CS 362: Intelligent Systems

Unit 6

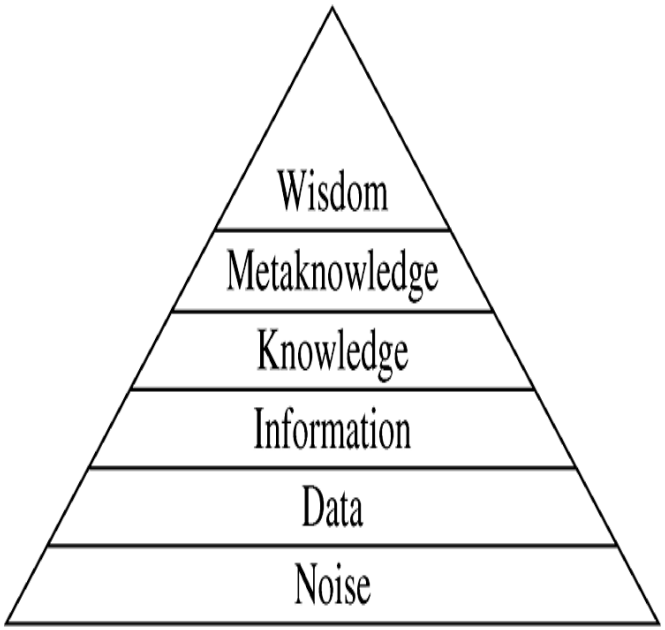
Knowledge Representation

Knowledge Representation

- Issues in Knowledge Representation
- AI Representational Systems
  - Associationist Theories of Meaning
  - Semantic Networks
  - Standardization of Network Relationships
  - Scripts
  - Frames
- Conceptual Graphs: a Network Language
  - Introduction to Conceptual Graphs
  - Types, Individuals, and Names
  - The Type Hierarchy
  - Generalization and Specialization
  - Propositional Nodes
  - Conceptual Graphs and Logic

Review(i) The Pyramid of Knowledge

- **The Pyramid of Knowledge**



Wisdom: Using knowledge in a beneficial way
Metaknowledge: Rules about knowledge
Knowledge: Rules about using information
Information: Potentially useful for knowledge
Data: Potentially useful information
Noise: No apparent information.

Review(ii) Knowledge vs. Expert Systems

- Knowledge is of primary importance in expert systems. In fact, an analogy to classic expression
- Algorithms + Data Structures = Programs
- Knowledge + Inference = Expert Systems

- Knowledge representation is key to the success of expert systems.
- Expert systems are designed for knowledge representation based on rules of logic called inferences.
- Knowledge affects the development, efficiency, speed, and maintenance of the system.

Review(iii) How is Knowledge Used?

- Knowledge has many meanings – data, facts, information.
- How do we use knowledge to reach conclusions or solve problems?

  Heuristics refers to using experience to solve problems – using precedents.

  Expert systems may have hundreds / thousands of micro-precedents to refer to.

Review(iv) Knowledge in Rule-Based Systems

- **Knowledge:**
  - **Knowledge** is part of a hierarchy.
  - **Knowledge** refers to rules that are activated by **facts** or other **rules**.
  - Activated rules produce new facts or conclusions.
  - Conclusions are the end-product of inferences when done according to formal rules.
- **Metaknowledge:**
  - **Metaknowledge** is knowledge about knowledge and expertise.
  - Most successful expert systems are restricted to as small a domain as possible.
  - In an expert system, an ontology is the metaknowledge that describes everything known about the problem domain.
- **Wisdom :**
  - **Wisdom** is the metaknowledge of determining the best goals of life and how to obtain them.

Review(v) Representation and Intelligence

- The question of representation, or how to best capture critical aspects of intelligent activity for use on a computer.

Three predominant approaches to representation taken by the AI research community over this time period.

The first theme(weak method problem-solving)

Articulated in the 1950s and 1960s by Newell and Simon in their work with the Logic Theorist , is known as weak method problem-solving.

The second (Strong method problem solving)

Common throughout the 1970s and 1980s, and adopted by the early expert system designers, is strong method problem solving .

In more recent years, (Agent-Based)

Especially in the domains of robotics and the internet (Brooks 1987, 1989; Clark 1997), the emphasis is on distributed and embodied or agent-based approaches to intelligence.

Issues in Knowledge Representation (i)

- The representation of information for use in intelligent problem solving offers important and difficult challenges that lie at the core of AI.
- Representation Schemes
  - Scheme – data/knowledge structure
  - Semantic network
    1. Scripts: are a data structure used to represent a sequence of events
    2. Frames provide visual context for interpretation
  - Conceptual graphs
  - Stochastic methods
  - Connectionist (neural networks)
- Implementation media
  - Medium – implementation languages
  - Prolog, Lisp, Scheme, even C and Java

Issues in Knowledge Representation (ii)

- knowledge base is described as: (Bobrow 1975)
  - A mapping between the objects and relations in a problem domain
  - The computational objects and relations of a program
- The results of inferences in the knowledge base are assumed to correspond to the results of actions or observations in the world.
- The computational objects, relations, and inferences available to programmers are mediated by the knowledge representation language.

Issues in Knowledge Representation (iii)

- There are general principles of knowledge organization that apply across a variety of domains and can be directly supported by a representation language.
- Example: class hierarchies are found in both scientific and commonsense classification systems.

1. How may we provide a general mechanism for representing them?
2. How may we represent definitions?
3. Exceptions?
4. When should an intelligent system make default assumptions about missing information and how can it adjust its reasoning should these assumptions prove wrong?
5. How may we best represent time? Causality? Uncertainty? Progress in building intelligent systems depends on discovering the principles of knowledge organization and supporting them through higher-level representational tools.

Associationist Theories of Meaning(ii)

- There are many problems that arise in mapping commonsense reasoning into formal logic.

- For example, it is common to think of the operators ∨ and → as corresponding to the English "or" and "if … then …".

- However, these operators in logic are concerned solely with truth values and ignore the fact that the English "if … then …" suggests specific relationship (often more correlational than causal) between its premises and its conclusion.

Associationist Theories of Meaning(iii)

- For example, the sentence “If a bird is a cardinal then it is red” (associating the bird cardinal with the color red) can be written in predicate calculus:

$$\forall X (cardinal(X) \rightarrow red(X)).$$

This may be changed, through a series of truth-preserving operations, Chapter 2, into the logically equivalent expression

$$\forall X (\neg red(X) \rightarrow \neg cardinal(X)).$$

These two expressions have the same truth value; that is, the second is true if and only if the first is true.

Associationist Theories of Meaning(vi)

- **Associationist theories:**

1. **It defines the meaning of an object in terms of a network of associations with other objects.**
2. **For the associationist, when humans perceive an object, that perception is first mapped into a concept.**
3. **This concept is part of our entire knowledge of the world and is connected through appropriate relationships to other concepts.**
4. **These relationships form an understanding of the properties and behavior of objects such as snow.**

Semantics of Calculus

- **Predicate calculus representation**
  - Formal representation languages
  - Sound and complete inference rules
  - Truth-preserving operations

- **Meaning(line of reasoning) – Semantics**
  - Logical implication is a relationship between truth values:

$$p \rightarrow q$$

- **Associationist theory**
  - Attach semantics to logical symbols and operators

Semantics of Calculus(2)

For Example:

"if a bird is cardinal then it is red"

(associating the bird cardinal with the color red)

can be written in predicate calculus:

$$\forall X((cardinal(X) \rightarrow red(X))$$

$$\forall X(\neg red(X) \rightarrow \neg cardinal(X))$$

These two expressions have the same truth value; that is, the second is true if and only if the first is true.

Semantic Networks(i)

- Using Graphs, by providing a means of explicitly representing relations using arcs and nodes, have proved formalizing associationist theories of knowledge.
- A semantic network represents knowledge as a graph, with the nodes corresponding to facts or concepts and the arcs to relations or associations between concepts. Both nodes and links are generally labeled.
- The term “semantic network” encompasses a family of graph-based representations. These differ chiefly in the names that are allowed for nodes and links and the inferences that may be performed. However, a common set of assumptions and concerns is shared by all network representation languages

Semantic Networks(ii)

- Semantic network developed by Collins and Quillianin, their research on human information storage and response times (Harmon and King 1985).
  - classic representation technique for propositional information
  - Propositions – a form of declarative knowledge, stating facts (true/false)
  - Propositions are called “atoms” – cannot be further subdivided.
- Semantic nets consist of nodes (objects, concepts, situations) and arcs (relationships between them).

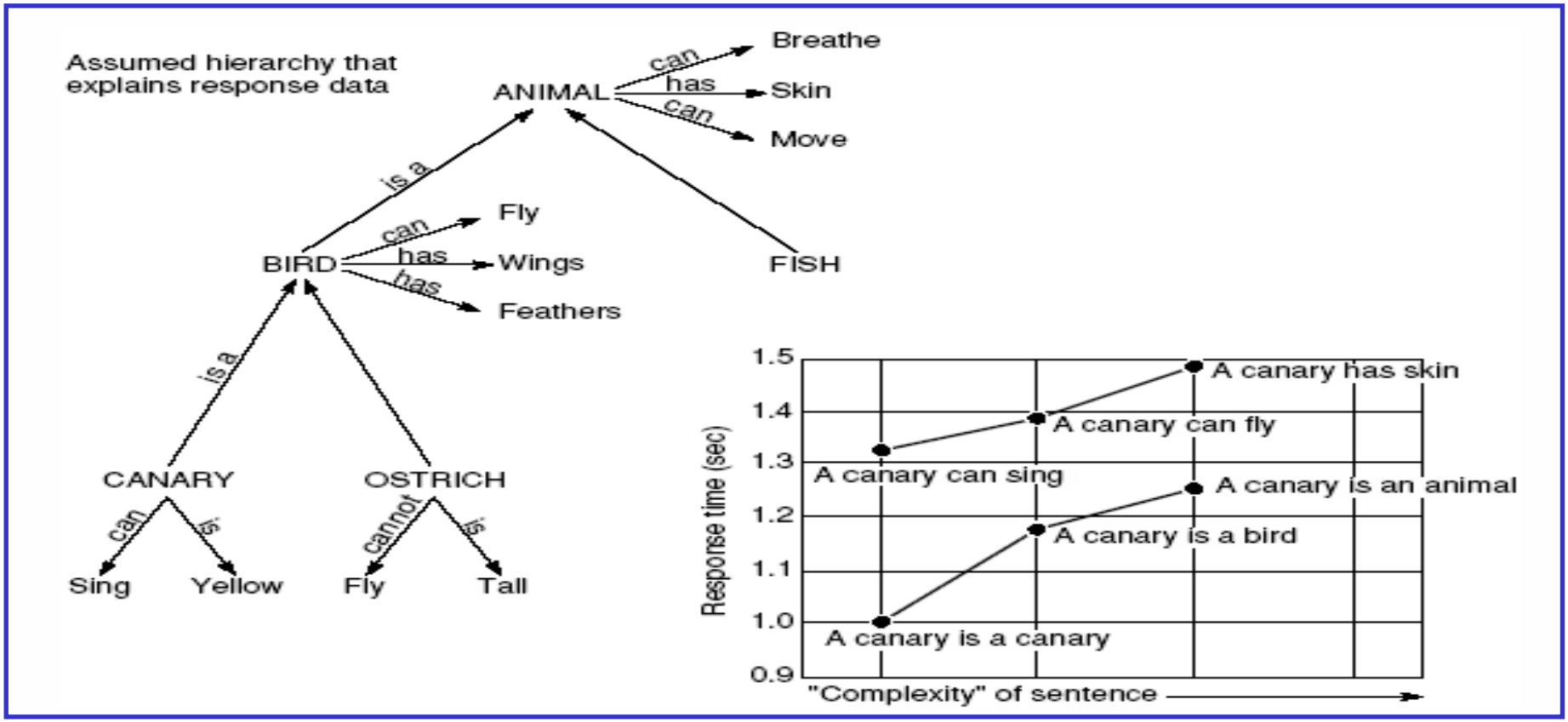
Semantic Networks(iii)

- **Definition**
  - Represent knowledge as a graph
  - Nodes correspond to facts or concepts
  - Arcs correspond to relations or associations between concepts
  - Nodes and arcs are labeled

- **Properties**
  - Labeled arcs and links
  - Inference is to find a path between nodes
  - Implement inheritance
  - Variations – conceptual graphs

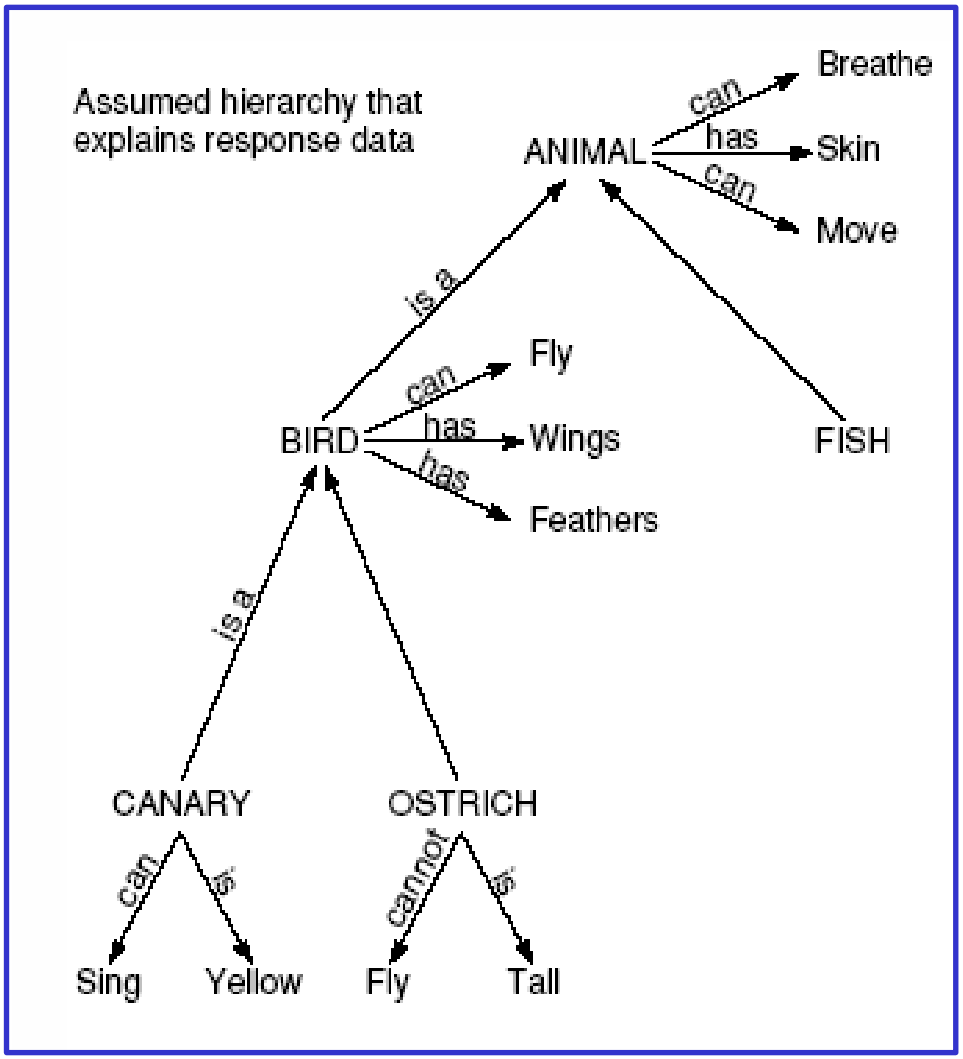
Semantic Networks Example 1(i)

- **Figure 7.1 Semantic network developed by Collins and Quillian in their research on human information storage and response times (Harmon and King 1985).**



Semantic Networks Example 1(ii)

- Different inferences with given questions.
- The structure of this hierarchy was derived from laboratory testing of human subjects.
- The subjects were asked questions about different properties of birds, such as, "Is a canary a bird?" or "Can a canary sing?" or "Can a canary fly?".



Semantic Networks Example 1(iii)

- reaction-time studies indicated that it took longer for subjects to answer “Can a canary fly?” than it did to answer “Can a canary sing?”
- Collins and Quillian explain this difference in response time by arguing that people store information at its most abstract level. Instead of trying to recall that canaries fly, and robins fly, and swallows fly, all stored with the individual bird, humans remember that canaries are birds and that birds have (usually) the property of flying.
- Even more general properties such as eating, breathing, and moving are stored at the “animal” level, and so trying to recall whether a canary can breathe should take longer than recalling whether a canary can fly. This is, of course, because the human must travel further up the hierarchy of memory structures to get the answer.

Semantic Networks Example 1 (iv)

- The fastest recall was for the traits specific to the bird, say, that it can sing or is yellow.
- Exception handling also seemed to be done at the most specific level.
- When subjects were asked whether an ostrich could fly, the answer was produced faster than when they were asked whether an ostrich could breathe. Thus the hierarchy
  Ostrich → bird → animal seems not to be traversed to get the exception information: it is stored directly with ostrich. This knowledge organization has been formalized in inheritance systems.

Inheritance systems

- **Inheritance systems allow us to store information at the highest level of abstraction, which reduces the size of knowledge bases and helps prevent update inconsistencies.**

Example:

- **If we are building a knowledge base about birds, we can define the traits common to all birds, such as flying or having feathers, for the general class bird.**
- **Allow a particular species of bird to inherit these properties.**
- **This reduces the size of the knowledge base by requiring us to define these essential traits only once, rather than requiring their assertion for every individual.**

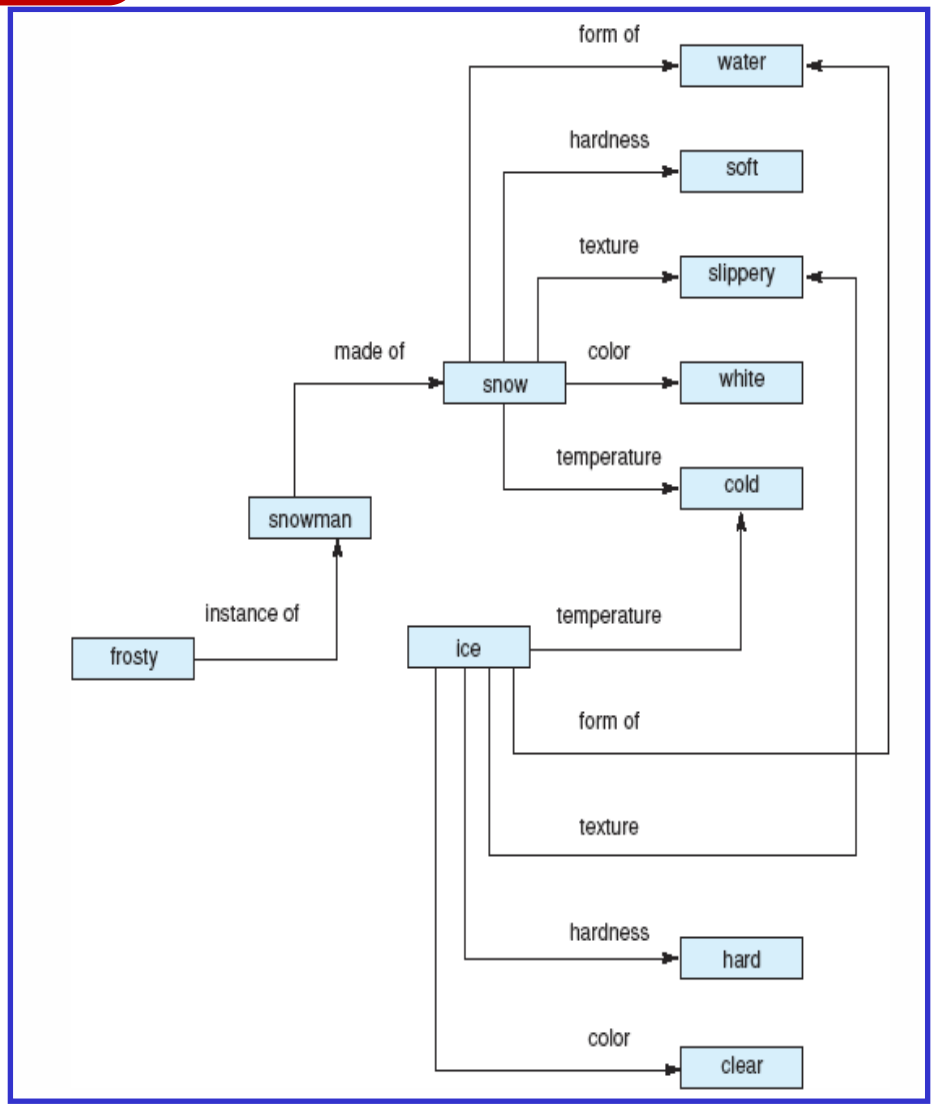
- **Inheritance also helps us to maintain the consistency of the knowledge base when adding new classes and individuals.**

Example:

- **Assume that we are adding the species robin to an existing knowledge base.**
- **When we assert that robin is a subclass of songbird; robin inherits all of the common properties of both songbirds and birds.**
- **It is not up to the programmer to remember (or forget!) to add this information.**

Semantic Networks Example 2

- This network could be used (with appropriate inference rules) to answer a range of questions about snow, ice, and snowmen.
- These inferences are made by following the links to related concepts.
- For example, through experience, we associate the concept snow with other concepts such as cold, white, snowman, slippery, and ice. Our understanding of snow and the truth of statements such as “snow is white” and “the snowman is white” manifests itself out of this network of associations.



Semantic Network in Natural Language Understanding(i)

- In the general case, language understanding requires an understanding of common sense, the ways in which physical objects behave, the interactions that occur between humans, and the ways in which human institutions are organized.

- A natural language program must understand intentions, beliefs, hypothetical reasoning, plans, and goals. Because of these requirements language understanding has always been a driving force for research in knowledge representation.

Semantic Network in Natural Language Understanding(ii)

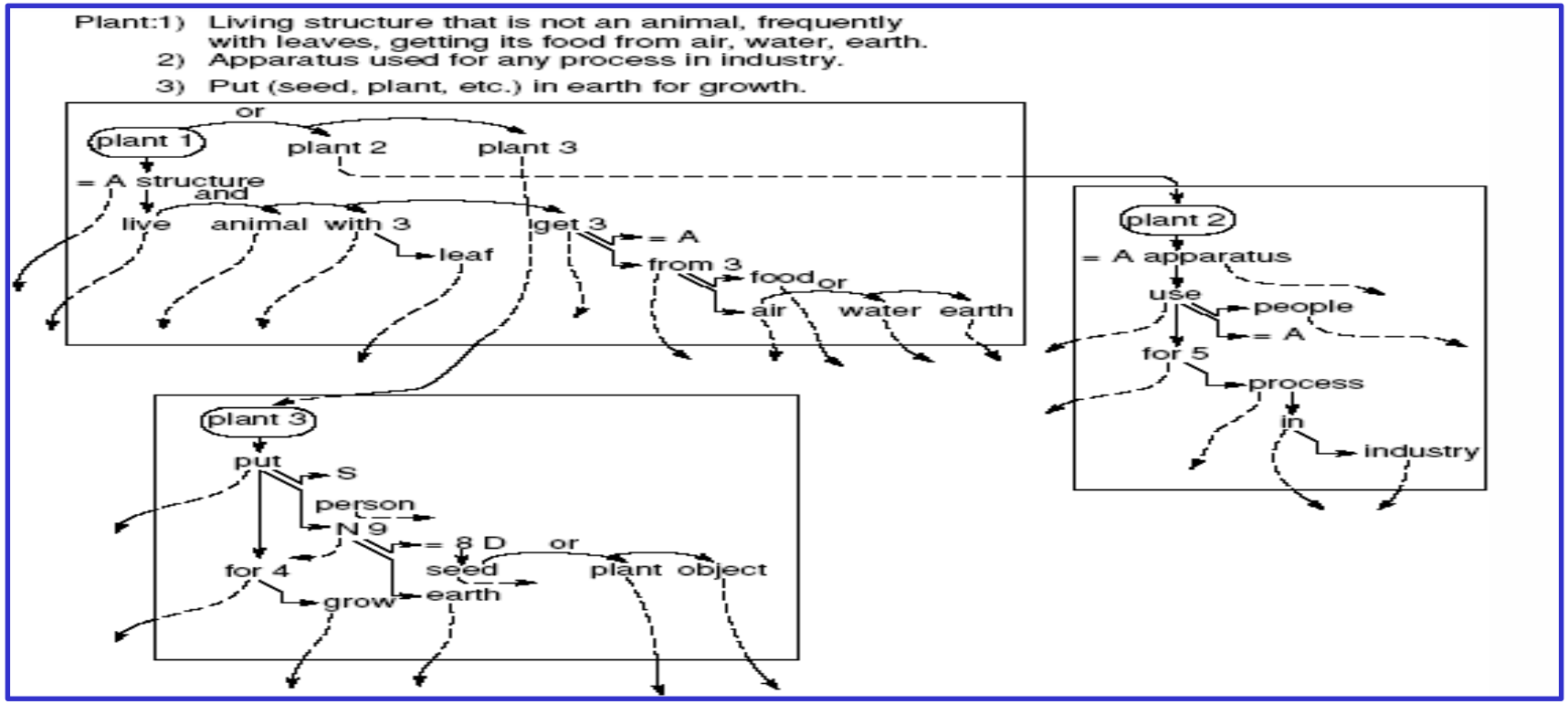
- The first computer implementations of semantic networks were developed in the early1960s for use in machine translation.
- Masterman (1961) defined a set of 100 primitive concept types and used them to define a dictionary of 15,000 concepts.
- Wilks (1972) continued to build on Masterman's work in semantic network-based natural language systems.
- Shapiro's (1971) MIND program was the first implementation of a propositional calculus based semantic network.

Semantic Network in Natural Language Understanding (iii)

- **Quillian's semantic network (1967)**
  - **Influential program**
  - **Define English in much the same way that a dictionary does: a word is defined in terms of other words, and the components of the definition are defined in the same fashion. Rather than formally defining words in terms of basic axioms**
  - **Each definition leads to other definitions in an unstructured and sometimes circular fashion**
  - **When look up a word, traverse the network until we are satisfied that we understand the original word.**
  - **Each node in Quillian's network corresponded to a *word concept*, with associative links to other word concepts that formed its definition.**
  - **The knowledge base was organized into planes, where each plane was a graph that defined a single word.**

Three planes representing three definitions of the word "plant" (i)

- Figure 7.3 below, taken from a paper by Quillian (1967), illustrates three planes that capture three different definitions of the word "plant:" a living organism (plant 1), a place where people work (plant 2), and the act of putting a seed in the ground (plant 3).

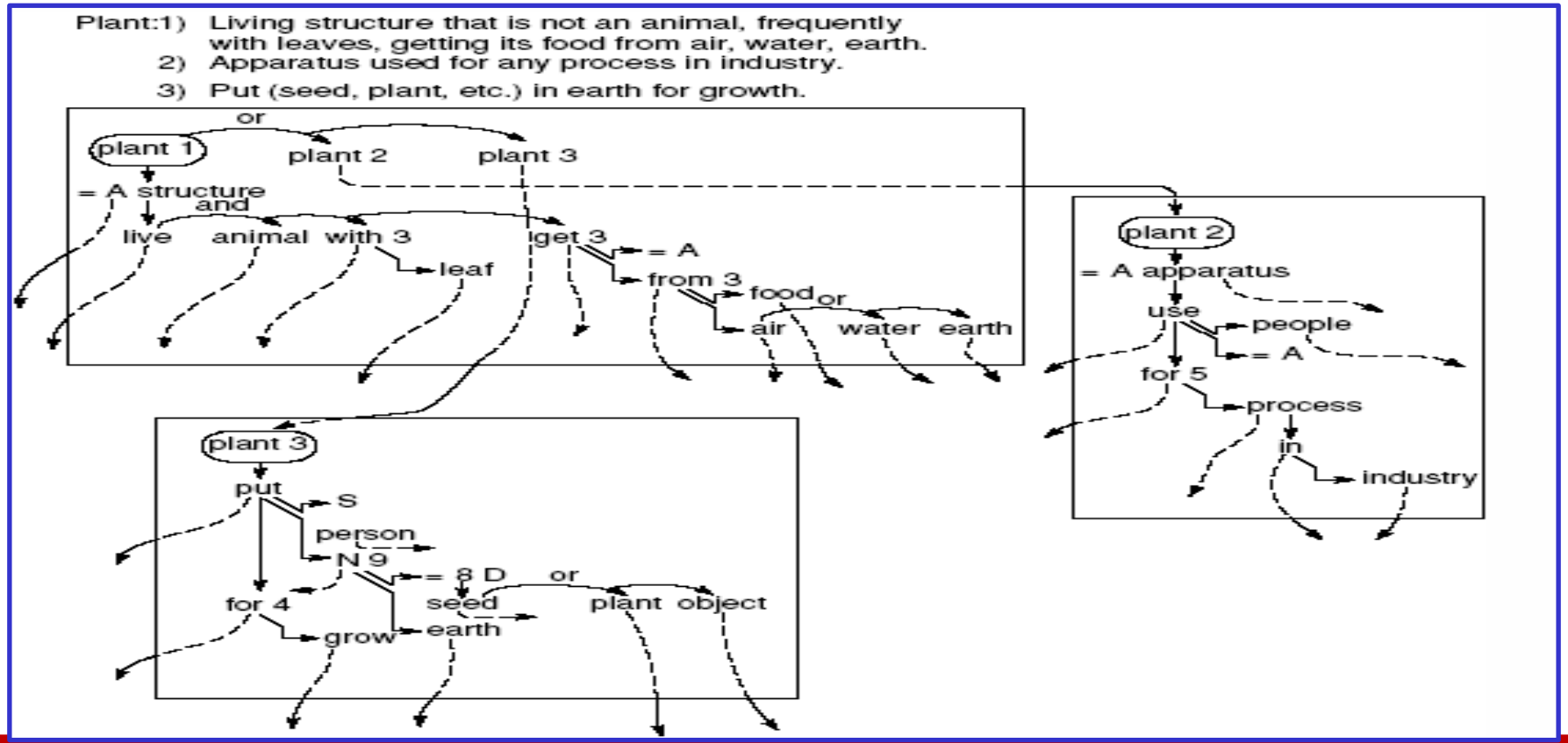


Three planes representing three definitions of the word “plant” (ii)

1 - A living organism (plant 1),

2- A place where people work (plant 2), and

3- The act of putting a seed in the ground (plant 3).

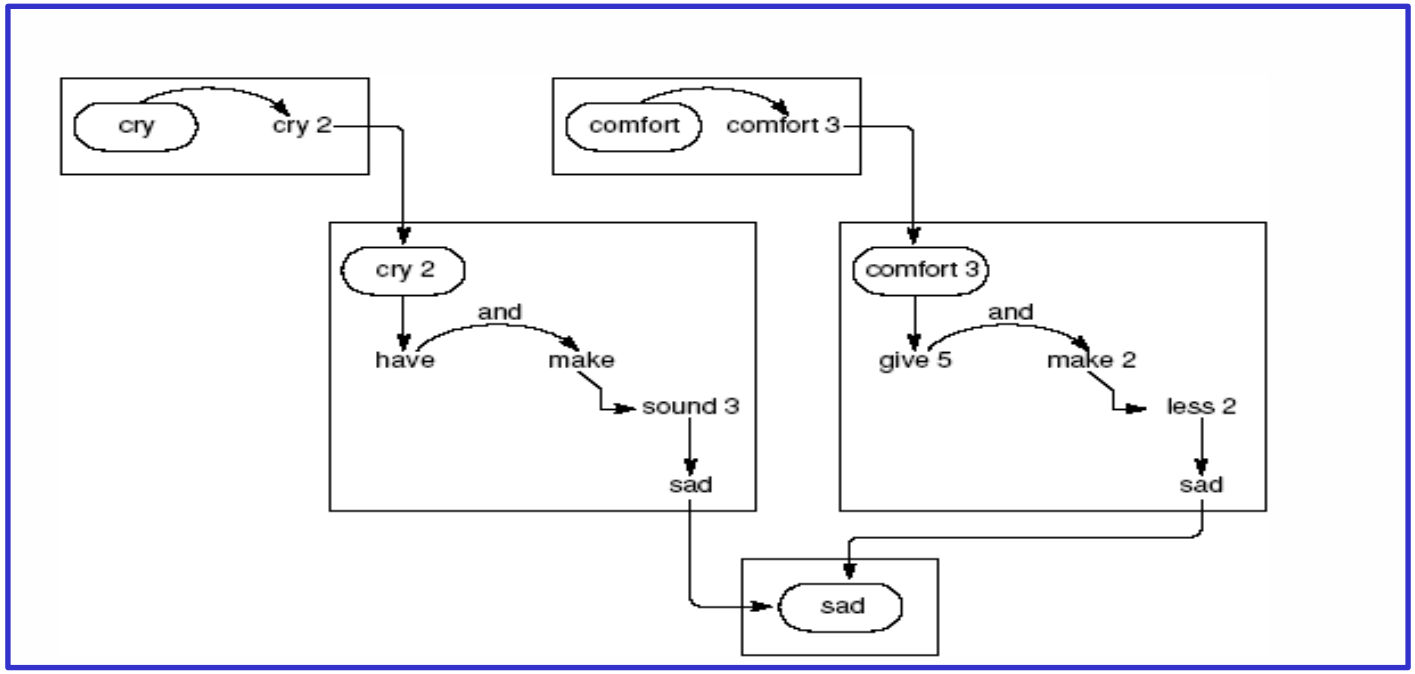


Inferences in Semantic Networks

- The program used this knowledge base to find relationships between pairs of English words.
  - Given two words, it would search the graphs outward from each word in a breadth-first fashion,
  - Searching for a common concept or *intersection node*.
  - *The paths to* this node represented a relationship between the word concepts.

Inferences in Semantic Networks Example

- Find the relationship (intersection path) between “cry” and “comfort”



Using this intersection path, the program was able to conclude:

cry 2 is among other things to make a sad sound.

To comfort 3 can be to make 2 something less sad

Quillian Suggestion

- Quillian (1967) suggested that this approach to semantics might provide a natural language understanding system with the ability to:
  1. Determine the meaning of a body of English text by building up collections of these intersection nodes.
  2. Choose between multiple meanings of words by finding the meanings with the shortest intersection path to other words in the sentence.

     Example: it could select a meaning for "plant" in "Tom went home to water his new plant" based on the intersection of the word concepts "water" and "plant."
  3. Answer a flexible range of queries based on associations between word concepts in the queries and concepts in the system.

Standardization of Network Relationships(i)

- Much of the work in network representations that followed Quillian's focused on defining a richer set of link labels (relationships) that would more fully model the semantics of natural language. By implementing the fundamental semantic relationships of natural language as part of the formalism, rather than as part of the domain knowledge added by the system builder, knowledge bases require less handcrafting and achieve greater generality and consistency.

- Simmons (1973) addressed this need for standard relationships by focusing on the case structure of English verbs. In this verb-oriented approach, based on earlier work by Fillmore (1968), links define the roles played by nouns and noun phrases in the action of the sentence. Case relationships include agent, object, instrument, location, and time.

Standardization of Network Relationships(ii)

- Sentence is represented as a verb node, with various case links to nodes representing other participants in the action.
- This structure is called a case frame.
- In parsing a sentence, the program finds the verb and retrieves the case frame for that verb from its knowledge base.
- It then binds the values of the agent, object, etc., to the appropriate nodes in the case frame.

Example: Using this approach, the sentence “Sarah fixed the chair with glue” might be represented by the network in Figure 7.5.

Standardization of Network Relationships(iii)

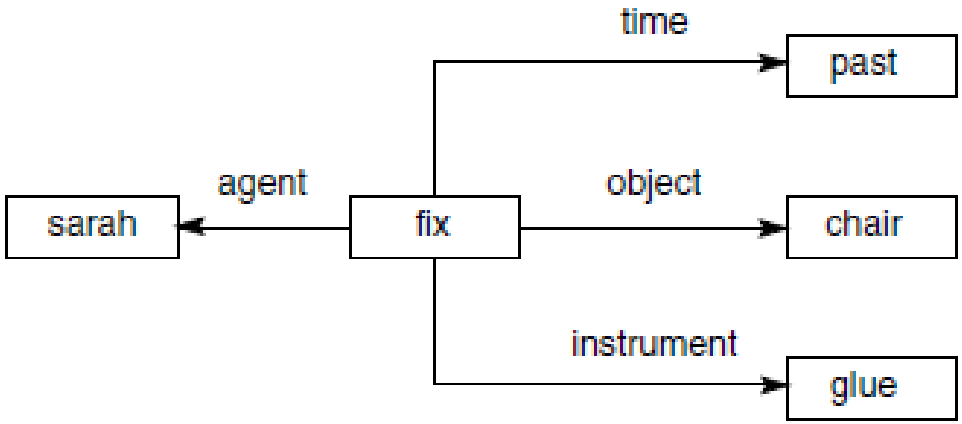


Figure 7.5 Case frame representation of the sentence Sarah fixed the chair with glue.

Conceptual Dependency Theory(i)

- Conceptual dependency theory offers a set of four primitive conceptualizations from which the world of meaning is built. These are equal and independent. They are:

Primitive	Meaning
ACTs	actions
PPs	objects (picture producers)
AAs	modifiers of actions (action aiders)
PAs	modifiers of objects (picture aiders)

Conceptual Dependency Theory(ii)

- For example, all actions are assumed to reduce to one or more of the primitive ACTs. The primitives listed below are taken as the basic components of action, with more specific verbs being formed through their modification and combination.

ACTs	Action
ATRANS	transfer a relationship (give)
PTRANS	transfer physical location of an object (go)
PROPEL	apply physical force to an object (push)
MOVE	move body part by owner (kick)
GRASP	grab an object by an actor (grasp)
INGEST	ingest an object by an animal (eat)
EXPEL	expel from an animal s body (cry)
MTRANS	transfer mental information (tell)
MBUILD	mentally make new information (decide)
CONC	conceptualize or think about an idea (think)
SPEAK	produce sound (say)
ATTEND	focus sense organ (listen)

Conceptual Dependency Theory(iii)

- These primitives are used to define conceptual dependency relationships that describe meaning structures such as case relations or the association of objects and values.
- Conceptual dependency relationships are conceptual syntax rules and constitute a grammar of meaningful semantic relationships.
- These relationships can be used to construct an internal representation of an English sentence. A list of basic conceptual dependencies (Schank and Rieger 1974) appears in Figure 7.6.

Conceptual Dependency Theory(iv)

PP ⇔ ACT	indicates that an actor acts.
PP ⇔ PA	indicates that an object has a certain attribute.
ACT ←(O) PP	indicates the object of an action.
ACT ←(R) PP, PP	indicates the recipient and the donor of an object within an action.
ACT ←(D) PP, PP	indicates the direction of an object within an action.
ACT ←(1) ⇕	indicates the instrumental conceptualization for an action.
X ⇑ Y	indicates that conceptualization X caused conceptualization Y. When written with a C this form denotes that X COULD cause Y.
PP ⇐ PA2, PA1	indicates a state change of an object.
PP1 ← PP2	indicates that PP2 is either PART OF or the POSSESSOR OF PP1.

Figure 7.6 Conceptual dependencies (Schank and Rieger 1974).

Conceptual Dependency Theory(v)

- These capture, their creators feel, the fundamental semantic structures of natural language. For example, the first conceptual dependency in Figure 7.6 describes the relationship between a subject and its verb, and the third describes the verb–object relation. These can be combined to represent a simple transitive sentence such as “John throws the ball” (see Figure 7.7).

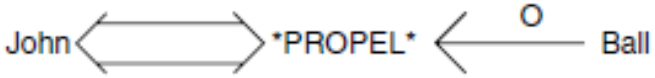


Figure 7.7 Conceptual dependency representation of the sentence John throws the ball.

Scripts

- A **script** is a structured representation describing a stereotyped sequence of events in a particular context.
- The **script** was originally designed by Schank and his research group (Schank and Abelson 1977) as a means of organizing conceptual dependency structures into descriptions of typical situations.
- **Scripts** are used in natural language understanding systems to organize a knowledge base in terms of the situations that the system is to understand.

Script Components

1. Entry conditions or descriptors of the world that must be true for the script to be called.
2. Results or facts that are true once the script has terminated.
3. Props or the “things” that support the content of the script.
4. Roles are actions that the individual participants perform.
5. Scenes are a sequence of what represents a temporal aspect of the script.

A Restaurant Script Example(i)

Script: RESTAURANT			
Track:	Coffee Shop	Entry cond.:	S hungry
Props:	Tables		S has money
	Menu		
	F=Food	Results:	S has less money
	Check		O has more money
	Money		S is not hungry
			S is pleased (optional)
Roles	S=Customer		
	W=Waiter		
	C=Cook		
	M=Cashier		
	O=Owner		

A Restaurant Script Example(ii)

1. **Entry conditions** or descriptors of the world that must be true for the script to be called.
   - In our example script, these include an open restaurant and a hungry customer that has some money.
2. **Results** or facts that are true once the script has terminated;
   - For example, the customer is full and poorer, the restaurant owner has more money
3. **Props** or the "**things**" that support the content of the script.
   - These will include tables, waiters, and menus. The set of props supports reasonable default assumptions about the situation: a restaurant is assumed to have tables and chairs unless stated otherwise.

A Restaurant Script Example(iii)

4. **Roles** are the actions that the individual participants perform. The waiter takes orders, delivers food, and presents the bill. The customer orders, eats, and pays.

5. **Scenes**. Schank breaks the script into a sequence of scenes each of which presents a temporal aspect of the script.
   - In the restaurant there is entering, ordering, eating, etc.

A Restaurant Script Example(iv)

Script: RESTAURANT (Cont'd)

Scene 1: Entering

S PTRANS S into restaurant
S ATTEND eyes to tables
S MBUILD where to sit
S PTRANS S to table
S MOVE S to sitting position

Scene 2: Order

(Menu on table) (W brings menu)
S PTRANS menu to S

(S asks for menu)
S MTRANS signal to W
W PTRANS W to table
S MTRANS 'need menu' to W
W PTRANS W to menu

W PTRANS W to table
W ATRANS menu to S

S MTRANS food list to CP (S)
* S MBUILD choice of F
S MTRANS signal to W
W PTRANS W to table
S MTRANS 'I want F' to W

W PTRANS W to C
W MTRANS (ATRANS F) to C

C MTRANS 'no F' to W
W PTRANS W to S
W MTRANS 'no F' to S
(go back to *) or
(go to Scene 4 at no pay path)

C DO (prepare F script)
to Scene 3

A Restaurant Script Example(v)

Script: RESTAURANT (Cont'd)	
Scene 3: Eating	Scene 4: Exiting
C ATRANS F to W W ATRANS F to S S INGEST F (Option: Return to Scene 2 to order more; otherwise, go to Scene 4)	W MOVE (write check) W PTRANS W to S W ATRANS check to S S ATRANS tip to S S PTRANS S to M S ATRANS money to M S PTRANS S to out of restaurant

Different scripts for different types of restaurants.

Script leaves details open.

Script greatly reduces search in a particular context.

A Restaurant Script Example(vi)

Script: RESTAURANT
Track: Coffee Shop
Props: Tables
Menu
F = Food
Check
Money

Roles: S = Customer
W = Waiter
C = Cook
M = Cashier
O = Owner

Entry conditions: S is hungry.
S has money.

Results: S has less money
O has more money
S is not hungry
S is pleased (optional)

Scene 1: Entering

S PTRANS S into restaurant
S ATTEND eyes to tables
S MBUILD where to sit
S PTRANS S to table
S MOVE S to sitting position

Scene 2: Ordering

(Menu on table) (W brings menu)
S PTRANS menu to S

(S asks for menu)
S MTRANS signal to W
W PTRANS W to table
S MTRANS 'need menu' to W
W PTRANS W to menu

W PTRANS W to table
W ATRANS menu to S

S MTRANS food list to CP (S)
*S MBUILD choice of F
S MTRANS signal to W
W PTRANS W to table
S MTRANS I want F to W

W PTRANS W to C
W MTRANS (ATRANS F) to C

C MTRANS 'no F' to W
W PTRANS W to S
W MTRANS 'no F' to S
(go back to *) or
(go to Scene 4 at no pay path)

C DO (prepare F script)
to Scene 3

Scene 3: Eating

C ATRANS F to W
W ATRANS F to S
S INGEST F

(Option: Return to Scene 2 to order more;
otherwise, go to Scene 4)

Scene 4: Exiting

S MTRANS to W
(W ATRANS check to S)

W MOVE (write check)
W PTRANS W to S
W ATRANS check to S
S ATRANS tip to W
S PTRANS S to M
S ATRANS money to M
S PTRANS S to out of restaurant

(No pay path)

Example 1 on Restaurant Script

- John went to restaurant last night. He ordered steak. When he paid he noticed he was running out of money. He hurried home since it had started to rain.

Using the script, the system can correctly answer questions:

- Did john eat dinner last night?
- Did John use cash or a card?
- How could John get a menu?
- What did John buy?

Example 2 on Restaurant Script

- Amy Sue went out to lunch. She sat at a table and called a waitress, who brought her a menu. She ordered a sandwich.

Using the script, the system can correctly answer questions:

- Why did the waiter bring any Sue a menu?
- Was Amy Sue in a restaurant ?
- Who paid?
- Who was the “she” who ordered the sandwich?

Frames

- Capture the implicit connections of information from the explicitly organized data structure
- Support the organization of knowledge into more complex units that reflect the organization of objects in the domain.
- Similar to classes in Object-oriented
- Proposed by Minsky in 1975
  - Here is the essence of the frame theory: When one encounters a new situation (or makes a substantial change in one's view of a problem) one selects from memory a structure called a "frame". This is a remembered framework to be adapted to fit reality by changing details as necessary.

Frame Example (Hotel Room) (i)

- The hotel room and its components can be described by a number of individual frames.
- In addition to the bed,
- A frame could represent a chair: expected height is 20 to 40 cm, number of legs is 4, a default value, is designed for sitting.
- A further frame represents the hotel telephone: this is a specialization of a regular phone except that billing is through the room, there is a special hotel operator (default), and a person is able to use the hotel phone to get meals served in the room, make outside calls, and to receive other services.

Frame Example (Hotel Room) (ii)

Each individual frame may be seen as a data structure, similar in many respects to the traditional "record", that contains information relevant to stereotyped entities.

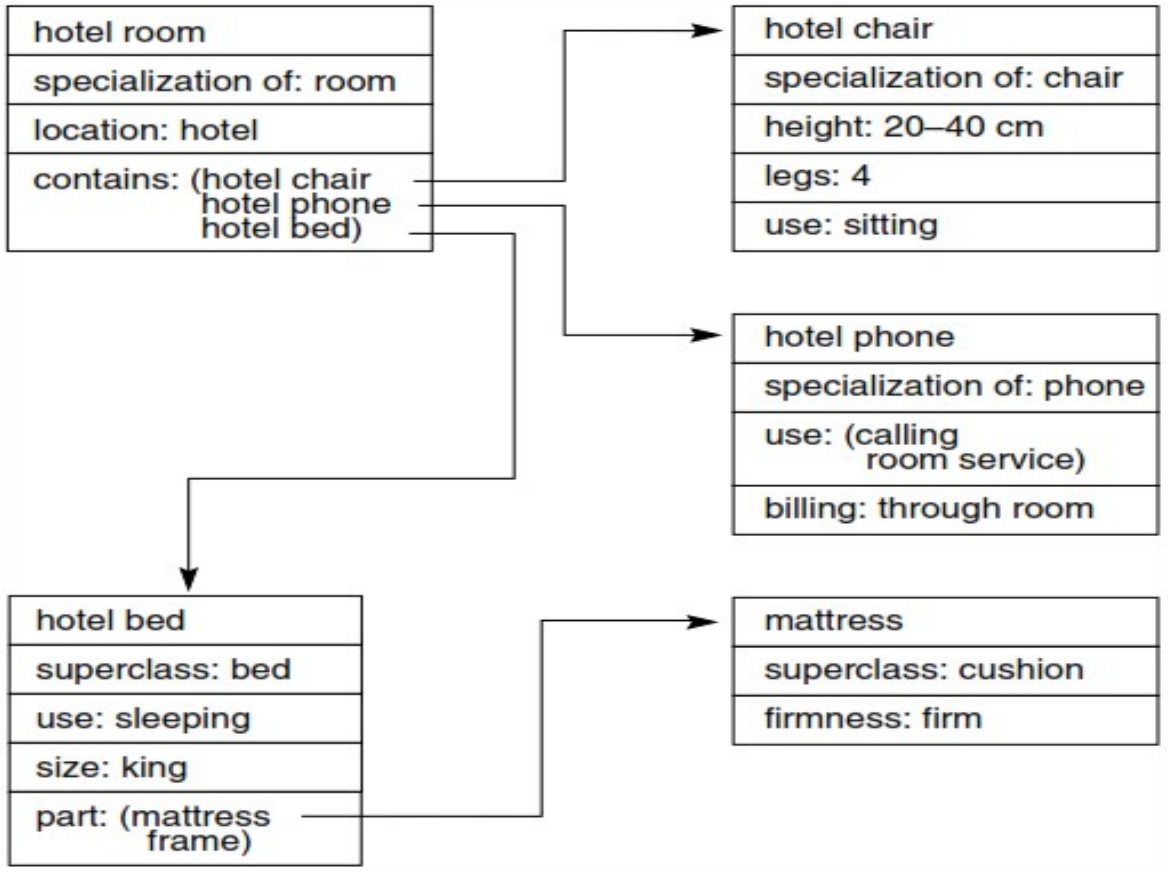


Figure 7.12 Part of a frame description of a hotel room. Specialization indicates a pointer to a superclass.

Frame Slots(i)

- A frame is a set of slots (similar to a set of fields in a class in OO)
- The slots in the frame contain information such as:

1. Frame identification information.
2. Relationship of this frame to other frames.

   The “hotel phone” might be a special instance of “phone,” which might be an instance of a “communication device.”
3. Descriptors of requirements for a frame.

   A chair, for instance, has its seat between 20 and 40 cm from the floor, its back higher than 60 cm, etc. These requirements may be used to determine when new objects fit the stereotype defined by the frame.

Frame Slots(ii)

4. **Procedural information on use of the structure described.**

   An important feature of frames is the ability to attach procedural instructions to a slot.

5. **Frame default information.**

   These are slot values that are taken to be true when no evidence to the contrary has been found. For instance, chairs have four legs, telephones are pushbutton, or hotel beds are made by the staff.

6. **New instance information.**

   Many frame slots may be left unspecified until given a value for a particular instance or when they are needed for some aspect of problem solving.

   For example,

   the color of the bedspread may be left unspecified.

Why Frames?

- Quotes from “A Framework for Representing Knowledge” [Minsky81]
- “It seems to me that the ingredients of most theories have been on the whole too minute, local, and unstructured to account common-sense thought”
- “When one encounters a new situation, one selects from memory

Why Frames?

- A structure called a frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.

- A frame is a data-structure for representing a stereotyped situation, like going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed."

Frames – What and Why?

[Minsky, 1981]:

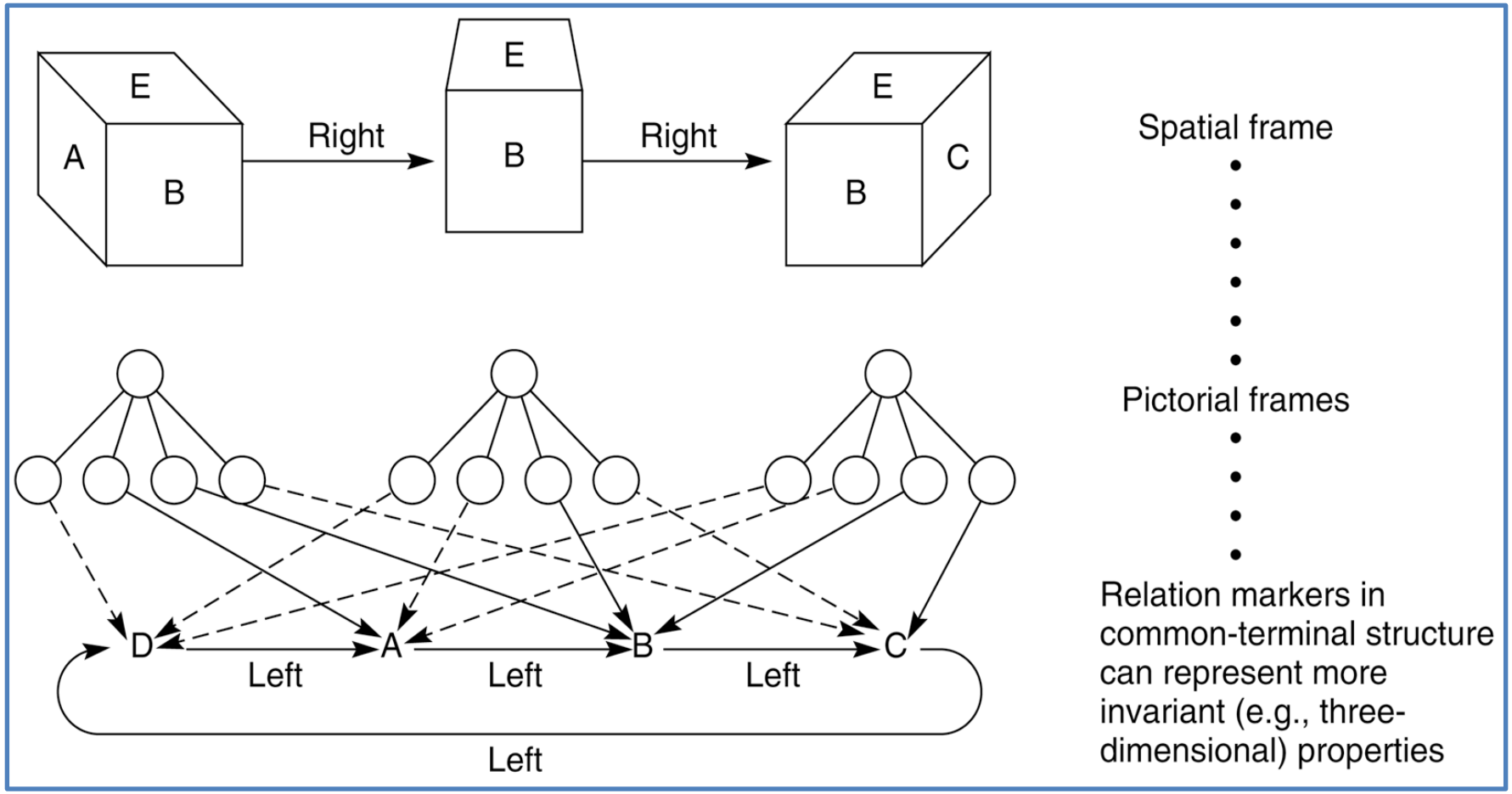
A Frame is a collection of questions to be asked about a hypothetical situation: it specifies issues to be raised and methods to be used in dealing with them.

To understand a situation, questions like:

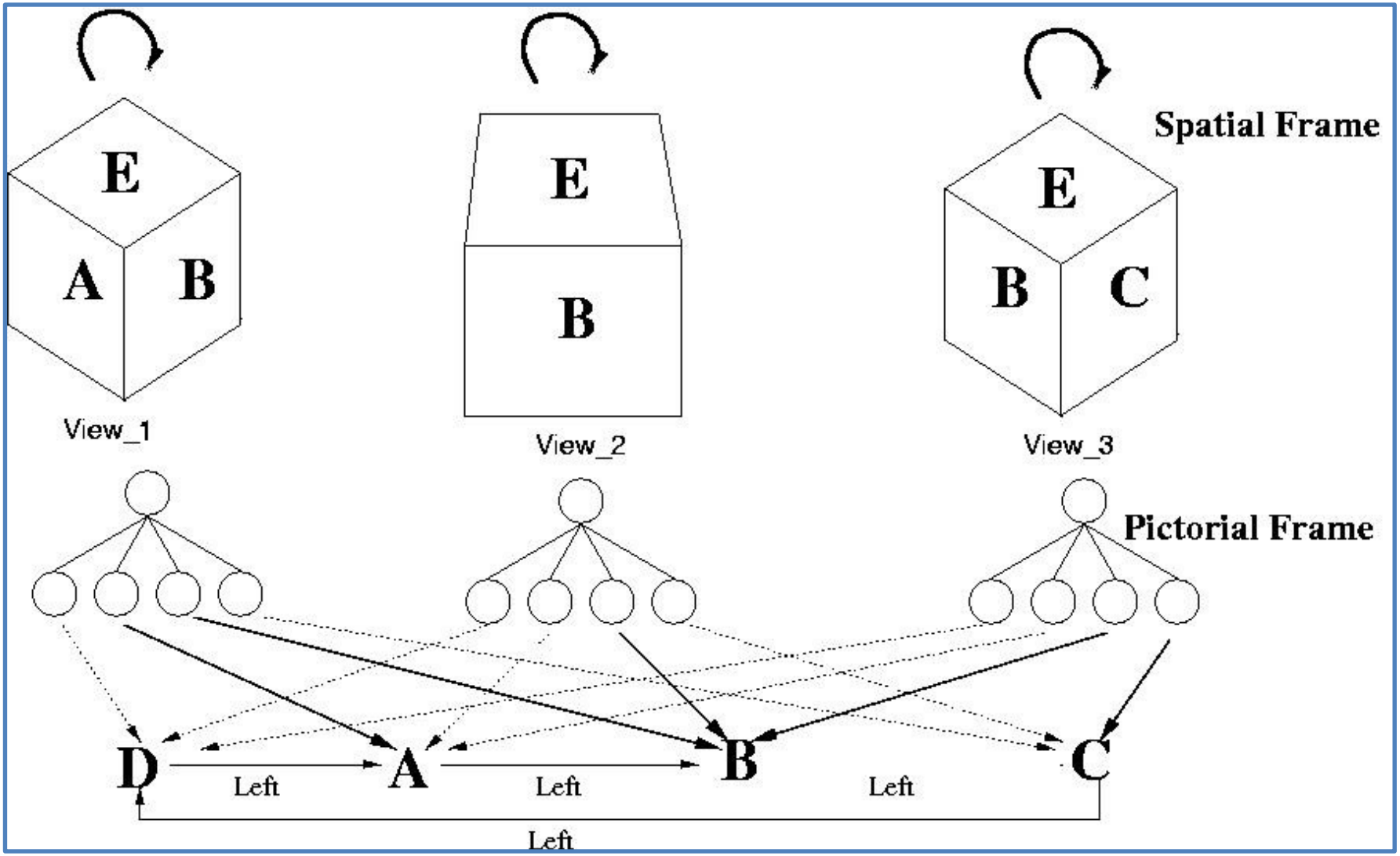
- What caused it (agent)?
- What was the purpose (intention)?
- What are the consequences (effects)?
- Whom does it affect (recipient)?
- How is it done (instruments)

Frame Cube Example(i)

Spatial frame for viewing a cube

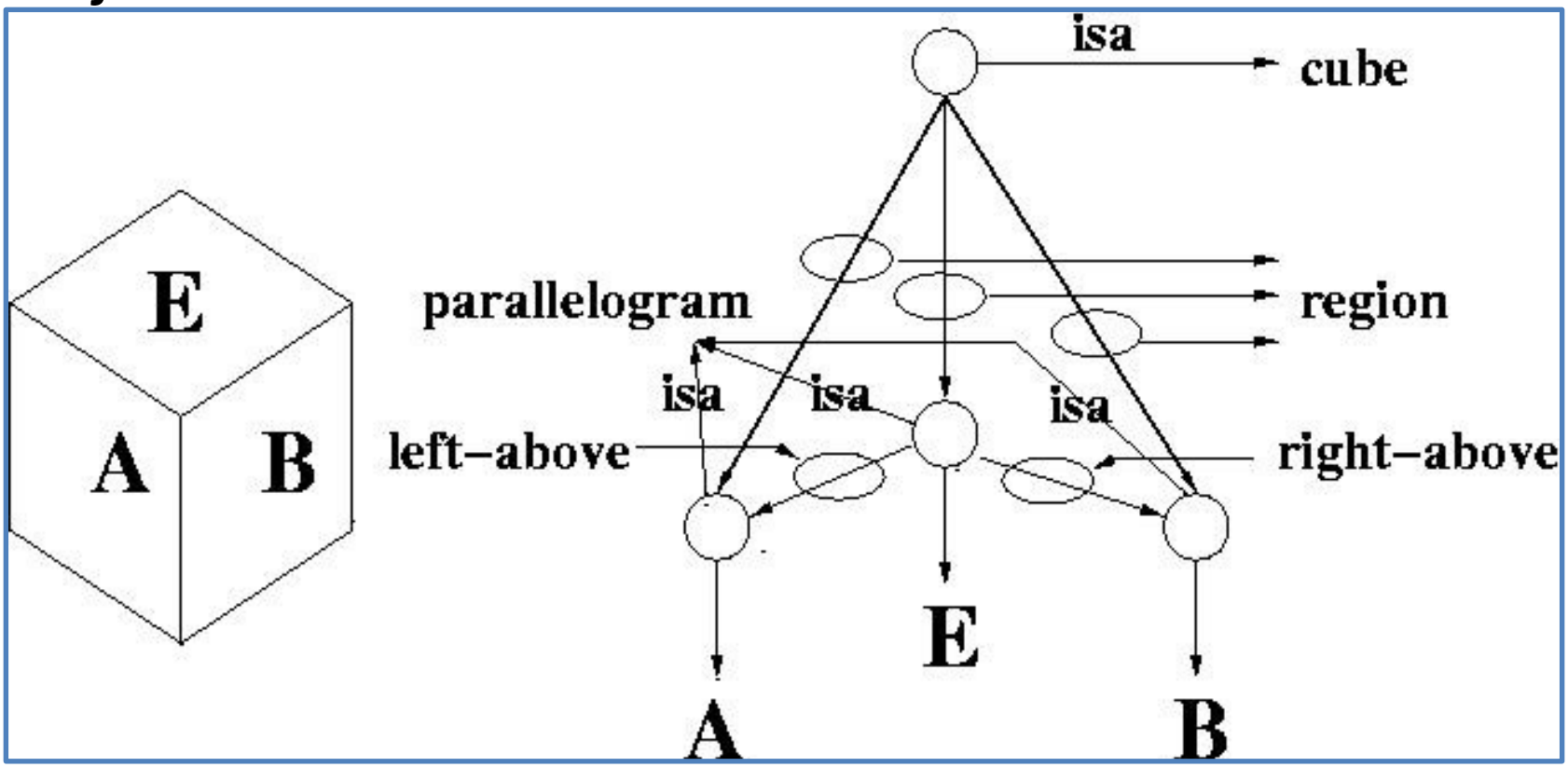


Different Views of a Cube (ii)



Frame Cube Example(iii)

- Consider conception object like a view of a cube as a composed object with relations.



Views of a Cube Tabular(iv)

- Part of a tabular representation of the frame for one view

View-of-a-Cube		
Slot	Filler	Constraint
Name	View_1	
region-of	A	parallelogram & visible
region-of	B	parallelogram & visible
region-of	C	parallelogram & invisible
region-of	D	parallelogram & invisible
region-of	E	parallelogram(E) &visible & left-above(E,A) & right-above(E,B)

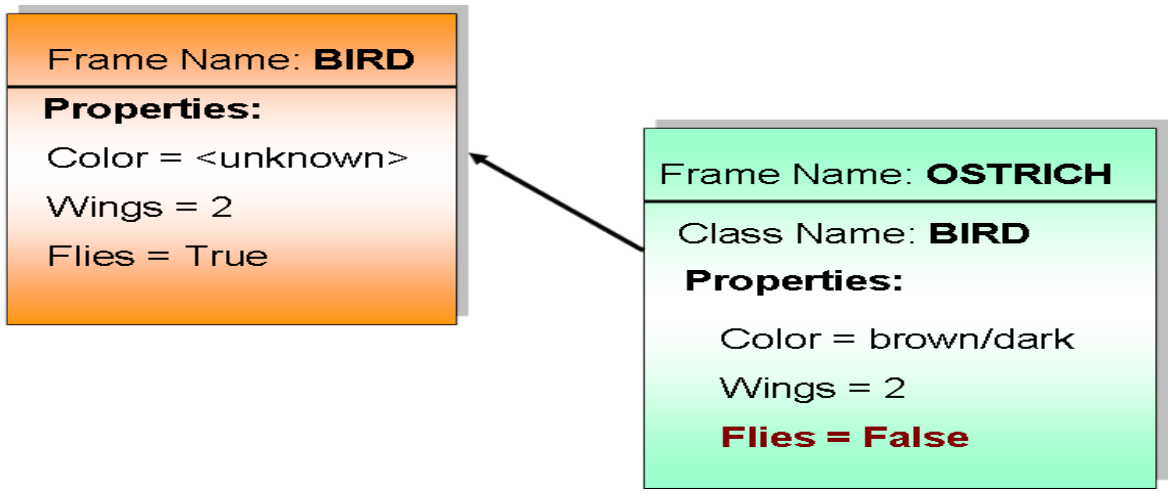
- **In the following 'Frame' example, what piece of knowledge can be inferred?**

  A. All birds with wings fly

  B. Some birds have two wings

  C. Some birds with wings do not fly

  D. All birds are brown or dark in color



Conceptual Graphs: a Network Language

- Graphs
- Introduction to Conceptual Graphs
- Types, Individuals, and Names
- The Type Hierarchy
- Generalization and Specialization
- Propositional Nodes
- Conceptual Graphs and Logic

Conceptual Graphs: a Network Language

- Following on the early research work in AI that developed representational schemes a number of network languages were created to model the semantics of natural language and other domains.

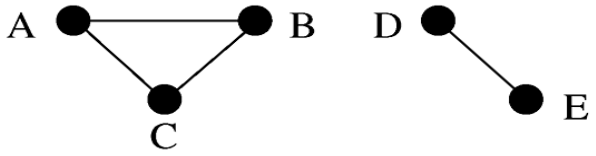
- In this section, we examine a particular formalism in detail, to show how, in this situation, the problems of representing meaning were addressed. *John Sowa's conceptual graphs (Sowa 1984) is an example of a network representation language*.

- We define the rules for forming and manipulating conceptual graphs and the conventions for representing classes, individuals, and relationships.

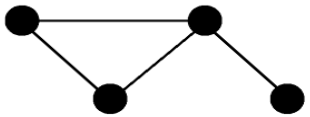
Graphs Review

- Graphs are sometimes called a network or net.
- A graph can have zero or more links between nodes – there is no distinction between parent and child.
- Sometimes links have weights – weighted graph; or, arrows – directed graph.
- Simple graphs have no loops – links that come back onto the node itself.
- A circuit (cycle) is a path through the graph beginning and ending with the same node.
- Acyclic graphs have no cycles.
- Connected graphs have links to all the nodes.
- Digraphs are graphs with directed links.
- Lattice is a directed acyclic graph.

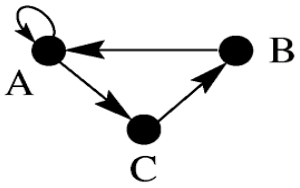
Simple Graphs



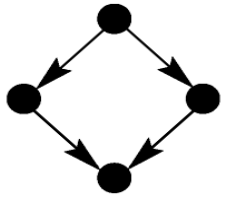
(a) A nonconnected graph



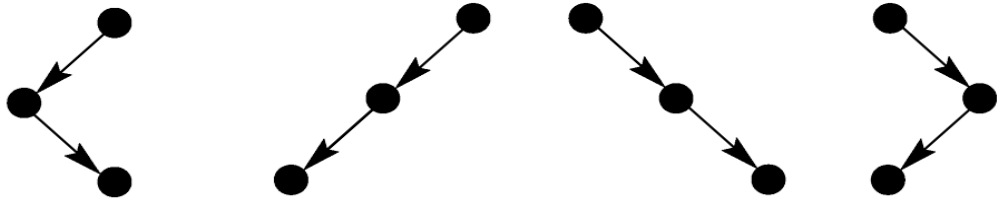
(b) A connected graph



(c) A digraph with a self-loop and circuit



(d) A lattice



(e) Degenerate binary trees of three nodes

Introduction to Conceptual Graphs(i)

Conceptual graph

- A finite, connected, bipartite graph
- No arc labels , (why ?).

  Conceptual graphs do not use labeled arcs; instead the conceptual relation nodes represent relations between concepts. Because conceptual graphs are bipartite, concepts only have arcs to relations, and vice versa.

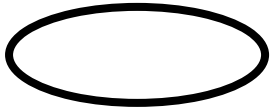
Introduction to Conceptual Graphs(ii)

- Nodes

1. Concept nodes – box nodes

   - Concrete concepts: cat, telephone, classroom, restaurant
     (are characterized by our ability to form an image of them in our minds)
   - Abstract objects: love, beauty, loyalty
     (that do not correspond to images in our minds.)

2. Conceptual relation nodes – ellipse nodes



   - Relations involving one or more concepts
   - Relation Arity are the number of box nodes linked to

Introduction to Conceptual Graphs(iii)

- **Conceptual relation nodes** indicate a relation involving one or more concepts.
- **One advantage of formulating conceptual graphs as bipartite graphs rather than using labeled arcs is that it simplifies the representation of relations of any arity.**
- **A relation of arity n is represented by a conceptual relation node having n arcs. (as shown in Figure 7.14.).**
- **Each conceptual graph represents a single proposition. A typical knowledge base will contain a number of such graphs.**
- **Graphs may be arbitrarily complex but must be finite.**

Conceptual Graphs Example(i)

- For example, one graph in Figure 7.14 represents the proposition “A dog has a color of brown”.

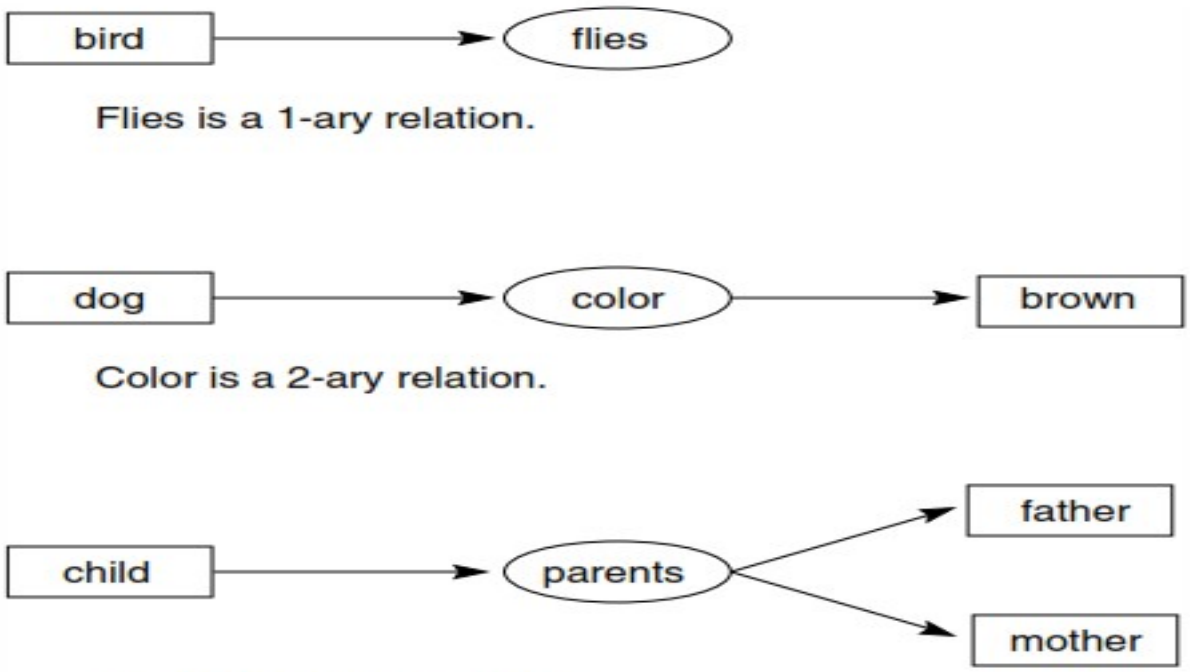


Figure 7.14 Conceptual relations of different arities.

Conceptual Graphs Example(ii)

- For example, one graph in Figure 7.15 is a graph of somewhat greater complexity that represents the sentence

  "Mary gave John the book".

- This graph uses conceptual relations to represent the cases of the verb "to give" and indicates the way in which conceptual graphs are used to model the semantics of natural language.

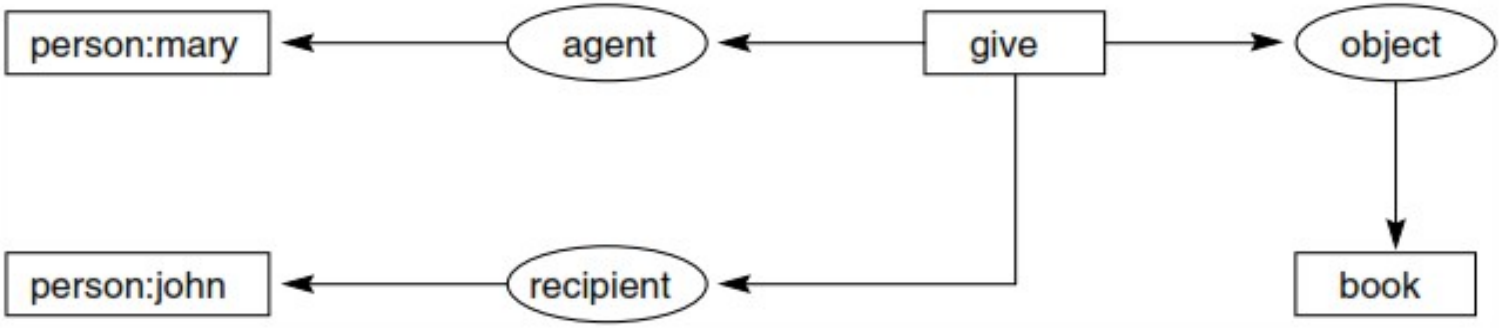


Figure 7.15 Graph of Mary gave John the book.

Types, Individuals, and Names(i)

- In conceptual graphs, every concept is a unique individual of a particular type.
- Each concept box is labeled with a type label, which indicates the class or type of individual represented by that node.

For example:

- A node labeled dog represents some individual of that type.
- Types are organized into a hierarchy. The type dog is a subtype of carnivore, which is a subtype of mammal, etc.

Types, Individuals, and Names(ii)

- Boxes with the same type label represent concepts of the same type; however, these boxes may or may not represent the same individual concept
- Each concept box is labeled with the names of the type and the individual.
- The type and individual labels are separated by a colon, ":".

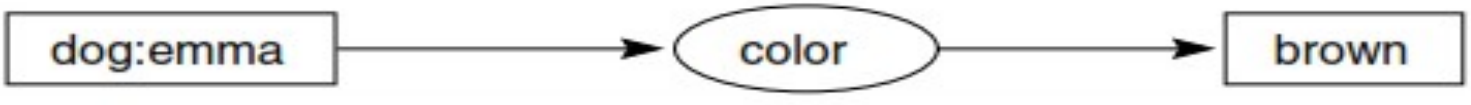


Figure 7.16 Conceptual graph indicating that the dog named emma is brown.

Types, Individuals, and Names(iii)

- Type
  - A class, a concept
  - Types are organized into hierarchy
- Individual -- Concrete entity
- Name – Identifier of type and individual
  - Conceptual Graph
    - Concept box with type label indicating the class or type of individual represented by a node
    - Label consists of (type, :, and individual)
    - Unnamed individual labeled as marker: #<number>
    - Marker can separate an individual from name

boy: Ali

boy: #12354

Types, Individuals, and Names Example 1

Figure 7.17 Conceptual graph indicating that a particular (but unnamed) dog is brown.

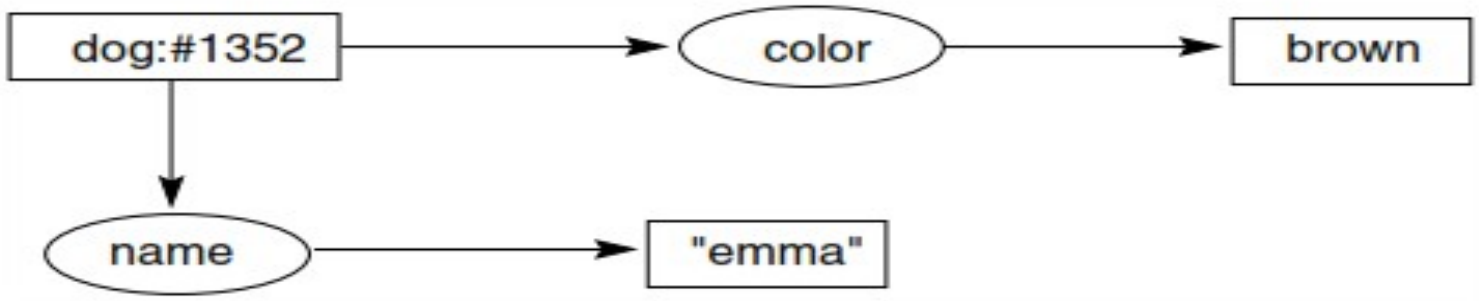


Figure 7.18 Conceptual graph indicating that a dog named emma is brown.

Types, Individuals, and Names Example 2

- For example:

"Her name was McGill, and she called herself Lil, but everyone knew her as Nancy"

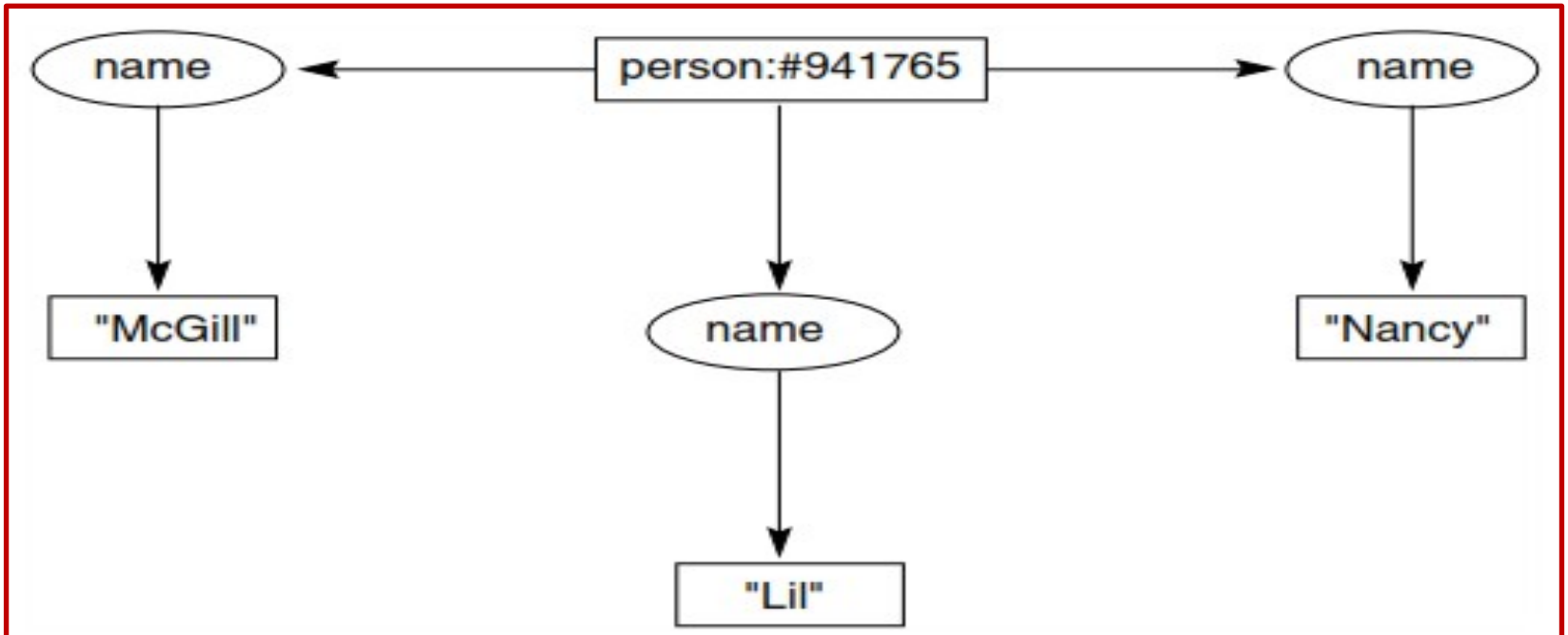


Figure 7.19 Conceptual graph of a person with three names.

Types, Individuals, and Names(iv)

- As an alternative to indicating an individual by its marker or name,
- we can also use the generic marker * to indicate an unspecified individual.
- This is often omitted from concept labels; a node given just a type label, dog, is equivalent to a node labeled dog:*.
- In addition to the generic marker, conceptual graphs allow the use of named variables.
- These are represented by an asterisk followed by the variable name (e.g.,*X or *foo).
- This is useful if two separate nodes are to indicate the same, but unspecified, individual.

Types, Individuals, and Names Example 3

- For example:

"The dog scratches its ear with its paw".

Although we do not know which dog is scratching its ear, the variable *X indicates that the paw and the ear belong to the same dog that is doing the scratching.

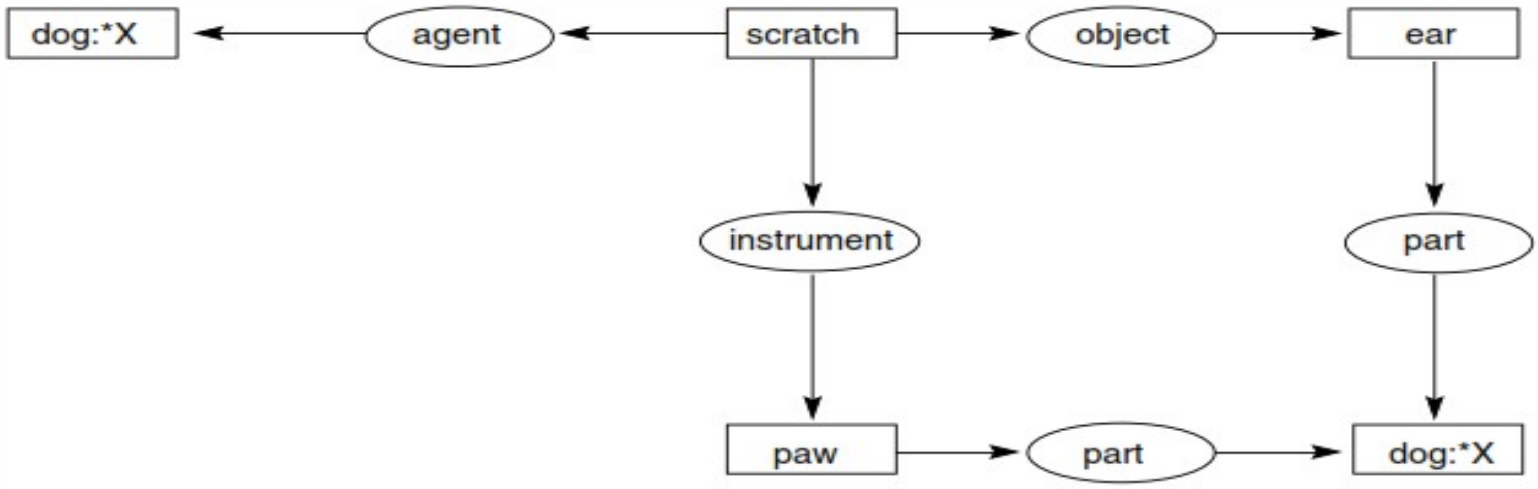


Figure 7.20 Conceptual graph of the sentence The dog scratches its ear with its paw.

The Type Hierarchy(i)

- The type hierarchy is a partial ordering on the set of types , indicated by the symbol ≤ .
- Represent inheritance relationship between types (sub and super)
- Type hierarchy forms a lattice.
- subtype and supertype

  If s and t are types and t ≤ s, then

  1- t is said to be a subtype of s and

  2- s is said to be a supertype of t.

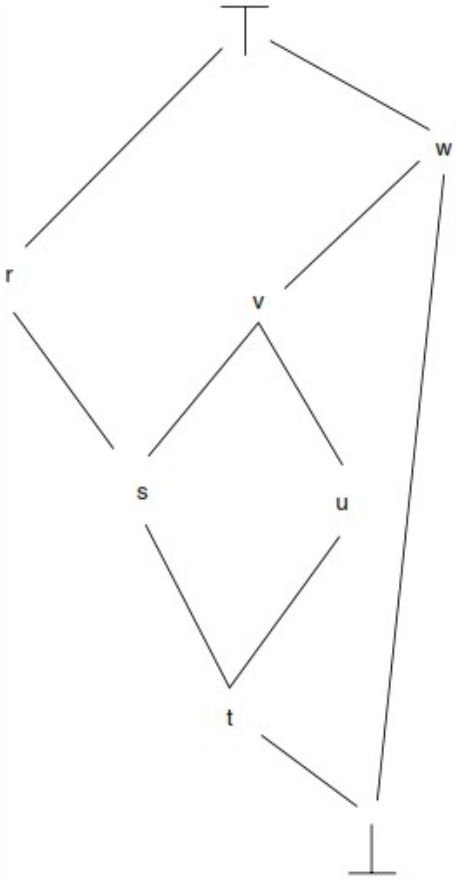


Figure 7.21 A type lattice illustrating subtypes, supertypes, the universal type, and the absurd type. Arcs represent the relationship.

The Type Hierarchy(ii)

- Common subtype
  - If s, t and u are types, with t ≤ s and t ≤ u, then t is a common subtype of s and u

Maximum common subtype:

- If t is a common subtype of s and u, and for any common subtype w of s and u, t ≤ w

- Common supertype
  - If s, u and v are types, with s ≤ v and u ≤ v, then v is a common supertype of s and u

Minimum common supertype:

- If w is a common supertype of s and u, and for any common supertype v of s and u, v ≤ w

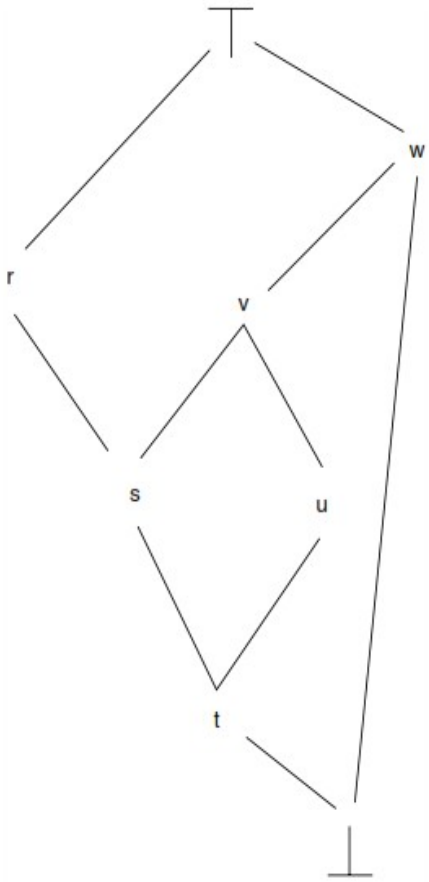
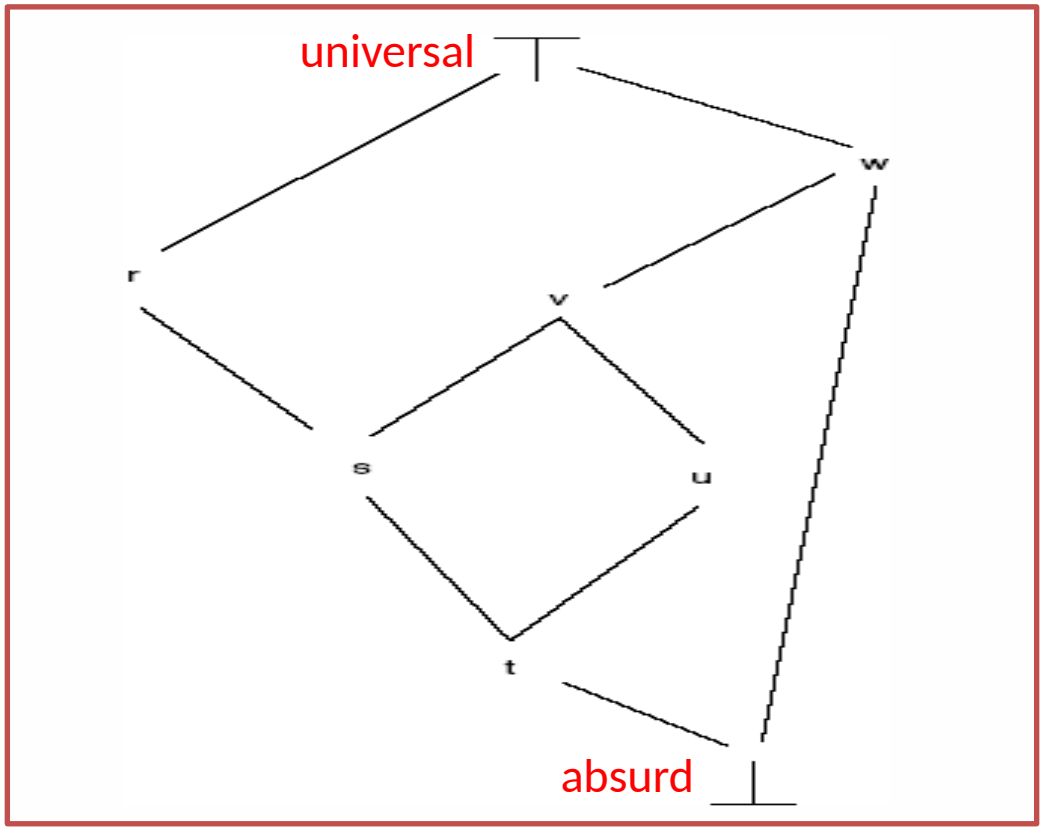


Figure 7.21 A type lattice illustrating subtypes, supertypes, the universal type, and the absurd type. Arcs represent the relationship.

The Type Hierarchy(iii)

- A type lattice Illustrating subtypes, supertypes, the universal type (⊤), and the absurd type(⊥).
- Arcs represent the relationship.



Generalization and Specialization(i)

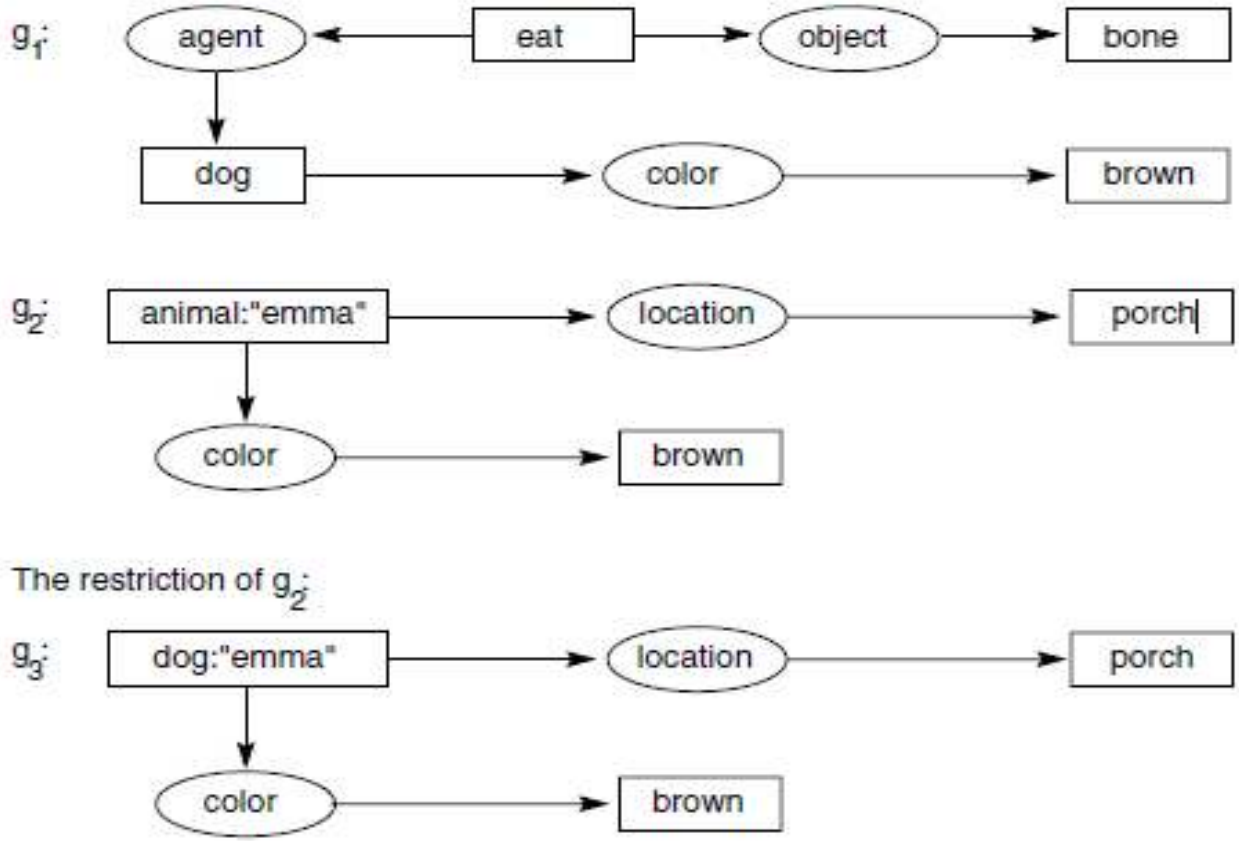
- The theory of conceptual graphs includes a number of operations that create new graphs from existing graphs.
- These allow for the generation of a new graph by either specializing or generalizing an existing graph, operations that are important for representing the semantics of natural language.
- The four operations are :
  1. Copy
  2. Restrict
  3. Join
  4. Simplify.

Generalization and Specialization(ii)

1. The copy rule allows us to form a new graph, g, that is the exact copy of g1.
2. Restrict allows concept nodes in a graph to be replaced by a node representing their specialization.
   - There are two cases:
     1. If a concept is labeled with a generic marker, the generic marker may be replaced by an individual marker.
     2. A type label may be replaced by one of its subtypes, if this is consistent with the referent of the concept. In Figure 7.22 we can replace animal with dog.

Restriction Example

- For example we can replace animal with dog.

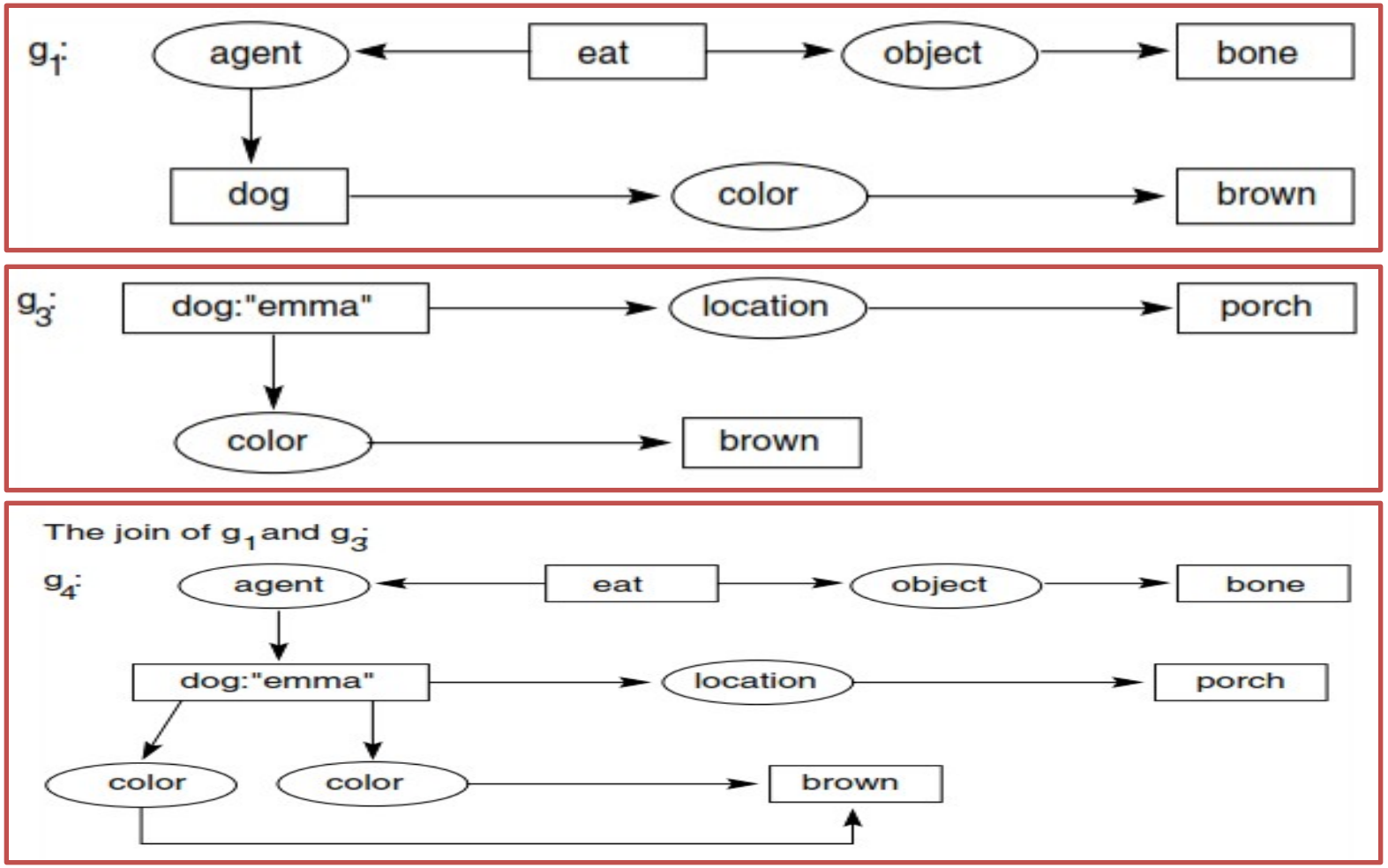


Generalization and Specialization(iii)

- The join rule lets us combine two graphs into a single graph.
- If there is a concept node c1 in the graph s1 that is identical to a concept node c2 in s2, then we can form a new graph by deleting c2 and linking all of the relations incident on c2 to c1.
- Join is a specialization rule, because the resulting graph is less general than either of its components.

Join Example

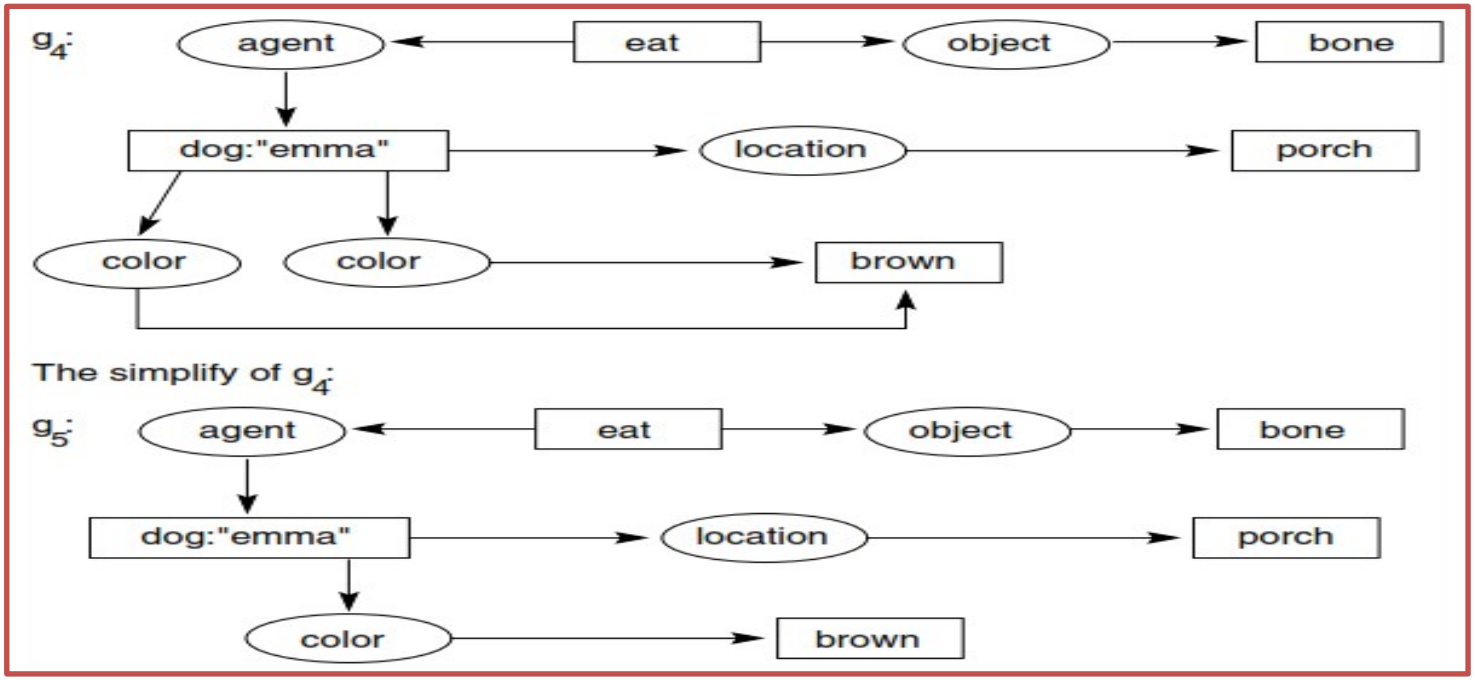
- The Join of g1 and g3



Generalization and Specialization(iv)

The simplify rule.

- If a graph contains two duplicate relations, then one of them may be deleted, along with all its arcs. This is the simplify rule.
- Duplicate relations often occur as the result of a join operation, as in graph g4.



Propositional Nodes

- In addition to using graphs to define relations between objects in the world, we may also want to define relations between propositions. Consider, for example, the statement "Tom believes that Jane likes pizza". "Believes" is a relation that takes a proposition as its argument.
- Conceptual graphs include a concept type, proposition, that takes a set of conceptual graphs as its referent and allows us to define relations involving propositions.
- Propositional concepts are indicated as a box that contains another conceptual graph.
- These proposition concepts may be used with appropriate relations to represent knowledge about propositions. Figure 7.24 shows the conceptual graph for the above assertion about Jane, Tom, and pizza.

Propositional Nodes Example(i)

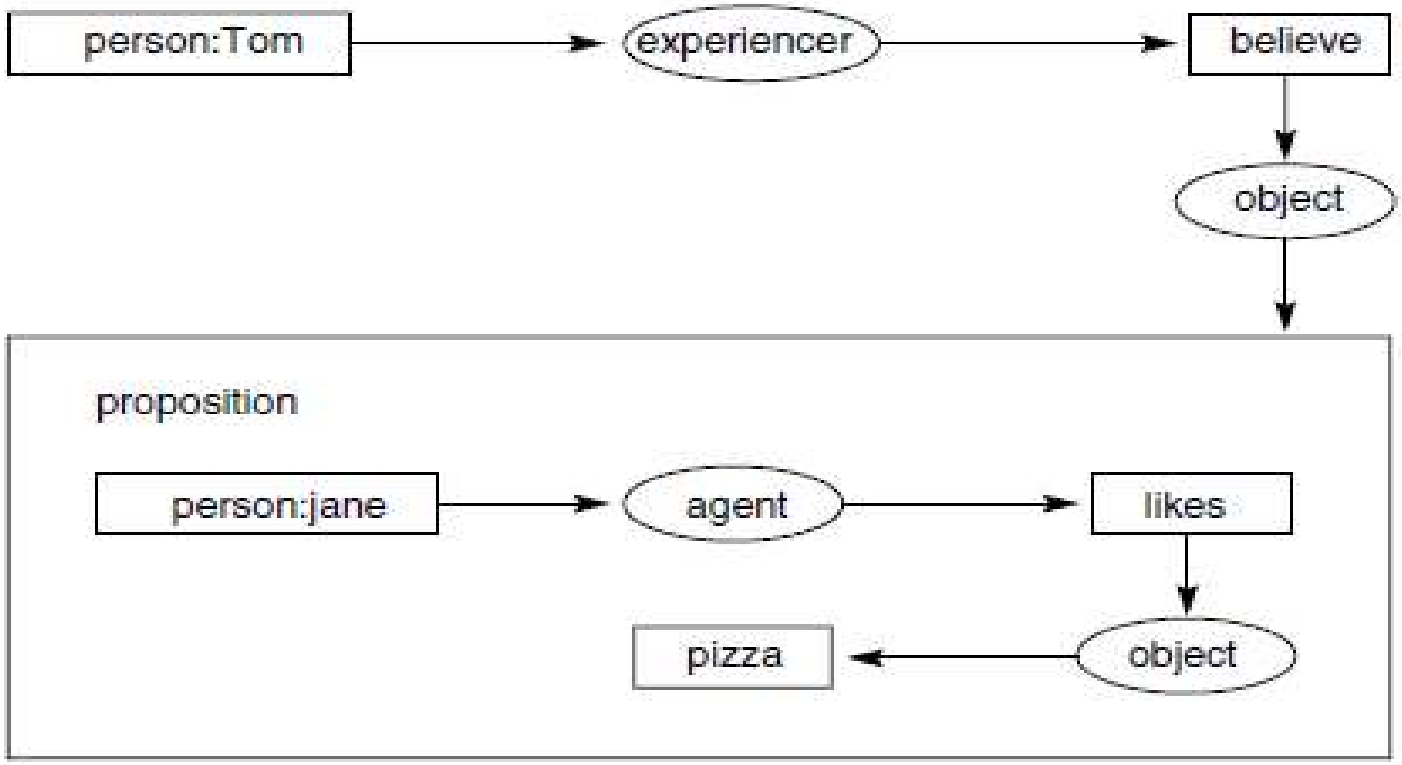


Figure 7.24 Conceptual graph of the statement Tom believes that Jane likes pizza, showing the use of a propositional concept.

Propositional Nodes Example(ii)

- The experiencer relation is loosely analogous to the agent relation in that it links a subject and a verb. The experiencer link is used with belief states based on the notion that they are something one experiences rather than does.

- Figure 7.24 shows how conceptual graphs with propositional nodes may be used to express the modal concepts of knowledge and belief. Modal logics are concerned with the various ways propositions are entertained: believed, asserted as possible, probably or necessarily true, intended as a result of an action, or counterfactual (Turner 1984).

Conceptual Graphs and Logic

- Using conceptual graphs, we can easily represent conjunctive concepts such as "The dog is big and hungry", but we have not established any way of representing negation or disjunction. Nor have we addressed the issue of variable quantification.
- We may implement negation using propositional concepts and a unary operation called neg. neg takes as argument a proposition concept and asserts that concept as false. The conceptual graph of Figure 7.25 uses neg to represent the statement "There are no pink dogs".

Conceptual Graphs and Logic Example

- Conceptual graph of Figure 7.25 uses neg to represent the statement “There are no pink dogs”.

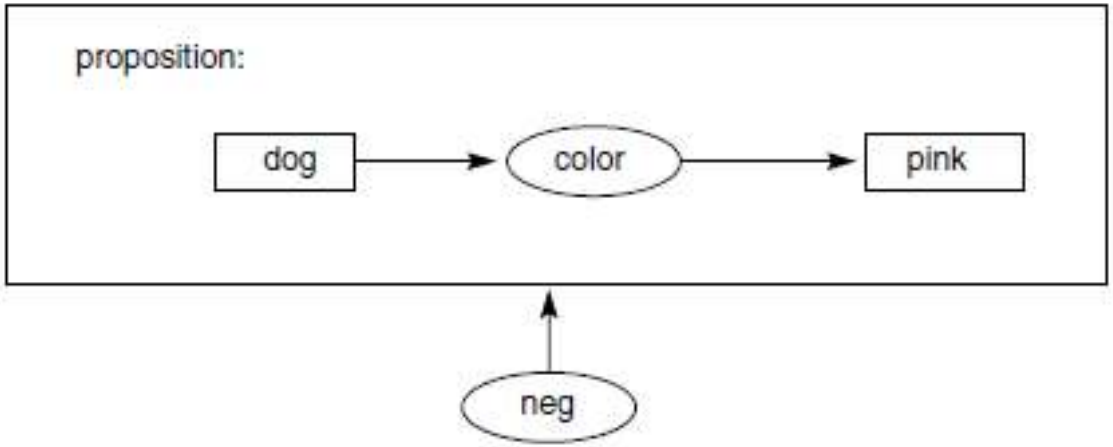


Figure 7.25 Conceptual graph of the proposition There are no pink dogs.

Conceptual graphs and predicate calculus(i)

- Conceptual graphs are equivalent to predicate calculus in their expressive power. As these examples suggest, there is a straightforward mapping from conceptual graphs into predicate calculus notation. The algorithm, taken from Sowa (1984), for changing a conceptual graph, g, into a predicate calculus expression is:

1. Assign a unique variable, x1, x2, . . . , xn, to each of the n generic concepts in g.
2. Assign a unique constant to each individual concept in g. This constant may simply be the name or marker used to indicate the referent of the concept.

Conceptual graphs and predicate calculus(ii)

3. Represent each concept node by a unary predicate with the same name as the type of that node and whose argument is the variable or constant given that node.
4. Represent each n-ary conceptual relation in g as an n-ary predicate whose name is the same as the relation. Let each argument of the predicate be the variable or constant assigned to the corresponding concept node linked to that relation.
5. Take the conjunction of all atomic sentences formed under 3 and 4. This is the body of the predicate calculus expressions. All the variables in the expression are existentially quantified.

Conceptual graphs and predicate calculus Example

- For example, the graph of Figure 7.16 below may be transformed into the predicate calculus expression

Figure 7.16 Conceptual graph indicating that the dog named emma is brown.

$$\exists X_1 (dog(emma) \wedge color(emma,X_1) \wedge brown(X_1))$$