UNIT-III (KNOWLEDGE ENGINEERING)

KNOWLEDGE ENGINEERING: The general process of constructing a knowledge base is called knowledge engineering. A knowledge engineer is who knows about a particular domain, learns what concepts are important in that domain, and creates a formal representation of the objects and relations in the domain.

The knowledge engineering process. Knowledge engineering projects vary widely in content, scope, and difficulty, but all such projects include the following steps: 1. Identify the task – The knowledge engineer must delineate (ie) describe the range of questions that the knowledge base will support and the kinds of facts that will be available for each specific problem instance. 2. Assemble the relevant knowledge – The knowledge engineer might already be an expert in the domain to extract what they know this process is called Knowledge Acquisition. At this stage, the knowledge is not represented and the main idea is is to understand the scope of the knowledge base, as determined by the task, and to understand how the domain actually works. 3. Decide on a vocabulary of predicates, functions, and constants - Translate the important domain-level concepts into logic-level names. Once the choices have been made, the result is a vocabulary that is known as the ontology of the domain. The word *ontology* means a particular theory of the nature of being or existence. The ontology determines what kinds of things exist, but does not determine their specific properties and interrelationships. 4. Encode general knowledge about the domain - The knowledge engineer writes down the axioms for all the vocabulary terms (ie) enabling the expert to check the content. This step reveals misconceptions or gaps in the vocabulary that must be fixed by returning to step 3 and iterating through the process. 5. Encode a description of the specific problem instance - It will involve writing simple atomic sentences about instances of concepts that are already part of the ontology. 6. Pose queries to the inference procedure and get answers - we can let the inference procedure operate on the axioms and problem-specific facts to derive the facts we are interested in knowing. 7. Debug the knowledge base – Answers will be correct for the knowledge base as written, assuming that the inference procedure is sound, but they will not be the ones that the user is expecting. A debugging process could confirm missing axioms or axioms that are too weak can be identified easily by noticing places where the chain of reasoning stops unexpectedly.

Cognitive architecture: It is a models of human reasoning. In such systems, the "working memory" of the system models human short-term memory, and the productions are part of long-term memory.

LOGICAL AGENTS

• The ***representation*** of knowledge and the ***reasoning*** processes that bring knowledge to life-are central to the entire field of artificial intelligence.
• Knowledge and reasoning are enable successful behaviors that would be very hard to achieve otherwise.
• Knowledge-based agents can benefit from knowledge expressed in very general forms, combining and recombining information to suit myriad purposes

• Knowledge and reasoning also play a crucial role in dealing with ***partially observable*** **environments.**
• *A* knowledge-based agent can combine general knowledge with current percepts to infer hidden aspects of the current state prior to selecting actions.

Knowledge Based Agents
• The central component of a knowledge-based agent is **its knowledge base, or KB**
• **SENTENCE** - a knowledge base is a set of sentences.
• Knowledge Representation Language:-Each sentence is expressed in a language called a **Knowledge Representation Language** and represents some assertion about the world.
• **Inference** :- There is a way to add new sentences to the knowledge base is TELL and to query what is know is ASK.

TELL :- A way to add new sentences to the knowledge base
ASK :- A way to query what is known.
Both tasks may involve **INFERENCE** -that is, deriving new sentences from old.
• In LOGICAL AGENTS, inference must obey the fundamental requirement that when one ASKS a question of the knowledge base, the answer should follow from what has been told (or rather, TELLED) to the knowledge base previously.

Knowledge Based Agent Program
function*KB-AGEN T(percept) retu***rns** an *action*
static: *KB, a knowledge base*
t, a counter, initially 0, indicating time
TELL(KB, MAKE-PERCEPT-SENTENCE(*percept* ,t*))*
action ← *AsK(KB,* MAKE-ACTION- QUERY(t))
TELL(KB, **MAKE-ACTION-SENTENCE(action,***t))*
T ←*t+1*
return*action*
fig 3.1 A generic knowledge-based agent
• Like all our agents, it takes a percept as input and returns an action.
• The agent maintains a knowledge base, KB, which may initially contain some **background knowledge.** Each time the agent program is called, it does three things.
• First, it TELLS the knowledge base what it perceives.
• Second, it ASKS the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.
• Third, the agent records its choice with TELL and executes the action.
• The second TELL is necessary to let the knowledge base know that the hypothetical *action* has actually been executed.
• The details of the Representation Language are hidden inside three functions that implement the interface between the sensors and actuators and the core representation and reasoning system.
• MAKE-PERCEPT-SENTENCE takes a percept and a time and returns a sentence asserting that the agent perceived the given percept at the given time.

• MAKE-ACTION-QUERY takes a time as input and returns a sentence that asks what action should be done at the current time.

• MAKE-ACTION-SENTENCE constructs a sentence asserting that the chosen action was executed. The details of the inference mechanisms are hidden inside TELL and ASK.

Knowledge level:- The knowledge-based agent is not an arbitrary program for calculating actions. It is amenable to a description at the knowledge level, where we need specify only what the agent knows and what its goals are, in order to fix its behavior.
Implementation level : For example : an automated taxi might have the goal of dropping a passenger to Marin County and might know that it is in San Francisco and that the Golden Gate Bridge is the only link between the two locations. Then we can expect it to cross the Golden Gate Bridge ***because it knows that that will achieve its goal. This analysis is independent of how the taxi works at the implementation level.*** It doesn't matter whether its geographical knowledge is implemented as linked lists or pixel maps, or whether it reasons by manipulating strings of symbols stored in registers or by propagating noisy signals in a network of neurons.
• One can build a knowledge-based agent simply by Telling it what it needs to know.
• The agent's initial program, before it starts to receive percepts, is built by adding one by one the sentences that represent the designer's knowledge of the environment.
• Designing the representation language to make it easy to express this knowledge in the form of sentences simplifies the construction problem enormously. This is called the **declarative approach to system building.**
• The **procedural approach** encodes desired behaviors directly as program code; minimizing the role of explicit representation and reasoning can result in a much more efficient system

LOGIC
• Knowledge bases consists of sentences. These sentences are expressed according to the syntax of the representation language, which specifies all the sentences that are well formed.
• The notion of syntax is clear enough in ordinary arithmetic:
• *"x + y = 4" is a well-formed sentence, whereas "x2y+ =" is not.*
• *The syntax of logical* languages is usually designed for writing papers and books.

SEMANTICS
• A logic must also define the semantics of the language.
• Semantics means "meaning" of sentences. In logic, the definition is more precise.
• The semantics of the language defines the truth of each sentence with respect to each possible world.
• For example, ***"x + y = 4" is true in a world where x is 2 and y is 2, but false in a world where x is 1*** **and** ***y is 1.1*** **.**
• *In standard logics, every sentence must be either true or false in each possible* world-there is no "in between.

Example : Assume x and *y* as the number of men and women sitting at a table playing bridge, for example, and the sentence $x + y = 4$ is true when there are four in total; formally, the possible models are just all possible assignments of numbers to the variables *x* and *y*. Each such assignment fixes the truth of any sentence of arithmetic whose variables are *x* and *y*.

Entailment

• Entailment means that one thing follows from another: Relation of logical entailment between sentences is involved that a sentence follows logical from another sentence.

• KB ⊨ α Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

– E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

– E.g., x+y = 4 entails 4 = x+y

– Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Models

• Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

• We say *m* is a model of a sentence α if α is true in *m*

• *M(α)* is the set of all models of α

• Then KB ⊨ α if *M(KB)* □ *M(*α)

– E.g. *KB*=Giants won and Reds won α = Giants won

We can apply the same kind of analysis to the wumpus-world reasoning example given in the preceding section. Consider the situation in Figure 3.3(b): the agent has detected nothing in [1,1] and a breeze in [2,1]. These percepts, combined with the agent's knowledge of the rules of the wumpus world (the PEAS description on page 197), constitute the KB. The agent is interested (among other things) in whether the adjacent squares [1,2], [2,2], and [3,1] contain pits. Each of the three squares might or might not contain a pit, so (for the purposes of this example) there are 23 = 8 possible models. These are shown in Figure 3.5.

Fig 3.5 : Possible models for the presence of pits in squares [1,2], [2,2], and [3,1], given observations of nothing in [1,1] and a breeze in [2,1]. (a) Models of the knowledge base and Ctl (no pit in [1,2]). (b) Models of the knowledge base and Ct2 (no pit in [2,2]).

Conclusion

• The KB is false in models that contradict what the agent knows-for example, the KB is false in any model in which [l,2] contains a pit, because there is no breeze in [1,1].

• Now let us consider two possible conclusions:

• ***α1= "There is no pit in [1,2]."***

• ***α2= "There is no pit in [2,2]."***

• In every model in which ***KB is true, α1 is also true.***

• Hence, ***KB ≠ α1 : there is no pit in [1,2].***

• In some models in which ***KB is true, α2 is false.***
• Hence, ***KB ¥ α2 :so, the agent cannot conclude that there is no pit in [2,2].***

Logical Inference : The above shown example illustrates entailment and also show how the definition of entailment can be applied to derived conclusions (i.e) to carry out logical inference.
Model Checking : The inference algorithm illustrated in Figure 3.5 is called model checking, because it enumerates all possible models to check the α is true in all models in which KB is true.
If an inference algorithm I can derive α from KB, then
KB ├i α =
Pronunciation for above notation is :
"α is derived from KB by i" (or) " i derives α from KB".
Sound : An inference algorithm that derives only entailed sentences is called sound or truth preserving.
• *i*is sound if whenever *KB* ├i α, it is also true that *KB* ╞ α
Soundness is a highly desirable property. An unsound inference procedure essentially makes things up as it goes along-it announces the discovery of nonexistent needles.
Complete: an inference algorithm is complete if it can derive any sentence that is entailed.
• *i*is complete if whenever *KB* ╞ α, it is also true that *KB* ├i α

if KB is true in the real *world, then any sentence 0: derived from KB by a sound inference procedure is also true in the real world.* So, while an inference process operates on "syntax"-internal physical configurations such as bits in registers or patterns of electrical blips in brains-the process *corresponds* to the real-world relationship whereby some aspect of the real world is the case. Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.
Illustration:
• Sentences are physical configurations of the agent.
• Reasoning is a process of constructing new physical configurations from old ones.
• Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

PROPOSITIONAL LOGIC
• Propositional logic is the simplest logic – illustrates basic ideas.
• the syntax of propositional logic and its semantics-the way in which the truth of sentences is determined.
• Then Look at entailment-the relation between a sentence and another sentence that follows from it-and see how this leads to a simple algorithm for logical inference. Everything takes place, of course, in the wumpus world.

• **Syntax** :-The syntax of propositional logic defines the allowable sentences.
• **The atomic sentences** - the indivisible syntactic elements - consist of a single proposition symbol. Each such symbol stands for a proposition that can be true or false.

Rules:
i. Uppercase names are used for symbols (ie) P,Q,R and so on.
ii. Names are Arbitrary For example, we might use *W1,3 to stand for the proposition* that the wumpus is in [1,3]. There are two proposition symbols with fixed meanings:
True is the always-true proposition and False is the always-false proposition.
□ Complex Sentences : Complex sentences are constructed from simpler sentences using logical connections. There are five connectives.
□ If S is a sentence, □ S is a sentence (negation)
□ If S1 and S2 are sentences, S1 □ S2 is a sentence (conjunction)
□ If S1 and S2 aresentences, S1 □ S2 is a sentence (disjunction)
□ If S1 and S2 are sentences, S1 □ S2 is a sentence (implication)
□ If S1 and S2 are sentences, S1 □ S2 is a sentence (biconditional / IF AND ONLY IF)

□ (not). A sentence such as □*Wl,3*is called the negation of *Wl,3.* A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal).
/\ (and). A sentence whose main connective is /\, such as *Wl,3 /\ P3,1,* is called a conjunction; its parts are the conjuncts. (The /\ looks like an *"N"* for "And.")
V (or). A sentence using V, such as *(W1,3/\P3,1)* V*W2,2,* is a disjunction of the Disjuncts*(Wl,3/\ P3,1)* and *W2,2.*
□ (implies). A sentence such as*(W1,3/\ P3,1)* □ □*W2,2* is called an implication (or conditional). Its premise or antecedent is *(Wl,3/\ P3,1),*and its conclusion or consequent is □ W*2,2.* Implications are also known as rules or if-then statements.
□ (if and only if). The sentence*W1,3*□ -,*W2,2* is a biconditional.
• Every sentence constructed with binary connectives must be enclosed in parentheses. This ensures that we have to write *((A /\ B) □ C) instead of A /\ B □ C,.*
• *O*rder of precedence for the connectives is similar to the precedence used in arithmetic.

For example, *ab+ c is read as ((ab) + c) rather than a(b + c) because m*ultiplication has higher precedence than addition.

The order of precedence in propositional logic
• The order of precedence in propositional logic is (from highest to lowest) □ , □ , □ , □ ,and□ .
• Hence, the sentence □ P □*Q /\ R* □ *S* is equivalent to the sentence ((□*P)* □ *(Q /\ R))* □ *S.*
• Precedence does not resolve ambiguity in sentences such as *A /\ B /\ C, which could be read* as *((A /\ B) /\ C) or as (A /\ (B /\ C)). Because these two readings mean the same thing*
• We also allow *A* □ *B* □ *C and A /\ B /\ C and A* □ B□ C*are is allowed but A* □**B** □ C is not allowed because it has different meanings. So we should use parenthesis (c) .

Semantics
• The semantics defines the rules for determining the truth of a sentence with respect to a particular model.

• In propositional logic. a model simply fixes the truth value, true or *false for every* proposition symbol.
• For example, if the sentences in the knowledge base make use of the proposition symbols *Pl,2, P2,2, and P3,1, then one possible model is*
• m1 = *{P1,2 = false, P2,2 = false, P3,1 = true}* .
• With three proposition symbols, there are 23 = 8 possible models.
• The semantics for propositional logic must specify how to compute the truth value of *any sentence, given a model. This is done recursively. All sentences are constructed from* atomic sentences and the five connectives; therefore, we need to specify how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives.

Atomic sentences
Atomic sentences are easy:
• *True is true in every model and False is false in every model.*
• The truth value of every other proposition symbol must be specified directly in the model.
• For example, in the model m1 given earlier, *P1,2 is false.*

Complex sentences
• For any sentence *s and any model m, the sentence is true in m if and only if s is* false in m.
• Such rules reduce the truth of a complex sentence to the truth of simpler sentences.
• The rules for each connective can be summarized in a **truth table that specifies the truth value** of a complex sentence for each possible assignment of truth values to its components.

Truth tables for connectives Using this table, the truth value of any sentence S can be computed with respect to any model m by a simple process of recursive evaluation.
Each model specifies true/false for each proposition symbol
E.g. P1,2 P2,2 P3,1
false true false
With these symbols, 8 possible models, can be enumerated automatically.
Rules for evaluating truth with respect to a model *m*:
□ S is true if S is false
S1 □ S2 is true if S1 is true and S2 is true
S1 □ S2 is true if S1is true or S2 is true
S1 □ S2 is true if S1 is false or S2 is true
i.e., is false if S1 is true and S2 is false
S1 □ S2 is true if S1□ S2 is true and S2□ S1 is true
Simple recursive process evaluates an arbitrary sentence, e.g.,
□ P1,2□ (P2,2 □ P3,1) =*true* □ (*true* □ *false*) = *true* □ *true = true.*
Confusion:
(i) *P* V*Q*is true when *P* is true or *Q* is true *or both.* There is a different connective called "exclusive or" ("xor" for short) that yields false when both disjuncts are true." There is no consensus on the symbol for exclusive or; two choices are

(ii) Any implication is true whenever its antecedent is false.

For example,

(a) "5 is even implies Sam is smart" is true, regardless of whether Sam is smart. (b) *"P □ Q"* saying that "If *P* is true, then I am claiming that *Q* is true. Otherwise I am making no claim. This sentence could be *false* is if *P* is true but *Q* is false. From the truth table, Bidirectional P □ Q, it is true whenever both P □ Q and Q □ P ar true (ie) "P if and only if Q" or "P if Q". We have defined the semantics for propositional logic, we can construct a knowledge base for the wumpus world. To understand easily, we will deal only with pits alone, i,j□ values Let P i,j be true if there is a pit in [i, j]. Let B i, j be true if there is a breeze in [i, j]. The knowledge base includes the following sentences, each one labeled for convenience: There is no pit in [l,l]: R1: □ P1,1 • A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares: R2 : B1,1 □ (P1,2 □ P2,1). R3: B2,1□ (P1,1 □ P2,2 □ P3,1) • The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation R4: □ B1,1. R5: B2,1. The knowledge base, then, consists of sentences *RI* through *R5.* It can also be considered as a single sentence-the conjunction *RI* □ *R2* □ *R3* □ *R4*□ R5- because it asserts that all the individual sentences are true. Inference Logical inference is to decide whether *KB* ⊨ *a*a for some sentence *a.* Inference algorithm for will be a direct implementation of the definition of entailment: enumerate the models, and check that a is true in every model in which *KB* is true. **Equivalence, validity, and satisfiability:**Logical equivalence: two sentences α and β are logically equivalent if they are true in the same set of models. We write this as a α □ β. Example : Two sentences α and β are logically equivalent if they are true in same models: • α ≡ ß if and only if α ⊨ β and β ⊨ α. • Equivalence for any two sentences α and β is α ≡ ß if and only if α ⊨ β and β ⊨ α.

Validity: A sentence is valid if it is true in all models.e.g., A □ □ A, A □ A, (A □ (A □ B)) □ B. These are valid / true statements. Valid sentences are also known as tautologies-they are *necessarily* true. Because the sentence *True* is true in all models, every valid sentence is logically equivalent to *True. For any sentences KB and* α ,*KB* ⊨ α if and only if (*KB* □ α) is valid. Every valid implication sentence describes a inference.

Satisfiability: A sentence is satisfiable if it is true in *some* model. For example, the knowledge base given earlier, *(R1* □ *R2* □ *R3* □ *R4* □ *R s),* is satisfiable because there are three models in which it is true. If a sentence a is true in a model m, then we say that m satisfies α, or that m is a model of α. Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence. Validity and satisfiability are of course connected: α is valid if □ α is unsatisfiable; contrapositively, α is satisfiable if □ α is not valid. We also have the following useful result: α ⊨ β if and only if the sentence (α □ □ β)*is unsatisfiable.* To Prove: Proving β from α by checking the unsatisfiability of (α □ □ β) corresponds exactly to the standard mathematical proof technique of *reductin of absurdum.* It is called proof by refutation or by contradiction. Assumption : Sentence β to be false and shows that this leads to a contradiction with known axioms α . This contradiction is exactly what is meant by saying that the sentence (α □ □ β) is unsatisfiable

FIRST ORDER LOGIC (FOL): FOL, a representation language of knowledge which is powerful than propositional logic (ie) Boolean logic. FOL is an expression: It follows Procedural approach. It derive facts from other facts(ie.) dependent. **Fuzzy:**

The word "fuzzy"means "vagueness".Fuzzinessoccurswhen

theboundary of a piece of information is not clear-cut.Fuzzy sets have beenintroducedby Lotfi A. Zadeh (1965) as anextension of the classical notion o f set.

Classical set theory allows the membership of the elements in the setin binar y terms, a bivalent condition - an element either belongs or does not belong to the set.

Fuzzy set theory permits the gradual assessment of the membershipof elements i n a set, described withthe aid of a membership functionvalued in the real unit int erval [0, 1] .**Example:**

Words like young, tall, good, or high are fuzzy.− There is no single quantita tive value which defines the term young.− For some people, age 25 is young, and for o thers, age 35 is young.− The concept young has no clean boundary.− Age 1 is def initely young and age 100 is definitely not young;− Age 35 has some possibility of being young and usually dependson the context in which it is being considered.**Rules for Knowledge Representation:**

One way to represent knowledge is by using rules that express what must happen orwha t does happen when certain conditions are met.

Rules are usually expressed in the form of IF . . . THEN . . . statements, such as: IF ATH EN B This can be considered to have a similar logical meaning as the following:$A \rightarrow B$

A is called the antecedent and B is the consequent in this statement.

In expressing rules, the consequent usually takes the form of an action or acon clusion.

In other words, the purpose of a rule is usually to tell a system (such as an expertsyst em) what to do in certain circumstances, or what conclusions to draw from a set ofinputs about the current situation.

In general, a rule can have more than one antecedent, usually combined either byAN D or by OR (logically the same as the operators $\wedge$and $\vee$).

Similarly, a rule may have more than one consequent, which usually suggests thatther e are multiple actions to be taken.

In general, the antecedent of a rule compares an object with a possible value, using anope rator.For example, suitable antecedents in a rule might beIF x > 3, IF name is “Bob”, IF weather is cold.

Here, the objects being considered are x, name, and weather; the operators are “>”and “is”, and the values are 3, “Bob,” and cold.

Note that an object is not necessarily an object in the real-world sense— the weather isnot a real world object, but rather a state or condition of the world.

An object in this sense is simply a variable that represents some physical object orstat e in the real world.An example of a rule might beIF name is “Bob”, AND weather is cold, THEN tell Bob ‘Wear a coat’.

This is an example of a recommendation rule, which takes a set of inputsand givesadv ice as a result.

The conclusion of the rule is actually an action, and the action takes the form of arec ommendation to Bob that he should wear a coat.

In some cases, the rules provide more definite actions such as “move left” or “closedoo

r," in which case the rules are being used to represent directives.

Rules can also be used to represent relations such as:IF temperature is below 0THEN weat her is cold.

RuleBased Systems:

Rulebased systems or **production systems** are computer systems that use rules toprovid e recommendations or diagnoses, or to determine a course of action in aparticular situa tion or to solve a particular problem.

A rulebased system consists of a number of components:

- a database of rules (also calle d a **knowledge base**),
- a database of factsan **interpreter**, or **inference engine.**

In a rulebased system, the knowledge base consists of a set of rules that represent thekno wledge that the system has.

The database of facts represents inputs to the system that are used to derivecon clusions, or to cause actions.

The interpreter, or inference engine, is the part of the system that controls the processof deriving conclusions. It uses the rules and facts, and combines them together todraw concl usions.Using deduction to reach a conclusion from a set of antecedents is called **forwardc haining**.

An alternative method, **backward chaining**, starts from a conclusion and tries tosho w it by following a logical path backward from the conclusion to a set ofantecedents that are in the database of facts. **Semantic Network:**A semantic net (or semantic network) is a knowledge representation technique used for propositional information. So it is also called a propositional net. Semantic nets convey meaning. They are two dimensional representations of knowledge. Mathematically a semantic net can be defined as a labelled directed graph. Semantic nets consist of nodes, links (edges) and link labels. In the semantic network diagram, nodes appear as circles or ellipses or rectangles to represent objects such as physical objects, concepts or situations. Links appear as arrows to express the relationships between objects, and link labels specify particular relations. Relationships provide the basic structure for organizing knowledge. The objects and relations involved need not be so concrete. As nodes are associated with other nodes semantic nets are also referred to as associative nets.Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages.Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages. A neural network is an interconnected group of nodes, akin to the vast network of neurons in the human brain. The main categories of networks are acyclic or feedforward neural networks (where the signal passes in only one direction) and recurrent neural networks (which allow feedback and short-term memories of previous input events). Among the most popular feedforward networks are perceptrons, multi-layer perceptrons and radial basis networks.