

# Artificial Intelligence (IT3160E)

**Than Quang Khoat**

*khoattq@soict.hust.edu.vn*

---

School of Information and Communication Technology  
Hanoi University of Science and Technology

2022

# Content:

- Introduction of Artificial Intelligence
- Intelligent agent
- Problem solving: Search, Constraint satisfaction
- Logic and reasoning
- **Knowledge representation**
  - **Production rule**
  - **Frame**
  - **Semantic network**
- Machine learning

# Definition of Data

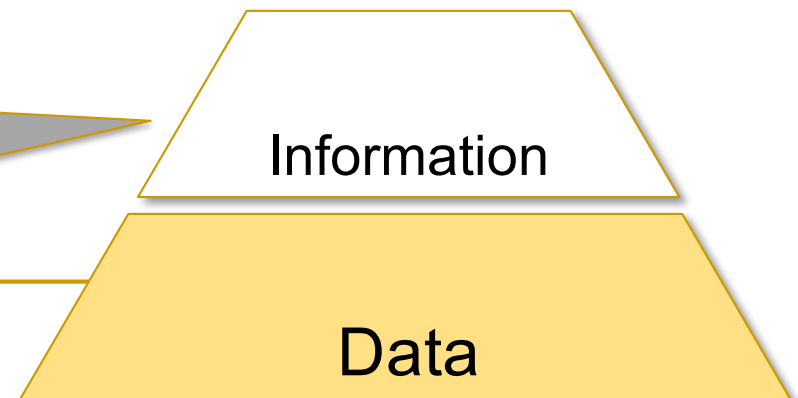
- *Data* (the plural of datum) are just raw facts (Long and Long, 1998)
- *Data* . . . are streams of raw facts representing events . . . before they have been arranged into a form that people can understand and use (Laudon and Laudon, 1998)
- *Data* is comprised of facts (Hayes, 1992)  
Recorded symbols (McNurlin and Sprague, 1998)

Dữ liệu là tín hiệu (signals) thu được do quan sát, đo đạc, thu thập... từ các đối tượng. Cụ thể, dữ liệu là giá trị (values) của các thuộc tính (features) của các đối tượng, được biểu diễn bằng dãy các bits, các con số hay ký hiệu...

# Definition of Information

- That property of data which represents and measures effects of processing them (Hayes, 1992)
- Data that have been shaped into a form that is meaningful and useful to human beings (Laudon and Laudon, 1998)
- Data that have been collected and processed into a meaningful form. Simply, information is the meaning we give to accumulated facts (data) (Long and Long, 1998)

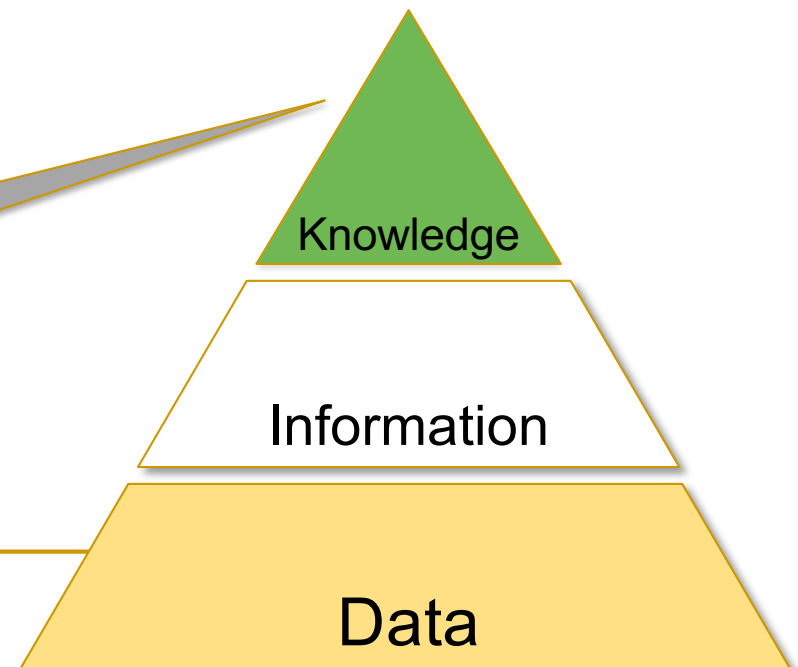
Thông tin là dữ liệu với ý nghĩa (data equipped with meaning), thu được khi xử lý dữ liệu để lọc bỏ đi các phần dư thừa, tìm ra phần cốt lõi đặc trưng cho dữ liệu.



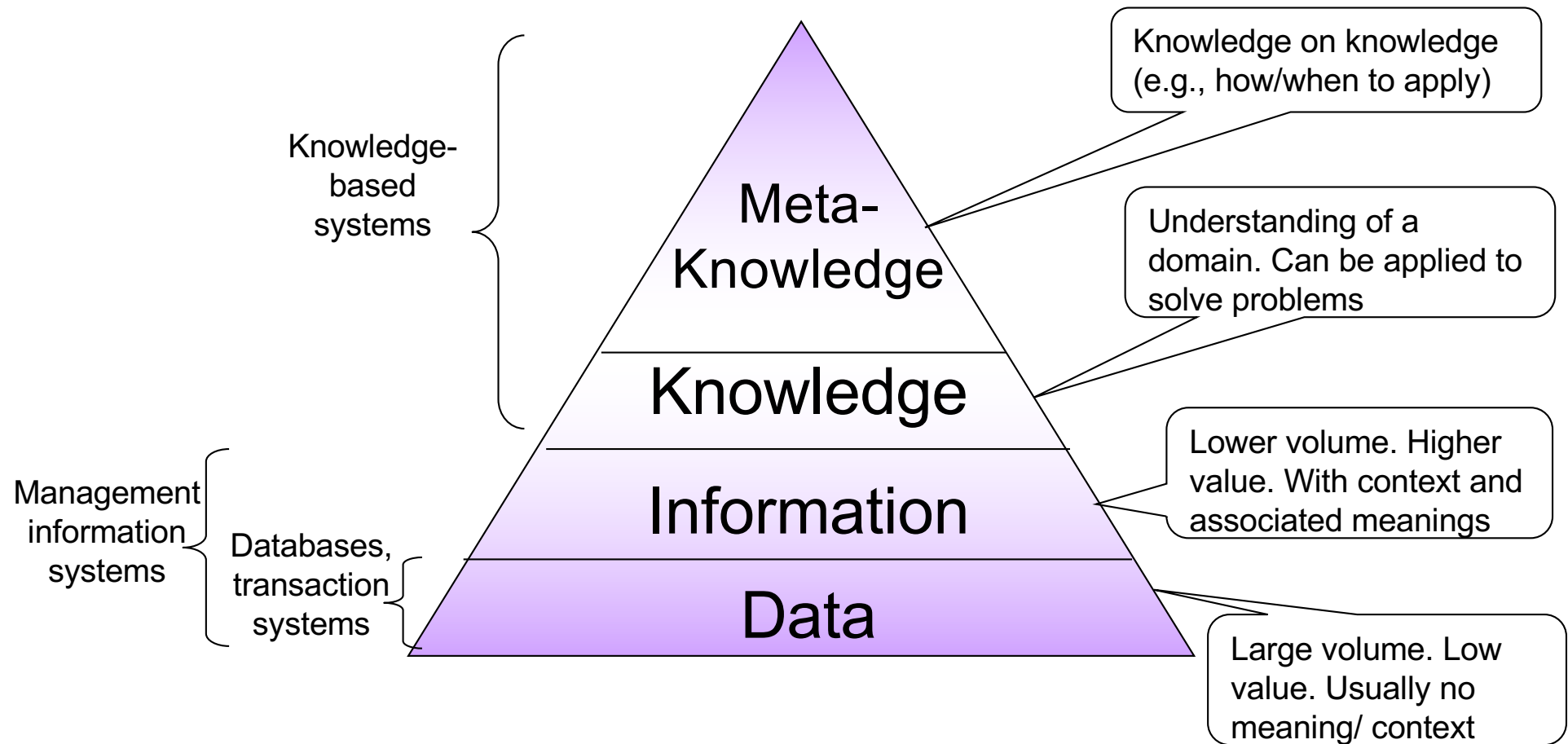
# Definition of Knowledge

- The result of the understanding of information (Hayes, 1992)
- The result of internalizing information (Hayes, 1992)
- Information with direction or intent – it facilitates a decision or an action (Zachman, 1987)

Tri thức là thông tin tích hợp, như quan hệ giữa các sự kiện, giữa các thông tin... thu được qua quá trình nhận thức, phát hiện hoặc học tập.



# Pyramid of Data/Information/Knowledge



(Adapted from “Knowledge Engineering course (CM3016), by K. Hui 2008-2009”)

# Example of Data/Information/Knowledge

- Data

- The temperature outside is 5 degree Celsius

- Information

- It is cold outside

- Knowledge

- If it is cold outside, then you should wear a warm coat

→ The perceived value of data increases as it is transferred into knowledge

→ Knowledge enables useful decisions to be made

# Knowledge representation (1)

- Knowledge representation is an important sub-field of Artificial intelligence
  - Aiming at developing methods and support tools for knowledge representation
- There exist several knowledge representation methods
  - **Production rule**
  - **Frame**
  - **Semantic network**
  - Ontology
  - Probabilistic models
  - Deep models
  - ...



# Knowledge representation (2)

- Completeness

- Does the representation method support the collection and expression of all aspects of knowledge (of the current application domain)?

- Conciseness

- Does the representation method allow efficient collection of knowledge?
- Does the representation method allow easy storage and access of knowledge?

- Computational efficiency

- Transparency

- Does the representation method allow interpretation (understandable by the end user) about the system's computation process and conclusions?

# Rule-based representation (1)

- Rules are classical type for knowledge representation
  - A rule provides some description of how to solve a problem
  - Rules are relatively easy to create and understand
- In the form of **IF  $A_1$  AND  $A_2$  AND ... AND  $A_n$  THEN B**
- **$A_i$** 
  - Are the **conditions (a.k.a. antecedents, premises)**
  - Match against facts (stored in the working memory)
- **B**
  - The **conclusion (a.k.a. consequence, action)**
  - To be added to the working memory

# Rule-based representation (2)

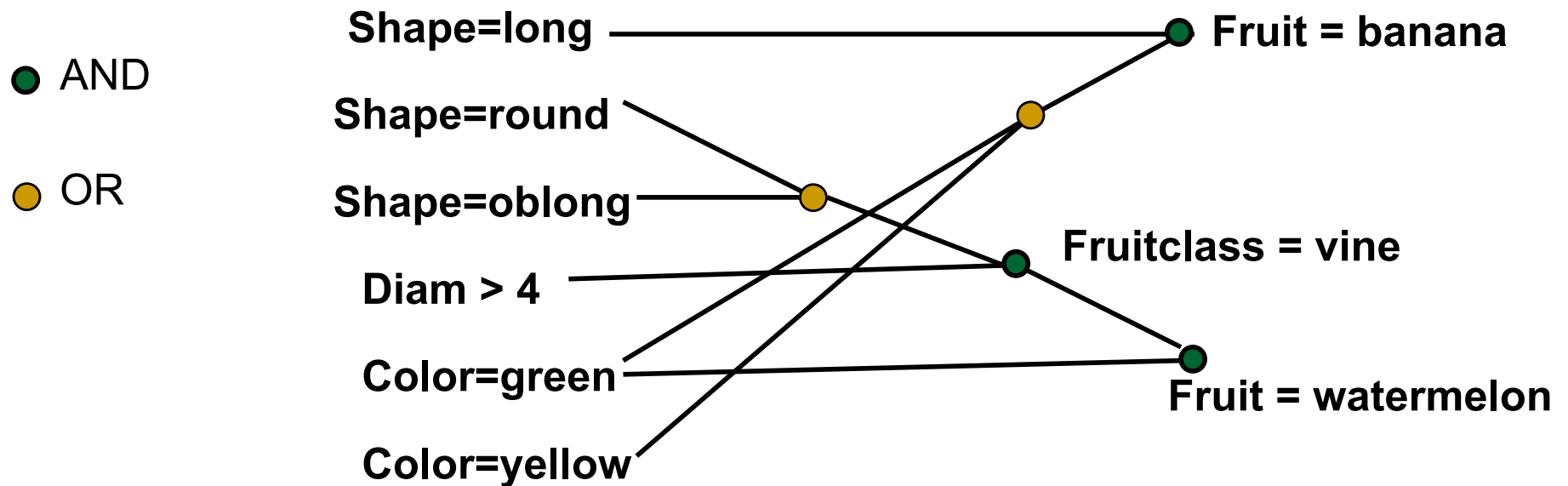
- The condition part of a rule
  - Not need to use disjunctions (OR)
  - A rule with disjunctions in the condition part is converted to a set of corresponding rules with no disjunctions
  - E.g., The rule (IF  $A_1 \vee A_2$  THEN B) is converted to the two rules (IF  $A_1$  THEN B) and (IF  $A_2$  THEN B)
- The conclusion part of a rule
  - Not need to use conjunctions (AND)
  - A rule with conjunctions in the conclusion part is converted to a set of corresponding rules with no conjunctions
  - E.g., The rule (IF ... THEN  $B_1 \wedge B_2$ ) is converted to the two rules (IF ... THEN  $B_1$ ) and (IF ... THEN  $B_2$ )
  - **Not allow to use disjunction (OR)!**

# Typical types of rule

- To represent different types of knowledge
- Associative relation
  - `IF addressAt(x, Hospital) THEN heathIs(x, Bad)`
- Causal relation
  - `IF diseaseType(x, Infection) THEN tempIs(x, High)`
- Situation and action (or recommendation)
  - `IF diseaseType(x, Infection) THEN takeMedicine(x, Antibiotic)`
- Logical relation
  - `IF tempGreater(x, 37) THEN isFever(x)`

# Rules representation by AND/OR graph (1)

- IF (Shape=long) **AND** (Color=(green **OR** yellow)) THEN (Fruit=banana)
- IF (Shape=(round **OR** oblong)) **AND** (Diam > 4) THEN (Fruitclass=vine)
- IF (Fruitclass=vine) **AND** (Color=green) THEN (Fruit=watermelon)



## Rules representation by AND/OR graph (2)

- The rule *IF (Shape=long) **AND** (Color=(green **OR** yellow)) THEN (Fruit=banana)* is combined of:
  - IF (Shape=long) **AND** (Color=green) THEN (Fruit=banana)
  - IF (Shape=long) **AND** (Color=yellow) THEN (Fruit=banana)
- The rule *IF (Shape=(round **OR** oblong)) **AND** (Diam > 4) THEN (Fruitclass=vine)* is combined of:
  - IF (Shape=round) **AND** (Diam > 4) THEN (Fruitclass=vine)
  - IF (Shape=oblong) **AND** (Diam > 4) THEN (Fruitclass=vine)

# Problems with rules

- Infinite rules (rules in loops)
  - IF A THEN A
  - {IF A THEN B, IF B THEN C, IF C THEN A}
- Inconsistent rules (rules contain contradictions)
  - {IF A THEN B, IF B THEN C, IF A AND D THEN  $\neg$ C}
- Unreachable conclusions  
i.e., we cannot use KB to infer some conclusions
- Difficult to modify (update) the knowledge base (KB)
  - The old KB: {IF  $A_1$  THEN  $B_1$ , IF  $A_2$  THEN  $B_2$ , ..., IF  $A_n$  THEN  $B_n$ }
  - Additional condition: C
  - The new KB: {IF  $A_1$  AND C THEN  $B_1$ , IF  $A_2$  AND C THEN  $B_2$ , ..., IF  $A_n$  AND C THEN  $B_n$ }

# Rule usage

## ■ Pattern matching

- ❑ To check whether or not a rule is applicable
- ❑ E.g., If the knowledge base contains the set of rules  $\{\text{IF } A_1 \text{ THEN } B_1, \text{ IF } A_1 \text{ AND } A_2 \text{ THEN } B_2, \text{ IF } A_2 \text{ AND } A_3 \text{ THEN } B_3\}$  and the facts (stored in the working memory) consist of  $A_1$  and  $A_2$ , then the first two rules are applicable

## ■ Chaining

- ❑ To associate (couple) the rules
- ❑ Given a set of rules and a set of facts (premises), which of them should be used, and in which order, to derive (reason) some conclusion?
- ❑ Two strategies of chaining: forward vs. backward
- ❑ We studied these two reasoning strategies in a previous lecture!



# Conflict resolution

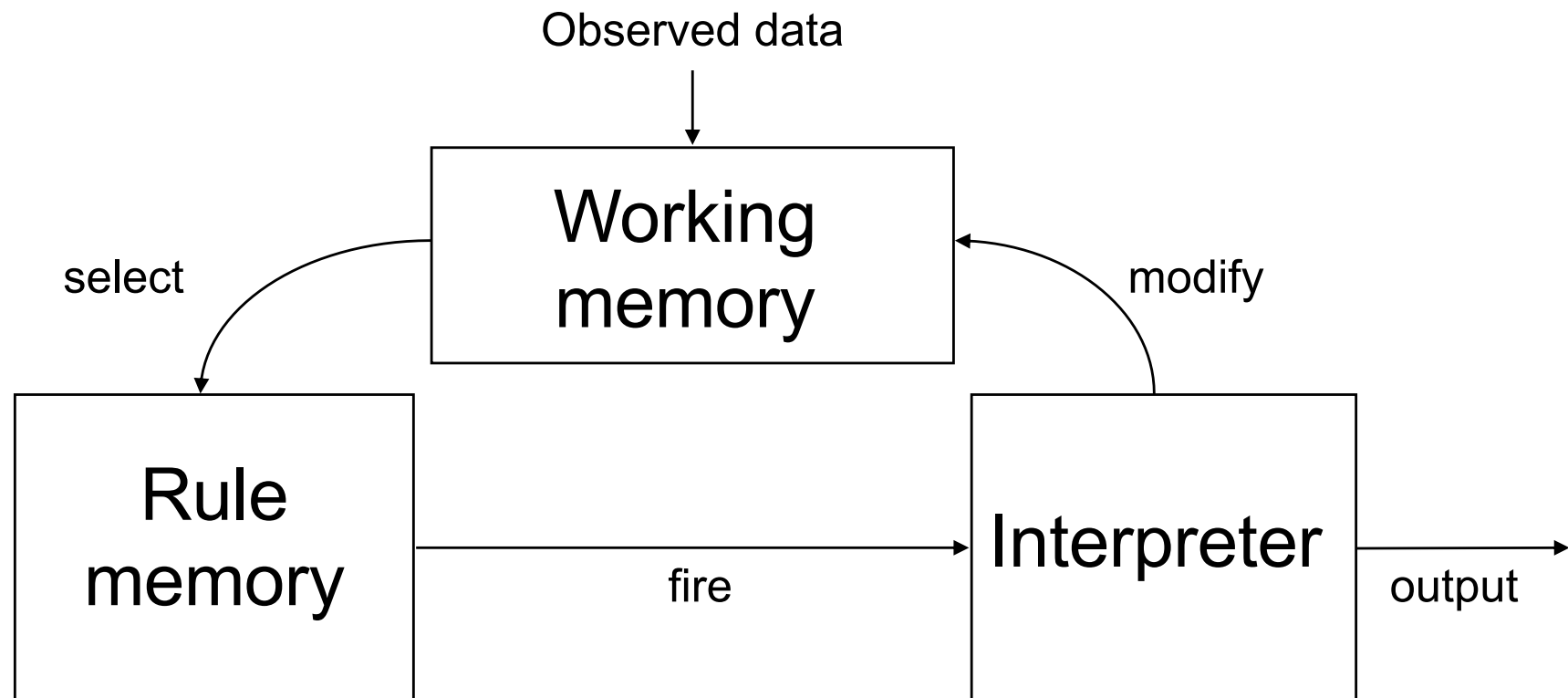
- A **conflict** occurs when more than one rule match the facts (contents) in the working memory
  - Note that a conflict is different from a contradiction (inconsistence)
- A *conflict resolution strategy* (CRS) is needed to decide, in case of a conflict, which rule is to fire (i.e., to be applied)
- An appropriate choice of CRS can make a significant improvement to the system's performance

# Conflict resolution strategy

- Select the **first applicable rule** (i.e., the one comes first in the rule base)
- **Don't** select rules that **duplicate existing results** (i.e., rules whose application result in existing facts)
- Select the **most specific rule** (i.e., the one with the most conditions attached)
- Prefer rules that **match the most recent facts**
- Select the rule with the **highest certainty** (in an uncertain rule base)
- ...
- A **combination** of the above strategies

# Rule-based system (1)

Architecture of a typical rule-based system



(<http://www.cwa.mdx.ac.uk/bis2040/johnlect.html>)

# Rule-based system (2)

- Working memory
  - Holds facts (items of data)
  - Their presence and/or their absence causes the interpreter to trigger certain rules
- Rule memory
  - Is the knowledge base of the system
  - Holds rules
- Interpreter
  - The system is started by putting a suitable data item into the working memory
  - When data in the working memory matches the conditions of one of the rules in the rule memory, the rule fires (i.e., is brought into action)

# Rule-based representation: Advantages (1)

- Notational convenience
  - Very close to expressions in a natural language
  - It is easy to express suitable pieces of knowledge by rules
- Easy to understand
  - IF-THEN rules are very easy (probably the easiest) to understand to human
  - Easy for the experts in the specific domain, which the system is concerned with, to criticize and improve

## Rule-based representation: Advantages (2)

- A purely declarative form of knowledge representation
  - One gathers pieces of knowledge (in the form of IF-THEN rules) about a particular subject, and puts them into a rule base
  - One doesn't need to care about when/how/in which sequence the rules are used – the system takes care of that

# Rule-based representation: Disadvantages

- Restricted power of expression
  - In many practical problems, useful pieces of knowledge don't fit the (IF-THEN) pattern
- The interaction and the order of rules in a rule base may cause some unexpected effects
  - In the design and maintenance of a rule base, each (new) rule can not be considered in isolation
  - Very hard and high cost to consider all possible rule interactions
- Expensive to build, maintain, update

# Frame-based representation: Motivation

- How to represent the knowledge “The bus is yellow”?
- Solution 1. `Yellow(bus)`
  - ❑ The question “What is yellow?” can be answered
  - ❑ The question “What is the color of the bus?” cannot be answered
- Solution 2. `Color(bus, yellow)`
  - ❑ The question “What is yellow?” can be answered
  - ❑ The question “What is the color of the bus?” can be answered
  - ❑ The question “Which property of the bus has value yellow?” cannot be answered
- Solution 3. `Prop(bus, color, yellow)`
  - ❑ All the questions mentioned above can be answered



# Frame-base representation

- Representation of an object:

## **Prop(Object, Property, Value)**

- Called the *object-property-value* representation

- If we merge many properties of the object of the same type into one structure, then we get the object-centered representation

*Prop(Object, Property<sub>1</sub>, Value<sub>1</sub>)*

*Prop(Object, Property<sub>2</sub>, Value<sub>2</sub>)*

*...*

*Prop(Object, Property<sub>n</sub>, Value<sub>n</sub>)*

## **Object**

Property<sub>1</sub>

Property<sub>2</sub>

...

Property<sub>n</sub>

# Object-centered representation

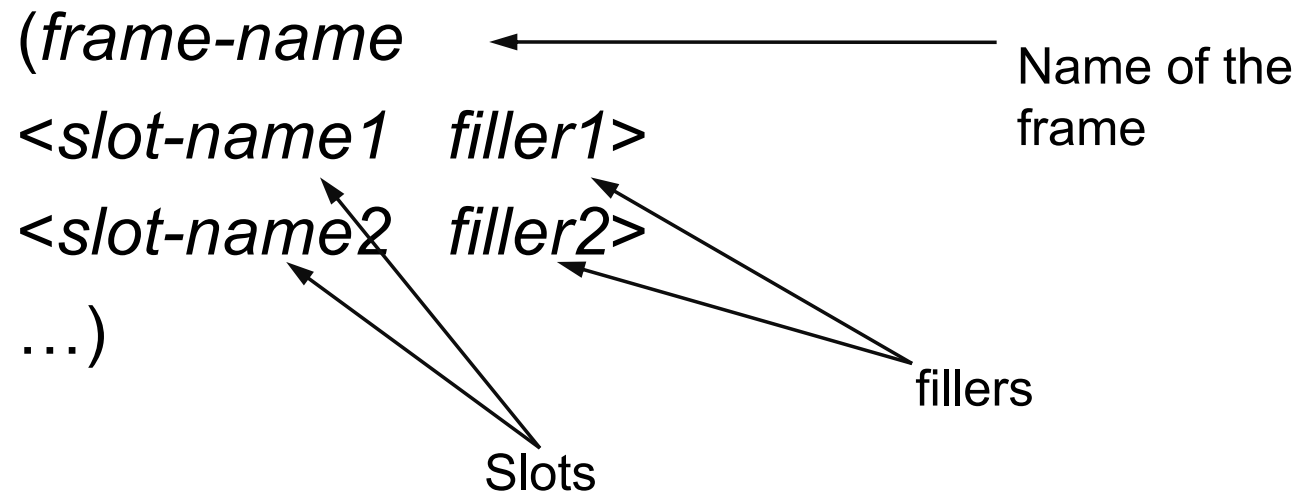
- The object-property-value is a natural way to represent knowledge about objects
- Physical objects
  - A *desk* has a surface-material, number of drawers, width, length, height, color, etc.
- Situations
  - A *class* has a room number, participants, teacher, day, time, etc.
  - A trip has a departure, destination, transportation means, accommodations, etc.

# Frame

- Two types of frames: individual and generic
- **Individual frames.** To represent a single object like a person, a trip, etc.
- **Generic frames.** To represent categories of objects, like students, trips, etc.
- Example
  - A generic frame: European\_City
  - An individual frame: City\_Paris

# Representation of a frame

- A frame is a named list of buckets called **slots**
- What goes in the bucket is called a **filler of the slot**



# Individual frames

- An individual frame has a special slot called **INSTANCE-OF** whose filler is the name of a generic frame

- Example

(toronto                      % lower case for individual frames  
<:**INSTANCE-OF** CanadianCity>  
<:Province ontario>  
<:Population 4.5M>  
...)

# Generic frames

- A generic frames may have **IS-A** slot whose filler is the name of a generic frame

- Example

```
(CanadianCity          % upper case for generic frames
  <:IS-A City>
  <:Province CanadianProvince>
  <:Country canada>
  ...)
```

# Frames: Inference control (1)

- Slots in generic frames can have associated procedures that are executed and 'control' inference

- Two types of procedures: **IF-NEEDED** and **IF-ADDED**

- **IF-NEEDED** procedure

- Executes when no slot filler is given and the value is needed

- E.g.,        (Table  
                  <:Clearance [**IF-NEEDED** computeClearance]>  
                  ...)

*computeClearance* is a procedure (for example) to calculate the clearance of the table

# Frames: Inference control (2)

## ■ **IF-ADDED** procedure

- If a slot filler is given its effect may propagate to other frames (e.g., to assure constraints)

- E.g., (Lecture

<:DayOfWeek WeekDay>

<:Date [**IF-ADDED** computeDayOfWeek]>

...)

The filler for the slot *:DayOfWeek* will be calculated when the slot *:Date* is filled



# Frames: Defaults (1)

- Let's consider the following generic frame

```
(CanadianCity  
  <:IS-A City>  
  <:Province CanadianProvince>  
  <:Country canada>  
  ...)
```

- Let's consider the following individual frame

```
(city134  
  <:INSTANCE-OF CanadianCity>  
  ...)
```

- For the frame `city134`, the (default) filler for the slot `:Country` is `canada`

## Frames: Defaults (2)

- Let's consider the following individual frame

```
(city135  
  <:INSTANCE-OF CanadianCity>  
  <:Country holland>  
  ...)
```

- For the frame `city135`, the filler for the slot `:Country` is `holland`

# Frames: Inheritance (1)

- Procedures and fillers of more general frame are applicable to (i.e., inherited by) more specific frame through the inheritance mechanism

- Example

(CoffeeTable  
<:**IS-A** Table>  
...)

(MahoganyCoffeeTable  
<:**IS-A** CoffeeTable>  
...)

# Frames: Inheritance (2)

## ■ Example

(Elephant

<:**IS-A** Mammal>

<:Colour gray>

...)

(RoyalElephant

<:**IS-A** Elephant>

<:Colour white>

...)

(clyde

<:**INSTANCE-OF** RoyalElephant>

...)

# Frames: Reasoning

- **Basic reasoning** goes like this
  1. The user instantiates a frame, i.e., declares that an object or situation exists
  2. Slot fillers are inherited where possible
  3. Inherited IF-ADDED procedures are run, which causes more frames to be instantiated and slots to be filled
- If the user or any procedure **requires the filler of a slot**, then:
  - If there is a filler, it is used
  - Otherwise, an inherited IF-NEEDED procedure is run, which potentially causes additional actions

# Frame-based representation: Advantages

- Combine procedural and declarative knowledge using one knowledge representation scheme
- Frames can be structured hierarchically, which allows easy classification of knowledge
- Reduce the complexity of knowledge base construction by allowing a hierarchy of frames to be built up
- Allow to constrain allowed values (i.e., to allow values to be entered within a specific range)
- Allow to store default values (i.e., by IS-A slot, the information of a more generic frame is used to automatically fill slots in a more specific frame)

## Frame-based representation: Disadvantages

- Require (much) attention in the design stage to ensure that suitable taxonomies, i.e., agreed structures for the terminology are created for the system
- Can lead to 'procedural fever', that is, the apparent requirement to focus on making appropriate procedures rather than checking the overall structure and content of the frames
- Can be inefficient at runtime because frames do not provide the most efficient method to store data in a computer

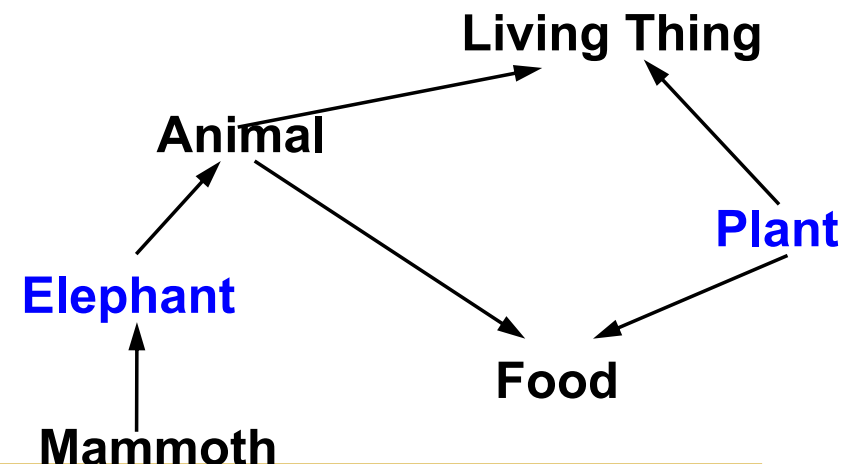
# Semantic networks (1)

- SN was first proposed by Quillian in 1966, as a model of human memory
- Motivations
  - To understand the structure of human memory, and its use in language understanding
  - What sort of representational format can permit the “meanings” of words to be stored, so that humanlike use of these meanings is possible?
  - Human brain uses the same memory structure for a variety of tasks?
- Psychological evidence: memory uses associative links to understand words
- Wish to encode dictionary definition of words, in order to
  - Compare and contrast meanings of two words
  - Generate quasi-English sentences, ...



# Semantic networks (2)

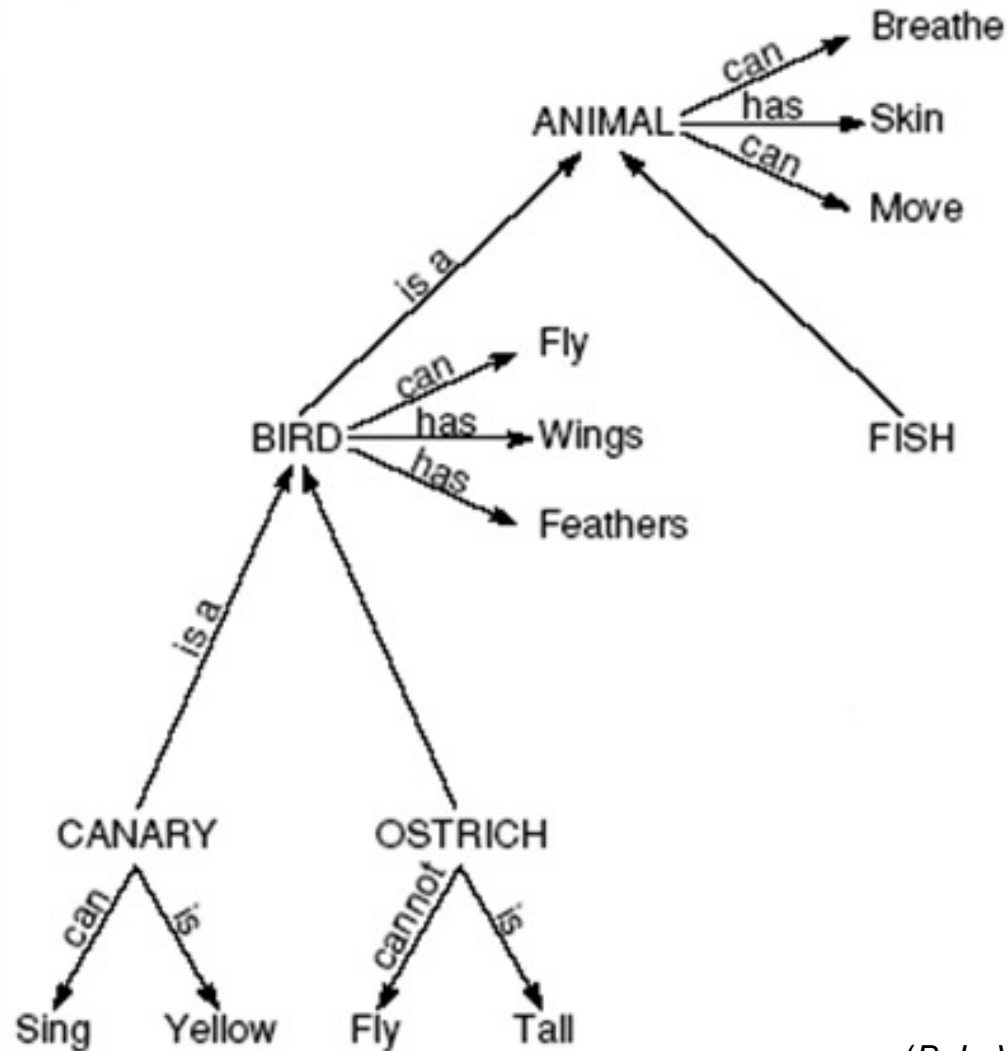
- Semantic network (SN) is a graph-based representation
- A SN is a network of **nodes** and **links** to represent the definition of a concept (or a collection of concepts)
  - The nodes represent concepts
  - The links represent the relations between concepts
- The reasoning (inference) in SNs is done based on the propagation of
  - Activation
  - Inheritance



# Semantic networks: Syntax

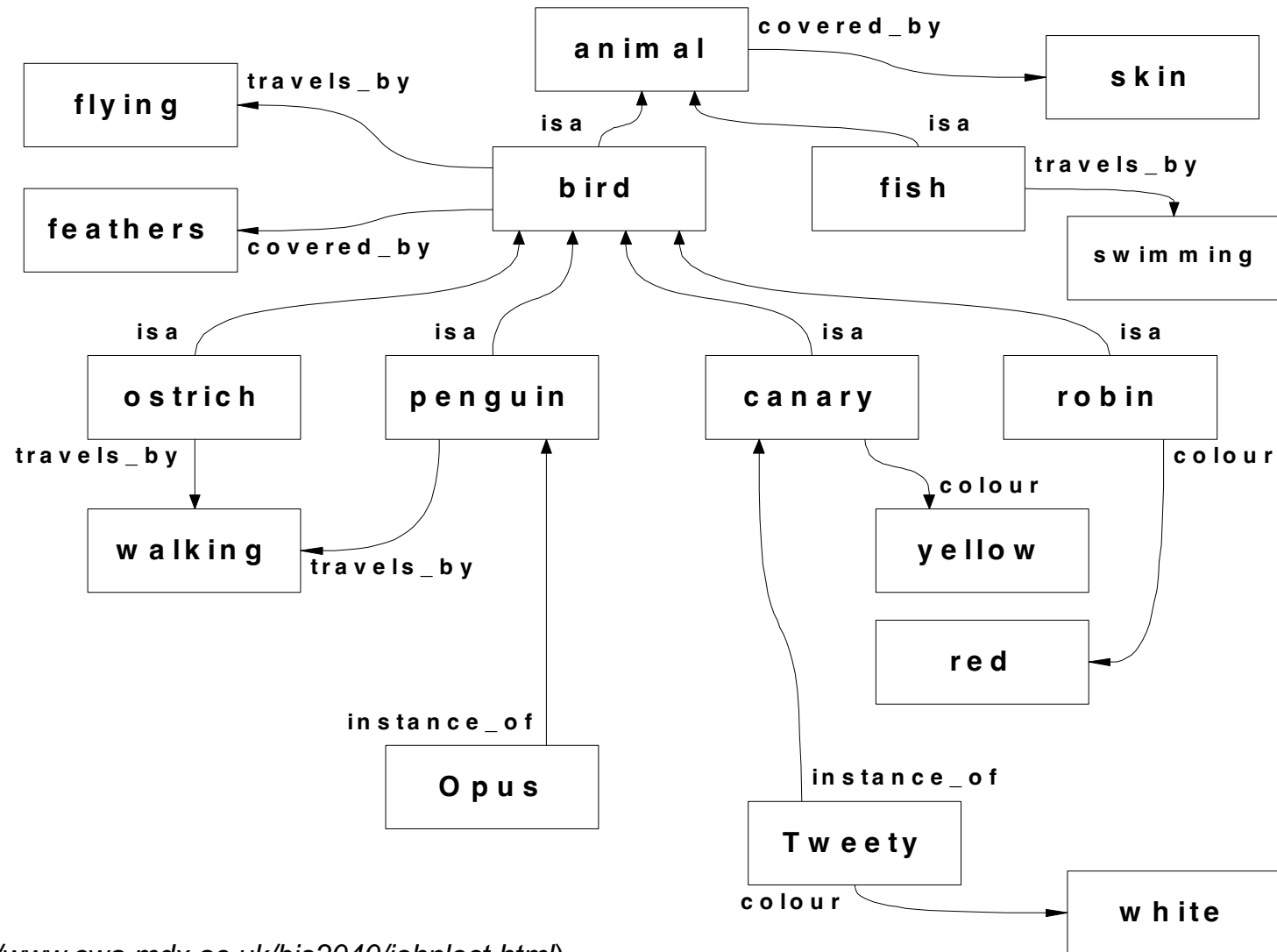
- **Nodes** represent *concepts, actions* or *objects* in the problem domain
- **Links** are directional and labeled relations between nodes
- Two kinds of links: **inheritance-oriented** and **domain-specific**
- **Inheritance-oriented** link, to represent that:
  - Node *A* is a sub-class (sub-category) of node *B* (i.e., *IS-A* link)
  - Node *A* is an instance of node *B* (i.e., *INSTANCE-OF* link)
- **Domain-specific** link, to represent that:
  - Node *A* relates to (i.e., has a relation with) node *B*
  - E.g., HAS, CAN, HAS-PART, CAUSES, HAS-COLOR, ... links

# Semantic network: Example (1)



(B. L. Vrusias, course AI-CS289)

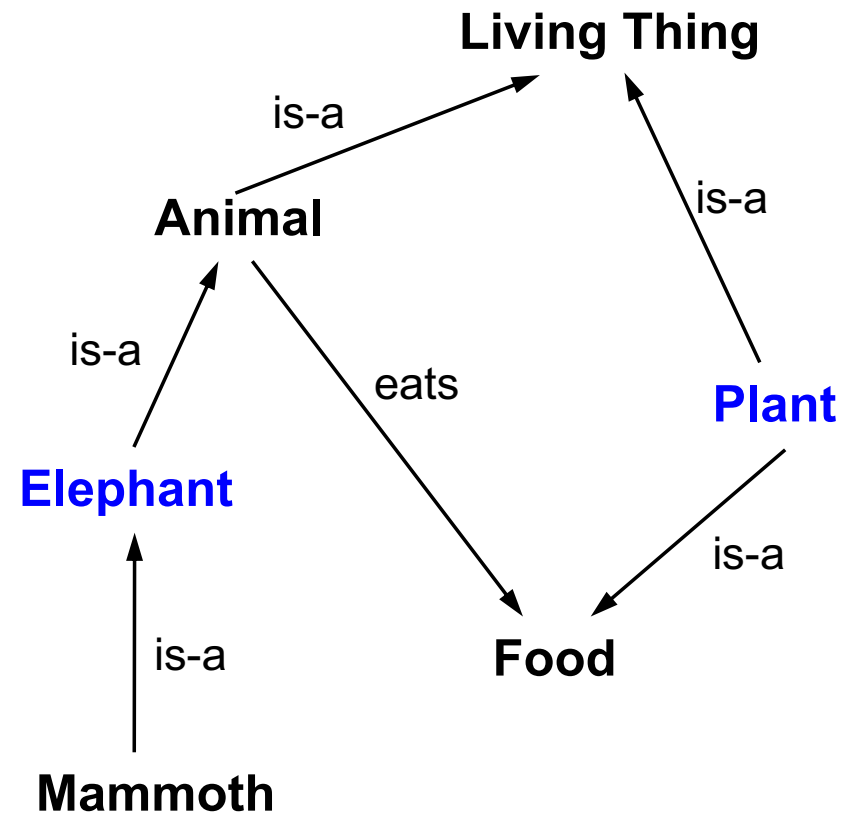
# Semantic networks: Example (2)



(<http://www.cwa.mdx.ac.uk/bis2040/johnlect.html>)

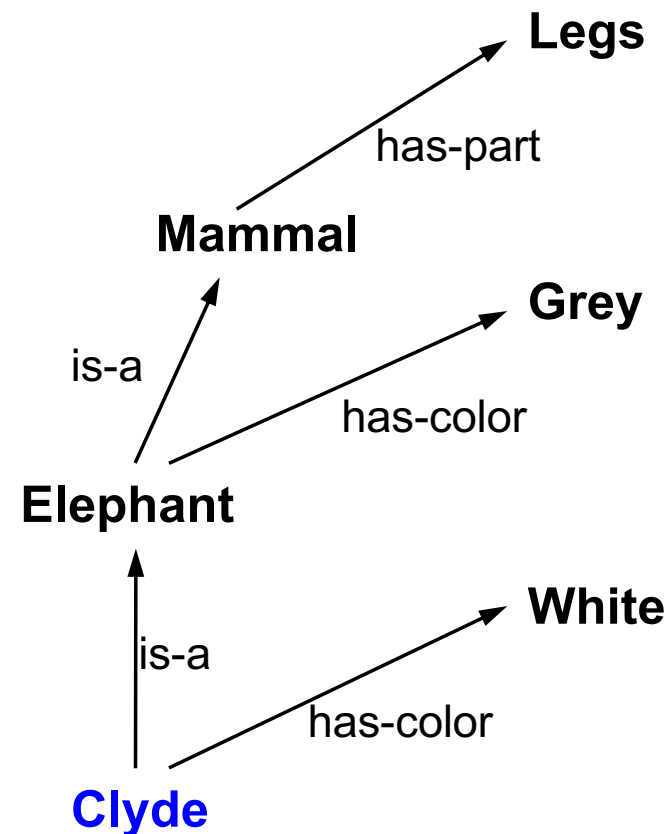
# Semantic networks: Spreading activation

- Given two concepts, spreading activation will activate concepts by following links from one or both
- Allows identification of concepts "between" these two concepts that help relate them
  - E.g., Spreading activation between the two concept "Elephant" and "Plant"



# Semantic networks: Inheritance

- Properties of super-classes (super-categories) are inherited to sub-classes (sub-categories)
- **Universal inheritance:** All relations are inherited
- **Default inheritance:** Relations are inherited unless there are conflicting information at a more specific node
- Psychological studies shows that in human's response time:  
"Clyde has-color White" < "Clyde has-part Legs"

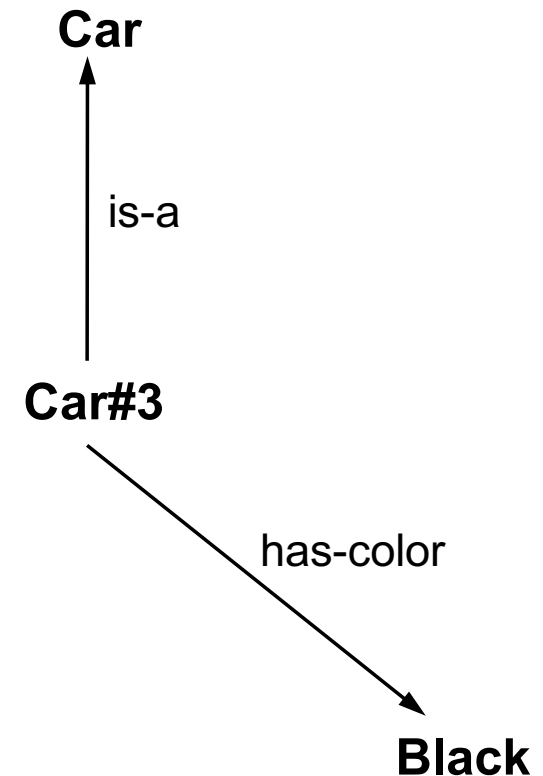


# Semantic networks: Semantics (1)

- Semantic networks appear intuitive – The syntax is seductively uniform
- But: Different systems can have *different interpretations* for the *same “graph”*
- What are the semantics of Semantic Networks?
  - “Since the semantics of any given language is dependent of the interpretation of the primitive elements....., the well-definedness of a network language rests heavily on the set of node and link types that it provides” (Brachman, p204, Readings in KR)

# Semantic networks: Semantics (2)

- What does the following SN mean?
  - This is a definition of a “black-colored” car (**Definitional information**)
  - This states that there exists a car that is black (**Assertional information**)
  - This states that a particular car (i.e., car#3) is black (**Asserts existence**)





# Semantic networks: Advantages

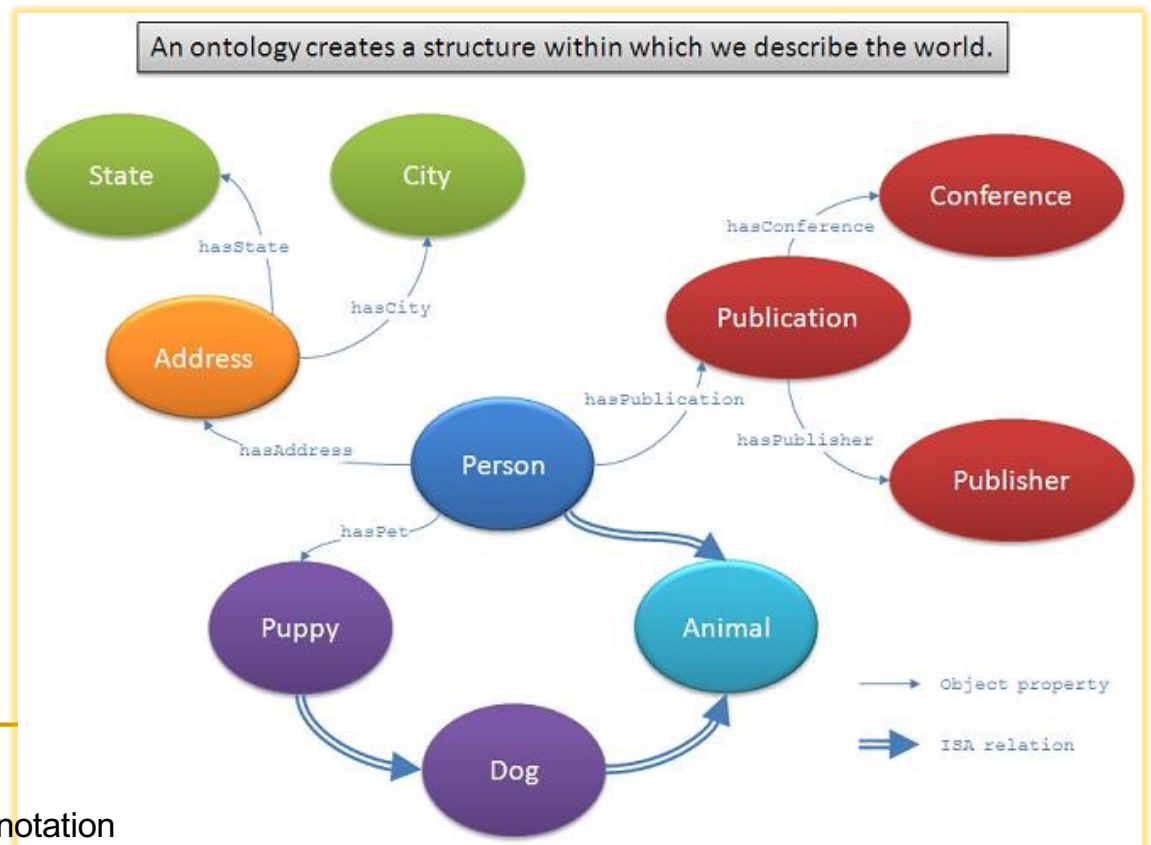
- Explicit in visualization and easy to understand
  - Often used as a communication tool between the knowledge engineer and the expert during the knowledge acquisition phase
- SNs are particularly good at representing knowledge in the form of hierarchies
  - Knowledge is hierarchically categorized (classified)
- The net is its own index – Quick inference possible

# Semantic networks: Disadvantages

- No common interpretation – Lack well-defined semantics
- Problematic in representation of negation and disjunction
  - E.g., “John does not go fishing”, “John eats pizza or fish and chips”
- Hard to choose good primitives

# Ontology

- **Ontology** provides formal representation (structured description) about concepts
- An **ontology** is a common dictionary for a specific domain, for modeling
  - Entities and/or concepts
  - Attributes and their relations



# Ontology (2)

- An ontology can be considered as a knowledge base
  - a database schema is an ontology
- An ontology can be used for different purposes
- A knowledge base contains the knowledge which is necessary for solving real problems in a domain

# Ontology: Motivation

## ■ Technological

- ❑ Many knowledge-based systems use ontology which describes the knowledge of a domain
- ❑ The construction of an ontology is often expensive (both time and cost)
- ❑ Why not share ontologies?

## ■ Scientific

- ❑ To understand the fundamental issues in human to conceptualize the world and realize the concepts.

# Aspects of an ontology (1)

## ■ Ontology contents

- Entity types, relation types
- Ex: Blocks organization
  - Entities: Blocks, Robot Hands
  - Attributes: shapes of blocks, color of blocks
  - Relations: On, Above, Below, Grasp
  - Processes: design or build a tower

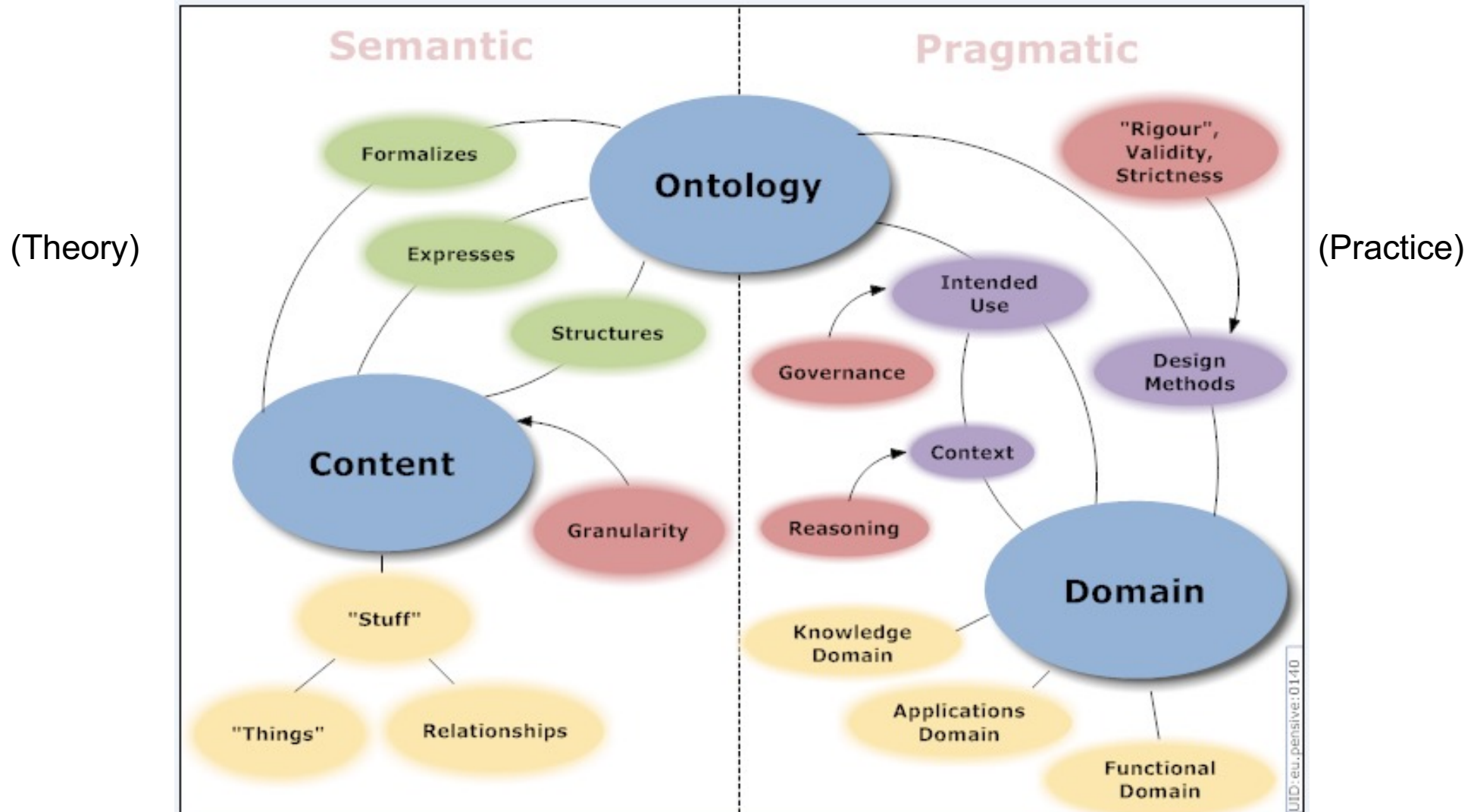
## ■ Ontology type

- Basic relations (e.g.: instance-of, subclass)?
- Definitions of concepts (and their constraints)?
- The ability of the chosen language to represent concepts?
- Process-centric or object-centric?

# Aspects of an ontology (2)

- Purpose of an ontology
  - Share knowledge
    - E.g.: between users, systems, ...
  - Reuse knowledge
    - E.g.: reuse parts of the knowledge when a model or system changes
- Construction of an ontology
  - Collect? or build from scratch?
  - When collecting knowledge, we need to check:
    - The knowledge quality?
    - The differences in contents?
    - The reliability of collected knowledge?
    - The ability to reuse?

# Ontology



Source: [http://www.emiliosanfilippo.it/?page\\_id=1172](http://www.emiliosanfilippo.it/?page_id=1172)



# Building ontology: issues

- Determine the limit (scope) and purpose
- Consider reuse of existing ontologies which relate to our problem domain
- List all the concepts
- Define the classification of concepts
- Define the attributes
- Define the aspects or constraints
- Provide specific examples (instances)
- Check anomalies

# Building ontology: scope and purpose

- No “standard” ontology for a domain
  - An ontology is an abstraction of a domain, and many ontologies can do the same thing.
- This abstraction should consider:
  - The future use of ontology
  - Scalability of ontology
- Some questions should be answered in this phase:
  - For what domains are this ontology?
  - What tasks can use this ontology?
  - What questions can this ontology answer?
  - Who will maintain this ontology?

# Building ontology: reuse

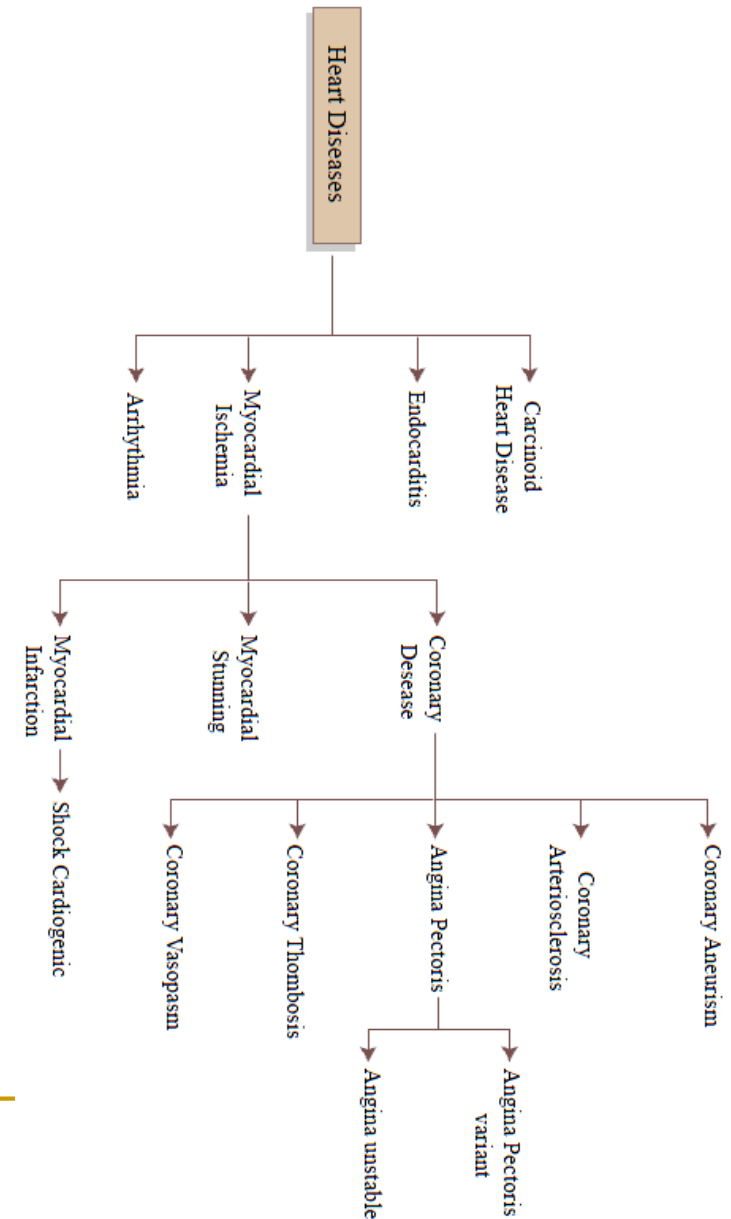
- There are many existing ontologies, due to the fast development of Internet and Semantic Web.

# Building ontology: concept list

- Determine the list of all concepts in an ontology
  - Nouns are often used as **class names**
  - Verbs or verb phrases are often used as **property names**
- Use some tools for knowledge construction, to obtain
  - Set of concepts
  - Hierarchy (or structure) of those concepts

# Building ontology: classification

- Related concepts should be organized in a taxonomic hierarchy
  - Two methods: top-down vs. bottom-up
  - Need considering: reliability (accuracy) and efficiency for inference
- Make sure the hierarchy to be a good classification of concepts
  - If A is a subclass of B, then every example of A should be an example of B

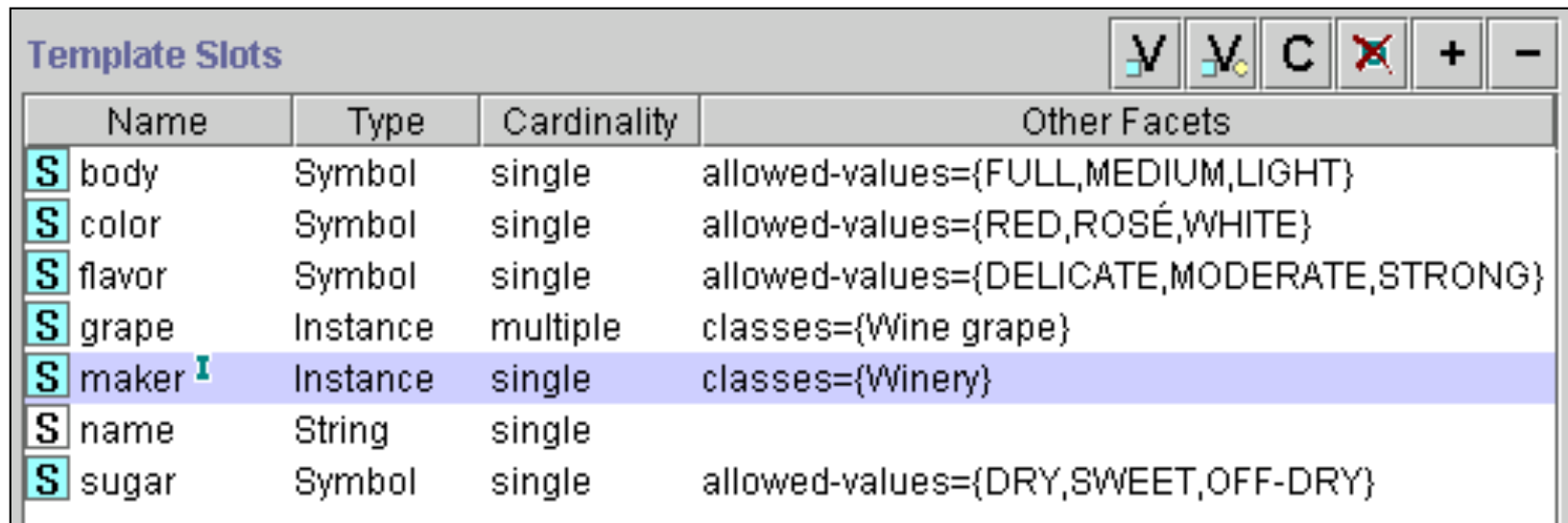










# Building ontology: attributes

- Defining attributes is often done when determining a classification of concepts
- **Requirement:** If  $A$  is a subclass of  $B$ , then every attribute of  $B$  applies to  $A$
- When defining an attribute for one class/concept, we must determine its *data type* and *value domain*

# Building ontology: constraints

- Constraints on cardinality (the size of a set)
- The possible values
- Properties of relations
  - symmetric, inverse, transitive, ...



Name	Type	Cardinality	Other Facets
 body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
 color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
 flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
 grape	Instance	multiple	classes={Wine grape}
 maker 	Instance	single	classes={Winery}
 name	String	single	
 sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

([protege.stanford.edu/amia2003/AMIA2003Tutorial.ppt](http://protege.stanford.edu/amia2003/AMIA2003Tutorial.ppt))

# Building ontology: examples

- Provide examples (instances) for each class
- The number of examples can be much larger than the number of classes
- This step is often done manually
  - Take from some existing sources (e.g., databases)
  - Extract from some textual datasets



# Building ontology: anomaly check

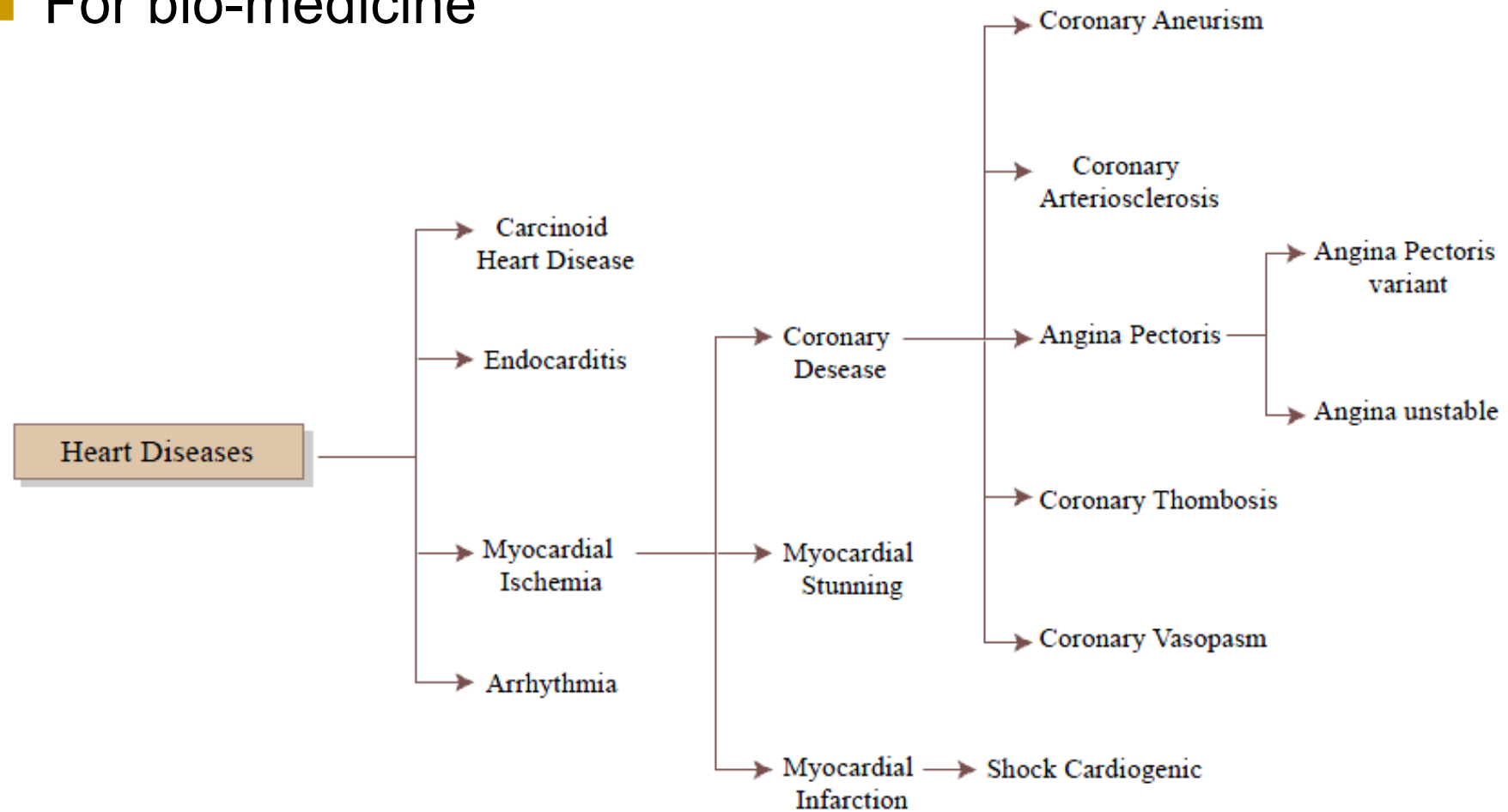
- Check to remove conflicts or contradictions in an ontology
- Examples of contradictions
  - Incompatible in value domains, in values, transitivity, symmetry, inversion, ...
  - Incompatible in constraints about cardinality of an attribute

# Some ontologies

- CYC (common, general)
  - $10^5$  concepts,  $10^6$  premises
- SENSUS (extension of WordNet for text)
  - 70.000 concepts
- SUMO (general)
  - 1000 concepts, 4200 evaluations
- UMLS (in medicine)
  - 135 semantic types, 54 semantic relations, 975.354 concepts
- WordNet (dictionary)
  - 152.059 nouns, 115.424 phrases

# UMLS ontology

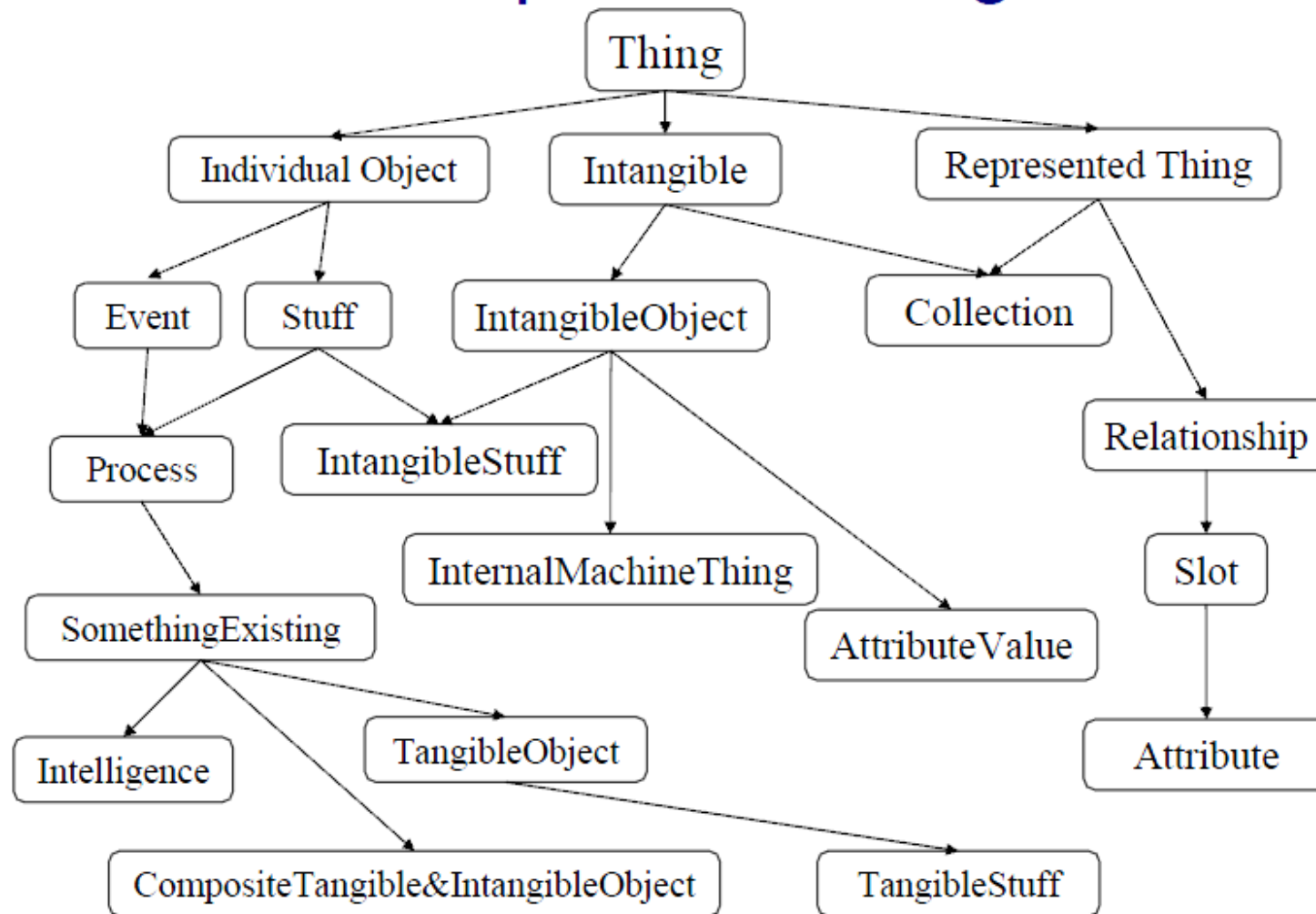
## ■ For bio-medicine



(Figure by MIT OCW)

# CYC ontology

- Encode all of human common sense knowledge



# References

- R. Hayes. *The Measurement of Information*. In Vakkari, P. and Cronin, B. (editors): *Conceptions of Library and Information Science*, pp. 97–108. Taylor Graham, 1992.
- K. C. Laudon and J. P. Laudon. *Management Information Systems: New Approaches to Organisation and Technology (5th edition)*. Prentice-Hall, 1998.
- L. Long and N. Long. *Computers (5th edition)*. Prentice-Hall, 1998.
- B. McNurlin and R. H. Sprague. *Information Systems Management in Practice (4th edition)*. Prentice-Hall, 1998.
- J. A. Senn. *Information Systems in Management*. Wadsworth Publishing, 1990.
- J. Zachman. *A Framework for Information Systems Architecture*. IBM Systems Journal, 26(3): 276–292, 1987.