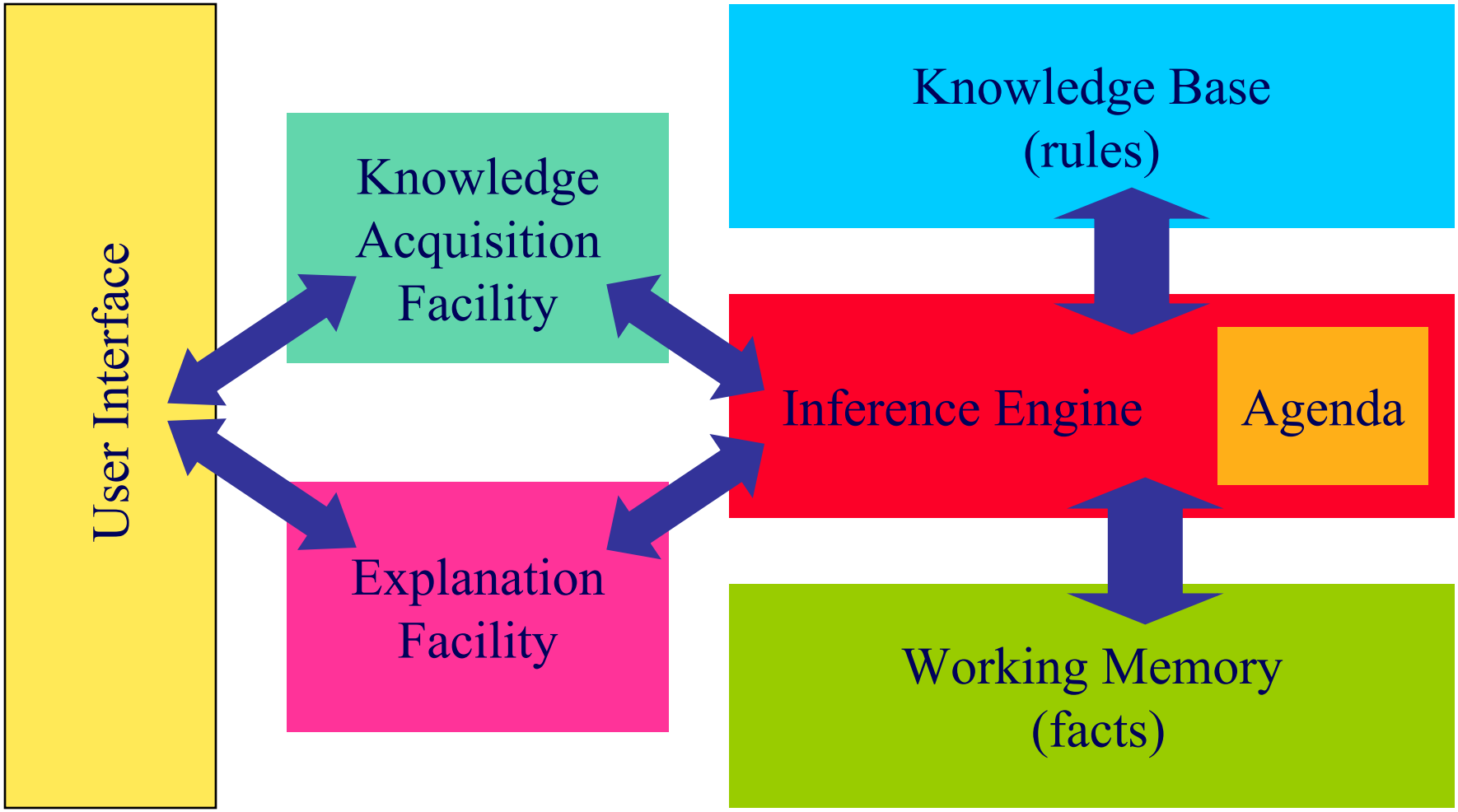
Các phần tử của HCG

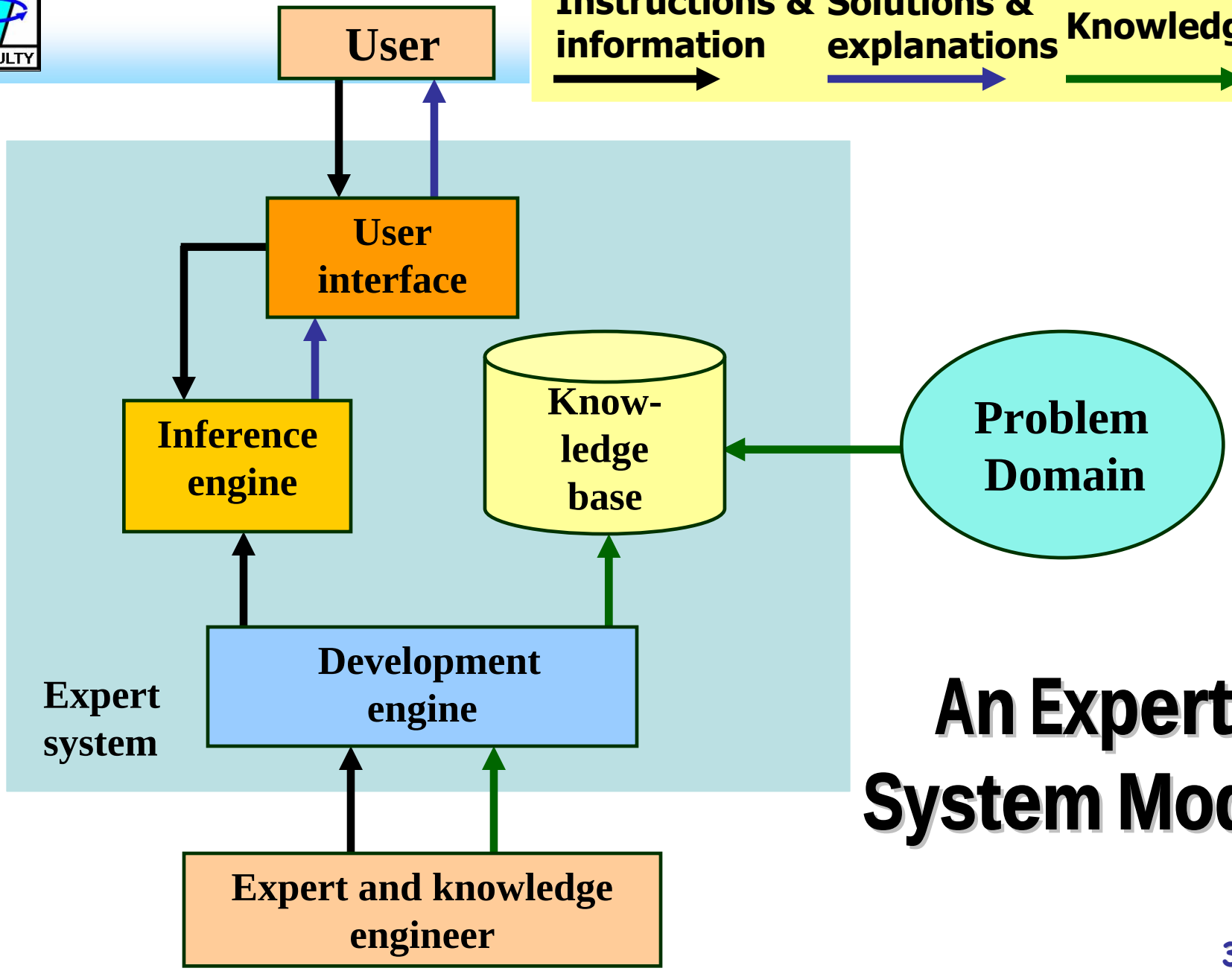
- Môt HCG gồm 3 thành phần cơ bản :
  - Cơ sở tri thức (Knowledge Base)
  - Máy duy diễn (Inference Engine)
  - Giao diện người sử dụng (User Interface)
- Ngoài ra, tuỳ theo mức độ tiếp cận, mà người ta bổ sung các thành phần khác, chẳng hạn :
  - Lịch công việc (Agenda)
  - Bộ nhớ làm việc (Working Memory)
  - Khả năng giải thích (Explanation Facility)
  - Khả năng thu nhận tri thức (Explanation Facility)
  - ...

General Architecture of SE



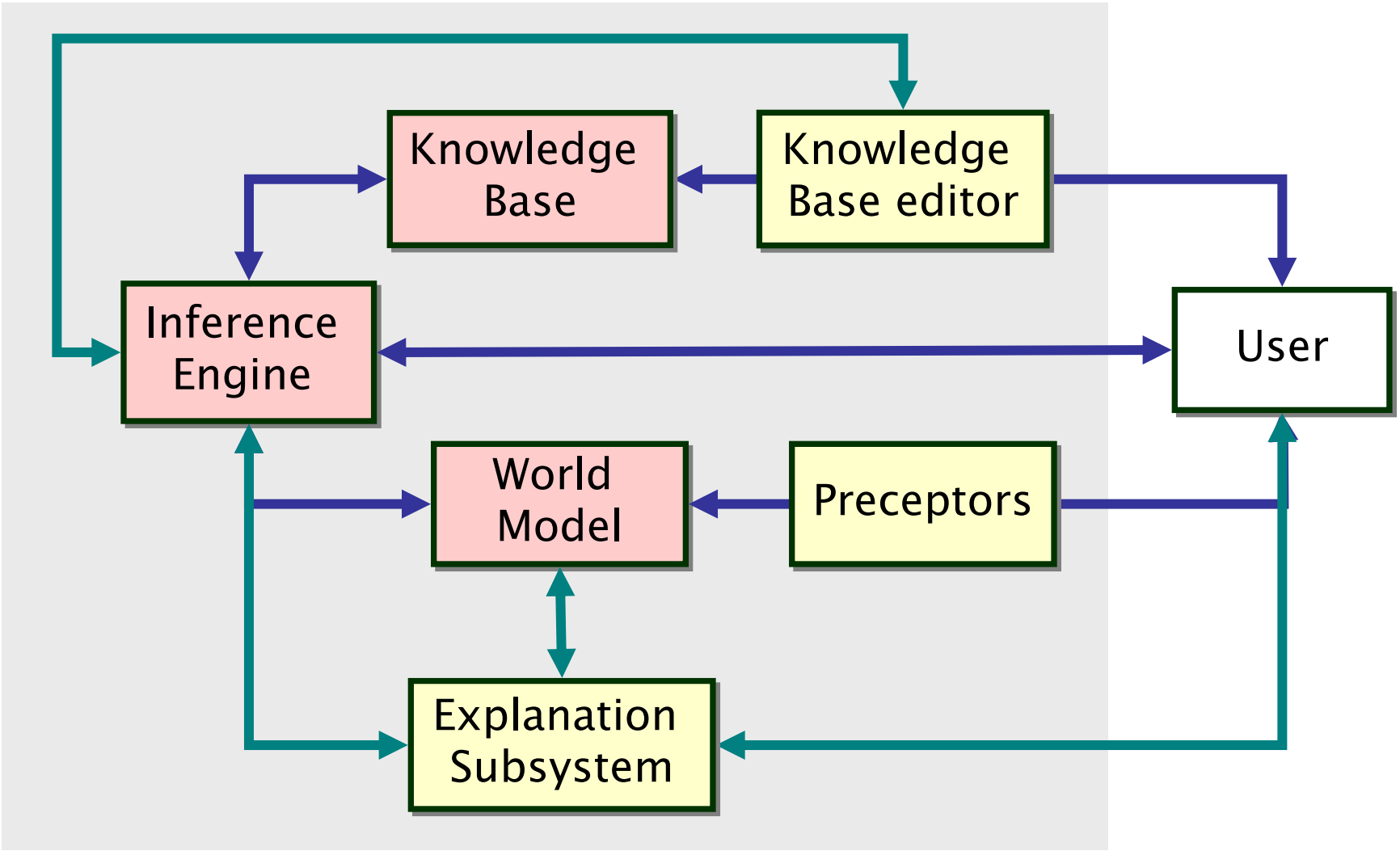
Expert system model - main parts



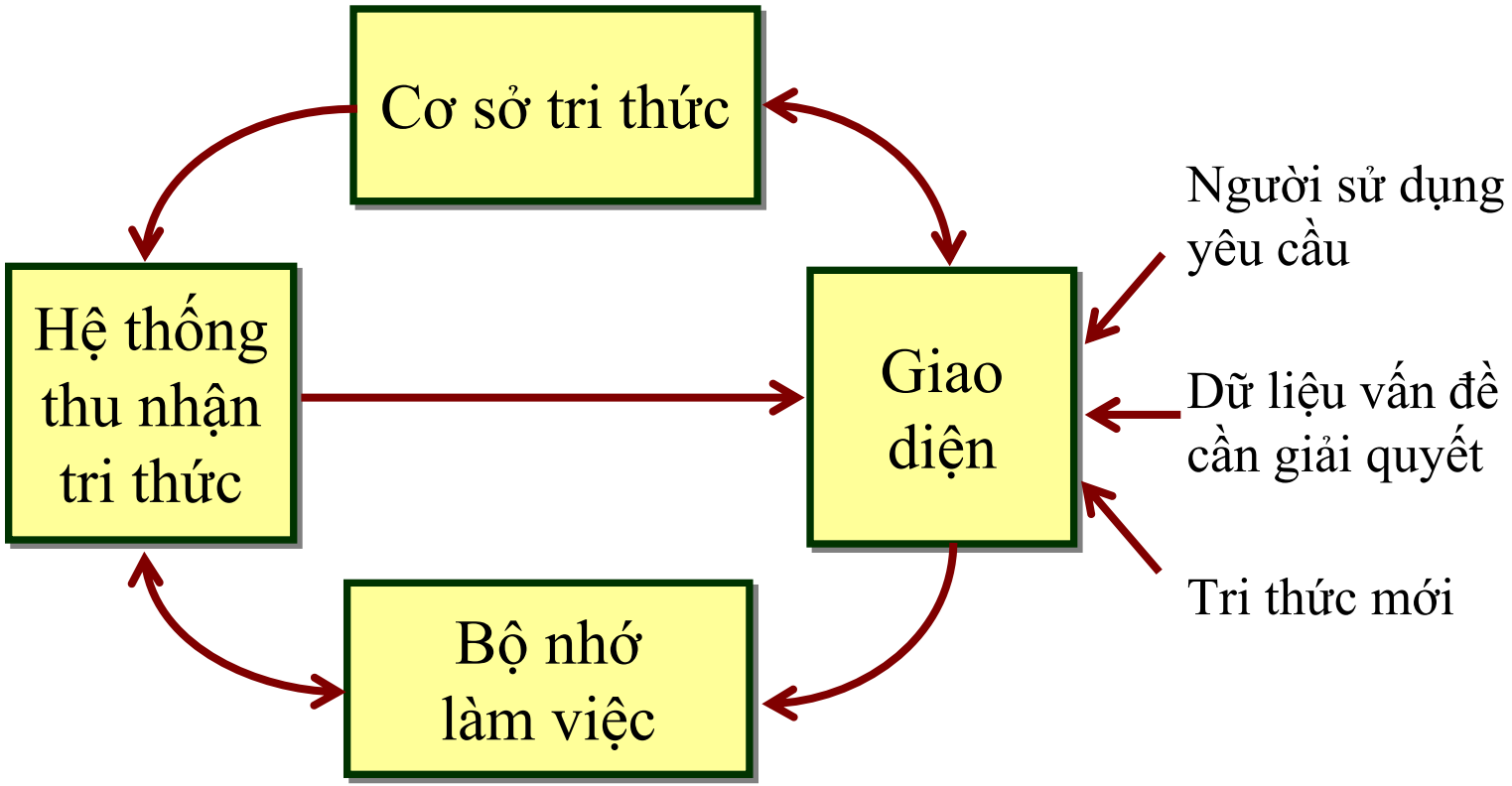


An Expert System Model

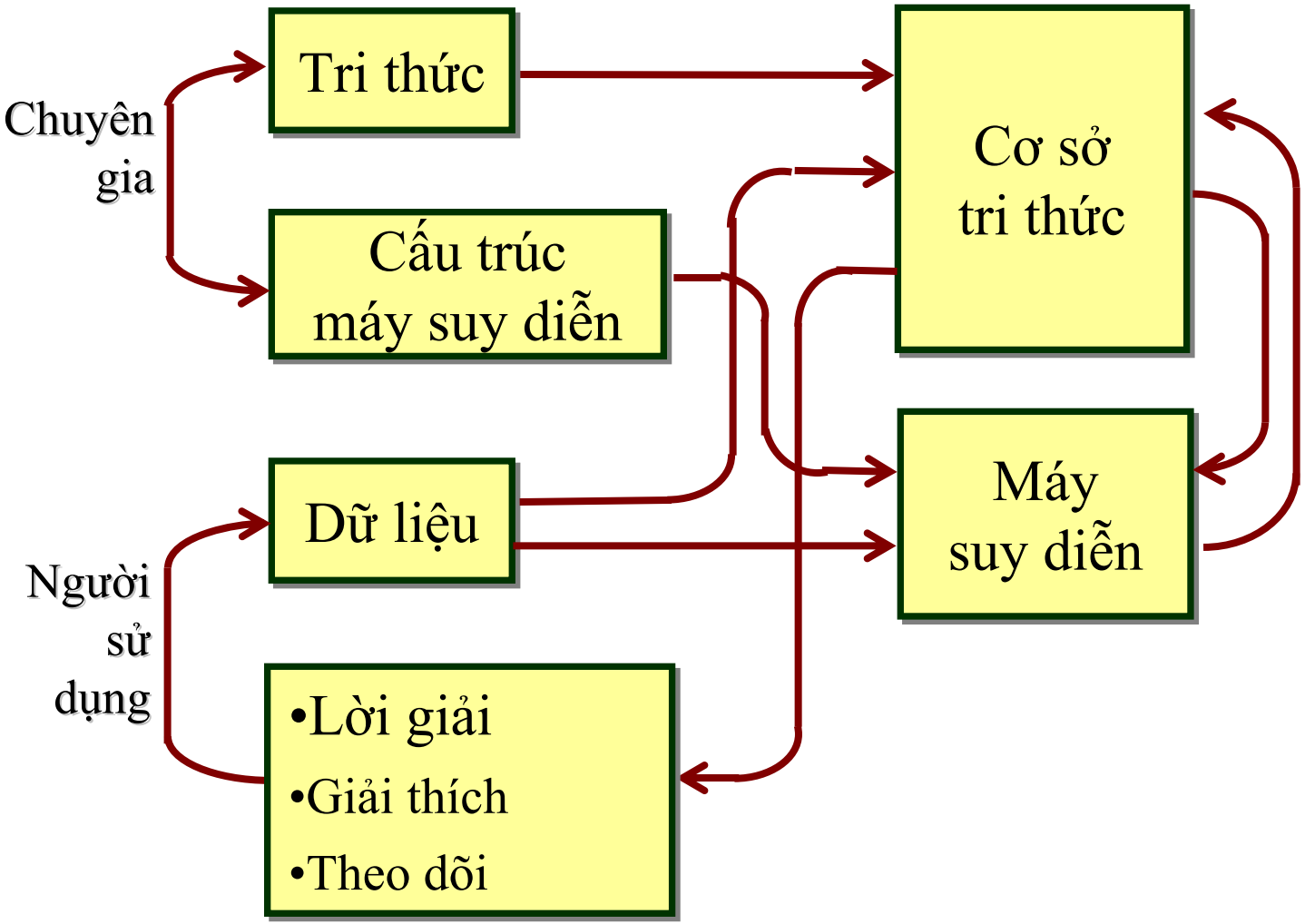
Một số mô hình khác



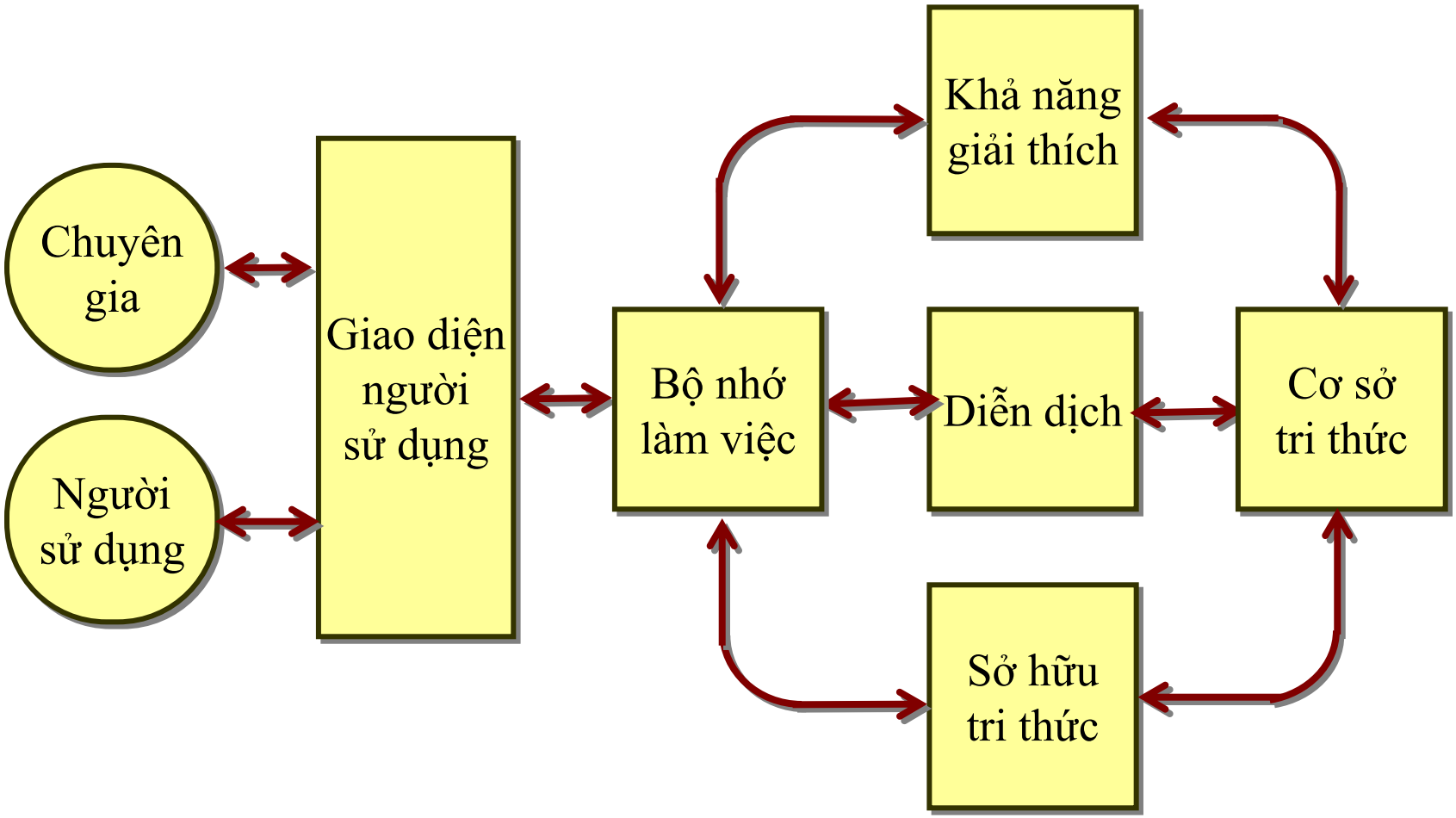
Mô hình J. L. Ermine



Mô hình C. Ernest



Mô hình E. V. Popov



Tri thức và Cơ sở tri thức

- Tri thức trong HCG :
  - Phản ánh sự tinh thông được tích tụ từ sách vở, tạp chí, từ các chuyên gia hay các nhà bác học
- Cơ sở tri thức gồm :
  - Các phần tử (hay đơn vị) tri thức được tổ chức như một CSDL
  - Tuỳ theo phương pháp phát triển HCG mà các phần tử có đặc trưng, tính chất khác nhau
  - Chẳng hạn trong các HCG dựa theo luật (Rules-Based ES) :
    - Mỗi phần tử tri thức là một luật
    - Cơ sở tri thức như vậy còn được gọi là bộ nhớ sản xuất (Production Memeory)

Phân loại tri thức

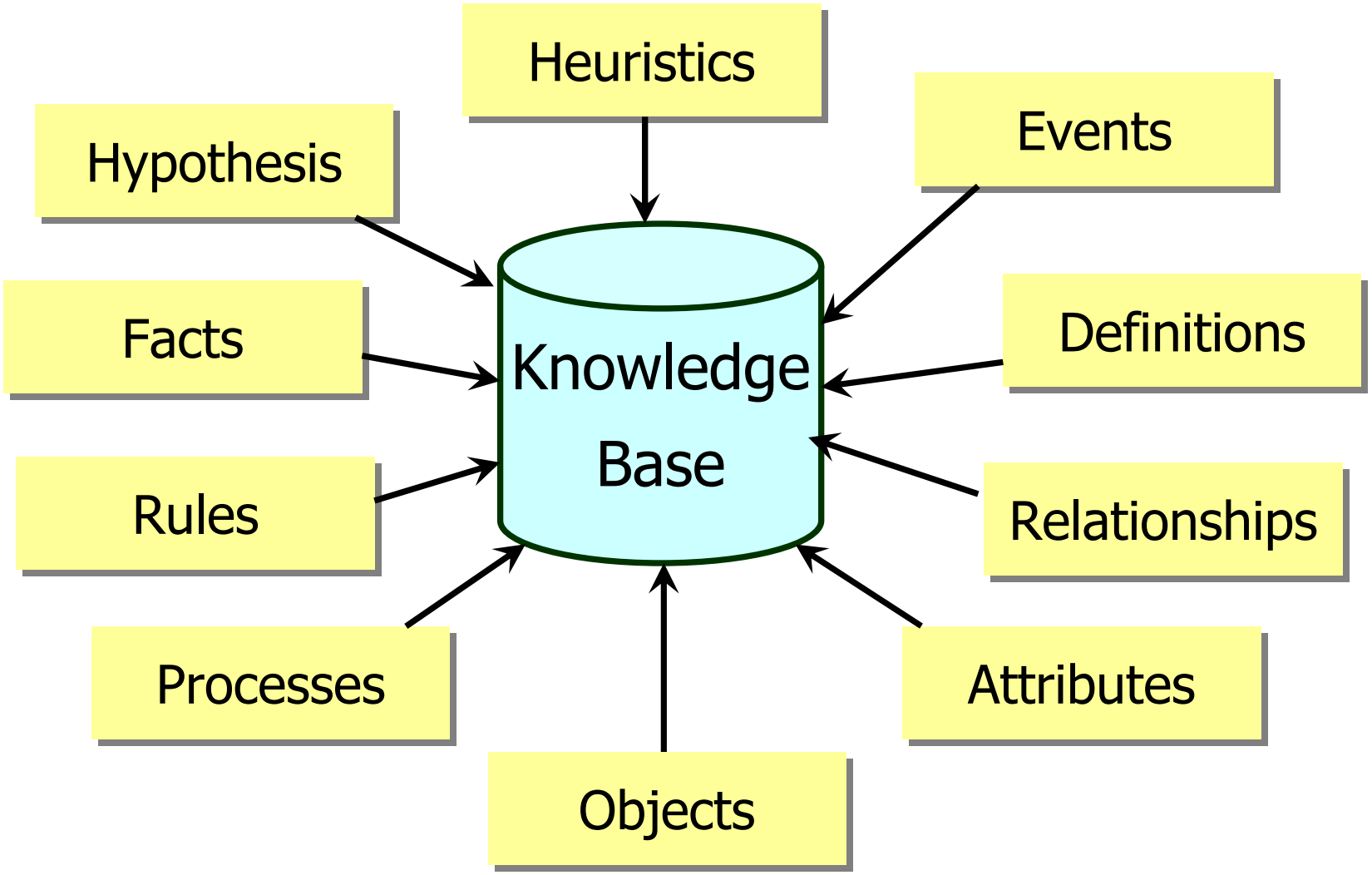
- Người ta thường phân loại tri thức theo nhiều cách
- Theo phân loại chức năng, có hai loại tri thức :
  - *Tri thức phán đoán* (Assertion Knowledge) :
    mô tả các tình huống đã được thiết lập hoặc sẽ được thiết lập
  - *Tri thức thực hành* (Operating Knowledge)
    thể hiện những hậu quả rút ra hay những thao tác cần phải hoàn thiện khi một tình huống đã được thiết lập hoặc sẽ được thiết lập trong lĩnh vực đang xét
- Tri thức thực hành :
  - Thường được thể hiện bởi các biểu thức Toán học+Lôgic dễ hiểu, dễ triển khai
  - NSD có thể thao tác với tri thức như khai thác một CSDL

Phân loại tri thức theo biểu hiện

- Theo phân loại theo biểu hiện, cũng có hai loại :
- Tri thức hiện, rõ (explicit knowledge)
  - Diễn đạt bằng ngôn ngữ hình thức, dễ trao đổi giữa các cá nhân
  - Có thể biểu diễn bằng các công thức khoa học, các thủ tục tường minh, hoặc nhiều cách khác
  - Bao gồm thông tin, dữ liệu, sách báo, văn bản, tài liệu đã được hệ thống bằng nhiều phương tiện

- Tri thức ngầm, ẩn (tacit knowledge)
  - Có được và ẩn chứa trong kinh nghiệm của từng cá nhân, mang tính chủ quan, bao gồm những hiểu biết riêng thấu đáo, trực giác, linh cảm, kỹ năng, …
  - Khó trao đổi hoặc chia sẻ với người khác
  - Chỉ có thể học được từ người khác nhờ quan hệ gần gũi trong một khoảng thời gian nào đó

Knowledge-base may really include many things



Khái niệm về Quản trị tri thức

- Người ta thường nói :
  - Tri thức là sức mạnh
  - Tri thức là của cải
- Các tổ chức của LHQ rút ra kết luận rằng :
  - Kinh tế thế giới phát triển mạnh mẽ do biết dựa vào tri thức
- Làm thế nào để sáng tạo và sử dụng đúng đắn các tri thức

- Quản trị tri thức (Knowledge Management)

ES Elements: Máy duy diễn

- *Máy duy diễn* (inference engine) :
  - Công cụ (chương trình, hay bộ xử lý) tạo ra sự suy luận bằng cách quyết định xem những luật nào sẽ làm thỏa mãn các sự kiện, các đối tượng
  - Chọn ưu tiên các luật thỏa mãn
  - Thực hiện các luật có tính ưu tiên cao nhất

ES Elements : Giao diện người sử dụng

- *Giao diện người sử dụng* (user interface) :
  - Là nơi người sử dụng và hệ chuyên gia trao đổi với nhau

ES Elements : Các thành phần khác

- *Lịch công việc* (agenda) :
  - Danh sách các luật ưu tiên do máy suy diễn tạo ra thoả mãn các sự kiện, các đối tượng có mặt trong bộ nhớ làm việc.
- *Bộ nhớ làm việc* (working memory) :
  - Cơ sở dữ liệu toàn cục chứa các sự kiện phục vụ cho các luật
- *Khả năng giải thích* (explanation facility) :
  - Giải nghĩa cách lập luận của hệ thống cho người sử dụng
- *Khả năng thu nhận tri thức* (explanation facility) :
  - Cho phép NSD bổ sung tri thức vào hệ thống
  - Khả năng thu nhận tri thức là yếu tố mặc nhiên của nhiều hệ chuyên gia

Phát hiện tri thức và khai phá dữ liệu

- Phát hiện tri thức (PHTT) và khai phá dữ liệu (KPDL)
- Bối cảnh :
  - Sự phát triển nhanh chóng của lĩnh vực CNTT và ứng dụng CNTT trong đời sống, kinh tế, xã hội, quốc phòng... hiện nay
  - Khối lượng dữ liệu thu thập, xử lý, khai thác và lưu trữ ngày càng tích luỹ nhiều lên
  - Tuy nhiên, chỉ có khoảng từ 5% đến 10% lượng dữ liệu có ý nghĩa sử dụng theo thống kê
  - Rất nhiều dữ liệu chưa được sử dụng hiệu quả nhưng vẫn phải tiếp tục thu thập rất tốn kém (nỗi lo sợ bỏ sót dữ liệu quan trọng có thể cần đến sau này)
  - Trong môi trường cạnh tranh, nhu cầu có nhiều thông tin với tốc độ nhanh để trợ giúp ra quyết định và ngày càng có nhiều câu hỏi mang tính chất định tính cần phải trả lời dựa trên một khối lượng dữ liệu khổng lồ đã có
  - Các phương pháp quản trị và khai thác cơ sở dữ liệu truyền thống ngày càng không đáp ứng được thực tế
- Sự ra đời kỹ thuật "Phát hiện tri thức và khai phá dữ liệu"

Tình hình nghiên cứu PHTT và KPDL

- Thuật ngữ :
  - Phát hiện tri thức (Knowledge Discovery)
  - Khai phá dữ liệu (Data Mining)
- Đã và đang được nghiên cứu, ứng dụng trong nhiều lĩnh vực khác nhau ở các nước trên thế giới
- Tại Việt Nam :
  - Kỹ thuật này vẫn còn tương đối mới mẻ
  - Đang được nghiên cứu
  - Đang dần đưa vào ứng dụng thực tiễn

Khái niệm "Khai phá dữ liệu"

- Khai phá dữ liệu (Data Mining) :
  quá trình trích xuất các thông tin có giá trị tiềm ẩn bên trong lượng lớn dữ liệu được lưu trữ trong các CSDL, kho dữ liệu…
- Hiện nay, ngoài thuật ngữ khai phá dữ liệu, người ta còn dùng một số thuật ngữ khác có ý nghĩa tương tự như :
  - Khai phá tri thức từ CSDL (Knowlegde Mining From Databases)
  - Trích lọc dữ liệu (Knowlegde Extraction)
  - Phân tích dữ liệu/mẫu (Data/Pattern Analysis)
  - Khảo cổ dữ liệu (Data Archaeology)
  - Nạo vét dữ liệu (Data Dredging)

Qui trình 5 bước PHTT và KPDL

1. Tìm hiểu lĩnh vực ứng dụng và hình thành bài toán

   Đây là bước quyết định cho phép chọn các phương pháp khai phá dữ liệu thích hợp với mục đích ứng dụng và bản chất của dữ liệu

2. Thu thập và xử lý thô (hay tiền xử lý dữ liệu)

   Bước này thường chiếm dụng nhiều thời gian trong toàn bộ qui trình

3. Khai phá dữ liệu, hay nói cách khác là trích ra các mẫu và/hoặc các mô hình ẩn dưới các dữ liệu

   Đây là giai đoạn thiết yếu, trong đó các phương pháp thông minh sẽ được áp dụng để trích xuất ra các mẫu dữ liệu hay tri thức hữu ích

4. Đánh giá mẫu (pattern evaluation) :

   Hiểu tri thức đã tìm được, đặc biệt là làm sáng tỏ các mô tả và dự đoán, đánh giá sự hữu ích của các mẫu biểu diễn tri thức dựa vào một số phép đo

5. Trình diễn dữ liệu (knowlegde presentation):

   sử dụng các kĩ thuật trình diễn và trực quan hoá dữ liệu để biểu diễn tri thức khai phá được cho người sử dụng

Xử lý thô dữ liệu

- Làm sạch dữ liệu (data cleaning) :

  loại bỏ nhiễu hoặc các dữ liệu không thích hợp

- Tích hợp dữ liệu (data integration) :

  tích hợp dữ liệu từ các nguồn khác nhau như :

  CSDL, Kho dữ liệu, các loại văn bản khác nhau...

- Chọn dữ liệu (data selection) :

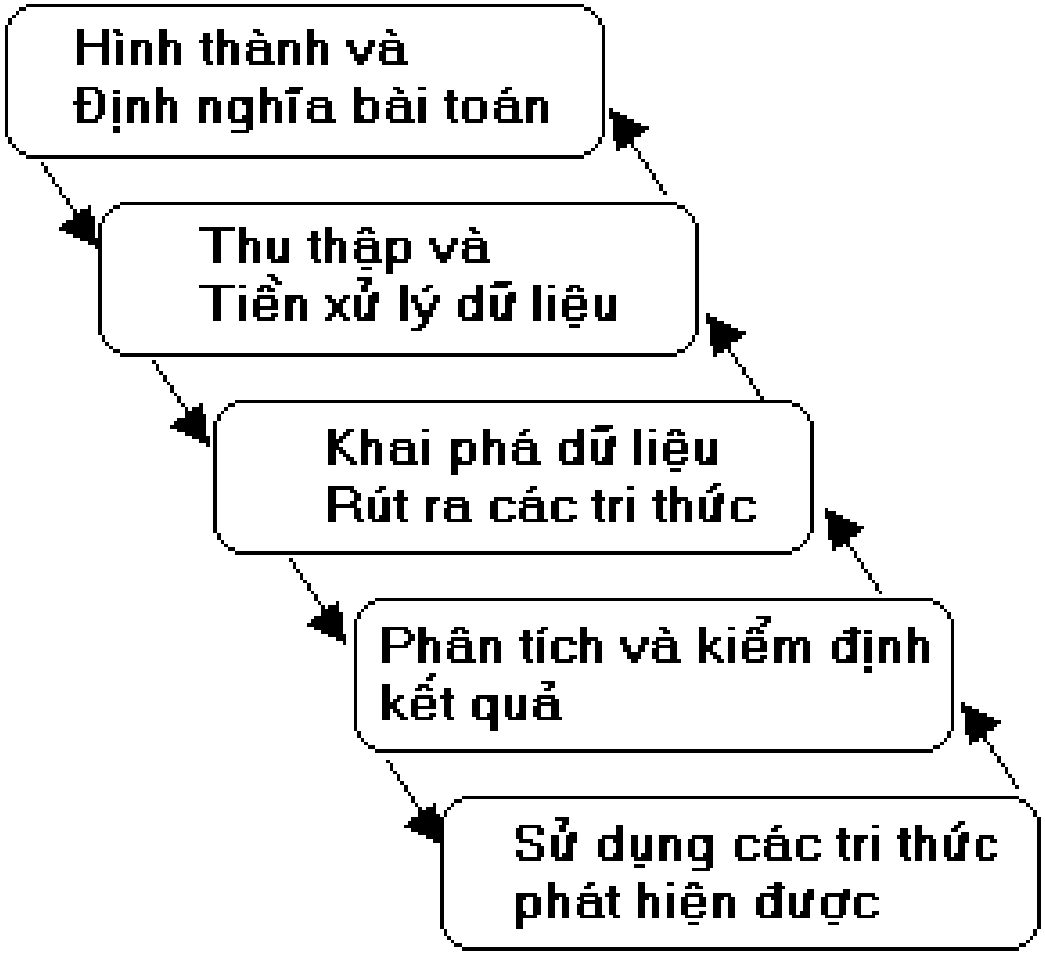
  ở bước này, những dữ liệu liên quan trực tiếp đến nhiệm vụ sẽ được thu thập từ các nguồn dữ liệu ban đầu

- Chuyển đổi dữ liệu (data transformation) :

  trong bước này, dữ liệu sẽ được chuyển đổi về dạng phù hợp cho việc khai phá bằng cách thực hiện các thao tác nhóm hoặc tập hợp

Qui trình PHTT và KPDL

- Các bước trong quy trình có thể lặp đi lặp lại một số lần
- Kết quả thu được có thể được lấy trung bình từ số lần thực hiện



Các phương pháp PHTT và KPDL

- Nhằm hai mục đích chính :
  - Dự đoán (Prediction)
  - Mô tả (Description)
- Thường sử dụng các phương pháp sau :
  - Phân loại (Classification)
  - Hồi qui (Regression)
  - Phân nhóm (Clustering)
  - Tổng hợp (Summarization)
  - Mô hình ràng buộc (Dependency modeling)
  - Dò tìm biến đổi và độ lệch (Change and Deviation Dectection)
  - Biểu diễn mô hình (Model Representation)
  - Kiểm định mô hình (Model Evaluation)
  - Phương pháp tìm kiếm (Search Method)

Các lĩnh vực liên quan đến PHTT và KPDL

- PHTT và KPD liên quan đến nhiều ngành, nhiều lĩnh vực :
  - Thống kê
  - Trí tuệ nhân tạo
  - Cơ sở dữ liệu
  - Thuật toán học
  - Tính toán song song và tốc độ cao
  - Thu thập tri thức cho các hệ chuyên gia
  - Quan sát dữ liệu...
- PHTT và KPD rất gần gũi với các lĩnh vực :
  - Thống kê, sử dụng các phương pháp thống kê để mô hình dữ liệu và phát hiện các mẫu, luật...
  - Ngân hàng dữ liệu (Data Warehousing) và các công cụ phân tích trực tuyến (OLAP)

Ứng dụng PHTT và KPDL

- Thông tin thương mại :
  - Phân tích dữ liệu marketing, khách hàng
  - Phân tích đầu tư
  - Phê duyệt cho vay vốn
  - Phát hiện gian lận…
- Thông tin kỹ thuật :
  - Điều khiển và lập lịch trình
  - Quản trị mạng
  - Phân tích các kết quả thí nghiệm…
- Thông tin khoa học
- Thông tin cá nhân...

Các thách thức của PHTT và KPDL

- Các CSDL thường rất lớn, thường xuyên biến đổi và phát triển không ngừng
- Đặc điểm :
  - Các CSDL thường có số chiều lớn
  - Thay đổi dữ liệu và tri thức có thể làm cho các mẫu đã phát hiện không còn phù hợp
  - Dữ liệu bị thiếu hoặc nhiễu
  - Quan hệ giữa các trường dữ liệu phức tạp
- Vấn đề giao tiếp với người sử dụng và kết hợp với các tri thức đã có
- Khả năng tích hợp với các hệ thống khác...

Hệ chuyên gia (Expert System)

PGS.TS. Phan Huy Khánh

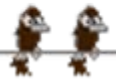
khanhph@vnn.vn

Chương 2

Biểu diễn tri thức nhờ logic vị từ bậc một

2.1

Biểu diễn tri thức nhờ logic vị từ bậc một



Biểu diễn tri thức trong một HCG

- Tri thức của một HCG có thể được biểu diễn theo nhiều phương pháp khác nhau
- Tuỳ theo từng HCG, người ta có thể sử dụng một hoặc đồng thời cả nhiều phương pháp
- Phân chia theo ngôn ngữ sử dụng, người ta có 3 cách :
  - Sử dụng ngôn ngữ tự nhiên và ngôn ngữ toán học
  - Sử dụng ngôn ngữ hình thức
  - Sử dụng ngôn ngữ máy tính

Nguyên lý làm việc

- Nguyên lý làm việc như sau :
  - Tri thức được thu nhận trong ngôn ngữ tự nhiên L →
  - Tri thức được dịch ra trong ngôn ngữ P →
  - Tìm giải pháp suy diễn trong P →
  - Lời giải được dịch ra trong ngôn ngữ L

Biểu diễn tri thức

- Một số phương pháp dùng ngôn ngữ hình thức :
  - Biểu diễn tri thức nhờ các luật sản xuất
  - Biểu diễn tri thức nhờ lôgic
  - Biểu diễn tri thức nhờ mạng ngữ nghĩa
  - Ngoài ra, người ta còn sử dụng :
    - Nhờ bộ ba : đối tượng, thuộc tính và giá trị (O-A-V: Object-Attribute-Value)
    - Nhờ khung (frame)
    - Ngôn ngữ nhân tạo, sự kiện không chắc chắn, v.v...

Biểu diễn tri thức bởi các luật sản xuất

- Hầu hết,các HCG hiện nay đều là các hệ thống dựa trên luật
- Lý do như sau :
  - *Bản chất đơn thể* (modular nature)
    - Có thể đóng gói tri thức
    - Có thể mở rộng HCG một cách dễ dàng
  - *Khả năng diễn giải dễ dàng* (explanation facilities)
    - Vận dụng luật có thể đặc tả chính xác các yếu tố tri thức
    - Dễ dàng diễn giải vấn đề nhờ các tiền đề
    - Rút ra được kết quả mong muốn
  - *Tương tự quá trình nhận thức của con người.*
    - Dựa trên các công trình của Newell và Simon, các luật được xây dựng từ cách con người giải quyết vấn đề
    - Cách biểu diễn luật nhờ IF THEN đơn giản cho phép giải thích dễ dàng cấu trúc tri thức cần trích lọc

Luật (Rule) là gì

- Luật là một kiểu sản xuất dạng :

  LHS → RHS

  - Phần trái LHS (Left Hand Side)
    có nội dung được gọi theo nhiều tên khác nhau, như :
    - tiền đề (antecedent)
    - điều kiện (conditional part)
    - mẫu so khớp (pattern part)
  - Phần sau luật RHS (Right Hand Side) là kết luận
    hay *hậu quả* (consequent)

Luật (Rule) là gì

- Trong một hệ thống dựa trên luật, công cụ suy luận sẽ xác định những luật nào là tiên đề thỏa mãn các sự việc
- Các luật sản xuất thường được viết dưới dạng IF THEN
- Có hai dạng luật :

IF < điều kiện > THEN < hành động >

- Một số HCG có thêm phần hành động (action) bên phần phải của luật :

IF < điều kiện > THEN < kết luận > DO < hành động >

Đặt tên luật

- Tuỳ theo HCG cụ thể mà mỗi luật có thể được đặt tên
- Có dạng Rule: Tên
- Ví dụ :

Rule: Đèn đỏ
IF
    Đèn đỏ sáng
THEN
    Dừng
Rule: Đèn-xanh
IF
    Đèn xanh sáng
THEN
    Đi

- Trong ví dụ trên, Đèn đỏ sáng và Đèn xanh sáng là những điều kiện, hay những khuôn mẫu

Cú pháp của các luật

- A simple rule :

  IF <antecedent>

  THEN <consequent>

- A rule can have multiple antecedents joined by the keywords **AND** (**conjunction**), **OR** (**disjunction**) or a combination of both

IF	<antecedent 1> AND	<antecedent 1> OR
	<antecedent 2>	<antecedent 2> OR
	.	.
	.	.
	.	.
	<antecedent *n*>	<antecedent *n*>
THEN	<consequent>	THEN <consequent>

Ví dụ

- Strategy

  IF the car is dead
  THEN the action is 'check the fuel tank';
  step1 is complete

  IF step1 is complete AND
  the 'fuel tank' is full
  THEN the action is 'check the battery';
  step2 is complete

- Heuristic

  IF the spill is liquid AND
  the 'spill pH' < 6 AND
  the 'spill smell' is vinegar
  THEN the 'spill material' is 'acetic acid'

Một số ví dụ khác

Rule: Điều trị sốt

IF

Bệnh nhân sốt

THEN

Cho uống thuốc Aspirin

Hệ thống chẩn đoán xe máy (OPS5)

IF

Máy xe không nổ khi khởi động

THEN

Dự đoán: Xe bị panne sức nén. Pittong, bạc xéc-măng và lòng xy lanh sai tiêu chuẩn, dễ tạo thành những khe hở nhỏ làm cho pittong không còn kín nên hoà khí không được nén lên đầy đủ.

Xử lý : nên điều chỉnh hoặc thay mới pittong, bạc xéc-măng và lòng xy lanh cho đúng tiêu chuẩn

IF

Máy xe nổ không ổn định, OR
máy xe nổ rồi lại tắt, AND bugi khô

THEN

Dự đoán : Xe đã bị nghẹt xăng.

Xử lý : nên xúc rửa bình xăng và bộ khoá xăng của xe

Ví dụ HCG MYCIN

- MYCIN là hệ thống chẩn đoán bệnh viêm màng não và hiện tượng có vi khuẩn bất thường trong máu (nhiễm trùng)
- Ví dụ một luật của MYCIN :

IF

Tại vị trí vết thương có máu, AND
Chưa biết chắc chắn cơ quan bị tổn thương, AND
Chất nhuộm màu âm tính, AND
Vi khuẩn có dạng hình que, AND
Bệnh nhân bị sốt cao

THEN

Cơ quan có triệu chứng (0.4) nhiễm trùng

Dùng lôgic vị từ biểu diễn tri thức

- Trong lôgic vị từ :
  - Các vị từ thường có chứa hằng, biến hay hàm
  - Người ta gọi các vị từ không chứa biến (có thể chứa hằng) là các *mệnh đề* (preposition)
  - Mỗi vị từ có thể là một *sự kiện* (fact) hay một luật
  - Luật là vị từ gồm hai vế trái và phải được nối nhau bởi một dấu mũi tên ($\rightarrow$)
- Người ta sử dụng các ký hiệu và các phép toán lôgic tác động lên các ký hiệu để thể hiện tri thức và suy luận lôgic

Dùng lôgic mệnh đề biểu diễn tri thức

- Ví dụ :
  - MAN(X), FATHER(X, Y) là các sự kiện
  - MAN(X) → MORTAL(X) là một luật
- Giải thích :
  - MAN(X) : «X là một người»
  - MORTAL(X) : «X chết»

Phát biểu	*Vị từ*
Tom là một người	MAN(tom)
Tom là cha của Mary	FATHER(tom, mary)
Tất cả mọi người đều chết	MAN(X) → MORTAL(X)

Ví dụ dùng lôgic vị từ

- Từ các tri thức sau :
  - Marc có tóc vàng hoe
  - Jean có tóc màu nâu
  - Pierre là cha của Jean
  - Marc là cha của Pierre
  - Jean là cha của René
  - Marc là con của Georges.
  - Giả sử X, Y và Z là những người nào đó
  - Nếu Y là con của X, thì X là cha của Y
  - Nếu X là cha của Z và Z là cha của Y, thì X là ông của Y

Ta có thể biểu diễn thành các sự kiện và các luật như sau :

BLOND (marc)

BROWN (jean)

FATHER (pierre, jean)

FATHER (marc, pierre)

FATHER (jean, rené)

SON (marc, georges)

FATHER (X, Y) ← SON (Y, X)

GRANDFATHER (X, Y) ← FATHER (X, Z), FATHER (Z, Y)

Biểu diễn tri thức nhờ mạng ngữ nghĩa

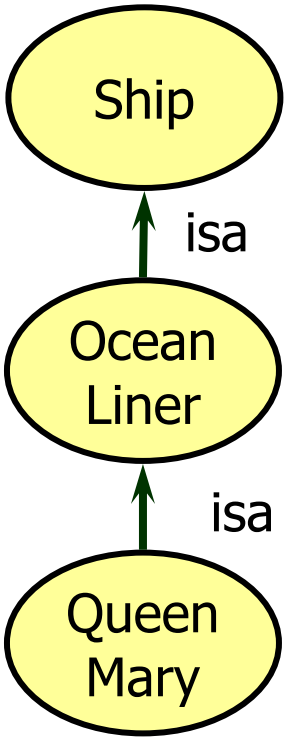
- Mạng ngữ nghĩa là một đồ thị :
  - Các nút (node) dùng để thể hiện :
    - Các đối tượng
    - Thuộc tính của đối tượng
    - Giá trị của thuộc tính
  - Các cung (arc) nối các nút để thể hiện mối quan hệ giữa các đối tượng
- Các nút và các cung đều được gắn nhãn
- Nghĩa sử dụng của một tri thức :
  - Một đường đi trong đồ thị gồm các nút nối các cung

Semantic Nets

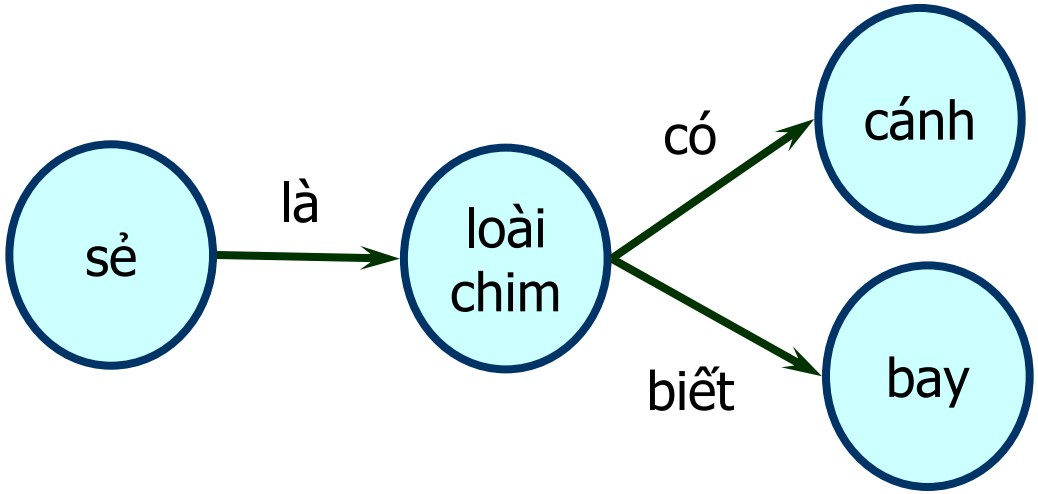
- Semantic net is a knowledge presentation method based on a network structure
- It consists of
  - points called nodes connected by
  - links called arcs
- Nodes - object, concepts, events
- Arcs - relationships between nodes
- Common arcs used for representing hierarchies include *isa* and *has-part*
- Knowledge represented as a network or *graph*

Example:

- The queen mary is an ocean liner
  Every ocean liner is a ship



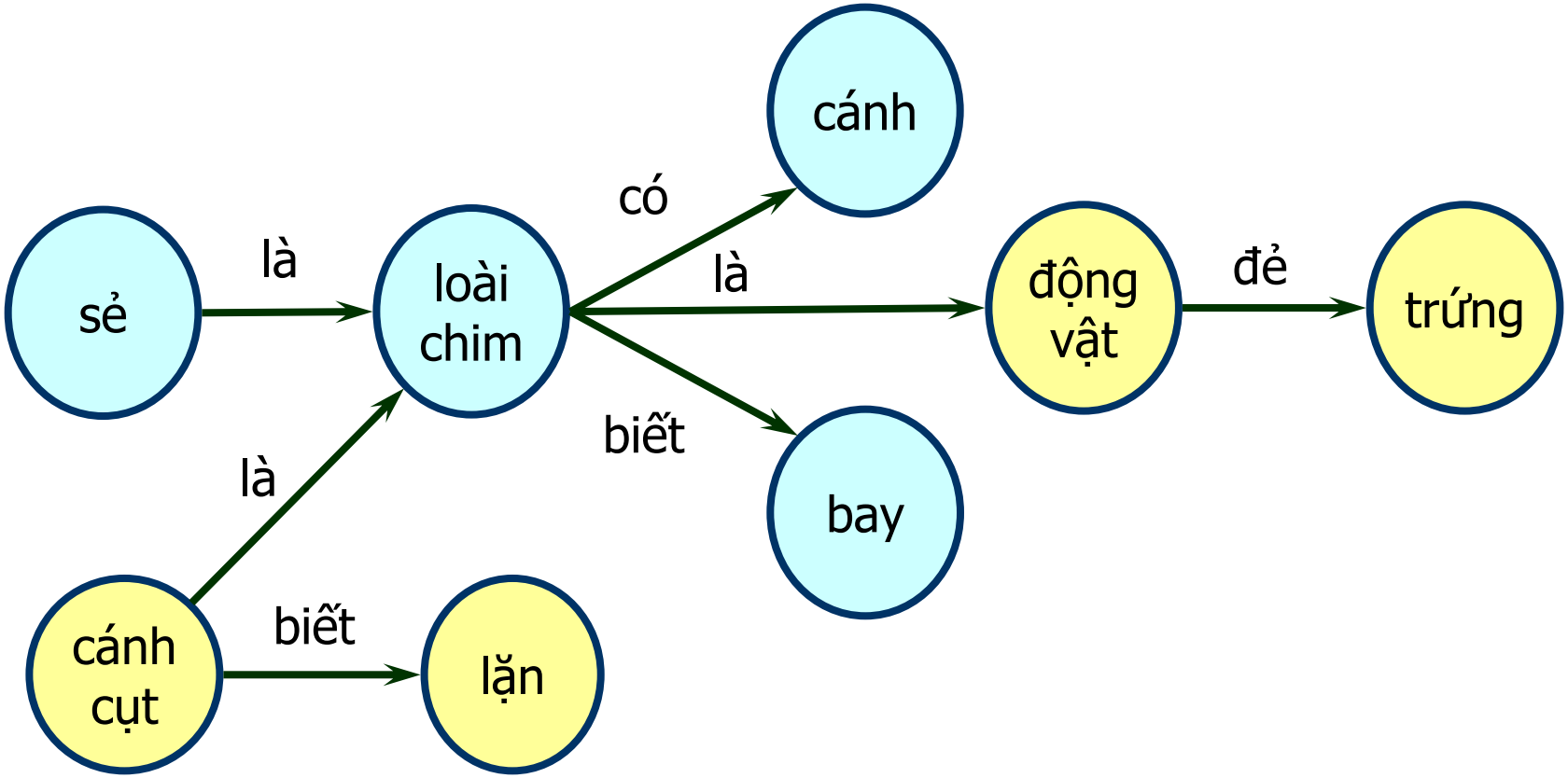
Sẻ là một loài chim có cánh và biết bay



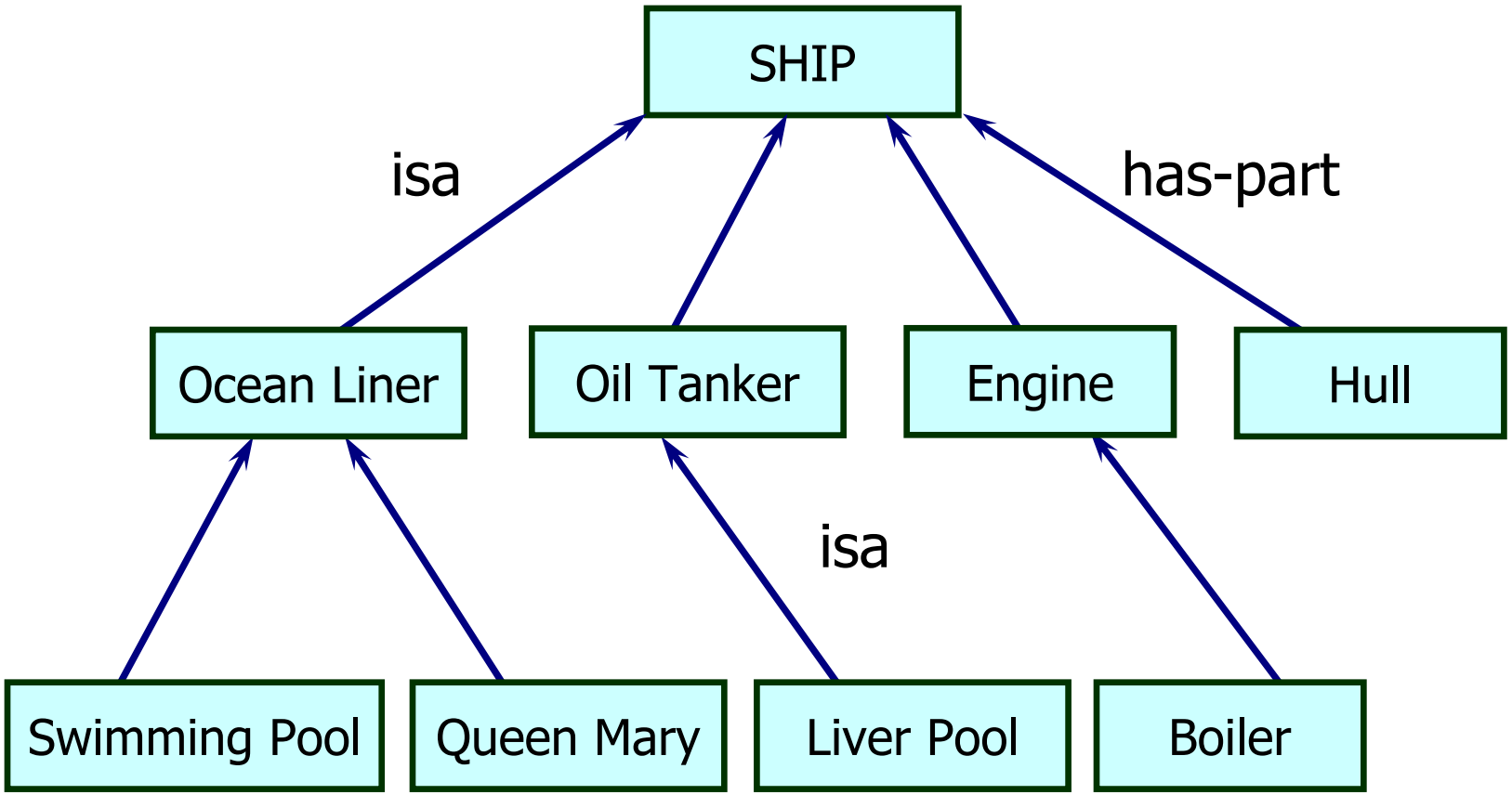
Tính thừa kế của mạng ngữ nghĩa

- Một trong những tính chất quan trọng của mạng ngữ nghĩa là tính thừa kế
- Bằng cách thêm vào đồ thị các nút mới và các cung mới, người ta có thể mở rộng một mạng ngữ nghĩa
- Các nút mới được thêm thể hiện các đối tượng tương tự (với các nút đã có trong đồ thị), hoặc tổng quát hơn
- Khi sử dụng mạng ngữ nghĩa để biểu diễn tri thức, người ta phải xây dựng các phép toán tương ứng

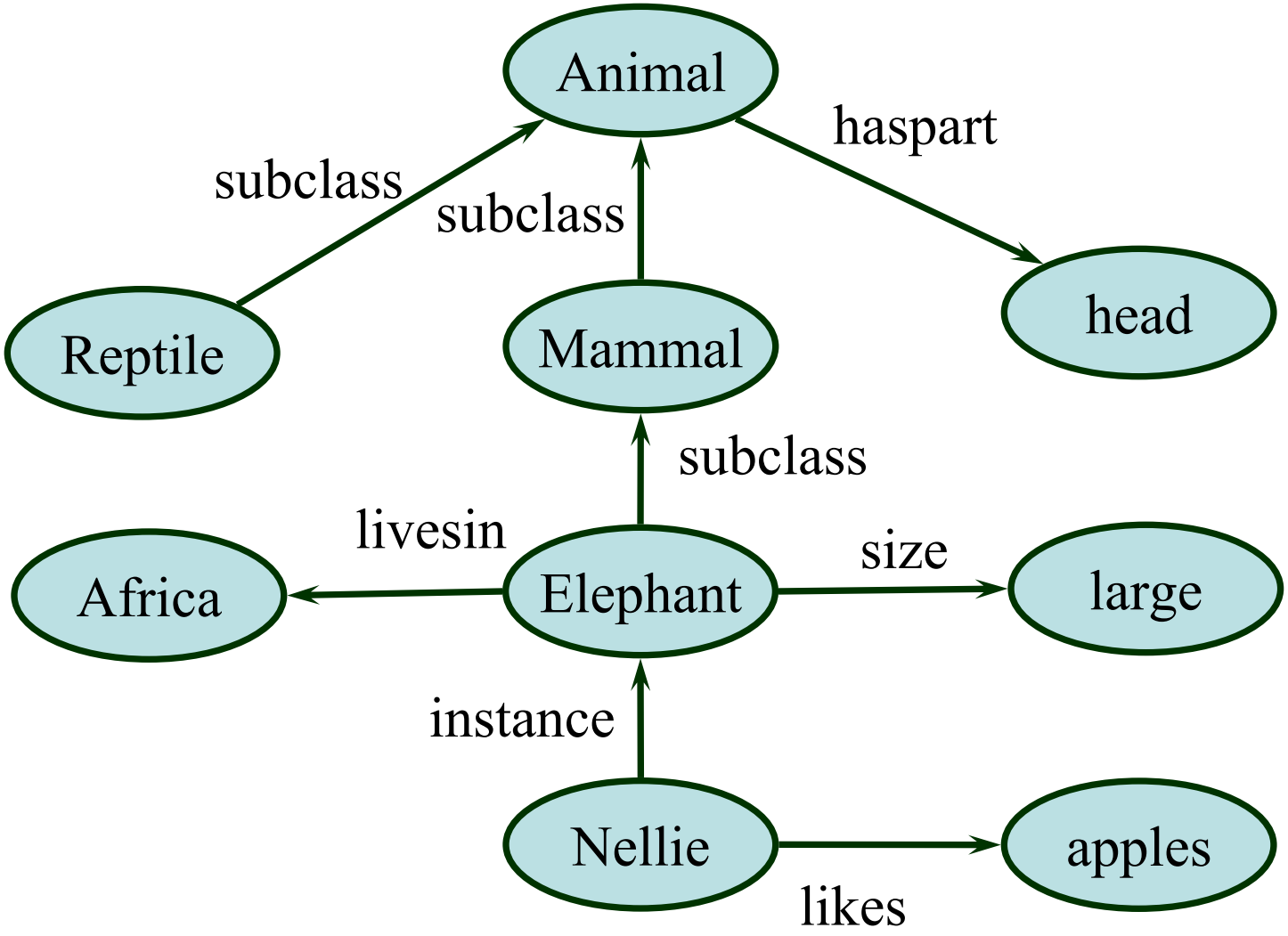
Mở rộng một mạng ngữ nghĩa



Mở rộng một mạng ngữ nghĩa



Mở rộng một mạng ngữ nghĩa



Semantic Networks

- Arity of Relations
  - Unary relations

    Ex: Person(Jim): IS-A link
  - Binary relations

    Ex: Age(Jim, 27 years): Age link
  - N-ary relations

    Ex: Disease(Jim, Mumps, 5 days): By creating a reified
  - disease-relation object with several cases : (patient, diagnosis, duration)

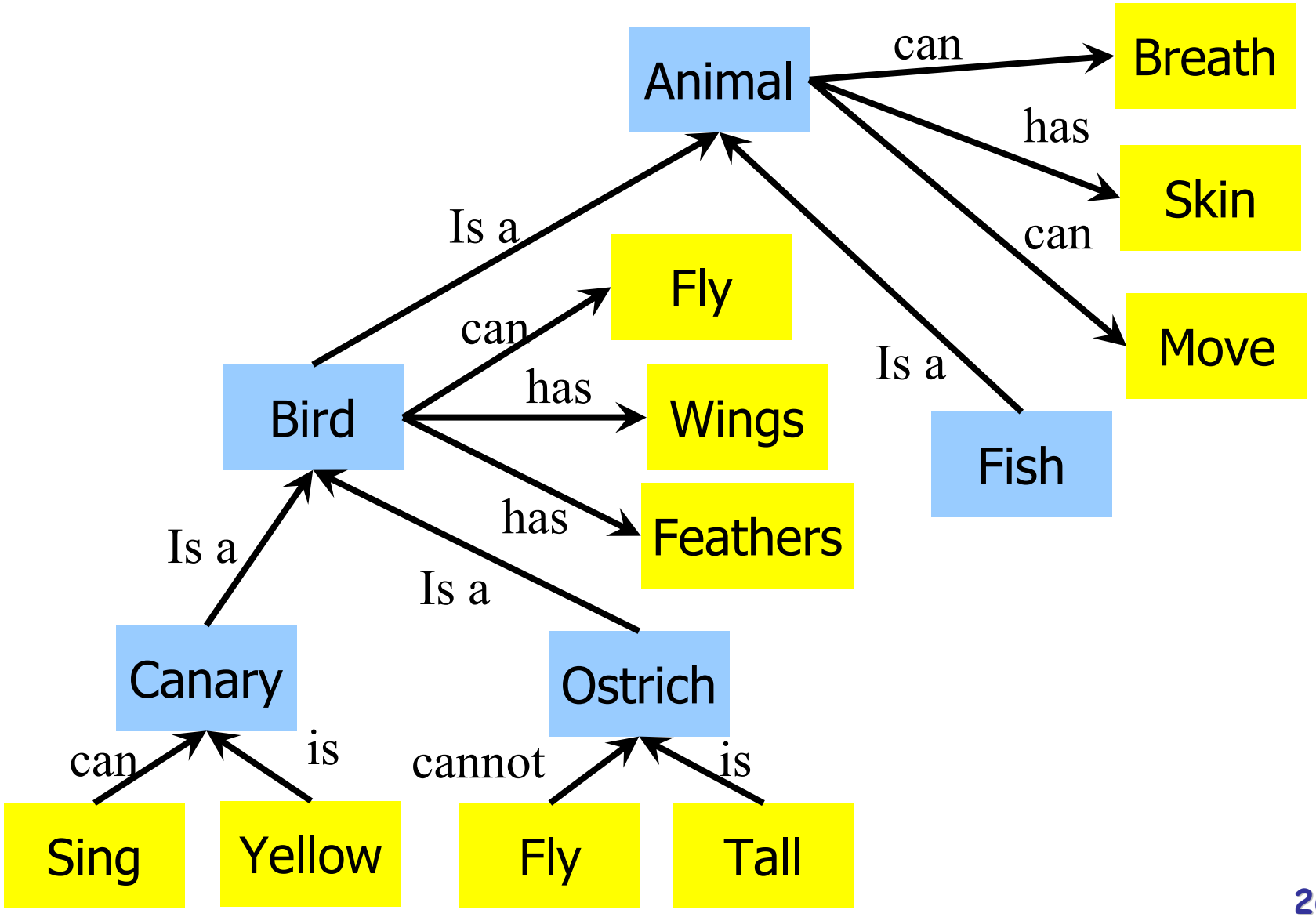
Bài tập tại lớp

- Cho các từ dưới đây, vẽ mạng ngữ nghĩa tương ứng :
  - Các đối tượng :

    Animal, Bird, Breath, Skin, Move, Fish, Fly, Wings, Feathers, Ostrich, Tall, Yellow, Canary, Sing
  - Các quan hệ :

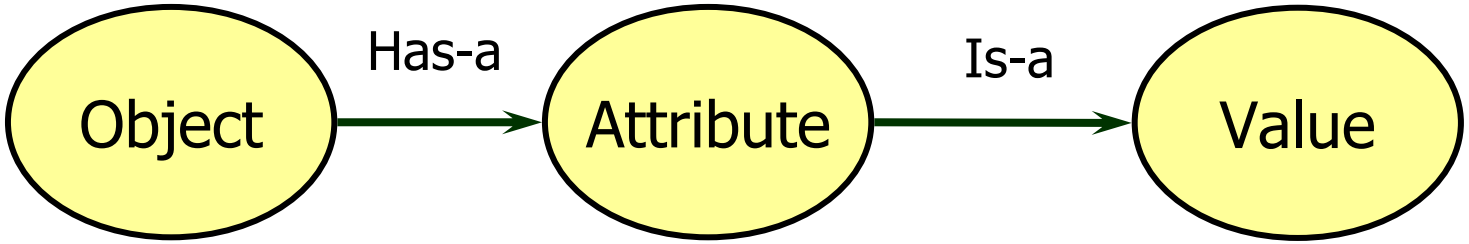
    Can, Cannot, Has, Is, Is-a

Semantic Network Representation



Biểu diễn tri thức nhờ bộ ba O-A-V

- Bộ ba O-A-V (Object-Attribute-Value) :
  đối tượng, thuộc tính và giá trị
  cách mô tả các mạng ngữ nghĩa
  - can be used to characterize the knowledge in a semantic net
  - quickly leads to huge tables



Ví dụ : OAV Table

Object	Attribute	Value
Beluga Whale	Dorsal Fin	No
Beluga Whale	Tail Fin	No
Blue Whale	Tail Fin	Yes
Blue Whale	Dorsal Fin	Yes
Blue Whale	Size	Very Large

Object	Attribute	Value
Astérix	profession	warrior
Obélix	size	extra large
Idéfix	size	petite
Panoramix	wisdom	infinite

Biểu diễn tri thức nhờ khung

- Khi phạm vi bài toán được mở rộng ra :
  - Số nút và số cung trong mạng tăng lên
  - Mạng ngữ nghĩa trở nên phức tạp hơn
- Semantic nets → Frame :
  - Trong trường phức tạp như vậy, người ta sử dụng khung frame
- Giới thiệu khung (frame) :
  Nhóm các thuộc tính, giá trị của các đối tượng
- Hệ thống khung :
  - Nhóm các khung có quan hệ với nhau.
  - Quan hệ có thể là giá trị của một thuộc tính trong frame này là giá trị của một frame khác
- Người ta xâ dựng các thủ tục suy diễn sử dụng khung

Biểu diễn tri thức nhờ Frames

- Frames were the next development, allowing more convenient "packaging" of facts about an object
- Frames look much like modern classes, without the methods:

```
mammal:
  subclass: animal

elephant:
  subclass: mammal
  size: large
  haspart: trunk

Nellie:
  instance: elephant
  likes: apples
```

- We use the terms "slots" and "slot values" (filler)

Frames

- Frames often allowed you to say
  which things were just typical of a class,
  and which were definitional, so couldn't be overridden
- Using an asterix to denote typical values:

```
Elephant:
   subclass: mammal
   haspart: trunk
   * colour: grey
   * size: large
```

- Frames also allowed multiple inheritance
  (Nellie is an Elephant and is a circus animal)
- Introduces problems in inheritance

Simple Frame Example

Thuộc tính

Giá trị

Slot Name	*Filler*
name	Astérix
height	small
weight	low
profession	warrior
armor	helmet
intelligence	very high
marital status	presumed single

Representing Knowledge in Frames

- Frame Architecture
  - A record-like data structure for representing stereotypical knowledge about some concept or object (or a class of objects)
  - A frame name represents a stereotypical situation/object/process
  - Attributes or properties of the object also called slot
  - Values for attributes called fillers, *facets* provide additional control over fillers

Frame Architecture

Frame Name: Object 1

Class: Object 2

Properties:

Property 1	Value 1
Property 2	Value 2
…	…
…	…

Types of Frames

- Class Frame
  - Represents general characteristics of common objects
  - Define properties that are common to all objects within class
  - Static & dynamic property
- Static:
  - Describes an object feature whose value does not change
- Dynamic:
  - Feature whose value is likely to change during operation

Mô hình kiểu của khung

Frame Name: Bird

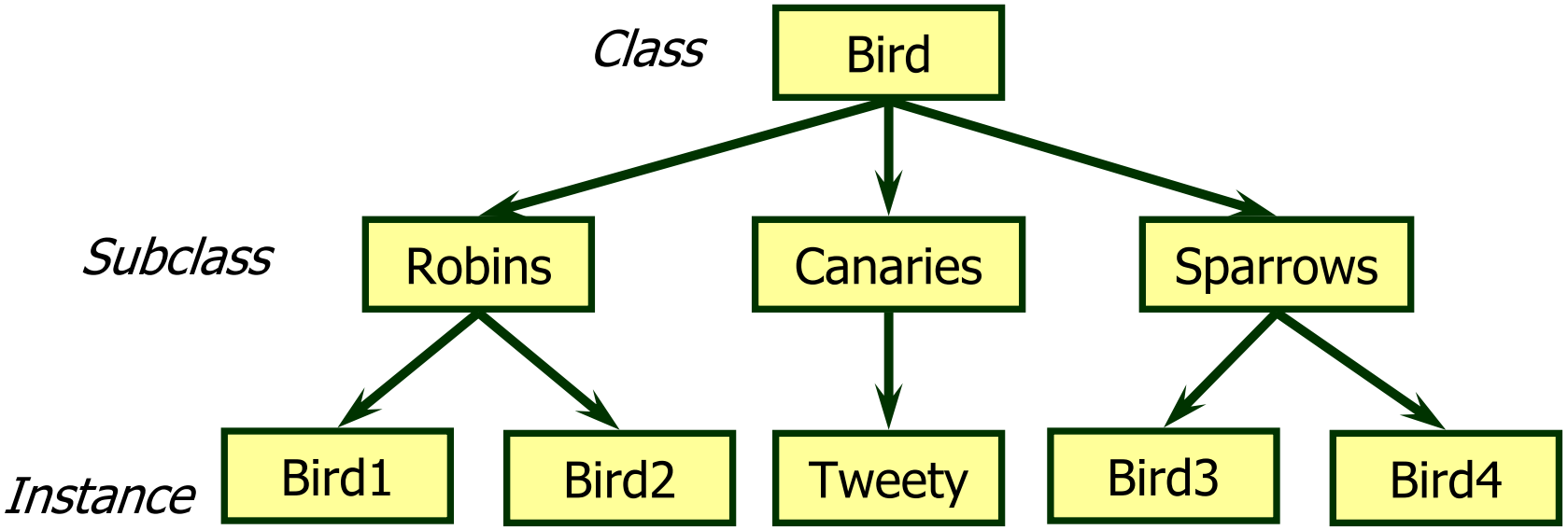
Class: Animal

Properties:

Color	Unknown
Eats	Worms
No._Wings	2
Flies	True
Hungry	Unknown
Activity	Unknown

Subclass Frame

- Represents subsets of higher level classes or categories
- Creates complex frame structures
- Class relationships



Instance Frame

- Represents specific instance of a class frame
- Inherits properties & values from the class
- Able to change values of properties & add new properties

Frame Name: Tweety

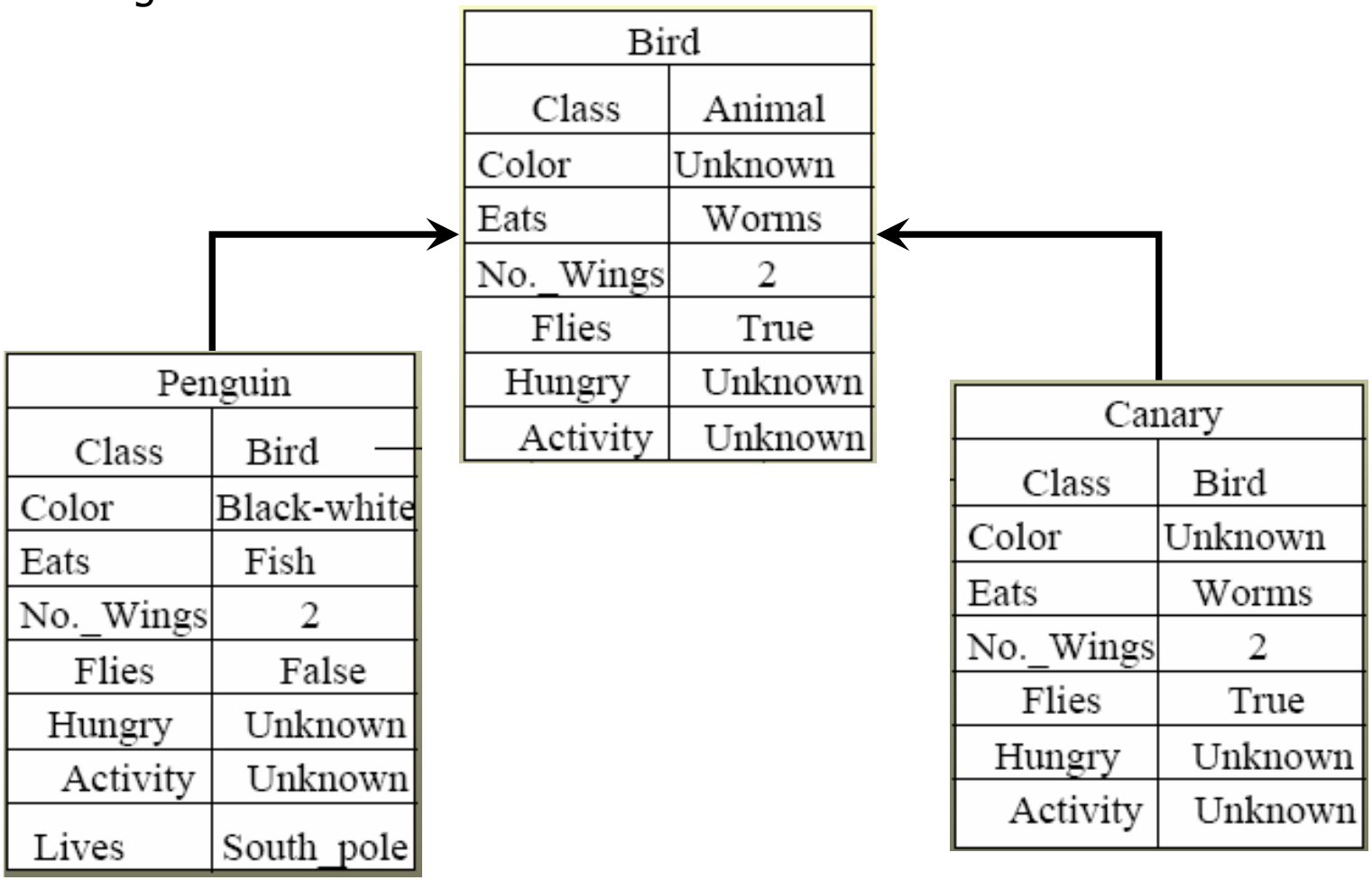
Class: Bird

Properties:

Color	Yellow
Eats	Worms
No._Wings	1
Flies	False
Hungry	Unknown
Activity	Unknown
Lives	Cage

Frame Inheritance

- Instance frame inherits information from its subclass frame and also its class
- Inheritance of behavior, facet
- Ease coding & modification of information



Bird	
Class	Animal
Color	Unknown
Eats	Worms
No._Wings	2
Flies	True
Hungry	Unknown
Activity	Unknown

Penguin	
Class	Bird
Color	Black-white
Eats	Fish
No._Wings	2
Flies	False
Hungry	Unknown
Activity	Unknown
Lives	South_pole

Canary	
Class	Bird
Color	Unknown
Eats	Worms
No._Wings	2
Flies	True
Hungry	Unknown
Activity	Unknown

Frames and procedures

- Frames often allowed slots to contain procedures
- So.. Size slot could contain code to run to calculate the size of an animal from other data
- Sometimes divided into “if-needed” procedures, run when value needed, and “if-added” procedures, run when a value is added (to update rest of data, or inform user)
- So.. Similar, but not quite like modern object-oriented languages

Overview of Frame Structure

- Two basic elements: *slots* and *facets* (fillers, values, etc.);
- Typically have parent and offspring slots
  - Used to establish a property inheritance hierarchy (e.G., Specialization-of)
- Descriptive slots
  - Contain declarative information or data (static knowledge)
- Procedural attachments
  - Contain functions which can direct the reasoning process (dynamic knowledge) (e.G., "Activate a certain rule if a value exceeds a given level")
- Data-driven, event-driven (bottom-up reasoning)
- Expectation-drive or top-down reasoning
- Pointers to related frames/scripts - can be used to transfer control to a more appropriate frame

[Rogers 1999]

Usage of Frames

- Filling slots in frames
  - Can inherit the value directly
  - Can get a default value
  - These two are relatively inexpensive
  - Can derive information through the attached procedures (or methods) that also take advantage of current context (slot-specific heuristics)
  - Filling in slots also confirms that frame or script is appropriate for this particular situation

[Rogers 1999]

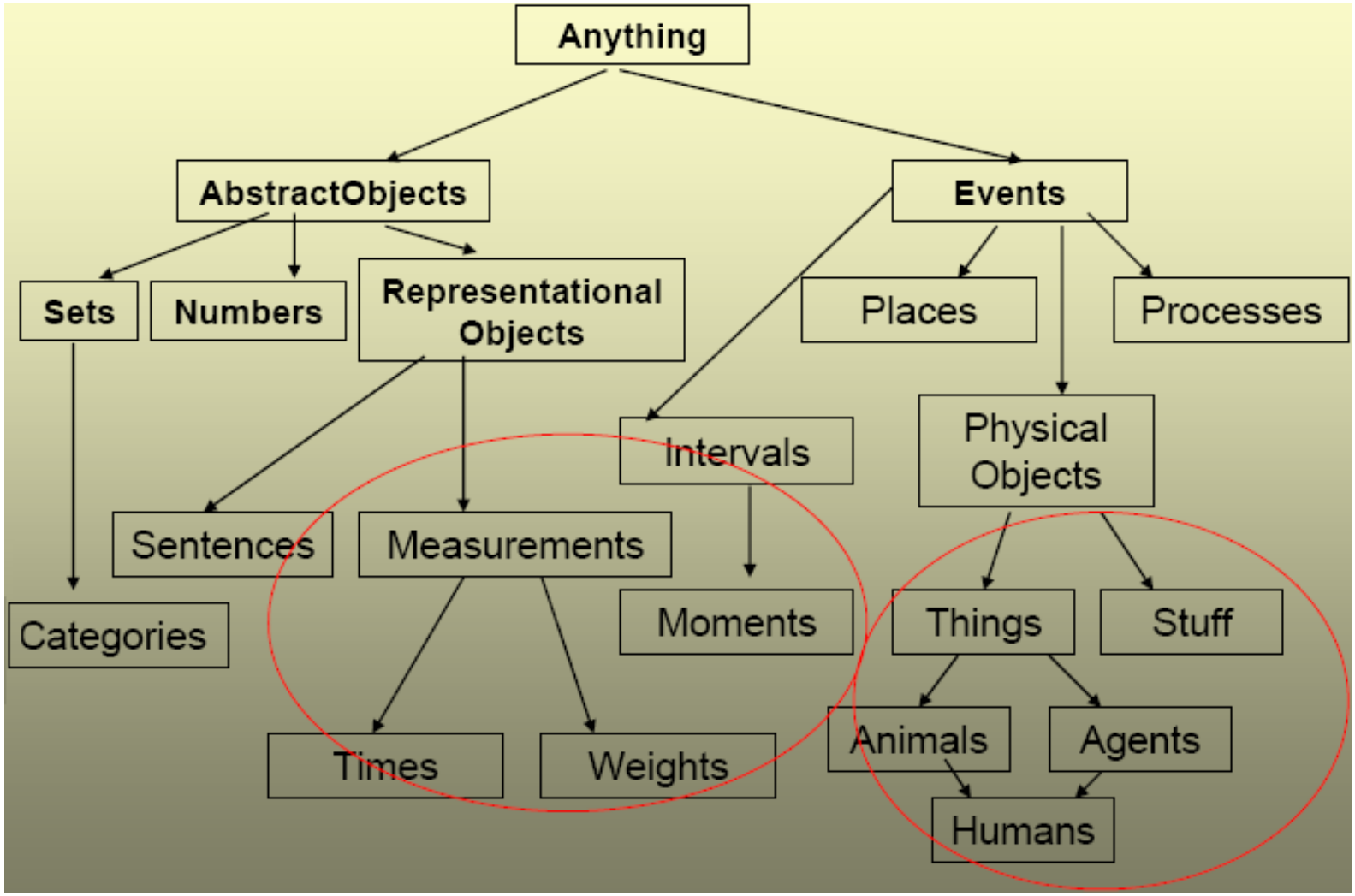
Problems with Frames

- Negation cannot be represented
  - Jim does not have pneumonia
- Disjunction cannot be represented naturally
  - Jim has Mumps or Rubella
- Qualification is not a part of the language
  - All of Jim's diseases are infectious
- => Thus, procedural attachments are often added

Biểu diễn tri thức nhờ ngôn ngữ nhân tạo

- Thực tế, ngôn ngữ tự nhiên :
  - Là phương cách thuận tiện nhất để giao tiếp với một HCG
  - Không những đối với người quản trị hệ thống (tư cách chuyên gia)
  - Mà còn đối với NSD cuối
- Hiện nay đã có những HCG có khả năng đối thoại trên ngôn ngữ tự nhiên (thông thường là tiếng Anh) nhưng chỉ hạn chế trong một số lĩnh vực chuyên môn

A Quick Ontological View



Hệ chuyên gia (Expert System)

PGS.TS. Phan Huy Khánh

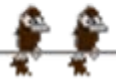
khanhph@vnn.vn

Chương 2
Biểu diễn tri thức nhờ logic vị từ bậc một
2.2

Chương 2
Biểu diễn tri thức nhờ logic vị từ bậc một

⌘Phần 2.2 :

- Khái niệm lôgic
- Lôgic mệnh đề



The 4 Color Theorem

- In 1879, Kempe produced a famous proof of the 4 color theorem:
  - Using only 4 colors
  - Any map of countries can be colored in such a way that no 2 bordering countries have the same color
- In 1890, Heawood showed:
  - The proof not to be a proof at all!
- When is a proof a proof, and when is it not a proof?
- Logic to the rescue!

What is the logic?

- Logic is the science of reasoning, proof, thinking, or inference
- Logic allows us to analyze a piece of reasoning and determine whether it is correct or not
- To use the technical terms, we determine whether the reasoning is valid or invalid
- When people talk of logical arguments, though, they generally mean the type being described here

Logic

- Logic is the study of reasoning
- In particular:
  - Logic studies the conditions under which we can say that a piece of reasoning is *valid*
  - I.e. that something (the conclusion) can be said to *follow from* something else (the premises, givens, assumptions)

- Ontology (ont = 'to be'; logica = 'word'):
  kinds of things one can talk about in the language

Arguments in Logic

- What is an Argument?
  - "An argument is a connected series of statements intended to establish a proposition"
- An argument refers to the formal way facts and rules of inferences are used to reach valid conclusions
- The process of reaching valid conclusions is referred to as logical reasoning

Logic in general

- A logic is a formal system of representing knowledge
- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences (statements) in the language
- Semantics define the "meaning" of sentences
  - i.e., define truth of a sentence in a world
- Proof theory
  - How conclusions are drawn from a set of statements

Deduction and Induction

- If the conclusion *has to be* true assuming the truth of the premises, we call the reasoning deductive
- If the conclusion is merely *more likely to be* true than false given the truth of the premises, we call the reasoning inductive
- Logic studies both deduction and induction, but does tend to focus on deduction, especially formal logic

Normative and Descriptive Theories of Reasoning

- Psychology of reasoning is a scientific study of how humans reason:
  - What do humans infer from what?
  - What is the mechanism behind human reasoning?
- As such, psychologists come up with *descriptive* theories of reasoning: *hypotheses* as to how humans reason based on *empirical* studies.
- Logicians, however, try to come up with *normative* theories of reasoning:
  - What *actually* follows from what?
- Question: But if not empirical, what is the basis for such theories? (Human!) reason alone?

Implication and Truth

- Logic tells us about implication, not truth
- Example:
  - "All flurps are toogle, but not all flems are toogle, so not all flems are flurps" is perfectly logical, but tells us nothing about what-is-the-case.
- One exception:
  - Implication itself can be seen as a kind of (necessary) truth
  - So, logic can tells us that certain statements of the form "If <premises> then <conclusion>" are necessarily true (i.e. true in all possible worlds), and hence true in our world as well

Logic and Science

- Of course, if I do know that my premises are true, then if the reasoning is (deductively) valid I know the conclusion to be true as well
- But that's just science: science combines observation (facts) with logic (reasoning), to get to truth (laws of physics, chemistry, etc)
- Of course, scientific reasoning is inherently *inductive*: a finite set of data is always compatible with multiple theories
- Hence: scientific *theories* can change over time.

Logic and Mathematics

- Most of what I just said for logic is true for mathematics as well!
  - Scientists use mathematics to help figure out (calculate, compute, etc) what-is-the-case but mathematics alone does not tell us what-is-the-case
  - Like logic, mathematical theorems are proven *from* a set of definitions or axioms: if those axioms or definitions don't apply to our world, then the theorem doesn't say anything about our world either.
  - The only thing we can claim to be certain of is a statement of the form "If <bunch of definitions/axioms> then <some claim>".
  - So, theorems like "There is no greatest prime number" are really expressions of "If we define 'number' to be ..., and 'prime' as ... and 'greater than' as ..., then there is no greatest prime number."

Further Similarities Between Logic and Mathematics

- Both logic and mathematics have been around for thousands of years
- Both logic and mathematics study abstractions that can be applied to any subject matter
- Formal logic is probably best seen as a branch of mathematics
- Mathematics can be applied to formal logic (mathematical logic)
- Formal logic can be applied to mathematics (theorem proving)

Formal Logic

- We can determine that "All flurps are toogle, but not all flems are toogle, so not all flems are flurps" is a valid inference because of the abstract form of the reasoning: "All P's are Q's, but not all R's are Q's, so not all R's are P's"
- Formal logic is just that: studying the validity of reasoning by looking at its abstract form:
  - Just as in mathematics:
    - 1) expressions of abstract symbols are assigned the objects of study, and
    - 2) by manipulating these expressions of abstract symbols, we can figure something out about these objects

Little History of Formal Logic

- Formal logic goes back at least to Aristotle, probably earlier
- In Medieval Times work was being done on categorical syllogisms like the one on previous page (that one would be classified as AOO-2)
- 'Modern' formal logic was developed in mid 19th century by people like Georges Boole and Augustus DeMorgan
- They developed the system of propositional or truth-functional logic
- The much more powerful system of first-order logic (or predicate logic or quantificational logic) was completed by the turn of the 20th century
- Many other systems of logic have been developed since; just as with mathematics, different systems have different applications

Truth-Functional Logic

- Applies to reasoning dealing with compound sentences built from truth-functional operators like ‘and’, ‘or’, ‘not’, and ‘if … then’.
- An operator is truth-functional in that the truth-value of a sentence like “P and Q” is a function of the truth-values of the sentences P and Q

Non-standard logics

1. Categorical logic
2. Combinatory logic
3. Conditional logic
4. Constructive logic
5. Cumulative logic
6. Deontic logic
7. Dynamic logic
8. Epistemic logic
9. Erotetic logic
10. Free logic
11. Fuzzy logic
12. Infinitary logic
13. Intensional logic
14. Intuitionistic logic
15. Linear logic
16. Many-valued logic
17. Modal logic
18. Non-monotonic logic
19. Paraconsistent logic
20. Partial logic
21. Prohairetic logic
22. Quantum logic
23. Relevant logic
24. Stoic logic
25. Substance logic
26. Substructural logic
27. Temporal (tense) logic

⌘ In short: a lot!

Propositional Logic

- A relatively simple framework for reasoning
- Can be extended for more expressiveness at the cost of computational overhead
- Important aspects
  - Syntax
  - Semantics
  - Validity and inference
  - Models
  - Inference rules
  - Complexity
- Principles of propositional logic
  - Sentences, syntax, semantics, inference
- But major limitations of propositional logic

Ví dụ

Mệnh đề lôgích	*Giải thích*
y > x + 1	Tuỳ theo giá trị của x và y mà có giá trị đúng hoặc sai. Chẳng hạn x=1 và y=3 thì có giá trị đúng
Hôm nay trời mưa !	Đúng nếu tại thời điểm nói ra trời mưa thật, sai nếu không phải
2 + 3 = 5	Luôn luôn có giá trị đúng
Luânđôn là thủ đô của nước Đức	Luôn luôn có giá trị sai
Hôm nay là ngày mấy ?	Câu hỏi không phải mệnh đề
Mời anh vào đây !	Câu mệnh lệnh cũng không phải là một mệnh đề, v.v...

Examples

- 2 + 2 = 4
  - Is a proposition which true
- The moon is made of cheese
  - Is a proposition which is false
- It will rain tomorrow
  - Is a proposition
- *This statement is false*
  - *Is not a proposition*
- *Vote for Mickey Mouse*
  - *Is not a proposition*

Syntax

- Symbols
  - Logical constants: true, false /* T, F | 1, 0 | Yes, No … */
  - Propositional symbols: P, Q, R, …
  - Logical connectives
    - Conjunction: $\wedge$
    - Disjunction: $\vee$
    - Negation: $\neg$
    - Implication: $\rightarrow$, (or $\Rightarrow$)
    - Equivalence: $\leftrightarrow$, (or $\Leftrightarrow$)
  - Parentheses for grouping: (,)
- Công thức chỉnh (CTC)
  Well-Formed formulas (wffs or sentences): w_1, w_2
  - Constructed from simple sentences: P, Q, R, …
  - Conjunction, disjunction, implication, equivalence, negation
    - $(P \wedge Q) \rightarrow \neg P$
    - $(\neg P \vee Q) \vee R \rightarrow S$

BNF Grammar Propositional Logic

<Sentence> ::= <AtomicSentence> | <ComplexSentence>

<AtomicSentence> ::= True | False | P | Q | R | ...

<ComplexSentence> ::= (<Sentence>) |
<Sentence> <Connective> <Sentence> |
¬ <Sentence>

<Connective> ::= ∧ | ∨ | → | ↔

Ambiguities are resolved through precedence ¬ ∧ ∨ → ↔ or parentheses

e.g. ¬ P ∨ Q ∧ R → S is equivalent to (¬ P) ∨ (Q ∧ R)) → S

BNF Grammar Propositional Logic

<Sentence> ::= <AtomicSentence> | <ComplexSentence>

<AtomicSentence> ::= True | False | P | Q | R | ...

<ComplexSentence> ::= (<Sentence>) |
<Sentence> <Connective> <Sentence> |
¬ <Sentence>

<Connective> ::= ∧ | ∨ | → | ↔

Ambiguities are resolved through precedence ¬ ∧ ∨ → ↔ or parentheses

e.g. ¬ P ∨ Q ∧ R → S is equivalent to (¬ P) ∨ (Q ∧ R)) → S

Semantics

- Interpretation of the propositional symbols and constants
  - Symbols can stand for any arbitrary fact
    - Sentences consisting of only a propositional symbols are satisfiable, but not valid
    - The value of the symbol can be true or false
    - Must be explicitly stated in the model
  - The constants True and False have a fixed interpretation
    - True indicates that the world is as stated
    - False indicates that the world is not as stated
- Specification of the logical connectives
  - Frequently explicitly via truth tables

Applications

- User defines the **semantics** of each propositional symbol:
  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then $\neg$S is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences,
    then (S $\vee$ T), (S $\wedge$ T), (S $\rightarrow$ T), and (S $\leftrightarrow$ T) are sentences
  - A sentence results from a finite number of applications of the above rules
- Eg.
  - Q $\rightarrow$ P means: "If it is humid, then it is hot"
  - (P $\wedge$ Q) $\rightarrow$ R means: "If it is hot and humid, then it is raining"

Bài tập tại lớp

- Chuyển các câu sau sang lôgic mệnh đề :

1. Quạ tắm thì ráo, sáo tắm thì mưa
2. Ông ăn chả, bà ăn nem
3. Tại anh, tại ả, tại cả đôi bên
4. (Chơ ai) đi mô rồi cũng nhớ về Hà Tịnh
5. Bởi chưng bác mẹ em nghèo
   Cho nên em phải băm bèo thái rau
6. Trăng khoe trăng tỏ hơn đèn
   Cớ sao trăng phải chịu luồn đám mây
7. Các chú học trò con
   Đang lúc tuổi còn non
   Các chú phải chăm học
   Có học mới nên khôn

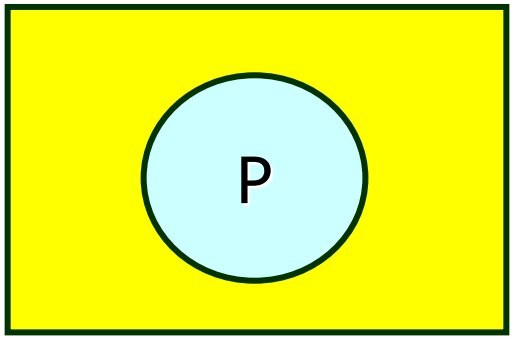
Mần reng tui có thể ?

Dạ, đây :

1. **Lập các câu đơn (NNTN) theo tình huống/sự kiện**
2. **Đặt tên các biến MĐề**
3. **Tìm các phép nối lôgic tương ứng**
4. **Kiểm tra tính hợp thức**
5. **Nhận kết quả**

English Translations of Negation

- Assume:
  - P represents the fact Penguins eat fish
  - Q represents the fact I have three children
- Then Negation:
  - ¬Q: Penguins do not like fish
  - ¬Q: I do not have three children
- Other Notation: ~P, ~Q



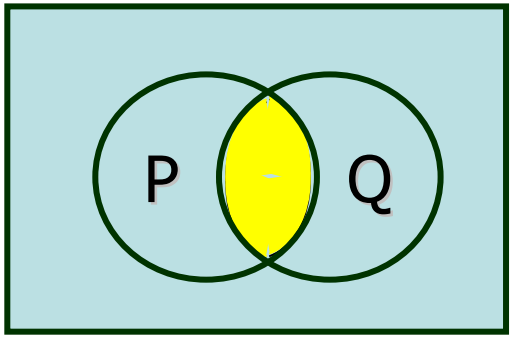
English Translations of Conjunction

- Assume:

  P represents the fact This subject is boring

  Q represents the fact I am tired

- Then Conjunction $P \wedge Q$:
  - This subject is boring and I am tired
  - This subject is boring although I am tired
  - This subject is boring but I am tired



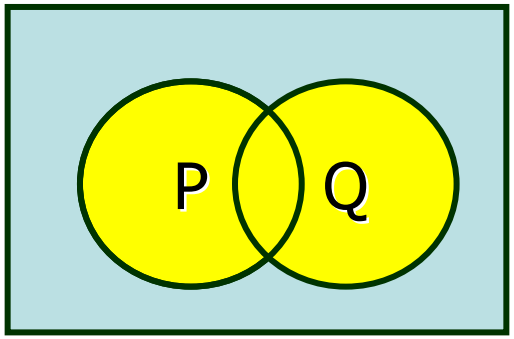
English Translations of Disjunction

- Assume:

  P represents the fact This subject is boring

  Q represents the fact I am tired

- Disjunction $P \vee Q$
  - P : Sue is a football player.
  - Q: Bob is lazy
  - $P \vee Q$: Sue is a football player or Bob is lazy
  - Note: Disjunction is inclusive

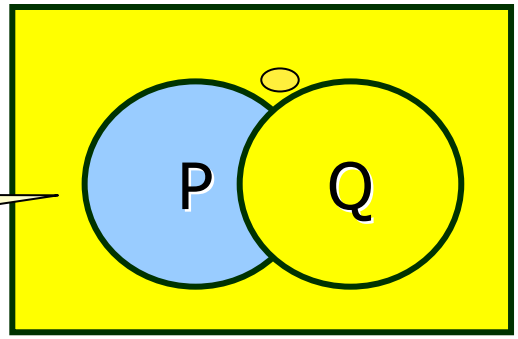


English Translations of Conditional

- Assume:
  - P represents the fact **It is Tuesday**
  - Q represents the fact **We are in Belgium**
- Then Conditional $P \rightarrow Q$
  - If it is Tuesday then we are in Belgium
  - It's being Tuesday implies we are in Belgium
  - It's Tuesday only if we are in Belgium
  - It's being Tuesday is sufficient for us to be in Belgium

Giải thích
vì reng rứa ?

$P \rightarrow Q \equiv \neg P \vee Q$



If_Then Variations

- If-then statements appear in various forms in practice
- More Translations $P \rightarrow Q$:
  - If P then Q
  - P implies Q
  - P only if Q
  - P is sufficient for Q
  - Q is necessary for P
  - Q provided that P
  - Q if P
  - If P, Q
  - Q whenever P
  - It is necessary for P that Q

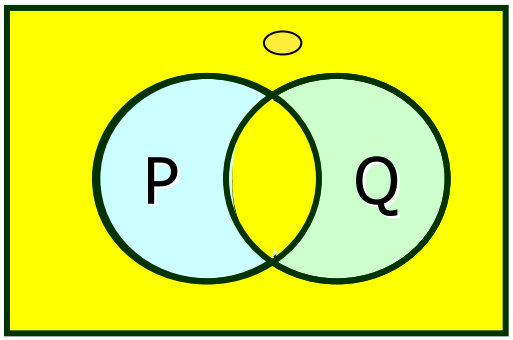
English Translations of Definition $\neg P \rightarrow Q$

- Translations
  - If not P then Q
  - Unless P, Q
  - Q, unless P
- Examples
  - Unless Max is at home, Claire won't get the message
    $\neg HOME(max) \rightarrow \neg GETSMESSAGE(claire)$
  - b is cube, unless it is large
    $\neg LARGE(b) \rightarrow CUBE(b)$

English Translations of Biconditional $P \leftrightarrow Q$

- Translations
  - P if and only if Q
  - P just in case Q
  - P is a necessary and sufficient condition for Q
- Equivalencies
  - P and Q are sufficient and necessary for each other
  - $P \leftrightarrow Q$
  - $(P \rightarrow Q) \wedge (Q \rightarrow P)$
  - $(P \wedge Q) \vee (\neg P \wedge \neg Q)$

Giải thích
vì reng rứa ?



Mary's Exam Example

- Assume:

  Today Mary has a Law exam or a Computer Science exam or both
  She doesn't have a Law exam
  Therefore she must have a Computer Science exam

- Then

  L: Mary has a Law exam today

  C: Mary has a Computer Science exam today

  Premises: $L \vee C$, $\neg L$

  Conclusion: C

More examples of PL

- Assume
  - P represents the fact "*Penguins eat fish*"
  - Q represents the fact "*Penguins like fish*"
- then
  - $P \wedge Q$ Penguins eat fish and penguins like fish
  - $P \vee Q$ Penguins eat fish or penguins like fish
  - $Q \rightarrow P$ Penguins eat fish therefore penguins like fish
  - $P \leftrightarrow Q$ Penguins eat fish therefore penguins like fish and penguins like fish therefore penguins eat fish
- Fitness Example

  S: If I exercise then I will get fit, and I do exercise, so I will get fit.

  E: I do exercise.

  F: I will get fit.

  S: $(E \rightarrow F) \wedge E \rightarrow F$

Some terms

- The meaning or semantics of a sentence determines its interpretation
- Given the truth values of all symbols in a sentence, it can be "evaluated" to determine its truth value (True or False)
- A valid sentence or tautology
  - A sentence that is True under all interpretations (equivalent with True)
  - Also called: "logically true"
  - Example: $P \vee \neg P$, "It's raining or it's not raining"

More terms

- An inconsistent sentence or contradiction
  - A sentence that is False under all interpretations (equivalent with False)
  - Also called: "logically false"
  - Example: $P \wedge \neg P$, "It's raining and it's not raining"
- As with equivalence: look at the truth tables
- P entails Q, written P |= Q, means that whenever P is True, so is Q
- In other words, all models of P are also models of Q

Truth Tables

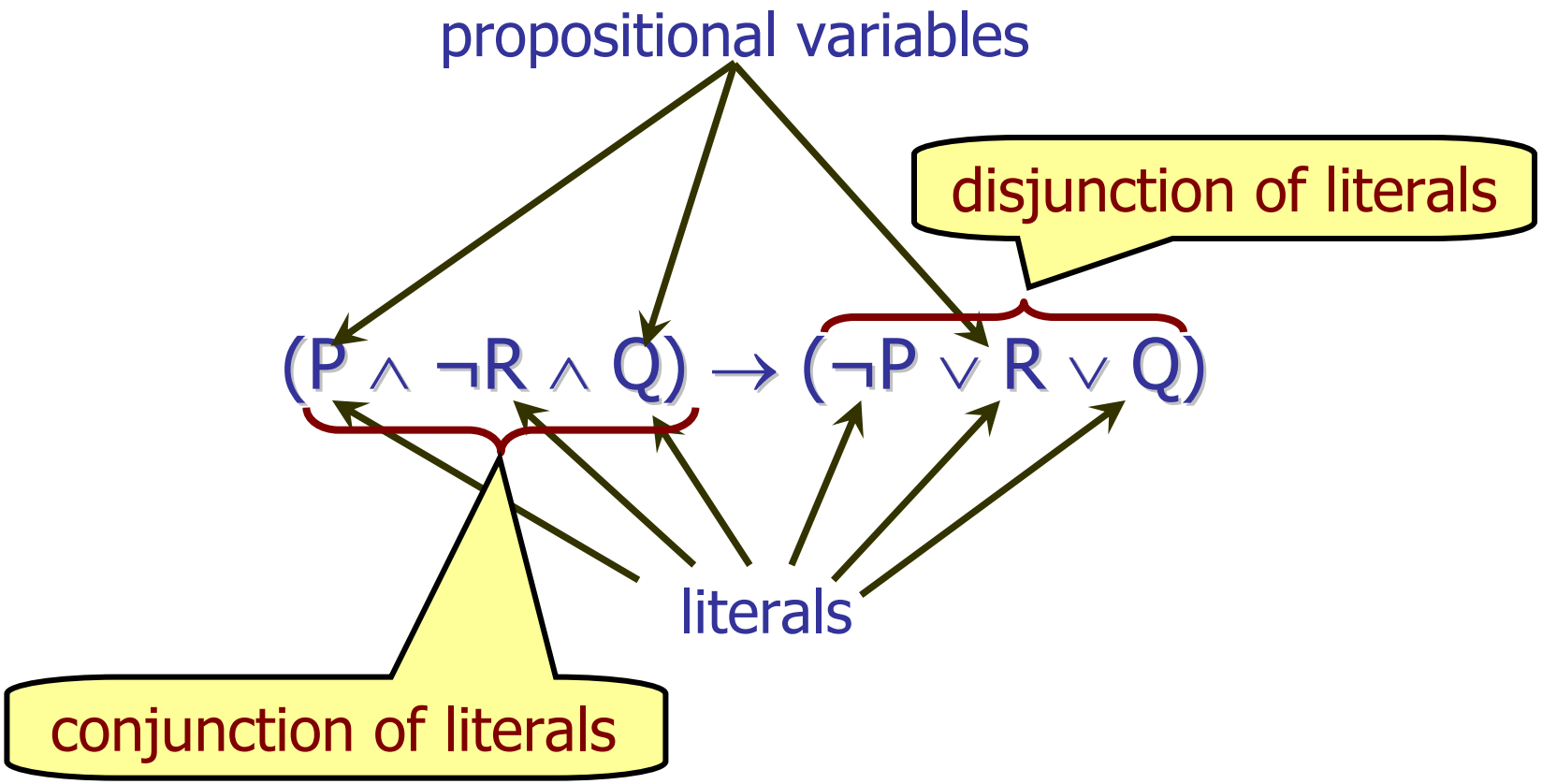
Truth Tables for Connectives

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

- One row for each possible combination of truth values for the symbols in the sentence
- The final value must be true for every sentence
- A variation of the model checking approach
- Not very practical for large sentences

Terminology

- A literal is either a propositional variable, or the negation of a propositional variable



Special Forms

- A formula is in Disjunctive Normal Form (DNF) if it is a disjunction

$$D_1 \vee D_2 \vee \ldots \vee D_n$$

where each D_i is a conjunction of literals

- A formula is in Conjunctive Normal Form (CNF) if it is a conjunction

$$C_1 \wedge C_2 \wedge \ldots \wedge C_n$$

where each C_i is a disjunction of literals

Example

- Disjunctive Normal Forms

  $(\neg P \wedge \neg R) \vee (\neg P \wedge Q) \vee (Q \wedge \neg R) \vee Q$

  $(\neg P \wedge \neg R \wedge \neg Q) \vee (\neg P \wedge \neg R \wedge Q) \vee (\neg P \wedge R \wedge Q)$
  $\vee (P \wedge \neg R \wedge Q) \vee (P \wedge R \wedge Q)$

- Conjunctive Normal Forms

  $(\neg P \vee \neg R \vee Q) \wedge (\neg P \vee R \vee Q) \wedge (P \vee \neg R \vee Q)$

  $(\neg P \vee Q) \wedge (\neg R \vee Q)$

  $P \wedge Q$

Logical Equivalence

Two propositions are (logically) **equivalent** if and only if for each case their truth values are the same ("≡")

also called ***identities***

Examples:

$P \equiv (P \vee P)$ "1+1=2" $\leftrightarrow$ "1+1=2 or 1+1=2"

$\neg\neg P \equiv P$ "He did not not do it" $\leftrightarrow$ "He did it"

$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

"If P then not P" ≡ "not P"

Proving that two propositions are equivalent can be done by comparing their two truth tables

Example of Equivalence

- How to prove "$Q \wedge (\neg R \rightarrow P) \leftrightarrow (R \wedge Q) \vee (P \wedge Q)$"?
- Earlier we saw...

while for the other proposition…

P	Q	R	$\neg R$	$\neg R \rightarrow P$	$Q \wedge (\neg R \rightarrow P)$	$R \wedge Q$	$P \wedge Q$	$(R \wedge Q) \vee (P \wedge Q)$
F	F	F	T	F	F	F	F	F
F	F	T	F	T	F	F	F	F
F	T	F	T	F	F	F	F	F
F	T	T	F	T	T	T	F	T
T	F	F	T	T	F	F	F	F
T	F	T	F	T	F	F	F	F
T	T	F	T	T	T	F	T	T
T	T	T	F	T	T	T	T	T

Biconditional vs. Equivalence

- Don't confuse
  the equivalence $\equiv$ with the biconditional $\leftrightarrow$
  (only the biconditional has a truth table)
- For example:
  $P \equiv P$ is a proposition/tautology,
  a statement *within* logic,
  $P \equiv P$ is mathematically correct, … *about* logic
  $P \equiv \neg P$ is a contradiction (False),
  $P \equiv \neg P$ is incorrect
  Hence $P \leftrightarrow P \equiv \neg(P \leftrightarrow \neg P)$, and so on

P	Q	P ↔ Q
T	T	T
T	F	F
F	T	F
F	F	T

"The Laws of Logic" 1

1) Double negation law: $\neg\neg P \equiv P$
2) De Morgan's laws: $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ and
   $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
3) Commutative laws: $P \wedge Q \equiv Q \wedge P$ and
   $P \vee Q \equiv Q \vee P$
4) Associative laws: $P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$ and
   $P \vee (Q \vee R) \equiv (P \vee Q) \vee R$
5) Distributive laws: $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$ and
   $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

Augustus De Morgan
(June 27, 1806 – March 18, 1871)

"The Laws of Logic" 2

6) Idempotent laws: $P \wedge P \equiv P$ and $P \vee P \equiv P$

7) Identity laws: $P \vee False \equiv P$ and $P \wedge True \equiv P$

8) Inverse laws: $P \vee \neg P \equiv True$ and $P \wedge \neg P \equiv False$

9) Domination laws: $P \vee True \equiv True$ and $P \wedge False \equiv False$

10) Absorption laws: $P \vee (P \wedge Q) \equiv P$ and $P \wedge (P \vee Q) \equiv P$

Proving Equivalences

Here is how to prove our "$Q \wedge (\neg R \rightarrow P) \equiv (R \wedge Q) \vee (P \wedge Q)$"

$Q \wedge (\neg R \rightarrow P)$	$\equiv Q \wedge (\neg\neg R \vee P)$	[$P \rightarrow Q \equiv \neg P \vee Q$ rule]
	$\equiv Q \wedge (R \vee P)$	[double negation]
	$\equiv (Q \wedge R) \vee (Q \wedge P)$	[distributive law]
	$\equiv (R \wedge Q) \vee (Q \wedge P)$	[commutative law]
	$\equiv (R \wedge Q) \vee (P \wedge Q)$	[commutative law]

Rules of Inference

- In real life when proving mathematical statements often we do not establish an equivalence but a consequence
  - **Theorems** are true/correct mathematical statements
  - **Axioms** are "self-evident" theorems
  - Using the **rules of inference** we can make a (valid) **argument** to derive other theorems from the axioms
  - An **argument** is **valid** if and only if the validity of the hypotheses implies the validity of the conclusion
- Typically, an argument uses **hypotheses** or **premises** to reach a **conclusion**
- Example:
  "Assuming the hypotheses H_1, H_2, H_3
  it follows that conclusion C holds"
- How to do this is described by the **rules of inference**

Theorem

- Every formula is logically equivalent to formula in disjunctive normal form

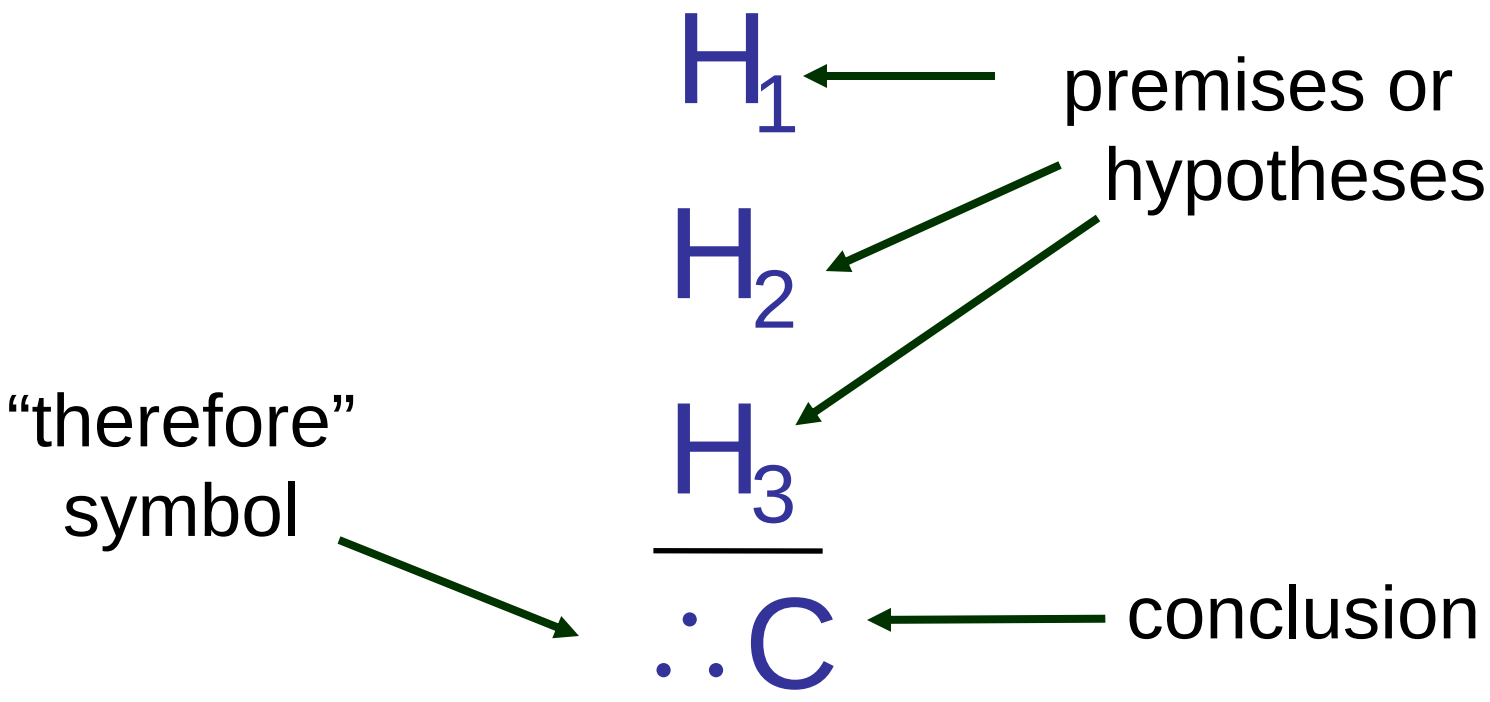
- Example:

$$(P \rightarrow Q\ ) \wedge (R \rightarrow Q)$$

is logically equivalent to:

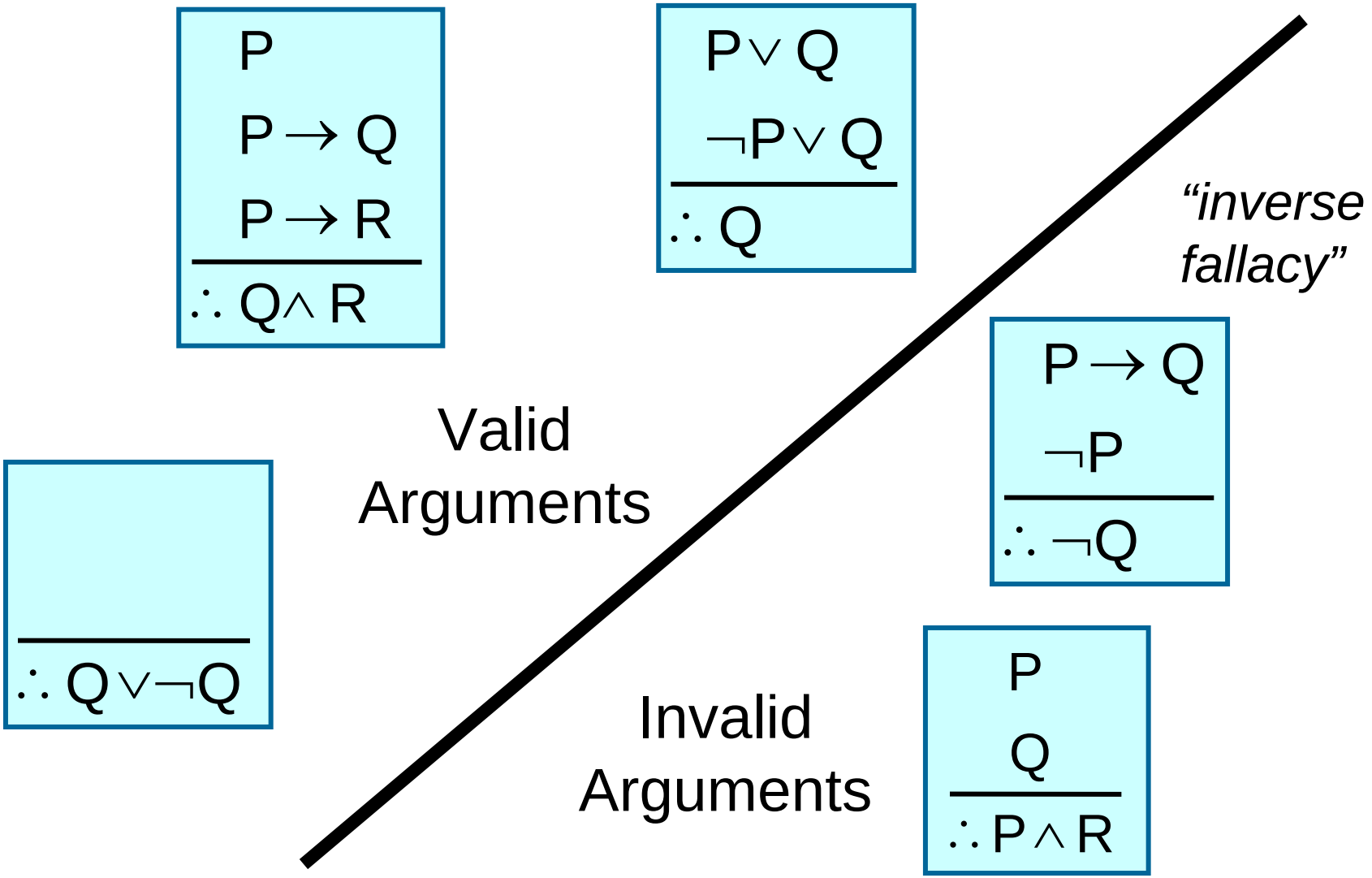
$$(\neg P \wedge \neg R \wedge \neg Q) \vee (\neg P \wedge \neg R \wedge Q) \vee$$
$$(\neg P \wedge R \wedge Q) \vee (P \wedge \neg R \wedge Q) \vee (P \wedge R \wedge Q)$$

Shape of an Argument



The therefore symbol "$\therefore$" is a bit old fashioned

Some Small Arguments



About Arguments

We can check arguments with the help of truth tables

An argument is valid if for all cases where the hypotheses are True, the Conclusion is True as well

Arguments are to conditionals ("$\rightarrow$"), what Equivalences ("$\leftrightarrow$") are to biconditionals ("$\leftrightarrow$")

Arguments are correct or incorrect / valid or invalid; a conditional is True or False

About Arguments

- For propositions P, Q,
  if P→Q is a tautology,
  then P logically implies Q
  This is denoted by "P → Q"

- Arguments are correct or incorrect / valid or invalid;
  a conditional is True or False

- Arguments are to conditionals ("→"),
  what Equivalences ("↔") are to biconditionals ("↔")

Checking Arguments

- An argument $(H_1 \wedge \ldots \wedge H_n) \rightarrow C$ is valid if
  - for all cases where the hypotheses H_j are True
  - the Conclusion C is True as well

- We can check arguments with the help of truth tables

  But just as with equivalences there are other ways of proving the validity of an argument

Rules of Inference I

$$\frac{P \rightarrow Q \quad P}{\therefore Q}$$ Rule of Detachment (Modus Ponens)

$$\frac{P \rightarrow Q \quad Q \rightarrow R}{\therefore P \rightarrow R}$$ Syllogism

$$\frac{P \rightarrow Q \quad \neg Q}{\therefore \neg P}$$ Modus Tollens

$$\frac{P \quad Q}{\therefore P \wedge Q}$$ Conjunctio n

Rules of Inference II

$P \vee Q$, $\neg P$ $\therefore Q$ — Rule of Disjunctive Syllogism

$P \wedge Q$ $\therefore P$ — Conjunctive Simplification

$P \rightarrow$ False $\therefore \neg P$ — Rule of Contradiction

P $\therefore P \vee Q$ — Disjunctive Amplification

Mần reng tui lấy ví dụ ?

1. **Tìm không gian các sự kiện, nhân vật thật**
2. **Tìm các phát biểu tương ứng với các biến trong luật**
3. **Gán nghĩa cho tứng thành phần của luật**
4. **Nhận kết quả**

Rules of Inference III

$P \wedge Q$

$P \rightarrow (Q \rightarrow R)$

$\therefore r$

Conditional Proof

$P \rightarrow R$

$Q \rightarrow R$

$\therefore (P \vee Q) \rightarrow R$

Proof by Cases

$P \rightarrow Q$

$R \rightarrow S$

$P \vee R$

$\therefore Q \vee S$

$P \rightarrow Q$

$R \rightarrow S$

$\neg Q \vee \neg S$

$\therefore \neg P \vee \neg R$

Proving Validity of Arguments

Using basic inference steps and equivalence rules one can prove the validity of arguments

Example:

$$\begin{array}{l} p \rightarrow q \\ q \rightarrow \neg p \\ \hline \therefore \neg p \end{array}$$

Valid?

Yes, according to truth tables

But also,

$$\begin{array}{l} p \rightarrow q \\ q \rightarrow \neg p \\ \hline \therefore p \rightarrow \neg p \end{array}$$

Syllogism

And because

$P \rightarrow \neg P \leftrightarrow \neg P \vee \neg P \leftrightarrow \neg P$

we have the validity proven a second time

Longer Arguments…

Example: $((\neg P \vee \neg Q) \rightarrow (R \wedge S)) \wedge (R \rightarrow T) \wedge (\neg T) \rightarrow P$

1)	$R \rightarrow T$	[Premise]
2)	$\neg T$	[Premise]
3)	$\neg R$	[Steps 1, 2 and Modus Tollens]
4)	$\neg R \vee \neg S$	[Step 3 and Disjunctive Amplification]
5)	$\neg (R \wedge S)$	[Step 4 and DeMorgan's Law]
6)	$(\neg P \vee \neg Q) \rightarrow (R \wedge S)$	[Premise]
7)	$\neg (\neg P \vee \neg Q)$	[Steps 5, 6 and Modus Tollens]
8)	$\neg \neg P \wedge \neg \neg Q$	[Step 7 and DeMorgan's Law]
9)	$P \wedge Q$	[Step 8 and Double Negation]
10)	P	[Step 9 and Conjunctive Simplification]

General Remarks

- Propositions that only use $\wedge$, $\vee$, $\neg$, (,) are the objects in **Boolean algebra** (without the implication "$\rightarrow$")
  - Note: the Laws of Logic do not use "$\rightarrow$"
- This is what you typically have in IF … THEN construction

- The implication becomes useful when you want to connect Boolean algebra with the rules of inference

- "False $\rightarrow$ P $\leftrightarrow$ True" follows from proof by contradiction
  - It holds that $(P \wedge \neg P) \rightarrow P$ hence $(P \wedge \neg P) \rightarrow P \leftrightarrow$ True
  - Take the two cases P $\leftrightarrow$ True and P $\leftrightarrow$ False

Terminology: 4 Conditionals

- For the propositions P and Q and the conditional $P \rightarrow Q$, we have the three other conditionals:

converse: $Q \rightarrow P$

inverse: $\neg P \rightarrow \neg Q$

contrapositive: $\neg Q \rightarrow \neg P$

Only one of these is equivalent with $P \rightarrow Q$…

… the contrapositive, hence: $(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$.

We also have for the other two: $(Q \rightarrow P) \leftrightarrow (\neg P \rightarrow \neg Q)$

but *not*: $(P \rightarrow Q) \leftrightarrow (Q \rightarrow P)$ or $(P \rightarrow Q) \leftrightarrow (\neg P \rightarrow \neg Q)$

Example of Inferencing

- Consider the following argument:
  1. Today is Tuesday or Wednesday
  2. But it can't be Wednesday, since the doctor's office is open today, and that office is always closed on Wednesdays
  3. Therefore today must be Tuesday
- This sequence of reasoning (inferencing) can be represented as a series of application of modus ponens to the corresponding propositions as follows

$$\begin{array}{l} P \rightarrow Q \\ P \\ \hline \therefore Q \end{array}$$

Example of Inferencing (Cont)

- The modus ponens is an inference rule which deduces **Q** from **P -> Q** and **P**
  - **T** Today is Tuesday
  - **W** Today is Wednesday
  - **D** The doctor's office is open today
  - **C** The doctor's office is always closed on Wednesdays
- The above reasoning can be represented by propositions as follows

1. **T** ∨ **W**

2. **D**
C

¬W
----

3. **T**

$$\frac{P \rightarrow Q \qquad P}{\therefore Q}$$

Example of Inferencing (Cont)

- To see if this conclusion **T** is correct, let us first find the relationship among **C**, **D**, and **W**

  C can be expressed using **D** and **W**

  That is, restate **C** first as the doctor's office is always closed if it is Wednesday

  Then $\mathbf{C} \equiv (\mathbf{W} \rightarrow \neg \mathbf{D})$

  Thus substituting $(\mathbf{W} \rightarrow \neg \mathbf{D})$ for **C**, we can proceed as follows

  D
  $\mathbf{W} \rightarrow \neg \mathbf{D}$

  $\neg \mathbf{W}$

  which is correct by modus tollens

Example of Inferencing (Cont)

- From this **¬W** combined with **T ∨ W** of **1.** above,

¬W
T ∨ W

T

which is correct by disjunctive syllogism

Thus we can conclude that the given argument is correct

To save space we also write this process as follows eliminating one of the **¬W**'s:

D
W → ¬D

¬W
T ∨ W

T

Limitations of Propositional Logic 1

- Propositional Logic :
  - is good for facts, not individuals
    But hard to identify individuals (terms)
  - E.g., Mary, John, 17, Canada
- We could try a variable **JohnIsTall**, but suppose we then want to encode a rule that tall people are good at basketball
  - E.g., TallPeople $\rightarrow$ GoodAtBasketball

  Given a knowledge base that consists of
  - JohnIsTall
  - TallPeople $\rightarrow$ GoodAtBasketball

Limitations of Propositional Logic 2

- Can't directly talk about properties of individuals or relations between individuals
  - E.g., how to represent the fact that John is tall?
- We have no way to conclude that John is good at basketball!
- Generalizations, patterns, regularities can't easily be represented
  - E.g., all triangles have 3 sides

Hệ chuyên gia (Expert System)

PGS.TS. Phan Huy Khánh

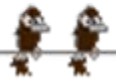
khanhph@vnn.vn

Chương 2
Biểu diễn tri thức nhờ logic vị từ bậc một
2.3

Biểu diễn tri thức nhờ logic vị từ bậc một

⌘ Phần 2.3 :

- Lôgic vị từ bậc một
- Biểu diễn tri thức nhờ logic vị từ bậc một

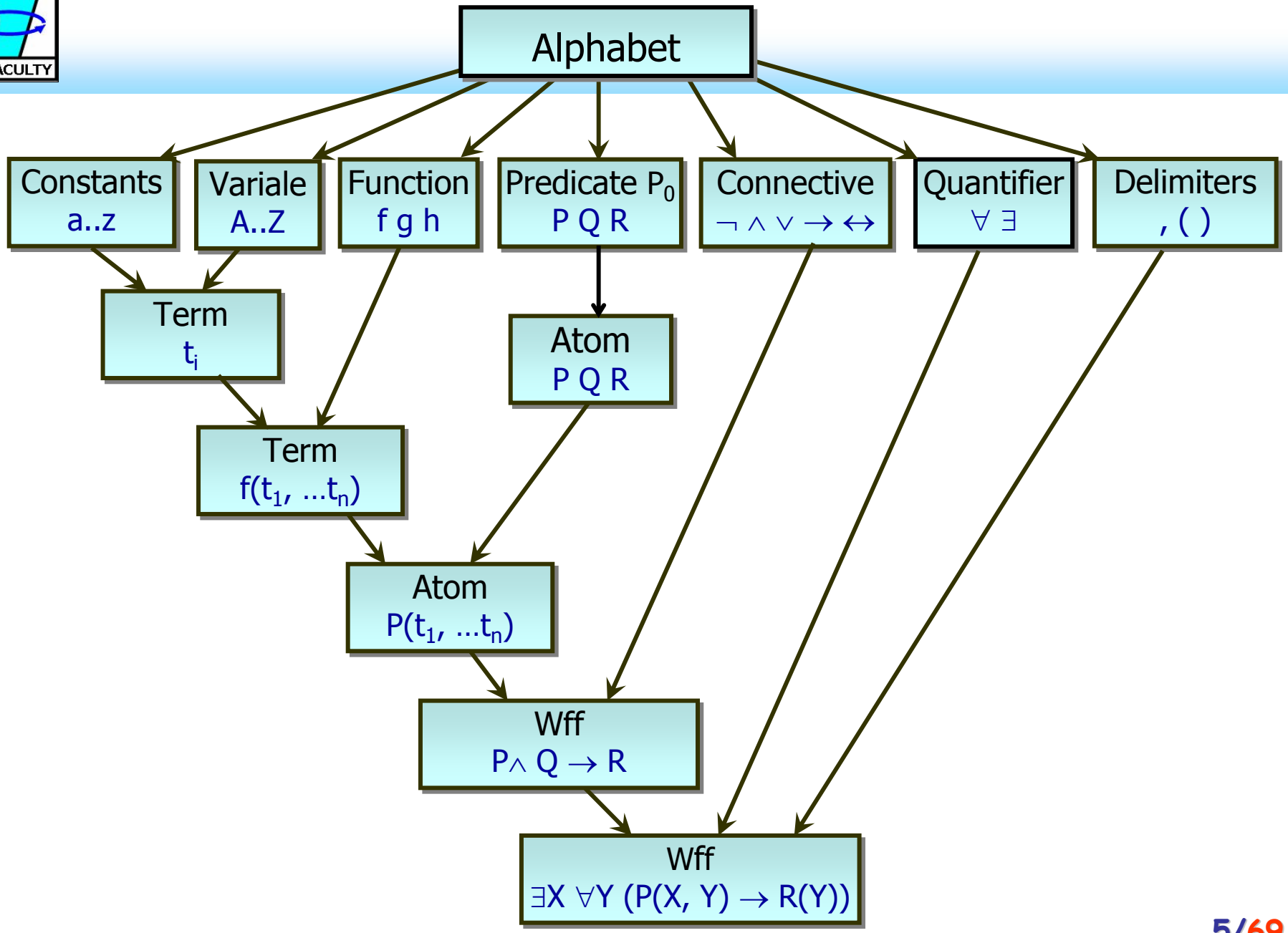


Limitations of Propositional Logic 2

- Can't directly talk about properties of individuals or relations between individuals
  - E.g., how to represent the fact that John is tall?
- We have no way to conclude that John is good at basketball!
- Generalizations, patterns, regularities can't easily be represented
  - E.g., all triangles have 3 sides

Predicate Logic Overview

- Predicate Logic
  - Principles
  - Objects
  - Relations
  - properties
- Syntax
- Semantics
- Extensions and Variations
- Proof in Predicate Logic
- Important Concepts and Terms


Alphabet
Constants
a..z
Variale
A..Z
Function
f g h
Predicate P_0
P Q R
Connective
¬ ∧ ∨ → ↔
Quantifier
∀ ∃
Delimiters
, ( )
Term
t_i
Atom
P Q R
Term
$f(t_1, \ldots t_n)$
Atom
$P(t_1, \ldots t_n)$
Wff
P∧ Q → R
Wff
∃X ∀Y (P(X, Y) → R(Y))

Bảng ký hiệu (Alphabet)

- Bảng ký hiệu để xây dựng các biểu thức đúng gồm :
  - Các *dấu phân cách* (separator signs) :

    dấu phẩy (**,**), dấu mở ngoặc (**(**) và dấu đóng ngoặc (**)**)
  - Các *hằng* (constant) :

    có dạng chuỗi sử dụng các chữ cái in thường **a..z**

    Ví dụ : a, block
  - Các *biến* (variable) :

    có dạng chuỗi sử dụng các chữ cái in hoa **A..Z**

    Ví dụ : X, NAME.
  - Các *vị từ* (predicate) :

    được viết tương tự các *biến*, sử dụng các chữ cái in hoa **A..Z**

    Ví dụ : ISRAINING, ON(table), P(X, blue), BETWEEN(X, Y, Z)

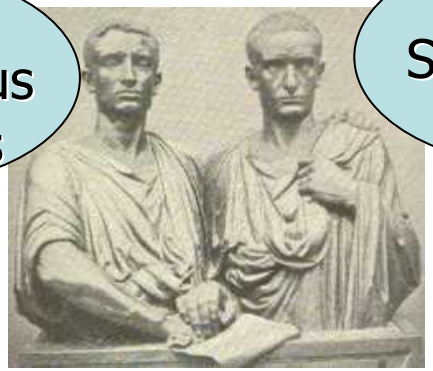
Bảng ký hiệu (Alphabet)

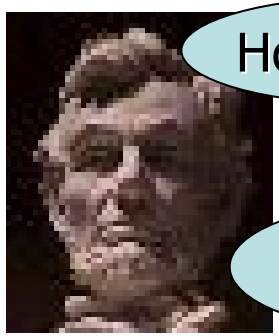
- Các phép nối logic (logical connector) :
  - $\neg$, $\wedge$, $\vee$, $\rightarrow$ và $\leftrightarrow$
    tương ứng với các phép phủ định, và, hoặc, kéo theo và kéo theo lẫn nhau (tương đương)
- Các dấu lượng tử
  - $\exists$ lượng tử tồn tại (existential quantifier)
  - $\forall$ lượng tử toàn thể (universal quantifier)

Names

⌘ Constants are used to name existing objects:

- The interpretation identifies the object in the real world
- No constant can name more than one object
- An object can have more than one name or no name at all





⌘ Variables:

V = {X, Y, Z, …}

BNF Grammar Predicate Logic

<Sentence> → <AtomicSentence>
| (<Sentence> <Connective> <Sentence>)
| <Quantifier> <Variable>, ... <Sentence>
| ¬ <Sentence>

<AtomicSentence> → <Predicate>(<Term>, ...) | <Term>= <Term>

<Term> → <Function>(<Term>, ...)
| <Constant> | <Variable>

<Connective> → ∧ | ∨ | → | ↔

<Quantifier> → ∀| ∃

<Constant> → a, b, c, max, carl, jim, jack

<Variable> → A, B, C, X_1 , X_2, COUNTER, POSITION

<Function> → father-of, square-position, sqrt, cosine

<Predicate> → P, Q, LARGER, BETWEEN, YOUNGER-THAN

Ambiguities are resolved through precedence or parentheses

First Order Predicate Logics Syntax

term	::=	variable	
		| function_symbol_of_arity_n(t_1, …, t_n)	n>0
		| function_symbol_of_arity_0	constant
atom	::=	predicate_symbol_of_arity_n(t_1, …, t_n)	n>0
		| predicate_symbol_of_arity_0	constant
literal	::=	atom	positive literal
		| $\neg$ atom	negative literal
wff	::=	atom	well formed formula (sentence)
		| ($\neg$ wff)	negation
		| (wff $\wedge$ wff)	conjunction
		| (wff $\vee$ wff)	disjunction
		| (wff $\rightarrow$ wff)	implication
		| (wff $\leftrightarrow$ wff)	equivalence
		| ($\forall$ variable wff)	universal formula
		| ($\exists$ variable wff)	existential formula

Các hàm (function)

- Các hàm :
  - có cách viết tương tự các hằng
  - sử dụng các chữ in thường **a..z**
  - Mỗi hàm có bậc (hay số lượng các đối) cố định, là một số nguyên dương
- Ví dụ :
  - f(X), weight(elephan), successor(M, N)
    là các hàm có bậc lần lượt là 1, 1, và 2
- Người ta quy ước rằng :
  - Các hằng là những hàm bậc không (nil)
  - Ví dụ : a, elephan, block là các hằng

Function Symbols

- Function symbols
  - function_name(arg_1, arg_2, …, arg_n)
  - Identifies the object referred to by a tuple of objects
  - May be defined implicitly through other functions, or explicitly through tables
- Function names begin with a lowercase letter or are expressed with a symbol
  - $F = \{f, g, h, \ldots\} = F_0 \cup F_1 \cup F_2 \cup \ldots$
- Function arities:
  - F_0: function symbols of arity 0 (constants): a, b, max, jim
  - F_1: function symbols of arity 1 (one argument)
  - F_2: function symbols of arity 2 (two arguments)
  - …

Functions Examples

- A function is used to express complex names
  - age(max) — Max's age
  - password(claire) — Claire's password
- A function may be nested
  - Max's age's double — double(age(max))
  - father(mother(max)) — Max's mother's father
  - starship(son(dr_crusher)) — Dr_Crusher's son's starship
- A function is never a predicate
  - Can't nest predicates — TALL(TALL(max))
- Function symbols of arity >1
  - youngestChild(max, ann) — Max and Ann's youngest child
  - *(5, +(2, 4)) — 30
- A predicate forms a sentence, while a function names an individual

Hạng, hay hạng tử (term)

- Hạng được tạo thành từ hai luật sau :
  - Các hằng và các biến là các hạng
  - Nếu f là một hàm có bậc $n \geq 1$
    và nếu $t_1,..., t_n$ đều là các hạng,
    thì hàm f $(t_1,..., t_n)$ cũng là một hạng
- Ví dụ các hàm sau đây đều là các hạng :
  - successor(X, Y), weight(b), successor(b, wight(Z))
- Nhưng các hàm sau đây không phải là hạng :
  - P(X, blue) vì P là vị từ
  - weight (P(b))
    vì P(b) không phải là hạng (vị từ không làm đối cho hàm)

Predicates

- Predicate symbols :
  - PREDICATE(arg_1, arg_2, …, arg_n)
  - A (determinate) property possessed by an object: Shape, Size
  - A (determinate) relationship among objects:
    Shape relationship, size relationship, positional relationship…
  - The number of arguments is called the predicate's arity
  - The order of the arguments is important
- Predicates have names beginning with an uppercase letter or are represented by an operator symbol
  - $P = P_0 \cup P_1 \cup P_2 \cup \ldots$
- Predicate arities:

  P_0: predicate symbols of arity 0 (constants: proposition) : P, Q, R, …

  P_1: predicate symbols of arity 1 (one argument)

  P_2: predicate symbols of arity 2 (two arguments) **...**

Nguyên tử (atom)

- Nguyên tử được tạo thành từ hai luật sau :
  - Các mệnh đề (vị từ bậc 0) là các nguyên tử
  - Nếu P là một vị từ bậc n ($n \geq 1$)
    và nếu $t_1,..., t_n$ đều là các hạng,
    thì $P(t_1,..., t_n)$ cũng là một nguyên tử
- Ví dụ các vị từ sau đây là các nguyên tử :
  - P(X, blue), EMPTY, BETWEEN(table, X, sill(window))
- Còn :
  - successor (X, Y, sill (window)
    không phải nguyên tử, mà là các hàm

Atomic Sentences

- A atomic sentence:
  - Expresses a claim that is either true or false
  - Formed by a single predicate followed by one or more arguments
- Example:
  - Max is tall

    TALL(max)
  - A is larger than B

    LARGER(A, B)
  - B is not larger than A

    ¬LARGER(B, A)
  - C is smaller than D, or D is not smaller than C

    SMALLER(C, D), ¬SMALLER(D, C)
  - A is between B and E:

    BETWEEN(A, B, E)

Các công thức chỉnh

- Các công thức chỉnh (CTC) được tạo thành từ ba luật sau :
  - Các nguyên tử là các CTC
  - Nếu G và H là các CTC,
    thì $(\neg G)$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$ và $(G \leftrightarrow H)$
    cũng là các CTC được tạo thành từ G và H
  - Nếu G là một CTC và X là một biến,
    thì $(\exists X)G$ và $(\forall X)G$ cũng là các CTC
- $(\exists X)G$ được đọc là :
  - Tồn tại biến X sao cho G được thoả mãn
- $(\forall X)G$ được đọc là :
  - Với mọi biến X thì G đều được thoả mãn

Bài tập ở lớp : Chuyển thành vị từ

- Ai đủ 18 tuổi mới được phép lái xe
- Gái đủ 18 tuổi, trai 20 tuổi mới được phép lập gia đình
- Kiểm tra hồ sơ
  - Nhập học tại các trường ĐH, CĐ
  - Sản phẩm
  - Quy trình công nghệ...

Mần reng tui lấy ví dụ ?

1. **Xác định không gian các sự kiện, nhân vật thật liên quan**
2. **Tìm các hằng, biến, hàm và/hoặc vị từ tương ứng với các phát biểu**
3. **Gán nghĩa cho tứng thành phần để kiểm tra tính đúng đắn**
4. **Nhận kết quả**

Well-formed Formula (wff)

- Any atomic sentence is a wff
- If A are B are wffs then so are

  $\neg A$

  $A \wedge B$

  $A \vee B$

  $A \rightarrow B$

  $A \leftrightarrow B$
- B is a cube or B is large (a large cube):

  $CUBE(B) \vee LARGE(B)$
- E and C are in the same row and E is in back of B:

  $SAMEROW(E, C) \wedge BACKOF(E, B)$

Equality

- Equality indicates that two terms refer to the same object =(A, B)
  - A and B are identical
  - Usually, written in infix form A = B
  - The equality symbol "=" is an (in-fix) shorthand
  - FATHER(jane) = jim, or =(FATHER(jane), jim)
  - E.g. Jim is Jane's and John's father
- Equality by reference and equality by value
  - Sometimes the distinction between referring to the same object and referring to two objects that are identical (indistinguishable) can be important
- $E \neq E$ E is not identical to iteself

Ví dụ

- Các công thức sau đây là chỉnh :
  - $(\exists X)\ (\forall Y)\ ((P(X, Y) \vee Q(X, Y) \rightarrow R(X))$
  - $((\neg(P(a) \rightarrow P(b))) \rightarrow \neg P(b))$
- Còn các công thức sau đây không chỉnh :
  - $(\neg(f(a))$: phủ định của một hàm,
  - f (P(a)) : hàm có đối là một vị từ
- Chú ý :
  - CTC được gọi là một trực kiện (literal) hay trị đúng nếu nó là một nguyên tử hay có dạng $(\neg G)$, với G là một nguyên tử
  - Trong một CTC, trước hoặc sau các ký tự nối, ký tự phân cách, các hằng, các biến, các hàm, các vị từ, người ta có thể đặt tùy ý các dấu cách (space hay blank)

Các bước xây dựng CTC

- Cho một phát biểu (sự kiện) trong NNTN :
  - (Chơ ai) đi mô cũng nhớ về Hà Tịnh
- Xác định các miền đối tượng :
  - Người : cu Tý, cu Tèo, cái Nơ… $X \in D1$
  - Quê : Hà Tịnh, Hà Tây, Hà Giang… $Y \in D2$
- Xác định các quan hệ để xây dựng các mệnh đề
  - Cu Tý quê Hà Tịnh : QUÊ(cutý, hàtịnh), QUÊ(X, Y)
  - Cu Tý xa quê Hà Tịnh : XAQUÊ(cutý, hàtịnh), XAQUÊ(X, Y)
  - Cu Tý nhớ quê Hà Tịnh : NHỚQUÊ(cutý, hàtịnh), NHỚQUÊ(X, Y)
- Xây dựng CTC
  - $\forall X, \exists Y$ (QUÊ(X, Y) $\wedge$ XAQUÊ(X, Y) $\rightarrow$ NHỚQUÊ(X, Y))

Predicate Logics: some terminology

- There is a predicate logic for each basis B=(F, P) of function and predicate symbols
- Terms formed on basis B are called B-terms: the set of all B-terms is denoted T_B
- Formulas formed on basis B are called B-formulas: the set of all B-formulas is denoted WFF_B
- Formulas with all variables bound to a quantifier are called closed formulas
- Formulas with no quantifier are called quantifier free formulas
- Formulas with no quantifier and no variable are called ground formulas

Một số nhận xét 1

- Từ nay ta quy ước rằng, trong một CTC :
  - Một biến được lượng tử hóa sẽ xuất hiện ngay sau ∃ hay ∀
  - Phạm vi lượng tử hóa của biến kể từ vị trí xuất hiện trở đi
  - Có thể có các biến tự do (free variable),
    là các biến không được lượng tử hóa
  - Ví dụ : P(X) và (∃Y) Q(X, Y) có chứa biến tự do X
- Logic vị từ được gọi là «bậc một» (first–order) :
  - Trong CTC không định nghĩa lượng tử cho vị từ hay cho hàm
  - Ví dụ : (∀P)P(a) và (∀f) (∀f) (∀X) P(f (X), b)
    không phải là những vị từ bậc một,
    mà có bậc cao hơn (higher-order)

Một số nhận xét 2

- Tri thức diễn tả theo ngôn ngữ tự nhiên hay toán học không phải luôn luôn dễ dàng chuyển đổi thành các CTC trong lôgic vị từ bậc một
- Chẳng hạn, để diễn tả rằng :
  «Nếu hai vật y chang nhau thì chúng có cùng tính chất», người ta có thể viết :

  $(\forall P)\ (\forall X)\ (\forall Y)\ (EQUAL(X, Y) \rightarrow (P(X) \leftrightarrow P(Y)))$
- Nhưng biểu thức trên không phải là logic vị từ bậc một vì có lượng tử $\forall$ áp dụng cho một ký tự vị từ là P
- Trong lôgic vị từ bậc một, sự kiện trên được viết :

  $(\forall X)\ (\forall Y)\ (SAME\_P(X, Y) \rightarrow (P(X) \leftrightarrow P(Y)))$, hoặc

  $(\forall X)\ (\forall Y)\ (SAME\_P(X, Y) \rightarrow (HAVE(X, p) \leftrightarrow HAVE\ (Y, p)))$

Example: Family Relationships

- Objects: people
- Properties: gender, …
  - Expressed as unary predicates:

    MALE(X), FEMALE(Y) hoặc SEX(X, male), SEX(Y, female)
- Relations: parenthood, brotherhood, marriage
  - Expressed through binary predicates

    PARENT(X, Y), BROTHER(X, Y), …
- Functions: motherhood, fatherhood
  - Because every person has exactly one mother and one father:

    mother(X), father(Y)
  - There may also be a relation

    mother-of(X, Y), father-of(X, Y)

Atomic Sentences Translation

- Function translation:
  - Brando is Nancy's favorite actor:

    brando = favoritecctor(nancy)
  - Sean is his own favorite actor

    sean = favoriteactor(sean)
- Sentences Translation:
  - Nancy's favorite actor is better than Max's favorite actor:

    BETTERACTOR(favoriteactor(nancy), favoriteactor(max))

Examples Atomic Sentences

FATHER(jack, john), MOTHER(jill, john), SISTER(jane, john)

PARENTS(jack, jill, john, jane)

MARRIED(jack, jill)

MARRIED(father-of(john), mother-of(john))

MARRIED(father-of(john), mother-of(jane))

FATHER(jack, john) $\wedge$ MOTHER(Jill, john) $\wedge$ SISTER(jane, john)

$\neg$ SISTER(john, jane)

PARENTS(jack, jill, john, jane) $\wedge$ MARRIED(jack, jill)

PARENTS(jack, jill, john, jane) $\rightarrow$ MARRIED(jack, jill)

OLDER-THAN(jane, john) $\vee$ OLDER-THAN(john, jane)

OLDER(father-of(john), 30) $\vee$ OLDER (mother-of(john), 20)

Avoid Ambiguity

- Both C and E are cubes
  $CUBE(c) \wedge CUBE(e)$
- Either C or E is a cube
  $CUBE(c) \vee CUBE(e)$
- Neither C nor E is a cube
  $\neg(CUBE(c) \vee CUBE(e))$
- Both A or B and C
  $A \vee B \wedge C$
- Either A or both B and C
  $A \vee (B \wedge C)$
- Either both A and B or C
  $A \wedge B \vee C$
- Both A and either B or C
  $A \wedge (B \vee C)$
- Either both Max is at home and Claire is tall or Carl is happy
  $(HOME(max) \wedge TALL(claire)) \vee HAPPY(carl)$

Quantifiers

- Quantifiers can be used to express properties of collections of objects
- Quantifiers eliminates the need to explicitly enumerate all objects
- The universal quantifier, represented by the symbol ∀ means "for every" or "for all"
- The existential quantifier, represented by the symbol ∃ means "there exists"
- Limitations of predicate logic – *most* quantifier

Universal Quantifiers ∀ (for all)

- ∀X P(x) states that
  - a predicate P is holds for all objects X in the universe (domain) under discourse
  - The sentence is true if and only if all the individual sentences where the variable X is replaced by the individual objects it can stand for are true
- Example
  - All dogs are happy

    ∀ X (DOG(X) → HAPPY(X))
  - No dog is happy

    All dogs are unhappy

    ∀ X (DOG(X) → ¬HAPPY(X))

Usage of Universal Qualification

- Universal quantification is frequently used to make statements like:
  - "all humans are mortal"
  - "all cats are mammals"
  - "all birds can fly" …
- This can be expressed through sentences like

$\forall X\ \text{HUMAN}(X) \rightarrow \text{MORTAL}(X)$

$\forall X\ \text{CAT}(X) \rightarrow \text{MAMMAL}(X)$

$\forall X\ \text{BIRD}(X) \rightarrow \text{CAN-FLY}(X)$

- These sentences are equivalent to the explicit sentence about individuals

$\text{HUMAN}(\text{john}) \rightarrow \text{MORTAL}(\text{john}) \wedge$

$\text{HUMAN}(\text{jane}) \rightarrow \text{MORTAL}(\text{jane}) \wedge$

$\text{HUMAN}(\text{jill}) \rightarrow \text{MORTAL}(\text{jill}) \wedge \ldots$

Existential Quantifier ∃ (there exists)

- ∃X P(x) states that
  - a predicate P holds for some objects in the universe
  - The sentence is `true` if and only if there is at least one `true` individual sentence where the variable X is replaced by the individual objects it can stand for
- Example:
  - Some dogs are happy

    ∃ X (DOG(X) ∧ HAPPY(X))
  - Some dogs are not happy

    ∃ X (DOG(X) ∧ ¬HAPPY(X))

Usage of Existential Quantification

- Existential quantification is used to make statements like:
  - "some humans are computer scientists"
  - "john has a sister who is a computer scientist"
  - "some birds can't fly" …
- This can be expressed through sentences like

  $\exists X$ HUMAN(X) $\wedge$ COMPUTER-SCIENTIST(X)

  $\exists X$ SISTER(X, john) $\wedge$ COMPUTER-SCIENTIST(X)

  $\exists X$ BIRD(X) $\wedge \neg$CAN-FLY(X)
- These sentences are equivalent to the explicit sentence about individuals
  - HUMAN(john) $\wedge \neg$ COMPUTER-SCIENTIST(john) $\vee$
  - HUMAN(jane) $\wedge$ COMPUTER-SCIENTIST(jane) $\vee$
  - HUMAN(jill) $\wedge \neg$COMPUTER-SCIENTIST(jill) $\vee$ …

Multiple Quantifiers

- More complex sentences can be formulated by multiple variables and by nesting quantifiers
  - The order of quantification is important
  - Variables must be introduced by quantifiers, and belong to the innermost quantifier that mention them
- Examples

$\forall X, \forall Y\ PARENT(X, Y) \rightarrow CHILD(Y, X)$

$\forall X\ HUMAN(X) \wedge \exists Y\ MOTHER(Y, X) \Rightarrow \forall X\ \exists Y\ HUMAN(X) \wedge MOTHER(Y, X)$

$\forall X\ HUMAN(X) \wedge \exists Y\ LOVES(X, Y) \Rightarrow \forall X\ \exists Y\ HUMAN(X) \wedge LOVES(X, Y)$

$\exists X\ HUMAN(X) \wedge \forall Y\ LOVES(X, Y) \Rightarrow \forall X\ \forall Y\ HUMAN(X) \wedge LOVES(X, Y)$

$\exists X\ HUMAN(X) \wedge \forall Y\ LOVES(Y, X) \Rightarrow \exists X\ \forall Y\ HUMAN(X) \wedge LOVES(Y, X)$

- They can translate to the form:

Q<V> M<V>, with Q: quantifiers, M: Matrix, wffs including <V>
V: Variable

Connections between ∀ and ∃

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation
  - ∀ is a conjunction over all objects under discourse
  - ∃ is a disjunction over all objects under discourse
- De Morgan's rules apply to quantified sentences

$$\forall X \neg P(X) \equiv \neg \exists X\ P(X) \qquad \neg \forall X\ P(X) \equiv \exists X \neg P(X)$$

$$\forall X\ P(X) \equiv \neg \exists X \neg\ P(X) \qquad \neg \forall X \neg P(X) \equiv \exists X\ P(X)$$

- Strictly speaking, only one quantifier is necessary
  - Using both is more convenient

Family Relationships

- One's mother is one's sibling's mother

  $\forall M, C, D\ MOTHER(M, C) \wedge SIBLING(C, D) \rightarrow MOTHER(M, D)$

- Brothers are siblings

  $\forall X, Y\ BROTHER(X, Y) \rightarrow SIBLING(X, Y)$

- Sibling is transitive

  $\forall X, Y, Z\ SIBLING(X, Y) \wedge SIBLING(Y, Z) \rightarrow SIBLING(X, Z)$

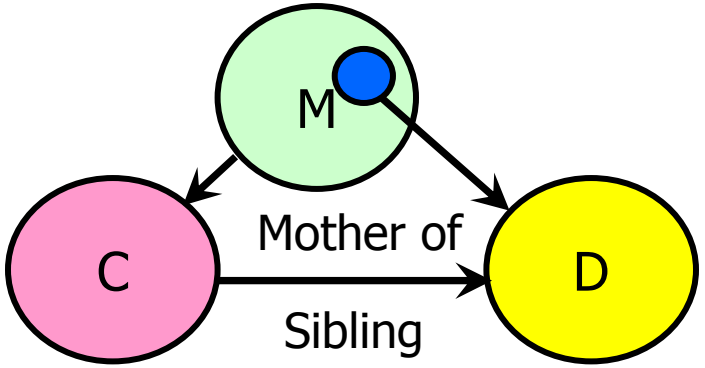
- One's mother is one's sibling's mother

  $\forall M, C\ MOTHER(M, C) \wedge SIBLING(C, D) \rightarrow MOTHER(M, D)$

- A first cousin is a child of a parent's sibling

  $\forall C, D\ FIRSTCOUSIN(C, D) \leftrightarrow$

  $\exists P, PS\ PARENT(P, D) \wedge SIBLING(P, PS) \wedge PARENT(PS, C)$



Exercises at class bài tập

- Tìm mệnh đề tương ứng đặt vào dấu ?

Người này là mẹ của người kia khĩ một người là phụ nữ và là thân sinh của người kia

$\forall M, \forall C\ MOTHER(C) = M \leftrightarrow FEMALE(M) \wedge PARENT(M, C)$

Người này là chồng của người kia khĩ người này là đàn ông và người kia là vợ của người này

$\forall W, \forall H\ HUSBAND(H, W) \leftrightarrow MALE(H) \wedge SPOUSE(W, H)$

- ?

$\forall X\ MALE(X) \leftrightarrow \neg FEMALE(X)$

- ?

$\forall G, \forall C\ GRANDPARENT(G, C) \leftrightarrow \exists P\ PARENT(G, P) \wedge PARENT(P, C)$

- ?

$\forall X, \forall Y\ SIBLING(X, Y) \leftrightarrow$
$\neg(X=Y) \wedge \exists P\ PARENT(P, X) \wedge PARENT(P, Y)$

Translating English to FOL

- Every gardener likes the sun

  $\forall X\ GARDENER(X) \rightarrow LIKES(X, sun)$

- You can fool some of the people all of the time

  $\exists X\ \forall T\ (PERSON(X) \wedge TIME(T)) \rightarrow CAN\text{-}FOOL(X, T)$

- You can fool all of the people some of the time

  $\forall X\ \exists T\ (PERSON(X) \wedge TIME(T) \rightarrow CAN\text{-}FOOL(X, T)$

- All purple mushrooms are poisonous

  $\forall X\ (MUSHROOM(X) \wedge PURPLE(X)) \rightarrow POISONOUS(X)$

- No purple mushroom is poisonous

  $\neg(\exists X)\ PURPLE(X) \wedge MUSHROOM(X) \wedge POISONOUS(X)$

  or, equivalently,

  $(\forall X)\ (MUSHROOM(X) \wedge PURPLE(X)) \rightarrow \neg POISONOUS(X)$

Translating English to FOL…

- There are exactly two purple mushrooms

  $(\exists X)(\exists Y)\ \text{MUSHROOM}(X) \wedge \text{PURPLE}(X) \wedge \text{MUSHROOM}(Y) \wedge \text{PURPLE}(Y) \wedge \neg(X=Y) \wedge (\forall Z)\ (\text{MUSHROOM}(Z) \wedge \text{PURPLE}(Z)) \rightarrow ((X=Z) \vee (Y=Z))$

- Bob is not tall

  $\neg\text{TALL}(\text{bob})$

- X is above Y if X is on directly on top of Y or else there is a pile of one or more other objects directly on top of one another starting with X and ending with Y

  $(\forall X)(\forall Y)\ \text{ABOVE}(X, Y) \leftrightarrow (\text{ON}(X, Y) \vee (\exists Z)\ (\text{ON}(X, Z) \wedge \text{ABOVE}(Z, Y)))$

Colonel West Example

- Assume:
  - It is a crime for an American to sell weapons to a hostile nation. The country Nono, an enemy of America, has some missiles, and all its missiles were sold to it by Colonel West, who is an American
  - Constants: nono, america, west
  - Predicates: CRIMINAL, AMERICAN, WEAPON, HOSTILE, NATION, ENEMY, MISSILE, OWNS, SELLS
- Then
  - It is a crime for an American to sell weapons to a hostile nation
  - $\Rightarrow$ If any american X sells any weapon Y to any hostile nation Z, then that american X is a criminal

$$\forall X,Y,Z \quad (AMERICAN(X) \wedge WEAPON(Y) \wedge NATION(Z) \wedge HOSTILE(Z) \wedge SELLS(X, Z, Y) \rightarrow CRIMINAL(X))$$

Other Translating

- Nono's missiles
  - Nono has some missiles

$$\exists X\ (MISSILE(X) \wedge OWNS(nono, X))$$

  - All of Nono's missiles were sold to it by West

$\Rightarrow$ If X is a missile owned by Nono then West sold X to Nono

$$\forall X\ ((MISSILE(X) \wedge OWNS(nono, X)) \rightarrow SELLS(west, nono, X))$$

- The other facts
  - An enemy of America is hostile

$$\forall X\ (ENEMY(X, america) \rightarrow HOSTILE(X))$$

  - West is an American

$$AMERICAN(west)$$

  - Nono is an enemy of America

$$NATION(nono) \wedge NATION(america) \wedge ENEMY(nono, america)$$

Other Examples

- Love
  - There is a girl who is loved by every boy

  $\Rightarrow$ There is a girl X and if Y is a boy then Y loves her

  $\exists X\,(GIRL(X) \wedge \forall Y(BOY(Y) \rightarrow LOVES(Y,X)))$
  - Every boy loves some girl
  - For every boy Y there exists a girl X that he loves

  $\forall Y(BOY(Y) \rightarrow \exists X\,(GIRL(X) \wedge LOVES(Y,X)))$
- Socrates Example
  - All men are mortal

  $\forall X\,(MAN(X) \rightarrow MORTAL(X))$
  - Socrates is a man

  $MAN(socrates)$
  - Socrates is mortal

  $MORTAL(socrates)$

Curiosity Example

- Assume:
  - Jack owns a dog
  - Every dog owner is an animal lover
  - No animal lover kills an animal
  - Either Jack or Curiosity killed the cat
  - The cat's name is Tuna

  Then
  - Constants: jack, curiosity, tuna.
  - Predicates: OWNS, DOG, ANIMALLOVER, KILLS, ANIMAL

Curiosity Example

- Dog owners
  - Jack owns a dog
  - ⇒ Some dog is owed by Jack

$$\exists X\,(DOG(X) \wedge OWNS(jack, X))$$

  - Every dog owner is an animal lover

$$\forall X\,((\exists Y\,(DOG(Y) \wedge OWNS(X,Y))) \rightarrow ANIMALLOVER(X))$$

- Animal Lovers
  - No animal lover kills an animal
  - ⇒ An animal lover does not kill an animal
  - ⇒ For any animal lover X and any animal Y, then Y is not killed by X

$$\forall X, Y\,((ANIMALLOVER(X) \wedge ANIMAL(Y)) \rightarrow \neg KILLS(X,Y))$$

Curiosity Example

- Cat Knowledge
  - The cat's name is Tuna

$$CAT(tuna)$$

  - Either Jack or Curiosity killed the cat

$$KILLS(jack, tuna) \vee KILLS(curiosity, tuna)$$

  - All cats are animals

$$\forall X\ CAT(X) \rightarrow ANIMAL(X)$$

Bài tập 1

- Cho các câu sau :
  - Marcus là một người
  - Marcus là người xứ Pompeii (hay xứ Campanie gần Naple nước Ý)
  - Mọi người Pompeii đều là người La Mã
  - Ceasar là một kẻ cầm quyền
  - Người La Mã hoặc là trung thành với Ceasar, hoặc là thù ghét Ceasar
  - Mỗi người đều trung thành với một người nào đó
  - Nhân dân chỉ muốn giết những kẻ cầm quyền mà họ không trung thành
  - Marcus muốn giết Ceasar
- Biểu diễn các câu trên thành các wff's

Biểu diễn wff's trong lôgic vị từ

- Marcus là một người

  $MAN(marcus)$
- Marcus là một người Pompeii

  $POMPEIAN(marcus)$
- Mọi người Pompeian đều là người La Mã

  $\forall x : POMPEIAN(X) \rightarrow ROMAN(X)$
- Caesar là một kẻ cầm quyền
  (giả sử không có sự trùng tên)

  $RULER(caesar)$
- Người La Mã hoặc là trung thành với Caesar,
  hoặc là thù ghét Caesar

  $\forall x : ROMAN(X) \rightarrow LOYALTO(X, caesar) \vee HATE(X, caesar)$
- Tuy nhiên khi sử dụng với nghĩa hoặc có loại trừ, có thể viết lại :

  $\forall x : ROMAN(X) \rightarrow ((LOYALTO(X, caesar) \wedge HATE(X, caesar)) \wedge (\neg LOYALTO(X, caesar) \wedge HATE(X, caesar)))$

Biểu diễn wff's trong lôgic vị từ

- Mọi người đều trung thành với một người nào đó

  $\forall X, \exists Y$ LOYALTO(X, Y) hoặc $\exists Y, \forall X$ LOYALTO(X, Y)

- Nhân dân chỉ muốn giết những kẻ cầm quyền mà họ không trung thành

  $\forall X, \forall Y$ PERSON(X) $\wedge$ RULER(Y) $\wedge$ TRYASSASSINATE(X, Y) $\rightarrow$
  $\neg$LOYALTO(X, Y)

  Mệnh đề này tỏ ra mập mờ (ambiguous). Có phải chỉ những kẻ cầm quyền mà nhân dân muốn giết là những kẻ họ không trung thành (theo nghĩa đã biểu trưng) hay chỉ những điều mà nhân dân có ý định là giết những kẻ cầm quyền mà họ không trung thành ?

- Marcus muốn giết Ceresar
  - TRYASSASSINATE(marcus, caesar)
- Câu hỏi : Marcus có trung thành với Caesar không ?

Bài tập 2

- Cho các câu sau :
  - Marcus là một người
  - Marcus là người Pompeii
  - Marcus sinh năm 40 trong công nguyên (A.D.: Anno Domini)
  - Mọi người đều (ai cũng phải) chết
  - Tất cả mọi người dân Pompeii đều bị chết vì núi lửa phun vào năm 79 A.D.
  - Không có người nào (không ai) sống nhiều hơn 150 tuổi
  - Bây giờ là năm 2007
  - Còn sống có nghĩa là không chết
  - Nếu ai đó chết, thì người ấy có thể chết ở mọi thời điểm sau đó
  - Marcus còn sống không ?
- Biểu diễn các câu trên thành các wff's

- Marcus là một người

  MAN(marcus)
- Marcus là người Pompeii

  POMPEIAN (marcus)
- Marcus sinh năm 40 trong công nguyên (A.D.: Anno Domini)

  BORN(marcus, 40)
- Mọi người đều (ai cũng phải) chết

  $\forall X\ MAN(X) \rightarrow MORTAL(X)$
- Tất cả mọi người dân Pompeii đều bị chết vì núi lửa phun vào năm 79 A.D.

  $\forall X\ ERUPTED(vocalno, 79) \wedge POMPEIAN(X) \rightarrow DIED(X, 79)$

- Không có người nào (không ai) sống nhiều hơn 150 tuổi

  $\forall X\ \forall T1\ \forall T2$
  $MAN(X) \wedge BORN(X, T1) \wedge GT(T2 - T1, 150) \rightarrow DIED(X, T2)$

- Bây giờ là năm 2007

  $now = 2007$

- Còn sống có nghĩa là không chết

  $\forall X\ \forall T\ (ALIVE(X, T) \rightarrow \neg DEAD(X, T))$
  $\wedge\ ((DIED(X, T) \rightarrow \neg ALIVE(X, T))$

- Nếu ai đó chết,
  thì người ấy có thể chết ở mọi thời điểm sau đó

  $\forall X\ \forall T1\ \forall T2\ DIED(X, T1) \wedge GT(T2, T1) \rightarrow DEAD(X, T2)$

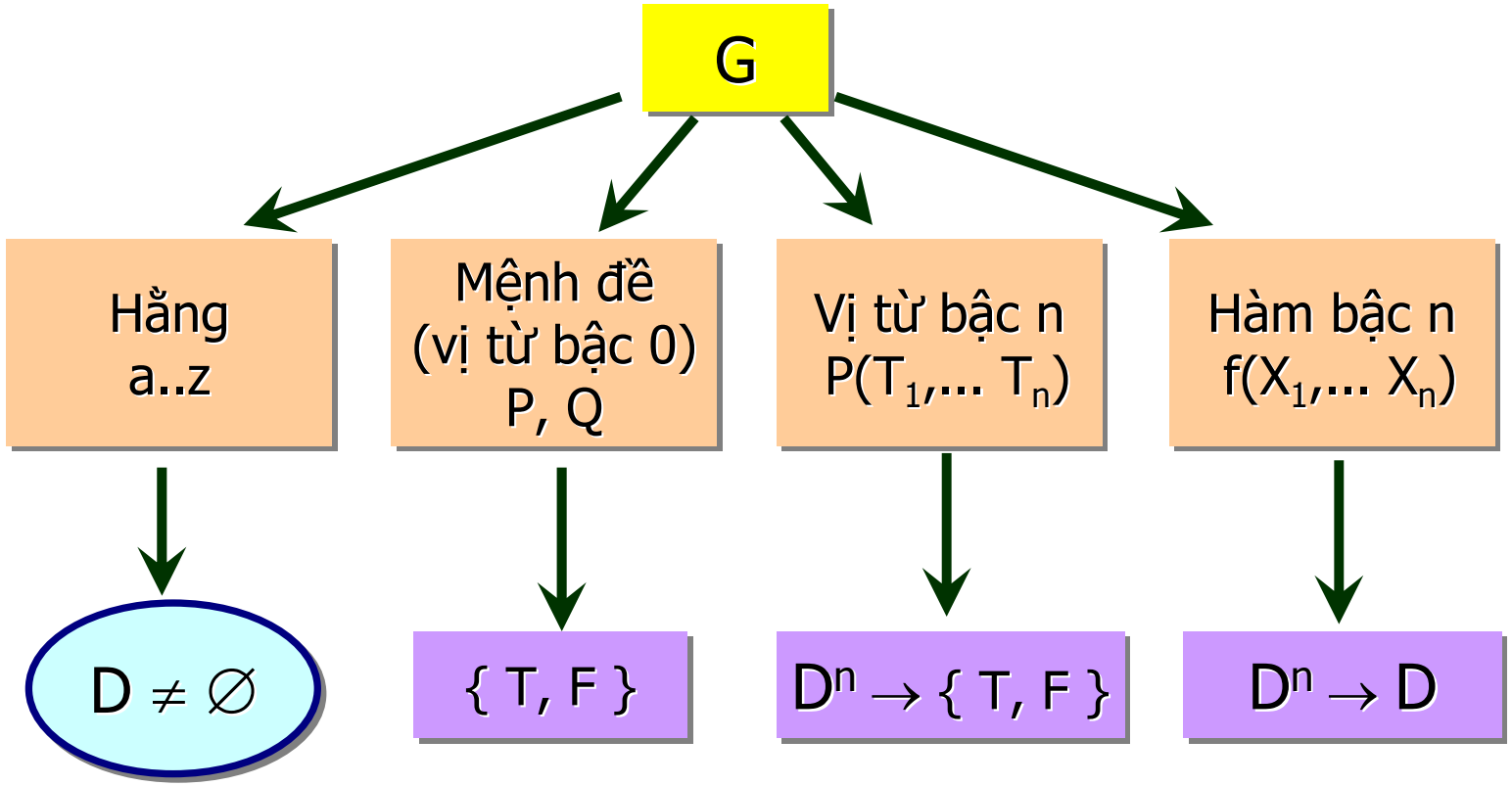
Diễn giải (Interpretation)

- Cho G là một CTC, một diễn giải của G, ký hiệu I, được xác định từ năm bước sau đây :
  - Chọn miền diễn giải (Interpretation Domain) là các tập hợp khác rỗng, ký hiệu $D \neq \varnothing$
  - Gán (Assignation) cho mỗi hằng của G một phần tử của D_i
  - Gán cho mỗi mệnh đề (hay vị từ bậc 0) một giá trị true (T) hoặc false (F)
  - Gán cho mỗi vị từ bậc n ($n \geq 1$) ánh xạ từ D^n lên { T, F } :

    $P(T_1, \ldots T_n) : D^n \rightarrow \{ T, F \}$
  - Gán cho mỗi hàm bậc n ($n \geq 1$) ánh xạ từ D^n lên D :

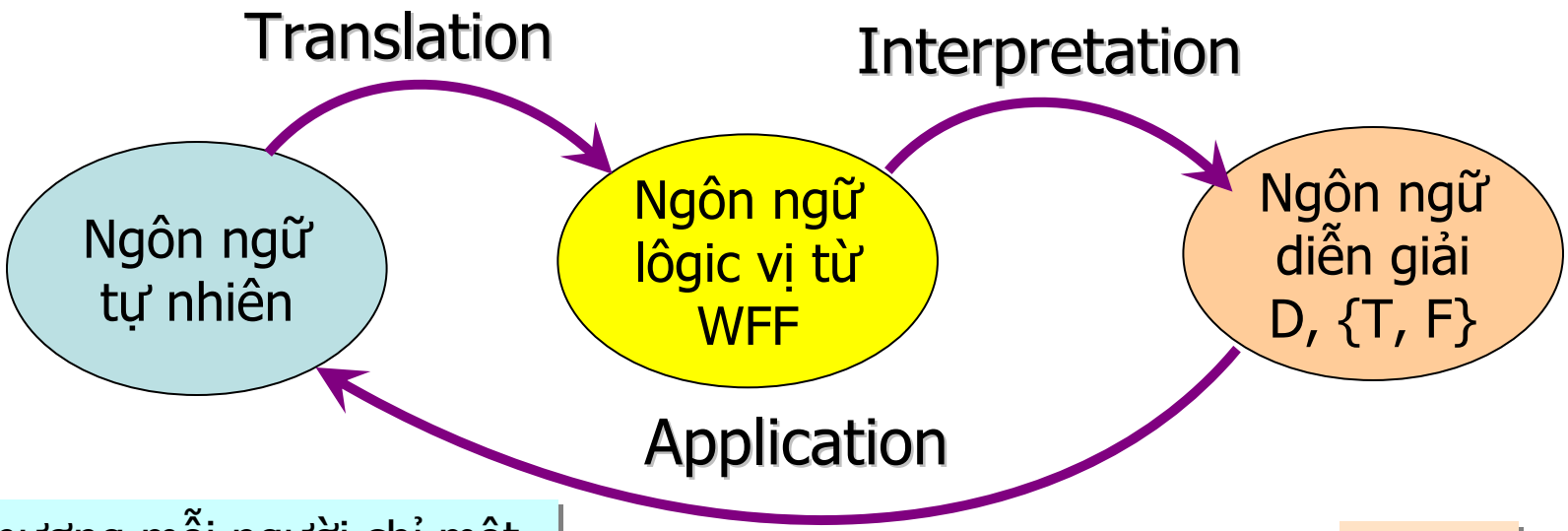
    $f(X_1, \ldots X_n) : D^n \rightarrow D$

Mô hình diễn giải I từ G lên D



⌘ Khi một CTC G có giá trị là T theo một diễn giải I, người ta nói rằng diễn giải I là một mô hình của G

Quan hệ Translation-Interpretation



Quê hương mỗi người chỉ một, như là chỉ một mẹ thôi

T

$$\forall X \exists Y \exists Z\ (QUE(X,Y) \wedge (QUE(X,Z) \rightarrow EQ(Y,Z)) \leftrightarrow \forall T \exists U \exists V\ (FEMALE(U) \wedge (MOTHER(U,T)) \wedge (FEMALE(V) \wedge (MOTHER(V,T)) \Rightarrow EQ(U,V))$$

Tính giá trị của CTC theo diễn giải

- Cho G là một CTC và một diễn giải I trên một miền D
- Khi đó :
  - Nếu G là một mệnh đề, giá trị gán cho G qua I là I(G)
  - Nếu G là một trực kiện, G nhận một giá trị T hay F
  - Nếu G có dạng $(\forall X)G'$:
    - I(G) = T nếu I(G') = T cho mọi giá trị của biến X trong D
    - I(G) = F nếu không phải
  - Nếu G có dạng $(\exists X)G'$:
    - I(G) = T nếu I(G') = T với ít nhất một giá trị của X trong D
    - I(G) = F nếu không phải

Tính giá trị của CTC theo diễn giải (tiếp)

- Nếu G có dạng ($\neg$G′) :
  - I(G) = T nếu I(G′) = F trong D
  - I(G) = F nếu I(G′) = T trong D
- Nếu G có dạng (G′ $\wedge$ G″), hoặc (G′ $\vee$ G″), hoặc (G′ $\rightarrow$ G″), hoặc (G′ $\leftrightarrow$ G″), khi đó :

G′	G″	(G′ $\wedge$ G″)	(G′ $\vee$ G″)	G′ $\rightarrow$ G″	G′ $\leftrightarrow$ G″
F	F	F	F	T	T
F	T	F	T	T	F
T	F	F	T	F	F
T	T	T	T	T	T

Tính hợp thức và tính nhất quán

- Một CTC được gọi là :
  - *Hằng đúng* (tautology), hay *hợp thức* (valid) nếu và chỉ nếu mọi diễn giải đều cho giá trị T
  - Nếu không, được gọi là *không hợp thức* (non–valid)
- Một CTC được gọi là :
  - *Mâu thuẫn* (contradiction), hay *không nhất quán* (inconsistent) nếu và chỉ nếu với mọi diễn giải đều cho giá trị F
  - Nếu không, được gọi là *nhất quán* (consistent)
- Quy ước :
  - □ biểu diễn một CTC hợp thức-hằng đúng
  - ∇ biểu diễn một CTC mâu thuẫn-không nhất quán

Công thức tương đương

- Cho G và H là hai CTC
- G và H được gọi là tương đương, $G \equiv H$:
  - Nếu và chỉ nếu G và H có cùng giá trị (T hoặc F) cho mọi diễn giải I, I(G) = I(H)
- Ví dụ :
  - $(P(a) \rightarrow Q(b)) \equiv ((\neg P(a) \vee Q(b))$
- Có thể dùng bảng chân lý để kiểm tra tính tương đương của các CTC

Bảng các công thức tương đương

Công thức tương đương		*Tên công thức*
$(G \rightarrow H)$	$((\neg G) \vee H)$	
$(G \leftrightarrow H)$	$((G \rightarrow H) \wedge (H \rightarrow G))$	
$(\neg(\neg G))$	G	
$(\neg(G \wedge H))$	$((\neg G) \vee (\neg H))$	Luật De Morgan
$(\neg(G \vee H))$	$((\neg G) \wedge (\neg H))$	
$((G \wedge (H \vee K))$	$((G \wedge H) \vee (G \wedge K))$	Luật phân phối
$((G \vee (H \wedge K))$	$((G \vee H) \wedge (G \vee K))$	
$(G \wedge H)$	$(H \wedge G)$	Luật giao hoán
$(G \vee H)$	$(H \vee G)$	
$((G \vee H) \vee K)$	$(G \vee (H \vee K))$	Luật kết hợp cho phép loại bỏ dấu ngoặc
$((G \wedge H) \wedge K))$	$(G \wedge (H \wedge K))$	
$(G \rightarrow H)$	$((\neg H) \rightarrow (\neg G))$	Luật đối vị

Bảng các công thức tương đương

Công thức tương đương		*Tên công thức*
(G ∧ □)	**G**	
(G ∧ ∇)	∇	
(G ∨ □)	□	
(G ∨ ∇)	G	
(G ∨ (¬G))	□	
(G ∧ (¬G))	∇	
(∀X)(G(X))	(∀Y)(G(Y))	*Luật dùng chung các biến*
(∃X)(G(X))	(∃Y)(G(Y))	
¬((∀X)G(X))	(∃Y)(¬G(Y))	
¬((∃X)G(X))	(∀Y)(¬G(Y))	
(∀X)(G(X) ∧ H(X))	((∀X)G(X) ∧ (∀Y)H(Y))	
(∃X)(G(X)) ∨ H(X))	((∃X)G(X) ∨ (∃Y)H(Y))	

Biến đổi các CTC : loại bỏ lượng tử ∀ và ∃

- Standardize Variables
  - Make sure that you are not using the same variable name twice in a single sentence (unless you really meant to)

    Eg.

    $(\forall X\ P(X)) \vee (\exists X\ Q(X))$ becomes $(\forall X\ P(X)) \vee (\exists Z\ Q(Z))$
- Move all quantifiers left, but keep them in order!
  - Eg.

    $(\forall X\ P(X) \vee \exists Y\ Q(Y))$ becomes $(\forall X\ \exists Y\ P(X) \vee Q(Y))$
- Translation into the form:

  Q<V> M<V>

  with Q: quantifiers, in order ∀s and then ∃s

  M: Matrix: wffs including V, <V>: Variable

Biến đổi các CTC

- Skolemize: Eliminate Existential Quantifiers
  - Existential quantifiers can be eliminated by the introduction of a new constant that does not appear elsewhere in the database
  - SUBST{ X|a, a∈D } /* *Substitution*
  - Eg.

    $\exists X\ P(X)$ becomes $P(a)$

    "*Có người vào lớp muộn*"

    becomes "*Cu Tý vào lớp muộn*"

    $\exists X\ P(X) \vee Q(X)$ becomes $P(a) \vee Q(a)$

    "*Chàng tìm đồng kia bãi nọ*"

    becomes

    "*Chàng tìm đồng dưới đồng trên*"

Biến đổi các CTC

- Skolemize: Drop Existential Quantifiers
  - One possible complication occurs if we also have Universal quantifiers… Consider

    $\forall X\ PERSON(X) \rightarrow \exists Y\ (HEART(Y) \wedge HAS(X, Y))$

    becomes by SUBST{Y|h}:

    $\forall X\ PERSON(X) \rightarrow HEART(h) \wedge HAS(X, h)$
  - Instead we have to create a new (Skolem) function to map from a person to their HEART(f(x))
    $\forall X\ \exists Y\ PERSON(X) \rightarrow (HEART(Y) \wedge HAS(X, Y))$ becomes:

    $\forall X\ PERSON(X) \rightarrow HEART(f(X)) \wedge HAS(X, f(X))$

    Eg., in general:

    $\forall X\ \forall Y\ \forall Z\ \ldots\ \exists T\ \ldots\ P(X, Y, Z,\ldots T,\ldots)$ becomes:
    $\forall X\ \forall Y\ \forall Z\ \ldots\ P(X, Y, Z,\ldots f(X, Y, Z,\ldots),\ldots)$
- Drop all Universal Quantifiers in the end:
  - $\forall X\ \forall Y\ \forall Z\ldots\ P(X, Y, Z,\ldots)$ becomes: $P(X, Y, Z,\ldots)$

Biến đổi các CTC

- Move the $\leftrightarrow$:
  - $P \leftrightarrow Q$ becomes $(P \rightarrow Q) \wedge (Q \rightarrow P)$
- Distribute $\wedge$ over $\vee$
  - $(A \wedge B) \vee C$ becomes $(A \vee C) \wedge (A \vee B)$
  - Just like distribution in arithmetic
  - $(5 + 4) * 6$ becomes $(5 * 6) + (4 * 6)$
- Flatten nested conjunctions and disjunctions
  - $(A \wedge B) \wedge C$ becomes $(A \wedge B \wedge C)$
  - $(A \vee B) \vee C$ becomes $(A \vee B \vee C)$

Done!!

- Biến đổi các CTC :
  - Standardize Variables
  - Move the $\leftrightarrow$
  - Move Quantifiers left
  - Skolemize:
    - Eliminate Existential Quantifiers
    - Drop Universal Quantifiers
  - Distribute $\wedge$ over $\vee$
  - Flatten nested conjunctions and disjunctions
- Attention:
  - In Proof Theory by Robinson Resolution, they need to before:
    - Eliminate Implications
    - Move $\neg$ inwards

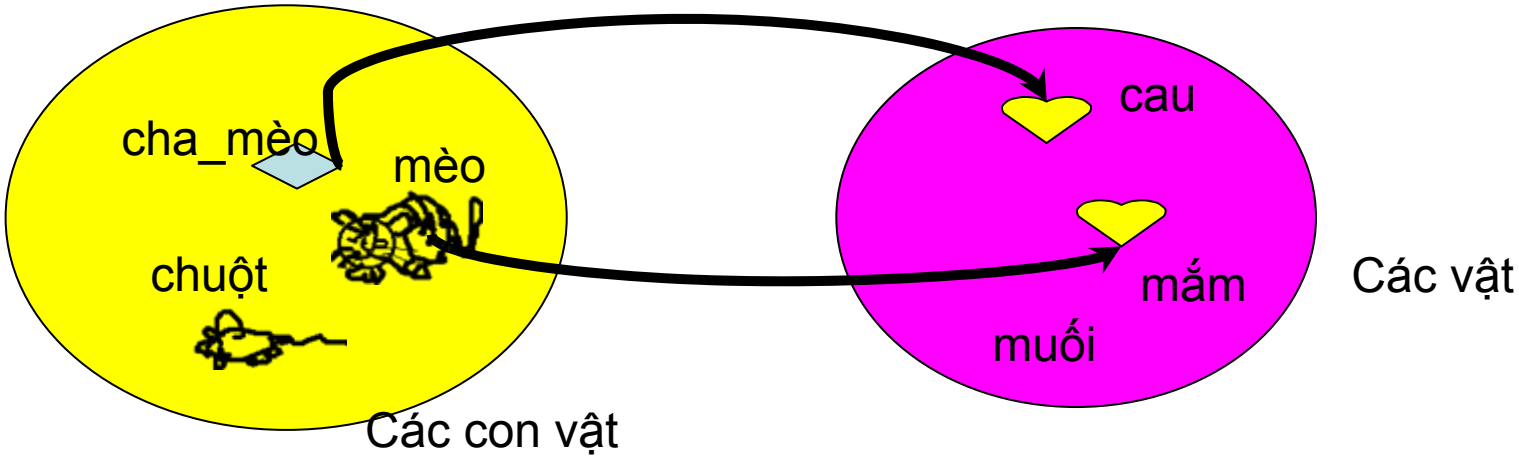
Xây dựng cơ sở luật cho HCG

- Sự kiện 1 :
  - Con mèo mà trèo cây cau
    Hỏi thăm chú chuột đi đâu vắng nhà
    Chú chuột đi chợ đường xa
    Mua mắm, mua muối về giỗ cha con mèo
- Hỏi :
  - Mèo có ăn được (gặp) chuột không rứa ?
- Sự kiện 2 :
  - Ông Trăng mà lấy bà Trời
    Tháng Năm đi cưới, tháng Mười nộp cheo
    Làng xã làm thịt một con mèo
    Làng ăn không hết làng treo cột đình
    Ông Xã đánh trống thình thình
    Bao nhiêu con nít ra đình gặm xương
- Hỏi :
  - Cu Tý có gặm được xương không rứa ?

Tạo không gian : các miền xác định

Sự kiện 1 :

- Con mèo mà trèo cây cau : TREO(X, Y)
- Hỏi thăm chú chuột đi đâu vắng nhà :THAM(X, Y), VANGNHA(X)
- Chú chuột đi chợ đường xa
- Mua mắm, mua muối về giỗ cha con mèo



Gợi ý dùng luật :

$MEO(X) \wedge CHUOT(Y) \wedge THAM(X, Y) \wedge COONHA(Y) \rightarrow ANTHIT(X,Y)$

Hệ chuyên gia (Expert System)

PGS.TS. Phan Huy Khánh

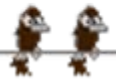
khanhph@vnn.vn

Chương 3

Máy suy diễn

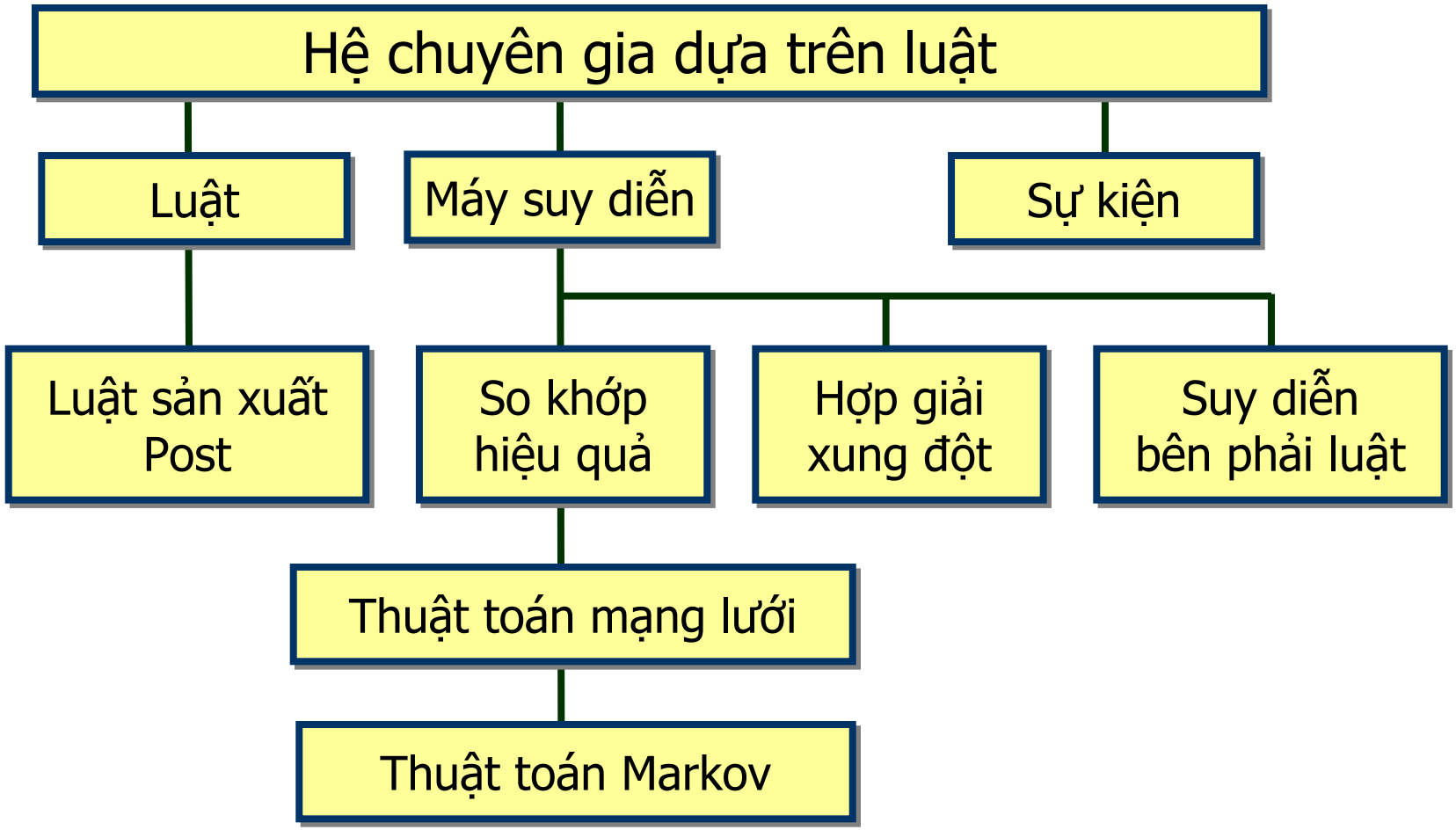
Chương 3 Máy suy diễn

- Các hệ thống sản xuất Post
- Thuật toán Markov và thuật toán mạng lưới
- Nguyên lý hoạt động của các máy suy diễn
- Một số phương pháp suy diễn thông dụng
  - Phương pháp suy diễn tiến
  - Phương pháp suy diễn lùi
  - Phương pháp hỗn hợp

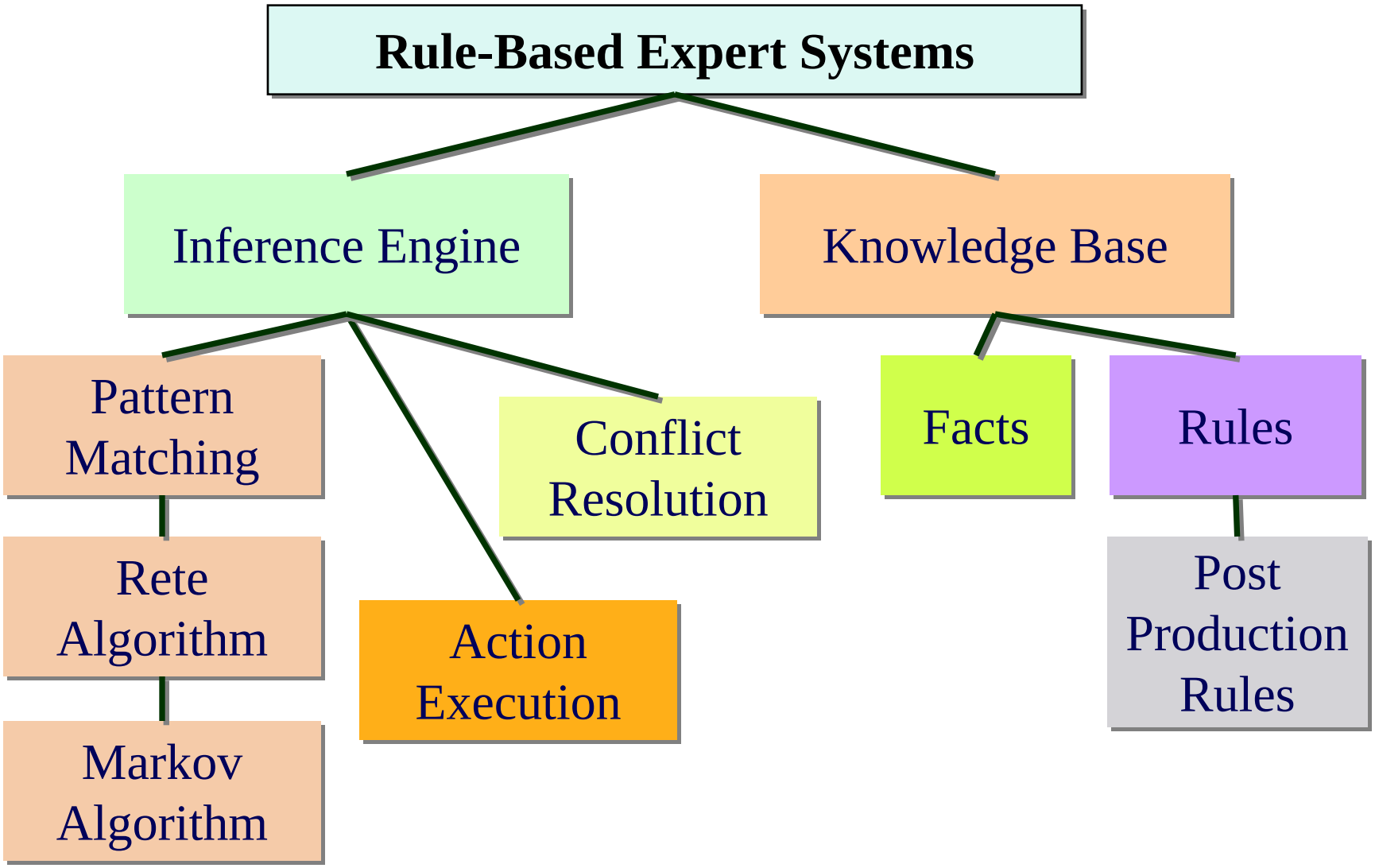


Nền tảng của công nghệ hệ chuyên gia hiện đại

(Foundation of Modern Rule-Based Expert System)



Foundations of Expert Systems



Hệ thống sản xuất Post

- Hệ thống sản xuất (SX) Post (Post production systems)
  - SX Post (production rule, also called condition-action, or situation-action rules)
  - Mỗi có dạng :

    $< \textit{xâu tiền đề} > \rightarrow < \textit{xâu kết quả} >$
  - Ý tưởng cơ bản của Post là :
    - Xuất phát từ một xâu *tiền đề* (antecedent)
    - Sản xuất ra một xâu kết quả mới khác (consequent)
  - Dấu mũi tên $\rightarrow$ chỉ ra rằng :
    xâu vào bên trái được *chuyển* (transformation)
    thành xâu kết quả bên phải

Lịch sử các hệ thống SX Post

- First developed by Post (1943), who studied the properties of rule systems based on productions & called his systems canonical systems
  - Production rules are grammar rules for manipulating strings of symbols, in automata theory, formal grammars, programming language design & used for psychological modeling before they were used for expert systems
  - Also called rewrite rules (they rewrite one string into another)
  - He proved any system of mathematics or logic could be written as a type of production rule system
  - Minsky showed that any formal system can be realized as a canonical system
  - Các ngôn ngữ lập trình thường được định nghĩa từ dạng chuẩn Backus-Naur Normal Form (BNF)

Example of a Canonical System

- Let the alphabet $\Sigma = \{a, b, c\}$

  With axioms a, b, c, aa, bb, cc
- Then these production rules will give :
  - all the possible palindromes (and only palindromes)
  - based on the alphabet, starting from the above axioms

    P1: $ -> a$a

    P2: $ -> b$b

    P3: $ -> c$c
- To generate the string *bacab*:
  - P1 is applied to the axiom *c* to get *aca* ($ = *c*)
  - Then we apply P2 to get *bacab*
    Using a different order gives a different result
  - If P2 is applied to *c* we get *bcb*
  - If P1 is applied after we get *abcba*

Ví dụ 2

- Cho hệ thống SX Post gồm các luật SX như sau
  (Chú ý số thứ tự trong dấu ngoặc chỉ dùng để trình bày) :

1. Car won't start	→ Check battery
2. Car won't start	→ Check gas
3. Check battery AND Battery bad	→ Replace battery
4. Check gas AND No gas	→ Fill gas tank

- Nếu đưa vào xâu *Car won't start*, thì các luật 1 và 2 có thể được áp dụng để sinh ra các xâu *Check battery* và *Check gas*
- Nếu đưa vào xâu *Battery bad* và *Check battery* thì luật 3 có thể được áp dụng để sinh ra xâu *Replace battery*

Hoạt động của hệ thống SX Post

- Hệ thống SX Post :
  - Không có cơ chế áp dụng đồng thời cả hai xâu vào
  - Chỉ có thể áp dụng được một luật trong hai, hoặc không
  - Không đặt ra thứ tự các luật trong hệ thống
  - Hệ thống giữ nguyên giá trị khi đảo thứ tự các luật

1. Car won't start → Check battery
2. Car won't start → Check gas
3. Check battery AND Battery bad → Replace battery
4. Check gas AND No gas → Fill gas tank

4. Check gas AND No gas → Fill gas tank
2. Car won't start → Check gas
1. Car won't start → Check battery
3. Check battery AND Battery bad → Replace battery

Nhận xét về hạn chế của SX Post

- Mặc dù các SX Post được sử dụng trong HCG nhưng không thuận tiện cho việc viết các trình ứng dụng
- Hạn chế chủ yếu của các sản xuất Post :
  - Không có các *chiến lược điều khiển* (control strategy) để định hướng sử dụng luật
  - Chỉ áp dụng luật cho một xâu vào theo cách tuỳ ý mà không chỉ ra cụ thể làm thế nào để luật được áp dụng
  - Sự lựa chọn luật một cách ngẫu nhiên làm mất nhiều thời gian tìm kiếm trong các hệ thống có nhiều luật

Thuật toán Markov

- Thuật toán Markov (Markov Algorithm) :
  - Đề xuất năm 1954 cải tiến cách áp dụng các SX từ một xâu vào
  - Nhóm các sản xuất theo thứ tự độ ưu tiên
  - Nếu SX có độ ưu tiên cao nhất không được áp dụng, thì SX tiếp theo sẽ được áp dụng và cứ thế tiếp tục
  - Thuật toán Markov dừng nếu :
    - sản xuất cuối cùng không được áp dụng cho xâu, hoặc
    - nếu sản xuất đó là cuối một giai đoạn được áp dụng
- A Markov algorithm:
  - is a string rewriting system that uses grammar-like rules to operate on strings of symbols
  - Markov algorithms have been shown to have sufficient power to be a general model of computation.
  - Important difference from canonical system: now the set of rules is **ordered**

Markov Algorithm (MA)

- The basic operation:
  1. Check the Rules in order from top to bottom to see whether any of the strings to the left of the arrow can be found in the Symbol string
  2. If none are found, stop executing the Algorithm
  3. If one or more is found, replace the leftmost matching text in the Symbol string with the text to the right of the arrow in the first corresponding Rule
  4. If the applied rule was a terminating one, stop executing the Algorithm
  5. Return to step 1 and carry on

Áp dụng thuật toán Markov

- Thuật toán Markov có thể áp dụng cho từng xâu con của xâu vào $w \in \Sigma^*$, bắt đầu từ bên trái :
  - Ví dụ : áp dụng luật *ab* $\rightarrow$ *hij* cho xâu vào *gabkab*
  - Nhận được xâu mới *ghijkab*
    Tiếp tục nhận được xâu mới *ghijkhij*
- Ký tự đặc biệt ε biểu diễn xâu rỗng. Ví dụ :
  - A $\rightarrow \varepsilon$ xóa tất cả các xuất hiện của A trong một xâu
  - A x B $\rightarrow$ B x A nghịch đảo các ký tự A và B
- Các ký hiệu đặc biệt khác có vai trò như biến biểu diễn một ký tự bất kỳ được viết bởi các chữ cái thường a, b, c...
- Các chữ cái Hy lạp α, $\beta \notin \Sigma$, chỉ các dấu đặc biệt của xâu

Ví dụ sử dụng thuật toán Markov

- Cho các SX có độ ưu tiên giảm dần như sau (ký hiệu ε hoạt động như là một biến trung gian) :

  εxy → yεx

  ε → ε

  ε → ε

- Cho xâu vào *abc*, cần di chuyển chữ cái đầu tiên *a* đến vị trí cuối cùng của xâu
- Quá trình di chuyển được cho trong bảng

Luật	*Thành công (S) hoặc thất bại (F)*	*Xâu kết quả*
1	F	*abc*
2	F	*abc*
3	S	*εabc*
1	S	*bεac*
1	S	*bcεa*
1	F	*bcεa*
2	S	*bca*

Another Example of MA

- Rules:
  1. "A" -> "apple"
  2. "B" -> "bag"
  3. "S" -> "shop"
  4. "T" -> "the"
  5. "the shop" -> "my brother"
  6. "a never used" -> ."terminating rule"
- Symbol string :

  "I bought a B of As from T S."

- The Symbol string will change in the following manner:

  R1: "I bought a B of apples from T S." .

  R2: "I bought a bag of apples from T S."

  R3: "I bought a bag of apples from T shop."

  R4: "I bought a bag of apples from the shop."

  R5: "I bought a bag of apples from my brother."

- The algorithm will then terminate

Another Example of MA

- They rewrite binary numbers to their unary counterparts
- For example: 101 will be rewritten to a string of 5 consecutive bars
- Rules:

  "|0" -> "0||"

  "1" -> "0|"

  "0" -> ""
- Symbol string:

  "101"

- If the algorithm is applied to the above example, it will terminate after the following steps
- Execution:

  "0|01"

  "00||1"

  "00||0|"

  "00|0|||"

  "000||||"

  "00|||||"

  "0|||||"

  "|||||"

- MA chọn áp dụng luật có độ ưu tiên nhất theo chiến lược điều khiển tất định (definite control strategy)
- Nếu không chọn được, MA tìm luật khác có độ ưu tiên thấp hơn
- MA thiếu tính hiệu quả trong những hệ chuyên gia có nhiều luật

Thuật toán mạng lưới (Rete Algorithm)

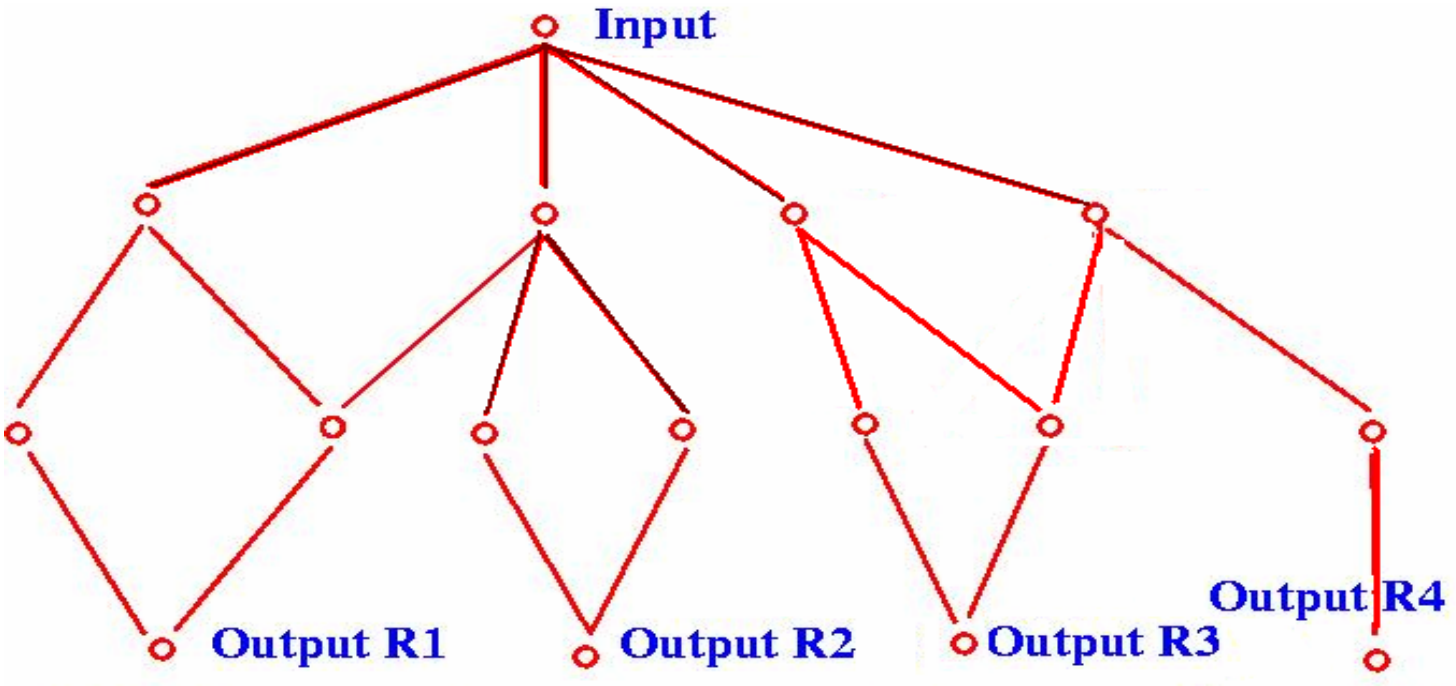
- Do Charles L. Forgy đề xuất năm 1979 tại trường ĐH Carnegie, Mellon, Hoa Kỳ trong luận văn tiến sĩ của ông về OPS (Official Production System)
- Thuật toán mạng lưới giải quyết vấn đề *hiệu suất* (efficient) của các hệ chuyên gia :
  - Đóng vai trò quan trọng khi giải quyết các bài toán thực tiễn chứa từ hàng trăm đến hàng ngàn luật
  - NSD không phải chờ đợi nhiều thời gian để nhận được câu trả lời
- Cần có thuật toán xử lý hết các luật để chọn ra các luật cần thiết để áp dụng thay vì thử lần lượt các luật

RETE algorithm

- Thuật toán mạng lưới :
  - Cho phép so khớp (pattern mattching) rất nhanh để nhận được câu trả lời tức thời bằng cách lưu giữ thông tin của các luật trong một mạng lưới (network)
  - Thay vì so khớp lặp đi lặp lại các sự kiện mỗi lần áp dụng một luật trong mỗi chu trình nhận thức (recognize-act cycle), thuật toán mạng lưới chỉ nhìn những thay đổi khi so khớp trong mỗi chu trình
- Activities
  - Creates a decision tree where each node corresponds to a pattern occurring at the left-hand side of a rule
  - Each node has a memory of facts that satisfy the pattern
  - Complete LHS as defined by a path from root to a leaf

The Rete-Algorithm

- The net encodes the **condition parts** (IF-parts) of the rules
- The input are the **changes** of the working memory, i.e.:
  - New elements or deleted elements
  - Modification of elements is simulated by first delete then add modified version)
- The output is the **conflict set** (i.e., the applicable rules)



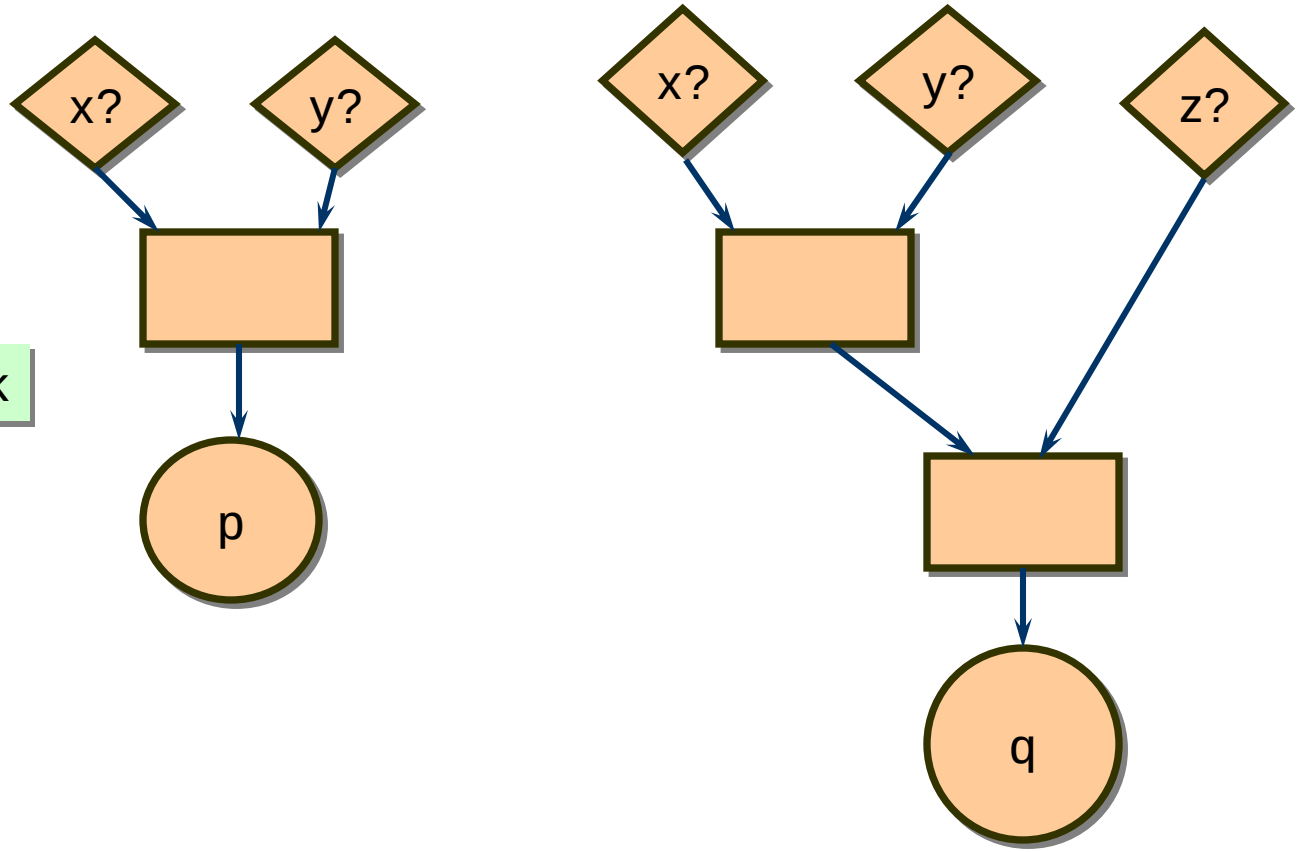
Rete example

Rules: IF x & y THEN p
IF x & y & z THEN q

Pattern Network

Join Network

8 nodes



(http://aaaprod.gsfc.nasa.gov/teas/Jess/JessUMBC/sld010.htm)

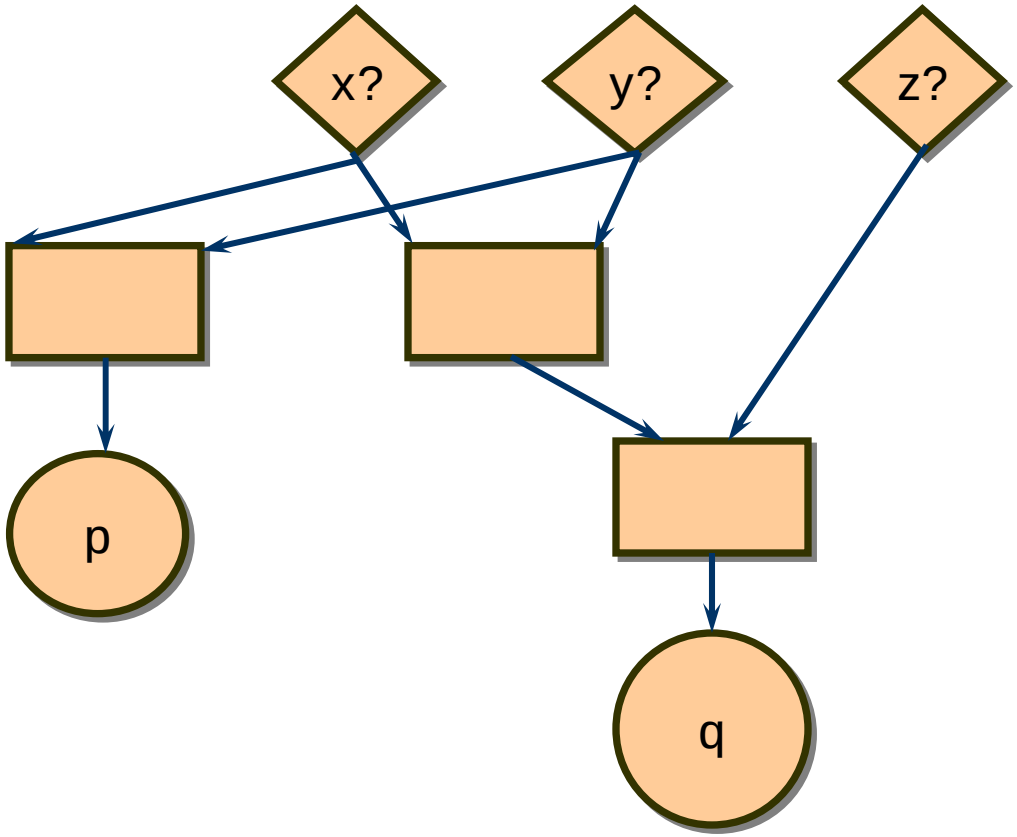
Rete example

Rules: IF x & y THEN p
IF x & y & z THEN q

Pattern Network

Join Network

8 nodes



(http://aaaprod.gsfc.nasa.gov/teas/Jess/JessUMBC/sld010.htm)

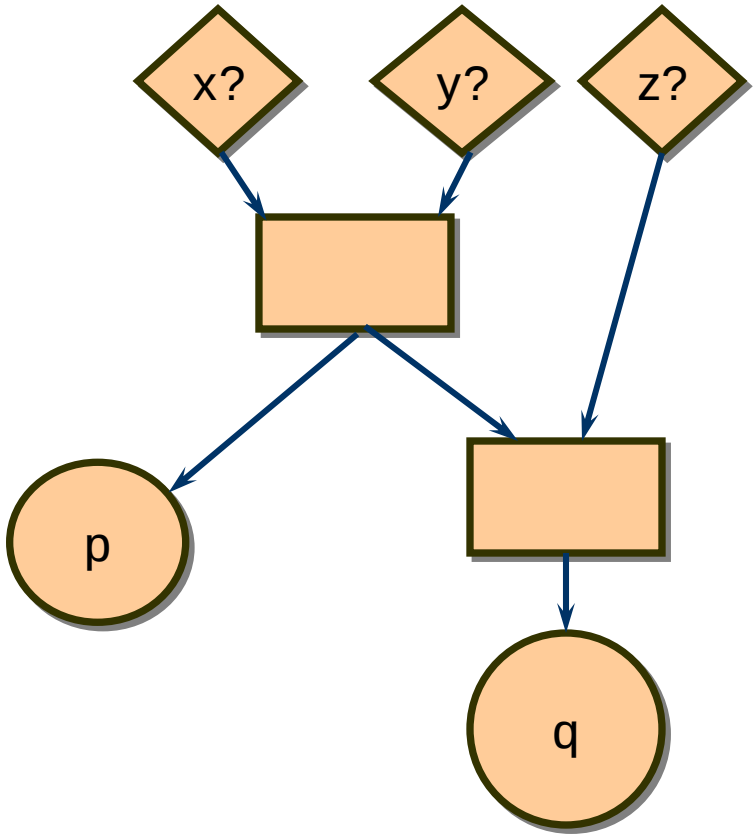
Rete example

Rules: IF x & y THEN p
IF x & y & z THEN q

Pattern Network

Join Network

8 nodes



(http://aaaprod.gsfc.nasa.gov/teas/Jess/JessUMBC/sld010.htm)

Matching Patterns

- At each cycle the interpreter looks to see which rules have conditions that can be satisfied
- If a condition has no variables:
  - It will only be satisfied by an identical expression in working memory
- If the condition contains variables then
  - It will be satisfied if there is an expression in working memory with an attribute-value pair that matches it in a way that is consistent with the way other conditions in the same rule have already been matched

Rule-Based Production Systems

- A production system consists of
  - a rule set / knowledge base / production memory
  - a rule interpreter / inference engine
    - that decides when to apply which rules
  - a working memory
    - that holds the data, goal statements, & intermediate results that make up the current state of the problem.
- Rules have the general form

  IF <pattern> THEN <action>

  $P_1, \ldots, P_m \rightarrow Q_1, \ldots, Q_n$
- Patterns are usually represented by OAV vectors

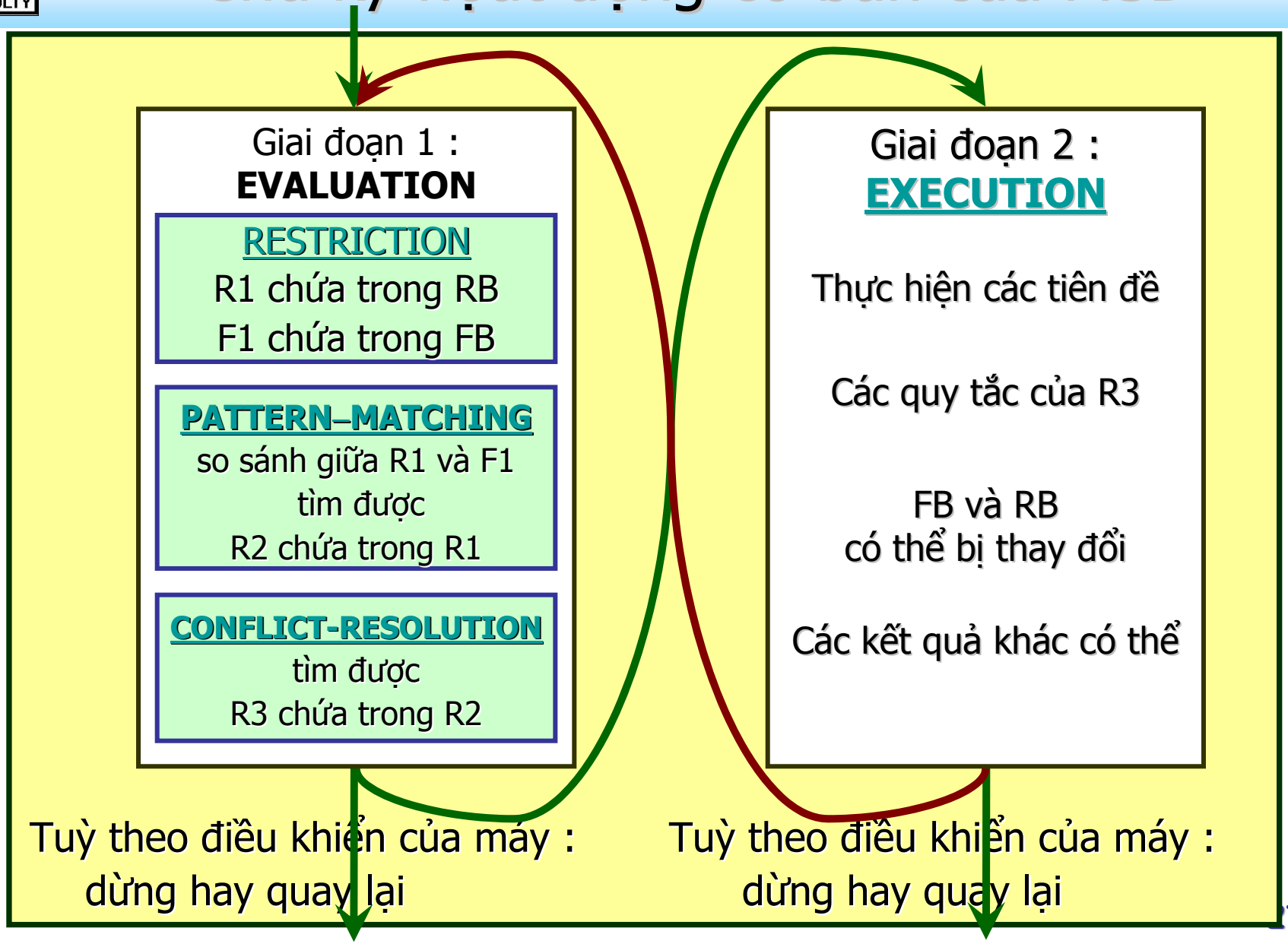
Nguyên lý hoạt động của các máy suy diễn

- Trong các hệ thống dùng luật, khi máy duy diễn (MSD) được khởi động, cơ sở tri thức chứa thông tin liên quan đến phát biểu bài toán cần giải :
  - Các sự kiện đã được xác nhận và các sự kiện sẽ được thiết lập biểu diễn bài toán hay đích
  - Những tri thức thực hành thuộc lĩnh vực tạo nên cơ sở luật
- Hoạt động suy diễn của MSD :
  - Suy luận bằng cách quyết định xem những luật nào sẽ làm thỏa mãn các sự kiện, các đối tượng
  - Chọn ưu tiên các luật thỏa mãn
  - Thực hiện các luật có tính ưu tiên cao nhất

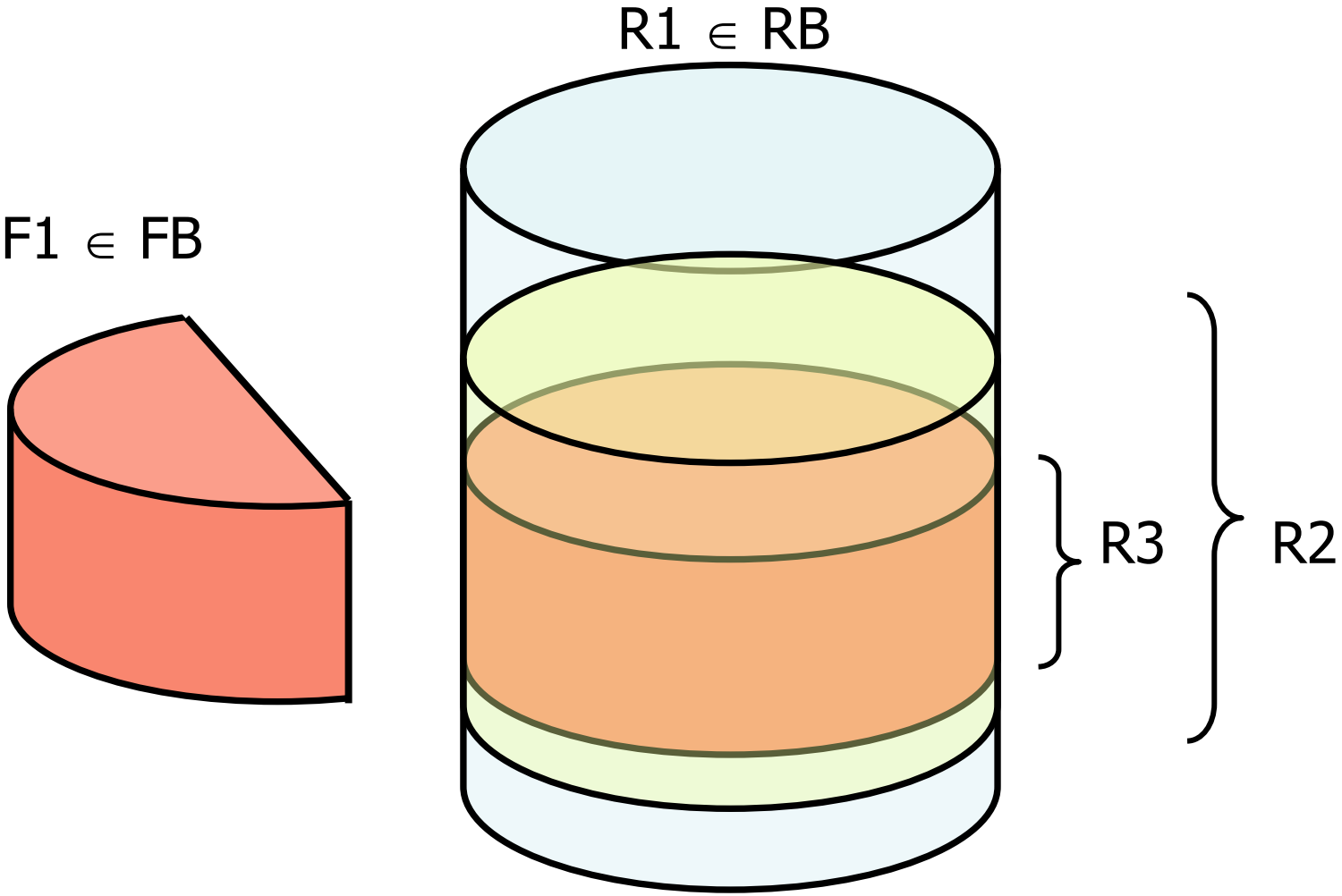
Một số quy ước

- Ta quy ước gọi :
  - RB (Rules Base) là *cơ sở luật* (CSL)
  - FB (Facts Base) là *cơ sở sự kiện* (CSSK)
- Máy suy diễn hoạt động theo *chu kỳ* (cycle), mỗi chu kỳ gồm hai *giai đoạn* (phase) :
  - Giai đoạn đánh giá (EVALUATION), gồm ba *bước* (step) :
    - Thu hẹp (RESTRICTION. hay SELECTION)
    - So khớp (PATTERN–MATCHING, hay FILTERING)
    - Giải quyết xung đột (CONFLICT-RESOLUTION)
  - Thực hiện (EXECUTION)

Chu kỳ hoạt động cơ bản của MSD



Mô hình so khớp luật trong một chu kỳ



Tiếp tục

Bước thu hẹp

- Là bước đầu tiên của giai đoạn đánh giá
- Đầu vào :
  - Một sự kiện f $\in$ FB (CSSK) và/hoặc luật r $\in$ RB (CSL)
- Đầu ra :
  - Một tập hợp con F1 $\subseteq$ FB
  - Một tập hợp con R1 $\subseteq$ RB sao cho có thể tiến hành so sánh được trong bước FILTERING tiếp theo
- Nguyên lý làm việc :
  - Ưu tiên cho một nhóm các luật
    hay một nhóm các sự kiện đối với một hoặc nhiều chu kỳ

Bước so khớp

- Là bước thứ hai của giai đoạn đánh giá
- Đầu vào :
  - Một sự kiện f $\in$ FB (CSSK) và/hoặc luật r $\in$ RB (CSL)
- Đầu ra :
  - Một tập hợp con các luật R2$\subseteq$R1 tương thích với F1, nghĩa là những luật r$\in$R2 có điều kiện khởi động thoả mãn các trạng thái của F1
- Nguyên lý làm việc :
  - Máy suy diễn so sánh phần khởi động của mỗi quy tắc của R1 với tập hợp các sự kiện F1
  - Tuỳ theo hệ thống mà có tiêu chuẩn thoả mãn khác nhau
  - R2 được gọi là *tập hợp xung đột* hay *tập hợp tương tranh* (conflict set)

Bước giải quyết xung đột

- Là bước thứ ba của giai đoạn đánh giá
- Đầu vào :
  - Tập hợp các luật trong tập xung đột R2
- Đầu ra :
  - Tập hợp con các luật R3 $\subseteq$ R2 cần được thực hiện
- Nguyên lý làm việc :
  - Nếu R3 rỗng, không thực hiện giai đoạn EXECUTION của chu kỳ
  - Nếu R3$\neq\varnothing$, chọn các luật dựa trên những tiêu chuẩn như sau :
    - Hoặc chọn theo thứ tự xuất hiện của luật với giả thiết RB đã được sắp xếp theo một thứ tự nào đó
    - Hoặc chọn các luật có mối quan hệ với bối cảnh áp dụng liên quan đến nghĩa (meaning/signification)
    - Hoặc chọn ngẫu nhiên : ưu tiên luật ít sử dụng, hoặc ít điều kiện cần kiểm tra, ít biến cần xác định trước khi khởi động, v.v...

Return

Conflict Resolution

- Selection a rule to fire
  - Production systems have a decision-making step between pattern matching & rule firing
  - All rules that have their conditions satisfied are put on the agenda (in CLIPS for ex.)
  - The set of rules on the agenda is sometimes called the conflict set
  - Conflict resolution selects which rule to fire from the agenda (Packages like CLIPS provide more than one option for conflict resolution)
  - Sensibility (quick response to changes in WM) and Stability (continuous reasoning)

Conflict Resolution in CLIPS

- First, CLIPS uses salience to sort the rules
- Then it uses the other strategies to sort rules with equal salience
- CLIPS uses refraction, recency & specificity in the form of following 7 strategies:
  - The depth strategy
  - The breadth strategy
  - The simplicity strategy
  - The complexity strategy
  - The LEX strategy
  - The MEA strategy
  - It is possible also to set strategy to random
- Syntax: (set-strategy <strategy>)

Giai đoạn thực hiện EXECUTION

- Khi R3 rỗng, tuỳ theo HCG mà có hai cách xử lý như sau :
  - Cho MSD tự động dừng : những MSD như vậy được gọi là hoạt động theo *chế độ điều khiển bắt buộc* (irrevocable control regime)
  - Cho MSD xem xét lại tập hợp xung đột R2 của chu kỳ trước đó và áp dụng một luật khác của R2
- Trường hợp không tìm được luật trong R2 :
  - Nếu sử dụng kết quả của các luật đã áp dụng, thì người ta cũng gọi những MSD như vậy hoạt động theo chế độ điều khiển bắt buộc
  - Ngược lại, người ta gọi MSD hoạt động theo *chế độ thăm dò* (tentative control regime)
- Khi MSD quay lại giải quyết các xung đột trước đó, bằng cách khởi động lại các luật, người ta gọi máy hoạt động quay lui (backtrack)

The Working Memory

- Role:
  - Holds data in the form of OAV vectors
  - These data are then used by the interpreter to activate the rules of RB
  - The presence, or absence, of data elements in the Working Memory will trigger rules by satisfying patterns on the LHS part of rules
  - Actions such as assert or modify the Working Memory

Nhận xét về chu kỳ cơ bản của MSD

- Trên thực tế, có nhiều cách sắp đặt các giai đoạn trong một chu kỳ cơ bản (hay chu kỳ suy diễn) của MSD
- Khi HCG làm việc, có thể cần đến hàng hàng trăm, thậm chí hàng ngàn chu kỳ
- Mỗi chu kỳ cơ bản của MSD :
  - Có liên hệ với chu kỳ lệnh của máy tính
  - Cần những máy tính có tốc độ lớn (hàng trăm chu kỳ cơ bản trong một giây)
  - Chẳng hạn dự án máy tính thế hệ 5 của Nhật đề xuất những kiến trúc đặc trưng để đạt được tốc độ hàng triệu hay hàng tỷ *lips* (Logical Inference Per Second)

Phương pháp suy luận của các MSD

- Có nhiều phương pháp tổng quát để suy luận trong các chiến lược giải quyết vấn đề của HCG
- Ba phương pháp hay gặp :
  - Phương pháp *suy diễn tiến* (Foward Chaining/Data-Driven)
  - Phương pháp *suy diễn lùi* (Backward Chaining/Goal-Driven)
  - Phương pháp phối hợp hai phương pháp này (Mixed Chaining)
- Ngoài ra còn một số phương pháp khác :
  - *Phân tích phương tiện* (means-end analysis)
  - *Rút gọn vấn đề* (problem reduction)
  - *Kiểm tra lập kế hoạch* (plan-generate-test)
  - *Lập kế hoạch phân cấp* (hierachical planning)...

Phương pháp suy diễn tiến

- Suy diễn tiến là lập luận từ các sự kiện, sự việc để rút ra các kết luận
  - Ví dụ :
    Nếu thấy trời mưa trước khi ra khỏi nhà (sự kiện)
    thì phải lấy áo mưa (kết luận)
- NSD cung cấp các sự kiện để MSD tìm cách rút ra các kết luận có thể
  - Kết luận có thể là những thuộc tính được gán giá trị
  - Trong số những kết luận này, có thể có :
    - những kết luận làm NSD quan tâm
    - một số khác không nói lên điều gì
    - một số khác có thể vắng mặt

Các sự kiện trong suy diễn tiến

- Các sự kiện thường có dạng :
  - Atthibute = value
  - Lần lượt các sự kiện trong cơ sở tri thức được chọn
  - Hệ thống xem xét tất cả các luật mà các sự kiện này xuất hiện như là tiền đề
  - Khi MSD tìm thấy những luật thoã mãn, MSD lấy ra để gán giá trị cho các thuộc tính thuộc kết luận tương ứng, người ta nói rằng các sự kiện đã được thoã mãn
  - Các thuộc tính được gán giá trị sẽ là một phần của kết quả chuyên gia
  - Sau khi mọi sự kiện đã được xem xét, kết quả được xuất ra cho NSD

Forward Chaining

- The inference engine works from the initial content of the workspace towards the final conclusion
- Conclude from "A" and "A implies B" to "B"

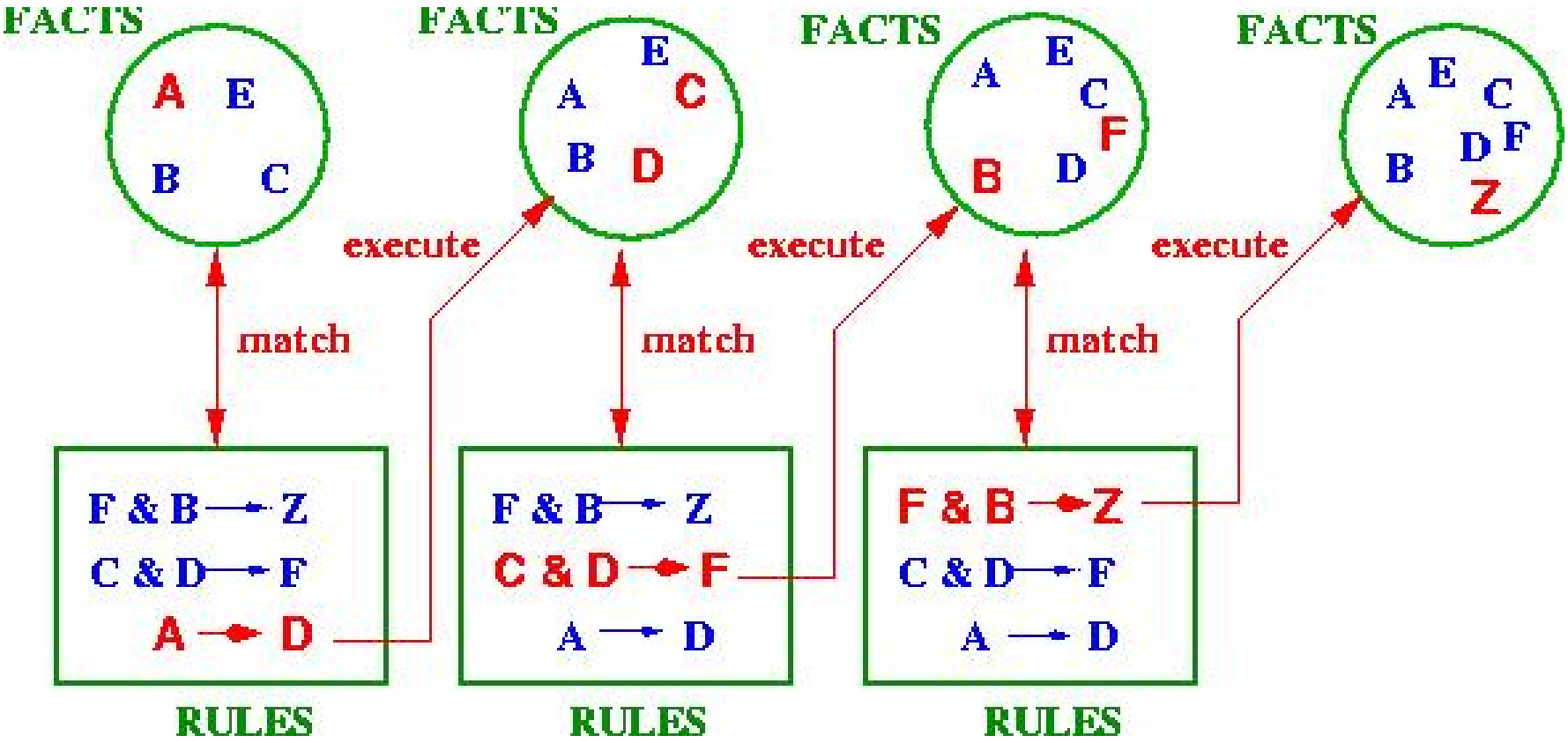
$$\frac{A \qquad A \rightarrow B}{B}$$

- Example:

  It is raining
  If it is raining, the dress is wet

  The dress is wet

Một ví dụ khác



Palindrome Example

- If we have the following grammar rules

  (P1) $ -> a$a

  (P2) $ -> b$b

  (P3) $ -> c$c

- They can be used Forward Chaining to generate palindromes:
  - apply P1, P1, P3, P2, to c:

  ⇒ *aca, aacaa, caacaac*, ...

- To generate *bacab*
  - P1 is applied to the axiom *c* to get *aca*
  - Then we apply P2 to get *bacab*
  - Using a different order gives a different result
  - If P2 is applied to *c* we get *bcb*
  - If P1 is applied after we get *abcba*

Example 1

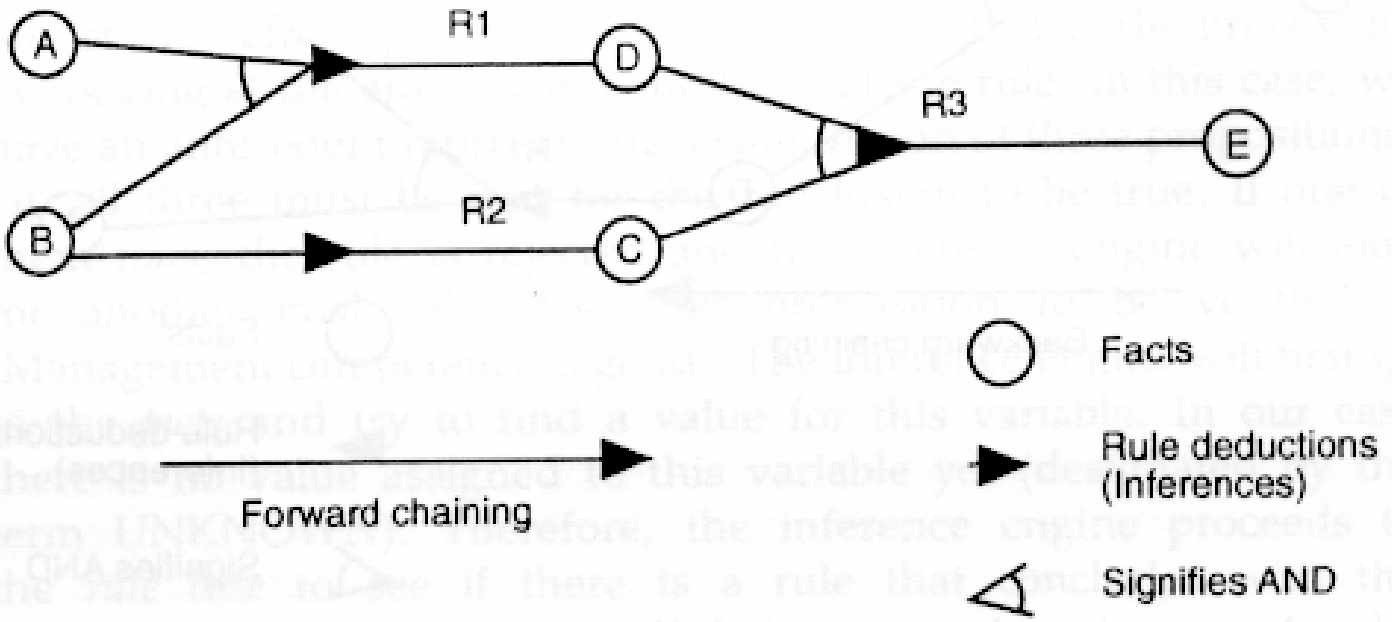
Rule base

R1: IF A AND B THEN D A,B

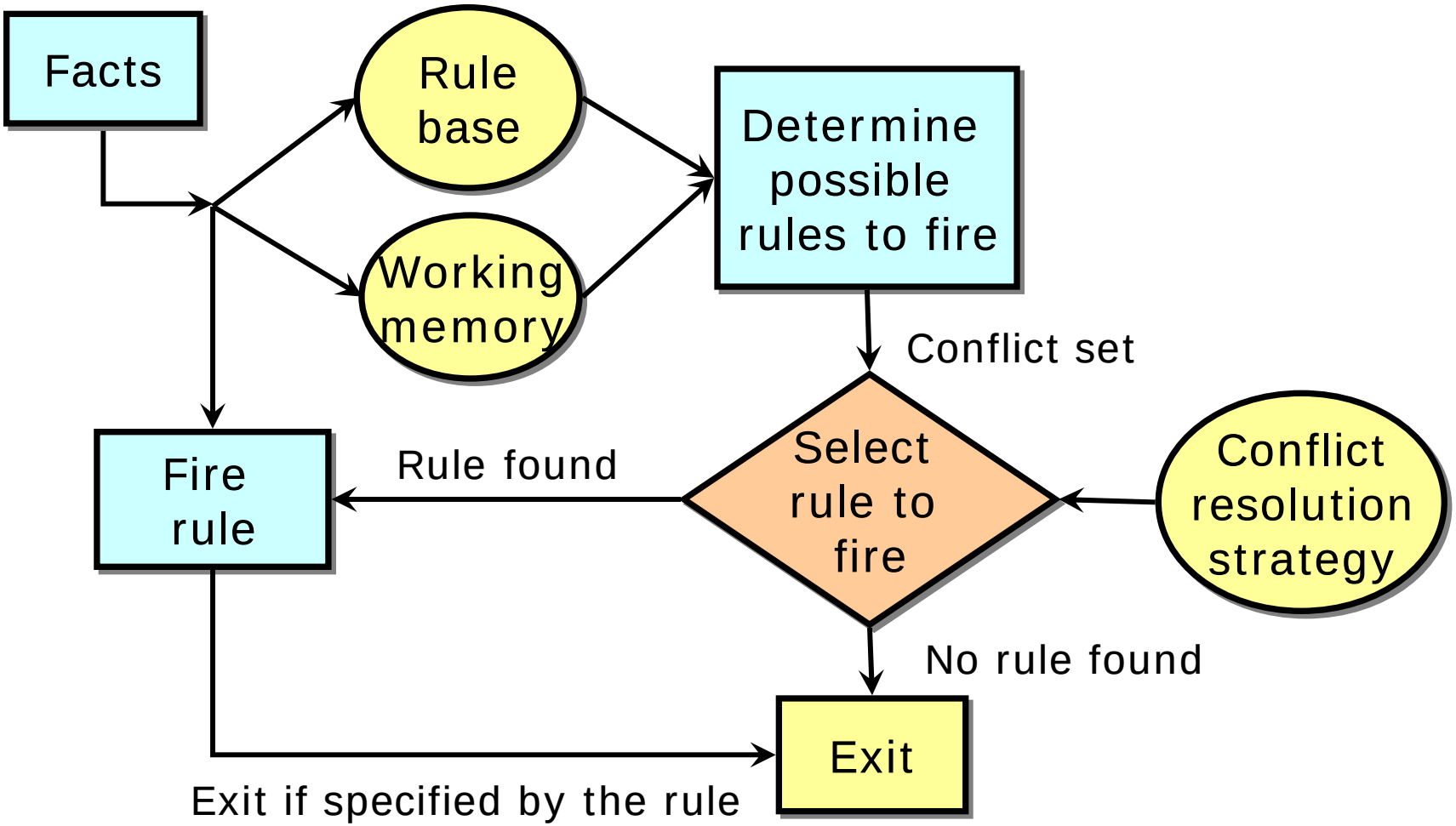
R2: IF B THEN C

R3: IF C AND D THEN E

Workspace



Forward Chaining Model



Thuật toán suy diễn tiến

- Definitions :

  FB, RB, R1 $\subseteq$ RB, Q : Sự kiện đưa vào xử lý

- Algorithme :

```
Procedure DEDUCE(Q)
If Q ∈ FB Then Write "Success" ;
Else Call CYCLE(RB) ;          ' Bắt đầu chu kỳ xử lý với r1 ∈RB
Procedure CYCLE(R1)
If R1 = ∅ Then Write "Failure" ; Return ;
    r ← CHOIX(r, R1) ;         ' Chọn một luật từ R1
    R1 ← R1 – { r } ;          ' Loại bỏ luật đã xử lý
If LHF(r) ∈ FB Then
    If Q ∈ RHF(r) Then Write " Success" ; Return ;
    Else ' Không tìm thấy Q nhưng vẫn tiếp tục áp dụng luật r
        FB ← FB ∪ RHF(r) ;     ' Thêm vào FB các sự kiện mới
        RB ← RB – { r } ;      ' Loại bỏ luật r đã xử lý
        Call  CYCLE(RB) ;
Else Call  CYCLE(R1) ;
Return ;
```

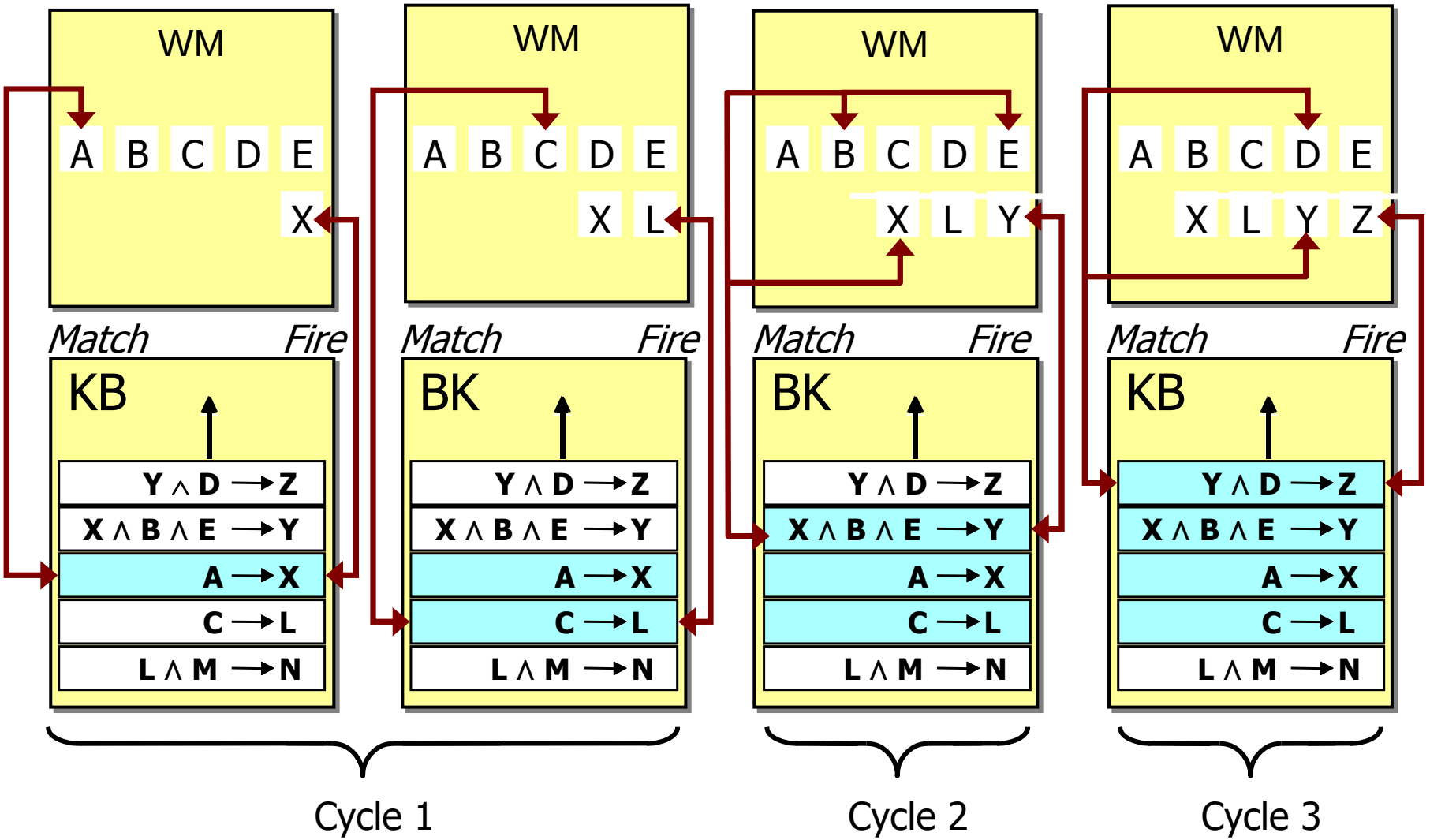
Forward chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

repeat until *new* is empty
- $new \leftarrow \{\}$
- **for each** sentence r **in** KB **do**
  - $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
  - **for each** θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$ for some $p'_1, \ldots, p'_n$ in KB
    - $q' \leftarrow$ SUBST(θ, q)
    - **if** q' is not a renaming of a sentence already in KB or new **then do**
      - add q' to new
      - $\phi \leftarrow$ UNIFY(q', α)
      - **if** ϕ is not *fail* **then return** ϕ
- add new to KB

return *false*

Forward chaining Example 2.

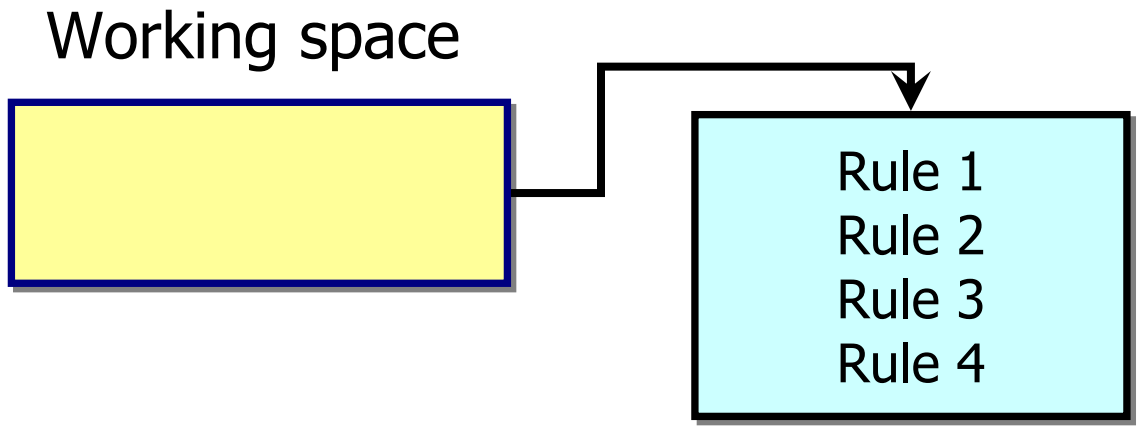


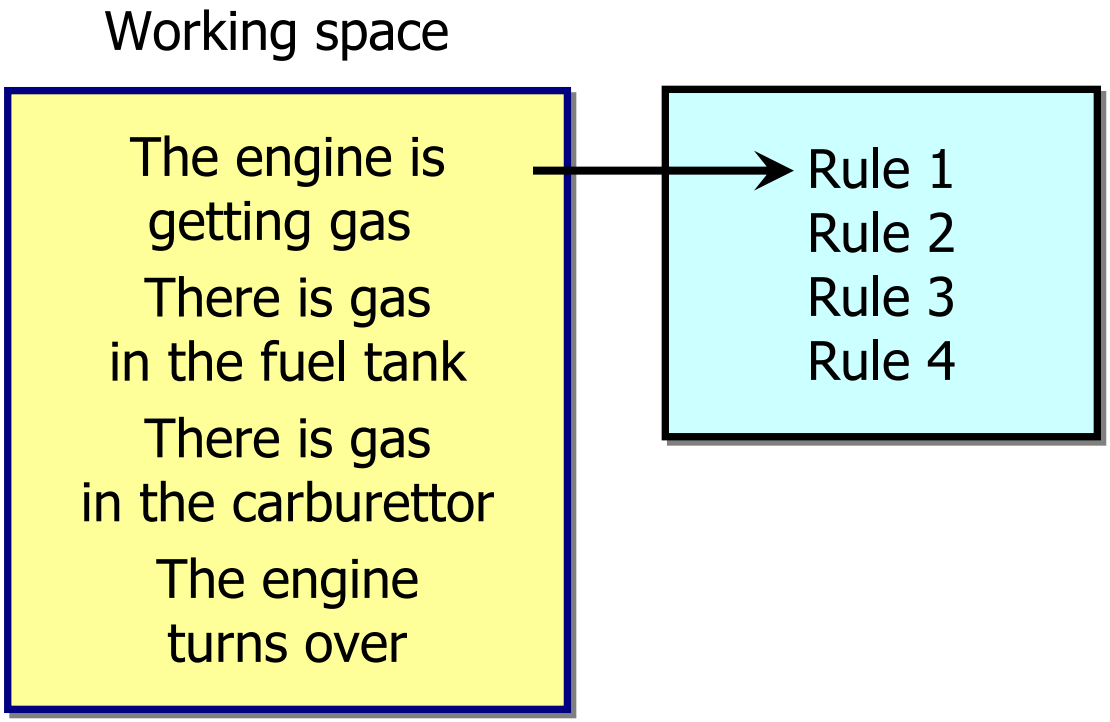
Example 3. Diagnosing car problems

- Rule 1:
  IF the engine is getting gas
  AND the engine will turn over
  THEN the problem is spark plugs
- Rule 2:
  IF the engine does not turn over
  AND the lights do not come on
  THEN the problem is battery or cables
- Rule 3:
  IF the engine does not turn over
  AND the lights do come on
  THEN the problem is the starter motor
- Rule 4:
  IF there is gas in the fuel tank
  AND there is gas in the carburettor
  THEN the engine is getting gas

⌘ Facts

The engine is getting gas

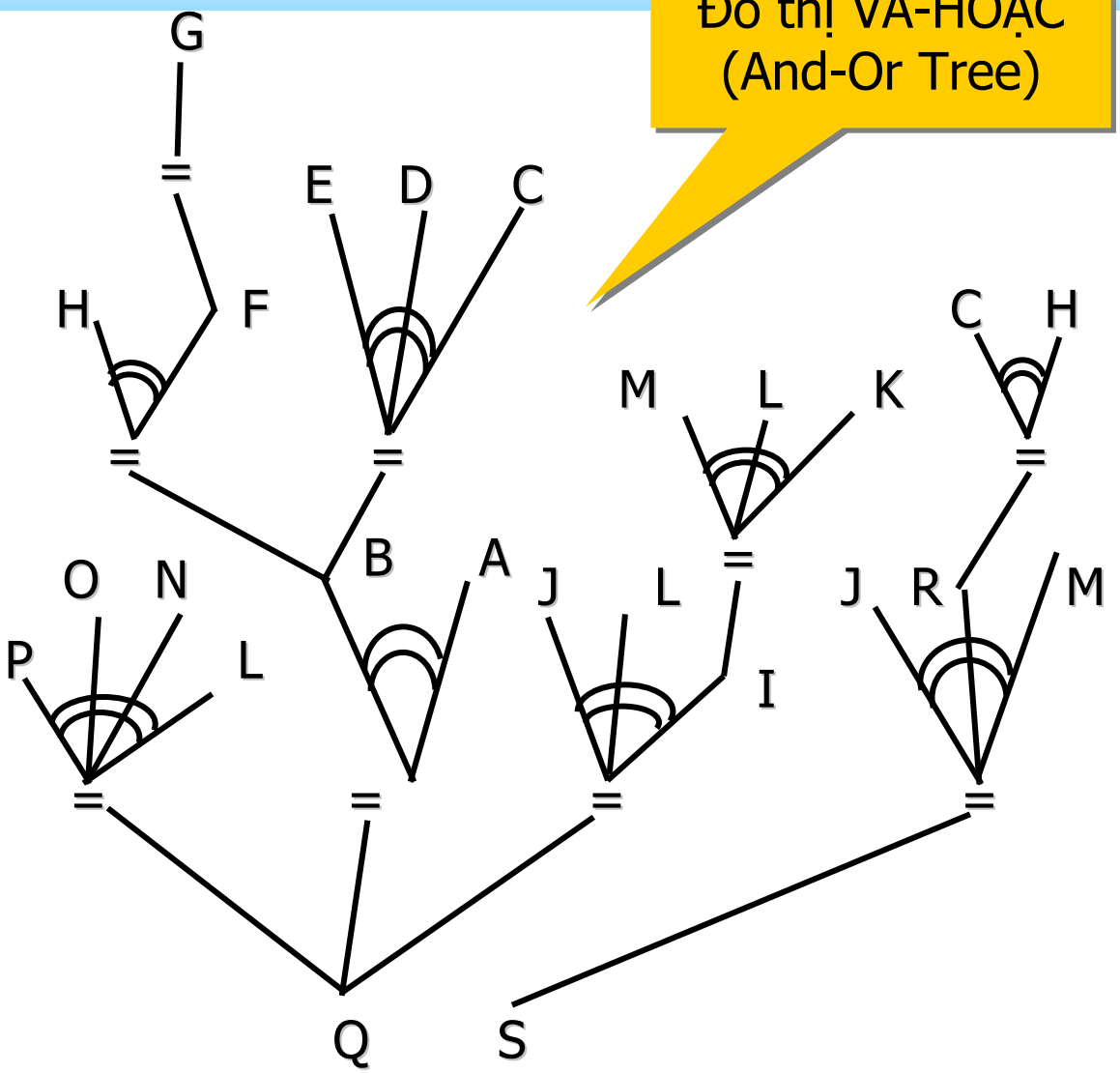



Working space
The engine is getting gas
There is gas in the fuel tank
There is gas in the carburettor
The engine turns over
Rule 1
Rule 2
Rule 3
Rule 4

Example 4.

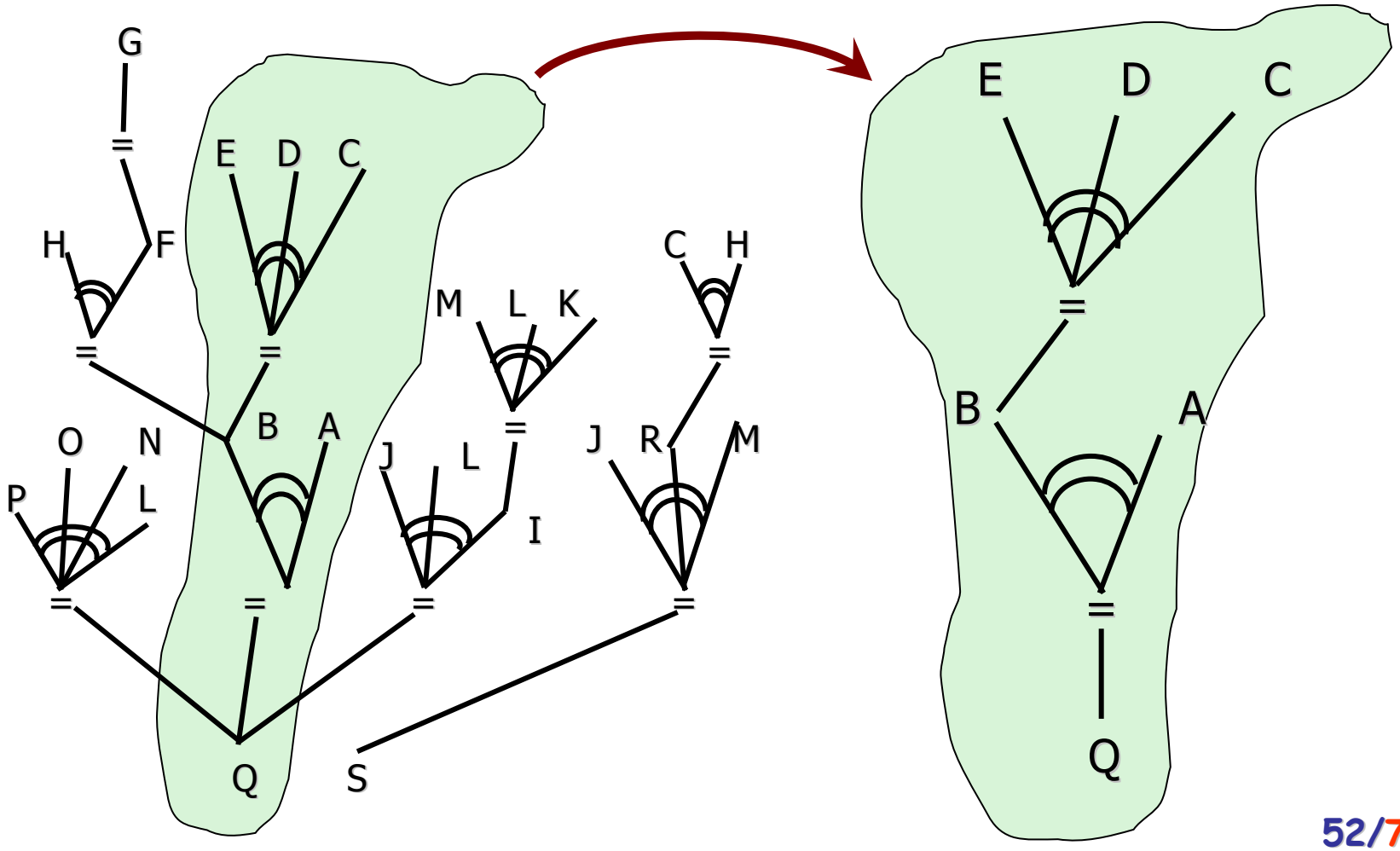
- Rulesbase:
  1. K, L, M → I
  2. I, L, J → Q
  3. C, D, E → B
  4. A, B → Q
  5. L, N, O, P → Q
  6. C, H → R
  7. R, J, M → S
  8. F, H → B
  9. G → F
- Factsbase:

  A, C, D, E, G, H, K



Example 4. Kết quả suy diễn

- Xây dựng đồ thị con VÀ-HOẶC từ đồ thị VÀ-HOẶC trên đây
  - Tìm được Q khi các sự kiện A, C, D và E được thiết lập



Exercise 1.

- Rulesbase:
  1. B, D, E → F
  2. G, D → A
  3. C, F → A
  4. B → X
  5. D → E
  6. X, A → H
  7. C → D
  8. X, C → A
  9. X, B → D
- Factsbase:

  B, C

- Yêu cầu :
  - Vẽ đồ thị và-hoặc
  - Áp dụng thuật toán suy diễn tiến để tìm kết quả H
  - Có nhận xét gì ?

Exercise 2.

⌘ Rules Base:

R1: IF management competence is good
AND External credit rating is fair
AND Bank's credit rating is marginal
THEN Loan is rejected

R2: IF Loan type is seasonal
AND Profitability rating is high
AND Solvency rating is low
THEN Bank's credit rating is marginal

R3: IF Cash/current liabilities > 0.1
AND Tentative solvency rating is low
THEN Solvency rating is low

⌘ Facts Base:

Bank's credit rating	UNKNOWN
Cash/current liabilities	0.18
External credit rating	FAIR
Loan	SEASONAL
Loan type	UNKNOWN
Management competence	UNKNOWN
Profitability rating	HIGH
Solvency rating	UNKNOWN
Tentative solvency rating	LOW

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$American(X) \wedge Weapon(Y) \wedge Sells(X,Y,Z) \wedge Hostile(Z) \rightarrow Criminal(X)$

Nono … has some missiles, i.e., $\exists X\ Owns(Nono,X) \wedge Missile(X)$:

$Owns(Nono,M1)$ and $Missile(M1)$

… all of its missiles were sold to it by Colonel West

$Missile(X) \wedge Owns(Nono,X) \rightarrow Sells(West,X,Nono)$

Missiles are weapons:

$Missile(X) \rightarrow Weapon(X)$

An enemy of America counts as "hostile":

$Enemy(X,America) \rightarrow Hostile(X)$

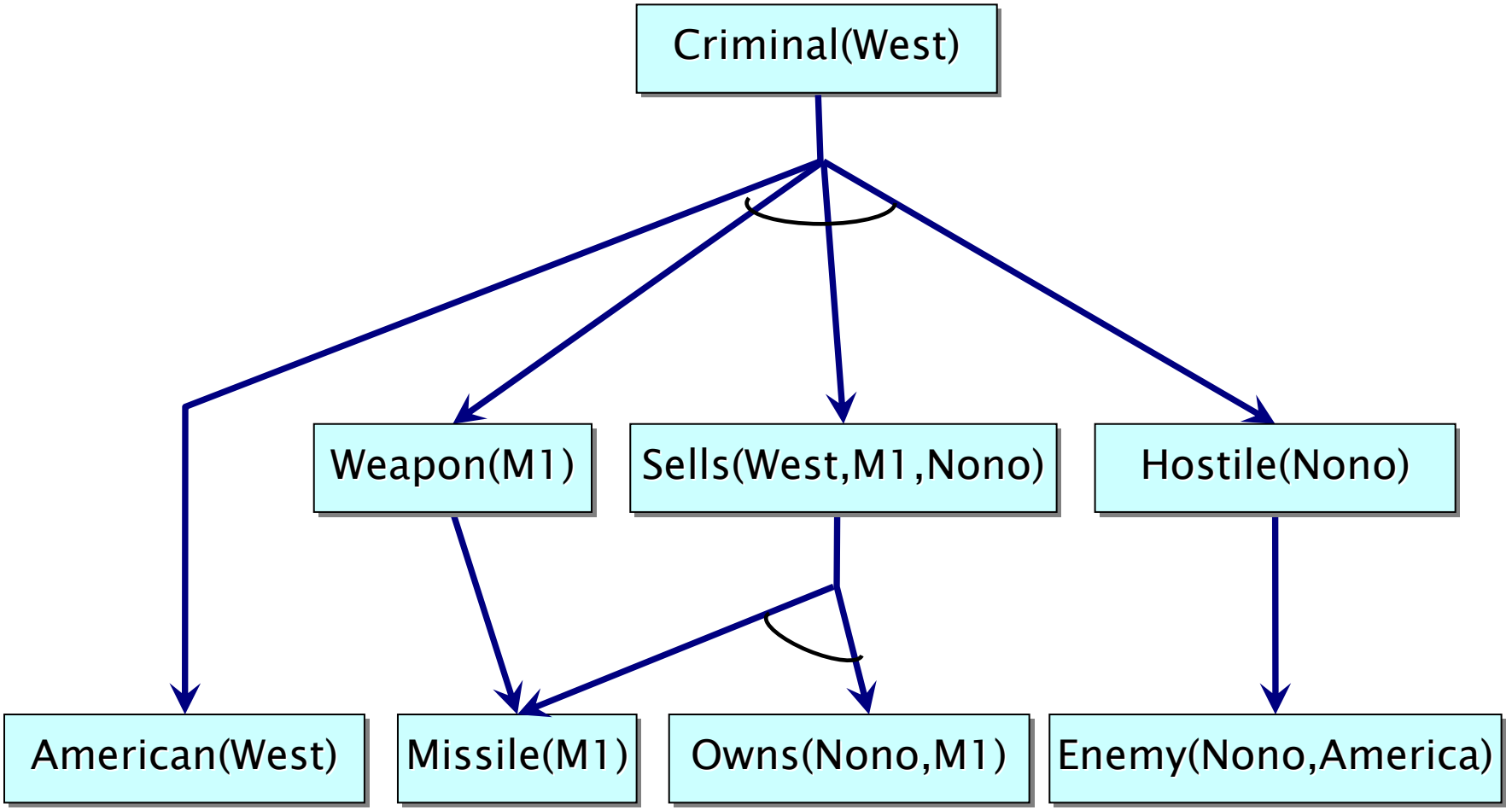
West, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

Forward chaining proof



Phương pháp suy diễn lùi

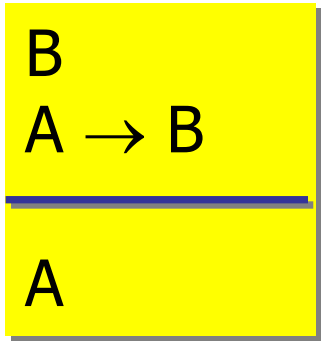
- Phương pháp suy diễn lùi :
  - Tiến hành các lập luận theo chiều ngược lại
    (so với phương pháp suy diễn tiến)
  - Đầu vào có dạng một câu hỏi «Cho biết giá trị thuộc tính đích (goal) A ?»
  - MSD suy diễn để đưa ra tình huống trả lời gồm các sự kiện là cơ sở của giả thuyết đã cho gồm để gán giá trị cho thuộc tính A
- Ví dụ :
  - Nếu ai đó vào nhà mà cầm áo mưa và áo quần bị ướt (hậu quả)
    thì giả thuyết là trời mưa (nguyên nhân)
  - Để củng cố giả thuyết này,
    ta sẽ hỏi người đó xem có phải trời mưa không ?
  - Nếu người đó trả lời có
    thì giả thuyết trời mưa đúng và trở thành một sự kiện
  - Nghĩa là trời mưa nên phải cầm áo mưa và áo quần bị ướt

Ý tưởng thuật toán suy diễn lùi

- Với mỗi thuộc tính đã cho, người ta định nghĩa nguồn của nó :
  - Nếu thuộc tính xuất hiện như là tiền đề của một luật (phần đầu của luật), thì nguồn sẽ thu gọn thành một câu hỏi
  - Nếu thuộc tính xuất hiện như là hậu quả của một luật (RHS), thì nguồn sẽ là các luật mà trong đó, thuộc tính là kết luận
  - Nếu thuộc tính là trung gian, xuất hiện đồng thời như là tiền đề và như là kết luận, khi đó nguồn có thể là các luật, hoặc có thể là các câu hỏi mà chưa được nêu ra
  - Nếu mỗi lần với câu hỏi đã cho, người sử dụng trả lời hợp lệ, giá trị trả lời này sẽ được gán cho thuộc tính và xem như thành công
  - Nếu nguồn là các luật, MSD sẽ tìm giá trị các thuộc tính thuộc tiền đề (LHS) bằng cách xét lần lượt các luật có thuộc tính đích xuất hiện ở kết luận
  - Nếu các luật thoã mãn, thuộc tính kết luận sẽ được ghi nhận

Backward Chaining

- Conclude from "B" and "A implies B" to "A"

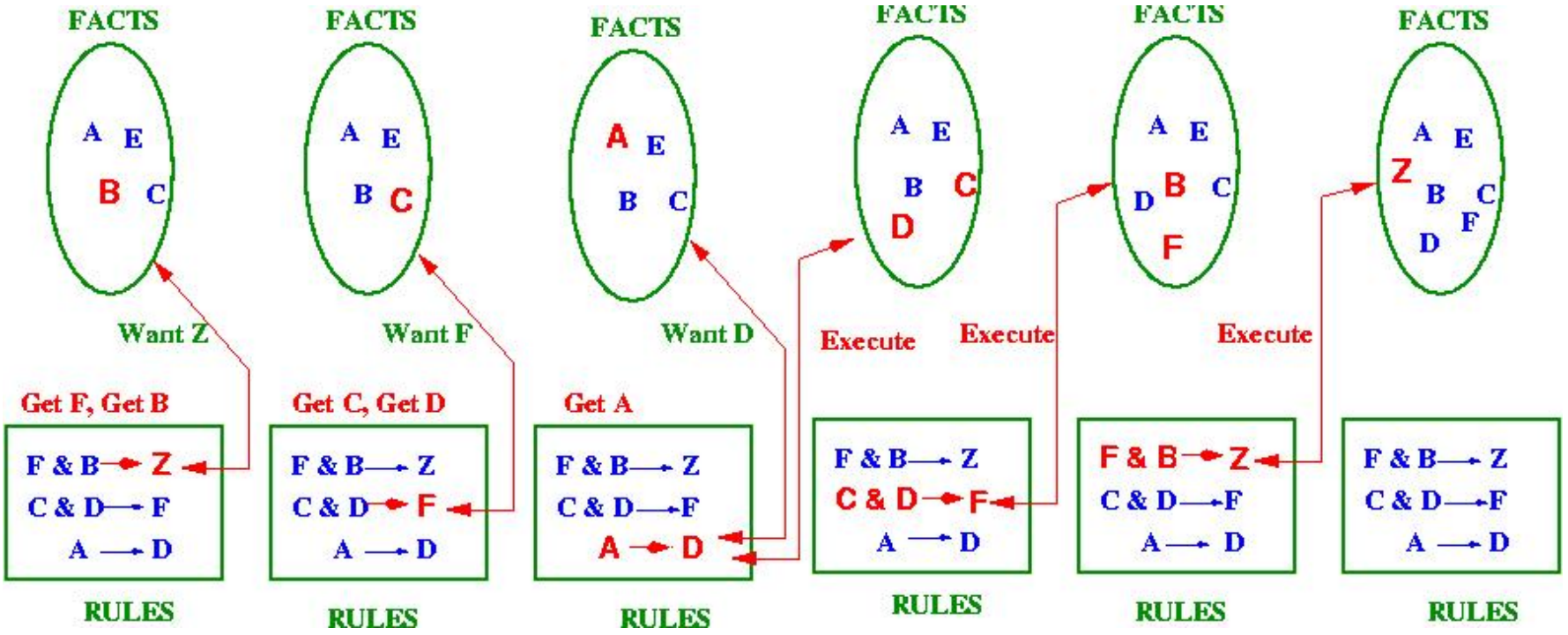


- Example:

The dress is wet
If it is raining, the dress is wet

It is raining

Backward Chaining



Thuật toán suy diễn lùi

⌘ Definitions :

FB, RB, R1 $\subseteq$ RB

Q : Sự kiện đưa vào xử lý

⌘ Algorithme :

Procedure DEDUCE (Q)

If Q $\in$ BF **Then** Write " Success" ;

Else

Push BR, BF ;

IP $\leftarrow$ 1 ;

Call CYCLE(BR, BF) ;

Return;

Thuật toán suy diễn lùi

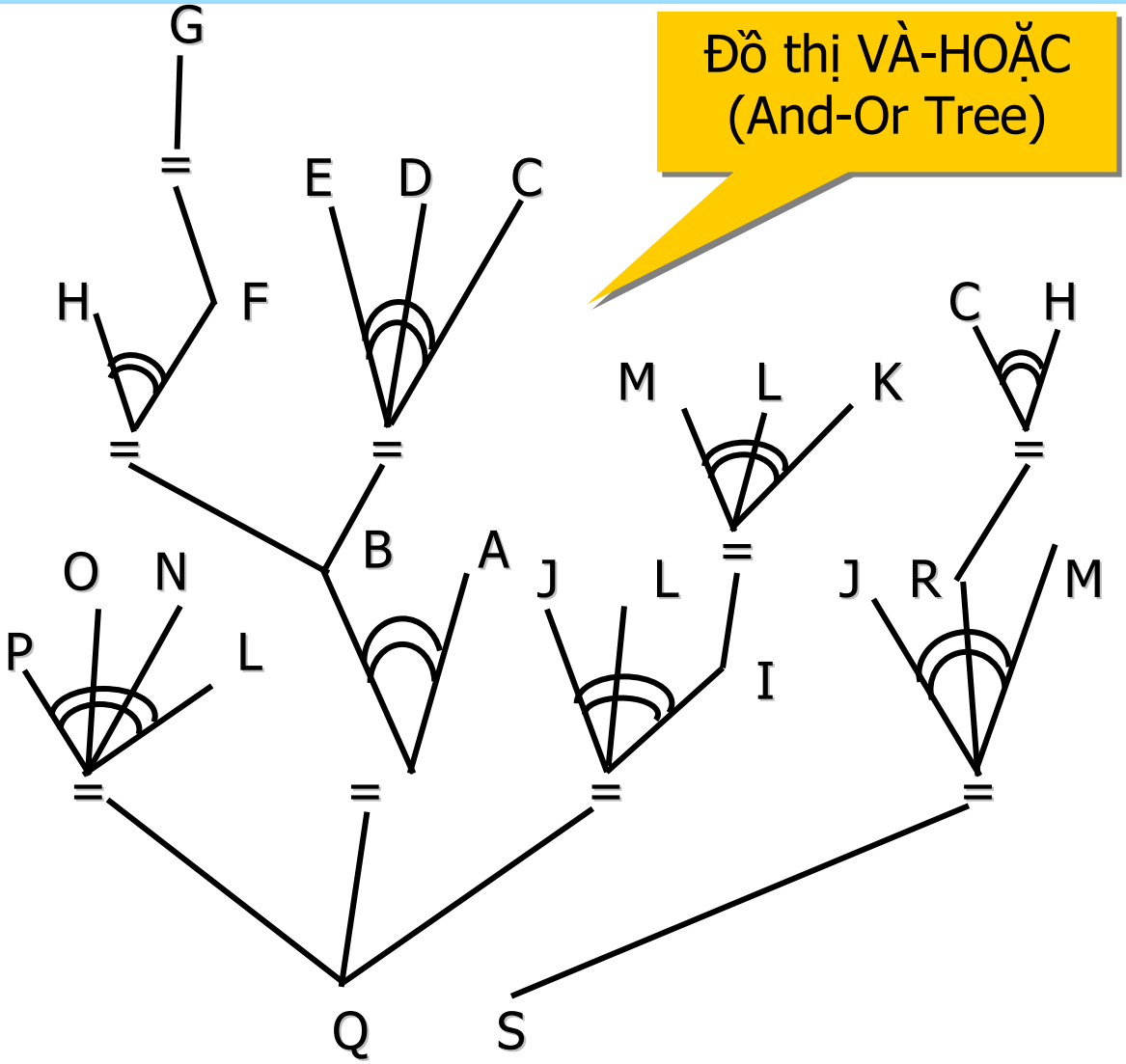
Procedure CYCLE(ER, EF)
Var : ER, EF, r ; ' Các biến làm việc địa phương
If ER = ∅ **Then**
    IP ← IP -1 ; ' Lấy một luật ở đỉnh Stack
    If IP = 0 **Then** Write " Failure" ; **Return**;
    Else ER ← Luật ở đỉnh Stack chưa xét
        EF ← Các sự kiện ở đỉnh Stack;
        Call CYCLE(ER, EF) ; **Return**;
r ← CHOOSE(r, ER) ; ' Chọn một luật
ER ← ER – { r } ; ' Loại bỏ luật đã xét ở ER
If LHS(r) ∈ EF **Then**
    If RHF(r) ∈ Q **Then** Write " Success" ; **Return**;
    Else Push (EF ∪ RHF(r)), BR ; ' (IP ← IP +1)
        Đánh dấu màu đỏ phần tử ngay dưới đỉnh Stack;
        Đánh dấu màu đỏ cho luật r ở đỉnh
        Đánh dấu màu xanh cho luật r tại đỉnh - 1 ;
        ER ← Các luật ở đỉnh chưa đánh dấu. Thông thường là màu đỏ;
        EF ← Các sự kiện ở đỉnh Stack ;
        Call CYCLE(ER, EF) ; **Return**;
Call CYCLE(ER, EF) ; **Return**;

Example 4. Backward Chaining

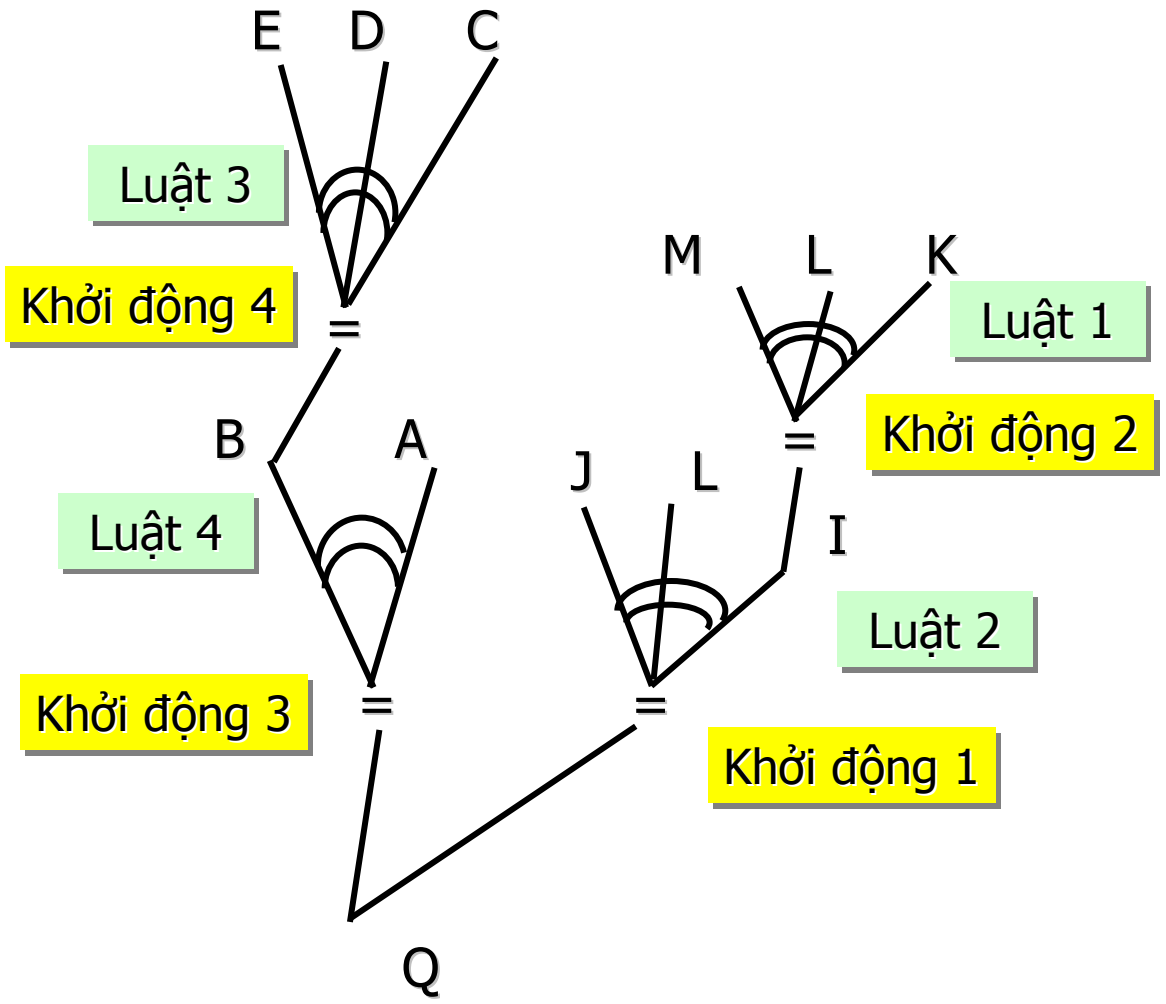
- Rulesbase:
  1. K, L, M → I
  2. I, L, J → Q
  3. C, D, E → B
  4. A, B → Q
  5. L, N, O, P → Q
  6. C, H → R
  7. R, J, M → S
  8. F, H → B
  9. G → F
- Factsbase:

  A, C, D, E, G, H, K
- Question (Goal):

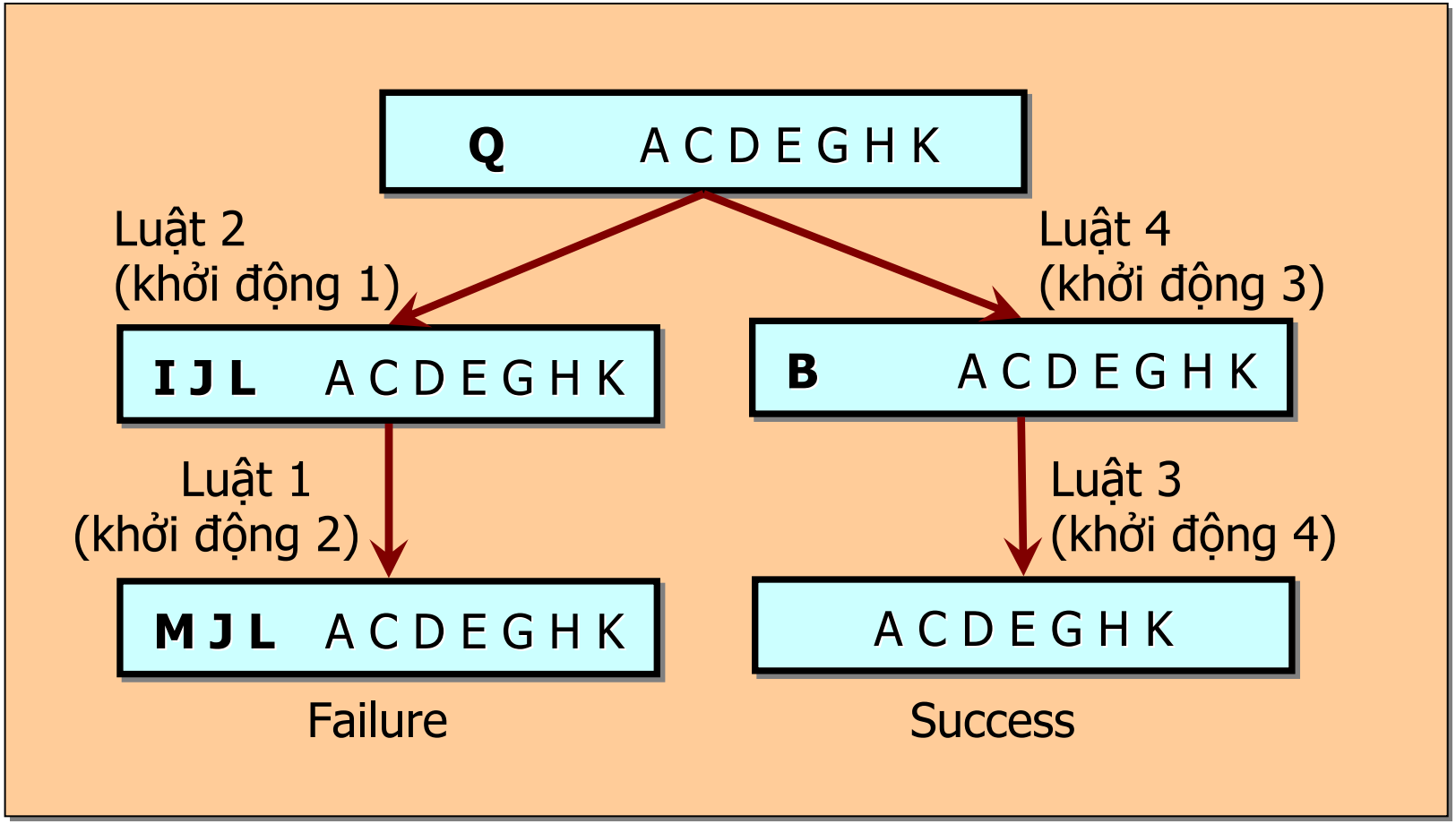
  Q



Đồ thị Và-Hoặc thiết lập Q



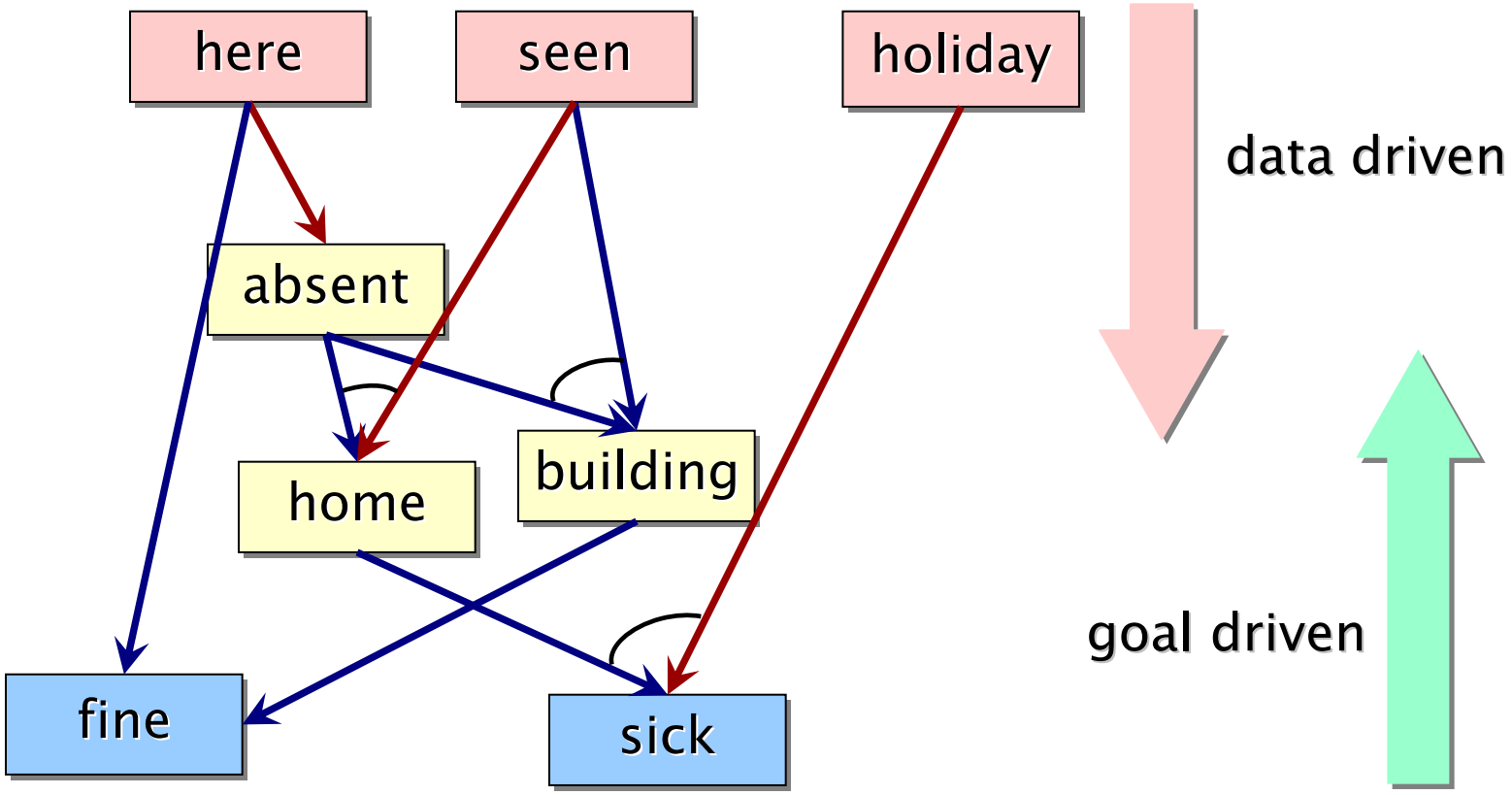
Áp dụng thuật toán suy diễn lùi



Exercise

- Áp dụng thuật toán suy diễn lùi cho ví dụ Colonel West

Data-driven × Goal-driven



Data-driven × Goal-driven

- Goal Driven (backward chaining) ~ blood diagnostic, theorem proving
  - Limited number of goal hypothesis
  - Data shall be acquired, complicated data about the object
  - Less operators to start with at the goal rather than at the data
- Data Driven (forward chaining) ~ configuration, interpretation,
  - reasonable set of input data
  - data are given at the initial state
  - huge set of possible hypothesis
- Taxonomy of rules, meta-rules, priorities, ...

Forward or Backward Reasoning?

- **Four major factors**
- More possible start states or goal states?
  Move from smaller set of states to the larger
  Assume you are at home and you need some bread. You've one initial state and many possible goal states where to get bread from (Sainsbury's, Tesco, Aldi, Morrison, ASDA, CornerShop, Bakery). That is, in such a situation it is probably a good approach to search in forward chaining fashion, since you have many chances to hit a goal. If you choose backward chaining you would have to commit to one of the goals, e.g. the Bakery and then search backwardly how the get there from home.
  If, however, there are 5 people Alice, Bob, Claire, Dora, and Edgar at 5 different places A, B, C, D, and E, and one of them should get the Tesco home brand of ketchup, then it would be better to start backward chaining from the Tesco store and stop search when one of the places A, B, C, D, or E is reached

Forward or Backward Reasoning? (Cont'd)

- Has program to justify reasoning?
  - Prefer direction that corresponds more closely to the way users think
- What kind of events triggers problem-solving?
  - If it is **arrival of a new fact**, forward chaining makes sense
  - If it is **a query to which a response is required**, backward chaining is more natural.
- In which direction is branching factor greatest?
  - Go in direction with lower branching factor

Hệ chuyên gia
(Expert System)

PGS.TS. Phan Huy Khánh

khanhph@vnn.vn

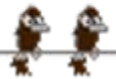
Chương 4
Phát triển các hệ chuyên gia

Phát triển các hệ chuyên gia

⌘ Tiến trình phát triển một dự án HCG

⌘ Tiếp nhận tri thức HCG

⌘ Bộ sinh của HCG



Tại sao cần xây dựng một HCG ?

- Câu hỏi này thường xuyên được đặt ra cho bất kỳ dự án Tin học nào
- Có thể trả lời ngay là do những đặc trưng và ưu điểm của các HCG
- Trước khi bắt đầu, cần xác định rõ đâu là bài toán, ai là chuyên gia, và ai là người sử dụng
- Tùy theo yêu cầu NSD mà có nhiều cách nhìn nhận khác nhau về một hệ chuyên gia :

Loại người sử dụng	*Vấn đề đặt ra*
Người quản trị	Tôi có thể dùng HCG để làm gì ?
Kỹ thuật viên	Làm cách nào để tôi vận hành HCG tốt nhất ?
Nhà nghiên cứu	Làm sao để tôi có thể mở rộng HCG ?
NSD cuối	Khai thác HCG sẽ giúp tôi cái gì đây ? Sử dụng HCG có rắc rối và tốn kém không ? Một phần mềm như HCG có đáng tin cậy không ?

Lựa chọn bài toán cho HCG

- Để xây dựng một HCG, trước tiên cần *lựa chọn một bài toán thích hợp* (selecting the appropriate problem)
- Xây dựng một HCG tương tự triển khai một dự án PM :
  - Được thực hiện bởi một tập thể
  - Nhằm đạt được kết quả mong muốn
- Cần phải có bốn yếu tố cơ bản :
  - Nguồn kinh phí
  - Nguồn nhân lực
  - Nguồn tài nguyên
  - Khoảng thời gian dự kiến
- Những yếu tố này ảnh hưởng đến giá thành của một HCG

Giá thành xây dựng một HCG

- Chi phí (pay-off), xây dựng một HCG phụ thuộc :
  - Nguồn nhân lực, tài nguyên và thời gian hoàn thiện
  - Chi phí huấn luyện, đào tạo (Training)
- Ví dụ ở Mỹ, chi phí để đào tạo sử dụng thành thạo một HCG có thể lên tới 2.500USD/tuần lễ/người
- Những yếu tố ảnh hưởng đến giá thành
  - Người sử dụng trả tiền tuỳ theo tính hiệu quả hay ưu điểm của HCG sử dụng
  - Nếu không có ai sử dụng HCG, thì sẽ không có ai trả tiền để bù lại chi phí và có lãi
  - HCG đòi hỏi sử dụng các công nghệ mới, có nhiều rủi ro hơn so với thực hiện các dự án Tin học

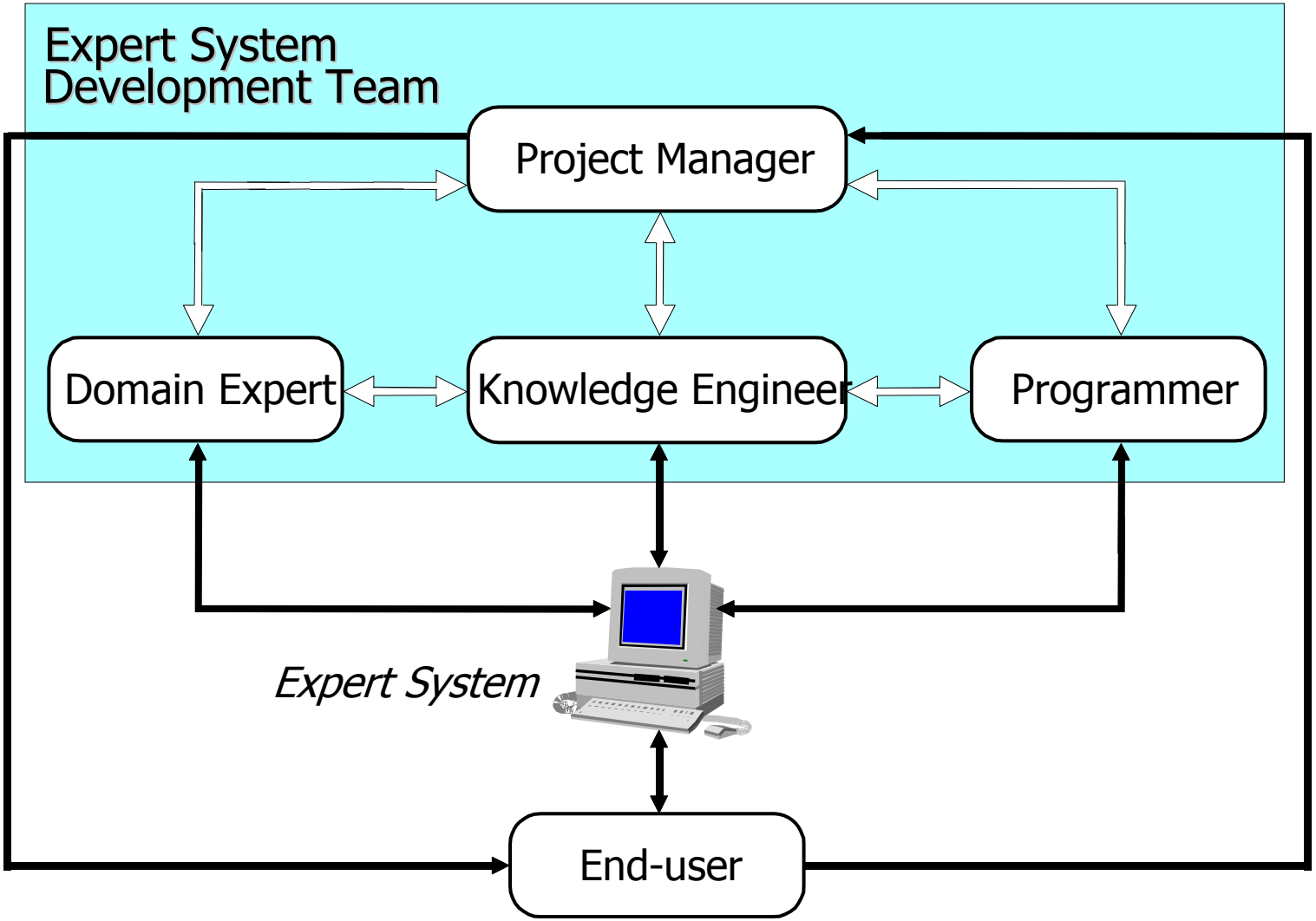
Công cụ xây dựng HCG

- Sau bước lựa chọn, phát biểu và đặc tả bài toán là các bước phát triển HCG
- Sử dụng những công cụ (tools) nào để xây dựng một HCG ?
- Hiện nay có rất nhiều công cụ để xây dựng HCG :
  - Mỗi công cụ đều có ưu điểm và nhược điểm nhất định
  - Những công cụ phổ biến là CLIPS và OPS5, ngoài ra có ART, ART-IM, Eclipse, Cognate...
  - Người ta cũng sử dụng một số ngôn ngữ lập trình thuộc lĩnh vực Trí tuệ nhân tạo (Artificial Intelligence), tính toán hình thức (Formal Calculus)... để phát triển một HCG :
    - Prolog, họ Lisp...

Tiến trình phát triển dự án HCG

- Tiến trình phát triển một dự án HCG phụ thuộc vào :
  - Nguồn nhân lực (development team)
  - Nguồn tài nguyên do chủ đầu tư cung cấp
  - Phương pháp tổ chức quản lý quá trình phát triển dự án, hay Quản lý dự án (Project Management)
  - Phương pháp Quản lý cấu hình sản phẩm (Product Configuration Management)
  - Phương pháp Quản lý tài nguyên (Resource Management)

The main players in the development team



The domain expert

- The *domain expert* is:
  - a knowledgeable and skilled person capable of solving problems in a specific area or *domain*
  - This person has the greatest expertise in a given domain
  - This expertise is to be captured in the expert system
- Therefore, the expert :
  - must be able to communicate his or her knowledge
  - must be willing to participate in the expert system development
  - and commit a substantial amount of time to the project
- The domain expert is the most important player in the expert system development team

The knowledge engineer

- The *knowledge engineer* is:
  - someone who is capable of designing, building and testing an expert system
  - He or she interviews the domain expert to find out how a particular problem is solved
- The knowledge engineer establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system
- The knowledge engineer then chooses some development software or an expert system shell, or looks at programming languages for encoding the knowledge
- And finally, the knowledge engineer is responsible for testing, revising and integrating the expert system into the workplace

The programmer

- The *programmer* is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand
- The programmer needs:
  - to have skills in symbolic programming in such AI languages as LISP, Prolog and OPS5
  - and also have some experience in the application of different types of expert system shells
- In addition, the programmer should know conventional programming languages like C, Pascal, FORTRAN and Basic

The project manager

- The *project manager* is the leader of the expert system development team, responsible for keeping the project on track. He or she makes sure that all deliverables and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user

The end-user

- The *end-user*, often called just the *user*, is a person who uses the expert system when it is developed
- The user must not only be confident in the expert system performance but also feel comfortable using it
- Therefore, the design of the user interface of the expert system is also vital for the project's success; the end-user's contribution here can be crucial

Quản lý dự án HCG

- Quản lý dự án HCG gồm các công đoạn như sau :
  - Quản lý hoạt động (Activity Management), gồm :
    - Lập kế hoạch (Planning)
    - Lập biểu công việc (Scheduling)
    - Phân bổ thời gian (Chronicling)
    - Phân tích (Analysis)
  - Quản lý cấu hình sản phẩm (Product Configuration Management) :
    - Quản lý sản phẩm (Product Management)
    - Quản lý thay đổi (Change Management)
  - Quản lý tài nguyên (Resource Management)

Quản lý hoạt động : Lập kế hoạch

- Quản lý lập kế hoạch :
  - Định nghĩa các hoạt động (define activities)
  - Xác định hoạt động ưu tiên (specify priority of activities)
  - Xác định nhu cầu tài nguyên (resource requirement)
  - Ghi nhớ các sự kiện (milestones)
  - Xác định thời gian (duration)
  - Phân công trách nhiệm (responsabilities)

Quản lý hoạt động : Lập thời biểu

- Lập biểu công việc :
  - Ấn định điểm bắt đầu và điểm kết thúc dự án
  - Giải quyết xung đột khi gặp các việc cùng mức ưu tiên
  - Phân bổ thời gian :
    - Kiểm tra thực hiện dự án (monitor project performance)
  - Phân tích :
    - Phân tích các hoạt động về lập kế hoạch
    - Lập biểu công việc và phân bổ thời gian hoạt động

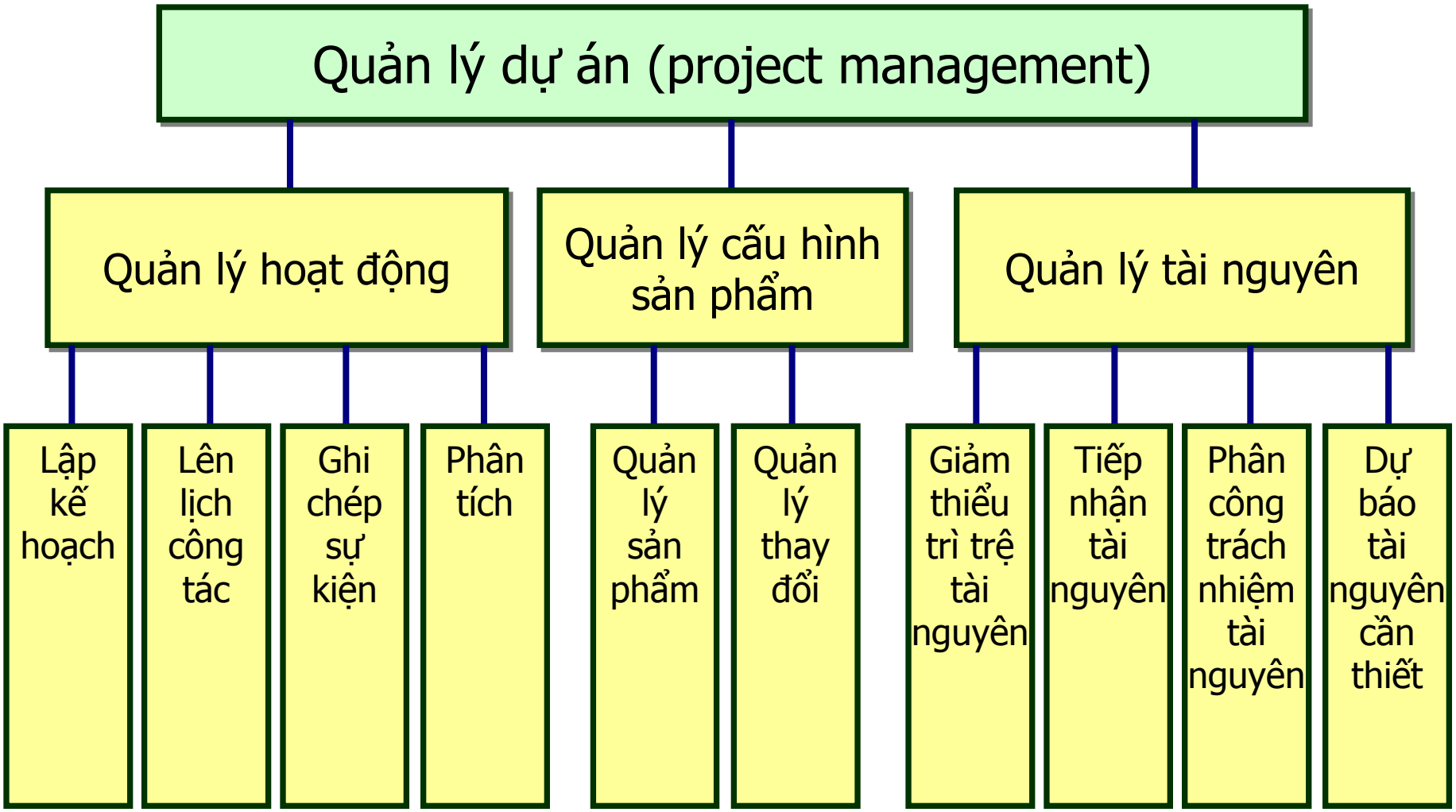
Quản lý cấu hình sản phẩm

- Quản lý sản phẩm :
  - Quản lý các phiên bản khác nhau của các sản phẩm (product management)
  - Quản lý thay đổi :
    - Quản lý các giải pháp sửa đổi sản phẩm và ước lượng ảnh hưởng của thay đổi sản phẩm
    - Phân công người sửa đổi hệ thống
    - Cài đặt phiên bản mới

Quản lý tài nguyên

- Dự báo nhu cầu tài nguyên (forecast needs for resource)
- Thu nhận tài nguyên (acquire resources)
- Phân công trách nhiệm để sử dụng tối ưu nguồn tài nguyên (assign responsabilities for optimium use of resources)
- Phân bổ tài nguyên để giảm thiểu tắc nghẽn (provide critical resources to minimize bottle-necks)

Quá trình quản lý dự án phát triển một HCG



Thuật toán tổng quát để thiết kế một HCG

Begin
- Chọn bài toán thích hợp
- Phát biểu và đặc tả bài toán
- **If** HCG giải quyết thoả mãn bài toán và có thể sử dụng **Then**
  - **While** Bản mẫu chưa được phát triển hoàn thiện **Do**
  - **Begin**
    - Thiết kế bản mẫu
    - Biểu diễn tri thức
    - Tiếp nhận tri thức
    - Phát triển hoàn thiện bản mẫu
  - **End**
  - Hợp thức hoá bản mẫu
  - Triển khai cài đặt
  - Hướng dẫn sử dụng
  - Vận hành
  - Bảo trì và phát triển
- **Else**
  - Tìm các tiếp cận khác thích hợp hơn
- **EnIf**
- Kết thúc

End

Tiếp nhận tri thức

- Các bước tiếp nhận tri thức cho một hệ HCG như sau :
  - Đối thoại trực tiếp với chuyên gia (con người) để thu nhận tri thức
  - Chọn cách biểu diễn tri thức một cách tường minh trong cơ sở tri thức
  - Các chuyên gia đánh giá HCG, trao đổi qua lại về nội dung chất lượng của tri thức, cho đến khi HCG hoàn toàn thỏa mãn yêu cầu
- Sự phát triển một hệ HCG cũng tác động nhiều trong một hệ thống chất lượng thương mại
- Người ta luôn mong muốn nhận được những thành công một khi HCG được phân phối đến người dùng

Tiếp nhận tri thức trong một HCG

Tri thức chuyên gia
(Human Expert)

Đối thoại (dialog)

Công nghệ tri thức
(Knowledge Engineer)

Tri thức tường minh
(explicit knowledge)

Cơ sở tri thức hệ chuyên gia
(Knowledge Base of Expert System)

Phân phối sản phẩm HCG

- Hệ thống được phân phối (Delivery) như thế nào ?
  - Vấn đề phân phối một sản phẩm phụ thuộc chủ yếu vào số lượng các HCG sẽ được phát triển và hệ thống dịch vụ
  - Xu thế thị trường đòi hỏi phát triển các HCG chạy trên nền các thiết bị phần cứng chuẩn
  - Mặt khác, một số HCG đòi hỏi cần có phần mềm hệ thống và các công cụ midleware theo yêu cầu, chẳng hạn bộ xử lý LISP, từ đó có thể làm tăng giá thành sản phẩm
- Nói chung, một HCG cần phải được tích hợp (integrated) với những chương trình đã có sẵn với ưu điểm :
  - Dùng lời gọi thủ tục từ một ngôn ngữ lập trình thông thường
  - Tận dụng môi trường hệ thống để trợ giúp khai thác HCG

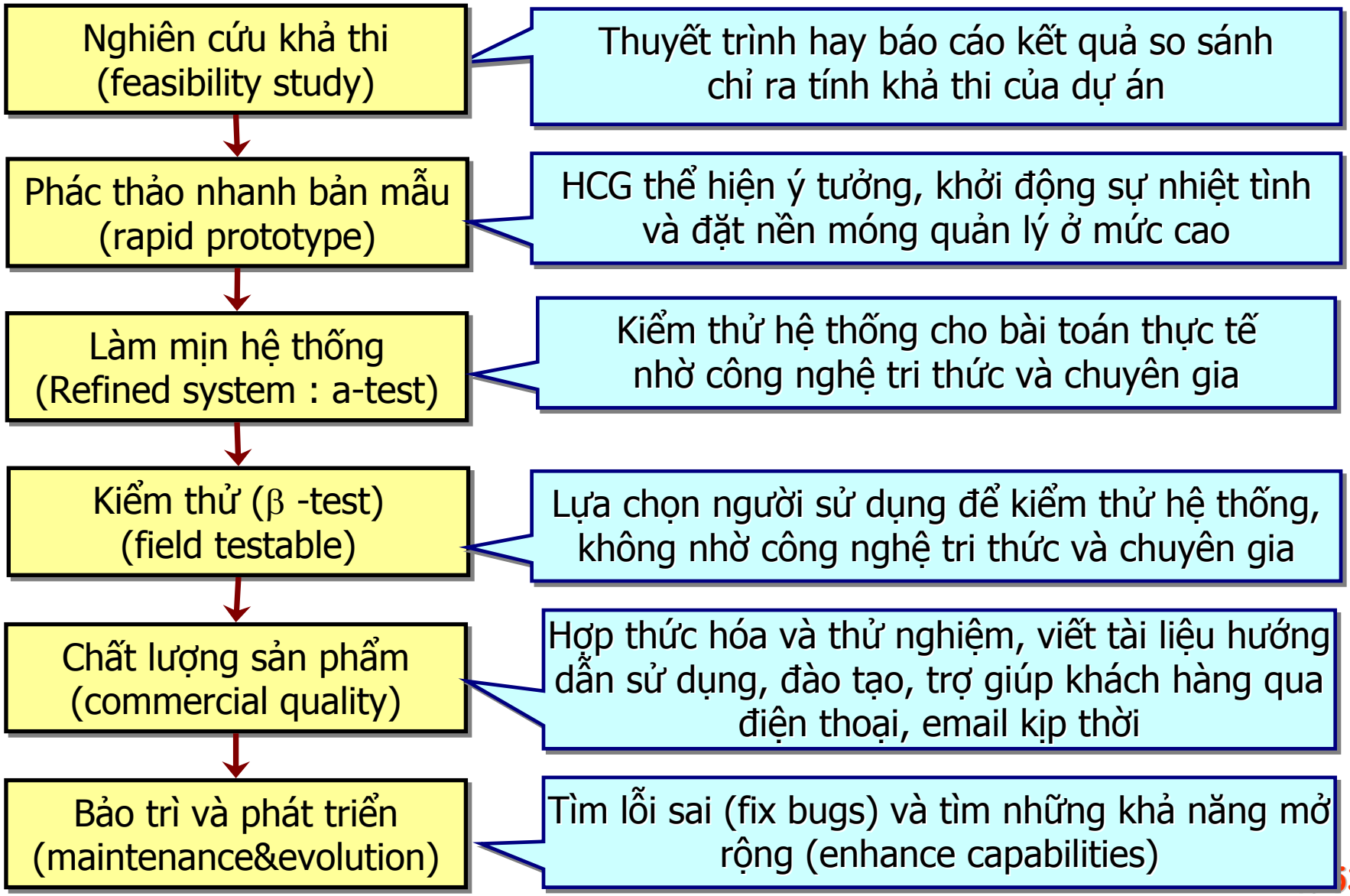
Bảo trì và phát triển HCG

- Các HCG đòi hỏi các hoạt động bảo trì (Maintenance) và tiến triển (Evolve) không hạn chế (Open-Ended) so với các chương trình thông thường
- Lý do :
  - HCG không dựa trên các thuật toán, mà thành tích (Performance) của chúng phụ thuộc vào tri thức có được
  - HCG đòi hỏi phải thường xuyên cập nhật tri thức (System Improves) :
    - Bổ sung mới
    - Thay đổi tri thức cũ, lạc hậu...

Quản lý khai thác HCG

- Một sản phẩm có chất lượng thương mại (commercial quality product) như các HCG, đòi hỏi phải có quy trình khai thác hiệu quả và chặt chẽ
  - Thu thập một cách có hệ thống và có định kỳ các báo cáo sai sót hệ thống do NSD phát hiện
  - Việc bảo trì chỉ được thực hiện tốt khi thu thập đầy đủ các báo cáo sai sót
- Lập kế hoạch phòng ngừa rủi ro khai thác :
  - Mô tả: Xác định vấn đề (rủi ro)
  - Giả thiết: Hoàn cảnh có thể làm xuất hiện rủi ro
  - Xác suất: Ước lượng khả năng xuất hiện (%)
  - Đánh giá ảnh hưởng đối với kết quả khai thác
  - Cách giải quyết (đối sách)

Các giai đoạn cơ bản để phát triển một HCG



Sai sót trong quá trình phát triển HCG

- Quá trình phát triển một SPPM nói chung, một HCG nói riêng, thường gặp phải rủi ro, sai sót
- Người ta phân ra thành nhiều loại sai sót chủ yếu :
  - Sai sót trong tri thức chuyên gia
  - Sai sót ngữ nghĩa
  - Sai sót cú pháp
  - Sai sót do máy suy diễn

Sai sót trong tri thức chuyên gia

- Chuyên gia là nguồn tri thức của một HCG
- Nếu tri thức chuyên gia không đúng và không đầy đủ, hậu quả sai sót sẽ ảnh hưởng suốt quá trình phát triển hệ thống và khai thác sau này
- Ví dụ trường hợp NASA, Hoa Kỳ :
  - Để hạn chế những sai sót có thể trong các chuyến bay vũ trụ, NASA đã sử dụng *bảng kỹ thuật bay* (Flight Technique Panels)
  - Các bảng này do nhiều thành phần chuyên gia tham gia kiểm soát nhằm bảo đảm tính đầy đủ và bao trùm hết mọi lĩnh vực :
    - Những NSD hệ thống
    - Các chuyên gia lĩnh vực độc lập
    - Những người phát triển hệ thống
    - Những người quản trị

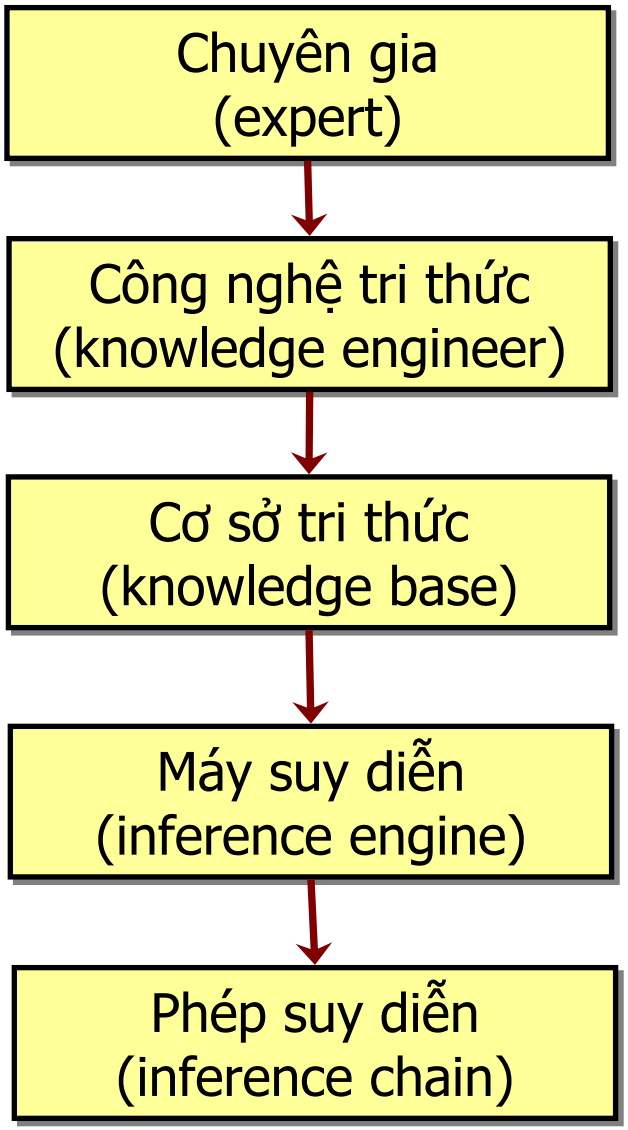
Sai sót ngữ nghĩa

- Sai sót ngữ nghĩa xảy ra do hiểu sai tri thức đưa vào HCG
- Ví dụ, giả sử một chuyên gia nói :
  - « You can extinguish a fire with water »
- Công nghệ tri thức lại hiểu câu này là :
  - « All fires can be extinguished by water »

Các sai sót khác

- Sai sót cú pháp
  - Do biểu diễn sai dạng các luật và các sự kiện
  - hoặc do sai sót ngữ nghĩa
  - hoặc sai sót trong tri thức chuyên gia ở các bước trước
- Sai sót do máy suy diễn
  - Máy suy diễn là một chương trình nên có thể gặp lỗi khi thực hiện và có thể xác định được nguyên nhân
  - Tuy nhiên, việc xác định lỗi trong một số HCG vẫn gặp khó khăn, do phụ thuộc vào công cụ phần mềm sử dụng

Sai sót và nguyên nhân sai sót

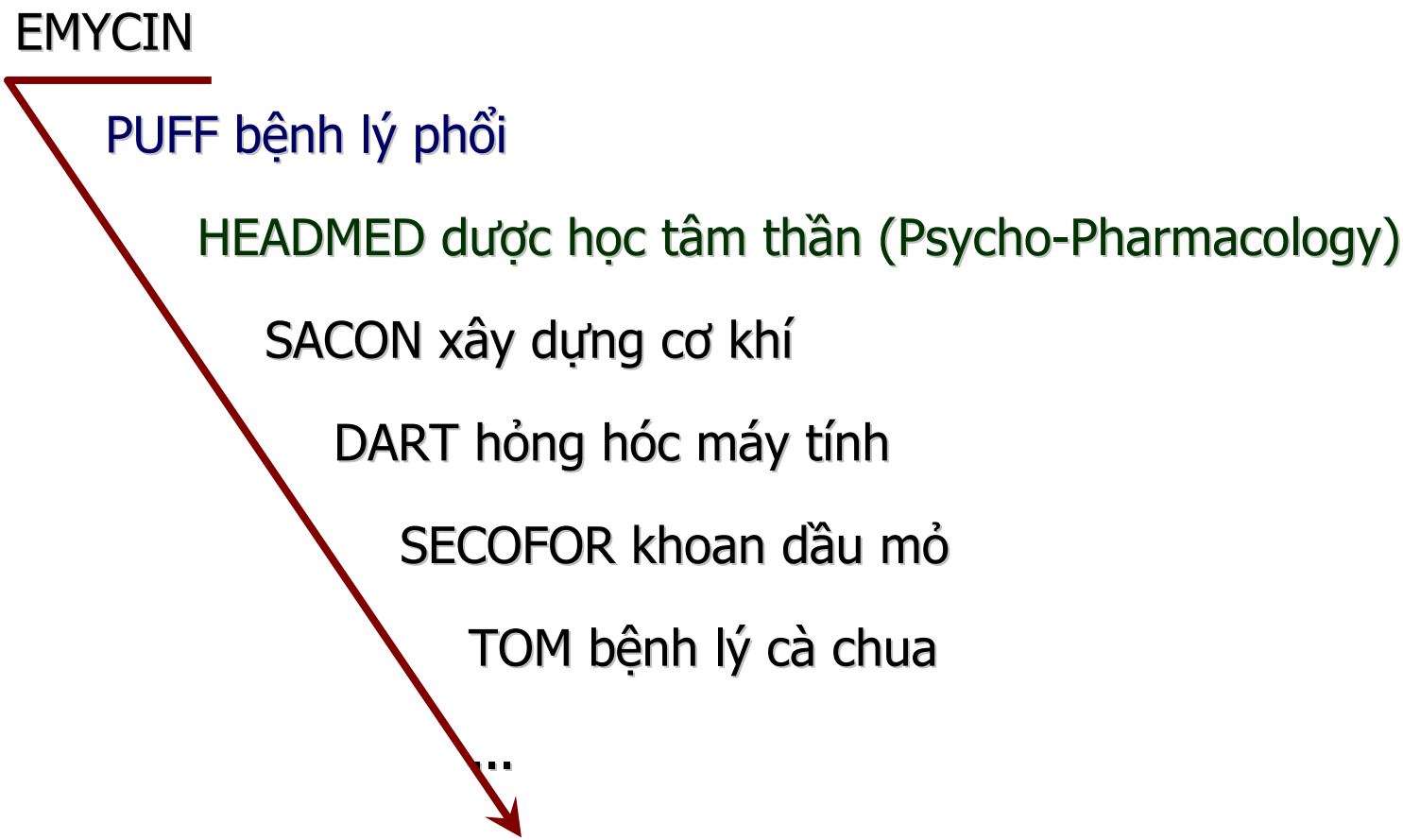


- Sai sót trong tri thức của chuyên gia : thiếu sót nhầm lẫn...
- Sai sót về mặt ngữ nghĩa giữa công nghệ tri thức và chuyên gia
- Suy luận (elicitation) không đầy đủ về tri thức từ chuyên gia
- Sai sót về mặt cú pháp
- Sai sót do thiếu sót và nhầm lẫn tri thức trong các luật và các sự kiện, tính không chắc chắn
- Lỗi của máy suy diễn, lỗi phần mềm công cụ HCG
- Lỗi suy diễn do xác định sai độ ưu tiên của các luật, tương tác giữa các luật, sai trong cơ sở tri thức, suy luận không nhất quán...

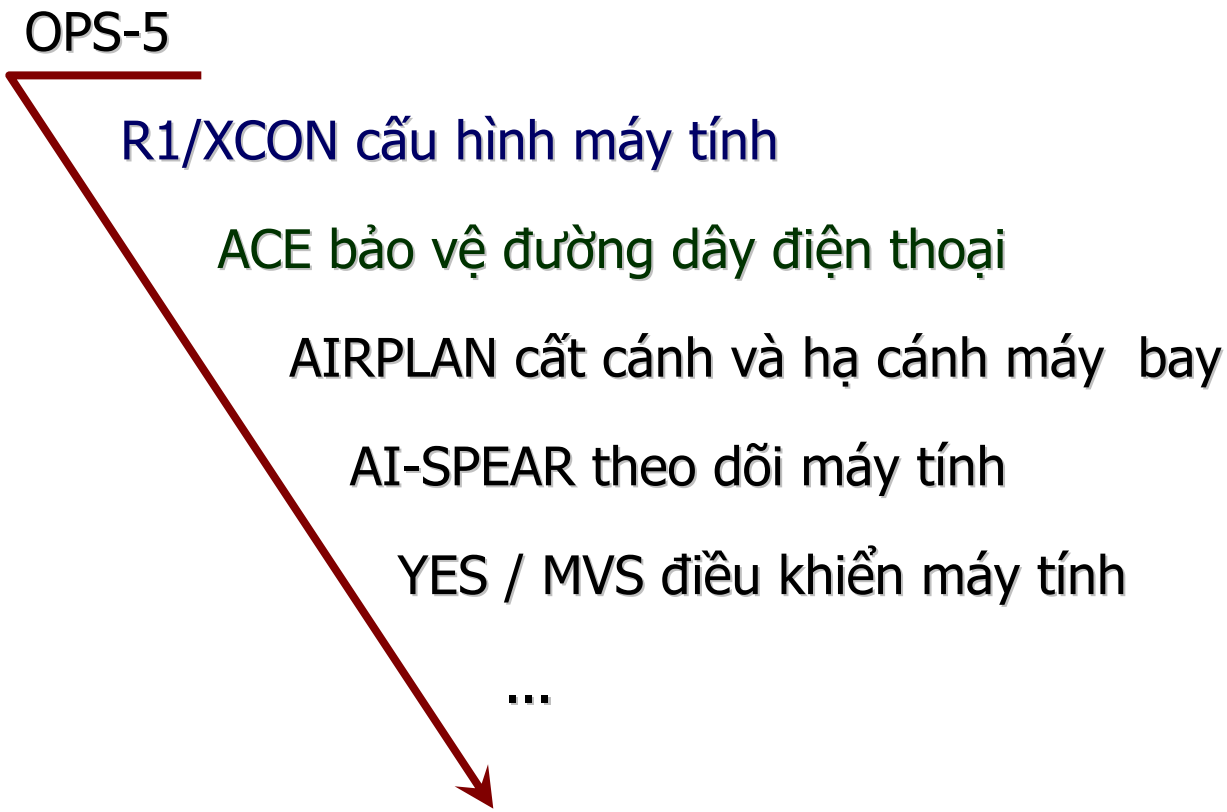
Bộ sinh của HCG

- Bộ sinh của HCG (Expert System Generator) bao gồm :
  - Một *máy suy diễn*,
  - Một *ngôn ngữ* thể hiện tri thức (bên ngoài)
  - Một *tập hợp các cấu trúc và các quy ước* thể hiện các tri thức (bên trong)
- Theo cách nào đó, các cấu trúc và các quy ước này xác định một cơ sở tri thức rỗng (hay rỗng bộ phận)
- Nhờ các tri thức chuyên môn để định nghĩa một HCG, người ta đã tạo ra bộ sinh để làm đầy cơ sở tri thức
  - Chẳng hạn, EMYCIN là tên của bộ sinh của HCG MYCIN và được tiếp tục áp dụng cho một số lĩnh vực
  - HCG R1 được xây dựng từ bộ sinh OPS (là hệ thống luật được phát triển bởi Charles Forgy năm 1975 tại Carnegie-Mellon University)

Một số hậu duệ của EMYCIN



Một số hậu duệ của OPS-5



Thống kê luật của cácHCG

- Nhờ bộ sinh, mỗi hệ HCG có thể chứa từ hàng trăm đến hàng ngàn luật
- Bảng dưới đây thống kê số luật của một số HCG :

HCG	*Lĩnh vực*	*Năm xuất hiện*	*Số luật*
MYCIN	Y học	1974	500
PROSPECTOR	Địa chất	1979	1 600
R1/XCON	Tin học	1980	> 7 000
LITHO	Địa chất	1982	500
SPHINX	Y học	1984	400
TOM	Nông học	1984	200

Khả năng «học» (learn)

- Một trong những nét hấp dẫn của tiếp cận HCG là khả năng «học» (learn) của hệ thống nhằm thường xuyên sửa đổi và hoàn thiện cơ sở tri thức vốn có
- Sơ đồ dưới đây cho biết sự tiến triển của hai HCG nổi tiếng của Mỹ là MYCIN và R1 :
  - MYCIN 1974 : 200 luật hiện nay : 500 luật
  - R1 1980 : 800
    1981 : 1 000
    1982 : 1 500
    1983 : 2 000
    1984 : > 3 000
    1985 : > 7 000

Soạn thảo các luật

- Người ta sử dụng một số quy ước soạn thảo luật như sau :
  - Mỗi luật do chuyên gia cung cấp :
    - Phải định nghĩa được các *điều kiện khởi động* (tác nhân) hay *tiền đề* của luật, nghĩa là các tình huống (được xác định bởi các quan hệ trên tập hợp dữ liệu đã cho) và *hậu quả* của luật, để luật này có thể áp dụng
    - Theo cách dùng thông thường, người ta đặt tên riêng cho luật để chọn áp dụng, hoặc cung cấp một nhóm các *sự kiện* (fact) tương thích với điều kiện khởi động của luật
  - Trong luật, không bao giờ người ta chỉ định một luật khác bởi tên riêng
- Ví dụ : luật R sau đây tuân thủ hai đặc trưng :

IF	bệnh nhân sốt
AND	tốc độ lắng huyết cầu trong máu tăng lên
THEN	bệnh nhân nhiễm bệnh virut

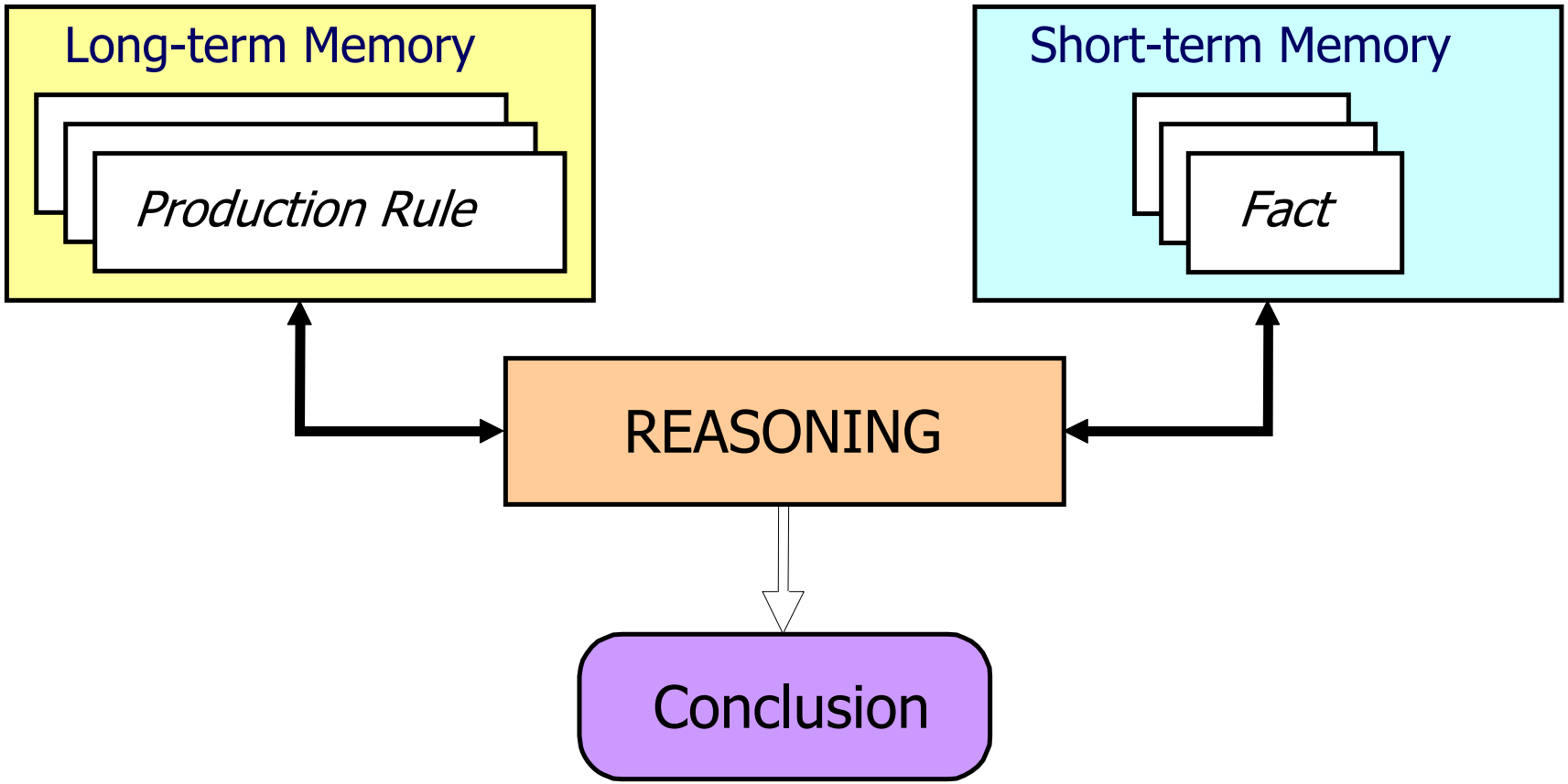
Nhận xét

- Phương pháp biểu diễn tri thức sử dụng luất có nhiều ưu điểm và người ta thường sử dụng phương pháp này để phát triển các HCG
  - Cách biểu diễn các điều kiện khởi động trong luật phù hợp với cách tư duy tự nhiên của các chuyên gia
  - Người ta dễ dàng thể hiện cũng như sửa đổi các tri thức tiếp nhận
  - Người ta không nhất thiết phải đặt tên cho luật để có thể gọi đến khi cần, mà có thể khai thác thông tin từ các điều kiện khởi động của luật

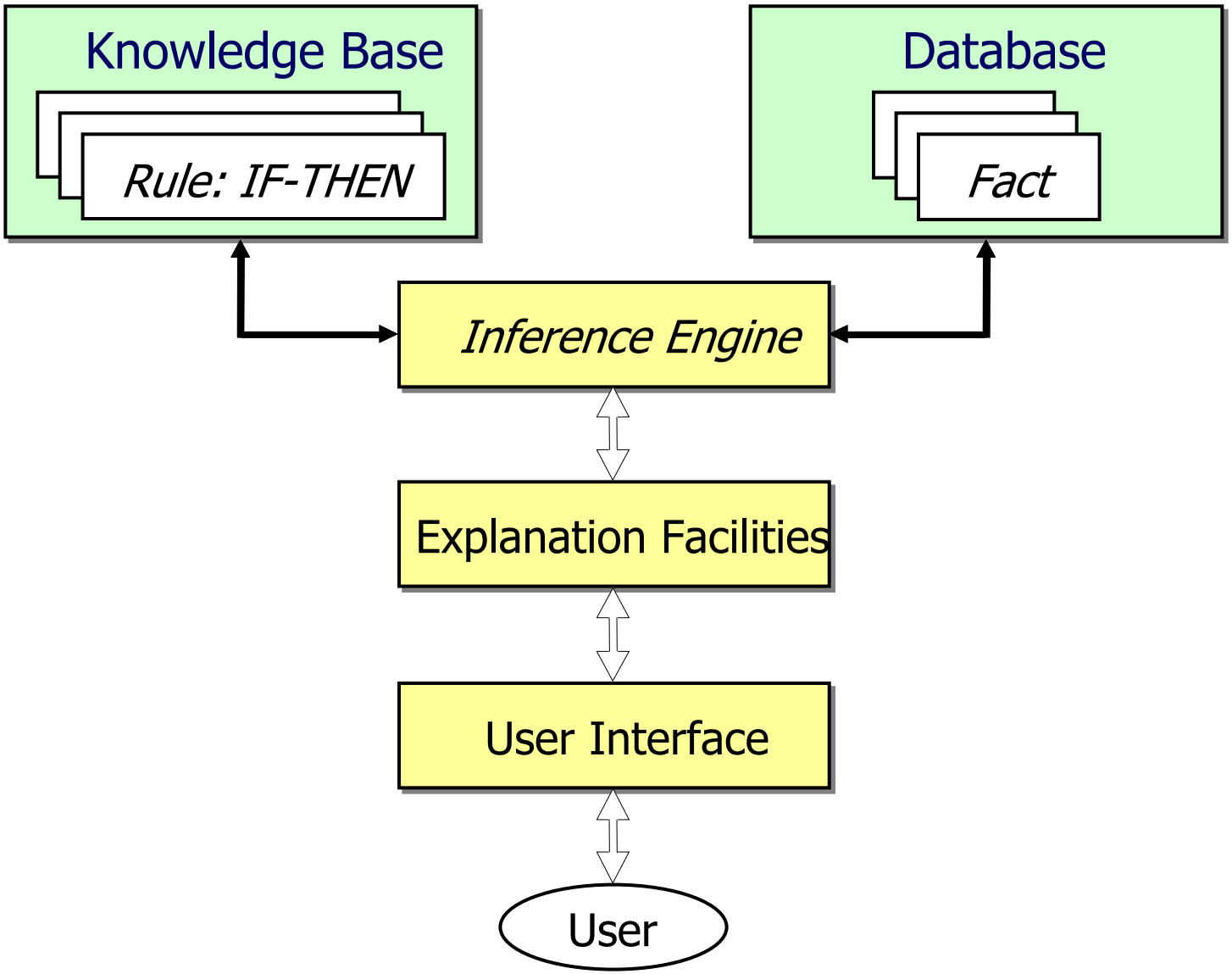
Structure of a rule-based expert system

- In the early seventies, Newell and Simon from Carnegie-Mellon University proposed a production system model, the foundation of the modern rule-based expert systems
- The production model is based on the idea that humans solve problems by applying their knowledge (expressed as production rules) to a given problem represented by problem-specific information
- The production rules are stored in the long-term memory and the problem-specific information or facts in the short-term memory

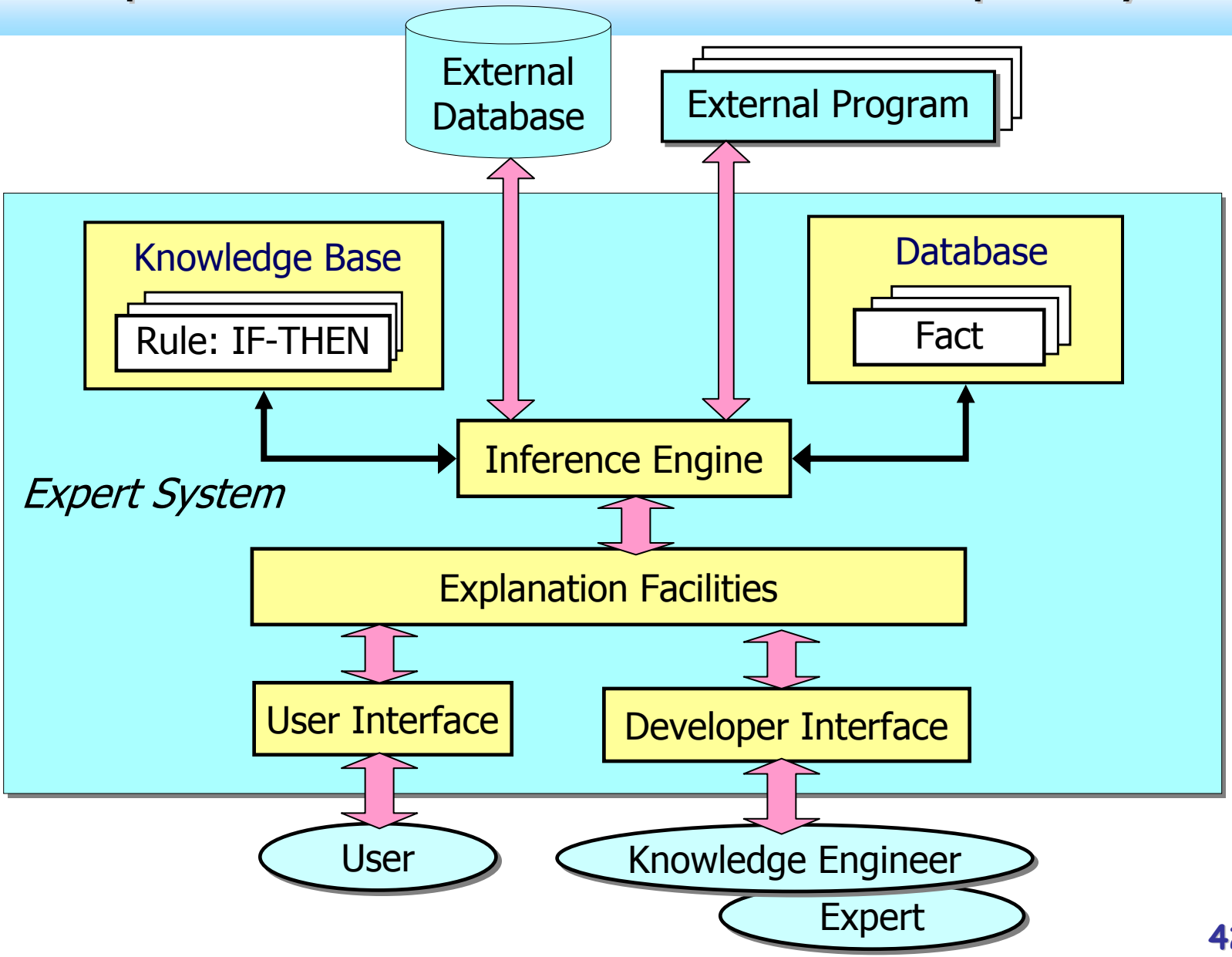
Production system model



Basic structure of a rule-based expert system



Complete structure of a rule-based expert system



Comparison of expert systems with conventional systems and human experts _ 1

Human Experts	*Expert Systems*	*Conventional Programs*
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a *narrow domain*.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a *clear separation of knowledge from its processing*.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	*Trace the rules fired* during a problem-solving session and *explain how* a particular conclusion was reached and *why* specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.

Comparison of expert systems with conventional systems and human experts _ 2

Human Experts	*Expert Systems*	*Conventional Programs*
Use inexact reasoning and can deal with incomplete, uncertain and fuzzy information.	Permit *inexact reasoning* and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Can make mistakes when information is incomplete or fuzzy.	*Can make mistakes* when data is incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data is incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, *changes are easy* to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.