Knowledge Representation

- Finding appropriate representation is a major part of problem solving

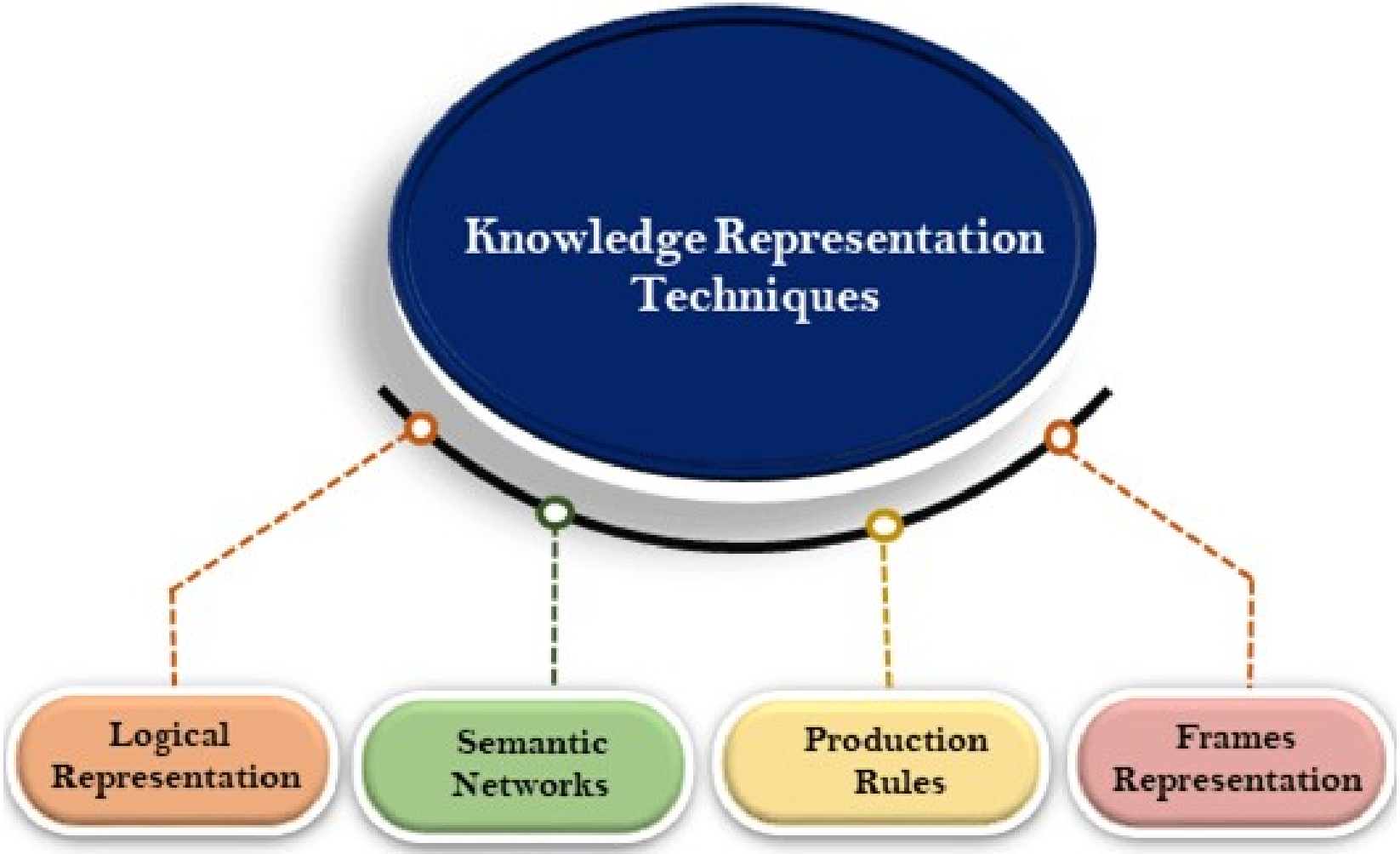
- Need to represent knowledge about the world
- Types of knowledge
  - objects
  - events
  - procedures
  - relations
  - mental states
  - meta knowledge

Closed worlds and Open worlds

- Do we know the world completely? Or is our knowledge incomplete?
- The closed world assumption says that
  - What you know is all there is to know about the world.
  - This implies that if you do not know something, it is false.
  - When we look at this idea of logic in programming language: Prolog there is this notion of negation by failure
    - If you fail to show that it is true then it must be false
  - It’s not saying that its false because you don’t know whether it is true or false
  - In the real world is not like that, there are so many things that we may not take cognisance of and which may be relevant to what we are trying to show

Closed worlds and Open worlds

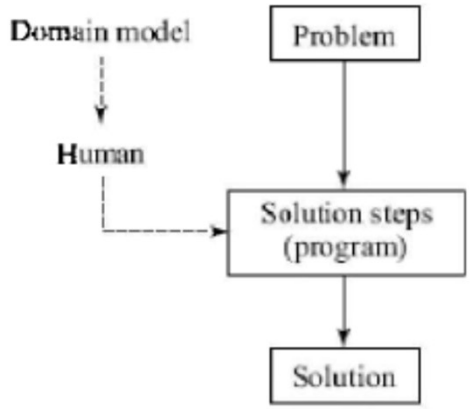
- The open worlds assumption says we have only partial knowledge of the world around us.
- This implies that the conclusions we make now may have to be withdrawn when more facts come to light.
  - This is also sometimes called as non monotonic reasoning

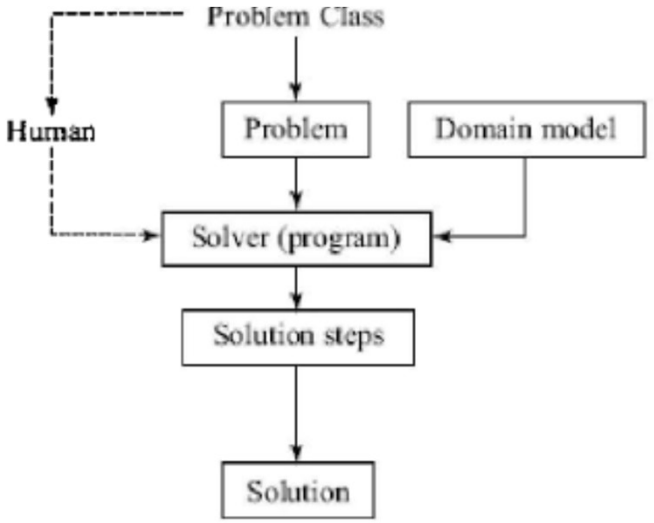

Knowledge Representation Techniques
Logical Representation
Semantic Networks
Production Rules
Frames Representation

Difference between traditional programming and AI approach

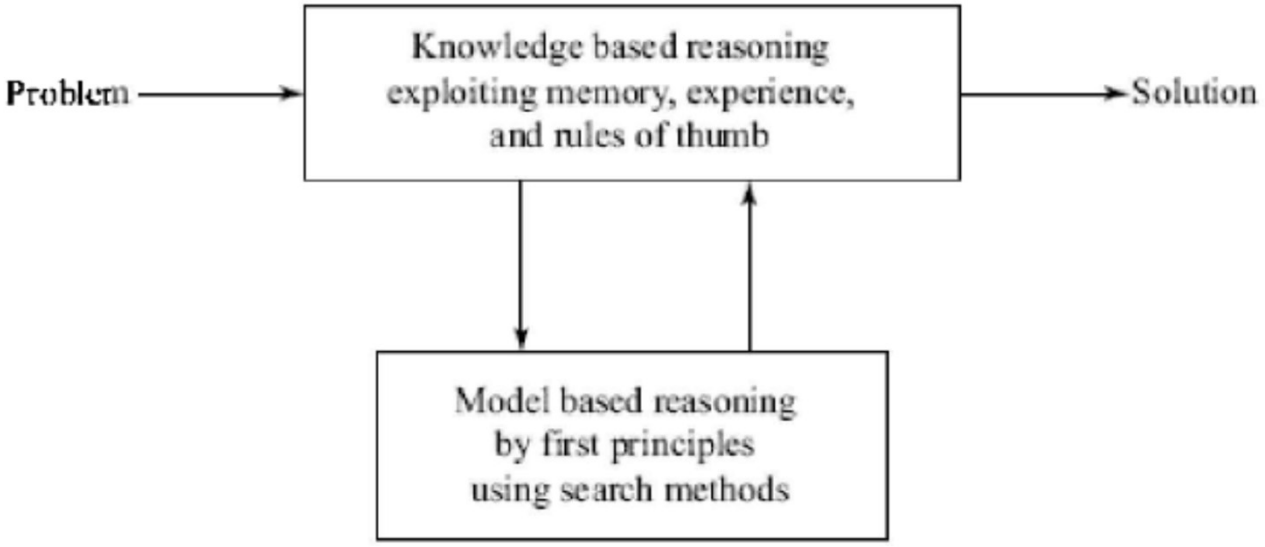
- Connection between input and output may be found at runtime and not encoded by the programmer.
- In traditional programming programmer implements the solution steps, for problems in a given domain, into a program.
- In AI programs the programmer implements a problem solving strategy in a more domain independent form.
  - AI program takes a domain knowledge and a given problem and produces the solution steps, based on the strategy it embodies.
- Both problem solving strategy and domain description call upon knowledge representation approaches.

Difference between traditional programming and AI approach





Knowledge based problem solving agent



- Human gets knowledge by experience, by books, by experience of other humans.
- But computers are not yet completely natural language enabled.
- They need crisp, unambiguous, precise formalism for representing and manipulating knowledge

Logical Representation

- Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation.
- Logical representation means drawing a conclusion based on various conditions.
- This representation lays down some important communication rules.
- It consists of precisely defined syntax and semantics which supports the sound inference.
- Each sentence can be translated into logics using syntax and semantics.

Logical Representation

- Syntax:
  - Syntaxes are the rules which decide how we can construct legal sentences in the logic.
  - It determines which symbol we can use in knowledge representation.
  - How to write those symbols.
- Semantics:
  - Semantics are the rules by which we can interpret the sentence in the logic.
  - Semantic also involves assigning a meaning to each sentence.
- Logical representation can be categorised into mainly two logics:

1.Propositional Logics

2.Predicate logics

FOL

- Domain: A set of Individuals
- Statements in logic (Predicate): relations about those individuals
- From the perspective of knowledge representation the question is what should those individuals in the domain be?
- We cannot represent the physical world as seen by physics
  - That is made of some fundamental particles
- We need to represent concepts in our cognitive models
  - A person, for example, Socrates
  - But if Socrates is an individual in the domain
    - What about the right hand of Socrates?
    - If the right hand another individual? What about his index finger?
- What about concepts like water, air, and sky?
  - These are not individuals (it is not clear whether they are individuals)
  - Or even categories
  - What does a “glass of water” represent?

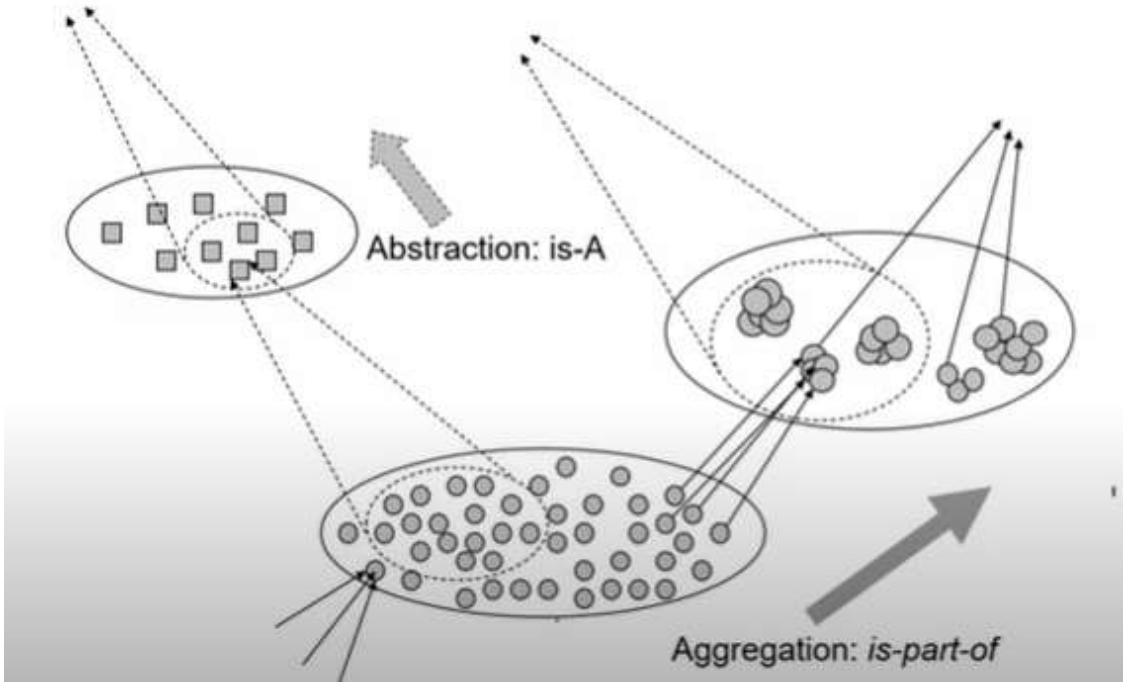
Aggregation and Reification (Abstraction)

- Individuals in our cognitive models are not the individual elements in the domain- which are the elementary particles.
- Instead we invent our own concepts in our heads
  - People, trees, dogs, nations, organizations, football teams, rivers etc.
- We also think of concepts like length and colour.
  - Are these individuals in some domain
  - These are all abstract or reified composites that we think of as individuals
  - When we say “the rose is red” we mean an individual rose
  - But when we say “violets are blue” we mean categories
  - We also think of time, events, actions
- Complex entities have some simple entities.
- In our conceptual domain (that is different from physical domain)

Hierarchies of Structured Knowledge

- Two types of hierarchies in structured knowledge :
  - Aggregation hierarchy
    - Is-part-of: Wheel is part of car…
    - Tells us how different components come together to form a larger abstract (physical) entity like car is made up of steering, wheels, seats etc. wheels are made up of ……etc.
    - We have to define the level of aggregation hierarchy- up to what level down
    - Obviously we can't go to the level of fundamental particles. We don't have the computational capability to do that.
  - Abstraction hierarchies:
    - Is-A
    - All these elements belong to the same category
      - Example: all individual cats to cat category, all individual dogs to dog category… etc. all cats and dogs …etc. categories to mammals so cat is a mammal, dog is a mammal etc.

Hierarchies of Structured Knowledge



Beyond categories and concepts

- The language of logic is concerned with sentences.
- Sentence is something that is true or false.
- But knowledge about the world goes beyond simple sentences
  - Complex organisms can be described only by collecting a large number of sentences and organizing them into meaningful structures.
  - That we call as hierarchies.

Situational Knowledge

- We can say that an agent is intelligent in a situation
  - Intelligence can be shown in a domain in a particular situation
- We do abductive reasoning as we have situational knowledge
  - What happens in a restaurant
  - What happens in a hospital etc.
- So we need that kind of situational knowledge
- If you are blindfolded and touching something
  - How can you guess that it is an elephant?
  - You have to know about elephants in the first place.

Knowledge structure

- How does one represent knowledge about bookstore?
- Or about restaurants, or about dental clinics?
- We can do it by putting together different kind of sentences which will describe those entities
  - In some sense we collect and organize individual bits of facts about the typical knowledge about these entities
    - These will be aggregated representations
  - Roger Schank called such structures Scripts
  - Marvin Minsky called them Frames
    - And gave rise to the idea of Object Oriented Programming Systems

Taxonomies

- One approach to representing knowledge would be to have a large pool of sentences
- An algorithm for making inferences would search through the sentences to make the right connections
- This would be at the cost of computational complexity
- Instead we tend to organize facts into taxonomies
- Taxonomies facilitate inheritance
  - For example, mammals have two eyes
  - Implies that cats have two eyes
- Taxonomies facilitate compactness
  - We only need to store a property in a super class
  - And inherit it into a subclass
- Minsky's frames led to OOPs

- First order logic allows us to talk about
  - Domain
  - Individuals in the domain
  - Relations between individuals
  - Relations between categories
- Notion of vars and quantifiers
- First order logic does not involve time
  - First order logic is change less (it is not the case that something was true yesterday but not today)
  - It is called as mathematical logic
  - It’s a logic of relations (predicates capture relations between individuals and so on)

First Order Logic (FOL)

- The domain D of FOL is a set of individuals
- Predicates of FOL capture
  - Relation between individuals. For example, Friend(Priya, Sonam)
  - Categories or concepts. For example, Man(Socrates)
  - Relation between categories. For example, ∀x (Deer(x) => Mammal(x))
  - Relations between named relations. For example, ∀x,y (Brother(x,y) => Sibling(x,y)
- Depending on the types of predicates we choose, we need to add such rules to our logic
- Inference involve traversing from known to the unknown
- Rules of inference facilitate this traversal
- The more the number of predicates the more the rules

To avoid the tsunami of rules of inferences

- How to choose the set of predicates so that our reasoning becomes tractable and computationally feasible.
  - So that our representation is compact and canonical
- Canonical set of predicate
  - Represent only the core concept
    - Same thing is not represented in many different ways
- If we want input in any natural language we need to map into set of canonical conceptual representation that is not as rich in words as

Advantages of logical representation

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

Disadvantages of logical Representation

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.
3. First order logic is computationally very expensive
4. And very often all its power is not required in the kind of reasoning that we want to do
5. Logic of noun prases

Production Rules

- Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:
  - The set of production rules
  - Working Memory
  - The recognize-act-cycle

- In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out.
- The condition part of the rule determines which rule may be applied to a problem.
- And the action part carries out the associated problem-solving steps.
- This complete process is called a recognize-act cycle.

- The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory.
- This knowledge match and may fire other rules.
- If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set.
  - In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

- Example:
  - **IF (at bus stop AND bus arrives) THEN action (get into the bus)**
  - **IF (on the bus AND paid AND empty seat) THEN action (sit down).**
  - **IF (on bus AND unpaid) THEN action (pay charges).**
  - **IF (bus arrives at destination) THEN action (get down from the bus).**

- There are two ways to pursue such a search that are forward and backward reasoning. The significant difference between both of them is that forward reasoning starts with the initial data towards the goal. Conversely, backward reasoning works in opposite fashion where the purpose is to determine the initial facts and information with the help of the given results.

Comparison Chart

BASIS FOR COMPARISON	FORWARD REASONING	BACKWARD REASONING
Basic	Data-driven	Goal driven
Begins with	New Data	Uncertain conclusion
Objective is to find the	Conclusion that must follow	Facts to support the conclusions
Type of approach	Opportunistic	Conservative
Flow	Incipient to consequence	Consequence to incipient

Forward Reasoning

- The solution of a problem generally includes the initial data and facts in order to arrive at the solution.
- These known facts and information is used to deduce the result.
- For example, while diagnosing a patient the doctor first check the symptoms and medical condition of the body such as temperature, blood pressure, pulse, eye colour, blood, etcetera.
- After that, the patient symptoms are analysed and compared against the predetermined symptoms of diseases.
- Then the doctor is able to provide the medicines according to the symptoms of the patient.
- So, when a solution employs this manner of reasoning, it is known as **forward reasoning**.

Steps that are followed in the forward reasoning

- The inference engine explores the knowledge base with the provided information for constraints whose precedence matches the given current state.
  - In the first step, the system is given one or more than one constraints.
  - Then the rules are searched in the knowledge base for each constraint. The rules that fulfil the condition are selected(i.e., IF part).
  - Now each rule is able to produce new conditions from the conclusion of the invoked one. As a result, THEN part is again included in the existing one.
  - The added conditions are processed again by repeating step 2.
  - The process will end if there is no new conditions exist.

Backward Reasoning

- The **backward reasoning** is inverse of forward reasoning in which goal is analysed in order to deduce the rules, initial facts and data.
- We can understand the concept by the similar example given in the above definition, where the doctor is trying to diagnose the patient with the help of the inceptive data such as symptoms.
- However, in this case, the patient is experiencing a problem in his body, on the basis of which the doctor is going to prove the symptoms.
- This kind of reasoning comes under backward reasoning.

Steps that are followed in the backward reasoning

- In this type of reasoning, the system chooses a goal state and reasons in the backward direction.
- Now, let's understand how does it happens and what steps are followed.
  - Firstly, the goal state and the rules are selected where the goal state reside in the THEN part as the conclusion.
  - From the IF part of the selected rule the subgoals are made to be satisfied for the goal state to be true.
  - Set initial conditions important to satisfy all the subgoals.
  - Verify whether the provided initial state matches with the established states.
  - If it fulfils the condition then the goal is the solution otherwise other goal state is selected.

Conflict Resolution

- We have two rules, Rule 1 and Rule 2, with the same IF part.
- Thus both of them can be set to fire when the condition part is satisfied.
- These rules represent a conflict set.
- The inference engine must determine which rule to fire from such a set.
- A method for choosing a rule to fired in a given cycle is called **conflict resolution**.
- In forward chaining, BOTH rules would be fired.
- Rule 1 is fired first as the topmost one, and as a result, its THEN part is executed.
- However, Rule 2 is also fired because the condition part of this matches.

Methods used for **conflict resolution**

- Fire the rule
  - with the **highest priority**. In simple applications, the priority can be established by placing the rules in an appropriate order in the knowledge base. Usually this strategy works well for expert systems with around 100 rules.
  - That is **most specific rule**. This method is also known as the **longest matching strategy**. It is based on the assumption that a specific rule processes more information than a general one.
  - That uses the data most recently entered in the database.
    - This method relies on time tags attached to each fact in the database.

Advantages of Production rule

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

Disadvantages of Production rule

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. During the execution of the program, many rules may be active hence rule-based production systems are inefficient.

Structured knowledge representation

- Semantic Nets,
- frames,
- Conceptual dependency,
- Scripts

Semantic Networks Representation

- Semantic networks are alternative of predicate logic for knowledge representation.
- In Semantic networks, we can represent our knowledge in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects and arc labels that denote particular relations.
- Semantic networks can categorize the object in different forms and can also link those objects.
- Semantic networks are easy to understand and can be easily extended.

- This representation consist of mainly two types of relations:
  1. IS-A relation (Inheritance)
  2. Kind-of-relation
- **Example:** Following are some statements which we need to represent in the form of nodes and arcs.
- Statements:

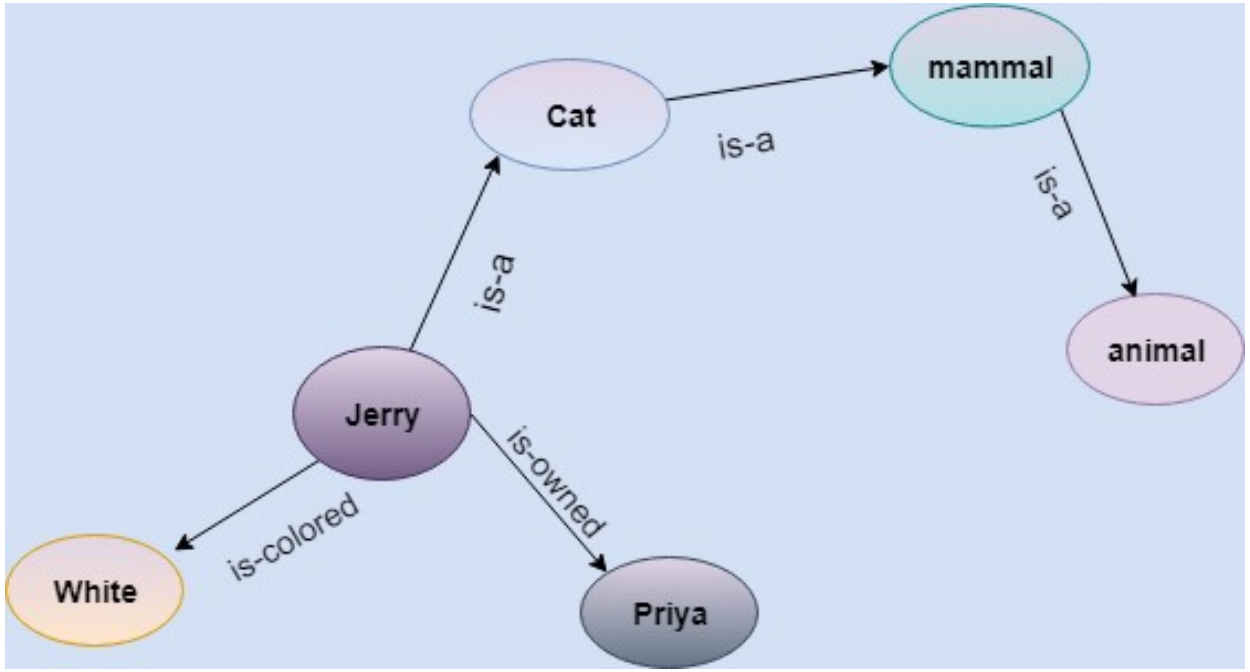
1.Jerry is a cat.

2.Jerry is a mammal

3.Jerry is owned by Priya.

4.Jerry is white colored.

5.All animals are mammal.


Cat
mammal
is-a
is-a
animal
is-a
Jerry
is-owned
is-colored
White
Priya

Drawbacks in Semantic representation

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions.
2. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
3. Semantic networks try to model human-like memory (Which has 86 billion neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
4. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
5. Semantic networks do not have any standard definition for the link names.
6. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network

1.Semantic networks are a natural representation of knowledge.

2.Semantic networks convey meaning in a transparent manner.

3.These networks are simple and easily understandable.

Frames Representation

- A frame is a collection of attributes or slots and associated values that describe some real world entity
- Each frame represents –
  - a class, or
  - an instance (an element of a class)
- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.
- Frames are the AI data structure which divides knowledge into substructures.
- Thus, It consists of a collection of slots and slot values.
- These slots may be of any type and sizes.
- Slots have names and values which are called facets.

Facets

- Each slot may contain one or more facets (called fillers) which may take many forms such as :
  ⇒value (value of the slot),
  ⇒default (default value of the slot),
  ⇒range (indicates the range of integer or enumerated values, a slot can have),
  ⇒demons (procedural attachments such as if_needed, if_deleted, if_added etc.) and
  ⇒other (may contain rules, other frames, semantic net or any type of other information).

Description of Frames

- Each frame represent either a class or an instance.
- Class frame represents a general concept whereas instance frame represents a specific occurrence of the class instance.
- Class frame generally have default values which can be redefined at lower levels.
- If class frame has actual value facet then decedent frames can not modify that value.
- Value remains unchanged for subclasses and instances.

- Frames are derived from semantic networks and later evolved into our modern-day classes and objects.
- A single frame is not much useful.
- Frames system consist of a collection of frames which are connected.
- In the frame, knowledge about an object or event can be stored together in the knowledge base.
- The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example: example of a frame for a book

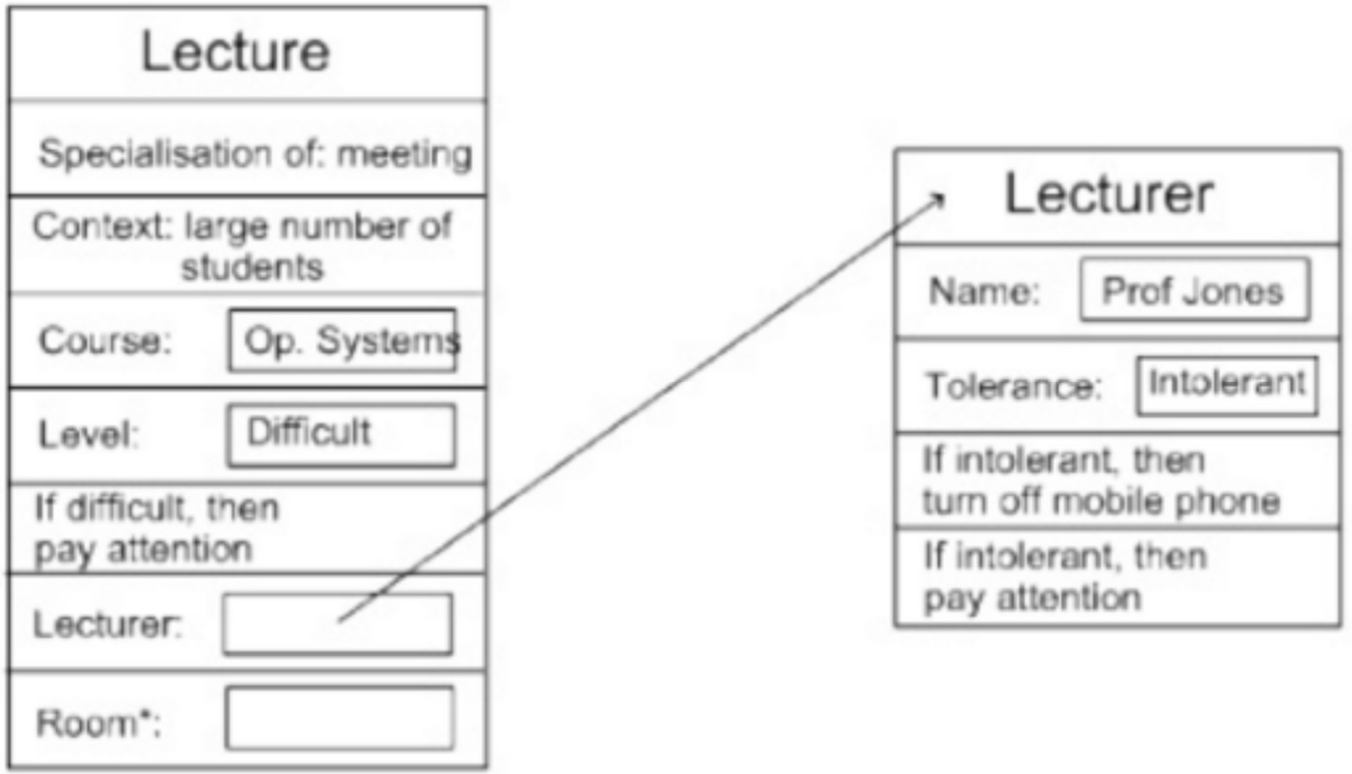
Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Example: an entity Peter

- Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

Frames: Example

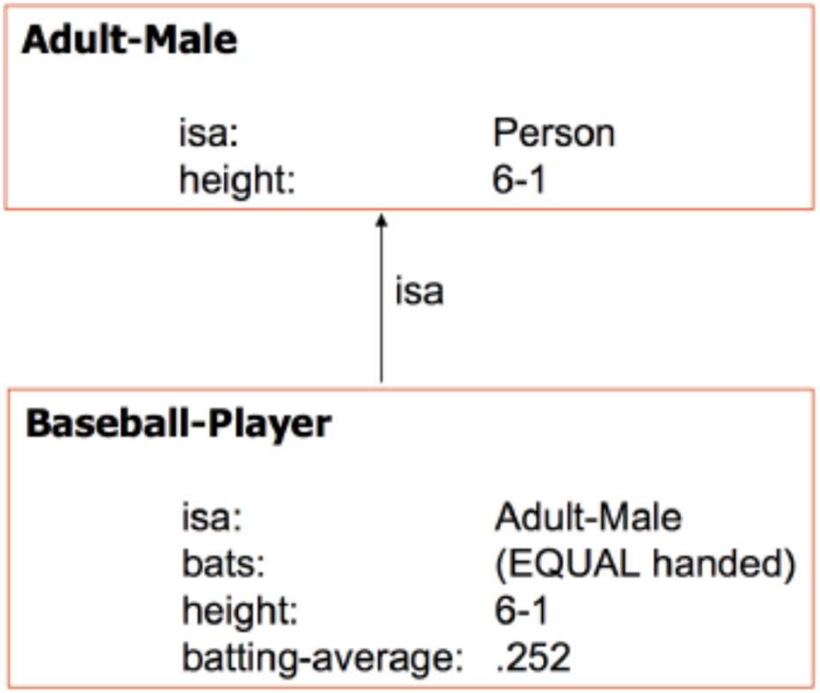


- Information retrieval when facing a new situation
  - information is stored in frames with slots
  - some of the slots trigger actions, causing new situations
- Frames are templates
  - need to be filled-in in a situation
  - filling them causes the agent to undertake actions and retrieve other frames
- Frames are extensions of record datatype in databases
- Also very similar to object oriented processing

Flexibility in Frames

- Slots in a frame can contain
  - information for choosing a frame in a situation
  - relationship between this and other frames
  - procedures to carry out after various slots filled
  - default information to use when input is missing
  - blank slots
  - left blank unless required for a task
  - other frames, which gives a hierarchy

Example: Frames Hierarchy



Advantages of frame representation

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation

1.In frame system inference mechanism is not be easily processed.

2.Frame representation has a much generalized approach.

Conceptual Dependency (CD)

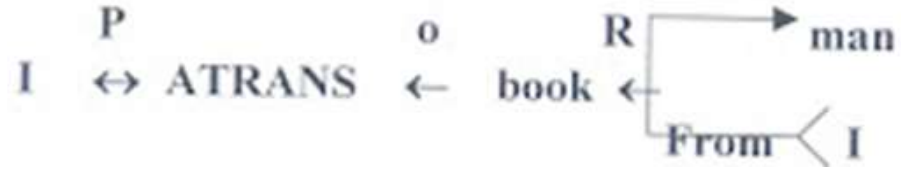
- A model of Natural language understanding in AI
- Conceptual Dependency originally developed to represent knowledge acquired from natural language input.
- Roger Schank at Stanford University introduced CD model in 1973 to 1975
- The goals of this theory are:
  - To help in the drawing of inference from sentences.
  - To be independent of the language.
  - That is to say: *For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.*

- CD representation of a sentence is not built using words in the sentence rather built using conceptual primitives which give the intended meanings of words.
- CD provides structures and specific set of primitives from which representation can be built.

CD

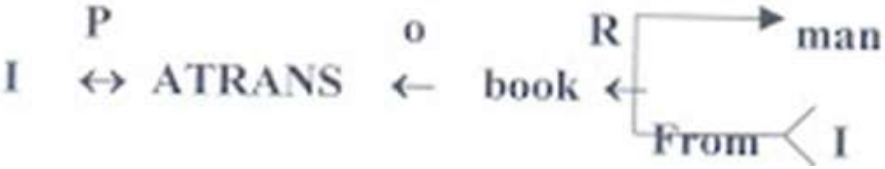
- It is independent of the language in which the sentences were originally stated.
- CD representations of a sentence is built out of primitives, which are not words belonging to the language but are conceptual
- These primitives are combined to form the meanings of the words
- As an example consider the event represented by the sentence

I gave the man a book.
In CD from the above sentence can be represented with primitives as

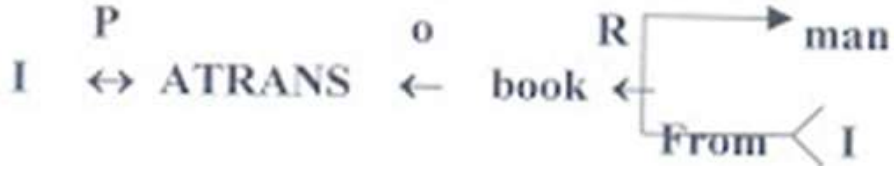


Few conventions

- In this representation the symbols have the following meaning:
- Arrows indicate direction of dependency
- Double arrow indicates two way link between actor and action
- ATRANS is one of the primitive acts used by the theory.
  - It indicates transfer of possession
- O- for the object case relation
- R- for the recipient case relation
- P- for past tense
- D- for destination {*not used in this representation}



- This representation is same for different saying with same meaning.
- For example
  - I gave the man a book.
  - The man got book from me.
  - The book was given to man by me etc.



- It has been used by many programs that pretend to understand English (*MARGIE, SAM, PAM*).
- CD provides:
  - a structure into which nodes representing information can be placed
  - a specific set of primitives
  - at a given level of granularity.
- Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.
  - The agent and the objects are represented
  - The actions are built up from a set of primitive acts which can be modified by tense.

Primitive Acts of CD theory

ATRANS
-- Transfer of an abstract relationship. *e.g. give*.
PTRANS
-- Transfer of the physical location of an object. *e.g. go*.
PROPEL
-- Application of a physical force to an object. *e.g. push*.
MTRANS
-- Transfer of mental information. *e.g. tell*.
MBUILD
-- Construct new information from old. *e.g. decide*.

SPEAK
-- Utter a sound. *e.g. say*.
ATTEND
-- Focusing of a sense organ toward a stimulus. *e.g. listen, watch*.
MOVE
-- Movement of a body part by owner. *e.g. punch, kick*.
GRASP
-- Actor grasping an object. *e.g. clutch*.
INGEST
-- Actor ingesting an object. *e.g. eat*.
EXPEL
-- Actor getting rid of an object from body. *e.g. cry*

Six primitive conceptual categories

- provide *building blocks* which are the set of allowable dependencies in the concepts in a sentence
- PP
  - Real world objects. {Picture producer}
- ACT
  - Real world actions {one of the CD Primitives}
- PA
  - Attributes of objects {Picture aiders}
- AA
  - Attributes of actions {Action aiders}
- T
  - Times
- LOC
  - Locations

The use of tense and mood in describing events

- p- past
- f- future
- t- transition
- t_s- start transition
- t_f- Finished transition
- k- continuing
- ?- introgative
- /- negative
- Nil- Present
- delta- timeless
- c- conditional
- The absence of any modifier implies the present tense

Rule 1: PP ⇔ ACT

- It describes the relationship between an actor and the event he/she causes
  - This is a two way dependency, since neither actor nor event can be considered primary
  - The letter p in the dependency link indicates past tense.
- Example
  - John ran
  - CD Representation John $\overset{p}{\Leftrightarrow}$PTRANS

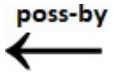
Rule 2: ACT ← PP

- It describes the relationship between a ACT and a PP (object) of ACT
  - The direction of the arrow is toward the ACT since the context of the specific ACT determines the meaning of the object relation
- Example: John pushed the bike
- CD Representation: John ⇔ PROPEL ← bike

Rule 3: PP↔ PP

- It describes the relationship between two PP's, one of which belongs to the set defined by the other.
- Example: John is doctor.
- CD Representation: John $\leftrightarrow$ doctor

Rule 4: PP ← PP

- It describes the relationship between two PP's, 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 POSS-BY), location (shown as LOC), and physical containment (Shown as CONT)
  - The direction of the arrow is again toward the concept being described.
  - Example: John's dog
  - CD representation: dog 

John

Rule 5: PP⇔PA

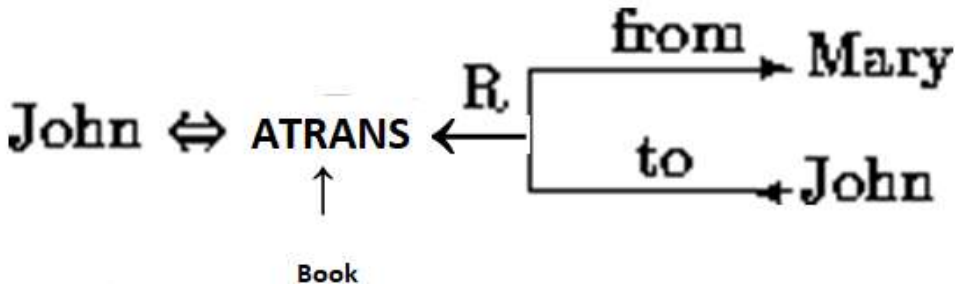
- It describes the relationship between a PP and a PA that is asserted to describe it.
  - PA represents states of PP such as height, health etc.
- Example: John is fat
- CD representation: John ⇔ weight (>80)

Rule 6: PP ←PA

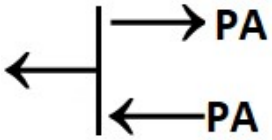
- It describes the relationship between a PP and an attribute that already has been predicated of it.
  - Direction is towards PP being described.
- Example: Smart John
- CD Representation: John ← smart

Rule 7: ACT ←R |→ PP (to), ←PP (from)

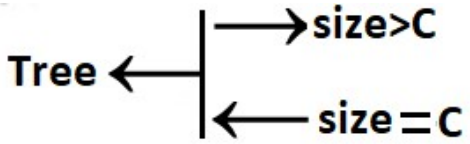
- It describes the relationship between an ACT and the source and the recipient of the ACT
- Example: John took the book from Mary
- CD Representation:

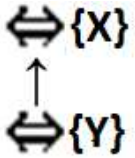


Rule 8: PP

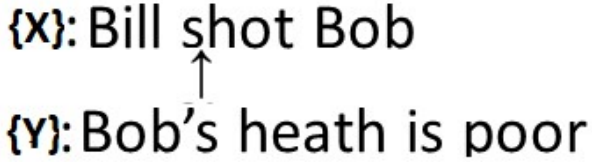


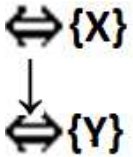
- It describes the relationship that describes the change in state.
- Example: Tree grows
- CD Representation:



Rule 9: 

- It describes the relationship between one conceptualization and another that causes it.
  - Here {x} causes {y} i.e., if x then y
- Example: Bill shot Bob. Bob's heath is poor.
- CD Representation:



Rule 10: 

- It describes the relationship between one conceptualization with another that is happening at the time of the first
  - Here {y} is happening while {x} is in progress.
- Example: while going home I saw a snake

I am going home

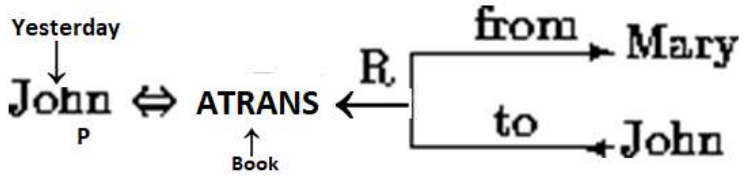
↓

I saw a snake

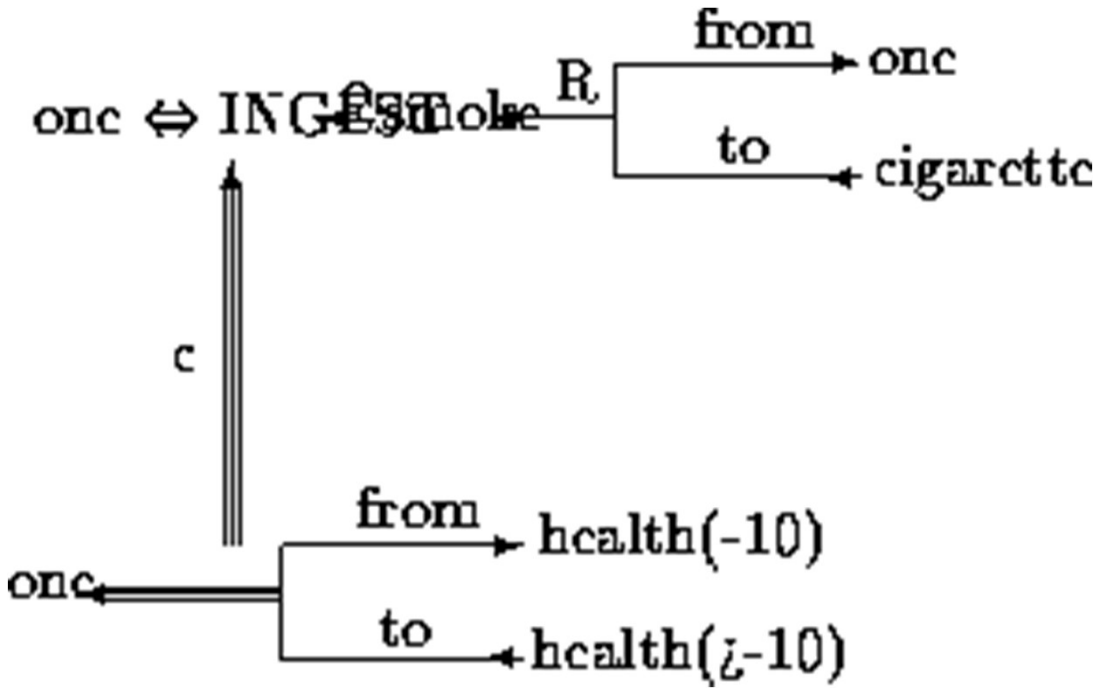
Primitive states

- *Primitive states* are used to describe many state descriptions such as height, health, mental state, physical state.
- here are many more physical states than primitive actions. They use a numeric scale.
- *E.g.* John ⇔ height(+10) *John is the tallest*
- John⇔height(< average) *John is short*
- Frank Zappa ⇔ health(-10) *Frank Zappa is dead*
- Dave ⇔mental_state(-10) *Dave is sad*
- Vase ⇔physical_state(-10) *The vase is broken*

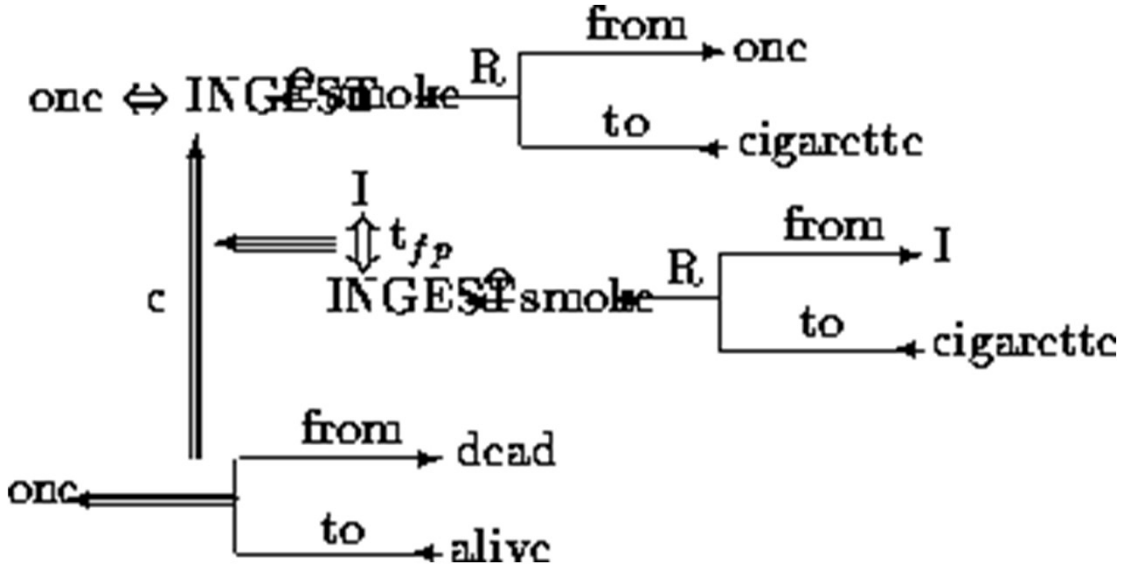
- You can also specify things like the time of occurrence in the relation ship.
- For Example: *John gave Mary the book yesterday*



- Now let us consider a more complex sentence
  - *Since smoking can kill you, I stopped*
- Lets look at how we represent the inference that *smoking can kill*:
  - Use the notion of *one* to apply the knowledge to.
  - Use the *primitive act* of INGESTing smoke from a cigarette to one.
  - Killing is a transition from being alive to dead. We use *triple arrow*s to indicate a transition from one state to another.
  - Have a conditional, *c* causality link. The *triple arrow* indicates dependency of one concept on another.


from
onc
R
to
cigarcttc
c
from
health(-10)
onc
to
health(¿-10)

- To add the fact that *I stopped smoking*
  - Use similar rules to imply that I smoke cigarettes.
  - The qualification t_{fp} attached to this dependency indicates that the instance INGESTing smoke has stopped.


onc ⇔ INGEST smoke
R
from
onc
to
cigarette
I
t_{fp}
c
INGEST smoke
R
from
I
to
cigarette
from
dead
onc
to
alive

Generation of CD representations

Sentences	CD Representations
Jenny cried	p o d ? Jenny ⇔ EXPEL ← tears eyes poss-by ⇑ Jenny
Mike went to India	p d India Mike ⇔ PTRANS ? (source is unknown)
Mary read a novel	p o d CP(Mary) Mary ⇔ MTRANS ← info novel ↑ i (instrument) p o d novel Mary ⇔ ATTEND ← eyes ?

- *Primitive states* are used to describe many state descriptions such as height, health, mental state, physical state.
- There are many more physical states than primitive actions. They use a numeric scale.
- *E.g.* John height(+10) *John is the tallest*
- John height(< average) *John is short*
- Frank Zappa health(-10) *Frank Zappa is dead*
- Dave mental_state(-10) *Dave is sad*
- Vase physical_state(-10) *The vase is broken*

Sentence	CD Representation
Since drugs can kill, I stopped.	o r One One ⇔ INGEST ← durgs Mouth c health = -10 One health > -10 c t_p o r I I ⇔ INGEST ← durgs mouth

Inferences Associated with Primitive Act

- General inferences are stored with each primitive Act thus reducing the number of inferences that need to be stored explicitly with each concept.
- For example, from a sentence "John killed Mike", we can infer that "Mike is dead".
- Let us take another example of primitive Act INGEST.
- The following inferences can be associated with it.
  - The object ingested is no longer available in its original form.
  - If object is eatable, then the actor has less hunger.
  - If object is toxic, then the actor's heath is bad.
  - The physical position of object has changed. So PTRANS is inferred.

Cont...

- Example: The verbs {give, take, steal, donate} involve a transfer of ownership of an object.
  - If any of them occurs, then inferences about who now has the object and who once had the object may be important.
  - In a CD representation, these possible inferences can be stated once and associated with the primitive ACT "ATRANS".
- Consider another sentence "Bill threatened John with a broken nose"
  - Sentence interpretation is that Bill informed John that he (Bill) will do something to break john's nose.
  - Bill did (said) so in order that John will believe that if he (john) does some other thing (different from what Bill wanted) then Bill will break John's nose.

Advantages of CD

- Using these primitives involves fewer inference rules.
- Many inference rules are already represented in CD structure.
- The holes in the initial structure help to focus on the points still to be established.

Disadvantages of CD

- Knowledge must be decomposed into fairly low level primitives.
- Impossible or difficult to find correct set of primitives.
- A lot of inference may still be required.
- Representations can be complex even for relatively simple actions. Consider:*Dave bet Frank five pounds that Wales would win the Rugby World Cup*.
- Complex representations require a lot of storage

Applications of CD

- **MARGIE** (*Meaning Analysis, Response Generation and Inference on English*)
  - model natural language understanding.
- **SAM** (*Script Applier Mechanism*)
  - Scripts to understand stories. See next section.
- **PAM** (*Plan Applier Mechanism*)
  - Scripts to understand stories.
- Schank *et al.* developed all of the above.

Ontologies

- an **ontology** is a formal explicit description of concepts in a domain of discourse (**classes** (sometimes called **concepts**)), properties of each concept describing various features and attributes of the concept (**slots** (sometimes called **roles** or **properties**)), and restrictions on slots (**facets** (sometimes called **role restrictions**)).
- An ontology together with a set of individual **instances** of classes constitutes a **knowledge base.**

- Classes are the focus of most ontologies.
- Classes describe concepts in the domain.
- For example, a class of wines represents all wines.
- Specific wines are instances of this class.
- The Bordeaux wine in the glass in front of you while you read this document is an instance of the class of Bordeaux wines.
- A class can have **subclasses** that represent concepts that are more specific than the superclass.
- For example, we can divide the class of all wines into red, white, and rosé wines.
- Alternatively, we can divide a class of all wines into sparkling and non-sparkling wines.

- Slots describe properties of classes and instances: `Château Lafite Rothschild Pauillac` wine has a full body;
- It is produced by the `Château Lafite Rothschild` winery.
- We have two slots describing the wine in this example:
  - the slot body with the value full and the slot maker with the value `Château Lafite Rothschild` winery.
- At the class level, we can say that instances of the class `Wine` will have slots describing their `flavor`, `body`, `sugar level`, the `maker` of the wine and so on.
- All instances of the class `Wine`, and its subclass Pauillac, have a slot `maker` the value of which is an instance of the class Winery.
- All instances of the class Winery have a slot `produces` that refers to all the wines (instances of the class Wine and its subclasses) that the winery produces.

- In practical terms, developing an ontology includes:
  - defining classes in the ontology,
  - arranging the classes in a taxonomic (subclass–superclass) hierarchy,
  - defining slots and describing allowed values for these slots,
  - filling in the values for slots for instances.
- We can then create a knowledge base by defining individual instances of these classes filling in specific slot value information and additional slot restrictions.
- Suggested Readings: https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Scripts

- A script is a structured representation describing a stereotyped sequence of events in a particular context.
- Scripts are used to organize events in knowledge bases
- Scripts are very related to the idea of frames

Components of a Script

- A script is composed of several components
- Entry conditions that must be true for the script to be called
- Results or facts that are true once the script has terminated
- Props or the ”things” that make up the content of the script
- Roles are the actions that the individual participants perform
- Scenes which present temporal aspects of the script

Canonical Example: Restaurant Visit

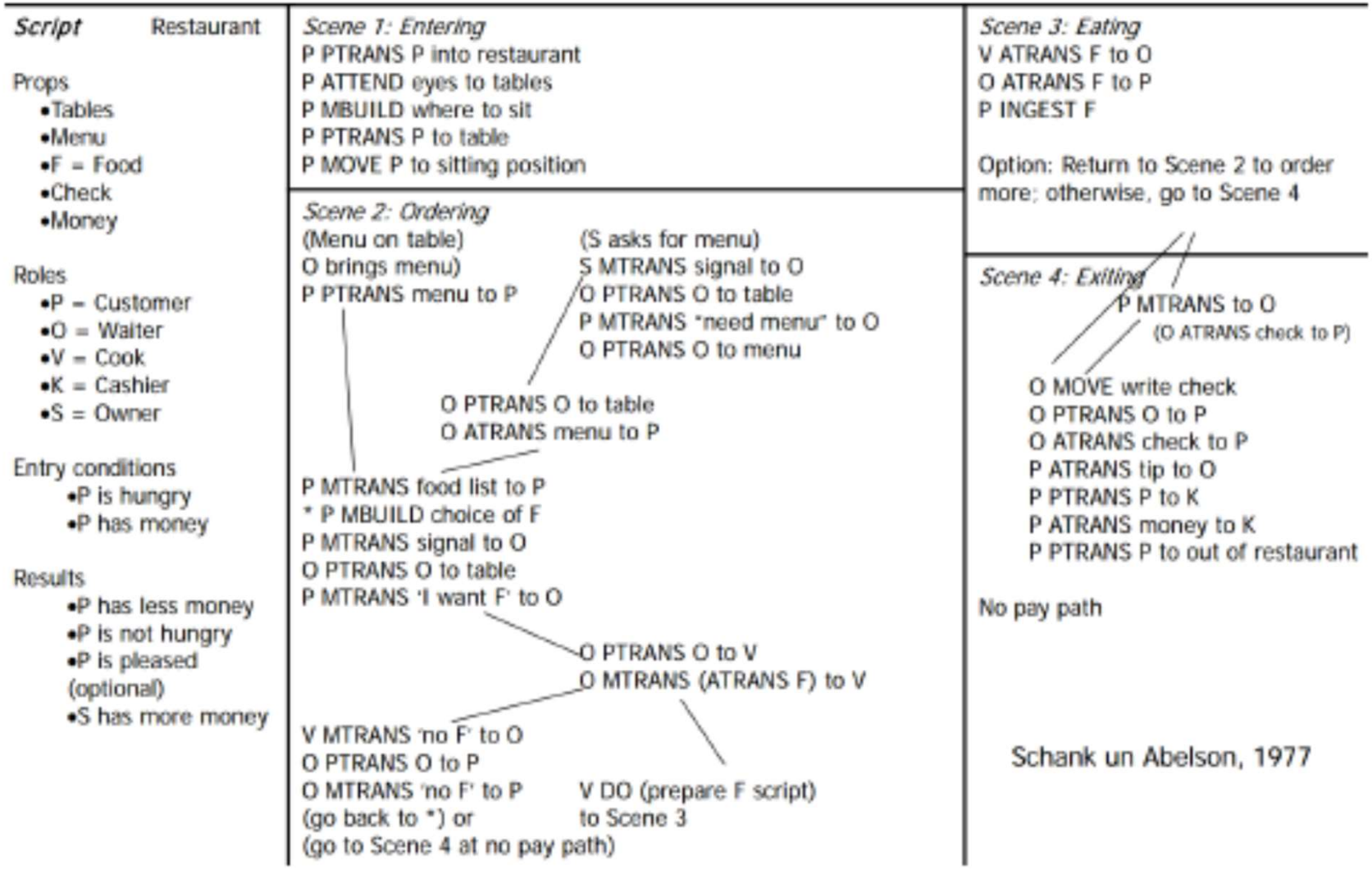
- Objects: tables, menu, food, check, money, ...
- Roles: customer, waiter, cook, cashier, owner, ...
- Entry conditions: customer hungry, customer has money
- Results: customer not hungry, customer has less money, owner more money, ...
- Scenes
  - Scene 1: Entering
    - customer enters restaurant
    - customers looks at tables
    - customer decides where to sit
    - ...
  - Scene 2: Ordering
    - waiter brings menu
    - ...
  - ...

Script Actions

Describing a script a special symbols of actions are used:

Symbol	Meaning	Example
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*

Detailed Script



Schank un Abelson, 1977