UNIT-IV

Knowledge Representation

In this unit, we are going to discuss various Knowledge Representation (KR) Techniques. Knowledge representation in AI mainly focus on storing knowledge in such a manner, that programs can process it and achieve human intelligence. In AI, Knowledge Representation is used to solve a problem intelligently. KR makes the problem solving easier. KR system should possess following properties

→ Learning: Refers to a capability that acquires new knowledge, behaviour, understanding etc. In this new information to be classified to avoid redundancy and replication prior to storage, it enables easy retrieval.

→ Efficiency in acquisition: Refers to the ability to acquire new knowledge using automatic methods.

→ Representational adequacy: Refers to the ability to represent the required knowledge.

→ Inferential adequacy: Refers to the ability of manipulate knowledge to produce new knowledge from existed one.

2. Approaches to knowledge Representation:

AI programs use structures known as knowledge structures to represent objects, facts, relationships and procedures. Knowledge structures are usually composed of complex structures such as semantic network, frames, scripts etc. Knowledge base can share some characteristics of database systems such as storing, updating, and retrieving information. Some basic knowledge representation schemes are.

1) Relational Knowledge:

It Comprises objects Consists of attributes and associated values.. In this each fact is stored in a row of relational table. as like as in relational database. A table is organized using set horizontal rows and vertical columns The columns are identified by attributes, rows are identified by corresponding values. in particular Column

Ex:

Name	Age	Sex	Qualification	Salary
John	38	male	PG	20000
Mike	25	Male	UG	15000
Mary	30	female	Ph.D.	25000

we cannot obtain answer for queries such as does a person ~~having~~ having a Ph.D earn more?

2) Knowledge Represented as logic

It provides inferential capability, when knowledge is represented in the form of formal Logic.

Ex: All humans are mortal can be represented in logic as $(\forall x)$ human$(x) \leftarrow$ mortal(x).

→ If we have fact 'John is human', then we can easily infer that John is mortal. The advantage of this approach is we can represent a set of rules, derive more facts, truths, and verify the correctness of new statements.

3) procedural Knowledge:

In this Knowledge is encoded in the form of procedure which carry out specific tasks based on relevant knowledge.

Ex: Interpreter can interpret program on the basis of the available knowledge regarding syntax & semantics of lan

→ The advantage of this approach is, domain-specific knowledge can be easily represented. But there is a problem of Completeness and Consistency.

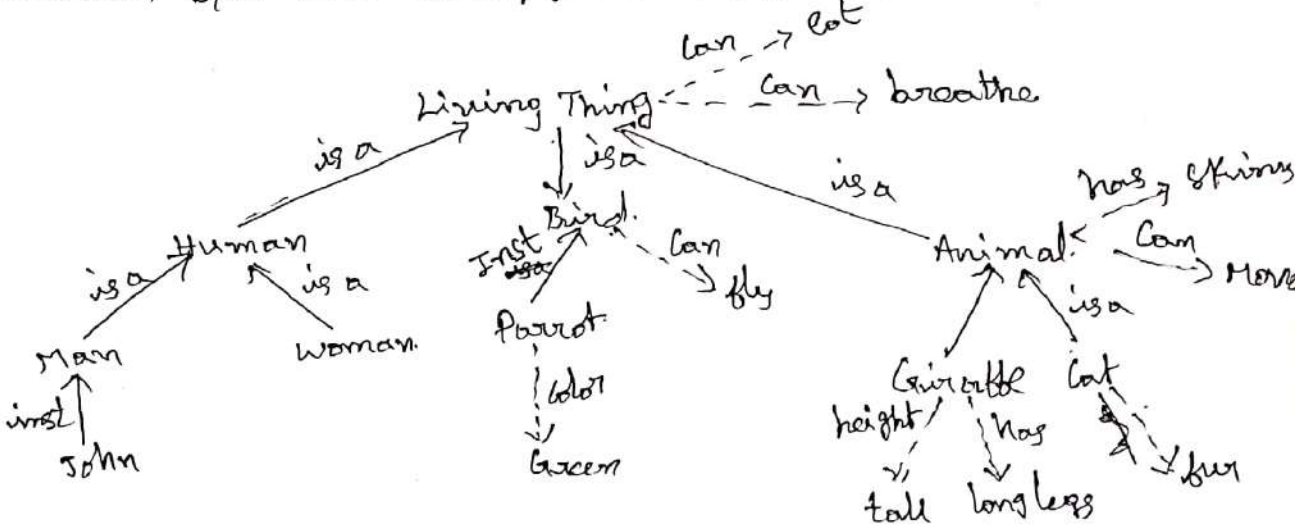
→ Completeness means all cases may not be represented. Consistency means all deductions may not be correct.

3. Knowledge Representation using Semantic network.

In this the meaning of a Concept is derived from its relationship with other concepts and the information is stored by interconnecting nodes with labelled arcs.

Ex: Every human, animals, and birds are living things who can breathe and eat. All birds can fly. Every man and woman are human who have two legs. A Cat has fur and is an animal. All animals have skin and can move. A parrot is a bird and is green in color. John is a man

We can represent such knowledge using a structure called a semantic networks., in a graphical notation where nodes represent concepts or objects and arcs represent relation b/w two concepts. as shown below.



Concepts or objects are related to each other by certain relations The relations may be defined as follows.

1) is a: This relation connects two concepts when one concept is a kind of sub class of other concept. Ex; Man is a Human

2) inst: This relation specific members of a class Ex; John is inst of Man

other relations such as {can, has, color, height} are known as property relations. These have represented by dotted lines point from the concept to its property.

In this property inheritence is achieved. Ex; Does a parrot breathe? can be easily answered as 'yes'. It inherits this property from its super class named as Living_Thing.

The Semantic net interpreter cannot understand all the attribute links unless their semantics are encoded into it.

Inheritence in Semantic net:

Hierarchical structure of knowledge representation allows knowledge to be stored at the highest possible level of abstraction which reduces the size of the knowledge base. Inheritance mechanism is in-built and facilitates inferencing of information associated with nodes in Semantic net. It ensures that all members and Sub-Concepts of a Concept share common properties. properties attached to a particular Concept can be inherited by all sub-Concepts and their members. The Algorithm to retrieve a property of an object is given below.

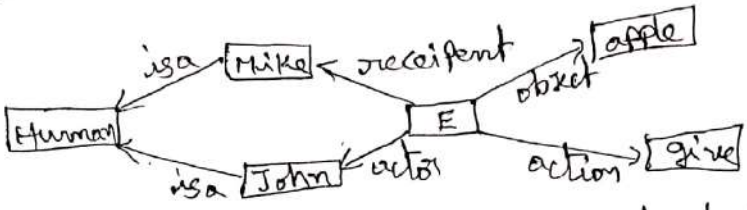
Input: object and property to be found from Semantic net.
output: returns yes, if the object has desired property else return false.

procedure:

→ Find an object in the semantic net
→ Found = False
→ while [(object ≠ root) & Found] do
{
→ If there is an attribute attached with an object then
Found = true;
else { object = isa(object. class) & object = inst(object.class
}
→ If Found = true then report 'yes' else report 'NO'.

4. Extended Semantic Networks for KR.

Logic and semantic networks are used to represent Knowledge. Semantic networks can only express collections of variable-free assertions. The English sentence "John gives an apple to mike and John and mike are human". represented in semantic network as shown below.



Here E' represents an event, which is an act of Giving, whose actor is John, the object is an apple, and receipt is mike The relationships in the network can be expressed in clausal form of logic as follows.

object(E, apple)
action(E, give)
actor(E, John)
receipt(E, Mike)
is_a(John, Human)
isa(Mike, Human).

predicate relations corresponding to labels on the arcs of semantic networks always have two arguments. It is useful when additional information is added to the given system.

Ex: John give an apple to mike in the kitchen. It is easy to add location(E, kitchen) to the set of facts.

In conventional semantic network, we cannot express clausal form of logic. To overcome this, R Kowalski and his colleagues proposed an extended semantic network, that combines the advantages of both logic and semantic networks. ESnet can be interpreted as a variant syntax for the clausal form of logic. It has the same expressive power as that of predicate logic with well-defined semantics, inference rules and procedural information. ESNet is much powerful representation as compared to Semantic network.

In ESNet the terms are represented by nodes, Some constants, variable, and functional terms are represented by constant, variable and functional nodes.

Binary predicate symbols is are represented by labels on arcs of ESNets. Ex: An atom of the form love(John, Marry) is an arc labelled as 'love' with two ends John and Marry The direction of arc indicates the order of arguments of predicat

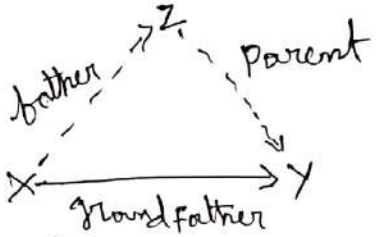
John —Love→ Marry.

Conclusions and conditions are represented in ESNet by arcs.

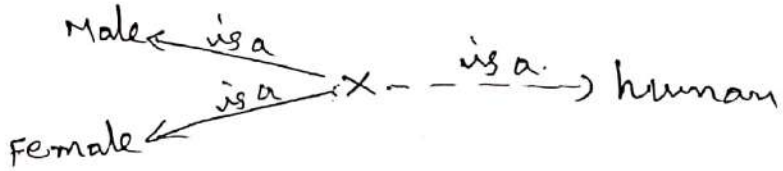
→ The arc denoting conditions are drawn with dotted lines.

→ The arcs denoting Conclusions are drawn with Continuous lines

Ex: grandfather(X, Y) ← father(X, Z), Parent(Z, Y) Can be represented in ~~Es~~ in ESNet as shown below.



Ex: male(X), female(X) ← human(X) Can be represented as in ESNet as



Inference Rules:

Inference rules are embedded in the representation. Some of the inference rules are.

→ The representation of the inference for every action of giving, there is an action of taking in clausal logic is action(b(X), take) ← action(X, give). The event of taking action is a function of the event of giving action. In ESNet, functional terms ie b(X) represented by a single node.

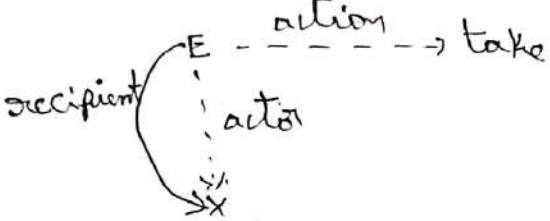
b(X) —action→ take

X — action → give

→ The inference rule that an actor who performs a taking action is also the receipt recipient of this action and can be easily represented in clausal logic.

recipient(E, X) ← action(E, take), actor(E, X).

Here E is a an event of an action of taking. The clause represented in ESNet is

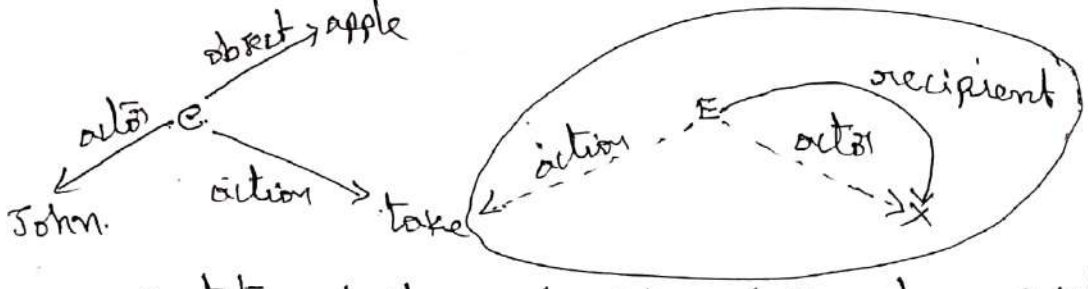


Ex: Represent the following clauses in ESNet:

recipient(E, X) ← action(E, take), actor(E, X)

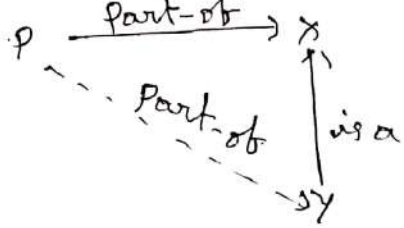
object(e, apple), action(e, take), actor(e, John).

Sol: Here 'E' is a variable for some event, and 'e' is an actual event. ESNet representation of the clauses is shown in the given



Here, the partition is shown to delimit the clause pictorially by enclosing the clausal rule in an ellipse

→ Contradiction in ESNet can be represented as shown below.



Here P part-of X is a conclusion and P part-of Y is condition where Y is linked with X via is a link. Such kind of representation is contradictory and hence there is a contradiction in ESNet.

Deduction in Extended Semantic Networks:

Inference rules along with its semantics, form a part of ESNet.
There are two types of inference mechanisms, namely.

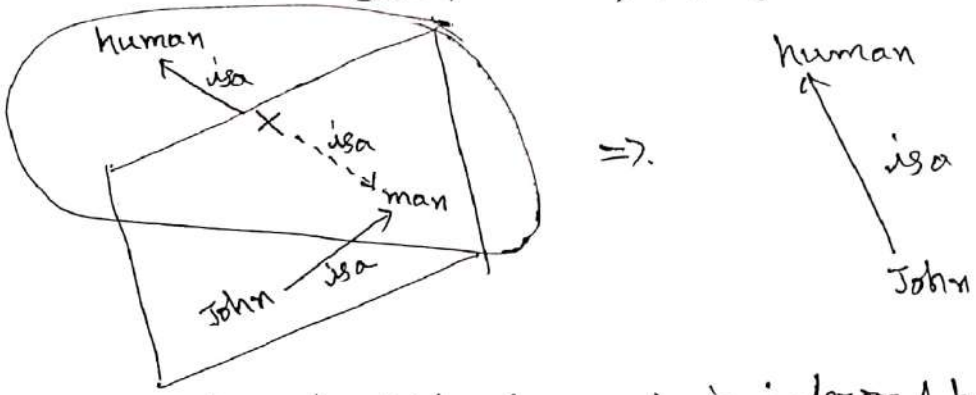
→ Forward reasoning inference mechanism
→ Backward reasoning inference mechanism.

The Forward reasoning inference mechanism (bottom-up App) in clausal logic derives new assertion from old ones. Given an ESNet, apply the following reduction (resolution) using modus Ponen rule of logic i.e if given $A \leftarrow B$ and B, then conclude A.

Ex: Consider the following set of clauses.

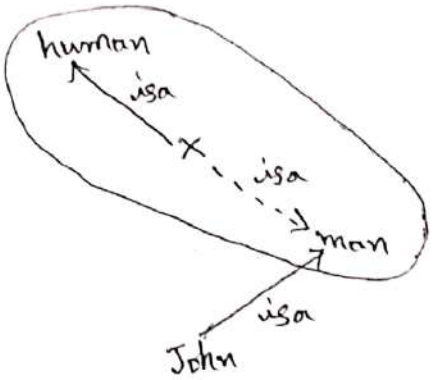
isa(X, human) ← isa(X, man), isa(John, man)

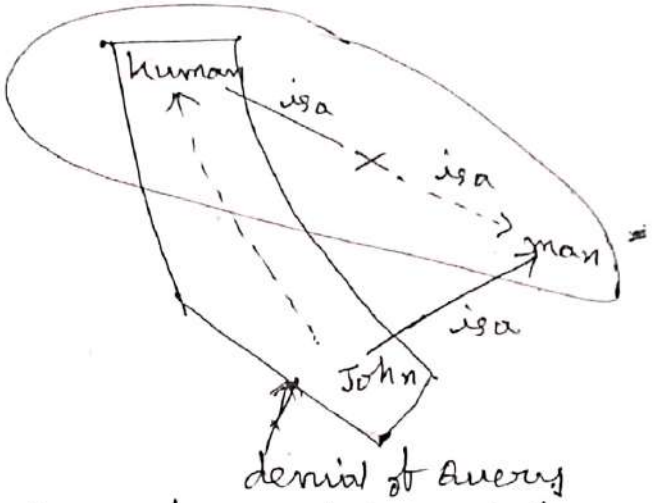
we can easily derive that isa(John, human) holds true. Now let us derive or inference isa(John, human) using ESNet representation



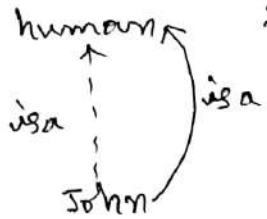
The new assertion isa(John, human) is inferred by elimination of assertion and their denial links with the help of Substitution. Here isa(X, human) is an assertion link and isa(X, man) is denial link. Here the contradiction is enclosed in a rectangular box.

In Backward reasoning inference mechanism (top-down App) we prove the query from the set of clauses using resolution - refutation method in clausal logic. we can prove a conclusion from a given ESNet by adding the denial of the conclusion to the network and show that the resulting set of clauses in the network gives contradiction. It is similar to proof by resolution refutation in clausal form of logic. Consider the above set of clauses and prove isa(John, human) using the same network.





After adding denial link in ESNet, we get the reduction in ESNet as shown below, by elimination of assertion and their denial link with the help of appropriate substitution.



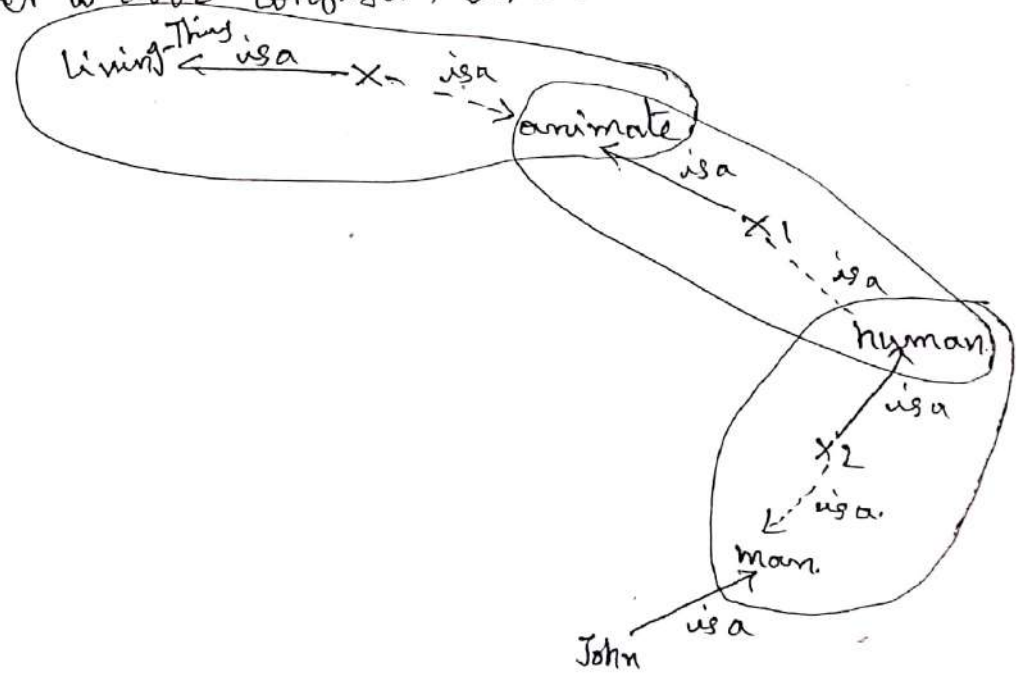
X = John.

Contradiction of empty network is generated. Hence isa(John, human) is proved.

Ex: Consider the following example, represented in clausal form

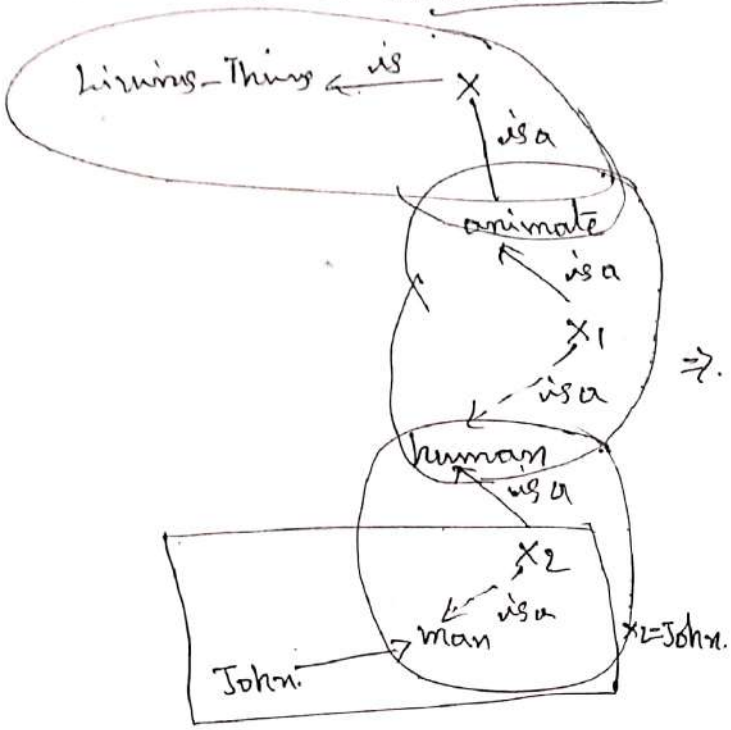
isa(X, living_thing) ← isa(X, animate)
isa(X, animate) ← isa(X, human)
isa(X, human) ← isa(X, man)
isa(John, man)

The ESNet representation is shown below, since the variables in different clauses are different, we choose different variables in ESNet to avoid confusion in the representation.

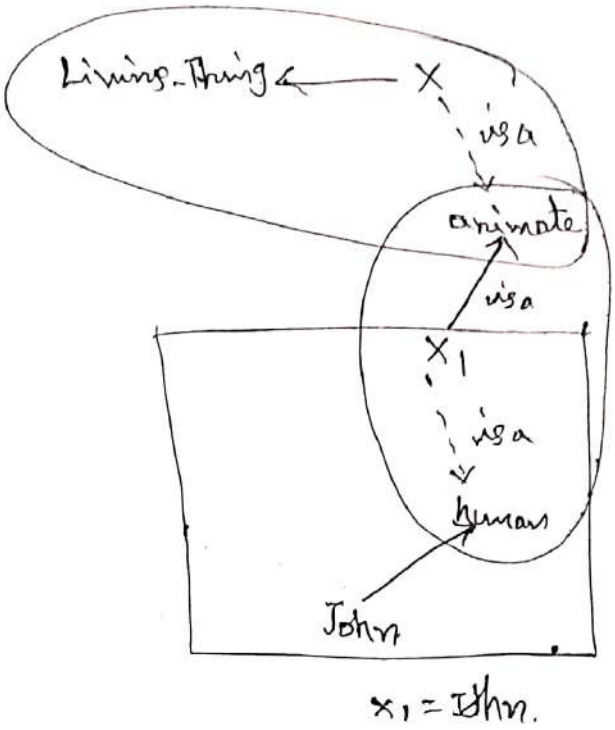


Forward Reasoning Inference:

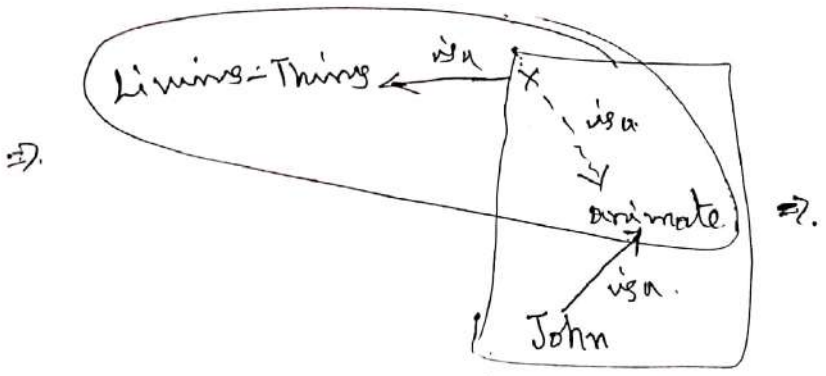
The new assertion John is human can be inferred as show below.



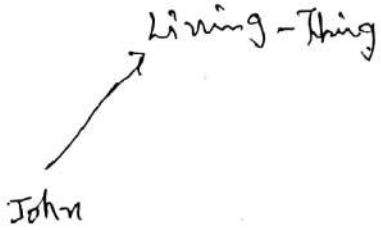
John is human



John is animate.



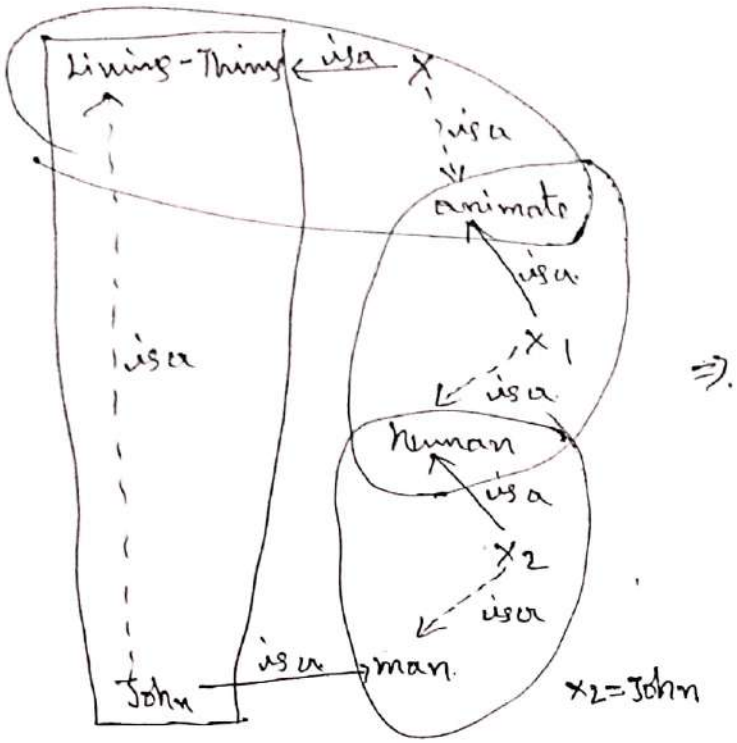
John is animate is derived.

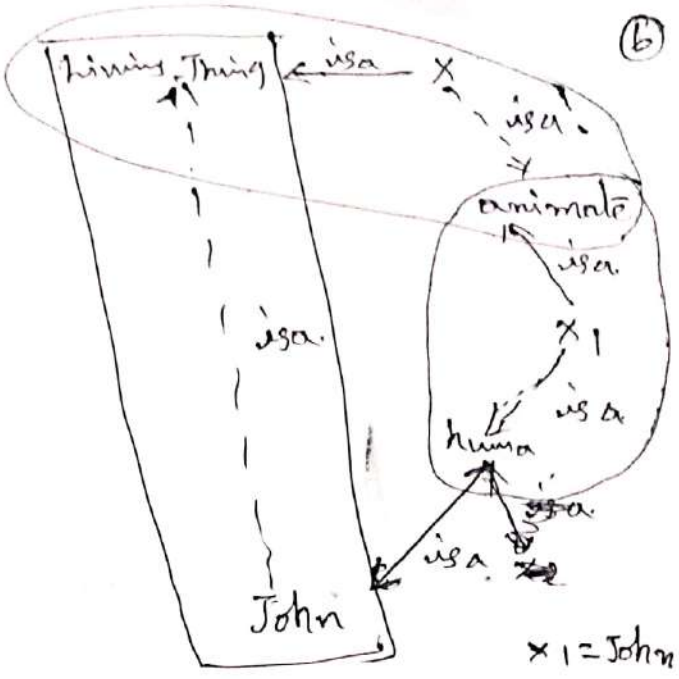


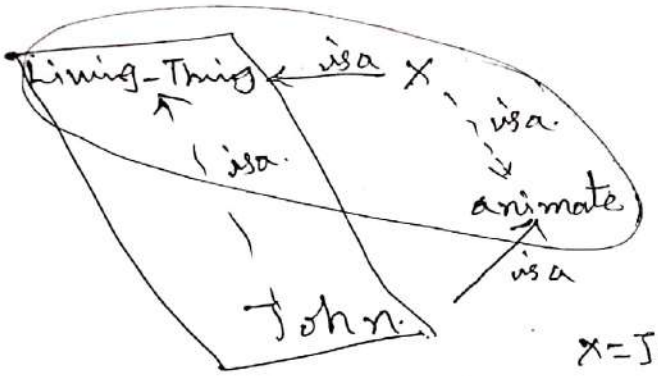
Backward Reasoning Inference: Solve the query isa(John, Living_Thing)

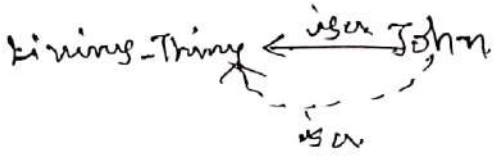
In this, we add the denial of the fact that "John is not a Living_Thing" in ESNet and see if we can derive inconsistency (or) empty Network.

The reduction in the network by elimination of assertions and their denial links with the help of appropriate substitutions.







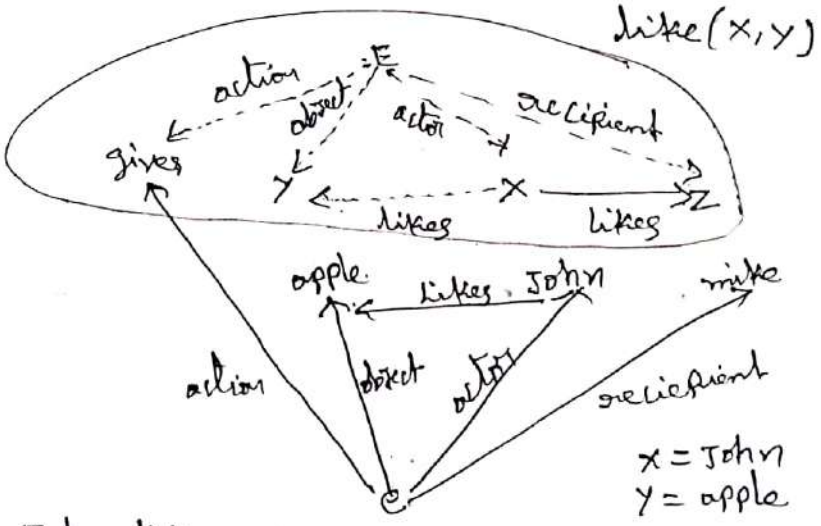


EX2: The sentence. "Anyone who gives something he likes to a person likes that person also. John gives an apple to mike. John likes an apple'. Expressed in clausal form and ESnet form.

clausal Representation:

likes(X,Y) ← action(E, give), object(E, Y), actor(E, X), recipient(E, Z)

action(e, give),
object(e, apple),
actor(e, John)
recipient(e, mike)
like(John, apple)

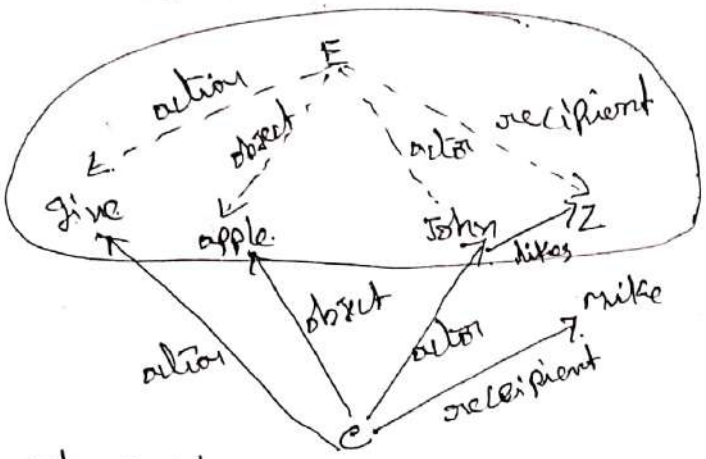


→. ESNet Representation

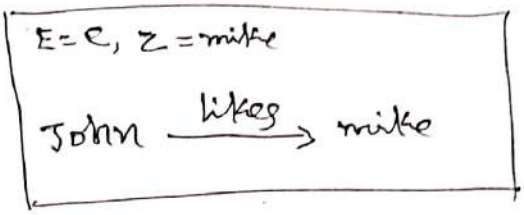
Now we need to prove John likes mike ie likes(John, mike).

Forward Reasoning Inference:

Here we have to reduce the network by resolving clauses till we obtain the desired assertion. we can easily eliminate assertion and its corresponding denial with likes links by unifying y = apple and x = John. The reduced ESNet is



⇒. The network further reduced to the following conclusion by unifying E = e and Z = mike

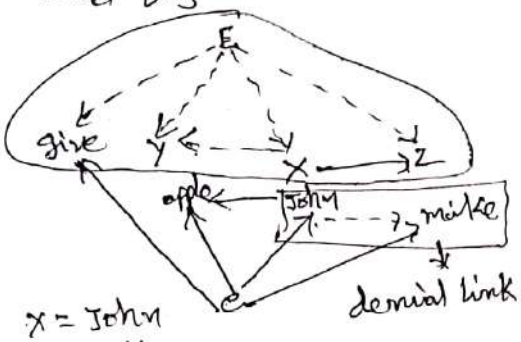


y = apple, x = John

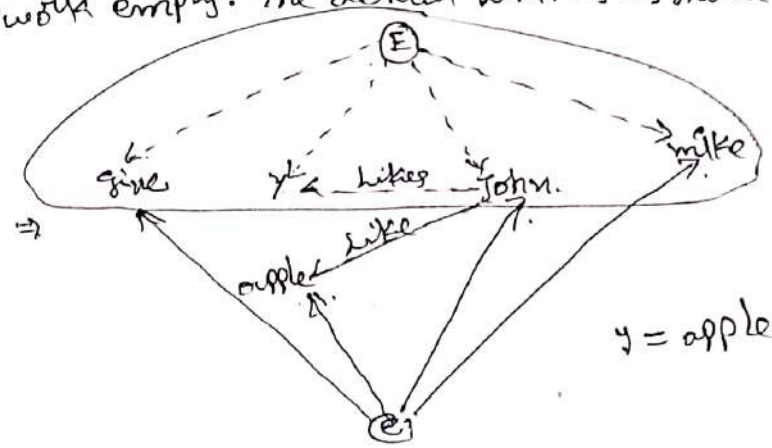
Here, we can clearly see that a link likes b/w John and mike is deduced after eliminating all links which are assertions and denials. we are left with the final assertion likes (John, mike).

Backward Reasoning Inference:

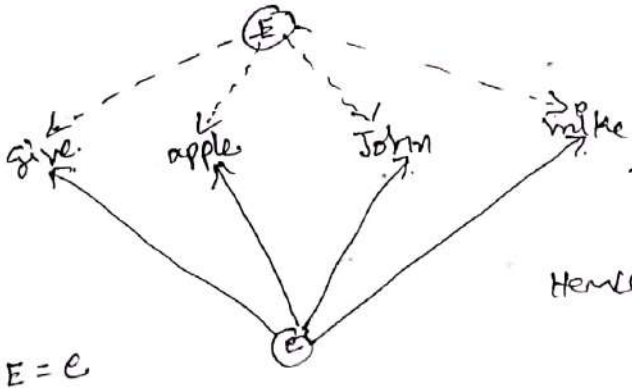
In order to prove likes (John, mike), add the denial link to the network and try to reduce the network empty. The denial link is as shown below



x = John
z = mike



y = apple



E = e

⇒. we get empty network.

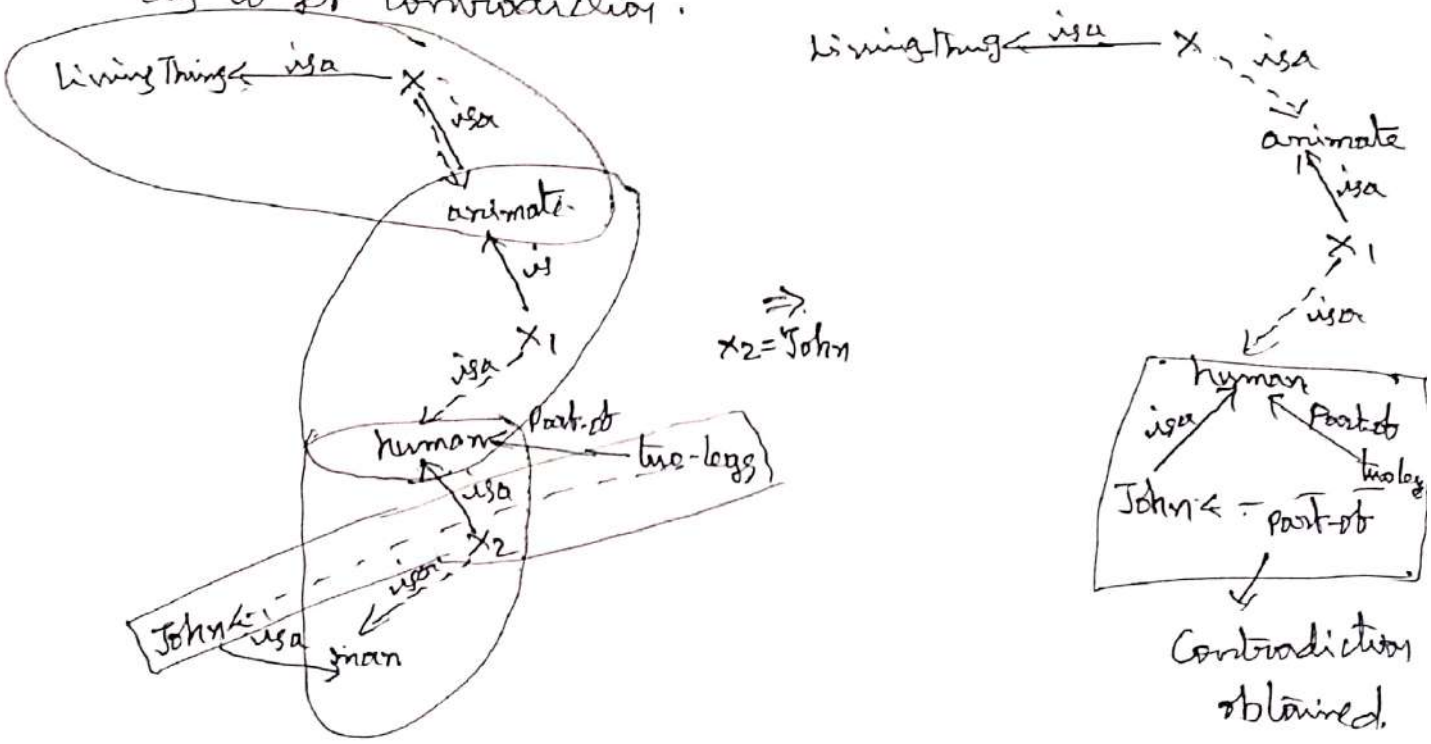
Hence the query likes (John, mike) is proved using Backward Reasoning Inference.

Inheritance:

In semantic network, lower level nodes in isa hierarchy inherits properties from higher level nodes unless the properties are redefined in the node itself. Consider the following logic.

isa(X, living_thing) ← isa(X, animate)
isa(X, animate) ← isa(X, human)
isa(X, human) ← isa(X, man)
isa(John, man)
Part-of(human, two_legs).

These logic clauses are represented as ESNet, we can show that John inherits the property of having two_legs from human. In order to show John has two legs, we add a denial link of John has two-legs to the network as shown below. Now we use Backward reasoning inference mechanism and try to get contradiction.



We notice that John is human who has two-legs is our assertion and John has two-legs is a denial. So, we get a contradiction. Thus, we have proved John has two_legs using Backward Reasoning method.

Implementation:

Implementation of ESNet can be done in any programming language or using a tool which facilitates implementation of semantic networks. The clauses are represented either explicitly, by adding them to the network. The arguments may be represented either explicitly or implicitly and by pointers to their parents.

5. Knowledge Representation using Frames:

The ideas about frames and frame systems and how they can be used for the process of knowledge representation were first introduced by Marvin Minsky (1975). Frames are an extension to semantic nets. Each node of a semantic net is represented by a frame.

A frame may be defined as a data structure, used for representing a stereotyped or situation. It Consists of collection of attributes or slots and associated values that describe some real-world entity. Frame Systems form a powerful way of encoding information that supports reasoning. Frames may be considered to represent the ways of organizing as well as packaging knowledge in a more structured form.

Frames are similar to the Concept of class of object-oriented paradigm. Frame Consists of attributes or slots described by attribute-value pairs <slot-name, value>. The value of slot may be primitive, such as a text string, Constant or an integer or it may be another frame. Frames may contain triggers for checking consistency or obtaining updates of other slots. Therefore, frames are basically a machine-usable formalization of Concepts. The general structure of frame is given below.

Structure of a Frame.

frame_name
slot - filler
default - values
Constraints on values within the slots of a frame
pointers to other frames
ako (a-kind-of or sub class)
inst (instance)
instantiation procedure
inheritence procedure
default inference procedure
Triggers

A frame may contain as many <slot-filler> pairs as required to describe an object. Each slot may contain one or more fillers (Facet) from the list given in the Table.

Facet Name		Description
value	—	value of the slot
default	—	Default value of the slot and it may be redefined by lower classes
range	—	The range of integer or enum values that a slot have
demons	—	procedural attachments such as if-needed, if-deleted, etc
other	—	Contain rules, other frames, semantic net, or other info

A class frame has certain default values which can be redefined at lower levels. A class frame possesses an actual value, then decedent frames can't modify that value.

Ex: A hospital frame has been defined in Table.

Slot_Name	Facet Name	Facet value
F_Name	value	Metro-hospital
Country	default	India
Phone_No	default	23456778
Address	default	abc

Frames in a network are connected using the links discussed below.

→ AKO: This link connects two class frames, one of which is a kind of the other class. Ex: child-hospital is kind of hospital.

→ Inst: Connects a particular instance frame to class frame.
Ex: AIIMS is an instance of class frame hospital

→ Part-of: Connects two class frames one of which is constraint in the other class. Ex: Ward is part-of the class hospital

Ex: Define a network of frame system for hospitals as described below. The following are descriptions of some frames of the network.

Hospital Frame (Root of the network)
F-Name: hospital
Country: value-India
Phone-no: Default-23456778
Address: Default- New Delhi
Director: Default-XYZ
Labs: Lab (Lab Frame)
Wards: Ward (ward Frame)
Doctors: Doctor (Doctor Frame)

Child Hospital Frame
F-name: child-hospital
AKO: hospital (Hospital Frame)
Age: (range-[0-12]), (rule_to-added)

Heart Hospital Frame:
F-name: heart-hospital
AKO: hospital (Hospital Frame)

Lab Frame
F-name: Lab
Part-of: hospital (Hospital Frame)
Pathology: Pathology (pathology Frame)
X-Ray: X-Ray (X-Ray Frame)

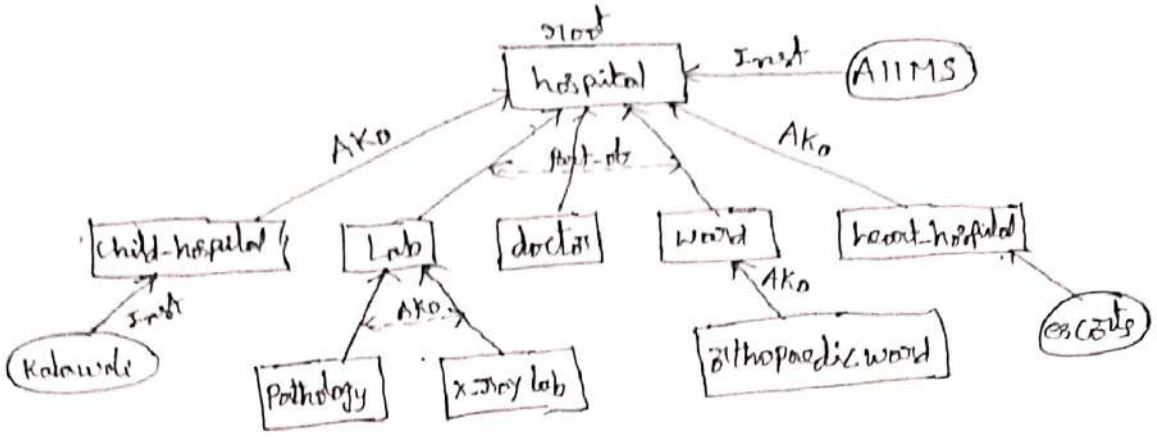
Ward Frame
F-name: ward
Part-of: hospital (Hospital Frame)
Ortho: Orthopaedic (Orthopaedic ward fr)

Doctor Frame
F-name: doctor
Part-of: hospital
Qualification: default-MBBS

Pathology Frame
F-name: Pathology
Part-of: Lab (Lab Frame)
Incharge: default-XYZ
Types of test: ----

X-Ray Frame
F-name: X-ray
Part-of: lab (Lab Frame)
Incharge: (default-PQR)

Orthopaedic Ward Frame
F-name: Orthopaedic
AKO: ward (Ward Frame)



→ A frame can be an instance of some frame and a part of another frame. So, it can have both Inst and Part-of links.

→ A frame can be a-kind-of frame of one frame and a part of another frame. So, it can have AKO and Part-of links

→ However, it is not possible to have a frame which is both an instance and a-kind-of some class at the same time. So, the links Inst and AKO in a frame are not possible.

Inheritence in Frames:

Inheritence is defined as a mechanism used for passing knowledge from one frame to other frames, from general to specific frame.

Attachment of Demons:

Demons allow us to invoke rules within the frames. These can be attached with a slot along with other required facets.

Ex: if-needed may be used for monitoring the behavior of systems
if-added may be used to validate data, when data added in value sl

Another use of attaching demons is that they allow dynamic information retrieval and storage. Here, we may get the value directly from slot or may have to dynamically calculate value of the required slot on the basis of other values

Implementation of Frame

Frames can be implemented using OOP languages. The concept of class in OOP can be directly used to code frames, that has the concepts of subclass (AKO) and containment (Part-of link) of a class within another class. Instances of the frames can be declared as objects of the relevant classes. slots can be defined as variables and procedural attachment as methods.

Advanced Knowledge Representation Techniques

These representations are inadequate for representing knowledge required for Core Natural Language understanding applications. We require structures to hold deep sense and meaning of the sentences. For capturing semantics of the knowledge, there are various approaches.

1. Conceptual Dependency (CD) Theory:

CD is a theory of deep semantics for concepts used for natural language understanding. It mainly focuses on the concept and meaning rather than on syntax or structure. All propositions that describe events are made up of conceptualizations, that comprise an actor, an action and a set of cases that depends on that action. In CD different words and structures having same meaning should also have the same representation. The types of conceptual roles do not correspond to syntatic roles, rather than words are broken into their conceptual parts and are then placed in a meaningful representation with respect to their conceptual roles. Similarity in meaning of words are handled by the concept of the Primitive Actions. The primitives are elements that can be used in varied combinations to express the underlying meaning of a given word.

Conceptual Dependency representation use eleven primitive acts as the basis for most activities. A large set of states and a set of conceptual roles can be expressed using this minimal set of primitive actions.

Conceptual Primitive Actions

Most of the verbs, describe actions in natural language have been categorized under the following eleven primitive actions. Sometimes one verb may fall into two categories, then most relevent in the context may be choosen at the time of Interpreting Sentence. The primitive ACTIONS are listed as follows.

1. ATRANS 2. PTRANS 3. PROPEL 4. MOVE 5. GRASP
6. INGEST 7 MTRANS 8. MBUILD 9. EXPEL 10. SPEAK
11. ATTEND

1) ATRANS: Transfer of an abstract relationship such as possession, control or ownership. It requires an actor, object, and recipient.
EX: give, take are represented by ATRANS

2) PTRANS: Transfer of the physical location of an object that requires an actor, object and direction.
EX: The verb go or walk are represented by PTRANS.

3) PROPEL: Application of physical force to an object
EX: PUSH, PULL, throw, are represented by PROPEL act.

4) MOVE: Movement of a body part by it's owner. EX: KICK, hit, throw.

5) GRASP: Grasping of an object by an actor EX: catch is GRASP.

6) INGEST: Ingesting of an object by an animal; It requires an actor, object and direction. EX: Eat or drink, breathe.

7) MTRANS: Transfer of mental information b/w animals or within an animal itself EX: tell, speak and sing, read, remember.

8) MBUILD: The construction of new information from old information within an actor to create new thoughts internally.
EX: decide, describe, imagine, consider and answer.

9) EXPEL: Expulsion of something from the body of an animal.
EX: weep, cry, sweat, EXPEL tears / moisture

10) SPEAK: Producing Sounds. EX: Say, tell, sing.

11) ATTEND: Focusing of a sense organ toward a stimulus
EX: Listen, watch, see, and Look.

All these primitives are classified in the broad categories as

Category	Primitive Acts
State change primitives	ATRANS and PTRANS
Mental action primitives	MTRANS and MBUILD
Senses (eyes, ears, tongue)	ATTEND and SPEAK
Physical Action primitives	INGEST, MOVE, GRASP, PROPEL, EXPE

CD structure also helps in disambiguating ambiguous words.

EX: The verb 'take' in take a medicine or Juice would be INGEST rather than ATRANS. Here the knowledge about how words fit together in english are used to make such distinctions. The correct Conceptual Structure is assigned by asking a series of questions about the nature of object around it.

Conceptual Category:

CD provides a set of building blocks used to make representations Building blocks are the set of allowable dependencies among the Conceptualizations for different events. There are four Categories from which dependency structures can be built.

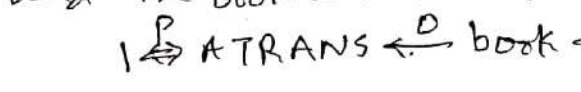
ACT – Actions { representing verb }
PP – objects { Picture producers or noun/pronoun }
AA – modifiers of actions { action aiders or adverbs }
PA – Modifiers of PP { picture aiders or adjectives }

The relationship b/w concepts are called dependencies. The main concept of a clause is a two-way dependency b/w a PP and an action. It is important to note that actions are broken down into sequence of primitive ACTs.

A set of rules describe the syntax of the conceptual level. There exists a dictionary of ACTs, specifies different meanings with its Conceptual structure for each verb.

Ex: The CD representation for sentences such as.

" I took a book from the man", " The Man gave me a book" and " The Book was given by man to me". have same intended meaning



Here we use special notations, to explain as we move ahead.

→ Arrows indicate directions of dependency
⇒ Double arrows indicates two-way links b/w actor & action
→ O – for the object case relation
R – for the recipient " "
d – destination
p – Past
f – future
t – Transition
ts – start Transition
tf – finish Transition
k – Continuing
c – Conditional

Rules for Conceptualization Blocks in CD

Dependency structures can serve as components for larger dependency structures. The dependencies among conceptualization correspond to semantic relations among the underlying concepts. Following is the list of allowable structures & the rules.

Rule 1: Representing relationship between actor and the event (ACT) caused by him & her is called a two-way dependency. The letter 'p' in the dependency link indicates that the event occur in Past

Ex: ~~John~~ PP $\leftrightarrow$ ACT

1) John ran — John $\overset{p}{\leftrightarrow}$ PTRANS
2) Mary cried — Mary $\overset{p}{\leftrightarrow}$ EXPEL.

Rule 2: Representing relationship b/w an ACT and PP (object) of the ACT is shown by the direction of an arrow towards the ACT.

ACT $\leftarrow$ PP

1) John gave a book to Mary — John $\leftrightarrow$ PTRANS $\leftarrow$ book.
2) Mary kicked the ball — Mary $\leftrightarrow$ MOVE $\leftarrow$ ball.

Rule 3: Shows relationship both ways between two PPs. one PP belongs to the set defined by the other PP. PP $\leftrightarrow$ PP

1) John is a doctor — John $\leftrightarrow$ doctor
2) Mary is a teacher — Mary $\leftrightarrow$ teacher.

Rule 4: shows relationship between two PPs. one of the PP provides a particular kind of information about the other PP. The most common types of information to be encoded in this ways Possession (poss-by) and location (loc): PP $\leftarrow$ PP

1) John's car — car $\overset{\text{poss-by}}{\leftarrow}$ John
2) Mary is in Delhi — Delhi $\overset{\text{loc}}{\leftarrow}$ Mary

Rule 5: Shows relationship between a PP and a PA that is asserted to describe it. PA represents the states of PP such as height, weight, health on numeric scales and has both ways arrows. PP $\leftrightarrow$ PA

1) John is fat — John $\leftrightarrow$ weight (>50)
2) Mary is poor — Mary $\leftrightarrow$ income (< 2000)

Rule 6: shows relationship between a PP and an attribute that already has been predicated of it. The direction towards PP PP $\leftarrow$ PA

1) Smart John — John $\leftarrow$ Smart
2) Charming Mary — Mary $\leftarrow$ Charming

Rule 7: Shows relationship between an ACT and its Physical source and destination locations of ACT. Here 'd' indicates source and destination case relation. ACT ←d→ PP (destination), PP (source).

Ex: John went to school from home. John ⟺P PTRANS ←d→ School, Home

Marry traveled from Delhi to Bombay by train. Marry ⟺P PTRANS ←d→ Bombay, Delhi; PTRANS ←I Train

Rule 8: Shows relationship between an ACT and its source and the recipient of ACT. The letter 'R' indicates recipient case relation. ACT ←R→ PP (to), PP (from)

1) John gave a book to Mary. John ⟺P ATRANS ←O book ←R→ Mary, John

2) Mary told story to John. Mary ⟺P MTRANS ←O story ←R→ John, Mary

Rule 9: Shows a relationship b/w an ACT and the instrument using which it is performed. Here 'I' represent instrument and 'DO' represents act done by actor using instrument.

Ex: Mike ate custard with spoon

Mike ⟺P INGEST ←I Spoon; INGEST ↑O Custard

Rule 10: Shows a relationship that describes the change in state b/w PP and a state in which it started and state, it ended.

PP ⟸ PA, PP

Ex: The trees grow. Trees ⟸ size < C, size = C

John is sick. John ⟸ health (< normal), health = normal.

Rule 11: Relationship b/w one Conceptualization and other Conceptualization that causes it. Here 'if X then Y' (or) 'Y' is a Consequence of 'X'.

⟸ {X}

C ⇑

⟸ {Y}

Ex: Terrorist shot dead John

Terrorist ⟺ PROPEL ←O bullet ←d→ John, gun

⇑

John ⟸ dead, alive

Smoking is harmful for health

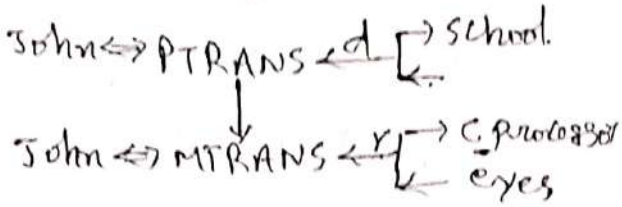
one ⟺ INGEST ←O Smoke ←R→ one, cigarette

C ⇑

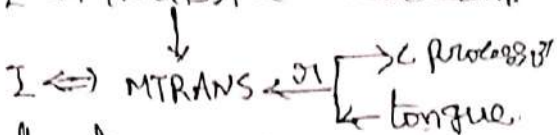
one ⟸ health < X, health = X

Rule 12: Shows relationship b/w one conceptualization with another that is happening at the time of the first. Event 'y' is happening while event 'x' simultaneously.

Ex: While going school John saw a snake.

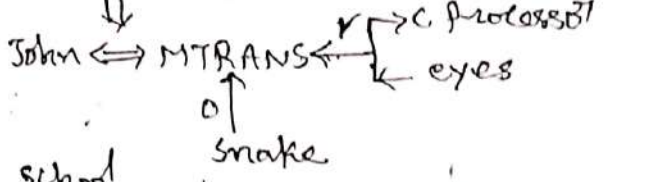


I enjoyed while eating ice-cream

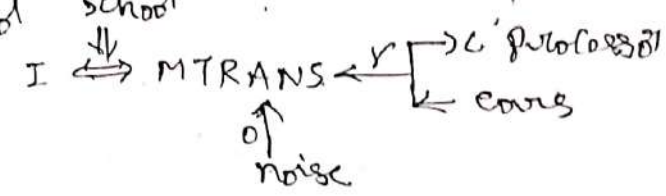


Rule 13: shows a relationship b/w conceptualization and the place at which it happened.

Ex: John saw a snake in the house



I heard noise in the school



Conceptual parsing:

It required for generating CD representation from source sentences in natural language. The main steps involved are.

→ Syntactic processor extracts main verb and noun along with syntactic category of the verb from the sentence.

→ Conceptual processor then makes use of verb-ACT dictionary. once the correct entry from the dictionary is chosen, CD processor analyses the rest of sentence looking for arguments for empty slots of the verb.

Inferences Associated with primitive ACTs:

The meaning of the concept consists of the meanings of the primitive itself and its associated inferences. Inference is a conceptualization that is inferred by another conceptualization.

Ex: From a sentence "John killed Mike", we can infer that "Mike is dead".

The words in the dictionary are maintained with the following information.

→ Syntactic Category with all possible attributes and properties
→ The verbs with all possible senses and primitive acts.
→ With each primitive Acts, Inference rules are stored.

Usefulness of CD Representation:

→ Implicit information is deduced and stored as a part of CD representation for future use. It is achieved by applying a set of inference rules associated with specific information. once knowledge is generated, it can be repeatedly used again & again
→ CD representation is a good model for representing simple events. In CD knowledge in the event is decomposed into very low level primitives.
→ In CD representation, unspecified elements of the representation of information can be used for the understanding of later events.

problems with CD Representation are

→ It is difficult to construct original sentence from its corresponding CD representation.
→ It is difficult to represent general knowledge, because this is based on representations of events as well as related info.
→ Rules are to be carefully designed for each primitive action in order to obtain semantically correct interpretation.
→ Many verbs may fall under different primitive ACTs, it becomes difficult to find correct primitive in the given Context.
→ CD representation becomes complex, requiring lot of storage for many simple actions.

2. Script Structure: Schank and Abelson introduce scripts in 1977.

→ Scripts are structures representing procedural knowledge by describing a set of events in a particular situation, which could be expected to be followed from one to another event.

→ It consists of set of slots, which contain default values along with some information about the type of values they may contain

→ It looks similar to frames, except the values of the slots must be ordered and have more specialized roles.

→ Script structure allows individual to make inferences.

→ Each script contains the following main components.

Entry conditions: must be satisfied before events in the script can occur.

Results: conditions will be true after events in script occur.

Props: slots representing objects involved in the events

Roles: persons involved in the events

Track: specific variation on more general pattern in the Script

Scenes: The sequence of events that occur. Events are represented in conceptual dependency form.

Ex: To better understand the concept of Script, let us consider example of developing a Script for "going to theater to see a play".

→ It involves the following scenes.

→ Going to theater
→ Buying ticket
→ Going inside hall and sitting on a seat
→ watching play
→ Exiting from Theater.

Various components of Script for going to Theater are.

Entry conditions: → P wants to see a play → P has money.

Results: → P saw play → P has less money.

Props: → Ticket, seat, play.

Roles: → Person, → Ticket distributor, → Ticket checker

Entire script have five scenes for Event - Going to Theater.

~~Script~~
Scene 1: Going to Theater
→ P PTRANS P into Theater
→ P ATTEND Eyes to ticket Counter

Scene 2: Buying a ticket
→ P PTRANS P to ticket Counter → P MTRANS to TD(Ticket distri
→ TD ATRANS ticket to P

Scene 3: Going inside hall and sitting on a seat
→ P PTRANS P into Hall → TC ATTEND eyes on ticket Pass by P
→ TC MTRANS to P → P PTRANS P to seat → P MOVES P to sit

Scene 4: watching a play.
→ P ATTEND eyes on play → P MBUILD (Good moments) from play.

Scene 5: Exiting
→ P PTRANS P out of Hall and Theater.

~~Text~~ A particular script is to be applied based on its significance Into the topic is an important, then the script should be opened. Into a topic is just mentioned, then a pointer to that script could be held. Heuristics can be applied to get the 'significance value.

If events follow known track, we can use scripts to answer implicit or explicit questions. Different tracks may allow different outcomes of scripts.

Ex: play was cancelled because of unseen situations
could not see play as it was full and no tickets available
came out of the hall as bomb exploded.

→ The main advantage of script structures are, they are capable of predicting implicit events and that a single interpretation may be build up from a collection of observation
→ The disadvantages of script is, they are more specific and less general than frames. It is not suitable to represent all kinds of knowledge, because scripts are ~~inflexible~~ Inflexible.

11. CYC Theory

The CYC is a theory designed for describing the world knowledge to be useful in AI applications and more specifically in natural language understanding similar to CD Theory. CYC is used for capturing Commonsense knowledge from 100 randomly selected articles in the Encyclopedia Britannica. The CYC structure Contains representations of events, objects, attitudes, space, time, motion, etc and huge in structure. Some of the reasons for large ~~KB~~ Knowledge Base are as follows.

Brittleness: Specialized Knowledge base are brittle. It is hard to encode new situations and there is degradation in performance.

Form and Content: Commonsense strategies could point out where difficulties in Content may affect the form and temporarily focus on content of KBs rather than on their form.

Shared Knowledge: Small Knowledge Base System should allow greater Communication among themselves with Common Bases and assumptions.

Initially the approach taken by CYC was to encode 10 millions of facts that make commonsense knowledge and then make itself a learning system. Machine learning methods are employed to acquire knowledge. Special language called CYCL generalizes the notation of Inheritence, so that properties Can be inherited along any link rather than only ⊆ "is a" and "Instance" links.

5. Case Grammars:

It was proposed by the American linguist Charles J. Fillmore in 1968. for representing linguistic knowledge that removed the strong distinction b/w syntactic and semantic knowledge of a language. He initially introduced six cases (roles) AGENTIVE(Agent), OBJECTIVE(object), INSTRUMENTAL(Instrument) DATIVE(covers Experience), FACTIVE(covers result of an action) and LOCATIVE(Location of an action).

The goal of case grammer theory was to extract deep meanings of the sentences and express in the form of cases mentioned above. Different syntactic structures with same meaning would map to similar structure.

Ex: The door was broken by John with hammer,
using hammer John broke the door.

The case frame contains semantic relation rather than syntactic ones. Sentences have same syntactic parse structure but have different meanings are stored differently ~~meanings~~ in case frame.

Ex: Mother baked for one hour / cake baked for one hour will have same syntactic parse structure (NP VP PP), But case grammer show the difference. In the first 'mother' is agent in the second sentence, 'cake' is object of baking action.

There are two levels of every sentence structure.

Surface structure is basically a spoken/written sentence and.

deep structure: gives underlying meaning of the sentence.

A verb may have more than one meaning and represented using two different representations in a case grammer.

Ex: John runs an Industry / John runs in the field.

→ Agent:
Noun phrase of a sentence fill the Agent case, if it describes the instigator of the action

→ object: The noun being acted upon by action is used to fill up object case. Ex: door broke, John broke the door.

→ Dative: Refers to object of an action that is animate.
Ex: John killed Mike, Mike is Dative case.

→ Experiencer: Subject of sentences having intransitive verbs ie (verb with no object) is called experiencer.
Ex: John Cried, Mike laughs, Here John & Mike are Experiencer

→ Beneficiary: It is an animate person for whom a certain event is performed for its benefit.
Ex: I gave an apple to Mike, Mike is beneficiary.

→ Location: Refers to place of location
Ex:, The man was killed in the garden — garden is location

→ Instrument: used to perform some action
Ex: John ate ice-cream with spoon.

→ Co-Agent: when two people perform some action together, then second person fills up Co-Agent case.
Ex: John and Mike lifted Box — Here Mike is Co-Agent.

→ Time: Containing time when the action was performed.
Ex: John went to market yesterday at 11 o'clock.

→ Tense (present, past, future) of the sentence can be stored in order to generate surface structure sentence.
Ex: I am going to school → Tense is present Continuous

Ex: Let us generate case frame for a sentence using case structure.

John gave an apple to Mike in the kitchen (or)
~~Mike in the kitchen~~
Mike was given an apple by John in the kitchen.

Cases	values
Action	Give
Agent	John
objective.	Apple.
Beneficiary	Mike.
Time	Past.
Location	kitchen.

Inference rule with the event of giving can be attached as follows

6. Semantic web:

Current web technologies are based on the that, humans can read, understand and operate information present on web. The search engines can search required documents, but can't understand the required information stored in the document. because it is not in the structured form.

The semantic web is an extension of traditional web that provides a common frame work allowing data to be shared and reused across different applications. The semantic web adds meaning or semantics to web content in order to make it easier to be used by both humans and machines. The semantic web describes all the data and information in the form of machine readable form called Meta Data. It allows the system to process the data automatically. so, The semantic web having data and documents on the web, so that machines can process, transform assemble, and even act on the data in useful ways.

→ The semantic web comprises the XML, XML schema, RDF, RDF schema, and web ontology language (OWL).

Extensible Markup Language (XML):

It is a general-purpose specification for creating custom markup languages similar to HTML. In XML user may define his/her own tags. It is used to facilitate the sharing of structured data across different information systems via the internet.

Ex: Represent "a circular from head to faculty for a meeting" in XML, it will look like.

```
<circular>
<to> Faculty </to>
<from> Head </from>
<heading> Faculty Meeting </heading>
<body> There is a meeting of the faculty in Committee on 11/2/2019 </body>
</circular>
```

XML provides syntax for documents, but no semantics on the meaning of documents.

XML Schema: It is a language for describing the structure of XML documents, expressed in terms of constraints on the structure and content of documents. They provide structure, content and semantics of XML documents in more detail. The XML document associates with schema via markup within the XML document (or) via some external means. The purpose of an XML schema is to define the building blocks of an XML documents. It consists of following things.

→ Elements and attributes that can appear in a document.
→ elements appearing as child elements, their order & child elements
→ whether an element is empty or can include text.
→ data types for elements and attributes with default values

Ex: XML document describes a book and corresponding XML schema.

```
<book>
<title> .... </title>
<author> .... </author>
<abstract> .... </abstract>
<publisher>
        <name> .... </name>
        <address> .... </address>
</publisher>
</book>
```

XML schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
</xs:schema>
<xs:element name="title" type="xs:string" />
<xs:element name="author" type="xs:string" />
```

XML schema is similar to database schema, where we define structure consisting of attributes with their corresponding types.

Resource Description Framework:

It is basic data model to build the semantic web. It is a domain-independent model for describing resources. A resource may be name, place, constant, webpages (URL) & Physical objects can be represented by a Uniform Resource Identifier (URI). The RDF uses URI to represent data in triple structures <subject, relation, object>. subject is URI of a resource, the predicate express relation, object is another resource represented by URI. It is a simple data model for referring to objects/resources and their relationships and can be represented in XML syntax.

Ex: A webpage "http://www.cs.xUniv.ac.in" has been created by Criss; RDF for various parts of the statements are

→ The subject is the URL; "http://www.cs.xUniv.ac.in/~criss.
→ The predicate is the word 'creator'
→ The object is the phrase ("criss".

The RDF statement as follows.

<http://www.cs.xUniv.ac.in/~criss> <rel:creator> <"criss">.

RDF Schema (RDFS)

It is an extensible knowledge representation language for describing properties and classes of RDF resources, with the semantics for generalization of such properties and classes. The main RDFS constructs are.

→ Classes: "rdfs:class" - to declare a resource as a class for other resources
→ Subclasses: "rdfs:subclassof" - to declare hierarchies of classes.

Ex: property domain and range: Hierarchies of classes support inheritance from a class to its subclasses. The "rdfs:domain" and "rdfs:range" of an "rdf:property" declares the class of the subject (object) in a triple using this property as predicate.

RDF schema is a simple ontology language written in RDF that allows creation of vocabularies (ontologies) with classes, properties and subclasses/superclass hierarchies.

Ex: Every Man is a person.

foaf:man rdfs:subClassOf foaf:person.

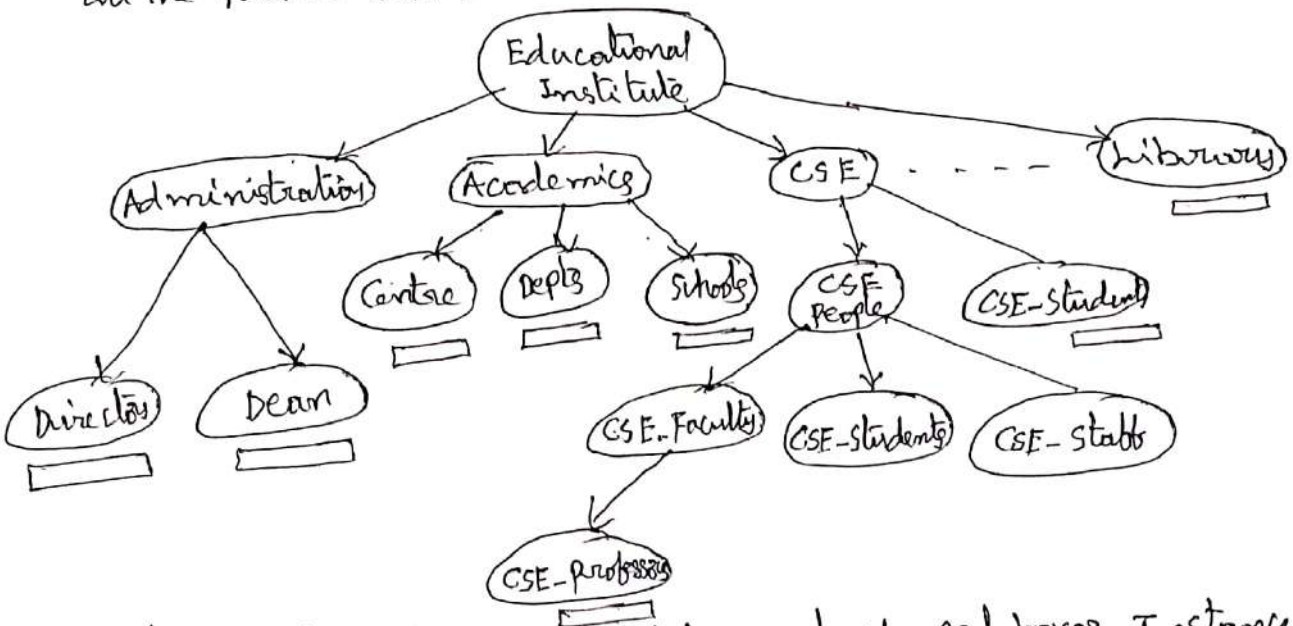
Foaf (is a friend of a friend) is describing persons himself.

6.2. ontology

It is specification of a conceptualization for objects, and relations that hold among them. ontology defines terms used to describe and represent an area of knowledge used by people, database and applications that need to share domain information. The domain is just a specific subject area, like educational, medicine etc. It includes computer-usable definitions of basic concepts in the domain and the relationship among them. It allows reuse of domain knowledge that is seperate from the operational knowledge ontology is always represented in a language suitable to its specification. Consider designing of ontology for "Educational Institutions".

→ The important terms related to educational institutions are academics, administration, faculty, professor, department, head of the department, students, and so on. Initially make list of terms without worrying about overlap b/w concepts they represent, relations among the terms.

→ Define class hierarchy structure for these terms often enumerating all the possible terms of the domain as shown below.



The class nodes are represented by oval-shaped boxes, Instances have been shown as rectangular boxes at the leaf nodes.

→ ontology is represented using ontology language such as RDF schema and OWL (web ontology language).

Web ontology Language (OWL)

It is a semantic markup language and is developed as a vocabulary extension of RDF. OWL allows does not enforce a strict separation of classes, properties, individuals and data values.

→ There are two ways of developing ontology either Hand Code the graphical structure of ontology in language such as OWL (or) use some standard tool.

Ex: An ontology description in OWL language that describing attributes of a student is shown below.

Attribute	Name	Birthdate	Email
Type	string	date	string

```
<rdf:RDF>
  <owl:DataTypeProperty rdf:about="#name"> ← name attribute
    <rdfs:domain>
      <owl:class rdf:about="#student"/> ← student class.
    </rdfs:domain>
    <rdfs:range rdf:resource="&xsd;string"/> ← Type of name
  </owl:DataTypeProperty>
  <owl:DataTypeProperty rdf:about="#birthdate"> ← Birthdate attri
    <rdfs:domain>
      <owl:class rdf:about="#student"/> ← Type of student class
    </rdfs:domain>
    <rdfs:range rdf:resource="&xsd;date"/> type of birthdat
  </owl:DataTypeProperty>
  <owl:DataTypeProperty rdf:about="#email"> ← email attribu
    <rdfs:domain>
      <owl:class rdf:about="#student"/> ← student class
    </rdfs:domain>
    <rdfs:range rdf:resource="&xsd;string"/> type of email
  </owl:DataTypeProperty>
</rdfs:RDF>.
```

→ Most RDF documents have the OWL code to be the sub-element of a "rdf:RDF" element. The file extension is either (.rdf) (or) (.owl).