A.I

Date / /

1.

- Introduction to AI.

1. NLP: This components enables the computers to understand human languages like English & interact effectively.

2. Knowledge Representat°: unit helps in storing info gathered from various sources like sensors, NLP.

- What is PEAS Descriptor:
PEAS stands for performance Measure, Environment, Actuators & sensors. It's used to define tasks environment of an intelligent agent.
  - Performance measure: criteria to evaluate how successfully an agent is performing.
  - Environment: surrounding in which agent operates.
  - Actuators: Agent's output devices used to interact with environment.
  - Sensors: agent's input devices used to observe environment.

- Applications of AI.

1) Education: AI is used to training simulators to give real-world practice, in smart tutors that help students learn. For small kids, AI makes learning fun through games & app

2) Entertainment: is used in movies for special effects & animations.

3. Medical: AI assists doctors in fields like cardiology & neurology. helps perform complex surgery using robotic arms.

4. Military: used for training & real combat mission. It can quickly analyze large amount of battlefield data.

. Category of A.I: It is based on capability & functionality

1) Based on capability:
   1) Narrow AI (weak AI):
      . Design to perform a single specific task
      . can't perform tasks outside its programming.
      . eg: Siri, Alexa,
   2) General AI (strong AI):
      . Has ability to learn, understand & perform task like human.
      . Still under development & research.
   3) super AI:
      . Hypothetical AI that surpasses human intelligence.
      . can feel emotions & make better decisions than us
      . It doesn't exist yet & is a future scope

Q. 2.

- Planning in AI: It is proces of deciding a sequence of actions that an intelligent action must take to achieve a specific goal. It is like thinking ahead — "what todo, when, howo todo".

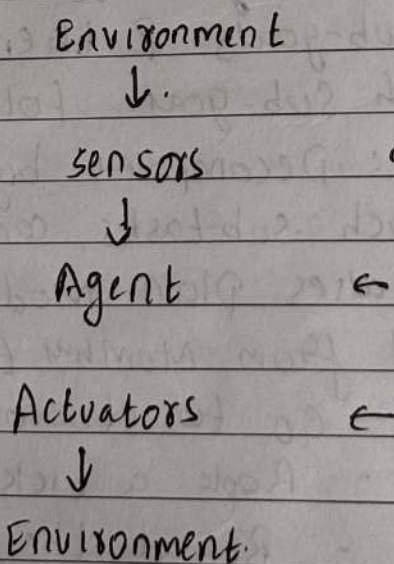
- Partial order Planning: is a technique where only necessary order of actions is maintained — not all orders are in strict sequence.
  Features: • only order actions that are logically dependent.
  • give more flexibility to agent.
  • Reduces unnecessary agent.
  Eg: Goal: cook vegetable:
  steps: • chop vegetable • turn on gas • cook veg.
  "chop veg" must come before "cook veg", but turn on gas can happen before or after chopping.

- Hierarchical Planning: It breaks dow complex goal into sub-goals & then plans how to achieve each sub-goal. follows top-down approach.
  • features: • Decomposes big tasks into simpler tasks.
  • Each sub-tasks can have its own plan.
  • Makes plan modular & reusable.
  Eg: Travel from Mombay to Delhi
  steps: • Go to station
  - Book a ticket.
  - Board the train
  - Reach Delhi station.
  each sub-goals handle seprately

- Wumpus world Environment: It is 4x4 grid based environment used to study intelligent agent behaviour. Agent goal is to find gold & exit safely while avoiding deadly pits & wumpus (monster). world is partially observable, meaning agent can see whole env at once.
  - Agent uses sensors to detect:
    - Breeze near pits
    - Glitter near gold.
    - Bump when hitting wall.
    - Scream when wumpus is killed.

- Agent: It is anything that can perceive its environment through sensors & act upon that environment through actuators.

* Block Diagram of simple agent:



- Types of Agent:
  1. Simple reflex agent:) use condition-action rules
     - responds directly to percepts.

Diagram: Percept → [cond"-act"-rule] ⟷ Action.

Characters: Fast, No learning, no history

Date / /

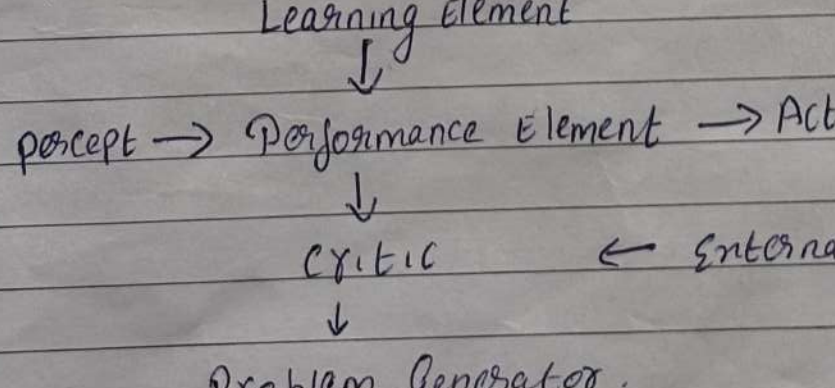
2. Model Based Agent: • Maintain an internal state based on percept history.
• More intelligent than simple reflex Agent.
Diagram: Percept → update-state → cond^n-act^n-rule → Action.
characteristics: keep track of world state.

3. Goal Based agent: • Has a goal to achieve.
• Makes decision based on desired outcomes.
Diagram: percept → update state → Goal info → search/plan → Action.
chas: More flexible, can choose blw multiple opt^n.

4. Utility Based agent: • Maximize expected utility.
• choose action based on utility funct^n.
• Diagram: Percept → state update → utility evaluat^n → Act^n selection → Action.
• char: Handle conflicting goals, choose best option.

5. Learning Agent: improves performance over time based on experience.
• components: 1) learning Element: makes improvement.
2) Critic: gives feedback based on performance.
3) Performance element: choose external action.
4) problem Generator: Suggest exploratory actions

Diagram:



Characteristics: • Continuously improves
• Adapts to new environment.

Date / /

4.

. Forward chaining: It is a data driven reasoning technique. In this technique, we start from known facts & apply rules to reach new conclusion.

+ works: 1) start with known facts
2) check which rule can be applied based on those facts
3) when rule matches, apply it to derive new facts.
4) Repeat this process until goal reached or no more new facts can be derived.

Eg: If someone is human, they are mortal.
fact: xyz is a human.
conclusion: xyz is mortal.

. Backward chaining: It is a goal driven reasoning technique. Here, we start with a goal & then check what facts & rules are required to prove it.

+ works: 1) Begin with a goal.
2) Look for rules that can help prove the goal
3) For each rule, try to prove the conditions
4) This process continues until we reach known facts.

Eg: Goal: is xyz mortal?
Rule: If someone is human, they are mortal?
fact: xyz is a human
conc: xyz is a mortal.

DOMS | Page No.
Date / /

Knowledge Representation

1) KB Agent (Knowledge Based): It is an intelligent System that can make decision based on stored knowledge & logical reasoning.
components:
1. KB: stores facts & rules
2. interference Engine: APPlies logic to KB to derive new knowledge.
3. percept History: stores seq. of observed input.
4. Actuators: performs action based on decision.

2) Propotional logic: This represents knowledge using simple declarative statements that are either True or false.
~~compone~~ concept:
- Propostn: A basic statement like it is raining → P
- Connectives: And (∧), or (∨), implies (→).

Eg: P = It is raining
q = I carry umbrella Rule: P → q.

3) FOL: It is more powerful than propositional logic. It can represent relationship b/w objects.

Syntax:
- constants: Ram, Delhi.
- predicates: Human(x); GreaterThan(x,y)
- Quantifiers: ∀ (universal): ∀x Human(x) → Mortal(x)

DOMS | Page No.
Date / /

Diff

Aspect	PL	FOL
Basic unit.	propositions	predicates with objects, variables & quantifiers.
Expressivness	less Expressive	More expressive
variables	Not supported	supported eg x, y
Quantifiers	Not available	available ∀, ∃
Complexity	simpler & easy to compute	More comple but powerful

+ Resolution Tree

. Convert to FoL.

All People who are graduating are happy
$\forall x$ (Graduating(x) → happy(x))

All happy people smile
$\forall x$ (happy(x) → Smile(x))

Someone is Graduating
$\exists x$ (Graduating(x))

We need to Prove that is someone Smiling?
$\neg \exists x$ Smile(x)

DOMS | Page No.
Date / /

- convert FOL to CNF

Eliminate implicat^n $\alpha \to \beta = \neg\alpha \vee \beta$.

$\forall x [\neg Graduating(x) \vee Happy(x)]$
$\forall x [\neg happy(x) \vee Smile(x)]$
$\exists x (Graduating(x))$
$\neg \exists x \; smile(x)$

- Standardize Variable Apart. take diff. variables.

$\forall x [\neg Graduating(x) \vee Happy(x)]$
$\forall y [\neg happy(y) \vee Smile(y)]$
$\exists z \; Graduating(z)$
$\neg \exists w \; smile(w)$.

- Move Negation inwards.

$\forall x [\neg Graduating(x) \vee Happy(x)]$.
$\forall y [\neg happy(y) \vee Smile(y)]$
$\exists z \; Graduating(z)$.
$\forall w \; \neg smile(w)$.

- Skolemization:

$\forall x [\neg Graduating(x) \vee Happy(x)]$
$\forall y [\neg happy(y) \vee Smile(y)]$
$Graduating(A)$.
$\forall w \; \neg smile(w)$.

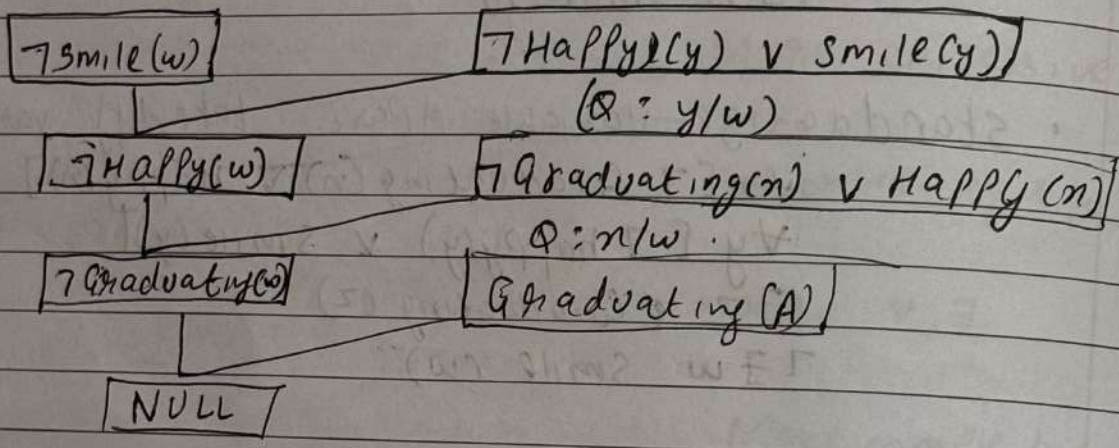
- Drop universal quantifier:

$\neg Graduating(x) \vee Happy(x)$
$\neg happy(y) \vee Smile(y)$
$Graduating(A)$
$\neg smile(w)$.

Now ~~so~~ ~~for~~ sentences are in CNF

+ Resolution tree:
- If Fact 'f' is to be proved than it starts If
- It contradicts all other rule in KB
- The process stop when it return NULL Clause.

Resolution Tree.



+ Belief Network:
- It is a graphical model that represents probabilistic relation among variables.
- Each node = a random variable (eg: disease, sym)
- Each edge (arrow) = a dependency (cause-effect).
- Based on probability rules.
- Each node has conditional probability based on its parent

Eg: If a person has flu, they are likely to have fever or cough.
- Flu is cause parent
- fever, cough are effect (children)

5.

Planning: Book M2.

State Space Representation: This is a way to model a planning problems in terms of:
- State: configuration of system at a point in time. eg: where is the robot now
- Initial state: where planning starts
- Goal state: Desired outcome
- Actions: Transit^{n} move from 1 state to another

+ State space is like a graph:
  - node = states
  - Edges = Actions that cause transitions

Eg: If you want to move a robot from room A to C
- state: Room A, B, C
- ~~Goal~~ initial: Room A
  Goal: Room C
  Action: Move from A to B then B to C.
  A → B → C.

- Necessity of state representation.
  1. Provides structure to problem.
  2. Allow search Algorithms like BFS, A^{+} etc
  3. Enables goal-driven behaviour.

+ Air Cargo transport Problem:

→ 1. Initial state: · cargo1 & Plane1 are at Mumbai
· cargo 2 & Plane 2 are at Delhi

2. Goal state: cargo 1 ~~&~~ ~~Plane1~~ should be at Delhi.
cargo 2 should be at Mumbai.

Available Actions:

1. load(cargo, Plane, Airport): load cargo onto plane at Airport
2. unload(" " "): unload cargo from plane at Airport
3. Fly(Plane, from, to): Fly a plane from 1 airport to another

Action: Move cargo1 from Mumbai to Delhi ; Move cargo 2 from Delhi to Mumbai.

	Plane2.
1. Load(Plane1, cargo1, Mumbai)	1. load(?cargo2, Delhi)
2. ~~UA~~Fly(Plane1, Mumbai→Delhi)	2. Fly(Pane 2, Delhi→Mumbai)
3. unload(Plane1, cargo1, Delhi)	3. unload(Plane2, cargo2, Mumbai)

+ ~~St~~ STRIP

Same as 1, 2 in 3rd.

operators (Actions), each having:

- Preconditions, Add effect, delete effect

initial state:

- At (cargo1, SFO), At(plane1, SFO)
- At (cargo2, JFK), At(Plane2, JFK).

Goal: At(cargo2, SFO)
At (cargo1, JFK).

Page No.
Date / /

Operations (Act^ns in STRIPS):

1. load (cargo, plane, Airport)
   - precondition : At(cargo, Airport) ^ (Plane, Airport)
   - Add effect : ~~At~~ In(cargo, plane).
   - Del effect : At (cargo, Airport).

2. unload (,, ,, ,,)
   - preconditions: ,, ,, ,,
   - Add effect = At (cargo, Airport)
   - Del effect: ~~At~~ In (cargo, Plane)

3 Fly (~~A~~ plane, from, to).
   - ,, At(Plane, from)
   - Add eff : At (plane, to)
   - Del : At (plane, from)

Final:

1. load (cargo1, plane1, SFO)
2. Fly (Plane1, SFO, JFK)
3. Unload (cargo1, plane1, JFK

DOMS | Page No. | Date / /

NLP.

It is a tangu field of AI that enables computer to understand, interpret & generate human language. It's use in application like chatbot, translat", etc.

Language models in NLP: It help in predicting & generating text by learning lang patterns.
They are classified as:

1. N-gram models: Statistical models that predict next word based on previous n words.

2. Hidden Markov Models: Probablistic model used for tasks like part-of-speech tagging & speech recognition by considering hidden states behind words.

3. Neural Networks-Based Models: used Deep learning to capture lang content.
   - RNN: Handle sequence but struggle with long-term cont
   - LSTM: Improved RNNs that manage longer dependencies.
   - Transformers: use self-attent" for powerful lang understnd across long text like GPT.

- Application of AI in Robotics

AI enhance intelligence & decision-making capabilities of Robots. Some key app" areas:

1. Computer vision: For object detect", recognizat" & navigation.
2. Motion Planning: Robots learn to move efficiently without collision
3. NLP: Enables robots to understand & respond humans commands
4. Autonomous ~~ve~~ Navigat": Self driving vehicles & drones.
5. ML: For adapting behaviours based on experience
6. Human Robot interact": Social robots that interacts naturally with People.

DOMS | Page No.
Date / /

- Supervised learning: This is a machine learning approach where model is trained on labelled data, i.e each input has a known output.
  - used for: classification & regression tasks.
  - eg: spam email classifier trained using email labelled as "spam" or "not"

unsupervised learning: It deals with unlabelled data. Model finds hidden patterns in data without any predefined output.
- used for: clustering
  Eg: Customer segmentation in marketing where customers are grouped based on purchased behaviour.

Reinforcement learning: This involves an agent interacts with environment, learning best actions through rewards and penalties.
used for: gaming, self-driving cars.