UNIT 3: REASONING AND PLANNING IN AI

Subject: *Artificial Intelligence*
Course Code: *[PCC-CSE-304G]*
Compiled By: *Deepak Modi (223100)*

Reasoning under Uncertainty: Basics of Probability Theory, Probabilistic Reasoning, Bayesian Reasoning, Dempster-Shafer Theory.
Planning: Introduction to Planning, Representation of Planning, Partial-order Planning.

---

1. Reasoning under Uncertainty

> PYQ: Explain how uncertainty is managed in AI. (2021, 5 marks)
> PYQ: What is Probability Theory in terms of Reasoning under Uncertainty? Elaborate with example. (2022, 15 marks)

In real life, AI often deals with situations where information is **incomplete**, **uncertain**, or **not fully reliable**. So, instead of making decisions with exact facts, AI uses **probability** and **logic** to make the best guess. This is where **probabilistic reasoning** comes in.

1.1 Basics of Probability Theory

> PYQ: What is Probability Theory in terms of Reasoning under Uncertainty? Elaborate with example. (2022, 15 marks)

What is Probability?

- **Probability** is a way to measure how **likely** something is to happen.
- It is a number between **0 and 1**:
  - `0` means **impossible**
  - `1` means **certain**
- Example: Probability of flipping a coin and getting heads = **0.5** (or 50%)

Formula of Probability:

$$P(E) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}}$$

Example: Rolling a die (6 sides), what's the probability of getting a 4?

- Favorable outcomes = 1 (only one '4')
- Total outcomes = 6
- So, P(4)=1/6≈0.167

Some Important Terms:

Term	Meaning	Example
Experiment	An action like tossing a coin or rolling a die	Drawing a card from a deck

Term	Meaning	Example
Sample Space (S)	All possible outcomes	For a die: {1, 2, 3, 4, 5, 6}
Event (E)	A specific outcome or set of outcomes	Getting an even number on a die
Favorable Outcome	The result you want to happen	Getting a 6 when you need it
Joint Probability	Probability of two events occurring together	P(A and B)
Conditional Probability	Probability of one event given another occurred	P(A|B)

Types of Probability:

1. **Theoretical Probability**
    - Based on logical reasoning
    - Example: Tossing a coin → 0.5 for heads
2. **Experimental Probability**
    - Based on real experiments/data
    - Example: After 100 tosses, heads came 48 times → P(Heads)=48 / 100=0.48
3. **Subjective Probability**
    - Based on personal judgment or expertise
    - Example: A doctor estimating 70% chance of recovery based on experience

Rules of Probability:

1. **Range Rule**: $0 \leq P(E) \leq 1$
2. **Sum Rule** (for all outcomes): $P(E_1)+P(E_2)+...+P(E_n)=1$
3. **Complement Rule**: $P(\text{Not } E)=1-P(E)$
4. **Multiplication Rule**: P(A and B) = P(A) × P(B|A)
5. **Addition Rule**: P(A or B) = P(A) + P(B) - P(A and B)

Conditional Probability:

The probability of an event A occurring given that event B has occurred is:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Example: In a class of 30 students, 15 are boys and 10 students passed the exam, including 3 boys. What's the probability that a randomly selected student is a boy, given that they passed the exam?

- P(boy|passed) = P(boy ∩ passed)/P(passed) = 3/10 = 0.3

Why Probability is Important in AI?

- AI often faces situations with **incomplete data**.
- Probability helps the system **make the best possible decision**.
- Used in medical diagnosis, spam detection, voice recognition, etc.
- Enables AI to **quantify uncertainty** rather than ignore it.
- Forms the foundation for **machine learning** algorithms.

Real-world Applications:

Application	How Probability Helps
Weather Forecasting	Calculating chance of rain based on atmospheric conditions
Stock Market Prediction	Estimating probability of price movements
Recommendation Systems	Determining likelihood a user will enjoy certain content
Natural Language Processing	Calculating probability of word sequences

1.2 Probabilistic Reasoning

What is Probabilistic Reasoning?

- **Probabilistic Reasoning** is a method used by AI to **make decisions or predictions** when it doesn't have full or certain information.
- It is based on **probability theory**.
- Instead of saying "Yes" or "No," AI says **"Most likely yes"** or **"Probably no."**

Why AI Needs It?

- In real-world situations, AI doesn't always know everything.
- For example:
    - A doctor AI sees a patient with **fever and cough**. It doesn't know the exact disease, but:
        - Flu: 70% chance
        - COVID-19: 20% chance
        - Cold: 10% chance
          → So, it selects **flu** as the most probable cause.

How It Works (Simple Steps):

1. **Collect Data**
   (e.g., symptoms, signals, images)
2. **Calculate Probabilities**
   (e.g., what's the chance of each possible result)
3. **Make the Best Guess**
   (select the one with the **highest probability**)
4. **Update Beliefs** (as new information comes in)

Example: Spam Email Classification

Consider an email classifier that determines if messages are spam or legitimate:

Email Features	Legitimate (P=0.6)	Spam (P=0.4)
Contains "FREE"	10% chance	60% chance
Contains "Meeting"	50% chance	5% chance
Has attachment	30% chance	40% chance

When a new email arrives containing "FREE" and an attachment:

- Initial probabilities: P(Legitimate)=0.6, P(Spam)=0.4
- After considering features:
  - P(Legitimate|features) ∝ 0.6 × 0.1 × 0.3 = 0.018
  - P(Spam|features) ∝ 0.4 × 0.6 × 0.4 = 0.096

→ After normalization, P(Spam|features) ≈ 0.84 (84%) → The AI classifies it as **Spam** with high confidence.

Another Example:

A robot is trying to decide whether an object is a ball or a fruit:

Object Feature	Ball (P=0.3)	Fruit (P=0.7)
Round shape	Very likely	Very likely
Soft texture	Less likely	More likely
Sweet smell	Very unlikely	Very likely

→ After analyzing all features, the robot chooses "Fruit" because the overall probability is much higher.

Use Cases in AI:

Area	How Probabilistic Reasoning Helps	Example Algorithm
Medical AI	Predict diseases based on symptoms	Bayesian networks
Spam Filter	Check if email is spam or not	Naive Bayes classifier
Self-driving cars	Predict pedestrian movements	Monte Carlo localization
Speech Recognition	Match voice to most probable words	Hidden Markov Models
Game AI	Make strategic decisions with incomplete information	Monte Carlo Tree Search

Benefits:

- Handles uncertainty and incomplete information effectively
- Provides a structured way to update beliefs
- Can model complex relationships between variables
- Widely applicable in various AI domains

Limitations:

- Requires good prior probabilities
- Can be computationally intensive for complex problems
- Sometimes makes unrealistic independence assumptions (in simplified models)
- May struggle with completely novel situations with no prior information

1.3 Bayesian Reasoning

> PYQ: Explain Probability and Bayes Theorem with example. (2023, 15 marks)
> PYQ: Bayesian Reasoning (2022, 2.5 marks)
> PYQ: What is the difference between prior and posterior probability? (2024, 1 mark)

What is Bayesian Reasoning?

- **Bayesian Reasoning** is a method in AI that **updates its beliefs** when **new information** (evidence) is available.
- It is based on **Bayes' Theorem** from probability theory.
- AI starts with a guess (called **prior probability**) and **updates** it using new data (called **evidence**).

Intuitive Example:

Imagine you're a doctor with a patient showing some symptoms.

- Initially, you think there's a 10% chance they have Disease X (prior)
- You run a test that is 90% accurate (if someone has Disease X, the test will be positive 90% of the time)
- The test comes back positive
- Should you now believe there's a 90% chance they have Disease X? Not necessarily!

Bayesian reasoning helps you calculate exactly how much to update your belief based on this new evidence.

Bayes' Theorem Formula:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Where:

- P(H) = **Prior**: Belief in hypothesis **H** before seeing the evidence
- P(EIH) = **Likelihood**: How likely is evidence **E** if hypothesis **H** is true
- P(E) = Total probability of evidence **E** (can be calculated as P(E|H)P(H) + P(E|¬H)P(¬H))
- P(HIE) = **Posterior**: Updated belief in **H** after seeing evidence **E**

Step-by-Step Calculation Example:

Let's work through a complete example:

- 1% of people have a certain disease (prior probability)
- The test is 95% accurate for those who have the disease (sensitivity)
- The test has a 10% false positive rate (90% specificity)
- If someone tests positive, what's the probability they actually have the disease?

Step 1: Identify what we know

- P(Disease) = 0.01 (prior)
- P(Positive|Disease) = 0.95 (likelihood/sensitivity)
- P(Positive|No Disease) = 0.10 (false positive rate)

Step 2: Apply Bayes' theorem

- P(Disease|Positive) = [P(Positive|Disease) × P(Disease)] / P(Positive)

Step 3: Calculate P(Positive)

- P(Positive) = P(Positive|Disease)×P(Disease) + P(Positive|No Disease)×P(No Disease)
- P(Positive) = 0.95×0.01 + 0.10×0.99 = 0.0095 + 0.099 = 0.1085

Step 4: Calculate posterior probability

- P(Disease|Positive) = (0.95×0.01)/0.1085 = 0.0095/0.1085 ≈ 0.0876 or about 8.8%

Despite the positive test result, there's only about a 9% chance the person actually has the disease! This is because the disease is rare (low prior probability).

Key Terms:

1. **Prior Probability**
    - **Meaning:** Initial belief or guess before seeing any evidence.
    - **Example:** There's a **1% chance** someone has a disease (before test).
    - **Mathematical notation:** P(H)
2. **Likelihood**
    - **Meaning:** How likely we are to see the evidence if the hypothesis is true.
    - **Example:** If someone has the disease, there's a 95% chance the test will be positive.
    - **Mathematical notation:** P(E|H)
3. **Evidence (E)**
    - **Meaning:** New information or data.
    - **Example:** Person gets a positive test result.
    - **Mathematical notation:** P(E)
4. **Posterior Probability**
    - **Meaning:** Updated belief after seeing the evidence.
    - **Example:** After the test, now there's an 8.8% chance they have the disease.
    - **Mathematical notation:** P(H|E)

Difference between Prior and Posterior Probability

> PYQ: What is the difference between prior and posterior probability? (2024, 1 mark)

Aspect	Prior Probability	Posterior Probability

Aspect	Prior Probability	Posterior Probability
Definition	Initial probability of an event before new evidence is considered	Updated probability after taking new evidence into account
When Used	At the beginning of analysis	After collecting and analyzing new evidence
Formula	P(H)	P(H|E) = P(E|H)·P(H)/P(E)
Notation	Simply P(H)	P(H|E) (read as "probability of H given E")
Role in Bayesian	Starting point for Bayesian inference	The result of Bayesian inference
Example	1% chance of having a disease before testing	8.8% chance of having disease after positive test
Mathematical Relationship	Input to Bayes' theorem	Output of Bayes' theorem

Prior probability represents our initial belief based on existing information, while posterior probability is the revised belief after incorporating new evidence. The posterior becomes the new prior if additional evidence is obtained later.

The Bayesian Update Process:

1. **Start with prior beliefs** - P(H)
2. **Observe new evidence** - E
3. **Calculate how likely the evidence would be** if hypothesis is true - P(E|H)
4. **Calculate total probability of seeing this evidence** - P(E)
5. **Update belief using Bayes' theorem** - P(H|E)
6. **Use this posterior as the new prior** for future updates

Naive Bayes: A Practical Implementation

Naive Bayes is a popular algorithm based on Bayesian reasoning, widely used for:

- Email spam filtering
- Document classification
- Sentiment analysis

It makes the "naive" assumption that features are independent of each other given the class, which simplifies calculations dramatically.

Example: Email Spam Classification

- Prior: P(Spam) = 0.3, P(Not Spam) = 0.7
- If "free" appears: P("free"|Spam) = 0.6, P("free"|Not Spam) = 0.05
- If "meeting" appears: P("meeting"|Spam) = 0.01, P("meeting"|Not Spam) = 0.4

For an email containing both "free" and "meeting":

- P(Spam|words) ∝ 0.3 × 0.6 × 0.01 = 0.0018
- P(Not Spam|words) ∝ 0.7 × 0.05 × 0.4 = 0.014

After normalization, P(Not Spam|words) ≈ 0.89 or 89%, so the email is classified as legitimate.

Why It's Useful in AI:

Use Case	How Bayesian Reasoning Helps	Example Application
Medical Diagnosis	Updates disease probabilities with new test results	COVID-19 risk assessment
Spam Detection	Analyzes word patterns to filter emails	Gmail's spam filter
Robot Vision	Updates object identification with better images	Self-driving cars
Voice Assistant	Improves word recognition as more is spoken	Siri, Alexa, Google Assistant
Recommendation Systems	Updates user preferences based on interactions	Netflix, Amazon recommendations

Limitations of Bayesian Reasoning:

- Requires good prior probabilities
- Can be computationally intensive for complex problems
- Sometimes makes unrealistic independence assumptions (in simplified models)
- May struggle with completely novel situations with no prior information

1.4 Dempster-Shafer Theory

> PYQ: Dempster-Shafer Theory (2023, 2.5 marks)
> PYQ: Explain in detail Dempster-Shafer Theory. (2022, 15 marks)
> PYQ: Explain Dempster shafer theory with the help of example. (2021, 10 marks)
> PYQ: What is Dempster-Shafer's theory? Explain with a suitable example. (2024, 15 marks)
> PYQ: What is the difference between Bayesian and Dempster-Shafer theory? (2024, 1 mark)

What is Dempster-Shafer Theory?

- It is an another **mathematical method** for reasoning under uncertainty — like probability, but more **flexible**. Unlike probability, it allows for **belief** and **doubt** (ignorance). Used when AI has **incomplete or conflicting data**.
- *It is also called Belief Theory or Evidence Theory.*
- It allows AI to work with **incomplete**, **uncertain**, or even **conflicting** information.
- It helps AI to **combine evidence** from different sources and make decisions based on that.
- It gives a **range of belief** instead of just one number.

Key Concepts in Detail

Term	Meaning	Mathematical Representation

Term	Meaning	Mathematical Representation
Frame of Discernment (Θ)	The set of all possible states or hypotheses	Θ = {A, B, C, ...}
Power Set (2^Θ)	All possible subsets of the frame	2^Θ = {∅, {A}, {B}, {A,B}, ...}
Basic Probability Assignment (m)	Assigns belief mass to elements of the power set	m: 2^Θ → [0,1], where m(∅)=0 and Σm(A)=1 for all A in 2^Θ
Belief (Bel)	How much the evidence supports a hypothesis	Bel(A) = Σm(B) for all B⊆A
Plausibility (Pl)	How much the evidence does not reject a hypothesis	Pl(A) = 1 - Bel(¬A)
Belief Interval	The range between belief and plausibility	[Bel(A), Pl(A)]

The key formula to remember is:

```
Belief ≤ Actual Probability ≤ Plausibility
```

DST vs. Traditional Probability

Probability Theory	Dempster-Shafer Theory
Needs exact probabilities	Works even with partial or unknown info
One number	Range of belief (lower & upper bounds)
P(A) + P(not A) = 1	Bel(A) + Bel(not A) ≤ 1
Cannot represent ignorance directly	Can explicitly represent ignorance
Requires complete knowledge	Works with incomplete knowledge

Step-by-Step Example: Medical Diagnosis

Let's work through a complete example to understand DST:

Step 1: Define the frame of discernment

- Θ = {Flu, COVID, Cold} (possible diseases)

Step 2: Consider evidence from Test 1

- Test 1 assigns:
  - m_1(Flu) = 0.7 (70% belief specifically in Flu)
  - m_1(Cold) = 0.1 (10% belief specifically in Cold)
  - m_1({Flu, COVID, Cold}) = 0.2 (20% ignorance/uncertainty)

Step 3: Consider evidence from Test 2

- Test 2 assigns:

- m_2(COVID) = 0.6 (60% belief specifically in COVID)
- m_2({Flu, COVID}) = 0.3 (30% belief it's either Flu or COVID)
- m_2({Flu, COVID, Cold}) = 0.1 (10% ignorance/uncertainty)

Step 4: Calculate belief for Flu

- Bel_1(Flu) = m_1(Flu) = 0.7
- Bel_2(Flu) = m_2(Flu) = 0 (Test 2 gives no specific belief to Flu alone)

Step 5: Calculate plausibility for Flu

- Pl_1(Flu) = m_1(Flu) + m_1({Flu, COVID, Cold}) = 0.7 + 0.2 = 0.9
- Pl_2(Flu) = m_2({Flu, COVID}) + m_2({Flu, COVID, Cold}) = 0.3 + 0.1 = 0.4

Step 6: Combine evidence using Dempster's rule

- Calculate the combined mass function m_{12} using Dempster's rule of combination
- After combination, we might get:
  - m_{12}(Flu) = 0.35
  - m_{12}(COVID) = 0.50
  - m_{12}(Cold) = 0.05
  - m_{12}({Flu, COVID}) = 0.10
  - m_{12}({Flu, COVID, Cold}) = 0.00

Step 7: Calculate final belief and plausibility

- Bel(Flu) = m_{12}(Flu) = 0.35
- Pl(Flu) = m_{12}(Flu) + m_{12}({Flu, COVID}) = 0.35 + 0.10 = 0.45

Conclusion: Based on both tests, we can say with 35% belief that the patient has the Flu, and the plausibility is 45%.

Real-Life Example: Fire Detection System

A building has multiple sensors for detecting fires:

- **Smoke Sensor** says:
  - m_1(Fire) = 0.6 (60% belief in fire)
  - m_1(No Fire) = 0.1 (10% belief in no fire)
  - m_1({Fire, No Fire}) = 0.3 (30% uncertainty)
- **Temperature Sensor** says:
  - m_2(Fire) = 0.7 (70% belief in fire)
  - m_2(No Fire) = 0.2 (20% belief in no fire)
  - m_2({Fire, No Fire}) = 0.1 (10% uncertainty)
- **After combining evidence:**
  - Belief in Fire = 0.88 (strong evidence for fire)
  - Plausibility of Fire = 0.94 (very little evidence against fire)

- Belief interval = [0.88, 0.94]

The AI can now make a decision to trigger the fire alarm based on this strong combined belief.

> PYQ: What is the difference between Bayesian and Dempster-Shafer theory? (2024, 1 mark)

Difference Between Bayesian and Dempster-Shafer Theory

Aspect	Bayesian Theory	Dempster-Shafer Theory
Basic Unit	Probabilities assigned to specific outcomes	Belief assigned to sets of possibilities
Handling Ignorance	Requires assigning probabilities even with no evidence	Can explicitly represent ignorance through belief intervals
Precision	Requires precise prior probabilities	Works with partial or incomplete knowledge
Representation	Single probability value P(H)	Belief-Plausibility interval [Bel(H), Pl(H)]
Mathematical Approach	Uses conditional probabilities	Uses belief functions and plausibility
Assumption	Assumes all possible hypotheses are known	Can handle unknown hypotheses
Evidence Combination	Uses Bayes' rule for updating	Uses Dempster's rule of combination
Best Used When	Prior probabilities are reliable	Working with conflicting or incomplete evidence

Bayesian theory requires specific probability values and assumes complete knowledge of all possible outcomes, while Dempster-Shafer theory can explicitly represent uncertainty using belief intervals and works better with incomplete information.

Dempster's Rule of Combination

Dempster's rule is used to combine evidence from multiple sources. For two basic probability assignments m_1 and m_2, the combined mass function m_{12} is:

$$m_{1,2}(A) = \frac{1}{1-K} \sum_{B \cap C = A} m_1(B) \cdot m_2(C)$$

Where K represents the degree of conflict between the sources:

$$K = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C)$$

The denominator (1-K) serves as a normalization factor that redistributes the mass of the conflicting evidence.

Example:
Source 1: $m_1(A) = 0.6$, $m_1(B) = 0.4$
Source 2: $m_2(A) = 0.5$, $m_2(C) = 0.5$
→ After applying Dempster's rule, we get the combined belief allocation.

Applications in AI

Field	Use of Dempster-Shafer Theory	Example
Medical Diagnosis	Combine results from multiple tests	Integrating blood test and imaging results
Sensor Fusion	Merge data from different sensors in robots	Combining LIDAR, camera, and radar in autonomous vehicles
Fault Detection	Detect problems when data is incomplete	Aircraft system monitoring
Decision Making	Delay or make decisions under uncertainty	Military tactical decision support systems
Image Recognition	Combine evidence from different features	Multi-feature face recognition systems

Advantages of DST:

- Can represent **total ignorance** (which Bayesian methods struggle with)
- Doesn't require **prior probabilities**
- Can handle **conflicting evidence**
- Provides **belief intervals** rather than single probabilities
- Allows for **incremental evidence gathering**

Limitations of DST:

- Computationally **expensive** for large frames of discernment
- Dempster's rule can give **counter-intuitive results** with highly conflicting evidence
- Interpretations can be **complex**
- Less **well-established** than Bayesian methods in some domains

---

2. Planning

Planning in AI is about creating a **step-by-step strategy** for achieving a goal. It's like making a **recipe** for a dish, where you need to know the ingredients (actions) and the steps (order of actions) to make it successfully.

2.1 Introduction to Planning

What is Planning in AI?

- **Planning** is the process of **deciding a sequence of actions** to reach a goal.
- It's like a **roadmap** for an AI agent to follow.
- AI uses planning to figure out **what to do next** to solve a problem or complete a task. It's like making a **step-by-step plan** before acting.
- It helps the agent think ahead and act in a **smart and goal-oriented** way.
- Example: A robot planning to go from room A to room B avoiding obstacles.

The Planning Process

1. **Define the goal state** - What should be true when the plan is complete?
2. **Analyze the current state** - What is true now?
3. **Identify available actions** - What can the agent do?
4. **Create a plan** - Find a sequence of actions to transform the current state to the goal state
5. **Execute the plan** - Perform the actions in order
6. **Monitor and replan if necessary** - Adjust if unexpected things happen

Searching vs Planning

> PYQ: What is the difference between searching vs planning? (2024, 1 mark)

Aspect	Search	Planning
Definition	Finding a solution in a space of possibilities	Creating a sequence of actions to achieve a goal
Goal	Find a path or solution	Create a plan of actions
State Space	Represents all possible states	Represents all possible actions and their effects
Search Space	Explores paths to find a solution	Explores actions to create a plan
Example	Finding a route on a map	Planning a trip with stops and activities
Focus	Finding a solution path	Understanding action effects and dependencies
Complexity	Usually simpler, well-defined space	Often more complex with many interacting factors

Why is Planning Important?

- Helps AI to **act intelligently** by considering future consequences.
- Without planning, an AI might take **random actions**.
- Planning ensures actions are **efficient, correct**, and **goal-directed**.
- Useful in robotics, games, assistants, and logistics.
- Enables AI to **solve complex problems** that require multiple steps.
- Allows for **optimizing resources** and managing constraints.

Key Elements of Planning

1. **Initial State -** The starting point of the agent.
2. **Goal State -** The desired condition or result.
3. **Actions -** The steps (possible moves or operations) that change the state.
4. **Plan -** A set of ordered actions to reach the goal.
5. **Action Model -** Defines how actions change the world.
6. **Constraints -** Limitations on possible plans.

Interactive Example: Robot Navigation

A robot needs to move from the bedroom to the kitchen to fetch a glass of water.

- **Initial state**: Robot in bedroom
- **Goal state**: Robot in kitchen with water glass
- **Actions available**:
    - Move(From, To) - Move from one room to another
    - Open(Door) - Open a closed door
    - GrabItem(Item) - Pick up an item
    - FillGlass(Glass, Water) - Fill a glass with water
- **Plan creation**:
    1. Move(Bedroom, Hallway)
    2. Open(KitchenDoor)
    3. Move(Hallway, Kitchen)
    4. GrabItem(Glass)
    5. FillGlass(Glass, Water)
- **Constraints**:
    - Cannot move through closed doors
    - Cannot grab items not in the current room
    - Cannot fill glass unless holding it

Characteristics of Planning Problems

Type	Description	Example
Single-agent	One agent plans alone	Robot vacuum cleaning a room
Multi-agent	Multiple agents plan together	Team of robots in a warehouse
Fully Observable	The agent sees the complete environment	Chess game
Partially Observable	Some parts are hidden or unknown	Exploring an unknown building
Deterministic	Every action has a predictable result	Basic assembly robot
Non-Deterministic	Actions may have different outcomes	Outdoor drone delivery
Static	World doesn't change during planning	Solving a puzzle
Dynamic	The world may change, planning must adapt	Self-driving car in traffic
Discrete	Finite number of states and actions	Chess game
Continuous	Infinite possible states/actions	Robot arm movement

Planning Algorithms

1. **Forward Planning**
    - Starts from initial state and works forward
    - Example: Progression planners
2. **Backward Planning**
    - Starts from goal state and works backward
    - Example: Regression planners
3. **Hierarchical Planning**
    - Breaks complex problems into simpler subproblems
    - Example: Hierarchical Task Network (HTN) planning
4. **Contingency Planning**
    - Plans for different possible outcomes
    - Example: Decision trees in uncertain environments

Real-World Uses of AI Planning

Field	Planning Example	Real-World Application
Robotics	Moving objects, navigating rooms	Amazon warehouse robots
Gaming	Enemy move strategies	AlphaGo, game NPCs
Delivery apps	Route and time planning	UPS delivery optimization
Smart homes	Task scheduling (lights, cleaning)	Google Nest routines
Space Exploration	Mission planning for rovers	NASA Mars rover operations
Manufacturing	Assembly line operations	Toyota production system
Emergency Response	Resource allocation during disasters	Wildfire containment planning

Challenges in AI Planning

- **Computational complexity** - Planning is often NP-hard
- **Uncertainty** - The real world isn't perfectly predictable
- **Dynamic environments** - Things change while planning/executing
- **Partial observability** - Cannot see everything
- **Continuous domains** - Infinite possible states
- **Resource constraints** - Time, energy, and other limitations

2.2 Representation of Planning

> PYQ: How do we represent states, goals and actions in planning? Explain with example. (2021, 8 marks)

What is Planning Representation?

- It means **how the planning problem is described** in a way the AI can understand and solve.

- The AI needs to know:
  - What the **initial situation** is,
  - What the **goal** is,
  - What **actions** are possible,
  - And how each action **changes the state**.

Components of Planning Representation

Component	Description	Example
States	A snapshot of the environment at a given time	"Robot is in room A, door is closed, light is off"
Actions	Steps that can be taken to move from one state to another	"Move(A,B)", "OpenDoor(D)", "TurnOnLight(L)"
Initial State	Where planning begins	"Robot is in Room A, all doors are closed"
Goal State	What we want to achieve	"Robot is in Room C, holding the book"
Transition Model	Describes how actions change the state	"After OpenDoor(D), door D becomes open"
Action Cost	Resources required for each action	"Moving between rooms takes 5 energy units"
Heuristic Function	Estimates cost to reach goal from current state	"Straight-line distance to goal room"

Formal Definition of a Planning Problem

A planning problem can be defined as a tuple (S, A, I, G) where:

- S = set of all possible states
- A = set of all possible actions
- I = initial state ($I \in S$)
- G = goal state or set of goal states ($G \subseteq S$)

Additionally:

- Each action $a \in A$ is defined by:
  - Preconditions: states where a can be applied
  - Effects: how a changes the state

Languages for Planning Representation

1. **STRIPS (Stanford Research Institute Problem Solver)**
   - The original and most influential planning language
   - Uses add and delete lists for effects

2. **ADL (Action Description Language)**
    - Extends STRIPS with more expressive constructs
    - Allows conditional effects and more
3. **PDDL (Planning Domain Definition Language)**
    - Modern standard for planning problems
    - Used in International Planning Competitions
    - Supports various extensions (temporal planning, preferences, etc.)

Action Representation: STRIPS

One of the most popular formats to represent planning actions is **STRIPS**:

- **STRIPS = Stanford Research Institute Problem Solver**

Each action in STRIPS has 3 parts:

1. **Preconditions** – What must be true before the action
2. **Action** – The task to be done
3. **Effects** – What changes after the action

Example of STRIPS

Goal: Robot moves from room A to room B

Action: Move(A, B)
Preconditions: At(Robot, A)
Effects: ¬At(Robot, A), At(Robot, B)

- The robot must be **in A** to move.
- After moving, it's **no longer in A** and is **now in B**.

Planning as State Space Search

- Planning problems can be **converted into a search problem**.
- Each **state** is a node, and **actions** are paths to next states.
- AI searches for the **path from the start state to the goal state**.

Search strategies used in planning:

1. **Forward search (progression)** - Start at the initial state and apply actions
2. **Backward search (regression)** - Start at the goal and work backward
3. **Bidirectional search** - Work from both initial and goal states

Common Representation Techniques

Method	Description	Advantages
State-space representation	Treats planning as a search in a graph of states	Intuitive, direct mapping to search algorithms

Method	Description	Advantages
STRIPS	Defines actions with preconditions and effects	Simple, well-studied, efficient
Planning Graphs	Shows all possible actions at each level	Accelerates planning, provides heuristics
Hierarchical Task Networks	Decomposes tasks into subtasks	Handles complex problems, mirrors human planning
Temporal Planning	Includes time and duration	Models real-world timing constraints
Probabilistic Planning	Includes uncertain outcomes	Handles realistic uncertainty

2.3 Partial-Order Planning

> PYQ: Discuss partial-order plan with example. (2021, 7 marks)
> PYQ: What is Partial-order Planning? Explain in detail. (2023, 15 marks)
> PYQ: Explain partial-order planning with a suitable example. (2024, 15 marks)
> PYQ: Which are the components of the partial-order planning? (2024, 1 mark)

What is Partial-Order Planning?

- Partial-Order Planning (POP) is a planning method where **not all steps** have to be in a **fixed order**. Only those steps that depend on each other are ordered.
- Also called **non-linear planning**.
- It **plans only what's necessary** — the rest can happen in **any order**.
- Also called **"least commitment planning"**, because it avoids making decisions too early.
- Partial-order planning is **flexible** and avoids unnecessary restrictions.

Total-Order vs. Partial-Order Plans

Total-Order Plans	Partial-Order Plans
All actions are arranged in a sequence	Only necessary ordering constraints are specified
Like a to-do list with numbered steps	Like a dependency graph
Less flexible during execution	More flexible, allows parallel execution
Example: "First A, then B, then C, then D"	Example: "A before C, B before D" (A and B can be in any order)
Represented as a sequence	Represented as a directed graph

Why Use Partial-Order Planning?

- More **flexible** than step-by-step planning.
- Useful when **some actions are independent** and can be done in **parallel** or **any order**.
- Helps in situations where **full details** are not known at the beginning.

- **Reduces backtracking** during planning.
- Allows for **resource optimization** and **parallel execution**.
- **Closer to human planning** - we often think in terms of dependencies rather than fixed sequences.

> PYQ: Which are the components of the partial-order planning? (2024, 1 mark)

Components of Partial-Order Planning

1. **Actions** - Steps or operations that change the state of the world
2. **Initial and Goal States** - Start and end conditions of the planning problem
3. **Plan** - A set of actions to achieve a goal
4. **Ordering Constraints (O)** - Rules that say which action should come before another
5. **Causal Links (L)** - Relationships showing that one action produces a condition needed by another
6. **Open Preconditions** - Conditions that are required but not yet satisfied by any action
7. **Threats** - Conflicts where one action might interfere with a causal link between other actions

The core components are the actions, ordering constraints, causal links, and the mechanism to identify and resolve threats through constraint posting.

The POP Algorithm

The basic algorithm for partial-order planning:

1. **Start** with an empty plan containing only Start and Finish actions
2. **Select** an open precondition p from some action a_2
3. **Choose** an action a_1 (existing or new) that achieves p
4. **Add** a causal link $a_1 \rightarrow^p a_2$
5. **Add** an ordering constraint $a_1 < a_2$
6. **Check** for threats and resolve them
7. **Repeat** until no open preconditions remain

How Partial-Order Planning Works:

1. Start with an **empty plan** (just start and goal).
2. Choose a goal condition.
3. Add actions that achieve that condition.
4. Add **ordering constraints** only when needed.
5. Repeat until all goals are achieved.
6. **Resolve threats** by adding ordering constraints or new actions.

Example of Partial-Order Planning

Let's say we want to create a simple plan for a robot to make tea:

1. **Initial State**: Robot is in the kitchen, no tea made.
2. **Goal State**: Robot has made tea.
3. **Actions**:
    - BoilWater
    - AddTea

    - PourWater
    - StirTea
4. **Causal Links**:
    - BoilWater → AddTea (need hot water to add tea)
    - PourWater → StirTea (need to stir after pouring)
5. **Ordering Constraints**:
    - BoilWater < PourWater (water must be boiled before pouring)
    - PourWater < StirTea (pouring must happen before stirring)
6. **Open Preconditions**:
    - AddTea (needs BoilWater)
    - StirTea (needs PourWater)
7. **Final Plan**:

```
1. BoilWater
2. PourWater
3. AddTea
4. StirTea
```

The robot can execute these actions in parallel where possible, like boiling water while preparing the tea.

Advantages of POP

- **More efficient** in many cases.
- Allows for **parallel execution** of tasks.
- Delays decisions to keep the plan flexible.
- Can handle **uncertainty** and **dynamic changes** better.
- **Less backtracking** compared to total-order planning.

Disadvantages of POP

- More **complex** to implement than total-order planning.
- Requires more **memory** to store the partial-order graph.
- Can be **harder to understand** and visualize.
- May require **more sophisticated algorithms** to resolve conflicts and threats.
