UNIT - IV: Knowledge representation

Syllabus:

Introduction, approaches to knowledge representation, knowledge representation using semantic network, extended semantic networks for KR, knowledge representation using frames.

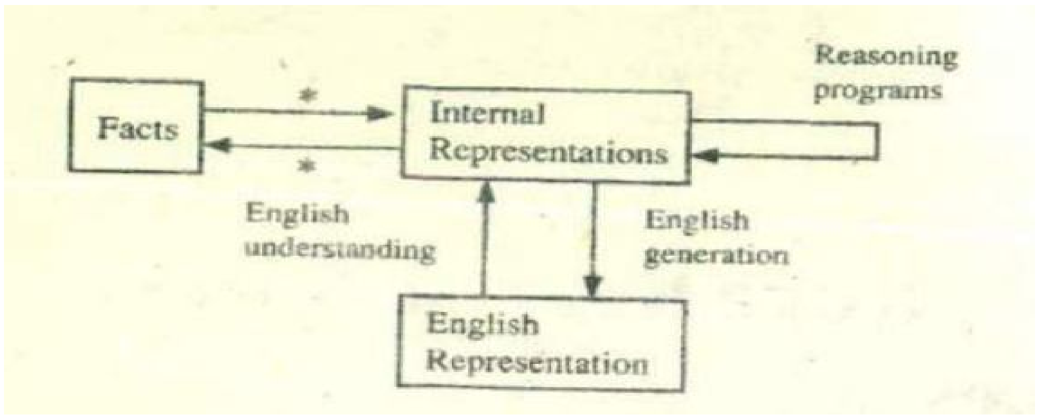
Advanced knowledge representation techniques: Introduction, conceptual dependency theory, script structure, semantic web.

Outcomes:

Student will be able to:

4.1 Introduction: Representations and Mappings

- In order to solve the complex problems encountered in artificial intelligence; we need both a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to new problems.
- A variety of ways of representing knowledge (facts) have been exploited in Al programs.
- We deal with two different kinds of entities:
    - **Facts:** truths in some relevant world, these are the things we want to represent.
    - **Representations of facts** in some chosen formalism. These are the things we will actually be able to manipulate.
- These entities are at two levels:
    - **The knowledge level**, at which facts (including each agent's behaviour. and current goals) are described.
    - **The symbol level**, at which representations of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs.



- The two-way mappings exist between facts and representations called **“Representation mappings”**. The forward representation mapping maps from facts to representations. The backward representation mapping maps from representations to facts.
- One representation of facts is: **natural language (particularly English) sentences.** Regardless of the representation for facts that we use in a program, we may also need to be concerned with an English representation of those facts in order to facilitate getting information into and out of the system.
- In this case, we must have mapping function from English sentences to the representation we are actually going to use and back to sentences.
- Consider the English sentence:

  Spot is a dog.

  The fact represented by that English sentence can also be represented in logic as:

  dog (Spot)

  Suppose that we also have a logical representation of the fact that all dogs have tails:

  $\forall x$: dog (x)$\rightarrow$hastail (x)

  Then, using the deductive mechanisms of logic, we may generate the new representation object:

  hastail(Spot)

  Using an appropriate backward mapping function, we could then generate the English sentence:

  Spot has a tail
- The available mapping functions are not one-to-one; they are many . to-many relations. In other words, each object in the domain may map to several elements in the range, and several elements in the domain may map to the same element of the range.)
- Consider an example of the mappings involving English representations of facts. The two sentences:

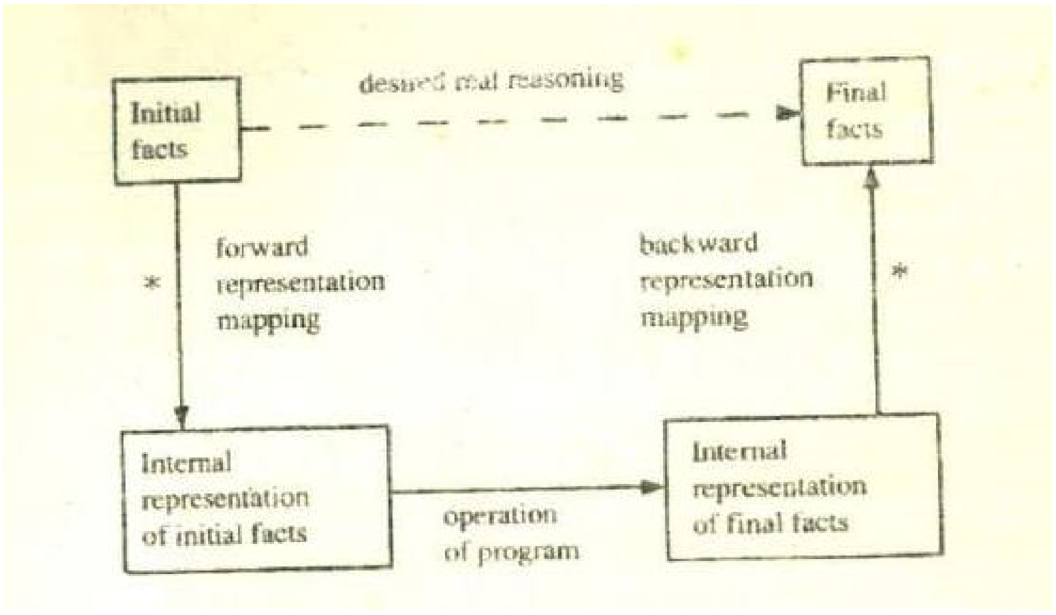
  "All dogs have tails" and "Every dog has a tail"

  Both represent the same fact, namely that **every dog has at least one tail.**

  On the other hand, the former could represent either the tact that every dog has at least one tail or the fact that each dog has several tails.

  The latter may represent either the fact that every dog has at least one tail or the fact that there is a tail that every dog has.
- So, when we try to convert English sentences into some other representation, such as logical propositions, we must first decide what

facts the sentences represent and then convert those facts into the new representation.

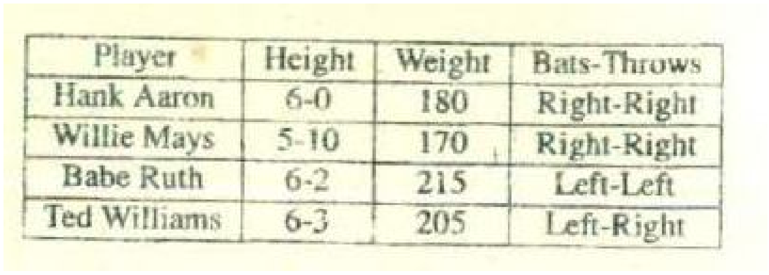

desired real reasoning
Initial facts
Final facts
forward representation mapping
backward representation mapping
*
*
Internal representation of initial facts
operation of program
Internal representation of final facts

4.2 Approaches to Knowledge Representation

- A good system for the representation of knowledge in a particular domain should possess the following four properties:
  - **Representational Adequacy:** the ability to represent all of the kinds of knowledge that are needed in that domain.
  - **Inferential Adequacy:** the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.
  - **Inferential efficiency:** the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.
  - **Acquisitional Efficiency:** the ability to acquire new information easily.
- But, no single system that optimizes all of the capabilities for all kinds of knowledge has yet been found. As a result, multiple techniques for knowledge representation exist. Many programs rely on more than one technique. Following are some knowledge representation techniques:

Simple Relational Knowledge:

- The simplest way to represent declarative facts is as a set of relations of the same sort used in database systems.



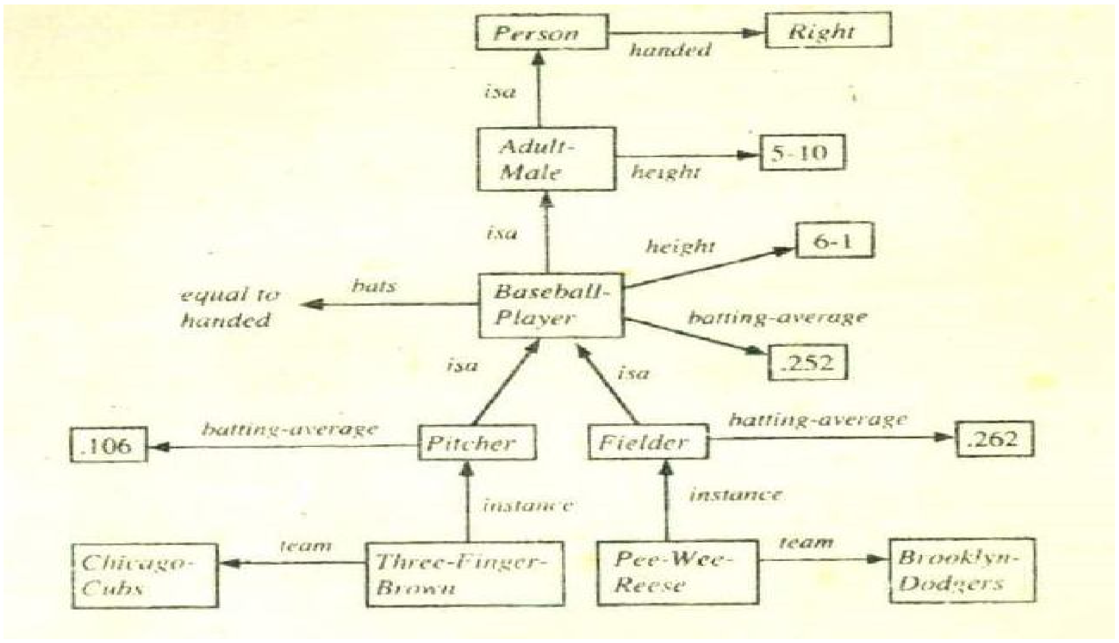
Player	Height	Weight	Bats-Throws
Hank Aaron	6-0	180	Right-Right
Willie Mays	5-10	170	Right-Right
Babe Ruth	6-2	215	Left-Left
Ted Williams	6-3	205	Left-Right

- The reason that this representation is simple is that standing alone it provides very weak inferential capabilities, but knowledge represented in this form may serve as the input to more powerful inference engines.
- The relational knowledge corresponds to a set or attributes and associated values that together describe the objects of the knowledge base.
- Providing support for relational knowledge is what database systems are designed to do.

Inheritable Knowledge

- Knowledge about objects, their attributes, and their values need not be as simple as that shown in relational knowledge.
- In particular, it is possible to augment the basic representation with inference mechanisms that operate on the structure of the representation.
- For this to be effective, the structure must be designed to correspond to the inference mechanisms that are desired.
- One of the most useful forms of inference is property **inheritance**, in which elements of specific classes inherit attributes and values from more general classes in which they are included.
- In order to support property inheritance, objects must be organized into classes and classes must be arranged in a generalization hierarchy.
- Lines represent attributes. Boxed nodes represent objects and values of attributes of objects. The arrows on the lines point from

an object to its value along the corresponding attribute line. The



structure called **slot-and-filler** structure.

- Each individual frame represents the collection of attributes and values associated with a particular node. This structure is called a **Semantic network or a collection of Frames.**

Baseball-Player isa:	Adult-Male
bars: (EQUAL handed)	
height:	6-1
batting-average:	.252

Basic mechanism of inheritance:

Algorithm: Property Inheritance

To retrieve a value V, for attribute A of an instance object O:

1. Find O in the knowledge base.

2. If there is a value there for the attribute A, report that value.

3. Otherwise, see if there is a value for the attribute instance. If not, then fail.

4. Otherwise, move to the node corresponding to that value and look for a value for the attribute A. If one is found, report it.

5. Otherwise, do until there is no value for the isa attribute or until an answer is found:

(a) Get the value of the isa attribute and move to that node.

(b) See if there is value for the attribute A. If there is, repori it.

- ➢ We can apply this procedure to our example knowledge base to derive answers to the following queries:
  - **team (Pee-Wee-Reese) = Brooklyn-fladgers.** This attribute had a value stored explicitly in the knowledge base.
  - **batting-average (Three- Finger-Brown) = .106**. Since there is no value for batting average stored explicitly for Three Finger Brown, we follow the instance attribute to Pitcher and extract the value stored there.
  - **height (Pee-Wee-Reese) = 6-1**. This represents another default inference. Notice here that because we get to it first, the more specific tact about the height of baseball players overrides a more general fact about the height of adult males.
  - **bats (Three- Finger-Brown) =Right.** To get a value for the attribute bats required going up the isa hierarchy to the class Baseball-Player. But what we found there was not a value but a rule for computing a value. This rule required another value as input. So the entire process must be begun again recursively to find a value for **handed**. This time, it is necessary to go all the way up to Person to discover that the default value for handedness for people is Right. Now the rule for bats can be applied, producing the result Right.

Inferential Knowledge

- ➢ The power of traditional logic is necessary to describe the inferences that are needed.
- ➢ This knowledge is useless unless there is also an inference procedure that can exploit it The required inference procedure is one that implements the standard logical rules of inference.
- ➢ The procedures, may reason forward from given facts to conclusions, or may reason backward from desired conclusions to given facts. One of the most commonly used of these procedures is **Resolution**, which exploits a proof by contradiction strategy.

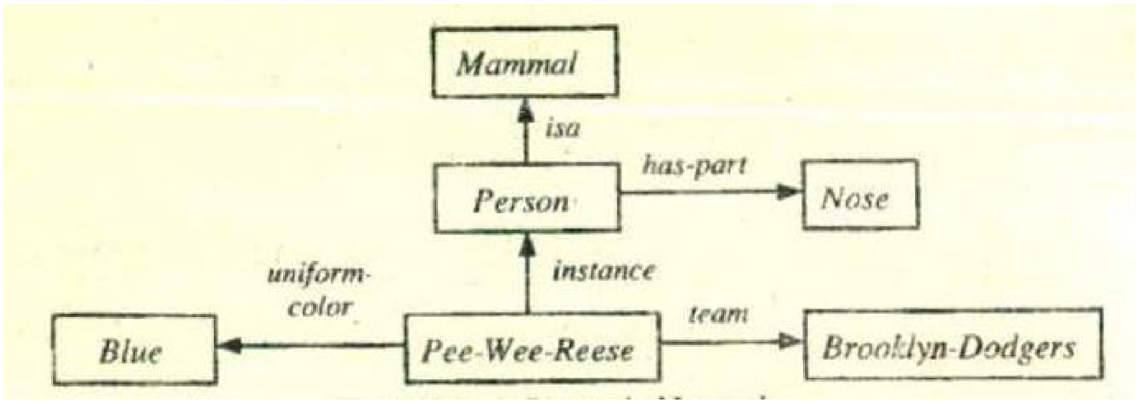
Procedural Knowledge

- ➢ This kind of knowledge is operational.
- ➢ Procedural knowledge specifies what to do and when to do.

- Procedural knowledge can be represented in programs in many ways. The most common way is simply as code (in some programming language such as LISP) for doing something. The machine uses the knowledge when it executes the code to perform a task.
- But, this way of representing procedural knowledge gets low scores for the following properties:
  - **Inferential adequacy**, because it is very difficult to write a program that can reason about another programs behaviour.
  - **Acquisitional efficiency**, because the process of updating and debugging large pieces of code becomes unwieldy.
- The most commonly used technique for representing procedural knowledge in Al Programs is the use of **production rules.**

4.3 Semantic Nets

- The main idea behind semantic nets is that the meaning of a concept comes from the ways in which it is connected to other concepts.
- In a semantic net, information is represented as a set of nodes connected to each other by a set of labeled arcs, which represent relationships among the nodes.



- This network contains examples of both the isa and instance relations, and domain-specific relations like team and uniform-color. In this network, we could use inheritance to derive the additional relation:

  has-part (Pee- Wee-Reese, Nose)
- Earlier, semantic nets were used to find relationships among objects by spreading activation out from each of two nodes and seeing where the activation met. This process is called **intersection search.**
- Using this process, it is possible to use the network to answer questions such as "**What is the connection between the Brooklyn**

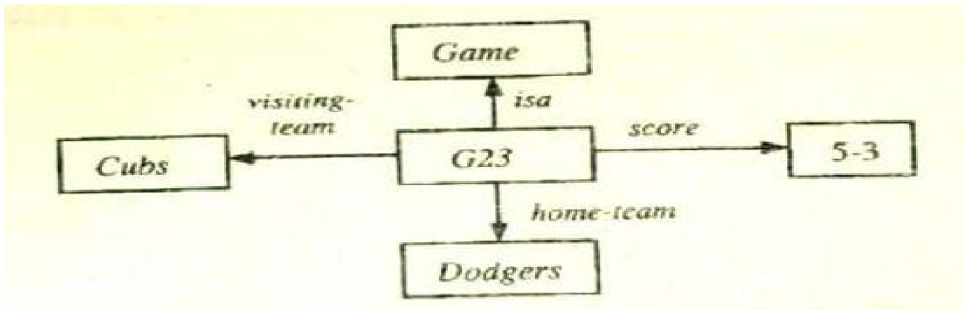
Dodgers and blue?" This kind of reasoning exploits one of the important advantages that slot-and-filler structures have over purely logical representations because it takes **advantage of the entity-based organization of knowledge** that slot-and-filler representations provide.

Representing Non binary Predicates:

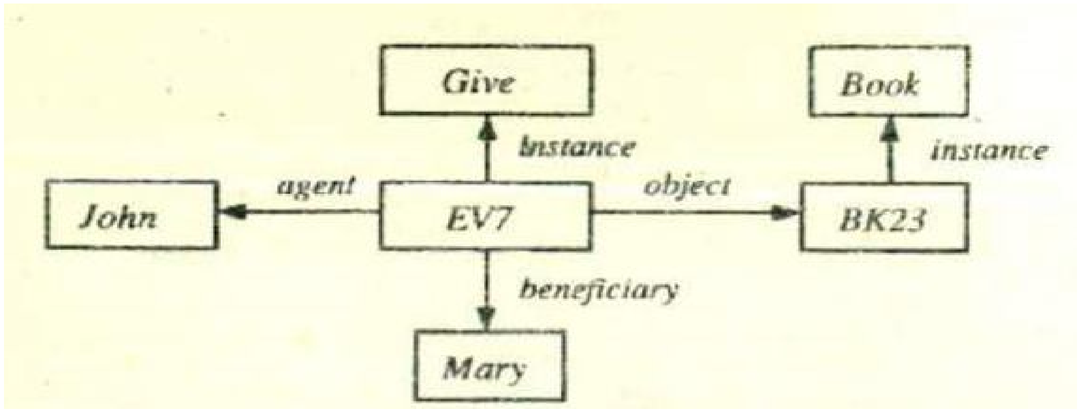
- Semantic nets are a natural way to represent relationships that would appear as ground instances of binary predicates in predicate logic.
- Some of the arcs can be represented in logic as:
    - isa(Person, mammal)
    - instance(Pee-Wee-Reese, Person)
    - team(Pee-Wee-Reese, Brooklyn- Dodgers)
    - uniform color(Pee-Wee-Reese, Blue)
- Unary predicates in logic can be thought of as binary predicates using general-purpose predicates, such as **isa and instance**. For example:

  Man (Marcus) could be rewritten as **instance (Marcus, Man)**

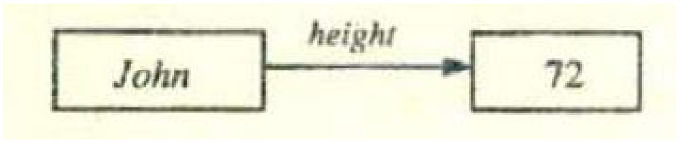
- Three or more place predicates can also be converted to a binary form by creating one new object representing the entire predicate statement and then introducing binary predicates to describe the relationship to this new object of each of the original arguments.
- For example, **score (Cubs, Dodger, 5-3)**
  This can be represented in a semantic net by creating a node to represent the specific game and then relating each of the three pieces of information to it.



- The sentence: **John gave the book to Mary** could be represented by the network as follows:

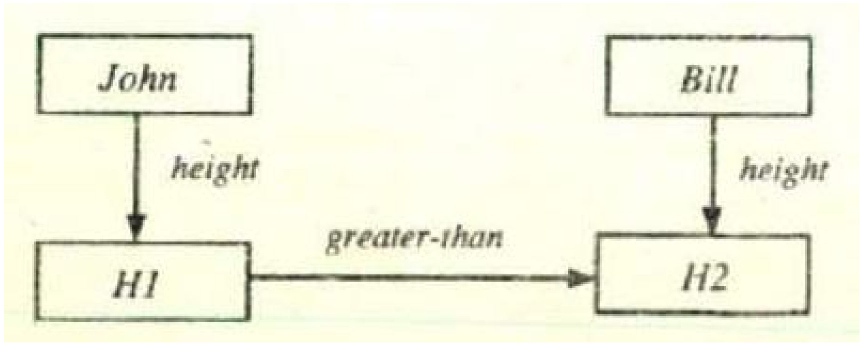


- There should be a difference between a link that defines a new entity and one that relates two existing entities. Consider the net:



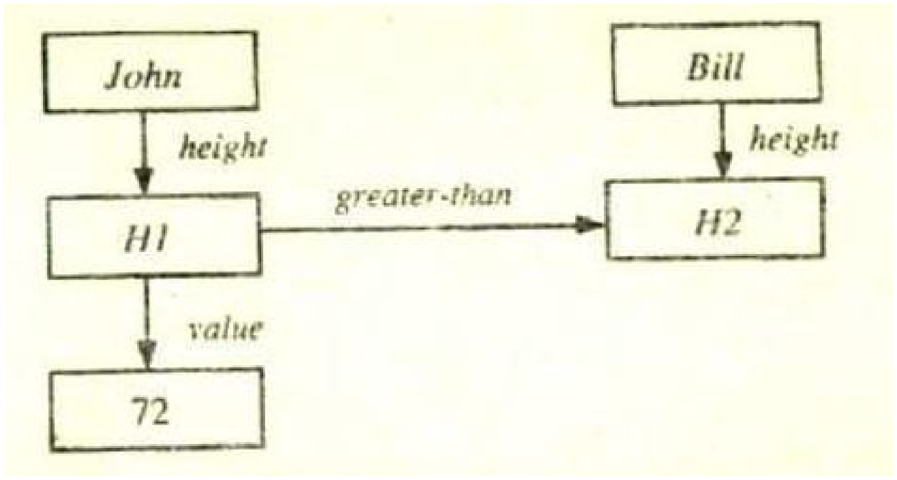
Both nodes represent objects that exist independently of their relationship to each other.

- Suppose we want to represent the fact that: **John is taller than Bill,** using the net:



The nodes HI and 112 are new concepts representing John's height and Bill's height, respectively. They are defined by their relationships to the nodes John and Hill.

- Sometimes it is useful to introduce the arc value to make this distinction clear. Thus we might use the following net to represent the fact that **John is 6 feet tall and that he is taller than Bill.**

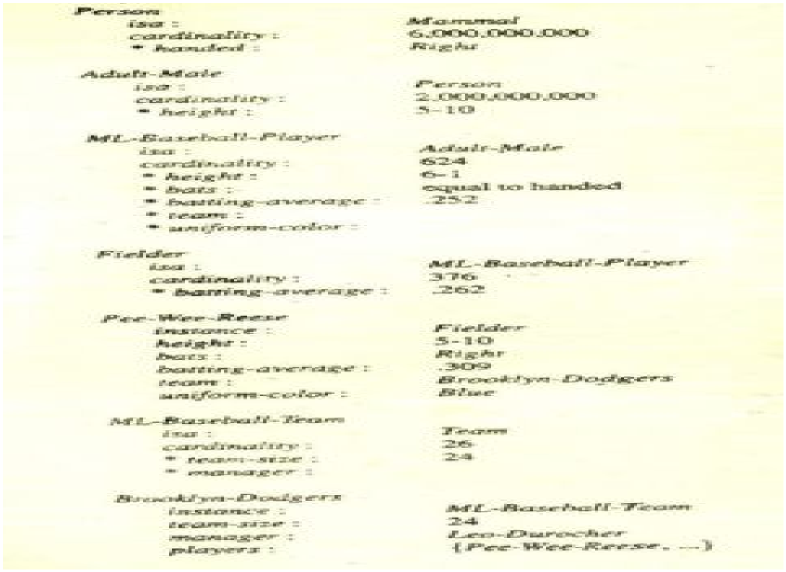


4.4 Frames

- A frame is a collection of attributes (usually called slots) and associated values (and possibly constraints on values) that describes some entity in the world.
- A single frame taken alone is rarely useful. Instead, we build **frame systems** out of collection of frames that are connected to each other. The value of an attribute of one frame may be another frame.

Frames as Sets and Instances:

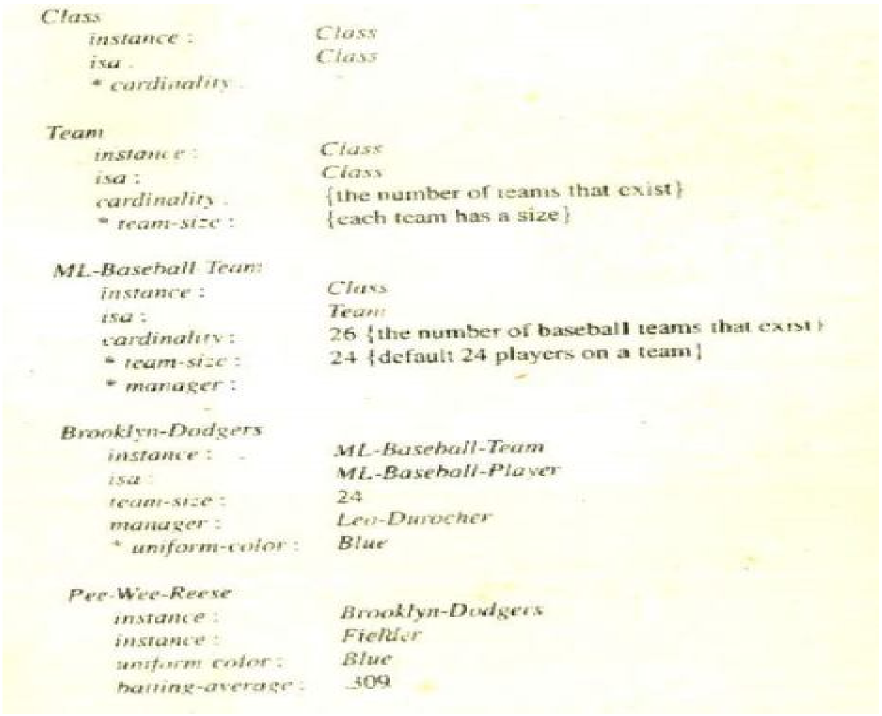
- Set theory provides a good basis for understanding frame systems. Each frame represents either a class (a set) or an instance (an element of a class).
- Consider the following frame system:



```
Person
    isa :                          Mammal
    cardinality :                  6,000,000,000
    * handed :                     Right

Adult-Male
    isa :                          Person
    cardinality :                  2,000,000,000
    * height :                     5-10

ML-Baseball-Player
    isa :                          Adult-Male
    cardinality :                  624
    * height :                     6-1
    * bats :                       equal to handed
    * batting-average :            .252
    * team :
    * uniform-color :

Fielder
    isa :                          ML-Baseball-Player
    cardinality :                  376
    * batting-average :            .262

Pee-Wee-Reese
    instance :                     Fielder
    height :                       5-10
    bats :                         Right
    batting-average :              .309
    team :                         Brooklyn-Dodgers
    uniform-color :                Blue

ML-Baseball-Team
    isa :                          Team
    cardinality :                  26
    * team-size :                  24
    * manager :

Brooklyn-Dodgers
    instance :                     ML-Baseball-Team
    team-size :                    24
    manager :                      Leo-Durocher
    players :                      {Pee-Wee-Reese, ...}
```

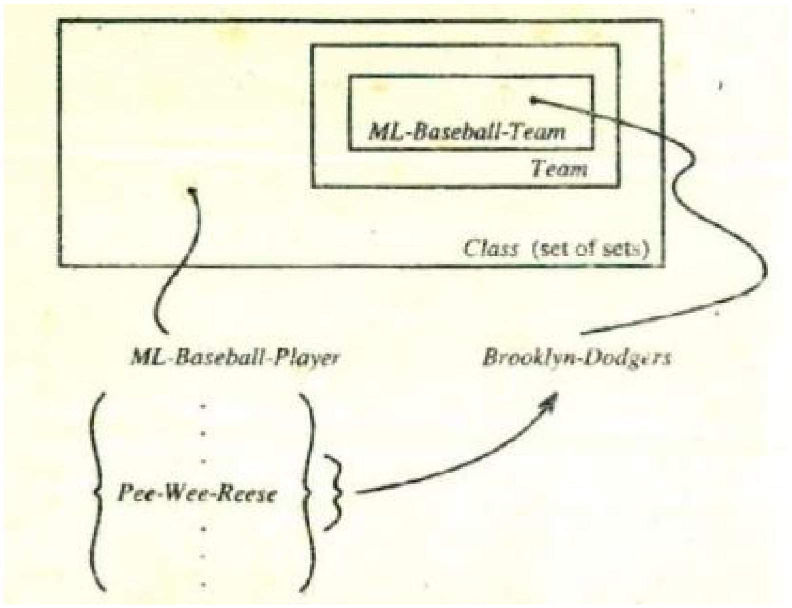
- The frames Person, Adult-Male, ML-Baseball Player (corresponding to major league baseball players), Pitcher, and ML-Baseball-Team (for major league baseball team) are all classes. The frames Pee-Wee-Reese and Brooklyn-Dodgers are instances.
- The set of adult males is a subset of the set of people. The set of major league baseball players is a subset of the set of adult males, and so forth. The instance relation corresponds to the relation "**element-of**". Pee Wee Reese is an element of the set of fielders. Thus he is also an element of all of the supersets of fielders, including major league baseball players and people.
- A class represents a set; there are two kinds of attributes that can be associated with it. They are:
  - attributes about the set itself, and
  - attributes that are to be inherited-by each element of the set

  We indicate the difference between these two by prefixing the latter with an asterisk (*).For example, consider the class **ML-Basebalf-Player**.
- It is important to distinguish between **regular classes**, whose elements are individual entities, and **metaclasses**, which are special classes whose elements are themselves classes. A class is now an element of (instance) some class (or classes) as well as a subclass (isa) of one or more classes. A class inherits properties from the class of which it is an instance, just as any instance does. In addition, a class passes inheritable properties down from its super classes to its instances.



```
Class
    instance :          Class
    isa :               Class
    * cardinality :

Team
    instance :          Class
    isa :               Class
    cardinality :       {the number of teams that exist}
    * team-size :       {each team has a size}

ML-Baseball Team
    instance :          Class
    isa :               Team
    cardinality :       26 {the number of baseball teams that exist}
    * team-size :       24 {default 24 players on a team}
    * manager :

Brooklyn-Dodgers
    instance :          ML-Baseball-Team
    isa :               ML-Baseball-Player
    team-size :         24
    manager :           Leo-Durocher
    * uniform-color :   Blue

Pee-Wee-Reese
    instance :          Brooklyn-Dodgers
    instance :          Fielder
    uniform color :     Blue
    batting-average :   .309
```

- The most basic **metaclass** is the class **Class**. It represents the set of all classes. All classes are instances of it, either directly or through one of its subclasses. In the example, **Team** is a subclass (subset) of **Class** and **ML-Baseball-Team** is a subclass of **Team**. The class **Class** introduces the attribute cardinality, which is to be inherited by all instances of Class (including itself).

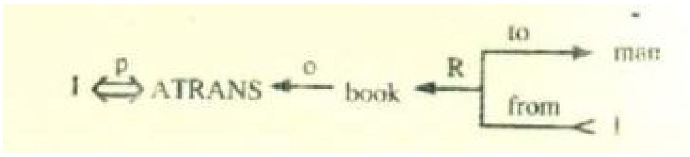


- Ways in which classes are related to each other:
  - Class1 can be a subset of Class2.
  - If Class2 is a metaclass, then Class1 can be an instance of Class2.
  - **Mutually-disjoint-with:** which relates a class to one or more other classes that are guaranteed to have no elements in common with it.
  - **Is-covered-by:** which relates a class to a set of subclasses, the union of which is equal to it. If a class is covered-by a set of mutually disjoint classes, then S is called a partition of the class.
- The following are the properties the frames would be able to represent and use in reasoning:
  - The classes to which the attribute can be attached
  - Constraints on either the type or the value of the attribute.
  - A value that all instances of a class must have by the definition of the class.
  - A default value for the attribute.

- Rules for inheriting values for the attribute. The usual rule is to inherit down isa and instance links.
- Rules for computing a value separately from inheritance.

4.5 Conceptual Dependency

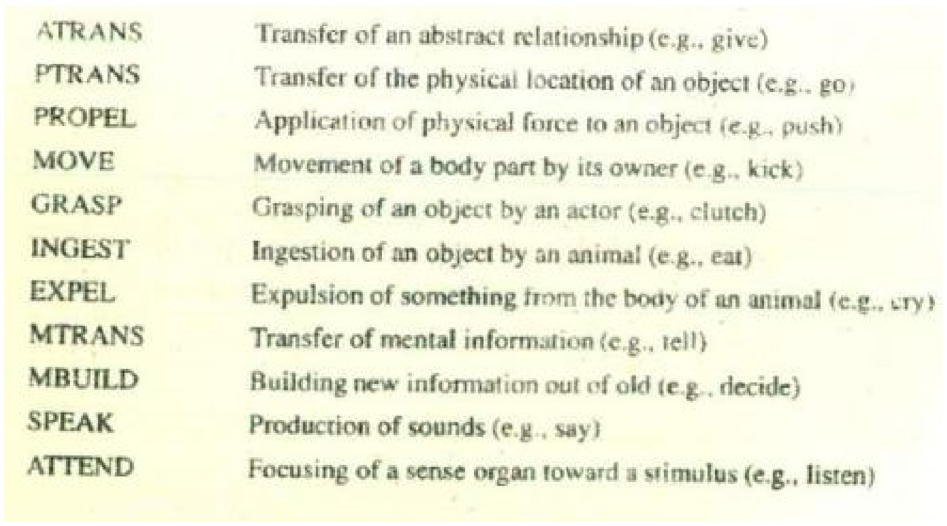
- ➢ Semantic networks and frame systems may have specialized links and inference procedures, but there are no hard and fast rules about what kinds of objects and links arc good in general for knowledge representation.
- ➢ Conceptual Dependency specifies what types of objects and relations are permitted.
- ➢ Conceptual Dependency (CD) is a theory of how to represent the kind of knowledge about events that is usually contained in natural language sentences. The goal is to represent the knowledge in a way that:
  - Facilitates drawing inferences from the sentences.
  - Is independent of the language in which the sentences were originally stated.
- ➢ The CD representation of a sentence is built not out of primitives corresponding to the words used in the sentence, but rather out of conceptual primitives that can be combined to form the meanings of words in any particular language.
- ➢ Semantic nets provide only a structure into which nodes representing information at any level can be placed. Conceptual dependency provides both a **structure and a specific set of primitives**, at a particular level of granularity out of which representations of particular pieces information can be constructed.
- ➢ Consider a Simple Conceptual Dependency Representation of : **I gave the man a book,** where the symbols have the following meanings:



- Arrows indicate direction of dependency.
- Double arrow indicates two way links between actor and action.
- p indicates past tense.
- ATRANS is one of the primitive acts used by the theory. It indicates transfer of possession.

- O indicates the object case relation.
- R indicates the recipient case relation.

➢ In CD, representations of actions are built from a set of primitive acts:



ATRANS	Transfer of an abstract relationship (e.g., give)
PTRANS	Transfer of the physical location of an object (e.g., go)
PROPEL	Application of physical force to an object (e.g., push)
MOVE	Movement of a body part by its owner (e.g., kick)
GRASP	Grasping of an object by an actor (e.g., clutch)
INGEST	Ingestion of an object by an animal (e.g., eat)
EXPEL	Expulsion of something from the body of an animal (e.g., cry)
MTRANS	Transfer of mental information (e.g., tell)
MBUILD	Building new information out of old (e.g., decide)
SPEAK	Production of sounds (e.g., say)
ATTEND	Focusing of a sense organ toward a stimulus (e.g., listen)

➢ The set of CD building blocks is the set of allowable dependencies among the conceptualizations described in a sentence. There are tour primitive conceptual categories from which dependency structures can be built. These are:

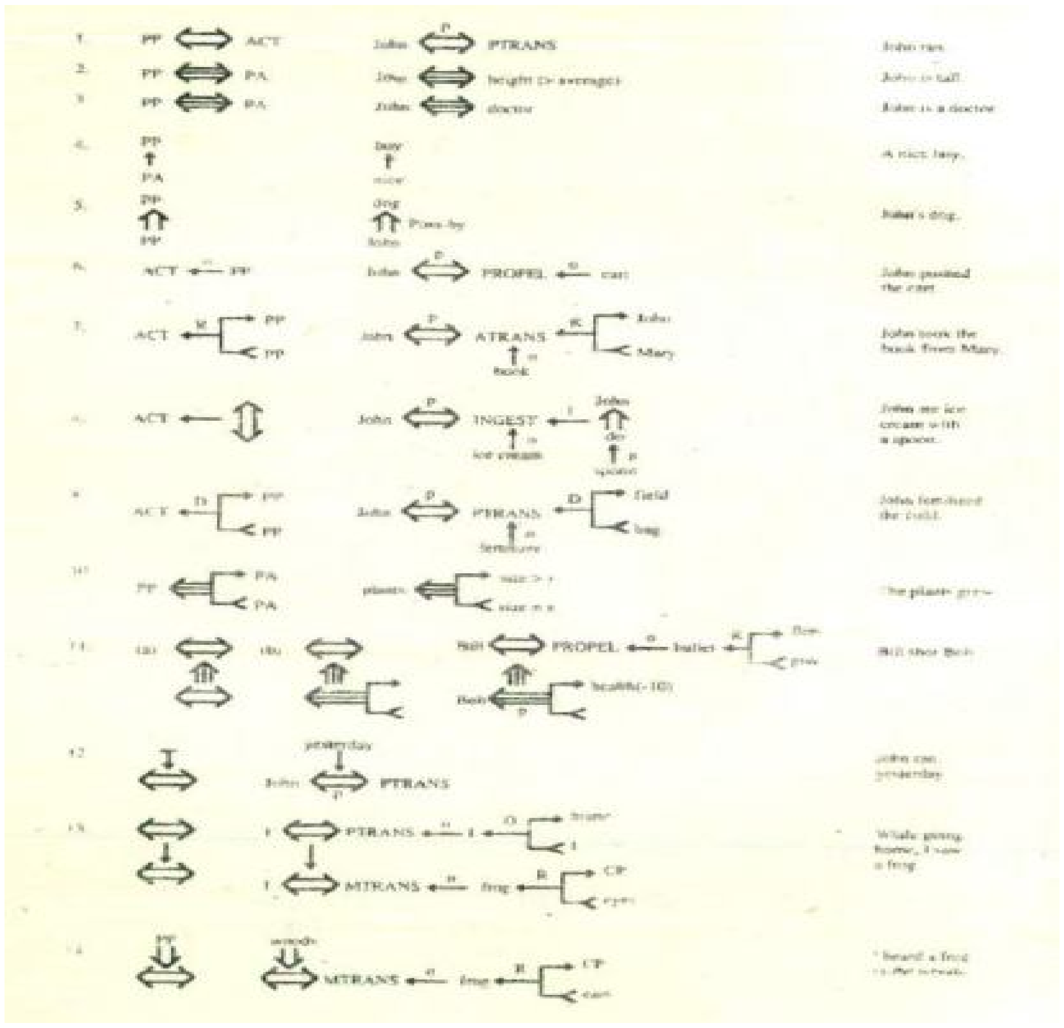
- ACTS Actions
- PPs Objects (picture producers)
- AAs Modifiers of actions (action aiders)
- PAs Modifiers of PPs (picture aiders)

➢ The Dependencies of CD:

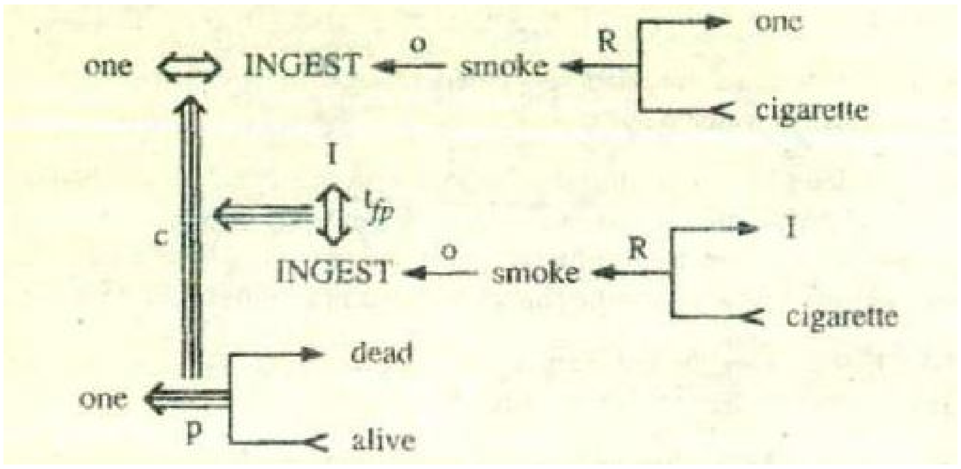
- Rule 1 describes the relationship between an actor and the event he or she causes. This is a two-way dependency since neither actor nor event can be considered primary. The letter p above the dependency link indicates past tense.
- Rule 2 describes the relationship between a PP and a PA that is being asserted to describe it.
- Rule 3 describes the relationship between two PPs, one of which belongs to the set defined by the other.
- Rule 4 describes the relationship a PP and an attribute that has already been predicated of it. The direction of the arrow is toward the PP being described.
- Rule 5 describes the relationship between two PPs, one of which provides a particular kind of information about the other. The three most common types of information to be provided in this way are:
  - possession (shown as POSSBY)
  - location (shown as LOC), and

    - physical containment (shown as CONT)
    - The direction of is again toward the concept being described
  - Rule 6 describes the relationship between an ACT and the PP that is the object oh that ACT. The direction of the arrow is toward the ACT since the context of the specific ACT determines the meaning of the object relation.
  - Rule 7 describes the relationship between an ACT and the source and the recipient of the ACT.
  - Rule 8 describes the relationship between an ACT and the instrument with which it is performed. The instrument must always be a full conceptualization, not just a single physical object.
  - Rule 9 describes the relationship between an ACT and its physical source and destination.
  - Rule 10 represents the relationship between a PP and a state in which it started and another in which it ended.
  - Rule 10 describes the relationship between one conceptualization and another that causes it. The arrows indicate dependency of one conceptualization on another and point in the opposite direction of the implication arrows. The two forms of the rule describe the cause of an action and the cause of a state change.
  - Rule 12 describes the relationship between a conceptualization and the time at which the event it describes occurred.
  - Rule 13 describes the relation between one conceptualization and another that is the time of the first.
  - Rule 14 describes the relationship between a conceptualization and the place at which it occurred.
- The set of conceptual tenses in a CD are:

p	Past
f	Future
t	Transition
t_x	Start transition
t_y	Finished transition
k	Continuing
?	Interrogative
nil	Present
delta	Timeless
c	Conditional

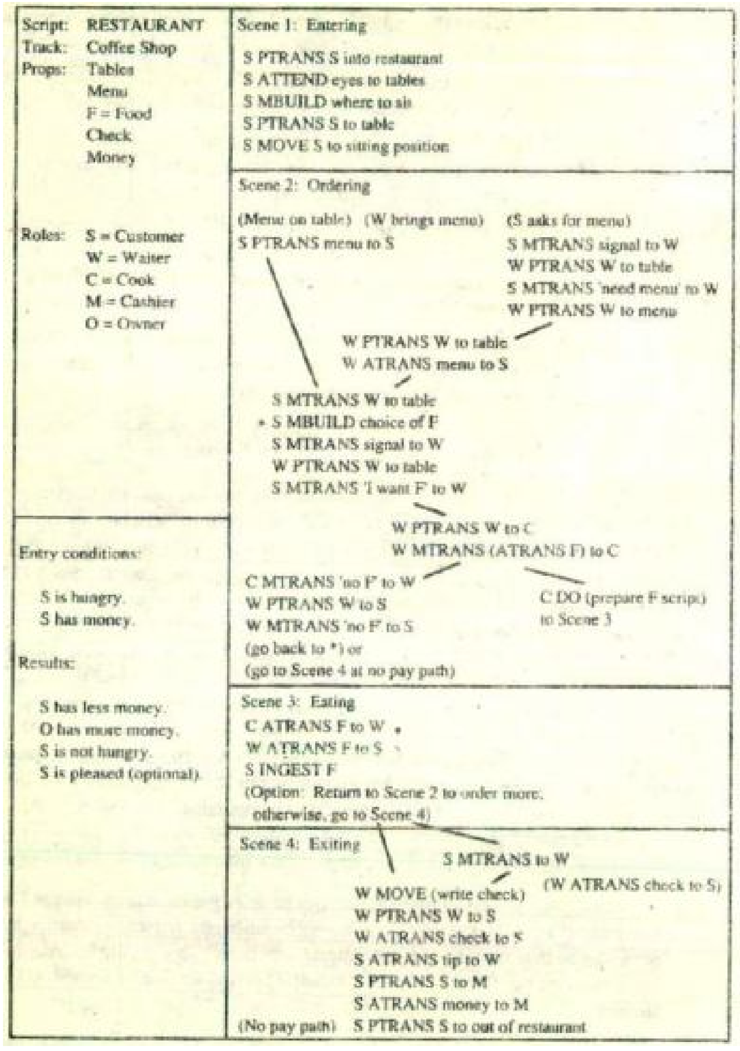


➢ Consider the sentence: **Since smoking can kill you, I stopped.**



4.6 Scripts

- CD is a mechanism for representing and reasoning about events. A script is a structure that describes a stereotyped sequence of events in a particular context.
- Script is a mechanism for representing knowledge about common sequences of events.
- A script consists of a set of slots. Associated with each slot may be some information about what kinds of values it may contain as well as a default value to be used if no other information is available.
- Consider the following example of a restaurant script:



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 asks for menu)
S PTRANS menu to S
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 W to table
* 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
(No pay path) S PTRANS S to out of restaurant

- The important components of a script are:
  - **Entry conditions:** Conditions that must, be satisfied before the events described in the script can occur.

- **Result:** Conditions that will, be true after the events described in the script have occurred.
- **Props:** Slots representing objects that are involved in the events described in the script. The presence of these objects can be inferred even if they are not mentioned explicitly.
- **Role:** Slots representing people who are involved in the events described in the script.
- **Track:** The specific variation on a more general pattern that is represented by this particular script. Different tracks of the same script will share many but not all components.
- **Scenes:** The actual sequences of events that occur.

➢ Scripts are useful because, in the real world, there are patterns to the occurrence of events. These patterns arise because of causal relationships between events.

➢ Agents will perform one action so that they will then be able to perform another. The events described in a script form a giant causal chain.

➢ The beginning of the chain is the set of entry conditions which enable the first events of the script to occur. The end of the chain is the set of results which may enable later events or event sequences to occur. Within the chain, events are connected both to earlier events that make them possible and to later events that they enable.

➢ Scripts can also be used to indicate how events that were mentioned relate to each other.

➢ Before a particular script can be applied, it must be activated. There are two ways in which a script can be activated:
  - For **fleeting scripts** (ones that are mentioned briefly and may he referred to again but are not central to the situation), it may be sufficient to store a pointer to the script so that it can be accessed later if necessary.
  - For **non-fleeting scripts** it is appropriate to activate the script fully and to attempt to fill in its slots with particular objects and people involved in the current situation.

➢ Once a script has been activated, there are varieties of ways in which it can be useful in interpreting a particular situation. The most important of these is the **ability to predict events that have not explicitly been observed**.
For example:

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

If you were then asked the question:

Did John eat dinner last night?

By using the restaurant script, a computer question-answerer would also be able to infer that John ate dinner, since the restaurant script could have been activated. Since all of the events in the story correspond to the sequence of events predicted by the script, the program could infer that the entire sequence predicted by the script occurred normally. Thus it could conclude that John ate.

- Scripts provide a way of building a **single coherent interpretation from a collection of observations.** Consider, for example:

> Susan went out to lunch. Site sat down at table and called the waitress. The waitress brought her a menu and she ordered a hamburger.

Now consider the question:

Why did the waitress bring Susan a menu?

The script provides two possible answers to that question:

- Because Susan asked her to.
- So that Susan could decide what she wanted to eat.

Unit- IV
Artificial Intelligence: Knowledge Representation

(Open Elective –I)

Assignment-Cum-Tutorial Questions

Objective Questions

1. The two different kinds of entities in AI are ____________ and _____________.

2. The _________ mappings exist between facts and representations.[ ]

(a) One-way (b) two-way (c) no mapping (d) both (a) & (b)

3.The forward representation mapping maps from ____________ to _________________.

(a) representations, facts (b) facts, representations [ ]

(c) facts, facts (d) representations, representations

4. The ability to represent all of the kinds of knowledge that are needed in that domain is called _____________. [ ]

(a) Referential Efficiency (b) Representational Adequacy

(c) Inferential Efficiency (d) Acquisitional Efficiency

5. The ability to acquire new information easily is called ________. [ ]

(a) Referential Efficiency (b) Representational Adequacy

(c) Inferential Efficiency (d) Acquisitional Efficiency

6. The two important attributes of inheritance are _______ and ________.

7. Weak slot-and-filler structures are __________and _____________.

8. Strong slot-and-filler structures are __________and _____________.

9. Procedural knowledge is _____________. [ ]

(a) Declarative (b) Operational (c) progressive (d) none

10. Procedural Knowledge get low scores for the properties_____ [ ]

(a) Inferential Adequacy (b) Acquisitional Efficiency

(c) Inferential Efficiency (d) both (a) and (b)

11. The most commonly used technique for representing procedural knowledge in Al Programs is the use of ____________. [ ]

(a) Production rules (b) symbols (c) facts (d) both (b) & (c)

12.The structure in which information is represented as a set of nodes connected to each other by a set of labeled arcs, which represent relationships among the nodes is called ____________. [ ]

13. Define the term "frame".

14. __________ theory provides a good basis for understanding frame systems.

(a) set (b) Graphics (c) Logic (d) none [ ]

15. The classes whose elements are themselves classes are called _______.

(a) Sub class (b) Base class (c) Meta class (d) Parent class [ ]

16. _____________ is a theory of how to represent the kind of knowledge about events that is usually contained in natural language sentences.

17.The primitive that represents transfer of an abstract relationship is ________.

(a) PTRANS (b) ATRANS (c) MOVE (d) GRASP [ ]

18. A ______________ is a structure that describes a stereotyped sequence of events in a particular context.

SECTION-B

Descriptive Questions

1. Enlist the four properties that a knowledge representation system must have?
2. Explain four knowledge representation techniques.
3. Enumerate the basic mechanism of retrieving a value of an attribute, using inheritance.

4. How non binary predicates are represented using semantic net. Explain with suitable example.
5. Represent the following facts using semantic nets:
   John gave the book to Mary
   John is 6 feet tall and that he is taller than Bill.
6. Justify the statement- "Set theory provides a good basis for understanding frame systems".
7. List the ways in which classes are related to each other in frames, with suitable example?
8. List the set of primitives and conceptual tenses used in Conceptual Dependency.

10. Explain the rules used in Conceptual Dependency.

11. Represent the following sentence in CD:

    Since smoking can kill you, I stopped.

12. Describe the important components of a script, with a suitable example.