ANSWER KEY IAT-3

Q.1 What is uncertainty (4)? Explain with two real-life examples (6).

Ans. Agents may need to handle uncertainty, whether due to partial observability, non-determinism, or a combination of the two. An agent may never know for certain what state it's in or where it will end up after a sequence of actions. We have seen problem-solving agents and logical agents designed to handle uncertainty by keeping track of a belief state—a representation of the set of all possible world states that it might be in—and generating a contingency plan that handles every possible eventuality that its sensors may report during execution. Despite its many virtues, however, this approach has significant drawbacks when taken literally as a recipe for creating agent programs:

• When interpreting partial sensor information, a logical agent must consider every logically possible explanation for the observations, no matter how unlikely. This leads to impossible large and complex belief-state representations.

• A correct contingent plan that handles every eventuality can grow arbitrarily large and must consider arbitrarily unlikely contingencies.

• Sometimes no plan is guaranteed to achieve the goal—yet the agent must act. It must have some way to compare the merits of plans that are not guaranteed.

Example 1: an automated taxi! automated has the goal of delivering a passenger to the airport on time. The agent forms a plan, A90, that involves leaving home 90 minutes before the flight departs and driving at a reasonable speed. Even though the airport is only about 5 miles away, a logical taxi agent will not be able to conclude with certainty that "Plan A90 will get us to the airport in time." Instead, it reaches the weaker conclusion "Plan A90 will get us to the airport in time, as long as the car doesn't break down or run out of gas, and I don't get into an accident, and there are no accidents on the bridge, and the plane doesn't leave early, and no meteorite hits the car, and."

Example 2: A_t: Leaving t minutes before the flight will get me to the airport.
Problems:

1. Partial observability (road state, other drivers' plans, etc.)
2. Noisy sensors (radio traffic reports)
3. Uncertainty in action outcomes (flat tire, etc.)
4. Immense complexity of modelling and predicting traffic

Q.2 (a) Define the following terms: (i) Unconditional or prior probability (2) (ii) Conditional probability (2) (iii) Independence (2)

Ans. (i) Unconditional or prior probability - Probabilities such as P(Total = 11) and P(doubles) are called unconditional or prior probabilities (and sometimes just "priors" for short); they refer to degrees of belief in propositions in the absence of any other information. Most of the time, however, we have some EVIDENCE information, usually called evidence, that has already been revealed. For example, the first die may already be showing a 5 and we are waiting with bated breath for the other one to stop spinning. In that case, we are interested not in the unconditional probability of rolling doubles.

(ii) Conditional probability – but in the conditional or posterior probability (or just "posterior" for short) of rolling doubles given that the first die is a 5. This probability is written P(doubles | Die1 = 5), where the " | " is pronounced "given." Similarly, if I am going to the dentist for a regular checkup, the probability P(cavity)=0.2 might be of interest; but if I go to the dentist because I have a toothache, it's P(cavity |toothache)=0.6 that matters. Note that the precedence of " | " is such that any expression of the form P(... | ...) always means P((...)|(...)).
(iii) Independence – Let us expand the full joint distribution by adding a fourth variable, Weather. The full joint distribution then becomes P(Toothache, Catch, Cavity, Weather), which has $2 \times 2 \times 2 \times 4 = 32$ entries. It contains four "editions", one for each kind of weather. For example, how are P(toothache, catch, cavity, cloudy) and P(toothache, catch, cavity) related? We can use the product rule: P(toothache, catch, cavity, cloudy) = P(cloudy |toothache, catch, cavity)P(toothache, catch, cavity) . Now, unless one is in the deity business, one should not imagine that one's dental problems influence the weather. And for indoor dentistry, at least, it seems safe to say that the weather does not influence the dental variables. Therefore, the following assertion seems reasonable: P(cloudy |toothache, catch, cavity) = P(cloudy). From this, we can deduce P(toothache, catch, cavity, cloudy) = P(cloudy)P(toothache, catch, cavity). A similar equation exists for every entry in P(Toothache, Catch, Cavity, Weather). In fact, we can write the general equation P(Toothache, Catch, Cavity, Weather) = P(Toothache, Catch, Cavity)P(Weather). Thus, the 32-element table for four variables can be constructed from one 8-element table and one 4-element table. The property is called independence (also marginal independence and absolute independence). In particular, the weather is independent of one's dental problems. Independence between propositions a and b can be written as $P(a \mid b) = P(a)$ or $P(b \mid a) = P(b)$ or $P(a \wedge b) = P(a)P(b)$

(b) Explain the Bayers' theorem (4).
Ans. we defined the product rule. It can actually be written in two forms:

$$P(a \wedge b) = P(a \mid b)P(b) \text{ and } P(a \wedge b) = P(b \mid a)P(a)$$

Equating the two right-hand sides and dividing by P(a), we get

$$P(b \mid a) = P(a \mid b)P(b)\ P(a)$$

This equation is known as Bayes' rule (also Bayes' law or Bayes' theorem). This simple equation underlies most modern AI systems for probabilistic inference. Quantifying Uncertainty The more general case of Bayes' rule for multivalued variables can be written in the P notation as follows: $P(Y \mid X) = P(X \mid Y)P(Y)\ P(X)$, As before, this is to be taken as representing a set of equations, each dealing with specific values of the variables. We will also have occasion to use a more general version conditionalized on some background evidence e: $P(Y \mid X, e) = P(X \mid Y, e)P(Y \mid e)\ P(X \mid e)$

Q.3 Define Forward chaining (2) and Backward chaining (2). Make a comparison table for forward chaining and backward chaining (6).
Ans. Forward Chaining – Forward chaining begins with atomic sentences in the knowledge base and proceeds to apply inference rules to derive new information until an endpoint or goal is achieved.
-Begin with facts that are known.
-Proceed to trigger all the inference rules whose premises are satisfied

- and then add the new data derived from them to the known facts,
- repeating the process till the goal is achieved or the problem is solved.

A forward-chaining algorithm for propositional definite clauses. The idea is simple: start with the atomic sentences in the knowledge base and apply Modus Ponens in the forward direction, adding new atomic sentences, until no further inferences can be made. Here, we explain how the algorithm is applied to first-order definite clauses. Definite clauses such as Situation ⇒ Response are especially useful for systems that make inferences in response to newly arrived information. Many systems can be defined this way, and forward chaining can be implemented very efficiently.

Backward Chaining - These algorithms work backward from the goal, chaining (linking and traversing) through rules while fining pre-established facts that support the proof. Ask whether the KB contains the clause of the farm lhs= rhs, where, lhs is s a list of conjuncts.
Example: American(west) is a clause with lhs is the empty list. Now, prove a query that contains variables with the substitution. Is a kind of or/and search- conjuncts and disjuncts
Backward chaining is a kind of AND/OR search—the OR part because the goal query can be proved by any rule in the knowledge base, and the AND part because all the conjuncts in the lhs of a clause must be proved. FOL-BC-OR works by fetching all clauses that might unify with the goal, standardizing the variables in the clause to be brand-new variables, and then, if the rhs of the clause does indeed unify with the goal, proving every conjunct in the lhs, using FOL-BC-AND. That function in turn works by proving each of the conjuncts in turn, keeping track of the accumulated substitution as we go.

1.	When based on available data a decision is taken then the process is called as Forward chaining.	Backward chaining starts from the goal and works backward to determine what facts must be asserted so that the goal can be achieved.
2.	Forward chaining is known as data-driven technique because we reaches to the goal using the available data.	Backward chaining is known as goal-driven technique because we start from the goal and reaches the initial state in order to extract the facts.
3.	It is a bottom-up approach.	It is a top-down approach.
4.	It applies the Breadth-First Strategy.	It applies the Depth-First Strategy.
5.	Its goal is to get the conclusion.	Its goal is to get the possible facts or the required data.
6.	Slow as it has to use all the rules.	Fast as it has to use only a few rules.

7.	It operates in forward direction i.e it works from initial state to final decision.	It operates in backward direction i.e it works from goal to reach initial state.
8.	Forward chaining is used for the planning, monitoring, control, and interpretation application.	It is used in automated inference engines, theorem proofs, proof assistants and other artificial intelligence applications.

Q.4 What is Joint Probability Distribution (3)? What is the probability of a *fracture* given the arm pain (3)? What is the probability of a *fracture* given the X-ray *detect*s (3)?

	Arm Pain		*~ Arm Pain*	
	detects	*~detects*	*detects*	*~detects*
Fracture	**.019**	**.101**	**.007**	**.052**
~Non-fracture	**.012**	**.086**	**.721**	**.103**

Ans. Joint Probability Distribution - In addition to distributions on single variables, we need notation for distributions on multiple variables. Commas are used for this. For example, P(Weather , Cavity) denotes the probabilities of all combinations of the values of Weather and Cavity. This is a 4 × 2 table of probabilities called the joint probability distribution of Weather and Cavity. We can also mix variables with and without values; P(sunny, Cavity) would be a two-element vector giving the probabilities of a sunny day with a cavity and a sunny day with no cavity. The P notation makes certain expressions much more concise than they might otherwise be.
For example, the product rules for all possible values of Weather and Cavity can be written as a single equation: P(Weather , Cavity) = P(Weather | Cavity)P(Cavity)
Quantifying Uncertainty instead of as these 4 × 2=8 equations (using abbreviations W and C):

$$P(W = sunny \land C = true) = P(W = sunny|C = true)\ P(C = true)$$
$$P(W = rain \land C = true) = P(W = rain|C = true)\ P(C = true)$$
$$P(W = cloudy \land C = true) = P(W = cloudy|C = true)\ P(C = true)$$
$$P(W = snow \land C = true) = P(W = snow|C = true)\ P(C = true)$$
$$P(W = sunny \land C = false) = P(W = sunny|C = false)\ P(C = false)$$
$$P(W = rain \land C = false) = P(W = rain|C = false)\ P(C = false)$$
$$P(W = cloudy \land C = false) = P(W = cloudy|C = false)\ P(C = false)$$
$$P(W = snow \land C = false) = P(W = snow|C = false)\ P(C = false)$$

	Arm Pain		*~ Arm Pain*	
	detects	*~detects*	*detects*	*~detects*
Fracture	**.019**	**.101**	**.007**	**.052**
~Non-fracture	**.012**	**.086**	**.721**	**.103**

P(*Arm Pain*) = *P*(*Arm Pain, detects, Fracture*) + *P(Arm Pain, ¬detects, Fracture*)+ *P*(*Arm Pain, detects, ¬Fracture*) + *P*(*Arm Pain, ¬detects, ¬ Fracture*)
= 0.019 + 0.101 + 0.012+ 0.086
=0.218

P(*Fracture* ∨ *Arm Pain*)
= 0.019 + 0.101 + 0.007 + 0.052 + 0.012+ 0.086
= 0.277

Q.5 What is normalization (3)? Calculate the normalization constant '*α*' for the evidence variable "toothache" (7).

	toothache		*~toothache*	
	catch	*~catch*	*catch*	*~catch*
cavity	**.108**	**.012**	**.072**	**.008**
~cavity	**.016**	**.064**	**.144**	**.576**

Ans. Normalization - In probability theory, a normalizing constant is a constant by which an everywhere non-negative function must be multiplied so the area under its graph is 1, e.g., to make it a probability density function or a probability mass function.

P(cavity | toothache) = P(cavity ∧ toothache) P(toothache)
= 0.108 + 0.012 0.108 + 0.012 + 0.016 + 0.064
= 0.6
Just to check, we can also compute the probability that there is no cavity, given a toothache:
P(¬cavity |toothache) = P(¬cavity ∧ toothache) P(toothache)
= 0.016 + 0.064 0.108 + 0.012 + 0.016 + 0.064
= 0.4
The two values sum to 1.0, as they should. Notice that in these two calculations, the term 1/P(toothache) remains constant, no matter which value of Cavity we calculate. In fact, it can be viewed as a normalization constant for the distribution P(Cavity |toothache), ensuring that it adds up to 1. Throughout the chapters dealing with probability, we use α to denote such constants. With this notation, we can write the two preceding equations in one:
P(Cavity|toothache) = α P(Cavity, toothache) = α [P(Cavity, toothache, catch) + P(Cavity, toothache,¬catch)]
= α [<0.108, 0.016> + <0.012, 0.064>]
= α <0.12, 0.08>
= <0.6, 0.4>
Normalization Constant, α = 1/(0.12+0.08)

Q.6 Describe an Expert system (6). Discuss how to represent the Domain Knowledge (4).

Ans. Artificial Intelligence is a piece of software that simulates the behaviour of a human that has experts in a particular domain is known as an expert system. It does this by acquiring relevant knowledge from its knowledge base and interpreting it according to the user's problem. The data in the knowledge base is added by humans that are expert in a particular domain and this software is used by a non-expert user to acquire some information. It is widely used in many areas such as medical diagnosis, accounting, coding, games etc. Knowledge Engineering is the term used to define the process of building an Expert System and its practitioners are called Knowledge Engineers. The primary role of a knowledge engineer is to make sure that the computer possesses all the knowledge required to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as a symbolic pattern in the memory of the computer.

Representing Domain Knowledge-

The knowledge base in an expert system is a critical component where domain-specific information is stored. This knowledge can be represented in various ways, each suited to different types of problems and reasoning methods.

1. Production Rules:
   Format: If-Then rules.
   Example: If a patient has a fever and a cough, then the patient might have the flu
2. Semantic Networks:
   Format: Graph structures with nodes representing concepts and edges representing relationships.
   Example: A network where "Dog" is connected to "Animal" with an "is a" relationship
3. Ontologies:
   Format: Formal representation of knowledge with defined concepts, relationships, and rules.
   Example: An ontology for medical diagnosis defining symptoms, diseases, and their interrelations.
4. Decision Trees:
   Format: Tree structures where each node represents a decision point and branches represent possible outcomes.
   Example: A decision tree for diagnosing a disease based on symptoms.
5. Logic-Based Representations:
   Format: Formal logic expressions, such as propositional and predicate logic.
   Example: All humans are mortal. Socrates is a human. Therefore, Socrates is mortal.