UNIT-3

Representation of Knowledge:

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**. Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describes behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge representation issues:

The fundamental goal of knowledge Representation is to facilitate inference (conclusions) from knowledge. The issues that arise while using KR techniques are many. Some of these are explained below.

Important Attributed:

Any attribute of objects so basic that they occur in almost every problem domain?
There are two attributed “instance” and “isa”, which are general significance. These attributes are important because they support property inheritance.

Relationship among attributes:

Any important relationship that exists among object attributed?
The attributes we use to describe objects are themselves entities that we represent. The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:

1. Inverse — This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways.
2. Existence in an isa hierarchy — This is about generalization-specification, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. For example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute. These generalization-specialization relationships are important for attributes because they support inheritance.
3. Technique for reasoning about values — This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like, height: must be in a unit of length, Age: of a person cannot be greater than the age of person's parents. The values are often specified when a knowledge base is created.
4. Single valued attributes — This is about a specific attribute that is guaranteed to take a unique value. For example, a baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

Choosing Granularity:

At what level of detail should the knowledge be represented?
Regardless of the KR formalism, it is necessary to know:

- At what level should the knowledge be represented and what are the primitives?
- Should there be a small number or should there be a large number of low-level primitives or High-level facts.
- High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.

Set of objects:

How should sets of objects be represented?
There are certain properties of objects that are true as member of a set but not as individual;

Example: Consider the assertion made in the sentences:

"there are more sheep than people in Australia", and "English speakers can be found all over the world."

To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English. The reason to represent sets of objects is: if a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.

This is done,

- in logical representation through the use of universal quantifier, and
- in hierarchical structure where node represent sets and inheritance propagate set level assertion down to individual.

Finding Right structure:

Given a large amount of knowledge stored in a database, how can relevant parts are accessed when they are needed?

This is about access to right structure for describing a particular situation. This requires, selecting an initial structure and then revising the choice. While doing so, it is necessary to solve following problems:

- How to perform an initial selection of the most appropriate structure.
- How to fill in appropriate details from the current situations.
- How to find a better structure if the one chosen initially turns out not to be appropriate.
- What to do if none of the available structures is appropriate.
- When to create and remember a new structure.

Predicate Logic

Predicate Logic deals with predicates, which are propositions, consist of variables.

Predicate Logic - Definition

A predicate is an expression of one or more variables determined on some specific domain. A predicate with variables can be made a proposition by either authorizing a value to the variable or by quantifying the variable.

The following are some examples of predicates.

- Consider E(x, y) denote "x = y"
- Consider X(a, b, c) denote "a + b + c = 0"
- Consider M(x, y) denote "x is married to y."

Quantifier:

The variable of predicates is quantified by quantifiers. There are two types of quantifier in predicate logic - Existential Quantifier and Universal Quantifier.

Existential Quantifier:

If p(x) is a proposition over the universe U. Then it is denoted as ∃x p(x) and read as "There exists at least one value in the universe of variable x such that p(x) is true. The quantifier ∃ is called the existential quantifier.

There are several ways to write a proposition, with an existential quantifier, i.e.,

(∃x∈A)p(x) or ∃x∈A such that p (x) or (∃x)p(x) or p(x) is true for some x ∈A.

Universal Quantifier:

If p(x) is a proposition over the universe U. Then it is denoted as ∀x,p(x) and read as "For every x∈U,p(x) is true." The quantifier ∀ is called the Universal Quantifier.

There are several ways to write a proposition, with a universal quantifier.

∀x∈A,p(x) or p(x), ∀x ∈A Or ∀x,p(x) or p(x) is true for all x ∈A

Negation of Quantified Propositions:

When we negate a quantified proposition, i.e., when a universally quantified proposition is negated, we obtain an existentially quantified proposition,and when an existentially quantified proposition is negated, we obtain a universally quantified proposition.

The two rules for negation of quantified proposition are as follows. These are also called DeMorgan's Law.

Example: Negate each of the following propositions:

1.∀x p(x)∧ ∃ y q(y)

Sol: ~.∀x p(x)∧ ∃ y q(y))

≅~∀ x p(x)∨~∃yq (y) (∴~(p∧q)=~p∨~q)

≅ ∃ x ~p(x)∨∀y~q(y)

Propositions with Multiple Quantifiers:

The proposition having more than one variable can be quantified with multiple quantifiers. The multiple universal quantifiers can be arranged in any order without altering the meaning of the resulting proposition. Also, the multiple existential quantifiers can be arranged in any order without altering the meaning of the proposition.

The proposition which contains both universal and existential quantifiers, the order of those quantifiers can't be exchanged without altering the meaning of the proposition, e.g., the proposition ∃x ∀ y p(x,y) means "There exists some x such that p (x, y) is true for every y."

Example: Write the negation for each of the following. Determine whether the resulting statement is true or false. Assume U = R.

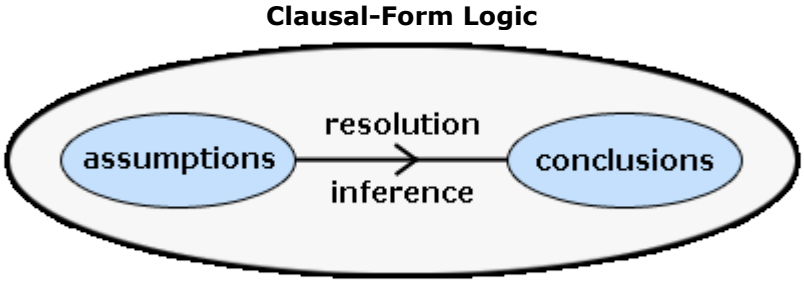
1.$\forall x \exists m(x^2<m)$

Sol: Negation of $\forall x \exists m(x^2<m)$ is $\exists x \forall m (x^2 \geq m)$. The meaning of $\exists x \forall m (x^2 \geq m)$ is that there exists for some x such that $x^2 \geq m$, for every m. The statement is true as there is some greater x such that $x^2 \geq m$, for every m.

Logic programming:

Artificial Intelligence (AI) is the ability for an artificial machine to act intelligently. Logic Programming is a method that computer scientists are using to try to allow machines to reason because it is useful for knowledge representation. In logic programming, logic is used to represent knowledge and inference is used to manipulate it.

The logic used to represent knowledge in logic programming is clausal form which is a subset of first-order predicate logic. It is used because first-order logic is well understood and able to represent all computational problems. Knowledge is manipulated using the resolution inference system which is required for proving theorems in clausal-form logic. The diagram below shows the essence of logic programming.



Prolog, PROgramming in LOGic, is a declarative programming language which is based on the ideas of logic programming, such as those discussed above. The idea of Prolog was to make logic look like a programming language and allow it to be controlled by a programmer to advance the research for theorem-proving.

Semantic nets:

Semantic networks in AI are graphical structures designed to represent and organize knowledge, enabling machines to understand and process information in human-readable form. These networks consist of nodes representing concepts or objects and links denoting relationships between them. This structured format provides efficient data retrieval and reasoning.

Semantic networks are significant in natural language processing, knowledge representation, and information retrieval systems. They focus on capturing the context and meaning of words. It involves processing for tasks like sentiment analysis, text summarization, and question answering. Moreover, semantic networks support machine reasoning, enabling AI systems to draw logical conclusions based on

the connections within the network. This capability plays an important role in problem-solving and decision-making scenarios.

By enhancing data intelligence and manipulation, semantic networks empower AI to provide more intelligent and context-aware solutions, ultimately enhancing user experiences across diverse applications and domains.

Types of Semantic Networks

There are six types of semantic networks, each having different significance in graphic representation for knowledge management and automated reasoning:

- **Definitional Networks**: This represents the relationship between concepts and their subtypes. Definitional networks are fundamental in ensuring clarity and precision in knowledge representation, allowing for a clear delineation of how different concepts relate to one another within a given domain.
- **Assertional Networks**: It is used to assert propositions and convey factual information. These networks are particularly useful for capturing and communicating structured information, making them a cornerstone for knowledge-based systems and databases.
- **Implicational Networks**: They rely on implications as the primary connections between nodes, emphasizing cause-and-effect relationships.By emphasizing causal connections, these networks are vital for predictive modeling, risk assessment, and scenario analysis, enabling the inference of potential outcomes based on established relationships.
- **Executable Networks**: Contain mechanisms capable of inducing changes within the network itself, allowing for dynamic adaptations.These networks are essential for systems that require real-time decision-making, as they can autonomously modify their structure or behavior to optimize performance or adapt to new information.
- **Learning Networks**: These expand knowledge representations by emphasizing insights from examples and focusing on adaptive learning. Learning networks are foundational in machine learning and AI applications, as they enable systems to continuously improve and refine their understanding of the world through exposure to new information.
- **Hybrid Networks**: This type of network consists of two or more of the above-mentioned techniques, either within a single network or between closely interacting networks, to fulfill diverse knowledge representation requirements.

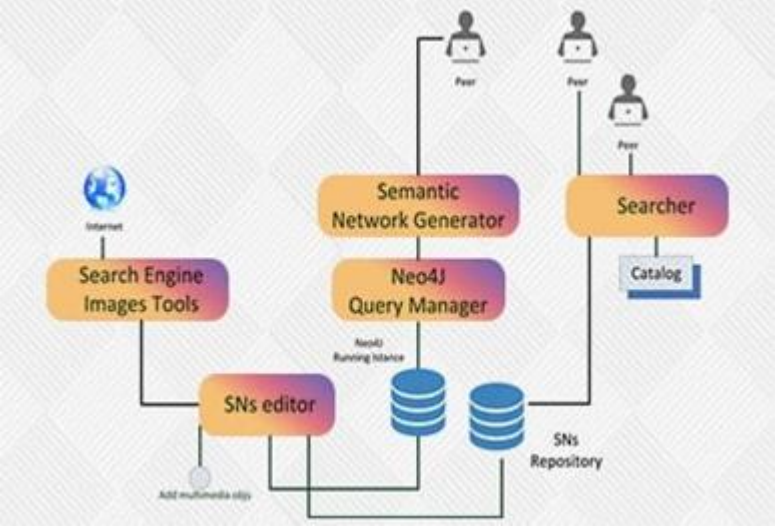
Components of Semantic Networks

These components collectively define the architecture of semantic networks,which represent and process knowledge effectively across various applications. Semantic networks can be further categorized by identifying their fundamental components:

- **Lexical Components:**
    - **Nodes**: These represent objects or concepts within the network.
    - **Links**: They denote the relationships connecting nodes.
    - **Labels**: Labels attached to nodes and links specify particular objects and relations, providing context.
- **Structural Component:** In this aspect, nodes and links combine to form a directed graph. Labels are placed strategically on both nodes and links to establish their roles within the network's structure.
- **Semantic Component:** This component gives meaning to the links and labels associated with nodes. The interpretations of these meanings guide the network's functionality, enabling knowledge representation and inference.
- **Procedural Part:** Constructors permit the creation of new nodes and links, expanding the network. Destructors, on the other hand, facilitate the removal of nodes and links, ensuring the network remains adaptable and up-to-date.

Semantic Network Architecture

Semantic networks use visual symbols to illustrate information or data, utilizing labeled nodes and directed arcs within a graph structure to encode knowledge comprehensively. Its uncomplicated architecture not only simplifies the process of adding and altering information but also contributes to enhanced understanding and accessibility, making it an invaluable tool in the realm of knowledge management and processing.



Examples of Semantic Network

These examples illustrate how semantic networks can represent hierarchical relationships and classifications in various domains, from biology to programming, making them a versatile tool for organizing and understanding complex information.

Animal Classification:

Nodes: Lion, Tiger, Bear, Wolf

Links: “is a” relation

Labels: Mammal, Carnivore, Predator

Frames in AI:

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 by representing stereotypes situations. 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: The various aspects of a slot is known as **Facets**. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

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: 1

Let's take an 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 2:

Let's suppose we are taking 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

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. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

Constraint Propagation: While selecting unassigned variable for computation if we look at some of the constraints earlier in the search (or even before search begins), we can drastically reduce search space. Following are techniques which are helpful for earlier constraints checks.

Forward Checking (FC)

1. Forward checking is one of the potential technique which uses constraints more effectively during search.

2. It removes values in neighbouring unassigned variables domain that conflict with assigned variable.

3. Forward checking uses MRV to select assigned variable.

4. Backtracking search is performed if failure occurs.

5. Search terminates when any variable has no legal values.

6. Generic method, do

i) Assigns variable X.

ii) Forward checking remarks unassigned variable Y connected to X.

iii) Remove values from D, where value is inconsistent with X.

Constraint Propagation using Arc Consistency

1. It is fast method of constraint propagation.

2. X→Y is consistent if (for every value of X there is some allowed value Y. For (a) example [$V_2 \rightarrow V_1$, is consistent iff V_1 = Red, V_2 = Blue] (i.e. for every value of x in X there is some allowed value y in Y). This is directed property example - [$V_1 = V_2$]

$V_1 = V_2$ is consistent iff

V_1 = Red and V_2 = Blue

3. As directed arcs between variables represent the domains of specified variables, they are consistent with each other.

4. Constraint propagation can be applied as preprocessing or propagation step.

i) Before search- Preprocessing.

ii) After search- Propagation.

5. The procedure for maintaining arc consistency, can be applied repeatedly.

Constraint Propagation using K-consistency

1. Arc consistency is not capable of detecing all inconsistencies. Partial assignments [V_1 = Red, V_2 = Red] are inconsistent.

2. K-consistency is very strong form of constraint propagation.

3. A CSP is K-consistency if for any set of K-1 variables and for any consistent assignment to those variable a consistent value can always be assigned to any K^{th} variable.

Note

1. Consistency means that each individual variable by itself is consistent; this is also called as node consistency.

2. Consistency is the same as arc consistency.

3. Consistency means that any pair of adjacent-variables can always be extended toa third neighboring variable; this is also called as path consistency.

4. Strongly, K-consistency graph exhibit some properties mentioned below -

i) It is K-consistent.

ii) It is also (K-1) consistent, (K-2) consistent all the way down to 1 consistent. 5. This is an idealist solution which requires O (nd) time instead of O (n^2 d^3).

• An exponential order time is required for establishing n-consistency in the worstcase.

Representing knowledge using rules:

Procedural and Declarative Knowledge

1. Procedural Knowledge

- The Procedural knowledge is a type of knowledge where the essential control information that is required to use the information is integrated in the knowledge itself.
- It also used with an interpreter to employ the knowledge which follows the instructions given in the knowledge.
- Ex - It can include a group of logical assertions merged with a resolution theorem prove to provide an absolute program for solving problems. Here, the implied income tax of an employee salary can be thought of as a procedural knowledge as it would require a process to calculate it as given below.
- So, this is how the tax of an employee is calculated by following a lengthy process instead of just collecting facts.

= GTI (Gross Taxable Income) = Annual Salary of an employee - (Standard deduction + deduction under section 80C)
= Tax computed on GTI (according to slab rate) = A,
= Rebate under section 87A = B;
= Total tax = A - Less B + add: health and education Cess @ 4% on (A-B)

2. Declarative Knowledge

- A Declarative knowledge is where only knowledge is described but not the use to which the knowledge is employed is not provided.
- So, in order to use this declarative knowledge, we need to add it with a program that indicates what is to be done to the knowledge and how it is to be done.

Difference the Procedural and Declarative Knowledge

PROCEDURAL KNOWLEDGE	DECLARATIVE KNOWLEDGE
It is also known as Interpretive knowledge.	It is also known as Descriptive knowledge.
Procedural Knowledge means how a particular thing can be accomplished	While Declarative Knowledge means basic knowledge about something.
Procedural Knowledge is generally not used means it is not more popular.	Declarative Knowledge is more popular.
Procedural Knowledge can't be easily communicate.	Declarative Knowledge can be easily communicate
Procedural Knowledge is generally process oriented in nature	Declarative Knowledge is data oriented in nature.
In Procedural Knowledge debugging and validation is not easy.	In Declarative Knowledge debugging and validation is easy.

Forward Reasoning

- The solution of a problem generally includes the initial data and facts in order to arrive at the solution. These unknown 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. 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

1. In the first step, the system is given one or more than one constraints.

2. Then the rules are searched in the knowledge base for each constraint. The rules that fulfill the condition are selected(i.e., IF part).

3. 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.

4. 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

1. Firstly, the goal state and the rules are selected where the goal state reside in the THEN part as the conclusion.

2. From the IF part of the selected rule the sub goals are made to be satisfied for the goal state to be true.

3. Set initial conditions important to satisfy all the sub goals.

4. Verify whether the provided initial state matches with the established states. If it fulfills the condition then the goal is the solution otherwise other goal state is selected.

Difference between backward chaining and forward chaining

Forward Chaining	Backward Chaining
It starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	It starts from the goal and works backward through inference rules to find the required facts that support the goal.
It is a bottom-up approach	It is a top-down approach
It is known as data-driven inference technique as we reach to the goal using the available data.	It is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
It applies a breadth-first search strategy.	It applies a depth-first search strategy.
It tests for all the available rules	It tests only for few required rules.
It is suitable for the planning, monitoring, control, and interpretation application.	It is suitable for diagnostic, prescription, and debugging application.
It can generate an infinite number of possible conclusions.	It generates a finite number of possible conclusions.
It operates in the forward direction.	It operates in the backward direction.

Rule- based deduction systems

The way in which a piece of knowledge is expressed by a human expert carries important information,

Example: if the person has fever and feels tummy-pain then she may have an infection.

In logic it can be expressed as follows:

$$\forall x.\ (has\_fever(x)\ \&\ tummy\_pain(x) \rightarrow has\_an\_infection(x))$$

If we convert this formula to clausal form we loose the content as then we may have equivalent formulas like:

(i) has_fever(x) & ~has_an_infection(x) → ~tummy_pain(x)
(ii) ~has_an_infection(x) & tummy_pain(x) → ~has_fever(x)

Notice that: (i) and (ii) are logically equivalent to the original sentence

- They have lost the main information contained in its formulation.

Forward production systems

The main idea behind the forward/backward production systems is:

- to take advantage of the implicational form in which production rules are stated by the expert
- and use that information to help achieving the goal.

In the present systems the formulas have two forms:

- rules
- and facts

Rules are the productions stated in implication form.

- Rules express specific knowledge about the problem.
- Facts are assertions not expressed as implications.
- The task of the system will be to prove a goal formula with these facts and rules.
- In a forward production system the rules are expressed as F-rules
- F-rules operate on the global database of facts until the termination condition is achieved.
- This sort of proving system is a direct system rather than a refutation system.

Facts

- Facts are expressed in AND/OR form.
- An expression in AND/OR form consists on sub-expressions of literals connected by & and V symbols.
- An expression in AND/OR form is not in clausal form.

Steps to transform facts into AND/OR form for forward system:

1. Eliminate (temporarily) implication symbols.

2. Reverse quantification of variables in first disjunct by moving negation symbol.

3. Skolemize existential variables.

4. Move all universal quantifiers to the front an drop.

5. Rename variables so the same variable does not occur in different main conjuncts

-Main conjuncts are small AND/OR trees, not necessarily sum of literal clauses as in Prolog.

Steps to transform the rules into a free-quantifier form:

1. Eliminate (temporarily) implication symbols.

2. Reverse quantification of variables in first disjunct by moving negation symbol.

3. Skolemize existential variables.

4. Move all universal quantifiers to the front and drop.

5. Restore implication.

All variables appearing on the final expressions are assumed to be universally quantified.

Backward production systems

B-Rules: We restrict B-rules to expressions of the form: W ==> L, where W is an expression in AND/OR form and L is a literal, and the scope of quantification of any variables in the implication is the entire implication.

Recall that W==>(L1 & L2) is equivalent to the two rules: W==>L1 and W==>L2.

An important property of logic is the duality between assertions and goals in theorem-proving systems. Duality between assertions and goals allows the goal expression to be treated as if it were an assertion.

Conversion of the goal expression into AND/OR form:

1. Elimination of implication symbols.

2. Move negation symbols in.

3. Skolemize existential variables.

4. Drop existential quantifiers. Variables remaining in the AND/OR form are considered to be existentially quantified.

Goal clauses are conjunctions of literals and the disjunction of these clauses is the clause form of the goal well-formed formula.

Rules

R1: manager(x,y) □ works_in(x,y)

R2: works_in(x,y) & manager(x,z) □ boss_of(y,z)

R3: works_in(x,y) & works_in(x,z) □ ~married(y,z)

R4: married(y,z) □ married(z,y)

R5: [married(x,y) & works_in(p-d,x) □ insured_by(x,eagle-corp)

With these facts and rules a simple backward production system can answer a variety of questions.

Build solution graphs for the following questions:

1. Name someone who works in the Purchasing Department.

2. Name someone who is married and works in the sales department.

3. Who is Joe Smith's boss?

4. Name someone insured by Eagle Corporation.

5. Is John Jones married with Sally Jones?

Reasoning under uncertainty:

In AI, there are numerous sources of uncertainty, including variation in specific data values and the sample of data collected from the domain. Probabilistic methods can be used to manage the inherent uncertainty in AI.

In AI, reasoning is the process of drawing conclusions or inferring something new about a domain of interest based on the knowledge we have. It is an essential component of what we refer to as "intelligence." In fact, this is the distinction between a traditional database and a knowledge base. Unlike the expert system's knowledge base, the database can't think and can only answer a limited number of specific questions.

Types of reasoning: Reasoning can go in one of two directions: forward toward the goal or backward away from the goal. In AI, both are used in different situations.

- Forward reasoning starts with known facts and works its way to the desired outcome. This is an example of deductive reasoning in action.
- Backward reasoning starts with the goal and creates sub-goals that must be completed before the main goal can be completed. This is an example of inductive reasoning in action.

Tools used: Using methods from probability theory and economics, AI researchers have developed a number of tools to solve these problems. The following are some of the most commonly used tools.

Bayesian network: A Bayesian network is a probabilistic graphical model that uses a directed acyclic graph (DAG) to represent a set of variables and their conditional dependencies. Bayesian networks are ideal for predicting the likelihood that any one of several possible known causes contributed to an event that occurred. For example, it could be used to represent the probabilistic relationships between diseases and symptoms. The network can be used to figure out how likely it is that different diseases are present based on the symptoms they have.

Hidden Markov model: The Hidden Markov Model (HMM) is a type of Markov model in which the modelled system is assumed to be a Markov process. Statistical mechanics, thermodynamics, physics, chemistry, finance, signal processing, and information theory are all fields that use hidden Markov models. Hidden Markov models are also used in pattern recognition applications like speech and handwriting recognition, gesture recognition, part-of-speech tagging, musical score following, partial discharge, and bioinformatics.

Kalman filter: Kalman filtering is a statistical and control theory algorithm that uses time series measurements with statistical noise and other errors. In addition, Kalman filtering has many technological uses. Common applications include dynamically positioned aircraft, spacecraft, and ships. Kalman filtering is also widely used in time series analysis for topics like signal processing and econometrics. Kalman filtering is a common topic in robotic motion planning and control, and can be used to optimize a trajectory. Furthermore, Kalman filtering can model the brain's control of movement.

Particle filter: Particle filters, also known as sequential Monte Carlo methods, are a family of Monte Carlo algorithms for solving filtering problems in signal processing and Bayesian statistical inference. Del Moral coined the term "particle filters" in 1996 to describe mean-field interacting particle methods, which have been used in fluid mechanics since the 1960s. In 1998, Liu and Chen coined the phrase "Sequential Monte Carlo." Moreover, Particle filtering is a way to show the probability distribution of a stochastic process based on incomplete or noisy data. It does this by using a set of particles, which are also called samples.

Decision theory: The study of an agent's choices is called decision theory (or theory of choice; not to be confused with choice theory). Economists, mathematicians, data scientists, psychologists, biologists, social scientists, philosophers, and computer scientists all study decision theory, which is closely related to the field of game theory.

Review of probability:

Why does AI need uncertainty?

- Reason: Sh*t happens
- Actions don’t have deterministic outcomes
- Can logic be the “language” of AI???
- Problem: General logical statements are almost always false
- Truthful and accurate statements about the world would seem to require an endless list of qualifications
- How do you start a car?
- Call this “The Qualification Problem”

The Qualification Problem

- Is this a real concern?
- YES!
- Systems that try to avoid dealing with uncertainty tend to be brittle.
- Plans fail
- Finding shortest path to goal isn’t that great if the path doesn’t really get you to the goal

Probabilities

- Natural way to represent uncertainty
- People have intuitive notions about probabilities
- Many of these are wrongor inconsistent
- Most people don’t getwhat probabilities mean
- Finer details of this question still debated

Bogus Probabilistic Reasoning

- Is the sequence 123456 any less likely than any other sequence of lottery numbers?
- Is it good to bet on rare events because they are "due" to come up?
- Cancer clusters
- The myth of the "hot hand"

Relative Frequencies

- Probabilities defined over events
- Space of all possible events is the "event space"
- Think: Playing blindfolded darts with the Venn diagram…
- P(A) ≅percentage of dart throws that hit A (assuming a uniform distribution of dart hits over the area of the box)

Understanding Probabilities

- Initially, probabilities are "relative frequencies"
- This works well for dice and coin flips
- For more complicated events, this is problematic
- Probability Trump runs as GOP nominee in 2016?–This event only happens once–We can't count frequencies–Still seems like a meaningful question
- In general, all events are unique
- "Reference Class" problem

Probabilities and Beliefs

- Suppose I have flipped a coin and hidden the outcome
- What is P(Heads)?
- Note that this is a statement about a belief, not a statement about the world
- The world is in exactly one state (at the macro level) and it is in that state with probability 1.
- Assigning truth values to probability statements is very tricky business
- Must reference speakers state of knowledge

Why probabilities are good

- Subjectivists: probabilities are degrees of belief
- Are all degrees of belief probability?–AI has used many notions of belief:
- Certainty Factors
- Fuzzy Logic
- Can prove that a person who holds a system of beliefs inconsistent with probability theory can be tricked into accepted a sequence of bets that is guaranteed to lose (Dutch book) in expectation

Atomic Events

- When several variables are involved, it is useful to think about atomic events
- Complete assignment to variables in the domain (compare with states in robot planning)

–Atomic events are mutually exclusive

–Exhaust space of all possible events

- For n binary variables, how many unique atomic events are there?

Why Probabilities Are Messy

- Probabilities are not truth-functional
- Computing P(a and b) requires the joint distribution–sum out all of the other variables from the distribution–It is not a function of P(a) and P(b)–It is not a function of P(a) and P(b)–It is not a function of P(a) and P(b)
- This fact led to many approximations methods such as certainty factors and fuzzy logic (Why?)
- Neat vs. Scruffy…

Conditional Probabilities

- Ordinary probabilities for random variables: unconditionalor priorprobabilities
- P(a|b) = P(a AND b)/P(b)
- This tells us the probability of a given that we know onlyb
- If we know c and d, we can't use P(a|b) directly (without additional assumptions)
- Annoying, but solves the qualification problem…

Probability Solves the Qualification Problem

- P(disease|symptom1)
- Defines the probability of a disease given that we have observed onlysymptom1
- The conditioning bar indicates that the probability is defined with respect to a particular state of knowledge, not as an absolute thing

Probability Conclusions

- Probabilistic reasoning has many advantages:

– Solves qualification problem

– Is better than any other system of beliefs (Dutch book argument)

- Probabilistic reasoning is tricky

– Some things decompose nicely: linearity of expectation, conjunctions of independent events, disjunctions of disjoint events

– Some things can be counterintuitive at first: conjunctions of arbitrary events, conditional probability

- Reasoning efficiently with probabilities poses significant data structure and algorithmic challenges for AI (Roughly speaking, the AI community realized sometime around 1990 that probabilities were the right thing and has spent the last 20 years grappling with this realization.)

Bayes' probabilistic interferences:

Probabilistic inference in Bayesian networks refers to the process of reasoning and making predictions about the probability distributions of unobserved variables given observed evidence or data. It utilizes the graphical structure and probabilistic dependencies encoded in the Bayesian network to perform inference.

In a Bayesian network, the joint probability distribution of all variables can be factorized using the chain rule of probability and the conditional independence assumptions represented by the network's structure. This factorization allows for efficient probabilistic inference.

Probabilistic inference in Bayesian networks involves two main tasks:

1. **Querying:** When provided with observed evidence, the objective is to determine the probability distribution of one or more target variables of interest. This involves conditioning the observed evidence and propagating the probabilities throughout the network to obtain the desired probability distribution.
2. **Learning:** This task involves updating the probabilities or parameters in the Bayesian network based on observed data. It aims to refine the network's structure and probability tables to reflect the data better.

Example 1: A person has undertaken a job. The probabilities of completion of the job on time with and without rain are 0.44 and 0.95 respectively. If the probability that it will rain is 0.45, then determine the probability that the job will be completed on time.

Solution:
Let E1 be the event that the mining job will be completed on time and E2 be the event that it rains. We have,
P(A) = 0.45,
P(no rain) = P(B) = 1 − P(A) = 1 − 0.45 = 0.55
By multiplication law of probability,
P(E1) = 0.44, and P(E2) = 0.95
Since, events A and B form partitions of the sample space S, by total probability theorem, we have
P(E) = P(A) P(E1) + P(B) P(E2)
⇒ P(E) = 0.45 × 0.44 + 0.55 × 0.95
⇒ P(E) = 0.198 + 0.5225 = 0.7205
So, the probability that the job will be completed on time is 0.7205

Dempstershafer theory:

Dempster-Shafer Theory (DST) is a theory of evidence that has its roots in the work of Dempster and Shafer. While traditional probability theory is limited to assigning probabilities to mutually exclusive single events, DST extends this to sets of events in a finite discrete space. This generalization allows DST to handle evidence associated with multiple possible events, enabling it to represent uncertainty in a more meaningful way. DST also provides a more flexible and precise approach to handling uncertain information without relying on additional assumptions about events within an evidential set.

Where sufficient evidence is present to assign probabilities to single events, the Dempster-Shafer model can collapse to the traditional probabilistic formulation. Additionally, one of the most significant features of DST is its ability to handle different levels of precision regarding information without requiring further assumptions. This characteristic enables the direct representation of uncertainty in system responses, where an imprecise input can be characterized by a set or interval, and the resulting output is also a set or interval.

The incorporation of Dempster Shafer theory in artificial intelligence allows for a more comprehensive treatment of uncertainty. By leveraging the unique features of this theory, AI systems can better navigate uncertain scenarios, leveraging the potential of multiple evidentiary types and effectively managing conflicts. The utilization of Dempster Shafer theory in artificial intelligence empowers decision-making processes in the face of uncertainty and enhances the robustness of AI systems. Therefore, Dempster-Shafer theory is a powerful tool for building AI systems that can handle complex uncertain scenarios.

The Uncertainty in this Model

At its core, DST represents uncertainty using a mathematical object called a belief function. This belief function assigns degrees of belief to various hypotheses or propositions, allowing for a nuanced representation of uncertainty. Three crucial points illustrate the nature of uncertainty within this theory:

1. **Conflict:** In DST, uncertainty arises from conflicting evidence or incomplete information. The theory captures these conflicts and provides mechanisms to manage and quantify them, enabling AI systems to reason effectively.
2. **Combination Rule:** DST employs a combination rule known as Dempster's rule of combination to merge evidence from different sources. This rule handles conflicts between sources and determines the overall belief in different hypotheses based on the available evidence.
3. **Mass Function:** The mass function, denoted as m(K), quantifies the belief assigned to a set of hypotheses, denoted as K. It provides a measure of uncertainty by allocating probabilities to various hypotheses, reflecting the degree of support each hypothesis has from the available evidence.

Example

Consider a scenario in artificial intelligence (AI) where an AI system is tasked with solving a murder mystery using Dempster–Shafer Theory. The setting is a room with four individuals: A, B, C, and D. Suddenly, the lights go out, and upon their return, B is discovered dead, having been stabbed in the back with a knife. No one entered or exited the room, and it is known that B did not commit suicide. The objective is to identify the murderer.

To address this challenge using Dempster–Shafer Theory, we can explore various possibilities:

1. **Possibility 1:** The murderer could be either A, C, or D.
2. **Possibility 2:** The murderer could be a combination of two individuals, such as A and C, C and D, or A and D.
3. **Possibility 3:** All three individuals, A, C, and D, might be involved in the crime.
4. **Possibility 4:** None of the individuals present in the room is the murderer.

To find the murderer using Dempster–Shafer Theory, we can examine the evidence and assign measures of plausibility to each possibility. We create a set of possible conclusions (P)(P) with individual elements {p1,p2,...,pn}{p1,p2,...,pn}, where at least one element (p)(p) must be true. These elements must be mutually exclusive.

By constructing the power set, which contains all possible subsets, we can analyze the evidence. For instance,

if P={a,b,c}P={a,b,c}, the power set would be

{o,{a},{b},{c},{a,b},{b,c},{a,c},{a,b,c}}{o,{a},{b},{c},{a,b},{b,c},{a,c},{a,b,c}},
comprising 23=823=8 elements.

Mass function m(K)

In Dempster–Shafer Theory, the mass function m(K) represents evidence for a hypothesis or subset K. It denotes that evidence for {K or B} cannot be further divided into more specific beliefs for K and B.

Belief in K

The belief in KK, denoted as Bel(K)$Bel(K)$, is calculated by summing the masses of the subsets that belong to KK. For example, if K={a,d,c},Bel(K)$K=\{a,d,c\},Bel(K)$ would be calculated as m(a)+m(d)+m(c)+m(a,d)+m(a,c)+m(d,c)+m(a,d,c)$m(a)+m(d)+m(c)+m(a,d)+m(a,c)+m(d,c)+m(a,d,c)$.

Plausibility in K

Plausibility in KK, denoted as Pl(K)$Pl(K)$, is determined by summing the masses of sets that intersect with KK. It represents the cumulative evidence supporting the possibility of K being true. Pl(K)$Pl(K)$ is computed as

m(a)+m(d)+m(c)+m(a,d)+m(d,c)+m(a,c)+m(a,d,c)$m(a)+m(d)+m(c)+m(a,d)+m(d,c)+m(a,c)+m(a,d,c)$.

By leveraging Dempster–Shafer Theory in AI, we can analyze the evidence, assign masses to subsets of possible conclusions, and calculate beliefs and plausibilities to infer the most likely murderer in this murder mystery scenario.

Characteristics of Dempster Shafer Theory

Dempster Shafer Theory in artificial intelligence (AI) exhibits several notable characteristics:

1. **Handling Ignorance:** Dempster Shafer Theory encompasses a unique aspect related to ignorance, where the aggregation of probabilities for all events sums up to 1. This peculiar trait allows the theory to effectively address situations involving incomplete or missing information.
2. **Reduction of Ignorance:** In this theory, ignorance is gradually diminished through the accumulation of additional evidence. By incorporating more and more evidence, Dempster Shafer Theory enables AI systems to make more informed and precise decisions, thereby reducing uncertainties.
3. **Combination Rule:** The theory employs a combination rule to effectively merge and integrate various types of possibilities. This rule allows for the synthesis of different pieces of evidence, enabling AI systems to arrive at comprehensive and robust conclusions by considering the diverse perspectives presented.

By leveraging these distinct characteristics, Dempster Shafer Theory proves to be a valuable tool in the field of artificial intelligence, empowering systems to handle ignorance, reduce uncertainties, and combine multiple types of evidence for more accurate decision-making.

Advantages and Disadvantages

Dempster Shafer Theory in Artificial Intelligence (AI) Offers Numerous Benefits:

1. Firstly, it presents a systematic and well-founded framework for effectively managing uncertain information and making informed decisions in the face of uncertainty.

2. Secondly, the application of Dempster–Shafer Theory allows for the integration and fusion of diverse sources of evidence, enhancing the robustness of decision-making processes in AI systems.
3. Moreover, this theory caters to the handling of incomplete or conflicting information, which is a common occurrence in real-world scenarios encountered in artificial intelligence.

Nevertheless, it is Crucial to Acknowledge Certain Limitations Associated with the Utilization of Dempster Shafer Theory in Artificial Intelligence:

1. One drawback is that the computational complexity of DST increases significantly when confronted with a substantial number of events or sources of evidence, resulting in potential performance challenges.
2. Furthermore, the process of combining evidence using Dempster–Shafer Theory necessitates careful modeling and calibration to ensure accurate and reliable outcomes.
3. Additionally, the interpretation of belief and plausibility values in DST may possess subjectivity, introducing the possibility of biases influencing decision-making processes in artificial intelligence.