AI ENDSEM UNIT – 5 PYQ

- **MAY / JUN 2022**

Q5)

a) Explain Forward and Backward chaining. What factors justify whether reasoning is to be done in forward or backward chaining. [9]

1. **Forward Chaining:**

   Forward chaining is a **data-driven reasoning method** that starts with known facts and applies inference rules to extract more facts until a goal is reached.

- **How it works:**
  - Begins with initial facts.
  - Applies rules whose **premises match the facts**.
  - Adds the **conclusion** of those rules as new facts.
  - Continues until the **goal** is derived.
  - **Example:**

```vbnet
Rule: If A and B, then C
Facts: A is true, B is true
⇒ Derive: C is true
```

Used in:
Expert systems, data monitoring systems, rule-based systems.

2. Backward Chaining:

- **Definition:**
  Backward chaining is a **goal-driven reasoning method** that starts with a goal and works backward to determine whether the known facts support the goal.
- **How it works:**
  - Begins with a **goal**.
  - Looks for rules that conclude the goal.
  - Checks if the **premises** of those rules are true.
  - If not, these premises become **sub-goals** and are verified recursively.

- **Example:**

```vbnet
Goal: C is true
Rule: If A and B, then C
Check: Are A and B true?
If yes ⇒ Goal is satisfied.
```

- **Used in:**
  Theorem provers, **AI planning systems**, Prolog interpreters.

FACTOR	FORWARD CHAINING	BACKWARD CHAINING
GOAL KNOWN?	Not known; system discovers all outcomes.	Goal is clearly specified.
DATA SIZE	Large dataset, few queries.	Small data, many specific goals.
INFERENCE DEPTH	Good for shallow, broad inference.	Good for deep, focused inference.
APPLICATION TYPE	Monitoring systems, diagnosis.	Problem-solving, question answering.

b) What are the reasoning patterns in propositional logic? Explain them in detail.[9]

Propositional logic uses **logical connectives** and **inference rules** to derive conclusions from known facts. The key reasoning patterns are also called **rules of inference**

✅ 1. Modus Ponens (Implication Elimination):

- **Form:**

```csharp
If P → Q
P is true
⇒ Q is true
```

- **Example:**
  If it rains, the ground gets wet.
  It rains.
  ⇒ The ground gets wet.

✅ 2. Modus Tollens:

- **Form:**

```csharp
If P → Q
¬Q is true
⇒ ¬P is true
```

- **Example:**
  If it is a dog, it barks.
  It does not bark.
  ⇒ It is not a dog.

✅ 3. Hypothetical Syllogism (Transitivity):

- **Form:**

```css
If P → Q
If Q → R
⇒ P → R
```

- **Example:**
  If I study, I will pass.
  If I pass, I will be happy.
  ⇒ If I study, I will be happy.

✅ 4. Disjunctive Syllogism:

- **Form:**

```csharp
P ∨ Q
¬P is true
⇒ Q is true
```

- **Example:**
  It is either a cat or a dog.
  It is not a cat.
  ⇒ It is a dog.

✅ 5. Conjunction Introduction:

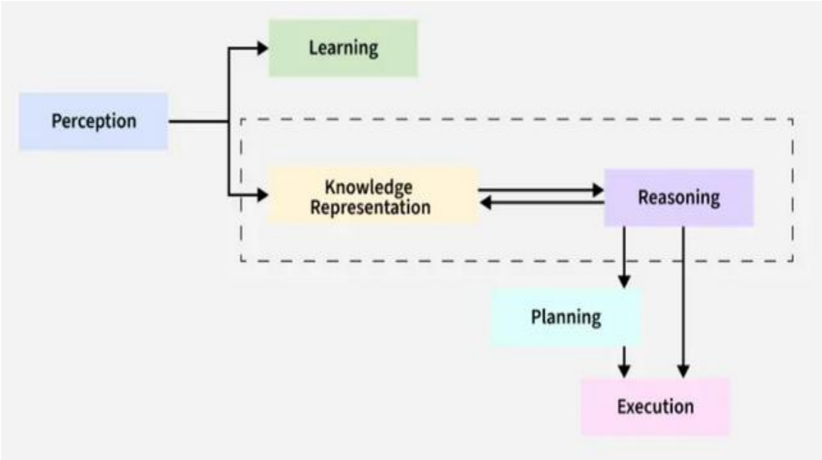
- **Form:**

```csharp
P is true
Q is true
⇒ P ∧ Q
```

- **Example:**
  It is raining.
  It is cold.
  ⇒ It is raining and cold.

Q6

b) Explain knowledge representation structures and compare them. [7]



change diagram !

Knowledge Representation is the method used to **encode real-world information** into a format that a computer system (especially AI) can use to reason, infer, and solve problems.

It provides the **structure and logic** for storing and processing knowledge.

Structure	Description	Example
Semantic Networks	Graph-based structure using **nodes (objects)** and **edges (relations)**	IsA(Dog, Animal)
Frames	Data structure for representing **objects** and their **attributes** (slots)	Car: [color=red, wheels=4]
Production Rules	Knowledge in the form of **IF-THEN** rules	IF hungry THEN eat
Logic (Propositional & First-Order)	Uses logical statements and inference rules	$\forall x$ Dog(x) → Animal(x)
Ontologies	Hierarchical structure defining **concepts and relationships**	Fruit $\subset$ Food

COMPARSION :

Feature	Semantic Network	Frames	Production Rules	Logic	Ontologies
Visual	Yes (Graph)	No	No	No	Yes (Hierarchical)
Structured Data	Moderate	High	Low	Logical only	High
Inference	Easy with	Slot	Rule-based	Strong	Reasoning via

	links	inheritance		(proofs)	relations
Use Case	Taxonomy	Object properties	Expert systems	Theorem proving	Semantic web, AI knowledge bases

c) What do you mean by Ontology of Situation Calculus? [3]

The **Ontology of Situation Calculus** refers to the **set of basic concepts** and **formal structure** used to describe **dynamic worlds** in terms of **actions, situations, and objects**.

Core Elements:

1. **Situations:** Represent the **state of the world** at a point in time.
2. **Actions:** Events that cause the transition from one situation to another.
3. **Fluents:** Properties that can **change** across situations (e.g., location, status).
4. **Result Function:** Result(a, s) gives the new situation after performing action a in situation s.

Purpose:

- It provides a **logical framework** for representing and reasoning about **actions and their effects** over time in AI systems.

- **MAY / JUN 2023**

Q5)

a) Explain Forward Chaining and Backward Chaining. With its Properties, with one. example. [9]

1. Forward Chaining:

Definition:

- Forward chaining is a data-driven reasoning method.
- It starts from known facts and applies inference rules to derive new facts until the goal is reached.

Properties:

- Approach: Bottom-up (from data to goal)

- Strategy: Breadth-first or depth-first
- Best used when: All facts are known, goal is not fixed
- Rule used: Modus Ponens
- Common in: Expert systems (e.g., CLIPS)

📌 **Example:**

```vbnet
Rule: If A and B, then C
Facts: A is true, B is true
→ C is inferred
```

2. Backward Chaining:

Definition:

- **Backward chaining** is a **goal-driven** inference method.
- It starts with a **goal** and works **backward** to check if known facts can support it.

Properties:

- **Approach:** Top-down (from goal to data)
- **Strategy:** Depth-first search
- **Best used when:** Goal is clearly defined
- **Rule used:** Modus Ponens
- **Common in:** Prolog, theorem provers

📌 **Example:**

```pgsql
Goal: C is true
Rule: If A and B, then C
→ Check if A and B are true
If both are true → C is confirmed
```

Q 5

b) remaining !!

Q6) a) Explain FOL inference for the following quantifiers:

i) Universal Generalization
ii) Universal Instantiation
iii) Existential Instantiation
iv) Existential Introduction
[8 Marks]

i) Universal Generalization (UG):

- Definition:
  If a sentence is true for any arbitrary object, then it is true for all objects.
- Notation:
  From P(x) → infer ∀x P(x)
- Example:
  If x is arbitrary and x > 0, then we can infer ∀x (x > 0)
- Usage:
  Used carefully—only when x is not bound to any specific constant.

ii) Universal Instantiation (UI):

- Definition:
  From a universally quantified statement, we can infer that the statement is true for any specific constant.
- Notation:
  From ∀x P(x) → infer P(c) for any constant c
- Example:
  Given ∀x Human(x) → Mortal(x)
  → we can infer Human(Socrates) → Mortal(Socrates)

iii) Existential Instantiation (EI):

- Definition:
  From an existentially quantified sentence, we can infer the statement is true for some unknown constant.
- Notation:
  From ∃x P(x) → infer P(c) where c is a new constant
- Example:
  If ∃x Owns(Nono, x), then introduce a new constant like M1
  → Owns(Nono, M1)

- Use Case:
  Helps convert FOL statements to propositional logic for inference.

iv) Existential Introduction (EI or ∃-Introduction):

- Definition:
  If a statement is true for a particular instance, then it must be true for some object in the domain.
- Notation:
  From P(c) → infer ∃x P(x)
- Example:
  If Loves(John, Mary) is true
  → we can infer ∃x Loves(John, x)

Q6)

b) What is Ontological Engineering? Explain in detail with its categories, objects, and model. [9]

1. Ontological Engineering – Definition:

Ontological Engineering is the branch of AI concerned with **creating structured representations** of knowledge using **concepts, categories, objects**, and their **relationships**.

It helps machines **understand, organize, and reason** about real-world domains.

2. Categories and Objects:

- **Categories:** General groupings of similar entities.
  Example: *Fruit*, *Vehicles*, *Animals*
- **Objects:** Specific instances that belong to categories.
  Example: *Apple* ∈ *Fruit*, *Car* ∈ *Vehicles*
- **Inheritance:** Objects inherit properties from categories.
  Example: If *Fruit* is edible, then *Apple* is edible.

2. **Ontological Model:**

Ontologies define a hierarchy of concepts, their relationships, and attributes.

An ontology model typically includes:

- **Classes (Categories):** General types (e.g., Food)
- **Instances (Objects):** Specific examples (e.g., Banana)
- **Relationships:** How entities relate (e.g., PartOf, SubclassOf)

- **Attributes:** Characteristics (e.g., color, weight)

Example:

```cpp
Fruit ⊂ Food
Apple ∈ Fruit
Color(Apple) = Red
```

This means Apple inherits the property of being food, and has an attribute color = Red.

4. Applications:

Semantic Web: Helps websites understand and link data meaningfully

Expert Systems: Represent domain-specific expert knowledge

Natural Language Understanding: Helps in context understanding

Knowledge Sharing: Enables interoperability between AI agents

---

- **MAY / JUN 2024**

Q5)

a) **Explain Forward Chaining and Backward Chaining. With its properties, advantages and disadvantages.**

FORWARD CHAINING :

Advantages:

- Efficient when goal is not clearly known
- Automatically derives all possible outcomes

Disadvantages:

- Can generate irrelevant facts
- May require more memory and time

BACKWARD CHAINING :

Advantages:

- Focused reasoning — checks only relevant facts
- Efficient for specific goal queries

Disadvantages:

- May fail if subgoals are hard to prove
- Less efficient for broad discovery

b) Explain: [8]

i) Unification in FOL
ii) Reasoning with Default Information

i) Unification in First-Order Logic (FOL):

Definition:

Unification is the process of **finding a substitution** that makes two FOL expressions **identical**.

Purpose:

Used in inference methods (e.g., resolution, forward/backward chaining) to match predicates and apply rules.

📌 **Example:**

```css
Unify: Knows(John, x) and Knows(John, Mary)
Substitution: θ = {x / Mary}
```

📌 **Key Concept:**

If `P(x)` and `P(John)` → unify by replacing `x` with `John`.

ii) Reasoning with Default Information:

[4]

Default reasoning is used when **complete information is not available**. It allows the system to make **plausible assumptions** until contradicted.

Type:

It is a form of **non-monotonic reasoning** (new facts can invalidate previous conclusions).

Example:

- Default Rule: "Birds typically fly."
- Known: Bird(Tweety)

- Inferred: Flies(Tweety) (by default)
- If later Penguin(Tweety) is known → retract Flies(Tweety)

Use Case:

Useful in real-world AI systems like **diagnosis, natural language understanding**, where **exceptions** are common.

Q 6

a) , b) already done !

➢ **NOV / DEC 2022**

Q5)

a) Explain Forward chaining algorithm with the help of example. [9]

Forward chaining is a **data-driven** inference technique where reasoning starts from known **facts** and applies rules to infer new facts until the **goal** is reached or no more rules apply.

It uses the **Modus Ponens** inference rule to trigger applicable rules.
This method is most effective when all or most data is available at the beginning, and the goal is not clearly defined.

2. Algorithm Steps:

1. Begin with initial known facts.
2. Identify rules whose premises match the current facts.
3. Apply those rules to derive new facts (conclusions).
4. Add the newly derived facts to the set of known facts.
5. Repeat the process using the updated facts.
6. Continue until the desired goal is derived or no further rules apply.

3. Example:

Rules:

mathematica Copy Edit

```
R1: If A and B → C
R2: If C and D → E
R3: If E → Goal
```

Facts:

css Copy Edit

```
A, B, D
```

Execution:

- A and B satisfy R1 ⇒ C
- C and D satisfy R2 ⇒ E
- E satisfies R3 ⇒ Goal is derived

4. Characteristics:

- **Approach:** Bottom-up
- **Inference Style:** Breadth-first or depth-first
- **Use Case:** Expert systems, control systems, production systems

b) Write and explain the steps of Knowledge Engineering Process.

[9]

Knowledge Engineering Process – Definition:

Knowledge Engineering is the process of **building knowledge-based systems** by identifying, organizing, and encoding expert knowledge into a form that an AI system can use.

Steps of the Knowledge Engineering Process:

1. Identify the Task

- Define the **problem** to be solved.
- Example: Medical diagnosis, product recommendation.

2. Assemble the Relevant Knowledge

- Collect data from **experts, books, databases**, etc.
- Organize it logically.

3. Decide on a Vocabulary

- Define a set of **terms, predicates, objects**, and relationships used in the domain.

4. Encode the Knowledge

- Represent the knowledge using **rules**, **frames**, or **logic** formats.

5. Test the Knowledge Base

- Run sample queries or test cases to check if the system behaves correctly.

6. Debug and Refine

- Fix errors, improve rule coverage, and enhance reasoning ability.

7. Deploy the System

- Integrate the knowledge base into the final application for end users.

Q6)

a) Explain Backward chaining algorithm with the help of example [9]

1. Definition:

Backward chaining is a **goal-driven inference method** where reasoning starts from the **goal** and works backward to determine if known **facts support** it.

2.Backward Chaining Algorithm:

1. Start with the goal to be proved and a knowledge base (KB) of rules and facts.
2. Check if the goal is already a known fact in the KB.
    - If yes, return success.

3. Find all rules where the goal appears in the conclusion (THEN part).
4. For each rule found, create sub-goals from its premises (IF part).
5. Recursively attempt to prove each sub-goal using the same backward chaining method.
6. If all sub-goals are proved true, then the original goal is proved.
7. If no rules apply or sub-goals fail, the goal cannot be proved.

3. Example:

Rules:

```mathematica
R1: If A and B → C
R2: If C and D → E
R3: If E → Goal
```

Facts:

```css
A, B, D
```

Goal: Prove Goal

Execution:

- Goal requires E (from R3)
- E requires C and D (from R2)
- D is known
- C requires A and B (from R1) → both are known
  → Therefore, C, E, and Goal are all proved.

b) Explain i) resolution ii) ubification

i) Resolution

- **Definition:**
  Resolution is a rule of inference used in propositional and first-order logic to derive new clauses by eliminating complementary literals.

- **Purpose:**
  It is mainly used in automated theorem proving to show that a set of statements is inconsistent (i.e., to derive a contradiction).

- **Working:**
  If two clauses contain a literal and its negation, they can be combined into a new clause by removing the resolved literals.

- **Example:**
  Clause 1: $P \lor Q$
  Clause 2: $\neg P \lor R$
  Resolution Result: $Q \lor R$

- **Application:**
  Used in logic programming, SAT solvers, and FOL theorem provers.

ii) Unification

- **Definition:**
  Unification is the process of finding a substitution that makes two logical expressions identical.
- **Purpose:**
  It is used to match predicates in inference rules such as resolution, forward chaining, and backward chaining.
- **Working:**
  Variables are replaced with constants or other variables to make two expressions equal.
- **Example:**
  Unify: Knows(John, x) and Knows(John, Mary)
  Substitution: {x / Mary}
- **Application:**
  Core to logic-based AI systems, especially in Prolog and FOL resolution.

➢ **NOV / DEC 2024**

Q5

a) Remaining

b) Done

Q6

a) What do you understand by Resolution? Describe the procedure to convert into CNF. Find the clause of the expression: (¬P ∨ Q) → R [9]

1. Resolution – Definition:

Resolution is a **rule of inference** used to deduce new clauses by eliminating **complementary literals** from two clauses. It is widely used in **automated theorem proving**.

2. CNF – Conjunctive Normal Form:

A logical expression is in CNF if it is a **conjunction of one or more disjunctions** of literals.
Form: (A ∨ B) ∧ (¬C ∨ D) ∧ ...

3. Procedure to Convert to CNF:

1. **Eliminate implications**:
   A → B becomes ¬A ∨ B
2. **Move NOT inwards**:
   Use De Morgan's Laws
   ¬(A ∨ B) → (¬A ∧ ¬B)
3. **Standardize variables** (if in FOL)
4. **Distribute ∨ over ∧**
   To achieve proper conjunctive form

4. Convert the given expression: (¬P ∨ Q) → R

Step 1: Eliminate implication:
¬(¬P ∨ Q) ∨ R
Step 2: Apply De Morgan's law:
(¬¬P ∧ ¬Q) ∨ R
(P ∧ ¬Q) ∨ R
Step 3: Distribute ∨ over ∧:
= (P ∨ R) ∧ (¬Q ∨ R)

5. Final CNF Clauses:

```less
Clause 1: P ∨ R
Clause 2: ¬Q ∨ R
```

Q6

b) What are the reasoning systems for Categories? Explain Semantic Network and Description Logic.

1. Reasoning Systems for Categories:

These are systems that allow AI to reason about categories, subcategories, and relationships among entities in a structured manner.

2. Semantic Network:

A graph-based structure representing objects, categories, and relationships.

Nodes represent concepts; edges represent relationships.

Example:

```nginx
Dog → IsA → Animal
Dog → Has → Tail
```

Advantages:

- Intuitive visual structure
- Supports inheritance (e.g., if Animal is alive, Dog is alive)

3. Description Logic:

- A **formal logic-based system** for defining and reasoning about **categories and their relationships**.
- Supports reasoning like:
  - Is A a subclass of B?
  - Does object X belong to class C?

Example:

```nginx
Human ⊑ Mammal
Doctor ⊑ Human
⇒ Doctor ⊑ Mammal
```

Used in: Ontologies (e.g., OWL in Semantic Web)