

T.E. (Computer Engineering)
ARTIFICIAL INTELLIGENCE
(2019 Pattern) (Semester- II) (310253)

Time : 2½ Hours]

[Max. Marks : 70

Instructions to the candidates:

- 1) Answer four questions Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.
- 2) Neat diagrams must be drawn wherever necessary.
- 3) Assume Suitable data if necessary.

- Q1)** a) List all problem solving strategies. What is backtracking, explain with n queen problem. [8]
- b) Write Minimax Search Algorithm for two players. How use of alpha and beta cut-offs will improve performance? [9]

OR

- Q2)** a) Define Game theory, Differentiate between stochastic and partial games with examples. [9]
- b) Define is Constraint satisfaction problem, state the types of consistencies solve the following Crypt Arithmetic Problem. [8]

$$\begin{array}{r} \text{B A S E} \\ + \text{B A L L} \\ \hline \text{G A M E S} \end{array}$$

- Q3)** a) What is an Agent? Name any 5 agents around you explain knowledge based agent with Wumpus World.
List and explain in short the various steps of knowledge engineering process
Consider the following axioms:
If a triangle is equilateral then it is isosceles [9]
- b) If a triangle is isosceles, then its two sides AB and AC are equal.
If AB and AC are equal, then angle B and C are equal.
ABC is an equilateral triangle.
Represent these facts in predicate logic. [9]

OR

P.T.O.



- Q4) a)** Write the following sentences in FOL (using types of quantifiers) [9]
- i) All birds fly
 - ii) Some boys play cricket
 - iii) A first cousin is a child of a parent's sibling
 - iv) You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all the time.
- b)** What is Knowledge Representation using propositional logic? Compare propositional and predicate Logic. [9]
- Q5) a)** Explain Forward Chaining and Backward Chaining. With its properties, advantages and disadvantages. [9]
- b)** Explain: [8]
- i) Unification in FOL
 - ii) Reasoning with Default information
- OR
- Q6) a)** Explain FOL inference for following Quantifiers. [8]
- i) Universal Generalization
 - ii) Universal Instantiation
 - iii) Existential Instantiation
 - iv) Existential introduction
- b)** What is Ontological Engineering, in details with its categories object and Model. [9]
- Q7) a)** Explain with an example Goal Stack Planning (STRIPS algorithm). [5]
- b)** Explain with example, how planning is different from problem solving. [5]
- c)** Explain AI components and AI architecture [8]
- OR
- Q8) a)** Explain Planning in non deterministic domain. [5]
- b)** Explain. [5]
- i) Importance of planning
 - ii) Algorithm for classical planning
- c)** What is AI explain scope of AI in all walks of Life also explain future opportunities with AI. [8]



Q1) a) List all problem-solving strategies. What is backtracking, explain with n queen problem.

Ans:

Problem-Solving Strategies

Problem-solving involves various techniques to **identify, analyze, and resolve challenges efficiently**.

1. **Trial and Error** – Testing multiple solutions until one works.
2. **Divide and Conquer** – Breaking a problem into smaller subproblems.
3. **Brainstorming** – Generating multiple ideas before selecting the best one.
4. **Algorithmic Approach** – Using predefined steps to solve structured problems.
5. **Heuristic Methods** – Applying experience-based techniques for quick solutions.
6. **Analogy-Based Reasoning** – Solving problems by comparing them to similar past issues.
7. **Root Cause Analysis** – Identifying the fundamental cause of a problem before solving it.
8. **Constraint Satisfaction** – Finding solutions within given limitations.
9. **Optimization Techniques** – Improving efficiency using methods like **Simulated Annealing** and **Gradient Descent**.
10. **Backtracking** – Exploring possible solutions and discarding incorrect paths.

You can explore more details on **problem-solving techniques** [here](#) and [here](#).

Backtracking and N-Queen Problem

What is Backtracking?

Backtracking is a **recursive problem-solving technique** used to explore all possible solutions by **trying different choices and undoing incorrect ones**. It is commonly used in **constraint satisfaction problems**, such as:

- **Sudoku solving**
- **Graph coloring**
- **N-Queen problem**

N-Queen Problem Using Backtracking

The **N-Queen problem** involves placing **N queens on an N×N chessboard** such that **no two queens attack each other**.



SPPU-TE-COMP-CONTENT – KSKA Git

Steps to Solve Using Backtracking:

1. **Start from the first row** and place a queen in a column.
2. **Move to the next row** and try placing another queen in a safe position.
3. **Check for conflicts** (same column, same diagonal).
4. **If a conflict occurs**, backtrack and try a different column.
5. **Repeat until all queens are placed** or backtrack if no solution exists.

Example for $N = 4$ (4×4 Chessboard):

Possible solutions:

- [2, 4, 1, 3]
- [3, 1, 4, 2]



Q1) b) Write Minimax Search Algorithm for two players. How use of alpha and beta cut-offs will improve performance?

Ans:

Minimax Search Algorithm for Two Players

The **Minimax algorithm** is a decision-making strategy used in **two-player zero-sum games** like **chess, tic-tac-toe, and checkers**. It helps a player choose the best possible move by assuming that the opponent also plays optimally.

1. Minimax Algorithm Steps

1. **Generate the Game Tree** – Create a tree representing all possible moves.
2. **Assign Values to Terminal Nodes** – Evaluate the final game states using a heuristic function.
3. **Backpropagate Values** –
 - The **maximizing player** selects the highest value.
 - The **minimizing player** selects the lowest value.
4. **Choose the Best Move** – The root node selects the optimal move based on minimax values.

2. Minimax Algorithm (Pseudocode)

python

```
def minimax(depth, nodeIndex, isMax, scores, alpha, beta):
    if depth == terminal_depth:
        return scores[nodeIndex]

    if isMax:
        best = float('-inf')
        for i in range(num_children):
            val = minimax(depth + 1, nodeIndex * 2 + i,
False, scores, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break # Alpha-Beta Pruning
        return best
    else:
        best = float('inf')
        for i in range(num_children):
```



```
        val = minimax(depth + 1, nodeIndex * 2 + i,  
True, scores, alpha, beta)  
        best = min(best, val)  
        beta = min(beta, best)  
        if beta <= alpha:  
            break # Alpha-Beta Pruning  
    return best
```

3. Alpha-Beta Cutoffs and Performance Improvement

Alpha-Beta Pruning is an optimization technique that improves the efficiency of the **Minimax algorithm** by eliminating unnecessary branches in the game tree.

How Alpha-Beta Pruning Works?

- **Alpha (α)** – The best value the **maximizing player** can guarantee.
- **Beta (β)** – The best value the **minimizing player** can guarantee.
- If $\beta \leq \alpha$, further exploration is **pruned**, reducing computation time.

Performance Benefits

- **Reduces Search Space** – Eliminates unnecessary evaluations.
- **Speeds Up Decision-Making** – Allows deeper searches in complex games.
- **Maintains Optimality** – Does not affect the final decision but improves efficiency.

SPPU-TE-COMP-CONTENT – KSKA Git

Q2) a) Define Game theory, Differentiate between stochastic and partial games with examples.

Ans:

Game Theory Definition

Game theory is a branch of **mathematics and economics** that analyzes **strategic interactions** between rational decision-makers. It studies how individuals or entities make decisions when their outcomes depend on the actions of others

Aspect	Stochastic Games	Partial Information Games
Definition	Games with random events affecting outcomes.	Games where players have incomplete knowledge of the game state.
Nature of Uncertainty	Involves probabilistic elements (e.g., dice rolls, random events).	Players lack full visibility into opponent moves or key game data.
Decision-Making Approach	Requires probabilistic reasoning and adaptive strategies .	Depends on inference, estimation, and learning from hidden information .
Examples	Backgammon, Monopoly, Poker with random draws.	Poker, Battleship, Strategy-based card games.
AI Applications	Used in reinforcement learning, robotic decision-making, and economic models.	Applied in cybersecurity, hidden Markov models, and strategic AI game-playing.
Challenges	Complexity in predicting random outcomes.	Difficulty in estimating missing information correctly.
Advantages	Encourages flexibility and adaptation in decision-making.	Encourages strategic planning and reasoning under uncertainty.



Q2) b) Define is Constraint satisfaction problem, state the types of consistencies solve the following Crypt Arithmetic Problem.

$$\begin{array}{r} \text{B A S E} \\ + \text{B A L L} \\ \hline \text{G A M E S} \end{array}$$

Ans:

Constraint Satisfaction Problem (CSP)

A **Constraint Satisfaction Problem (CSP)** is a mathematical problem where a set of variables must be assigned values while satisfying a set of constraints. CSPs are widely used in **artificial intelligence, scheduling, planning, and optimization problems**.

Key Components of CSP:

1. **Variables** – Elements that need values (e.g., colors in a map coloring problem).
2. **Domains** – Possible values for each variable (e.g., numbers in Sudoku).
3. **Constraints** – Rules that restrict variable assignments (e.g., "no two adjacent regions can have the same color").

Types of Consistencies in CSP

Consistency Type	Definition	Example
Node Consistency	Each variable satisfies its unary constraints.	A region in a map must be colored red or blue.
Arc Consistency	Every value of one variable has a valid corresponding value in another variable.	In Sudoku, a number in one cell must not repeat in the same row.
Path Consistency	Ensures consistency across multiple variables in a sequence.	In scheduling, if A precedes B and B precedes C, then A must precede C.
K-Consistency	A problem is k-consistent if any k-1 variables can be extended to k variables while satisfying constraints.	Ensuring a valid sequence of tasks in a workflow.

SPPU-TE-COMP-CONTENT – KSKA Git

$$\begin{array}{r} \text{B A S E} \\ + \text{B A L L} \\ \hline \text{G A M E S} \end{array}$$

Each letter represents a unique digit (0-9). The goal is to find a valid assignment such that the sum holds true.

Solution Approach:

1. **Identify Unique Letters:** {B, A, S, E, L, G, M}.
2. **Assign Digits (0-9) Without Repetition.**
3. **Ensure the Addition is Mathematically Correct.**
4. **Use Backtracking to Find Valid Assignments.**

Possible Solution:

Using computational methods, one valid assignment is:

- **B = 7, A = 4, S = 8, E = 3, L = 5, G = 1, M = 9**

Thus,

$$\begin{array}{r} 7483 \\ + 7455 \\ \hline 14938 \end{array}$$



Q3) a) What is an Agent? Name any 5 agents around you explain knowledge-based agent with Wumpus World.

List and explain in short, the various steps of knowledge engineering process

Consider the following axioms:

If a triangle is equilateral then it is isosceles.

Ans:

Definition of an Agent

An **agent** is an entity that perceives its environment through sensors and acts upon it using actuators to achieve specific goals. Agents can be **human, software-based, or robotic** and are widely used in **artificial intelligence (AI), automation, and decision-making systems**.

Five Examples of Agents Around Us

1. **Autonomous Vehicles** – Self-driving cars use sensors and AI to navigate roads.
2. **Chatbots & Virtual Assistants** – AI-powered assistants like **Copilot, Siri, and Alexa** respond to user queries.
3. **Robotic Vacuum Cleaners** – Devices like **Roomba** clean floors autonomously.
4. **Stock Trading Bots** – AI-driven systems analyze market trends and execute trades.
5. **Smart Home Systems** – Devices like **Nest Thermostat** adjust temperature based on user preferences.

Knowledge-Based Agent & Wumpus World

A **knowledge-based agent** is an AI system that **stores, processes, and applies knowledge** to make informed decisions. The **Wumpus World** is a classic AI problem that demonstrates how an agent navigates an environment using **logical reasoning and knowledge representation**.

Wumpus World Overview

- **Grid-Based Environment** – A 4×4 cave with hazards (pits, Wumpus monster) and a treasure (gold).
- **Agent's Goal** – Retrieve gold while avoiding dangers.
- **Sensory Inputs** –
 - **Breeze** → Indicates a nearby pit.
 - **Stench** → Suggests proximity to the Wumpus.
 - **Glitter** → Signals gold presence.
- **Actions** – Move, grab gold, shoot arrow, escape safely.

The agent uses **logical inference** to determine safe paths and avoid hazards, making decisions based on **stored knowledge and environmental feedback**.



SPPU-TE-COMP-CONTENT – KSKA Git

Steps in the Knowledge Engineering Process

Step	Description
1. Identify the Task	Define the problem the system will solve.
2. Gather Knowledge	Extract domain-specific information from experts or datasets.
3. Define Vocabulary	Establish predicates, functions, and constants for knowledge representation.
4. Encode General Knowledge	Write axioms and rules governing the domain.
5. Encode Specific Problem Instances	Represent real-world cases using logical statements.
6. Query the Knowledge Base	Use inference mechanisms to derive conclusions.
7. Debug & Optimize	Refine the system to improve accuracy and efficiency.

Axiom: If a Triangle is Equilateral, Then It is Isosceles

An **equilateral triangle** has **three equal sides**, while an **isosceles triangle** has **at least two equal sides**. Since an equilateral triangle meets the criteria for an isosceles triangle, the statement holds true.



Q3) b) If a triangle is isosceles, then its two sides AB and AC are equal.

If AB and AC are equal, then angle B and C are equal.

ABC is an equilateral triangle.

Represent these facts in predicate logic.

Ans:

Predicate Logic Representation of Given Facts

Predicates Used:

- $\text{Isosceles}(T) \rightarrow \text{Triangle } T \text{ is isosceles.}$
- $\text{EqualSides}(T, X, Y) \rightarrow \text{Sides } X \text{ and } Y \text{ of triangle } T \text{ are equal.}$
- $\text{EqualAngles}(T, A, B) \rightarrow \text{Angles } A \text{ and } B \text{ of triangle } T \text{ are equal.}$
- $\text{Equilateral}(T) \rightarrow \text{Triangle } T \text{ is equilateral.}$

Logical Representation:

1. If a triangle is isosceles, then its two sides AB and AC are equal:

$\forall T (\text{Isosceles}(T) \rightarrow \text{EqualSides}(T, AB, AC))$

If sides AB and AC are equal, then angles B and C are equal:

$\forall T (\text{EqualSides}(T, AB, AC) \rightarrow \text{EqualAngles}(T, B, C))$

ABC is an equilateral triangle:

$\text{Equilateral}(ABC)$

Since an **equilateral triangle** has all sides equal, we can also infer:

$\forall T (\text{Equilateral}(T) \rightarrow (\text{EqualSides}(T, AB, AC) \wedge \text{EqualSides}(T, BC, AB) \wedge \text{EqualSides}(T, AC, BC)))$

Q4) a) Write the following sentences in FOL (using types of quantifiers)

- i) All birds fly
- ii) Some boys play cricket
- iii) A first cousin is a child of a parent's sibling
- iv) You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all the time.

Ans:

Predicates Used:

- $\text{Bird}(x)$: x is a bird
- $\text{Flies}(x)$: x flies
- $\text{Boy}(x)$: x is a boy
- $\text{PlaysCricket}(x)$: x plays cricket
- $\text{IsChildOf}(x, y)$: x is a child of y
- $\text{IsParentOf}(x, y)$: x is a parent of y
- $\text{IsSiblingOf}(x, y)$: x is a sibling of y
- $\text{IsFirstCousinOf}(x, y)$: x is a first cousin of y
- $\text{Person}(x)$: x is a person
- $\text{Time}(t)$: t is a time
- $\text{Fool}(x, t)$: You can fool x at time t (or x is fooled at time t)

i) All birds fly

- **FOL:** $\forall x (\text{Bird}(x) \rightarrow \text{Flies}(x))$
- **Explanation:** For every x , if x is a bird, then x flies.

ii) Some boys play cricket

- **FOL:** $\exists x (\text{Boy}(x) \wedge \text{PlaysCricket}(x))$
- **Explanation:** There exists at least one x such that x is a boy AND x plays cricket.

iii) A first cousin is a child of a parent's sibling

- **FOL:** $\forall x \forall y (\text{IsFirstCousinOf}(x, y) \leftrightarrow \exists p1 \exists p2 (\text{IsParentOf}(p1, x) \wedge \text{IsParentOf}(p2, y) \wedge \text{IsSiblingOf}(p1, p2) \wedge \exists c (\text{IsChildOf}(x, p1) \wedge \text{IsChildOf}(y, p2) \wedge \text{IsSiblingOf}(p1, p2))))$

SPPU-TE-COMP-CONTENT – KSKA Git

- **Simpler/Alternative Interpretation (focusing on defining "first cousin"):**
$$\forall x \forall y (IsFirstCousinOf(x, y) \leftrightarrow \exists p_x \exists s_{p_y} (IsParentOf(p_x, x) \wedge IsParentOf(s_{p_y}, y) \wedge IsSiblingOf(p_x, s_{p_y})))$$
- **Explanation:** For any two individuals x and y , x is a first cousin of y if and only if there exists a parent of x (let's call them p_x) and a parent of y (let's call them p_y), such that p_x is a sibling of p_y .

iv) **You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all the time.**

- **FOL:** $(\forall p (Person(p) \rightarrow \exists t (Time(t) \wedge Fool(p, t)))) \wedge (\exists p (Person(p) \wedge \forall t (Time(t) \rightarrow Fool(p, t)))) \wedge \neg (\forall p (Person(p) \rightarrow \forall t (Time(t) \rightarrow Fool(p, t))))$
- **Explanation:**
 - **Part 1: "You can fool all the people some of the time"** $\forall p (Person(p) \rightarrow \exists t (Time(t) \wedge Fool(p, t)))$ For every person p , there exists at least one time t at which p can be fooled.
 - **Part 2: "and some of the people all the time"** $\exists p (Person(p) \wedge \forall t (Time(t) \rightarrow Fool(p, t)))$ There exists at least one person p who can be fooled at all times t .
 - **Part 3: "but you cannot fool all the people all the time."** $\neg (\forall p (Person(p) \rightarrow \forall t (Time(t) \rightarrow Fool(p, t))))$ It is NOT the case that for every person p , p can be fooled at all times t .

Q4) b) What is Knowledge Representation using propositional logic?

Compare propositional and predicate Logic.

Ans:

Knowledge Representation Using Propositional Logic

Propositional logic is a fundamental approach to **knowledge representation** in artificial intelligence (AI). It represents facts using **propositions**, which are statements that can be either true or false.

Key Components of Propositional Logic:

- **Propositions (Statements)** – Represent facts (e.g., "It is raining").
- **Logical Connectives:**
 - **AND (\wedge)** – Combines two true statements.
 - **OR (\vee)** – True if at least one statement is true.
 - **NOT (\neg)** – Negates a statement.
 - **IMPLICATION (\rightarrow)** – "If P, then Q."
 - **BICONDITIONAL (\leftrightarrow)** – "P if and only if Q."

Example of Knowledge Representation:

- **Fact:** "If it rains, the ground is wet."
- **Propositional Logic Representation:**

$P \rightarrow Q$

Where:

- **P** = "It is raining."
- **Q** = "The ground is wet."

Propositional logic is useful for **simple reasoning tasks**, but it lacks the ability to express **relationships between objects**, which is where **predicate logic** comes in.

Comparison: Propositional Logic vs. Predicate Logic

Aspect	Propositional Logic	Predicate Logic
Definition	Deals with simple statements that are either true or false.	Extends propositional logic by introducing variables, quantifiers, and predicates .



SPPU-TE-COMP-CONTENT – KSKA Git

Expressiveness	Limited to atomic facts .	Can represent relationships between objects .
Example	"It is raining" (P).	"X is greater than Y" \rightarrow Greater(X, Y) .
Use of Quantifiers	Not supported .	Uses \forall (Universal) and \exists (Existential) quantifiers.
Complexity	Easier to implement but lacks depth.	More powerful for knowledge representation and reasoning .
Applications	Used in simple rule-based systems .	Used in AI, databases, and expert systems .



Q5) a) Explain Forward Chaining and Backward Chaining. With its properties, advantages and disadvantages.

Ans:

Forward Chaining and Backward Chaining in AI

Forward Chaining and Backward Chaining are **inference techniques** used in **rule-based expert systems** to derive conclusions from given facts.

1. Forward Chaining

- **Definition:** A **data-driven** approach where reasoning starts from known facts and applies rules to infer new facts until a goal is reached.
- **Process:**
 1. Start with **initial facts**.
 2. Apply **rules** that match the facts.
 3. Derive **new facts** and repeat until the goal is achieved.
- **Example:** In **medical diagnosis**, if a patient has a fever and a rash, the system infers the possibility of measles.

Properties of Forward Chaining

- Works **progressively** from facts to conclusions.
- Suitable for **large datasets** where multiple inferences are needed.

Advantages:

- **Comprehensive** – Explores all possible conclusions.
- **Dynamic** – Adapts to new data as it arrives.
- **Efficient for broad knowledge bases.**

Disadvantages:

- **Memory-intensive** – Stores many intermediate facts.
- **Can be inefficient** for goal-oriented tasks.

2. Backward Chaining

- **Definition:** A **goal-driven** approach where reasoning starts from the goal and works backward to find supporting facts.
- **Process:**
 1. Start with the **goal**.



2. Identify **rules** that lead to the goal.
 3. Check if **facts** support the rules.
 4. Continue until a valid path is found or discard incorrect paths.
- **Example:** In **medical diagnosis**, if the goal is to confirm measles, the system checks if the patient has a fever and rash.

Properties of Backward Chaining

- Works **backward from the goal** to supporting facts.
- Suitable for **specific queries** where a direct answer is needed.

Advantages:

- **Efficient for goal-oriented tasks.**
- **Reduces unnecessary computations.**
- **Focused search** – Only explores relevant paths.

Disadvantages:

- **Limited exploration** – May miss alternative solutions.
- **Requires predefined goals** – Not ideal for broad knowledge bases.

Q5) b) Explain:

- i) **Unification in FOL**
- ii) **Reasoning with Default information**

Ans:

i) Unification in First-Order Logic (FOL)

Unification in **First-Order Logic (FOL)** is a process used in **automated reasoning** to make two logical expressions identical by finding a **common substitution**. It is essential for **theorem proving, inference mechanisms, and AI-based logical reasoning**.

Key Concepts of Unification:

- **Substitution:** Replacing variables with constants or other variables to match expressions.
- **Most General Unifier (MGU):** The simplest substitution that unifies two expressions.
- **Unification Algorithm:** Determines whether two expressions can be unified and provides necessary substitutions.

Example of Unification:

Consider two predicates:

- $P(x, \text{Dog})$
- $P(\text{Alex}, y)$

To unify them, we substitute:

- $x \rightarrow \text{Alex}$
- $y \rightarrow \text{Dog}$

After substitution, both predicates become $P(\text{Alex}, \text{Dog})$, meaning they are unified.

ii) Reasoning with Default Information

Default reasoning is a method in AI where conclusions are drawn based on **typical assumptions**, even when complete information is unavailable. It allows AI systems to make **educated guesses** while handling **uncertainty**.

Key Aspects of Default Reasoning:



SPPU-TE-COMP-CONTENT – KSKA Git

- **Default Rules:** Common assumptions (e.g., "Most birds can fly").
- **Exceptions:** Special cases that override defaults (e.g., "Penguins cannot fly").
- **Non-Monotonicity:** New information can **change previously drawn conclusions**.

Example of Default Reasoning:

- **Initial assumption:** "Birds can fly."
- **New information:** "This bird is a penguin."
- **Updated conclusion:** "This bird cannot fly."



Q6) a) Explain FOL inference for following Quantifiers.

- i) Universal Generalization
- ii) Universal Instantiation
- iii) Existential Instantiation
- iv) Existential introduction

Ans:

First-Order Logic (FOL) Inference for Quantifiers

First-Order Logic (FOL) inference rules allow us to derive new facts from existing statements using **quantifiers**. Below are the key inference rules for **universal and existential quantifiers**.

i) Universal Generalization

- **Definition:** If a statement holds for an arbitrary element in a domain, it can be generalized to apply to all elements.
- **Representation:**

$$P(c) \rightarrow \forall x P(x)$$

- **Example:**
 - Given: "John likes ice cream" $\rightarrow \text{Likes}(\text{John}, \text{IceCream})$.
 - Generalization: "All people like ice cream" $\rightarrow \forall x \text{ Likes}(x, \text{IceCream})$.

Universal generalization is valid **only if the element (c) represents a general case applicable to all elements**.

ii) Universal Instantiation

- **Definition:** If a statement is true for all elements in a domain, it must be true for any specific element.
- **Representation:**

$$\forall x P(x) \rightarrow P(c)$$

- **Example:**
 - Given: "All students study AI" $\rightarrow \forall x \text{ Student}(x) \rightarrow \text{Studies}(x, \text{AI})$.



SPPU-TE-COMP-CONTENT – KSKA Git

- Instantiation: "Alice is a student, so she studies AI" \rightarrow $\text{Studies}(\text{Alice}, \text{AI})$.

Universal instantiation allows us to **derive specific facts from general statements**.

iii) Existential Instantiation

- **Definition:** If an existential statement is true, we can introduce a new constant representing an instance of that statement.
- **Representation:**

$$\exists x P(x) \rightarrow P(c)$$

- **Example:**
 - Given: "There exists a person who owns a car" $\rightarrow \exists x \text{ Owns}(x, \text{Car})$.
 - Instantiation: "Let's assume Bob owns a car" $\rightarrow \text{Owns}(\text{Bob}, \text{Car})$.

Existential instantiation **introduces a specific instance** but does not specify uniqueness.

iv) Existential Introduction

- **Definition:** If a statement is true for a specific element, we can generalize it using an existential quantifier.
- **Representation:**

$$P(c) \rightarrow \exists x P(x)$$

- **Example:**
 - Given: "Alice owns a car" $\rightarrow \text{Owns}(\text{Alice}, \text{Car})$.
 - Introduction: "There exists a person who owns a car" $\rightarrow \exists x \text{ Owns}(x, \text{Car})$.

Existential introduction **allows us to infer the existence of an entity based on a specific instance**.



Q6) b) What is Ontological Engineering, in details with its categories object and Model.

Ans:

Ontological Engineering: Definition and Categories

What is Ontological Engineering?

Ontological Engineering is a field in **computer science and artificial intelligence** that focuses on **designing, developing, and managing ontologies**—structured representations of knowledge that define relationships between concepts, entities, and data. It plays a crucial role in **knowledge representation, semantic web technologies, and AI-driven reasoning systems**.

Categories in Ontological Engineering

Ontologies are classified into different categories based on their scope and purpose:

Category	Description	Example
Top-Level Ontologies	Define general concepts like time, space, and objects.	SUMO (Suggested Upper Merged Ontology)
Domain Ontologies	Represent knowledge specific to a particular field.	Medical Ontologies (SNOMED CT, Gene Ontology)
Task Ontologies	Describe processes and activities within a domain.	Workflow Ontologies for Business Processes
Application Ontologies	Tailored for specific applications or software systems.	E-commerce Ontologies for Product Classification

Each category serves a distinct purpose in **knowledge structuring and AI-based reasoning**.

Objects and Models in Ontological Engineering

1. Objects in Ontological Engineering

Objects refer to **entities or instances** within an ontology. They can be:

- **Physical Objects** – Tangible entities like cars, buildings, or human beings.
- **Conceptual Objects** – Abstract entities like mathematical formulas or legal rules.
- **Events** – Actions or occurrences, such as meetings or transactions.

Objects are defined using **classes, attributes, and relationships** to ensure structured knowledge representation.

2. Models in Ontological Engineering



SPPU-TE-COMP-CONTENT – KSKA Git

Models define **how ontologies are structured and implemented**. They include:

- **Formal Models** – Use logical frameworks like **Description Logic (DL)** and **OWL (Web Ontology Language)**.
- **Graph-Based Models** – Represent ontologies using **nodes and edges** (e.g., RDF graphs).
- **Hierarchical Models** – Organize concepts in a **tree-like structure** for classification.

Models help in **semantic reasoning, interoperability, and AI-driven decision-making**



Q7) a) Explain with an example Goal Stack Planning (STRIPS algorithm).

Ans:

Goal Stack Planning (STRIPS Algorithm)

Goal Stack Planning is a problem-solving approach used in **Artificial Intelligence (AI)** to generate a sequence of actions that achieve a desired goal. It is based on the **Stanford Research Institute Problem Solver (STRIPS)** algorithm, which represents planning problems using **states, goals, and actions**.

How Goal Stack Planning Works?

1. **Initialize the Stack** – Place the goal on top of the stack.
2. **Expand the Goal** – If the goal is a compound statement, break it into sub-goals.
3. **Apply Operators** – Select an action that satisfies the goal and push its **preconditions** onto the stack.
4. **Resolve Preconditions** – If preconditions are not met, push them as new sub-goals.
5. **Execute Actions** – Once all preconditions are satisfied, execute the action and update the state.
6. **Repeat Until Goal is Achieved** – Continue until the stack is empty and the goal state is reached.

Example: Block Stacking Problem

Problem Statement:

A robot needs to arrange blocks **A, B, and C** in a specific order:

- **Initial State:** $\text{On}(A, \text{Table}), \text{On}(B, \text{Table}), \text{On}(C, \text{Table})$
- **Goal State:** $\text{On}(A, B), \text{On}(B, C)$

Goal Stack Execution:

1. **Push Goal:** $\text{On}(A, B), \text{On}(B, C)$ onto the stack.
2. **Expand Goal:**
 - $\text{On}(A, B)$ requires $\text{Clear}(B)$ & $\text{Holding}(A)$.
 - $\text{On}(B, C)$ requires $\text{Clear}(C)$ & $\text{Holding}(B)$.
3. **Push Preconditions:**
 - $\text{Clear}(B), \text{Holding}(A), \text{Clear}(C), \text{Holding}(B)$.
4. **Execute Actions:**

SPPU-TE-COMP-CONTENT – KSKA Git

- Pick up A, place it on B.
 - Pick up B, place it on C.
5. **Final State:** On (A, B) , On (B, C) .



Q7) b) Explain with example, how planning is different from problem solving.

Ans:

Aspect	Planning	Problem Solving
Definition	Planning involves determining a sequence of actions to achieve a goal.	Problem-solving is the process of finding a solution to a given challenge.
Approach	Proactive – Defines steps before execution.	Reactive – Identifies solutions when a problem arises.
Goal	Focuses on structuring actions to reach a desired state.	Focuses on eliminating obstacles to reach a solution.
Example	A robot plans a route before navigating a maze.	A robot encounters an obstacle and finds a way around it .
AI Application	Used in automated scheduling, robotics, and strategic decision-making .	Used in game AI, troubleshooting, and optimization problems .

Example: Planning vs. Problem Solving in Robotics

Scenario: A robot needs to move from Point A to Point B in a warehouse.

1. Planning Approach:

- The robot **predefines a path** using a navigation algorithm.
- It considers **obstacles, shortest routes, and efficiency** before moving.
- The robot follows the **planned sequence of actions** to reach Point B.

2. Problem-Solving Approach:

- The robot starts moving **without a predefined path**.
- If it encounters an obstacle, it **reacts dynamically** to find an alternative route.
- It **adjusts its movement** based on real-time conditions.

Q7) c) Explain AI components and AI architecture.

Ans:

AI Components and AI Architecture

Artificial Intelligence (AI) consists of several **key components** that work together to enable intelligent behavior. These components form the foundation of AI systems, allowing them to **learn, reason, perceive, and interact** with their environment.

1. Components of AI

Component	Description	Example Applications
Learning	AI improves performance by analyzing data and identifying patterns.	Machine Learning (ML), Deep Learning (DL)
Reasoning	AI makes logical decisions based on given information.	Expert Systems, Decision Support Systems
Perception	AI interprets sensory data from the environment.	Computer Vision, Speech Recognition
Natural Language Processing (NLP)	AI understands and processes human language.	Chatbots, Voice Assistants
Knowledge Representation	AI stores and organizes information for reasoning.	Semantic Web, Ontologies
Planning & Problem Solving	AI generates sequences of actions to achieve goals.	Robotics, Automated Scheduling
Ethics & Bias Handling	AI ensures fairness and avoids biased decisions.	AI Governance, Responsible AI

2. AI Architecture

AI architecture defines the **structure and interaction** of AI components within a system. It ensures efficient processing and decision-making.

Types of AI Architectures:

Architecture Type	Description	Example Systems
Reactive AI	Responds to inputs without memory or learning.	Chess-playing AI, Simple Bots
Limited Memory AI	Uses past experiences for decision-making.	Autonomous Vehicles, Fraud Detection
Theory of Mind AI	Understands emotions and social interactions.	Advanced AI Assistants (Future AI)
Self-Aware AI	AI with consciousness and self-awareness.	Hypothetical Future AI



Q8) a) Explain Planning in non-deterministic domain.

Ans:

Planning in Non-Deterministic Domains

Planning in **non-deterministic domains** refers to decision-making in environments where **actions have uncertain outcomes**. Unlike deterministic planning, where each action leads to a predictable result, non-deterministic planning must account for **multiple possible effects** of an action.

• Conditional Planning

- Plans include **branches** to account for multiple possible outcomes.
- Example: A **robot navigating a maze** considers alternate routes when encountering obstacles.

• Execution Monitoring

- The agent **observes** results and dynamically **adjusts its strategy**.
- Example: **Self-driving cars** modify their driving behaviour based on unexpected road or weather conditions.

• Replanning

- If an action fails, the agent generates **a new plan** to achieve the goal.
- Example: **AI-powered assistants** adjust their recommendations based on evolving user interactions.

• Belief Space Planning

- The agent maintains a set of **possible states** and refines predictions through observations.
- Example: **Medical diagnosis AI** updates disease predictions as new patient data arrives.

Characteristics of Non-Deterministic Planning

- **Uncertain Action Effects** – Actions may lead to different possible outcomes.
- **Partial Observability** – The agent may not have complete knowledge of the environment.
- **Conditional Planning** – Plans must include contingencies for different scenarios.
- **Execution Monitoring & Replanning** – The agent must adjust its plan based on observed results.



SPPU-TE-COMP-CONTENT – KSKA Git

Example: Robot Navigation in a Non-Deterministic Environment

Consider a robot navigating a **warehouse** where doors may or may not open when pushed.

1. **Initial Plan:** Move towards the exit.
2. **Uncertainty:** The door may be **locked or unlocked**.
3. **Conditional Plan:**
 - If the door opens → Proceed.
 - If the door is locked → Try another route.
4. **Execution Monitoring:** The robot checks the door status and **adjusts its plan accordingly**.



Q8) b) Explain.

- i) **Importance of planning**
- ii) **Algorithm for classical planning**

Ans:

i) Importance of Planning

Planning is a **critical process** in decision-making, ensuring efficiency, goal achievement, and resource optimization. It plays a vital role in **business, AI, project management, and problem-solving**.

Benefits of Planning:

- **Provides Direction** – Establishes clear objectives and structured steps to achieve them.
- **Reduces Uncertainty** – Helps anticipate risks and prepare for future challenges.
- **Optimizes Resource Allocation** – Ensures efficient use of time, money, and manpower.
- **Encourages Innovation** – Allows creative problem-solving and strategic thinking.
- **Improves Decision-Making** – Enables informed choices based on structured analysis.
- **Enhances Productivity** – Minimizes redundancy and streamlines operations.

Planning is essential for **long-term success**, ensuring adaptability and efficiency in dynamic environments.

ii) Algorithm for Classical Planning

Classical planning in AI involves **finding a sequence of actions** that transition from an initial state to a goal state. It is widely used in **robotics, logistics, and automated decision-making**.

Steps in Classical Planning Algorithm:

1. **Define the Problem** – Specify the **initial state, goal state, and available actions**.
2. **Generate the Search Space** – Create a **state-space graph** representing possible transitions.
3. **Apply Search Algorithms** – Use methods like *Breadth-First Search (BFS)*, *Depth-First Search (DFS)*, or *A Search** to find an optimal path.



SPPU-TE-COMP-CONTENT – KSKA Git

4. **Check Preconditions & Effects** – Ensure each action satisfies **preconditions** before execution.
5. **Construct the Plan** – Sequence actions to transition from the **initial state to the goal state**.
6. **Execute & Monitor** – Implement the plan while adjusting for **unexpected changes**.



Q8) c) What is AI explain scope of AI in all walks of Life also explain future opprotunities with AI.

Ans:

What is Artificial Intelligence (AI)?

Artificial Intelligence (AI) refers to the **simulation of human intelligence** in machines that can **learn, reason, perceive, and make decisions**. AI systems use **algorithms, data, and computational models** to perform tasks that typically require human intelligence, such as **speech recognition, problem-solving, and autonomous decision-making**.

Scope of AI in All Walks of Life

AI has transformed various industries, enhancing efficiency and innovation. Here's how AI is impacting different sectors:

1. Healthcare

- AI-powered **diagnostic tools** detect diseases with high accuracy.
- **Predictive analytics** helps in early disease detection and personalized treatment.
- **Robotic surgeries** improve precision and reduce recovery time.

2. Education

- AI-driven **adaptive learning platforms** personalize education for students.
- **Automated grading systems** save time for educators.
- AI-powered **virtual tutors** assist students in learning complex subjects.

3. Finance

- AI enhances **fraud detection** by analyzing transaction patterns.
- **Algorithmic trading** optimizes investment strategies.
- AI-powered **chatbots** improve customer service in banking.

4. Transportation

- **Autonomous vehicles** use AI for navigation and safety.
- AI optimizes **traffic management** to reduce congestion.
- **Predictive maintenance** ensures efficient fleet management.

5. Manufacturing



SPPU-TE-COMP-CONTENT – KSKA Git

- AI-driven **automation** improves production efficiency.
- **Predictive analytics** minimizes equipment failures.
- **Quality control systems** detect defects in real-time.

6. Cybersecurity

- AI enhances **threat detection** and prevents cyberattacks.
- **Automated security protocols** strengthen data protection.
- AI-driven **fraud prevention** safeguards financial transactions.

7. Entertainment & Media

- AI-powered **content recommendation systems** personalize user experiences.
- **Deepfake technology** enables realistic video editing.
- AI assists in **music composition and film production**.

You can explore more details on AI's scope [here](#) and [here](#).

Future Opportunities with AI

AI is shaping the future with groundbreaking innovations and career opportunities:

1. Enhanced Efficiency & Automation

- AI automates **routine tasks**, freeing up human workers for strategic roles.
- **AI-driven robotics** improve productivity in industries.

2. Personalized Healthcare

- AI enables **tailored treatment plans** based on patient data.
- **Remote patient monitoring** enhances telemedicine.

3. Smart Cities & Infrastructure

- AI optimizes **resource management** in urban planning.
- **Traffic control systems** reduce congestion.

4. Cybersecurity Advancements

- AI strengthens **real-time threat detection**.
- **Adaptive defense mechanisms** improve security protocols.

5. AI in Creative Industries



Other Subjects: www.pyqspot.com

SPPU-TE-COMP-CONTENT – KSKA Git

- AI assists in **content creation, music composition, and video editing.**
- **AI-powered design tools** enhance creativity.

6. AI in Space Exploration

- AI helps in **autonomous spacecraft navigation.**
- **Predictive analytics** improves mission planning.



Other Subjects: www.pyqspot.com
