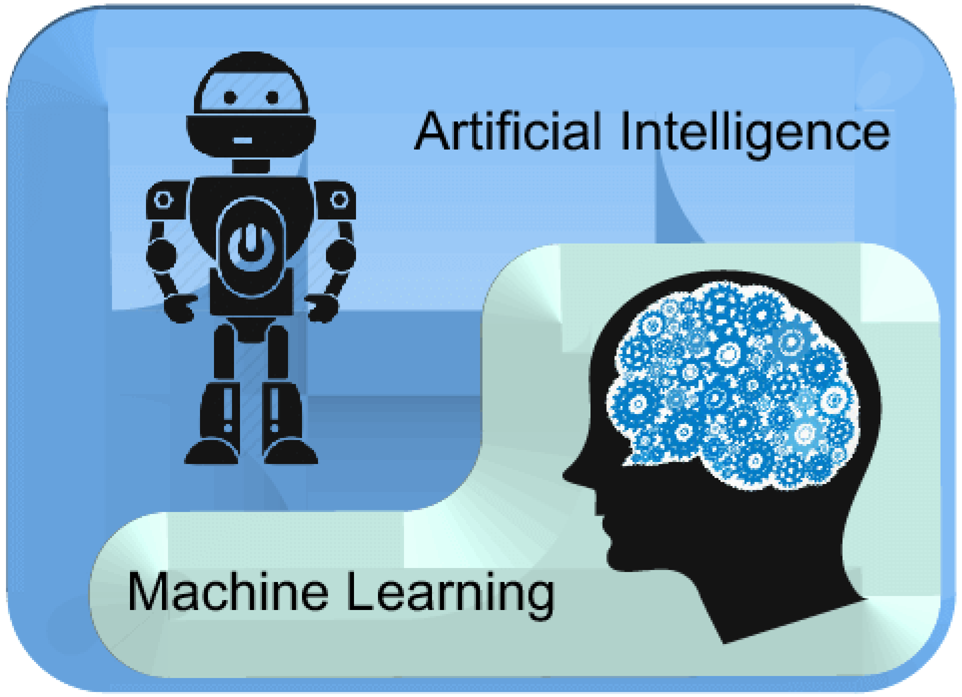


BECE309L-Artificial Intelligence and Machine Learning

Sample Use-case based Questions

Module 1: Foundations of AI

- Introduction
- Agents and rationality
- Task environment
- Agent Architecture Types

Characteristic of Agent

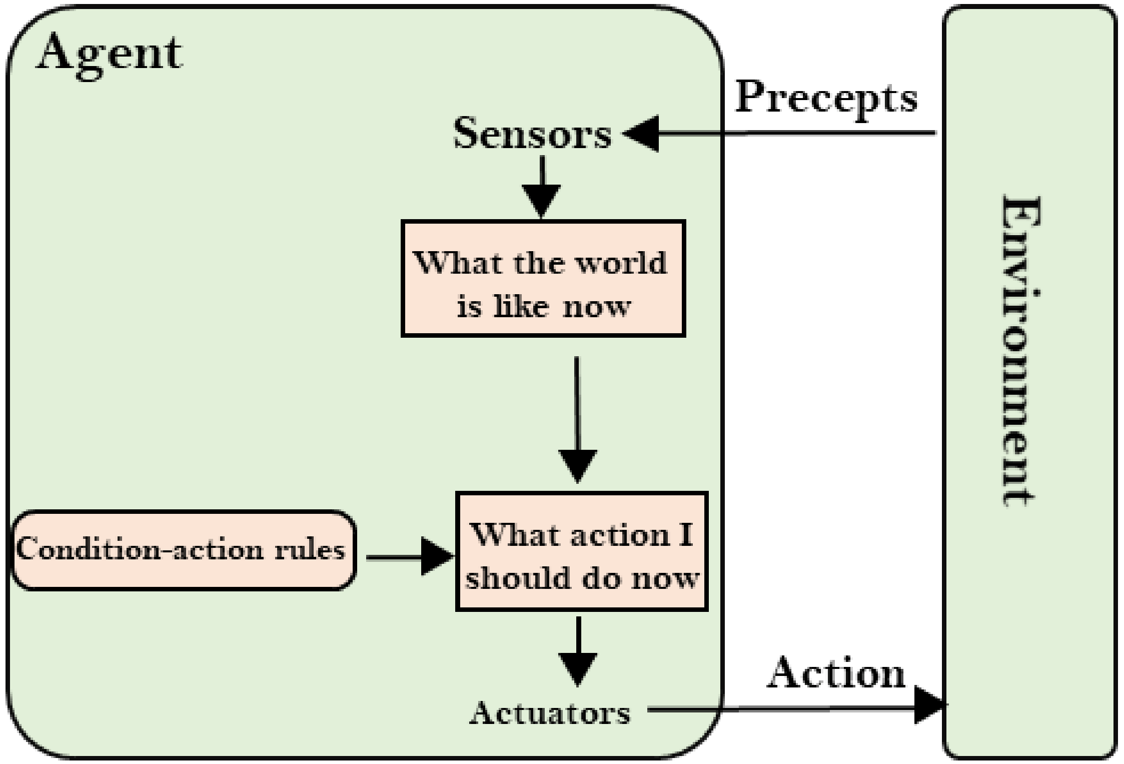
AI-Driven Personal Finance Assistant

- **The AI-driven personal finance assistant helps users manage their budgets, track expenses, and achieve their financial goals. It integrates with the user's bank accounts, credit cards, and investment portfolios to provide comprehensive financial advice and insights.**
- **Autonomy:** An AI virtual agent is capable of performing tasks independently without requiring constant human intervention or input.
  - The personal finance assistant autonomously categorizes transactions into predefined categories (e.g., groceries, entertainment, utilities) based on its algorithms and learned patterns. It generates monthly budget reports and financial summaries without requiring user input each time.
- **Perception:** The agent function senses and interprets the environment they operate in through various sensors, such as cameras or microphones.
  - **The assistant adapts to changes in the user's financial situation, such as changes in income, new spending habits, or shifts in financial goals**.
- **Reactivity:** An AI agent can assess the environment and respond accordingly to achieve its goals.
  - The **assistant sends real-time notifications** if a **transaction exceeds** a **user-defined budget limit** or if **there are unusual activities in the user's accounts**.
  - If the user's **credit card is charged for a large amount**, the assistant reacts by **sending an alert to ensure the transaction** is **recognized and authorized** by the user.
- **Reasoning and decision-making:** AI agents are intelligent tools that can analyze data and make decisions to achieve goals. They use reasoning techniques and algorithms to process information and take appropriate actions.
  - The assistant **proactively provides financial advice**, such as **suggesting ways** to **save more money** or **recommending adjustments** to the **budget before the user reaches a spending limit**. It might identify upcoming large expenses (e.g., annual subscriptions, recurring bills) and alert the user in advance, offering suggestions on how to plan for these expenses effectively.

Types of Agent

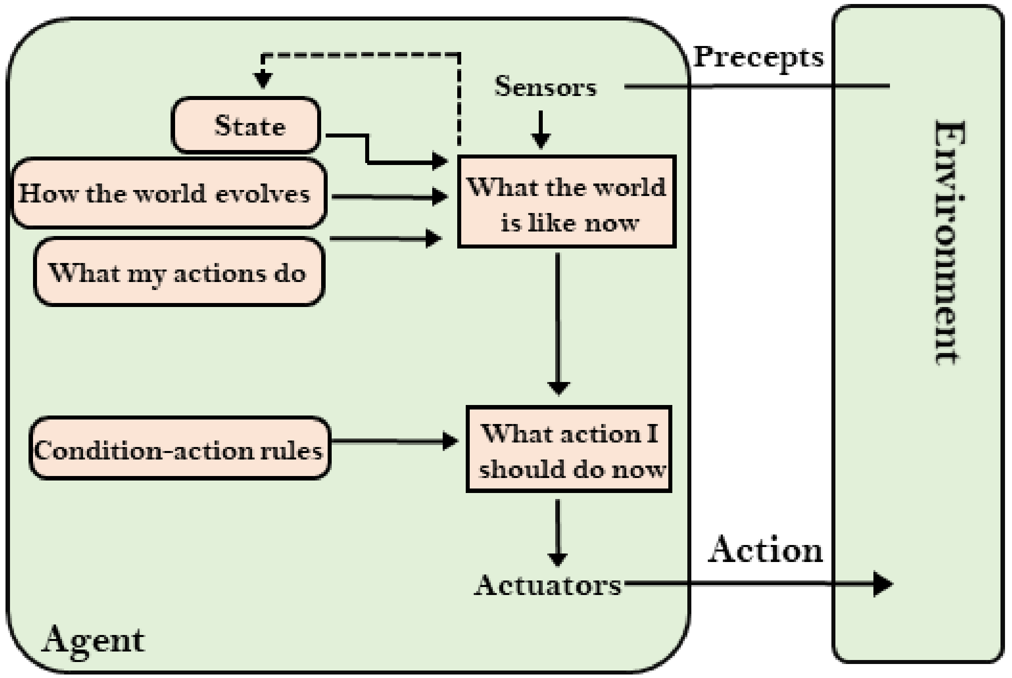
1. Simple Reflex Agent

- **Use Case: Basic Home Security System**
- **Description:** If a motion sensor in the front yard detects movement, the system turns on the exterior lights to illuminate the area. Once no further motion is detected, the lights turn off automatically after a set period.
- **Explanations:**
- **Sensors:** The system is equipped with motion detectors that monitor for any movement in designated areas (e.g., entrance or backyard).
- **Rule-Based Actions:** The **agent operates on a set of predefined rules**. For instance, if **the motion sensor detects movement during the night**, the agent **immediately turns on the lights in that area**.
- **Immediate Response:** The agent **reacts directly to sensor inputs without considering the broader context** or maintaining an internal model of the environment.



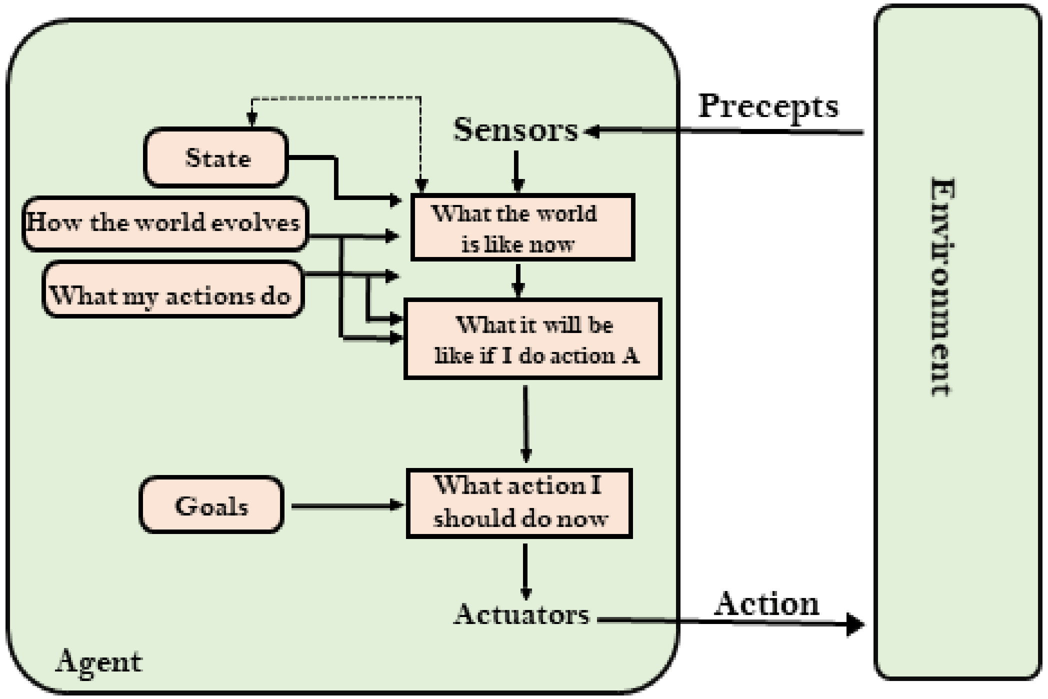
Types of Agent

- **2. Model-Based Reflex Agent**
- **Use Case: Intelligent Thermostat**
- **Description:** If the thermostat detects that the temperature has dropped below the desired level and the temperature has been falling steadily for a while, it decides to turn on the heater to reach the target temperature gradually.
- **Explanations:**
- **Internal Model:** The **thermostat maintains an internal model** of the **home's temperature history** and **current status**.
- **Decision Making:** It **uses this model** to **decide when to adjust the heating or cooling system**. For example, if the internal temperature has been consistently below the set point, it activates the heater even if no immediate sensor input is detected.
- **Context Awareness:** The agent **incorporates historical data to make better decisions**, such as preventing the heating system from turning on and off frequently.



Types of Agent

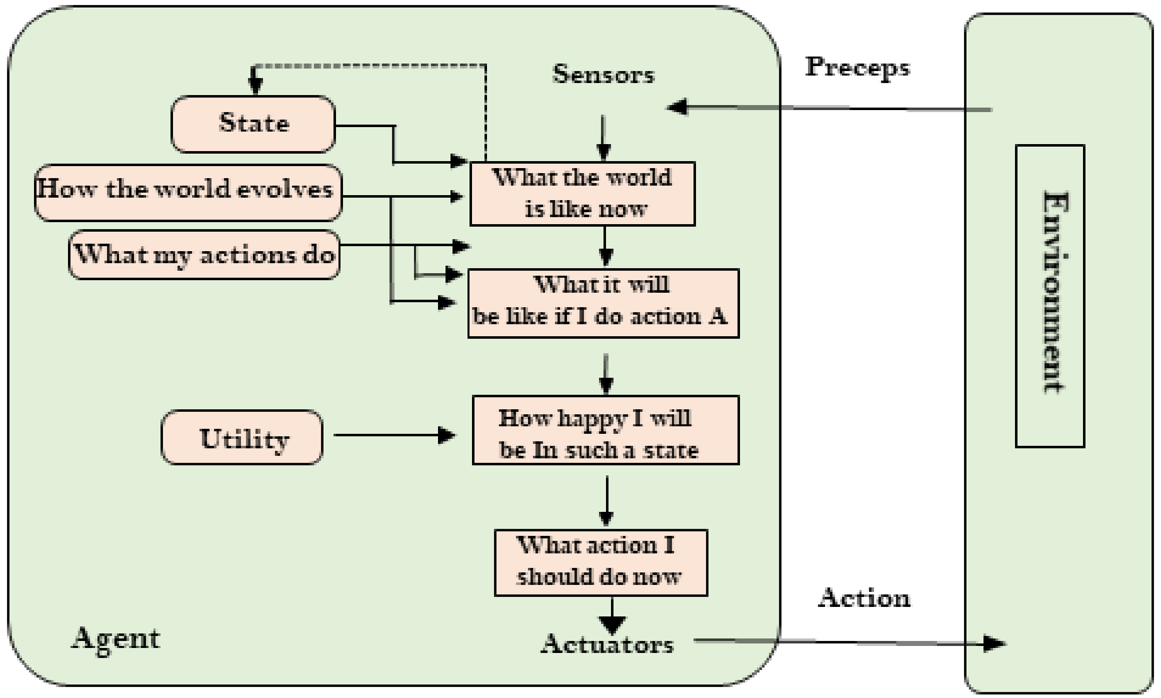
- **3. Goal-Based Agent**
- **Use Case: Autonomous Navigation System**
- **Description:** An autonomous navigation system in a self-driving car uses a goal-based agent to plan and execute a route to reach a destination.
- **Explanations:**
- **Goals:** The agent's primary goal is to **navigate from the car's current location to a specified destination** while **adhering to traffic rules and avoiding obstacles**.
- **Planning:** It assesses the **current environment and plans a route** that considers **traffic conditions, road closures, and navigation maps**.
- **Execution:** The agent **continually updates its plan and actions based on its progress** towards the goal and any changes in the environment



When a user sets a destination in the car's navigation system, the agent calculates the optimal route, taking into account current traffic conditions and roadblocks, and then drives the car accordingly, adjusting its route in real-time as needed.

Types of Agent

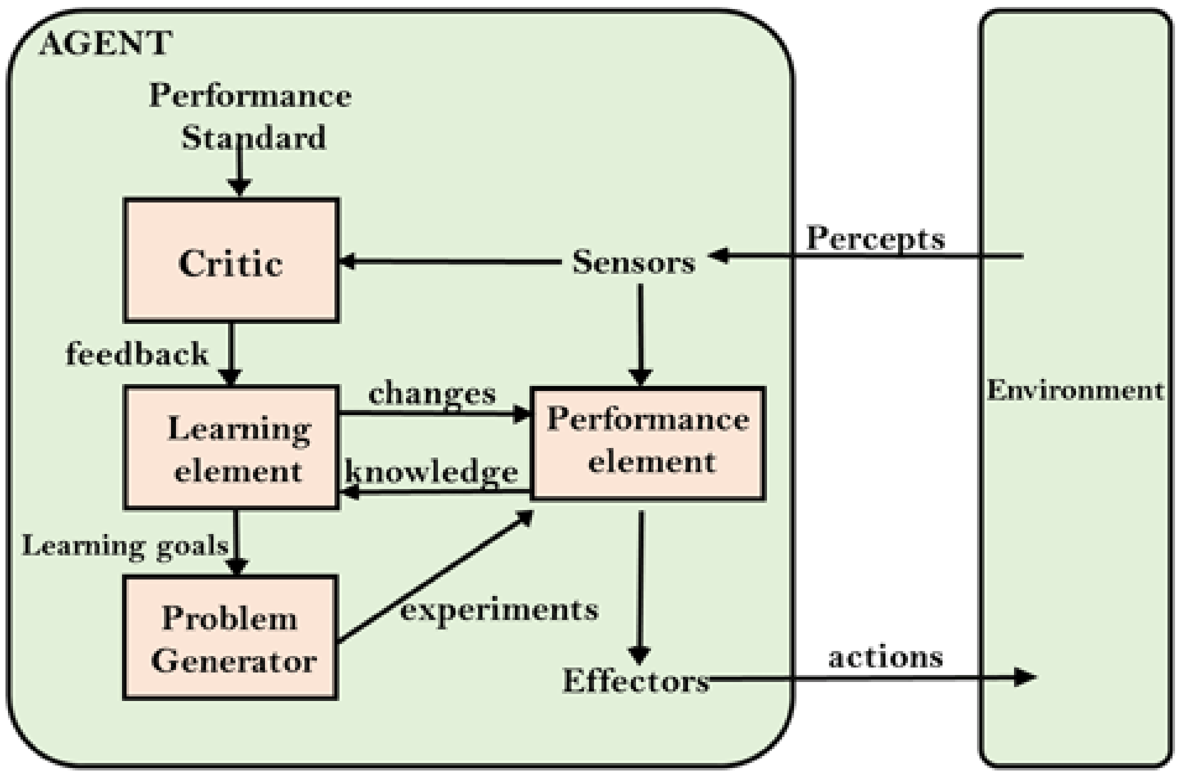
- **4. Utility-Based Agent**
- **Use Case: Smart Personal Finance Advisor**
- **Description:** A smart personal finance advisor uses a utility-based agent to help users make optimal financial decisions based on their preferences and goals.
- **Explanations:**
- **Utility Function:** The **agent evaluates different financial decisions** (e.g., saving options, investment strategies) based on a **utility function that quantifies the user's preferences for risk, returns, and liquidity**.
- **Decision Making:** It provides recommendations that **maximize the user's overall financial utility**, considering factors such as expected returns, safety, and personal financial goals.
- **Adaptation:** The agent adapts its recommendations as the user's financial situation and preferences change.


Sensors
Preceps
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Utility
How happy I will be In such a state
What action I should do now
Environment
Action
Agent
Actuators

If a **user is deciding between different investment options**, the **advisor evaluates each option based on potential returns, risk tolerance**, and liquidity needs, and recommends the investment that best aligns with the user's preferences and financial goals.

Types of Agent

- **5. Learning Agent**
- **Use Case: Personalized E-Learning Platform**
- **Description:** A personalized e-learning platform uses a learning agent to adapt educational content to the needs and preferences of individual students over time.
- **Explanations:**
- **Data Collection:** The agent **collects data on student interactions, performance on quizzes, engagement with content, and feedback**.
- **Learning Algorithms:** It uses machine learning algorithms to **analyze this data and identify patterns in how students learn best**.
- **Adaptation:** The *agent continuously updates* its *recommendations* and *adjusts the difficulty* and type of content based on the student's progress and learning style.



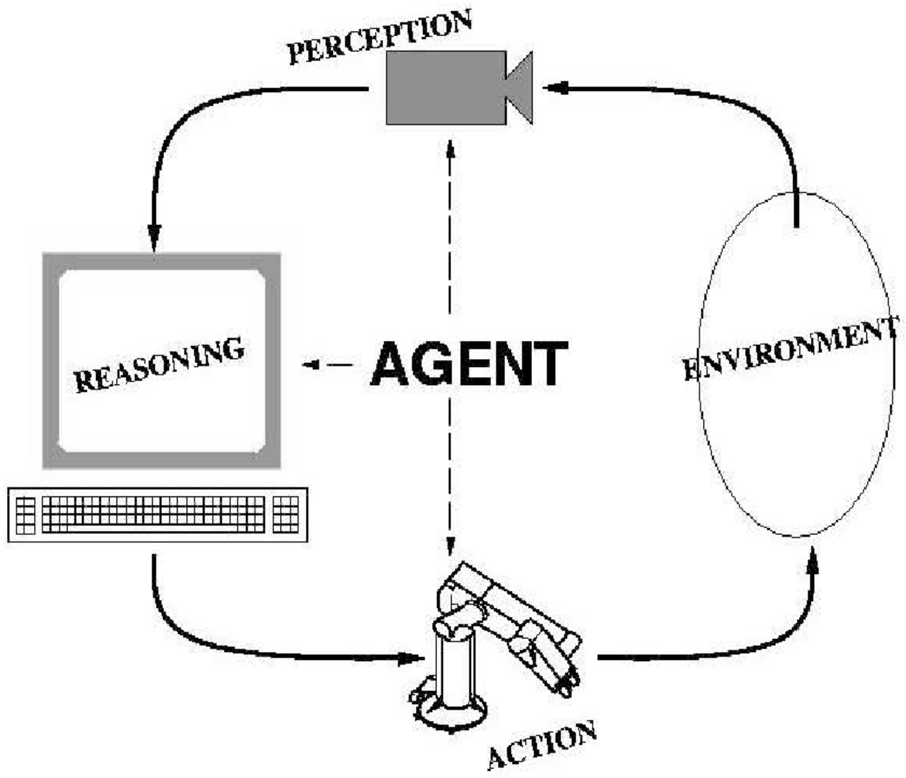
As a student interacts with the e-learning platform, the agent learns which topics the student struggles with and adjusts the content delivery, provides additional practice problems, and suggests supplementary resources to improve the student's learning experience.

Rationality based Agents

Smart Energy Management System for a Commercial Building

- **Uses a rationality-based agent to optimize energy consumption while maintaining comfort for occupants and minimizing operational costs.**

- **Utility Function:** The agent evaluates different actions based on a utility function that considers **energy costs, occupancy patterns**, and **temperature preferences**.
- **Perception and Data Collection:**
- **Historical Data from Sensors:** The system is equipped with various sensorscollects and analyzes historical data on energy usage, occupancy patterns, and external weather conditions.
- **Decision-Making Process:**
- **Optimization Algorithms:** The agent uses optimization algorithms to **determine the best combination of actions to achieve its goals** by considering factors like **the current weather forecast, occupancy schedules, and energy pricing**.
- **Action Selection:** Based on its analysis, the **agent decides how** to **adjust heating**, **cooling**, and **lighting systems**.



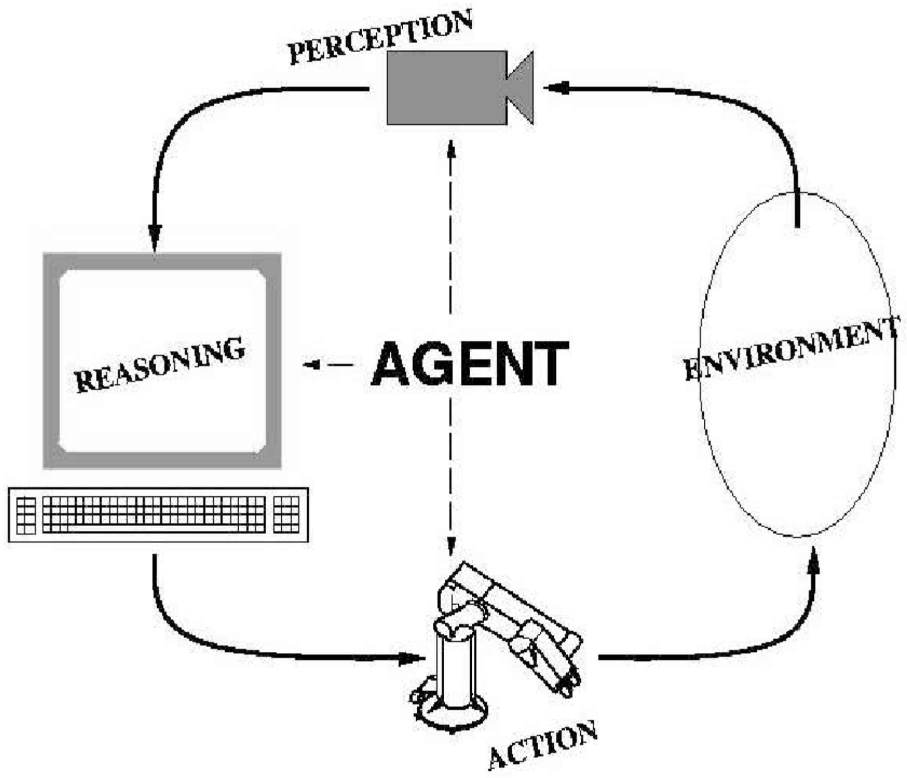
Rationality based Agents

Smart Energy Management System for a Commercial Building

- **Uses a rationality-based agent to optimize energy consumption while maintaining comfort for occupants and minimizing operational costs.**

Adaptation and Learning:

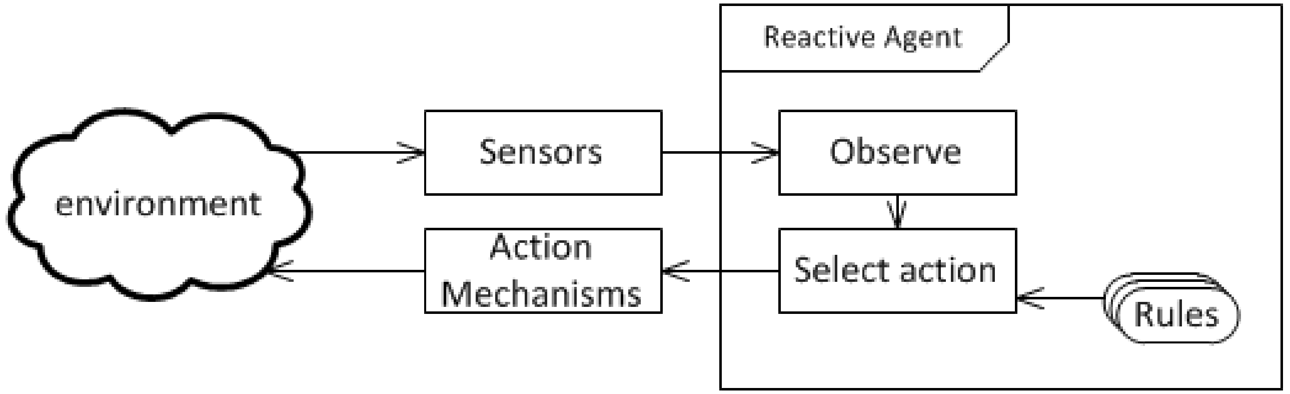
- **Dynamic Adjustments:** The agent **continually adjusts its strategy** based on **real-time data**. If **unexpected changes in occupancy or weather occur**, it **revises its plans to ensure optimal performance**.
- **Learning from Feedback:** The agent learns from its actions and outcomes to improve its **decision-making over time**. For instance, if it notices that certain temperature settings consistently lead to higher comfort levels or lower energy costs, it updates its preferences and recommendations accordingly.



Reactive Agent Architecture

Smart Home Climate Control

- **Description:** In a smart home, a reactive agent is responsible for managing the climate control system. The agent is equipped with sensors that detect the temperature and occupancy of different rooms. Based on predefined rules, the agent reacts to changes in these parameters.
- **Explanations:**
- **Sensor Data:** The agent receives real-time temperature readings and occupancy information from sensors.
- **Rule-Based Actions:** If the temperature in a room exceeds a certain threshold or if the room is occupied, the agent turns on the air conditioning or heating system.
- **Immediate Response:** The agent does not plan or deliberate but reacts instantly to sensor inputs, ensuring that the home remains comfortable at all times.

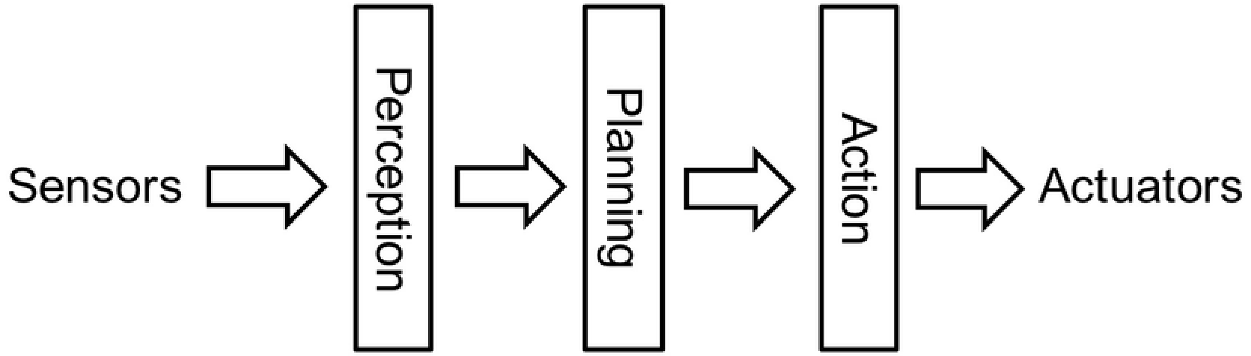


- Reactive agent architecture is based on the direct mapping of situation to action, and it is different from the logic-based architecture where no central symbolic world model and complex symbolic reasoning are used.

Deliberative Agent Architecture

Autonomous Delivery Robot

- **Description:** An autonomous delivery robot uses a deliberative agent architecture to navigate a complex environment like a university campus. The robot needs to deliver packages from one building to another, navigating through hallways, elevators, and outdoor paths.
- **Explanations:**
- **Internal Model:** The robot maintains a detailed model of the campus, including maps, obstacles, and known locations of buildings.
- **Planning:** It uses this model to plan the best route for each delivery, taking into account obstacles, traffic, and the robot's current location.
- **Reasoning:** The robot reasons about possible routes, updates its plan based on real-time changes (e.g., blocked paths), and makes decisions to optimize delivery time.

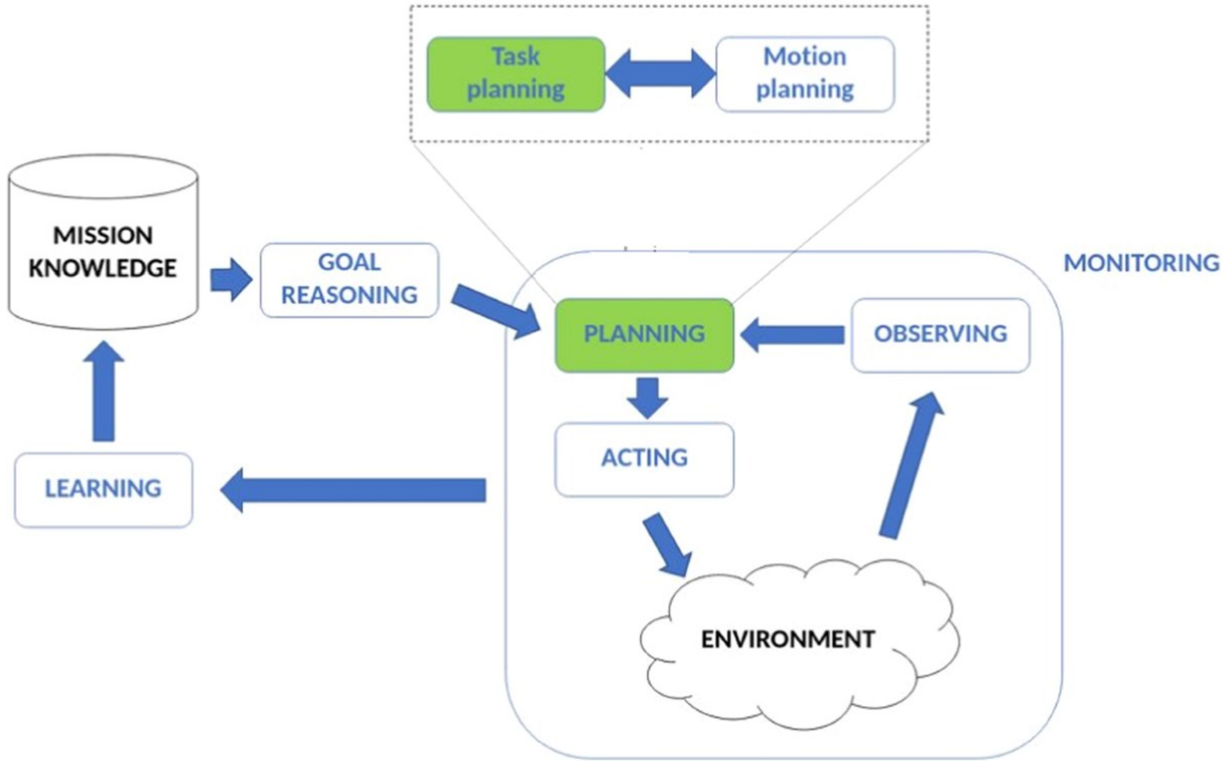


- They require symbolic representation with compositional semantics in all major functions, as their deliberation is not limited to present facts, but construes hypotheses about possible future states and potentially also holds information about memory.

Hybrid Agent Architecture

Customer Service Virtual Assistant

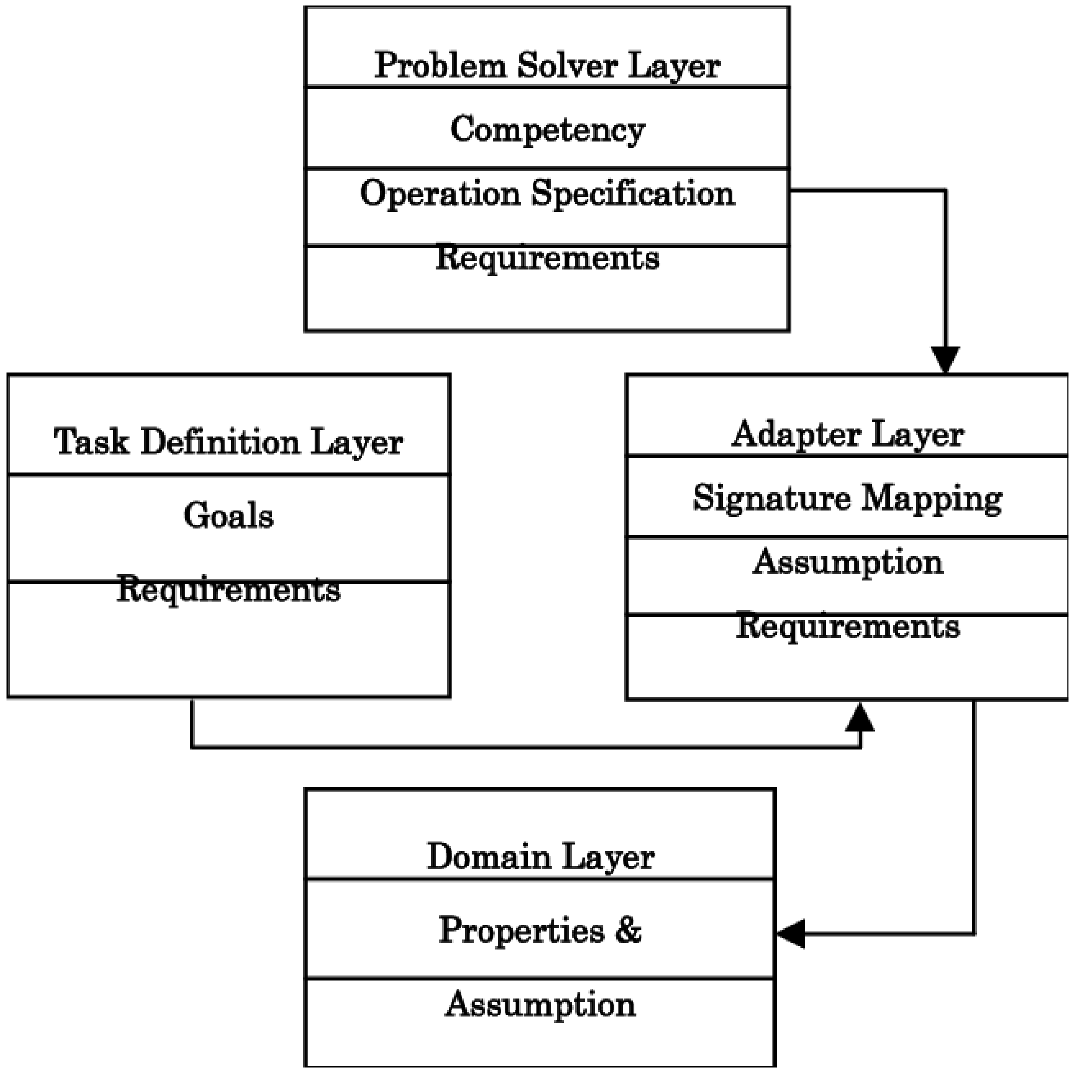
- **Description:** A customer service virtual assistant employs a **hybrid agent architecture** to **interact with customers on an e-commerce platform**. It combines reactive and deliberative behaviors to handle a wide range of customer inquiries.
- **Explanations:**
- **Reactive Component:** The assistant uses ***reactive rules to handle common requests*** like ***checking order status*** or ***updating account information*** immediately based on keywords or commands.
- **Deliberative Component:** For more **complex issues**, such as **troubleshooting a problem** with a product, the assistant uses deliberative reasoning to guide the user through troubleshooting steps, escalate issues if necessary, and offer personalized solutions based on past interactions.



Layered Agent Architecture

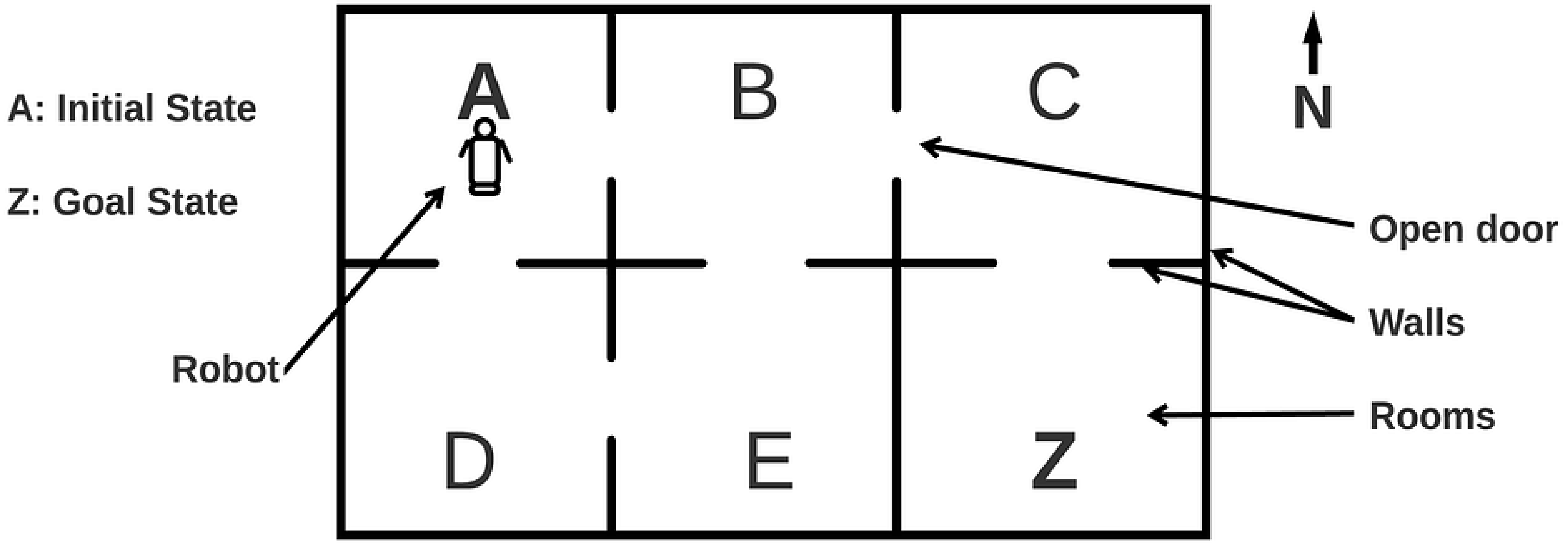
Smart City Traffic Management

- **Description:** In a smart city, a multi-agent system manages traffic flow to reduce congestion and improve transportation efficiency. The system involves multiple agents, each responsible for different aspects of traffic control.
- **Explanations:**
- **Traffic Signal Agents:** Each traffic light operates as an agent that adjusts its signal timing based on real-time traffic data and coordination with neighboring signals.
- **Vehicle Agents:** Agents represent vehicles, providing data about their location and destination to the system.
- **Coordination:** The agents **communicate and collaborate to optimize traffic flow, reduce bottlenecks, and ensure smooth transit throughout the city**. They can adjust traffic light timings dynamically based on current traffic conditions and vehicle data.



Module 2: Problem-solving by Searching

- Search Space –
- Search algorithms, strategies
- Search in complex environments



Source: https://medium.com/nerd-for-tech/ai-search-algorithms-with-examples-54772c6d973a

Search-Algorithms and Strategies

Uninformed (Blind) Search Algorithm

Use Case:

Job Scheduling with Deadline Constraints

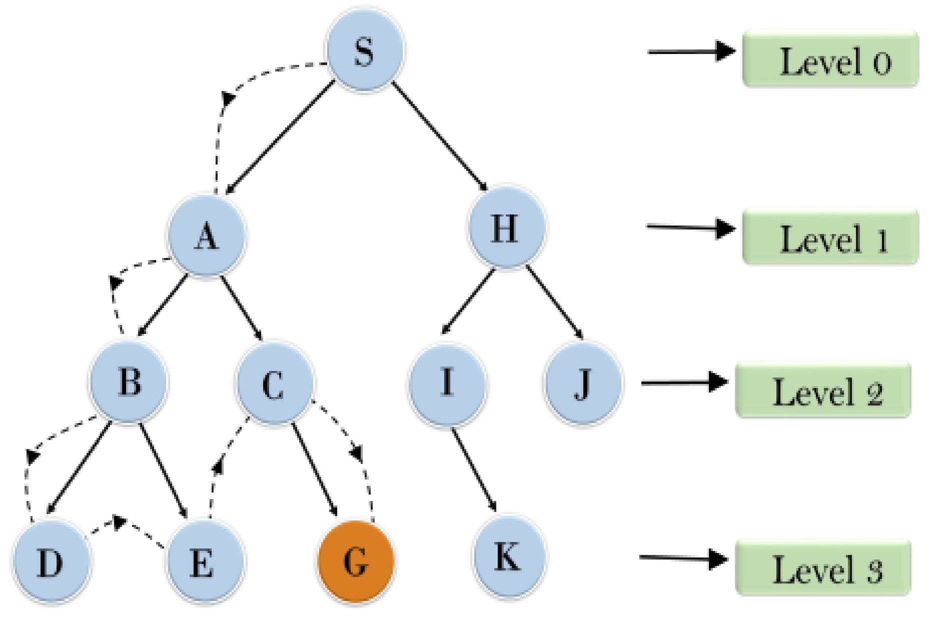
Description: A job scheduling system uses an uninformed search algorithm to schedule tasks that need to be completed before their respective deadlines. **Uninformed Search Algorithm: Depth-First Search (DFS)**

Explanations:

•**Initial State:** The **system starts with an empty schedule and a list of tasks, each with a deadline and a required duration**.

•**Search Strategy:** DFS explores **possible schedules by placing each task into the schedule** and **proceeding to the next task**. It does so by **going deep into one branch of the schedule before backtracking** to try other possibilities.

•**Stack-Based Exploration:** The system uses a stack to keep track of tasks being scheduled and their order.



Steps:

1.Initialization: Start by placing the first task into the schedule and mark it as scheduled.

2.Exploration: Add the next task to the current schedule and continue scheduling deeper into the possible configurations.

3.Deadline Check: If the current schedule meets all deadlines, the solution is considered valid. If not, backtrack and try different configurations.

4.Continuation: If no valid schedule is found, backtrack to adjust previous task placements and explore alternative schedules.

Search-Algorithms and Strategies Informed (Heuristic) Search Algorithm

Use Case: Route Planning for Navigation

Description: A GPS navigation system uses an informed search algorithm to **find the shortest path from a starting location to a destination**, considering real-world constraints such as traffic conditions and road types.

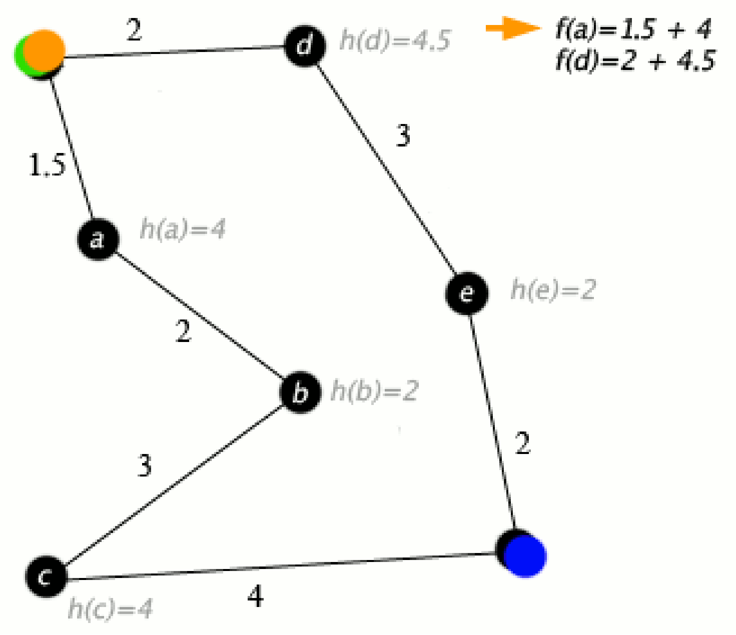
Informed Search Algorithm: A-Star Search

Explanations:

•**Initial State:** The system starts at the user's current location.

•**Search Strategy:** A* uses both the cost to reach the current node (g(n)) and a heuristic estimate of the cost to reach the goal from the current node (h(n)). It combines these to prioritize nodes that appear to lead most efficiently towards the goal.

•**Priority Queue:** Nodes are prioritized based on the function f(n) = g(n) + h(n), where g(n) is the path cost from the start node, and h(n) is the heuristic estimate of the remaining cost to the goal.



Steps:

1. Initialization: Start from the initial location with g(start) = 0 and h(start) estimated by a heuristic (e.g., straight-line distance to the destination).
2. Exploration: Expand nodes with the lowest f(n) value first. Calculate g(n) and h(n) for each adjacent node, update their f(n), and prioritize exploration based on this value.
3. Goal Check: When the goal location is reached, the system has found the optimal path.
4. Continuation: If not at the goal, add adjacent nodes to the priority queue and continue the process.

Search in Complex environments

Use Case: Optimization of a Marketing Campaign

- **Simple Hill Climbing**

Description: A **marketing team** wants to **optimize their digital ad campaign** to **maximize engagement metrics** (click-through rates, conversions, etc.). They use a simple hill climbing algorithm to iteratively improve their campaign strategy.

Process:

Initial Solution: Start with an **initial set of campaign parameters**, such as **ad placement**, **budget allocation**, and **targeting options**.

Evaluation: Evaluate the performance of the campaign using metrics like **click-through rate** (CTR) and conversions.

Modification: Make a small change to **one parameter** (e.g., increasing the budget for a specific ad group) and **evaluate the new performance**.

Selection: If the ***new parameters yield better results***, ***adopt the changes***; otherwise, ***revert to the previous parameters and try a different modification***.

Explanation:

The marketing team begins with a baseline ad campaign setup. They increase the budget for a specific ad and find that it improves engagement. They continue making small adjustments to other parameters, iteratively improving the campaign until no further improvements are observed.

Search in Complex environments

Use Case: Tuning Hyperparameters for Machine Learning Models

2. Steepest-Ascent Hill Climbing

Description: A data scientist uses steepest-ascent hill climbing to find the optimal hyperparameters for a machine learning model to achieve the highest possible accuracy.

Process:

Initial Solution: Start with an **initial set of hyperparameters** for the model, such as learning rate, number of layers, and batch size.

Evaluation: Evaluate the **model's performance** using **cross-validation** and **accuracy metrics**.

Search: **Systematically explore** all **possible adjustments** to **each hyperparameter** to **identify** the **one that improves** the **model's performance** the **most**.

Selection: Adopt the best-performing hyperparameter settings and continue exploring from this improved state.

Explanation :

The data scientist starts with default hyperparameters. They adjust each parameter systematically and evaluate performance improvements. They continue exploring until they identify the best combination of hyperparameters that yield the highest model accuracy.

Search in Complex environments

Use Case: Dynamic Pricing Optimization

3. Stochastic Hill Climbing

Description: A retail company uses **stochastic hill climbing** to **optimize dynamic pricing strategies** to maximize revenue across different product categories.

Process:

- **Initial Solution:** Begin with an initial pricing strategy for various products.
- **Random Moves:** Make random changes to the pricing strategy (e.g., adjusting prices up or down by a small percentage).
- **Evaluation:** Evaluate the impact of these changes on revenue and customer response**.**
- **Selection:** If the random change results in higher revenue, adopt the new pricing strategy; otherwise, revert and try a different random adjustment.

Explanation:

The company starts with a standard pricing model. It randomly adjusts prices across different products and observes the effects on sales and revenue. By iterating with random adjustments, they gradually find a pricing strategy that maximizes overall revenue.

Search in Complex environments

Use Case: Game Strategy Optimization

5. Local Beam Search

Description: A game AI uses local beam search to optimize strategies in a turn-based strategy game, where it evaluates multiple strategies in parallel.

Process:

- **Initial State:** Start with multiple initial game strategies (beams).
- **Evaluation:** Evaluate the performance of each strategy.
- **Selection:** At each iteration, generate new strategies based on the best-performing strategies. This involves creating multiple new strategies (successors) from the current strategies.
- **Beam Width:** Maintain a fixed number of the best strategies and discard the rest.
- **Explanation:**

The AI evaluates several strategies simultaneously in a game. After each round, it generates new strategies based on the most promising ones. By focusing on the top-performing strategies, it iteratively refines its approach to find the most effective game strategy.

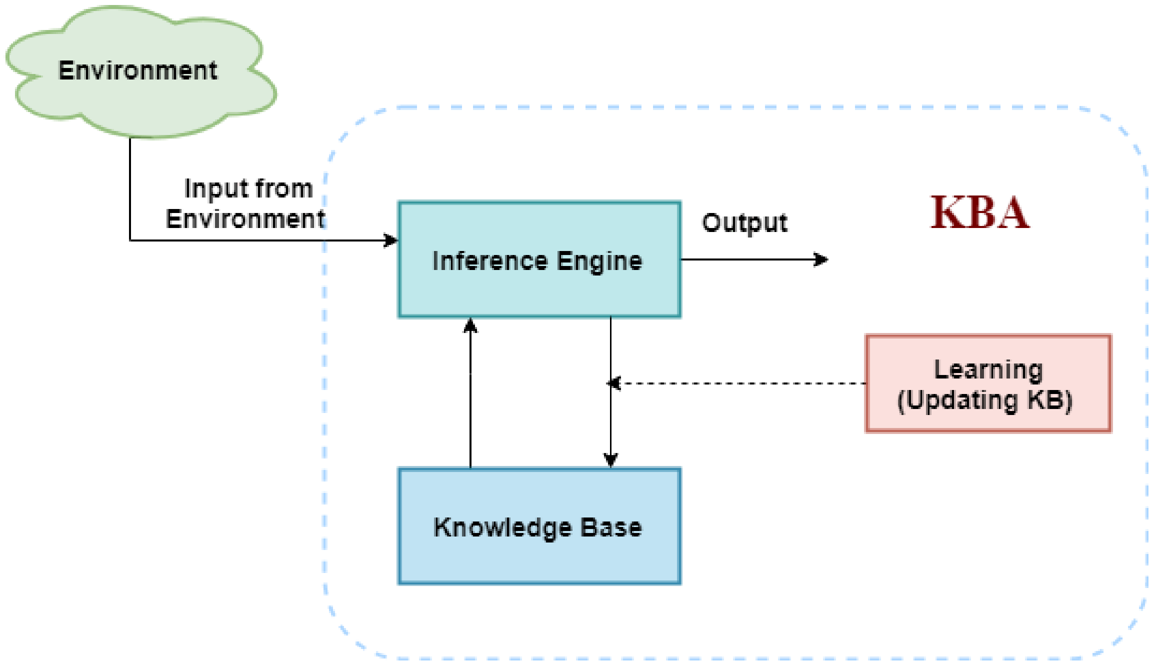
Module 3: Knowledge Representation

Knowledge-based agents

Use Case: Intelligent Customer Support System

An intelligent customer support system uses a knowledge-based agent to provide assistance to customers by answering queries, troubleshooting issues, and guiding users through various processes. This system leverages a knowledge base containing information about products, services, common issues, and solutions.

- **Knowledge Representation:**
  - **Knowledge Base:** The system maintains a **comprehensive knowledge** base that includes **product specifications, frequently asked questions (FAQs), troubleshooting guides, and service procedures**. The knowledge is organized into **categories and linked** with **rules** for **easy retrieval**.
  - **Rules and Relationships:** The knowledge base uses **if-then rules to match customer queries** with appropriate responses or actions. For example, if a customer reports an issue with a product not turning on, the system checks for common causes and solutions associated with this symptom.
- **Input Processing:**
  - **Customer Query:** The system receives a customer query via various channels, such as email, chat, or a support ticket.



  - **Information Extraction:** It processes the query to identify key information, such as product names, issue descriptions, and customer details. Natural language processing (NLP) techniques may be used to understand and interpret the query.

Knowledge-based agents

Use Case: Intelligent Customer Support System

- **Inference and Reasoning:**
  - **Matching and Retrieval:** The agent searches the **knowledge base** for **relevant information based** on the **identified keywords** and **context of the query**. It matches the **query against FAQs**, **troubleshooting guides, and known issues**.
  - **Contextual Understanding:** The agent uses **context** and **prior interactions** (if available) to **refine** its **search** and **provide more accurate** and **personalized responses**.
- **Response Generation:**
  - **Information Retrieval:** The system retrieves the **most relevant information from the knowledge base** and **formulates a response**. For example, if the query is about a common issue with a product, it provides step-by-step troubleshooting instructions.
  - **Recommendation:** If the issue is complex, the system might recommend additional actions such as contacting technical support.

Knowledge-based agents

Use Case: Intelligent Customer Support System

Learning and Updating:

- **Feedback Integration:** The system collects feedback from customers on the usefulness of the responses provided. This feedback is used to update and improve the knowledge base.

- **Continuous Improvement:** The system incorporates new information about products, services, and customer interactions to enhance its knowledge base and response accuracy over time.

Agents based on Propositional Logic

Use Case: Intelligent Task Scheduling System for Project Management

An intelligent task scheduling system uses an agent based on **propositional logic** to manage and optimize task assignments and deadlines within a project. The system ensures that tasks are assigned to appropriate team members, deadlines are met, and dependencies are respected.

1. **Knowledge Representation:**
   - **Propositional Logic:** The system represents task statuses, deadlines, and dependencies using propositional logic. Each proposition corresponds to a specific aspect of task management. For example:
     - `P1`: Task A is assigned to Team Member X.
     - `P2`: Task B's deadline is met.
     - `P3`: Task C depends on the completion of Task A.
   - **Logical Rules:** Task management policies and constraints are encoded as logical rules. For instance:
     - Rule 1: If `P3` (Task C depends on Task A) is true and `P1` (Task A is assigned) is true, then Task C can be scheduled only after Task A is completed.
     - Rule 2: If `P2` (Task B's deadline is met) is false, then generate an alert indicating that Task B is overdue.
2. **Input Processing:**
   - **Task Data Collection:** The system collects data about tasks, including assignments, deadlines, and dependencies from project management tools or input forms.
   - **Proposition Evaluation:** It translates this data into logical propositions. For example, if Task A is assigned to Team Member X, `P1` is true.

Agents based on Propositional Logic

Use Case: Intelligent Task Scheduling System for Project Management

3. Inference and Decision Making:

- **Task Scheduling:** The agent evaluates the propositions based on the logical rules to determine task scheduling and assignment. For example:
  - If `P3` is true (Task C depends on Task A) and `P1` is true (Task A is assigned), then Task C can be scheduled only after Task A is completed.
  - If `P2` is false (Task B's deadline is missed), Rule 2 generates an alert.
- **Conflict Resolution:** The system identifies scheduling conflicts and dependencies. It adjusts schedules or reassigns tasks to ensure deadlines are met and dependencies are respected.

4. Action and Reporting:

- **Task Assignment and Deadline Alerts :** The system suggests or automatically reassigns tasks based on availability, skillset, and workload of team members. Generates alerts and notifications for tasks approaching or missing their deadlines, allowing project managers to take corrective actions.
- **Progress Reports:** Provides reports on task status, upcoming deadlines, and any issues related to task dependencies or delays.
- **Feedback Integration:** The system learns from past projects and task completion patterns to improve its scheduling algorithms and recommendations. It can update its rules based on observed performance and feedback.
- **Process Improvement:** Continuously refines task scheduling and assignment rules based on project outcomes and changes in team dynamics or project scope.

Agents based on First-order logic

Use Case: Smart Home Automation System

A smart home automation system uses an agent based on First-Order Logic to manage and control various smart devices within a home. The system ensures that the home environment is optimized for comfort, security, and energy efficiency based on the occupants' preferences and environmental conditions.

- **Knowledge Representation:**
- **First-Order Logic:** The system uses FOL to represent knowledge about home devices, their states, and the rules for automation. This includes:
  - **Predicates:** `Device(DeviceID)`, `IsOn(DeviceID)`, `Temperature(TemperatureValue)`, `Occupant(OccupantID)`.
  - **Functions:** `GetPreferredTemperature(OccupantID)`, `GetSecurityStatus(DeviceID)`.
  - **Quantifiers:** `For all x (Occupant(x) → PreferredTemperature(x, Temp))`, `Exists y (Device(y) ∧ IsOn(y))`, `For all z (Device(z) ∧ SecurityDevice(z) → GetSecurityStatus(z) = True)`.
- **Input Processing:**
- **Device and Sensor Data:** The system collects data from various smart devices and sensors, such as temperature sensors, motion detectors, and smart thermostats.
- **Proposition Translation:** It translates this data into FOL statements. For example, if the living room temperature is 72°F and the smart thermostat is set to 70°F, the system generates statements like `Temperature(72)` and `IsOn(ThermostatID)`.

Agents based on First-order logic

Use Case: Smart Home Automation System

A smart home automation system uses an agent based on First-Order Logic to manage and control various smart devices within a home. The system ensures that the home environment is optimized for comfort, security, and energy efficiency based on the occupants' preferences and environmental conditions.

- **Inference and Reasoning:**
- **Comfort Optimization:** The agent applies FOL rules to ensure that the home environment matches occupants' preferences. For example:
  - Using the rule `For all x (Occupant(x) → PreferredTemperature(x, Temp))`, the system adjusts the thermostat to match the preferred temperature for each occupant.
- **Security Management:** It uses logical rules to manage home security. For example:
  - Using `For all z (Device(z) ∧ SecurityDevice(z) → GetSecurityStatus(z) = True)`, the system ensures all security devices are active and functioning.
- **Action and Reporting:**
- **Device Control:** The system adjusts devices based on the logical inference. For example:
  - It turns the thermostat on or off to maintain the preferred temperature: If `Temperature(current) < GetPreferredTemperature(OccupantID)`, then turn the thermostat on.
- **Alerts and Notifications:** Generates alerts for maintenance issues or security breaches. For example, if a security device fails or is deactivated, the system sends an alert to the homeowner.
- **Energy Efficiency:** Provides recommendations for optimizing energy use, such as suggesting times to reduce heating or cooling when no one is home.

Agents based on First-order logic

Use Case: Smart Home Automation System

A smart home automation system uses an agent based on First-Order Logic to manage and control various smart devices within a home. The system ensures that the home environment is optimized for comfort, security, and energy efficiency based on the occupants' preferences and environmental conditions.

Learning and Adaptation:

- **Preference Learning:** The system learns from occupants' behaviors and preferences over time to make more accurate adjustments. For example, if an occupant consistently prefers a certain temperature in the evening, the system adapts to this preference.
- **Adaptation to Environmental Changes:** The system adapts to changes in the environment, such as adjusting to seasonal temperature changes or updates in security protocols.

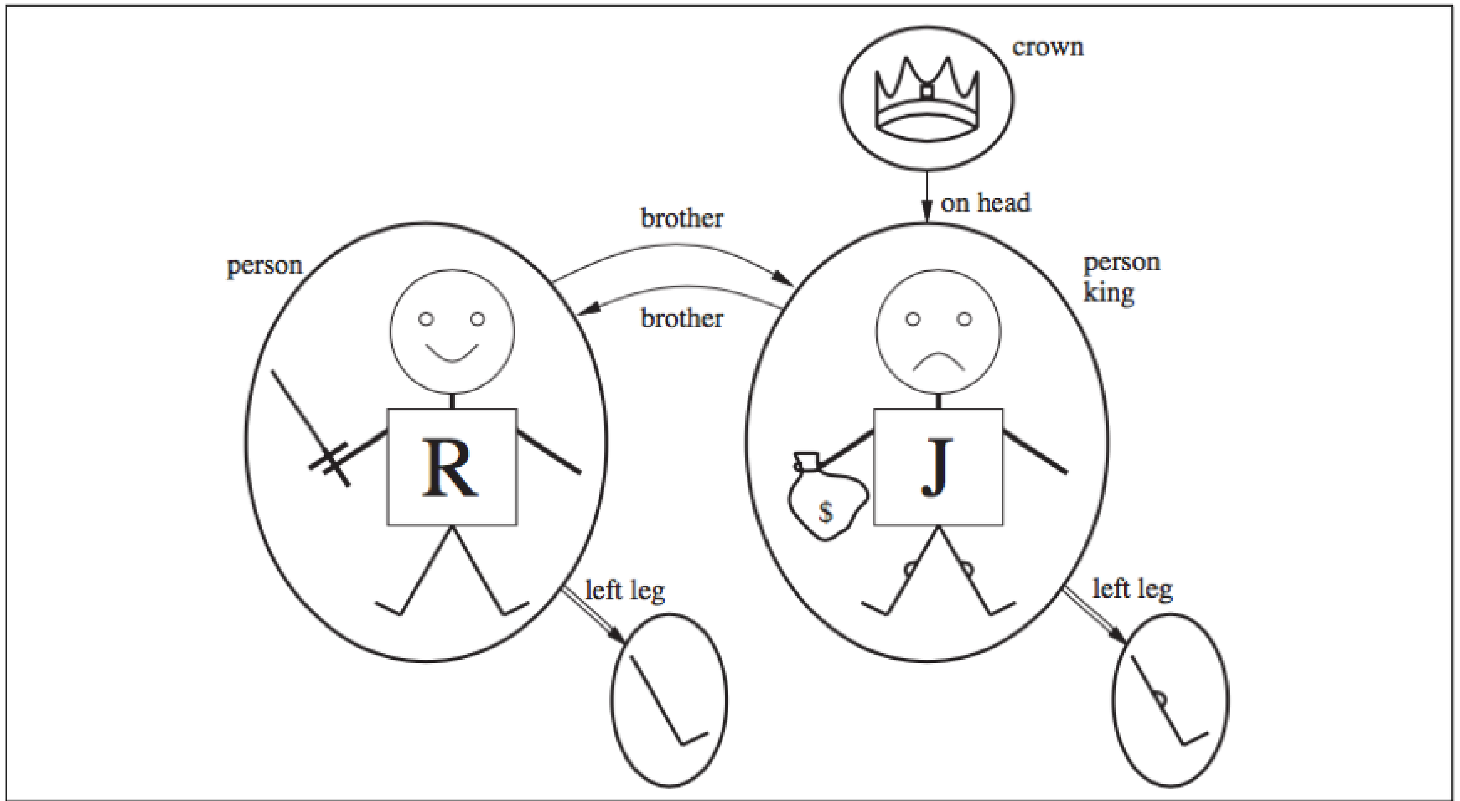


Figure 8.2 A model containing five objects, two binary relations, three unary relations (indicated by labels on the objects), and one unary function, left-leg.