Artificial Intelligence (CS280)

Noushin Saba

Lecture 2
Knowledge-Based Systems and Intelligent Agents: Overview and Conceptual Framework

Outline

Knowledge based Systems

- What and Why?
- Components of KBS
- Conceptual Framework
- Reasoning in KBS
- Applications and Limitations

Intelligent agents

- Definition
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Outline

Knowledge based Systems

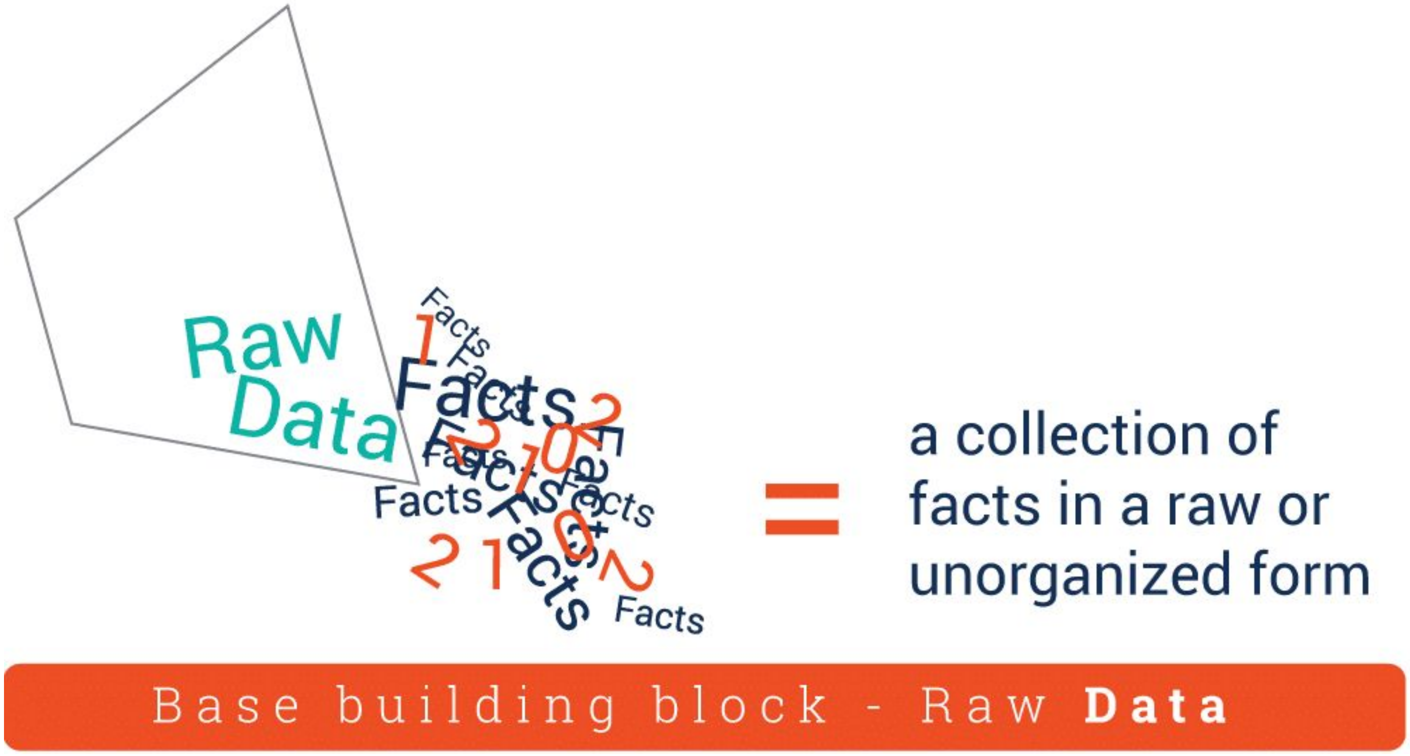
- What and Why?
- Components of KBS
- Conceptual Framework
- Reasoning in KBS
- Applications and Limitations

Intelligent agents

- Definition
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

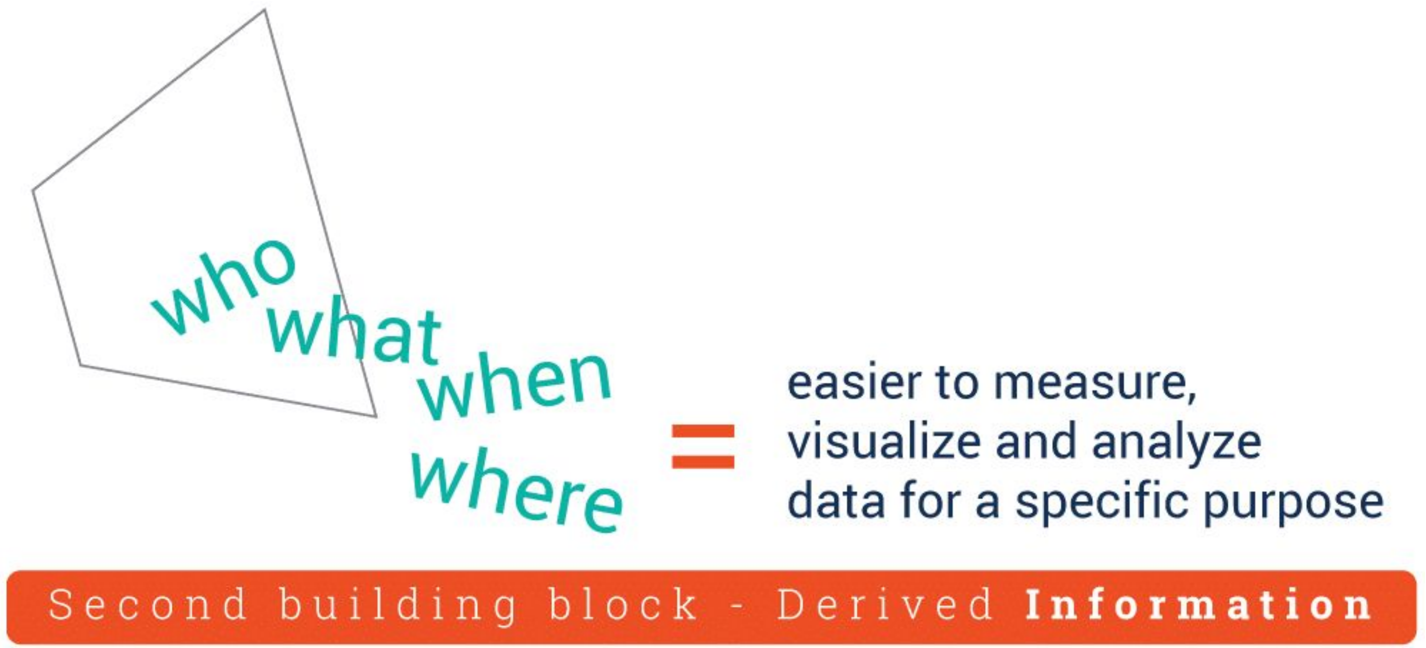
Introduction – Why Knowledge?

- **Data**
  - Raw, unorganized facts without meaning.
  - Example □ "Temperature = 102°F", "Patient age = 25".



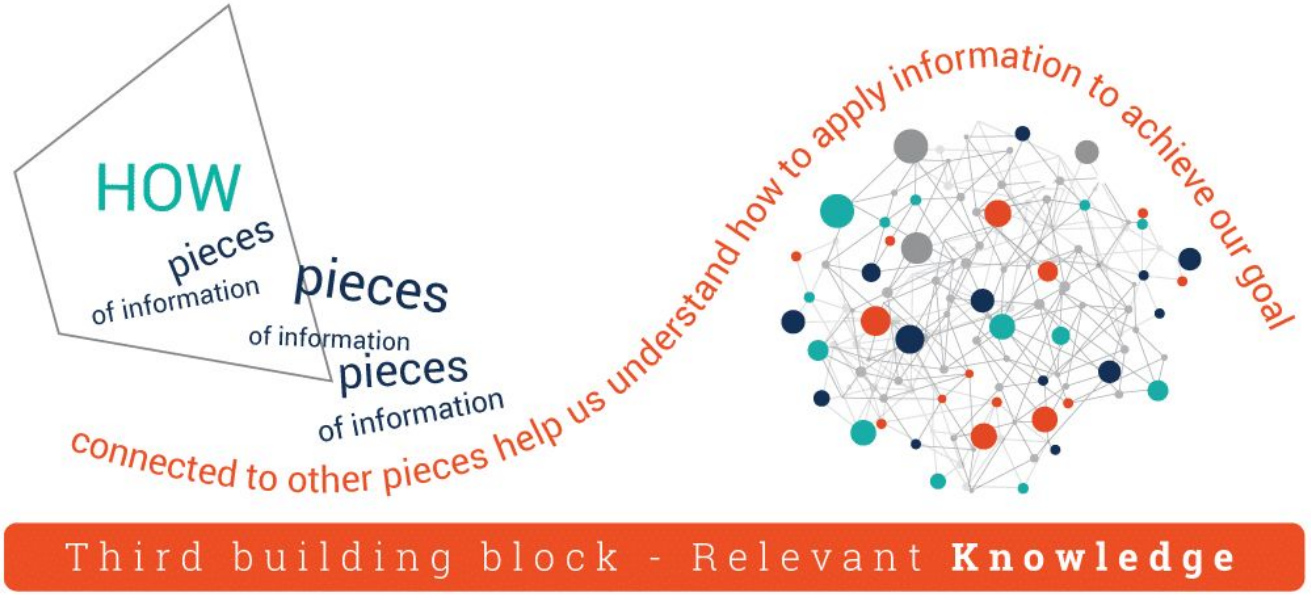
Introduction – Why Knowledge? (cont..)

- **Information**
  - Processed data that provides meaning or context.
  - Example: “A temperature of 102°F indicates high fever in a 25-year-old patient.”



Introduction – Why Knowledge? (cont..)

- **Knowledge**
  - Information combined with experience, rules, or interpretation.
  - Allows decision-making and problem solving.
  - Example: *"High fever may indicate an infection; antibiotics may be needed."*



Why AI needs Knowledge?

- Enables machines to reason about problems.
- Helps systems choose the best action in a given situation.
- Allows systems to explain their reasoning to humans (important in expert systems).
- Makes AI more reliable and intelligent compared to just using raw data.

What is a Knowledge-Based System?

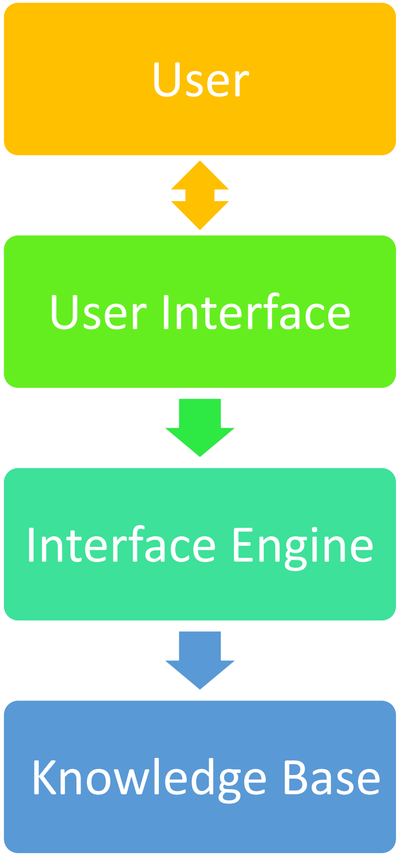
- **Knowledge-Based System (KBS)** is a computer program that uses stored knowledge and reasoning methods to solve problems.
- Two main components
  - Knowledge Base contains facts about the world and rules for reasoning.
    - Example: “If fever > 101°F and cough → possible flu.”
  - Inference Engine applies logical reasoning to the knowledge base to reach conclusions or suggest actions.
- Works like a human expert:
  - Uses prior knowledge (rules, experience).
  - Applies reasoning to make decisions.
- Example: **MYCIN system** diagnosed bacterial infections and recommended antibiotics.

Components of a KBS

- **Knowledge Base (KB)**
  - Stores domain-specific knowledge (facts + rules).
  - Example Fact: “John has a fever.”
  - Example Rule: “If fever > 101°F ⟶ possible infection.
  - ”Analogy: Like a library of expert knowledge.
- **Inference Engine (IE)**
  - The reasoning brain of the system.
  - Applies logical rules to the knowledge base.
  - Derives **new facts**, checks consistency, and provides solutions.
  - Analogy: Like a **doctor applying medical rules** to patient data.

Components of a KBS (cont..)

- **User Interface (UI)**
  - Medium through which users interact with the system.
  - Accepts inputs (questions, facts) and displays outputs (solutions, explanations).
  - Example: Dialog box, chatbot interface.
  - Analogy: Like a **conversation with a human expert**.



Conceptual Framework of KBS

- A KBS works through **four key processes**:
- **Knowledge Acquisition**
  - Collecting knowledge from experts, books, databases, or observations.
  - Example: Interviewing doctors to capture rules for diagnosis.
- **Knowledge Representation**
  - Storing knowledge in a structured way so the computer can use it.
  - Techniques: rules, logic, semantic networks, frames, ontologies.
  - *Example: IF fever AND cough → THEN possible flu.*

Conceptual Framework of KBS (cont..)

- **Reasoning Mechanism**
  - Inference engine applies reasoning to stored knowledge.
  - Can be **forward chaining** (data-driven) or **backward chaining** (goal-driven).
  - *Example: Given symptoms, derive a diagnosis.*
- **Explanation Facility**
  - Explains the reasoning steps to users.
  - Builds **trust** by showing *“Why this conclusion?”*.
  - *Example: “Diagnosis: Flu → because patient had fever + cough.”*


• Collecting knowledge form experts, books, sensors and databases
• Organizing knowledge in machine readable format
Knowledge Acquisition
Knowledge Representation
Explanation Facility
Reasoning Mechanism
• Provides reasoning steps and conclusion to the user
• Inference engine applies reasoning rules to stored knowledge

Reasoning in KBS

- **Reasoning** = The process of drawing conclusions from knowledge.
- Two main methods used in KBS
- **Forward Chaining (Data-Driven)**
  - Starts with known facts.
  - Applies rules step by step until a conclusion is reached.
  - Works like a chain reaction → facts trigger rules.
  - Example**:**
    - Fact: Patient has fever + cough.
    - Rule: *If fever AND cough → possible flu.*
    - **Conclusion:** Patient may have flu.

Reasoning in KBS (cont..)

- **Backward Chaining (Goal-Driven)**
  - Starts with a goal (hypothesis).
  - Works backward to check if facts support the goal.
  - Efficient when the number of possible goals is small.
  - Example**:**
    - Goal: Does patient have flu?
    - Rule: *If fever AND cough → flu.*
    - Check facts: Does patient have fever? Does patient have cough?
    - If yes → confirm goal.

Forward Chaining (Data-Driven)

Backward Chaining (Goal-Driven)

Applications of KBS

- **Medicine** □ Diagnosis and treatment recommendation.
    - Example: MYCIN – diagnosed bacterial infections and suggested antibiotics.
- **Business / Management** □ Decision support for planning, scheduling, and risk analysis.
    - Example: Expert systems for financial forecasting or loan approvals.
- **Education** □ Intelligent tutoring systems that adapt to student learning styles.
    - Example: Systems that provide personalized quizzes and guidance.
- **Engineering & Manufacturing** □ Fault diagnosis, process control, and system monitoring.
    - Example: Detecting faults in aircraft engines or industrial machines.
- **Customer Support / Help Desks** □ Chatbots and troubleshooting systems using stored knowledge.
    - Example: Virtual assistants for IT support.

Advantages of KBS

- **Consistency in Decision-Making**
  - Same problem → always gives the same answer.
  - Unlike humans, it doesn’t get tired, emotional, or biased.
- **Captures Expert Knowledge**
  - Preserves rare or specialized expertise for wider use.
  - Useful in domains with limited experts (e.g., medicine, aerospace).
- **24/7 Availability**
  - Can work continuously without breaks.
  - Provides instant responses anytime.
- **Handles Large Knowledge Bases**
  - Can store and process thousands of rules faster than humans.
- **Training & Education**
  - Can be used as a tutor or advisor to train new staff or students.

Limitations of KBS

- **Knowledge Acquisition Bottleneck**
  - Extracting expert knowledge is difficult, time-consuming, and expensive.
  - Experts may disagree or struggle to explain their reasoning.
- **Limited to Encoded Knowledge**
  - Can only solve problems it has rules for.
  - Cannot easily adapt to **new or unexpected situations**.
- **Lacks Common Sense & Intuition**
  - Unlike humans, it cannot use general world knowledge.
  - May give answers that are logically correct but impractical.
- **Maintenance Challenges**
  - Updating rules and knowledge is costly.
  - As the knowledge base grows, the system becomes harder to manage.
- **No Learning Ability (Traditional KBS)**
  - Static knowledge — does not improve automatically.
  - Needs experts to constantly update the system.

From KBS to Intelligent Agents

- **Knowledge-Based System (KBS)**
  - Focus: Reasoning only (solves problems using facts & rules).
  - Works like a consultant: you ask questions, it gives answers.
- **Intelligent Agent**
  - Focus: Reasoning + Acting in an environment.
  - Can perceive (sense inputs), decide (use knowledge & reasoning), and act (perform actions).
  - Works like a decision-maker that interacts with the world.
- **Relationship**
  - Many intelligent agents use a KBS internally as their reasoning engine.
  - KBS = brain of the agent; Agent = brain + sensors + actions.

From KBS to Intelligent Agents (cont..)

- **Example:**
- **KBS**
  - A medical rule-based system for diagnosis.
- **Agent**
  - A doctor chatbot that asks patients questions, reasons with medical rules, and then recommends treatment.

Knowledge based System

Knowledge Base + Rules	Inference Engine	Medical diagnosis system

Intelligent Agent

Use KBS for reasoning	Add sensors and Actuators	Doctor chatbot that ask patient questions

Output/ Action

Suggests treatment based on diagnosis	Take Antibiotics

Outline

Knowledge based Systems

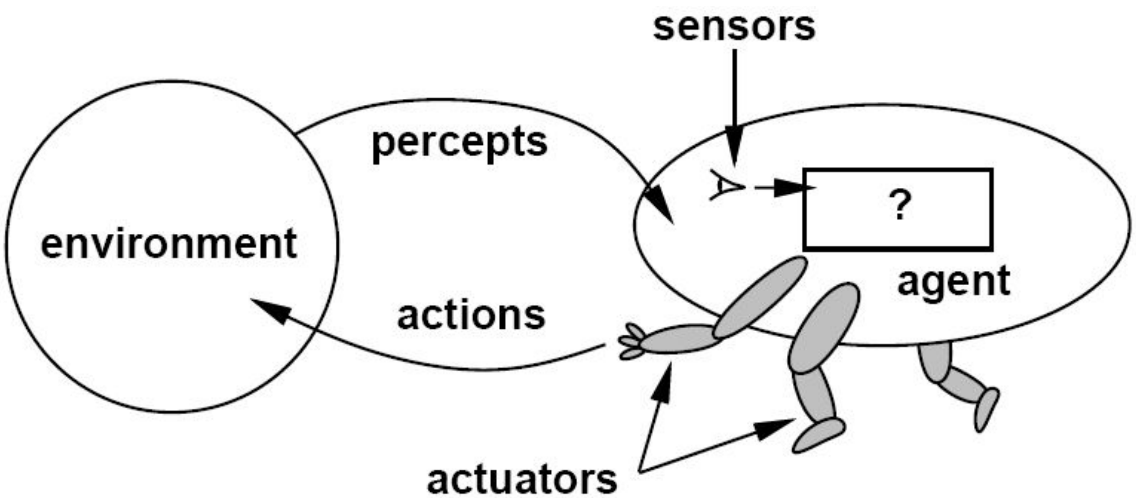
- What and Why?
- Components of KBS
- Conceptual Framework
- Reasoning in KBS
- Applications and Limitations

Intelligent agents

- Definition
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

What is Agents?

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

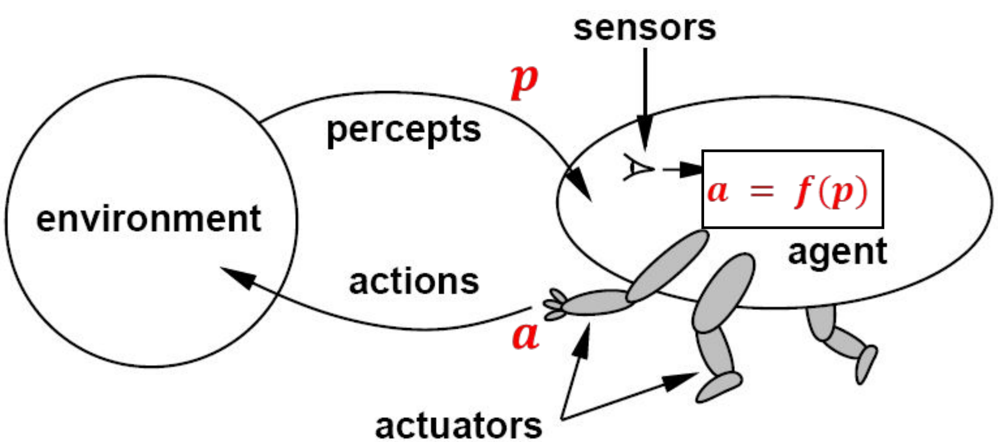


- **Control theory**: A **closed-loop control system** (= feedback control system) is a set of mechanical or electronic devices that automatically regulate a process variable to a desired state or set point without human interaction. The agent is called a controller.
- **Softbot**: Agent is a software program that runs on a host device.

Agent Function and Agent Program

The agent function maps from the set of all possible *percept sequences* P^* to the *set of actions* A formulated as an abstract mathematical function.

$$f : P^* \rightarrow A$$



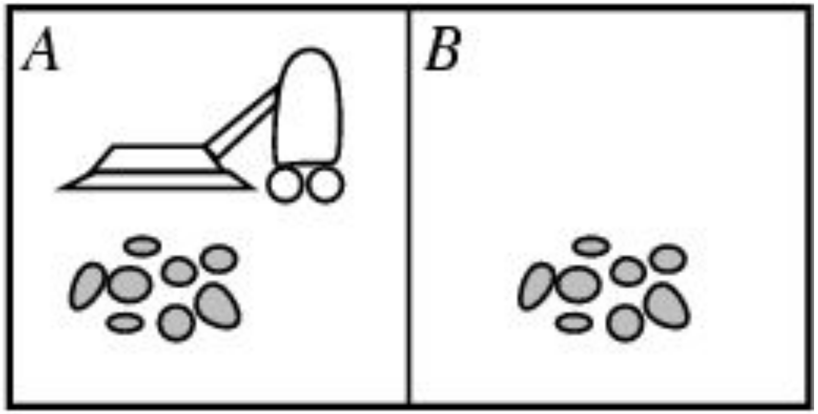
The agent program is a concrete implementation of this function for a given physical system.

Agent = architecture (hardware) + agent program (implementation of f)

- Sensors
- Memory
- Computational power

Example: Vacuum-cleaner World

- **Percepts:**
  Location and status,
  e.g., [A, Dirty]
- **Actions:**
  Left, Right, Suck, NoOp



Most recent Percept p

Agent function: $f : P^* \to A$

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
…	
[A, Clean], [B, Clean]	Left
…	
[A, Clean], [B, Clean], [A, Dirty]	Suck
…	

Implemented agent program:

function Vacuum-Agent([location, status]) returns an action a

```
if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

Problem: This table can become infinitively large!

Rational Agents: What is Good Behavior?

Foundation

- **Consequentialism**: Evaluate behavior by its consequences.
- **Utilitarianism**: Maximize happiness and well-being.

Definition of a rational agent:

"For each possible percept sequence, a rational agent should select an ***action*** *that* ***maximizes its expected performance measure****, given the evidence provided by the* ***percept sequence*** *and the* ***agent's built-in knowledge****."*

- **Performance measure**: An *objective* criterion for success of an agent's behavior (often called utility function or reward function).
- **Expectation**: Outcome averaged over all possible situations that may arise.

Rule: Pick the action that maximize the expected utility

$$a = \text{argmax}_{a \in \mathrm{A}} E(U \mid a)$$

Rational Agents

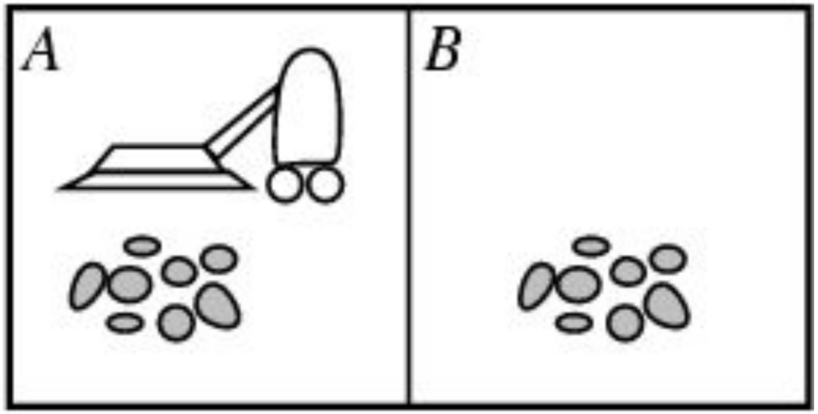
Rule: Pick the action that maximize the expected utility

$$a = \mathrm{argmax}_{a \in \mathrm{A}}\, E(U \mid a)$$

This means:

- **Rationality is an ideal** – it implies that no one can build a better agent
- **Rationality ≠ Omniscience** – rational agents can make mistakes if percepts and knowledge do not suffice to make a good decision
- **Rationality ≠ Perfection** – rational agents maximize **expected** outcomes not actual outcomes
- **It is rational to explore and learn** – I.e., **use percepts** to supplement prior knowledge and become autonomous
- **Rationality is often bounded** by available memory, computational power, available sensors, etc.

Example: Vacuum-cleaner World



- **Percepts:**
  Location and status,
  e.g., [A, Dirty]
- **Actions:**
  Left, Right, Suck, NoOp

Agent function:

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
…	
[A, Clean], [B, Clean]	Left
…	

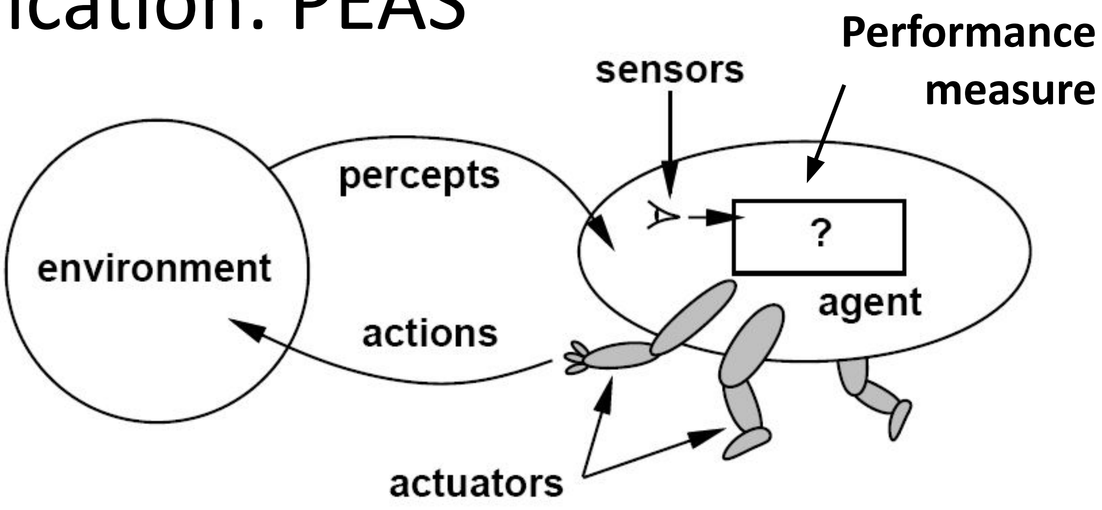
Implemented agent program:

function Vacuum-Agent([location, status])
returns an action

```
if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

What could be a performance measure?
Is this agent program rational?

Problem Specification: PEAS



Performance measure	Environment	Actuators	Sensors

Defines utility and what is rational

Components and rules of how actions affect the environment.

Defines available actions

Defines percepts

Example: Automated Taxi Driver

Performance measure	Environment	Actuators	Sensors
• Safe • fast • legal • comfortable trip • maximize profits	• Roads • other traffic • pedestrians • customers	• Steering wheel • accelerator • brake • signal • horn	• Cameras • sonar • speedometer • GPS • Odometer • engine sensors • keyboard

Example: Spam Filter

Performance measure

- Accuracy: Minimizing false positives (legitimate emails marked as spam), false negatives (spam emails not marked)

Environment

- A user's email account
- email server

Actuators

- Mark as spam
- delete
- etc.

Sensors

- Incoming messages
- other information about user's account

Environment Types

Fully observable: The agent's sensors give it access to the complete state of the environment. The agent can "see" the whole environment.

vs.

Partially observable: The agent cannot see all aspects of the environment. E.g., it can't see through walls

Deterministic: Changes in the environment is completely determined by the current state of the environment and the agent's action.

vs.

Stochastic: Changes cannot be determined from the current state and the action (there is some randomness).

Strategic: The environment is stochastic and adversarial. It chooses actions strategically to harm the agent. E.g., a game where the other player is modeled as part of the environment.

Known: The agent knows the rules of the environment and can predict the outcome of actions.

vs.

Unknown: The agent cannot predict the outcome of actions.

Environment Types (cont..)

Static: The environment is **not** changing while agent is deliberating. **Semi dynamic:** the environment is static, but the agent's performance score depends on how fast it acts.

vs.

Dynamic: The environment is changing while the agent is deliberating.

Discrete: The environment provides a fixed number of distinct percepts, actions, and environment states. Time can also evolve in a discrete or continuous fashion.

vs.

Continuous: Percepts, actions, state variables or time are continuous leading to an infinite state, percept or action space.

Episodic: Episode = a self-contained sequence of actions. The agent's choice of action in one episode does not affect the next episodes. The agent does the same task repeatedly.

vs.

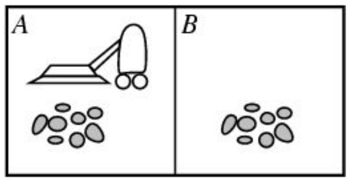
Sequential: Actions now affect the outcomes later. E.g., learning makes problems sequential.

Single agent: An agent operating by itself in an environment.

vs.

Multi-agent: Agent cooperate or compete in the same environment.

Examples of Different Environments



Vacuum cleaner world

Chess with a clock

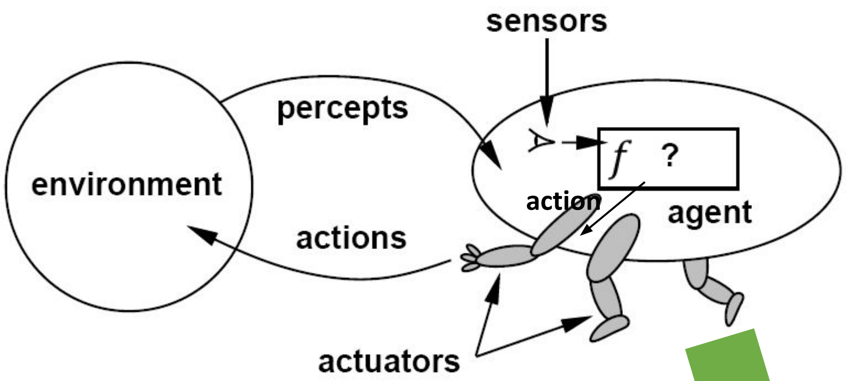
Scrabble

Taxi driving

Observable	Partially	Fully	Partially	Partially
Deterministic	Deterministic	Determ. game Mechanics + Strategic*	Stochastic +Strategic	Stochastic
Episodic	Episodic	Episodic	Episodic	Sequential
Static	Static	Semidynamic	Static	Dynamic
Discrete	Discrete	Discrete	Discrete	Continuous
Single agent	Single	Multi*	Multi*	Multi*

* Can be models as a single agent problem with the other agent(s) in the environment.

Designing a Rational Agent

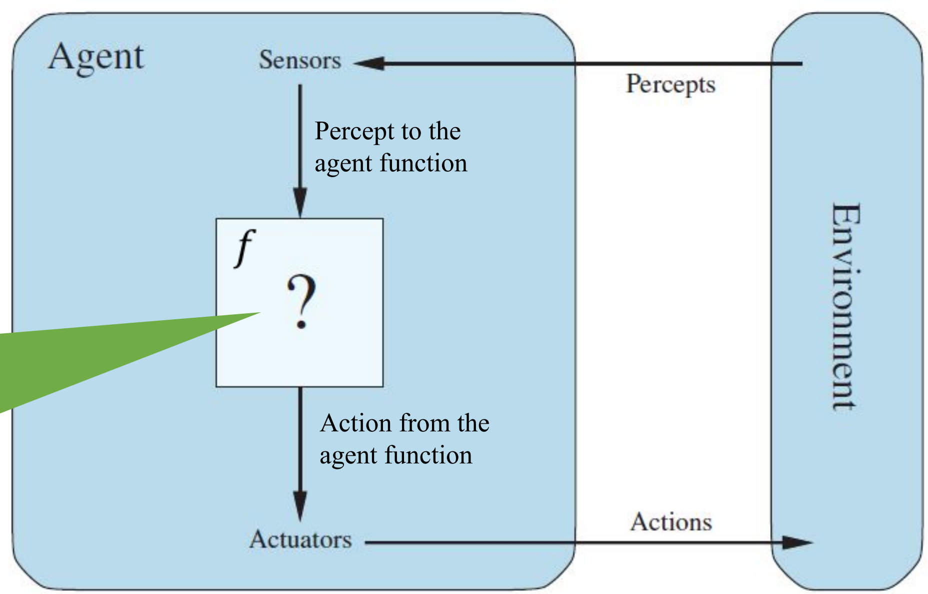


Remember the definition of a rational agent:

> *"For each possible percept sequence, a rational agent should select an **action** that **maximizes its expected performance measure**, given the evidence provided by the **percept sequence** and the **agent's built-in knowledge**."*

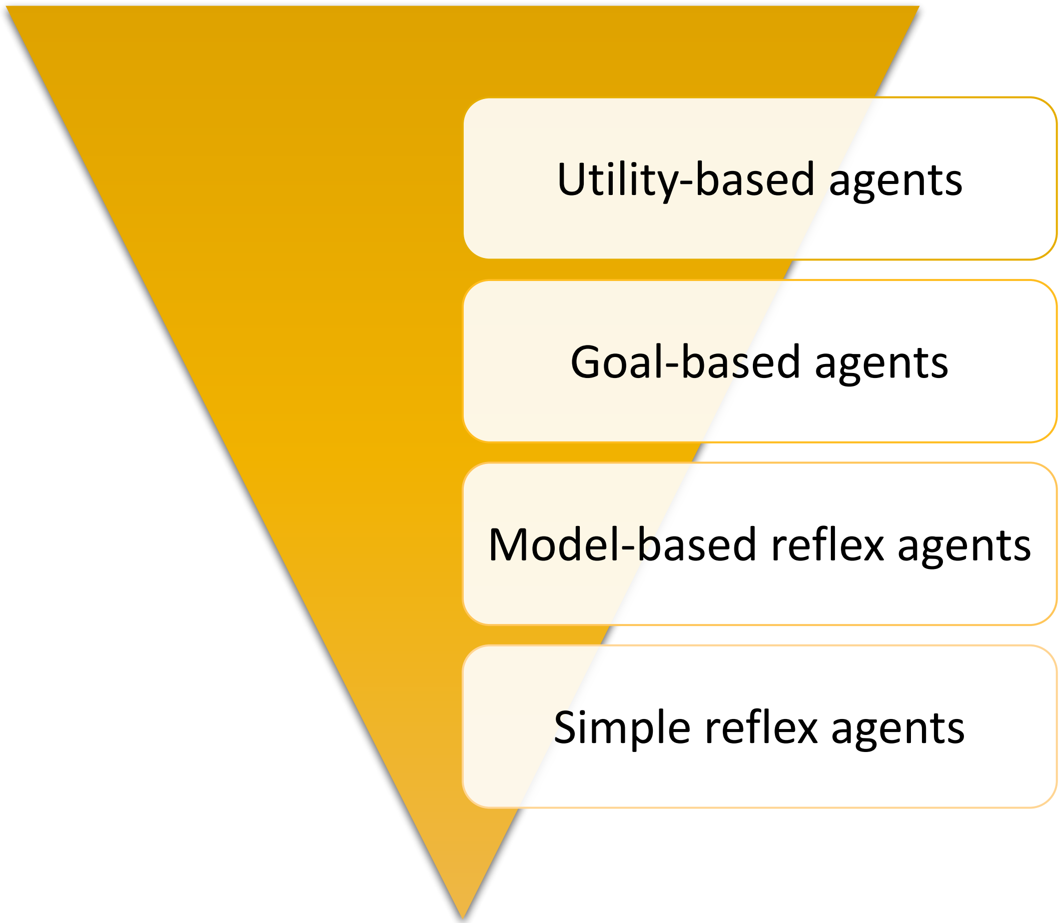
Agent Function

- Assess performance measure
- Remember percept sequence
- Built-in knowledge



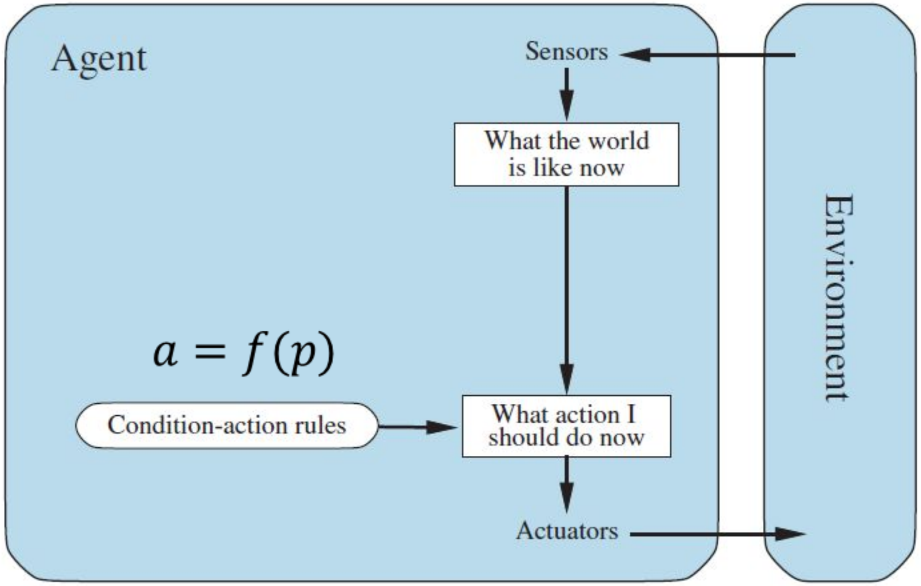
Note: Everything outside the agent function can be seen as the environment.

Hierarchy of Agent Types



Simple Reflex Agent

- Uses only built-in knowledge in the form of **rules** that select action only **based on the current percept.** This is typically very fast!
- The **agent does not know about the performance measure**! But well-designed rules can lead to good performance.
- The agent needs no memory and ignores all past percepts.

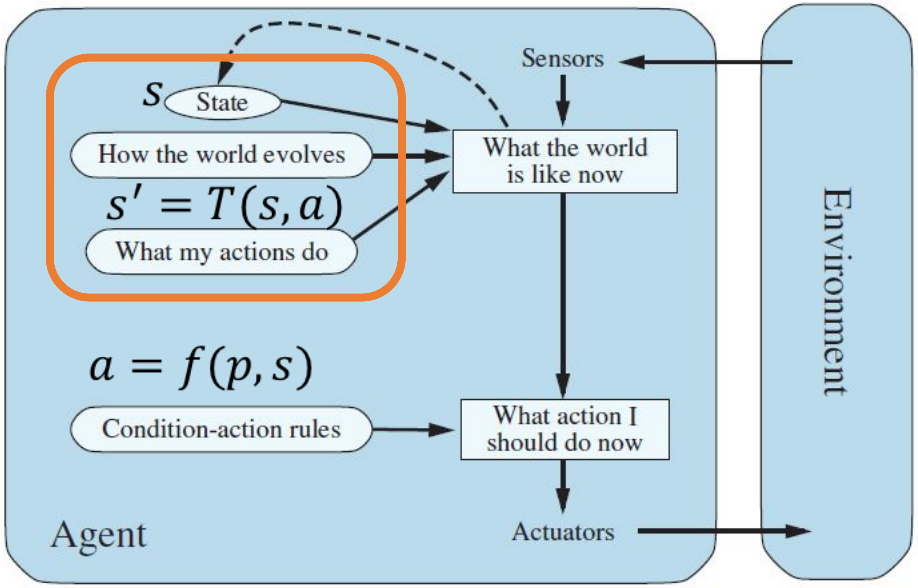


The interaction is a sequence: $p_0, a_0, p_1, a_1, p_2, a_2, \ldots p_t, a_t, \ldots$

Example: A simple vacuum cleaner that uses rules based on its current sensor input.

Model-based Reflex Agent

- Maintains a **state variable** to keeps track of aspects of the environment that cannot be currently observed. I.e., it has memory and knows how the environment reacts to actions (called **transition function**).
- The state is updated using the percept.
- There is now more information for the **rules** to make better decisions.



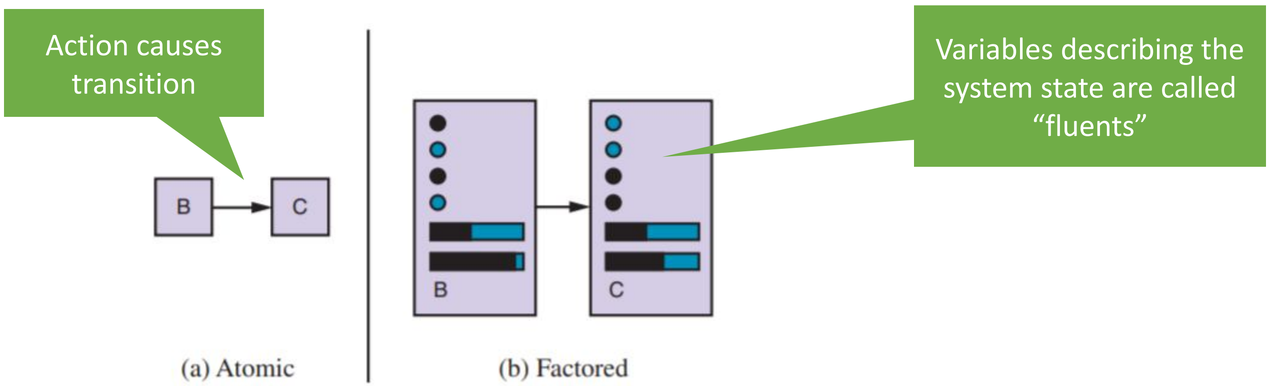
The interaction is a sequence: $s_0, a_0, p_1, s_1, a_1, p_2, s_2, a_2, p_3, \ldots, p_t, s_t, a_t, \ldots$

Example: A vacuum cleaner that remembers were it has already cleaned.

State Representation

States help to keep track of the environment and the agent in the environment. This is often also called the **system state**. The representation can be

- **Atomic**: Just a label for a black box. E.g., A, B
- **Factored**: A set of attribute values called fluents.
  E.g., [location = left, status = clean, temperature = 75 deg. F]

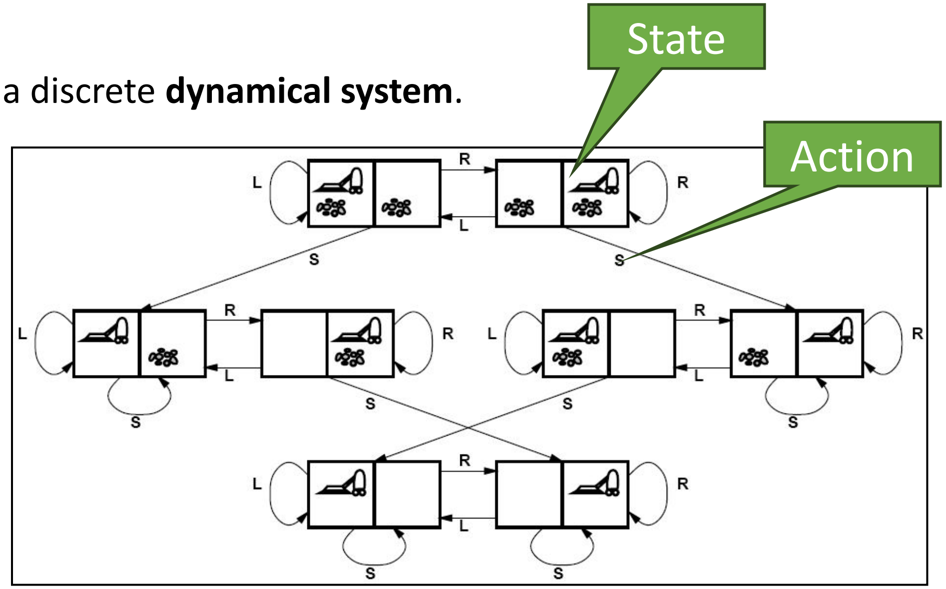


We often construct atomic labels from factored information. E.g.: If the agent's state is the coordinate x = 7 and y = 3, then the atomic state label could be the string "(7, 3)". With the atomic representation, we can only compare if two labels are the same. With the factored state representation, we can reason more and calculate the distance between states!

The set of all possible states is called the **state space** S. This set is typically very large!

Transition Function

- The environment is modeled as a discrete **dynamical system**.
- Example of a state diagram for the Vacuum cleaner world.



- States change because of
  a. System dynamics of the environment (the environment evolves by itself).
  b. The actions of the agent.

- Both types of changes are represented by the transition function written as

$$T: S \times A \to S$$ or $$s' = T(s, a)$$

S ... set of states
A ... set of available actions
$a \in A$ ... an action
$s \in S$ ... current state
$s' \in S$ ... next state

Old-school vs. Smart Thermostat

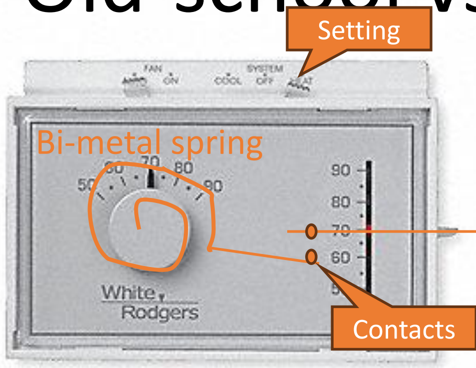
Old-school thermostat

Percepts	**States**

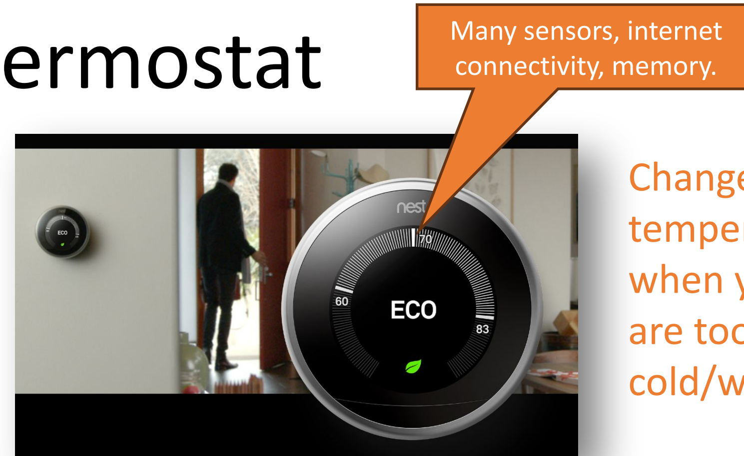
Smart thermostat

Percepts	**States**

Old-school vs. Smart Thermostat



Set target temperature



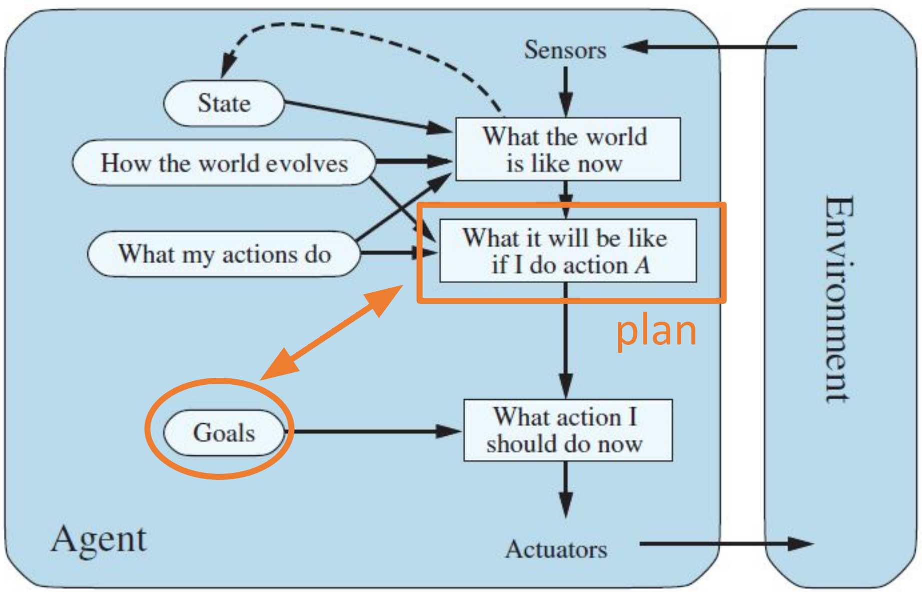
Change temperature when you are too cold/warm

Old-school thermostat	
Percepts	**States**
Setting: Cool, off, heat Contact: Open, closed	No states (only reacts to the current percepts)

Smart thermostat	
Percepts	**States**
Sensors • Temp: deg. F • Someone walking by • Someone changes temp. Internet • Outside temp. • Weather report • Energy curtailment • Day & time • …	Factored states • Estimated time to cool the house • Someone home? • How long till someone is coming home? • Schedule • ….

Goal-based Agent

- The agent has the task to reach a defined **goal state** and is then finished.
- The agent needs to move towards the goal. It can use **search algorithms** to plan actions that lead to the goal.
- Performance measure: the **cost to reach the goal**.



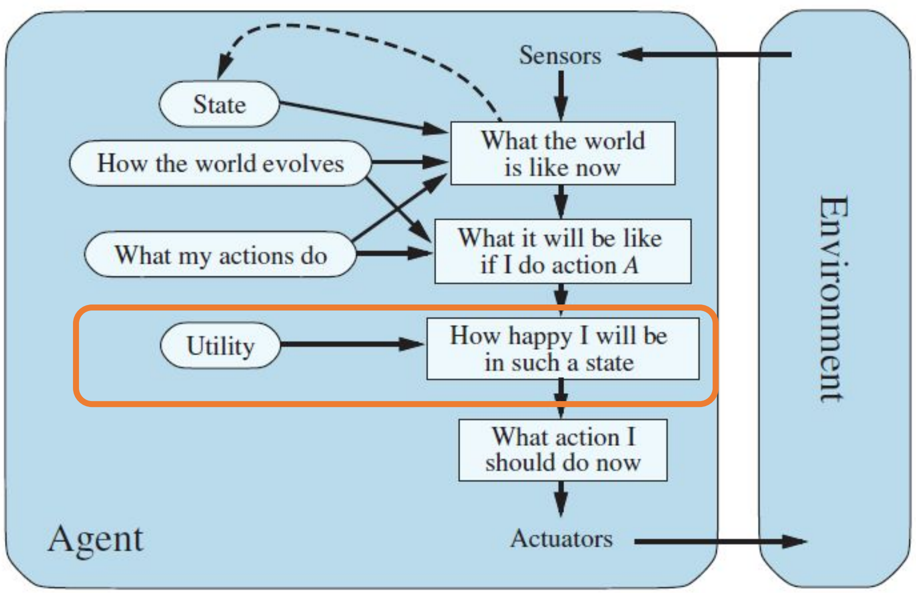
$$\underbrace{a = \text{argmin}_{a_0 \in A}\left[\sum_{t=0}^{T} c_t \,\middle|\, s_T \in S^{goal}\right]}_{\text{Sum of the cost of a planed sequence of actions that leads to a goal state}}$$

The interaction is a sequence: $\underbrace{s_0, a_0, p_1, s_1, a_1, p_2, s_2, a_2, \dots, s^{goal}}_{\text{cost}}$

Example: Solving a puzzle. What action gets me closer to the solution?

Utility-based Agent

- The agent uses a utility function to evaluate the **desirability of each possible states.** This is typically expressed as the reward of being in a state $R(s)$.
- Choose actions to stay in desirable states.
- Performance measure: The discounted sum of **expected utility over time**.



$$a = \text{arg}max_{a_0 \in A} \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]}_{\text{Utility is the expected future discounted reward}}$$

Techniques: Markov decision processes, reinforcement learning

The interaction is a sequence: $\underbrace{s_0, a_0, p_1, s_1, a_1 . p_2, s_2, a_2, \ldots}_{\text{reward}}$

Example: An autonomous Mars rover prefers states where its battery is not critically low.

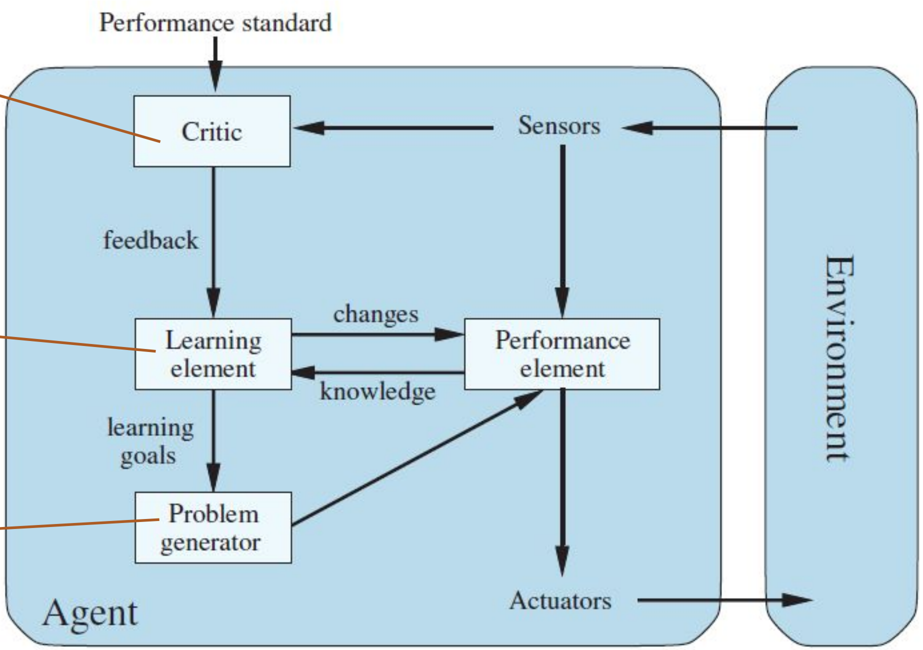
Agents that Learn

The **learning element** modifies the agent program (reflex-based, goal-based, or utility-based) to improve its performance.

How is the agent currently performing?

Updates how the performance element chooses actions.

Generate actions for exploration



Smart Thermostat: What Type of Agent is it?

Change temperature when you are too cold/warm

Simple Reflex Agent?

Model-based Reflex Agent?

Goal-based?

Utility-based?

Smart thermostat

Percepts

Sensors

- Temp: deg. F
- Someone walking by
- Someone changes temp.

Internet

- Outside temp.
- Weather report
- Energy curtailment
- Day & time
- …

States

Factored states

- Estimated time to cool the house
- Someone home?
- How long till someone is coming home?
- Schedule
- ….

Example: Modern Vacuum Robot

Features are:

- Control via App
- Cleaning Modes
- Navigation
- Mapping
- Boundary blockers

iRobot Roomba 960

Read TechHive's review

$247.17 See It

on Amazon (Prime Eligible)

The Roomba 960's flawless navigation, stellar cleaning, and advanced features set it apart from all other robot vacuums.

iRobot's Roomba brand has become as synonymous with robot vacuum as Q-tips is with cotton swabs. The Wi-Fi-enabled Roomba 960 is ample evidence why. It turns a tiresome chore into something you can almost look forward to. With three cleaning modes and dirt-detecting sensors, it kept all the floor surfaces in our testing immaculate, and its camera-driven navigation and mapping were superb. Its easy-to-use app provides alerts and detailed cleaning reports. The ability to control it with Amazon Alexa and Google Home voice commands are just the cherry on top.

Source: https://www.techhive.com/article/3269782/best-robot-vacuum-cleaners.html

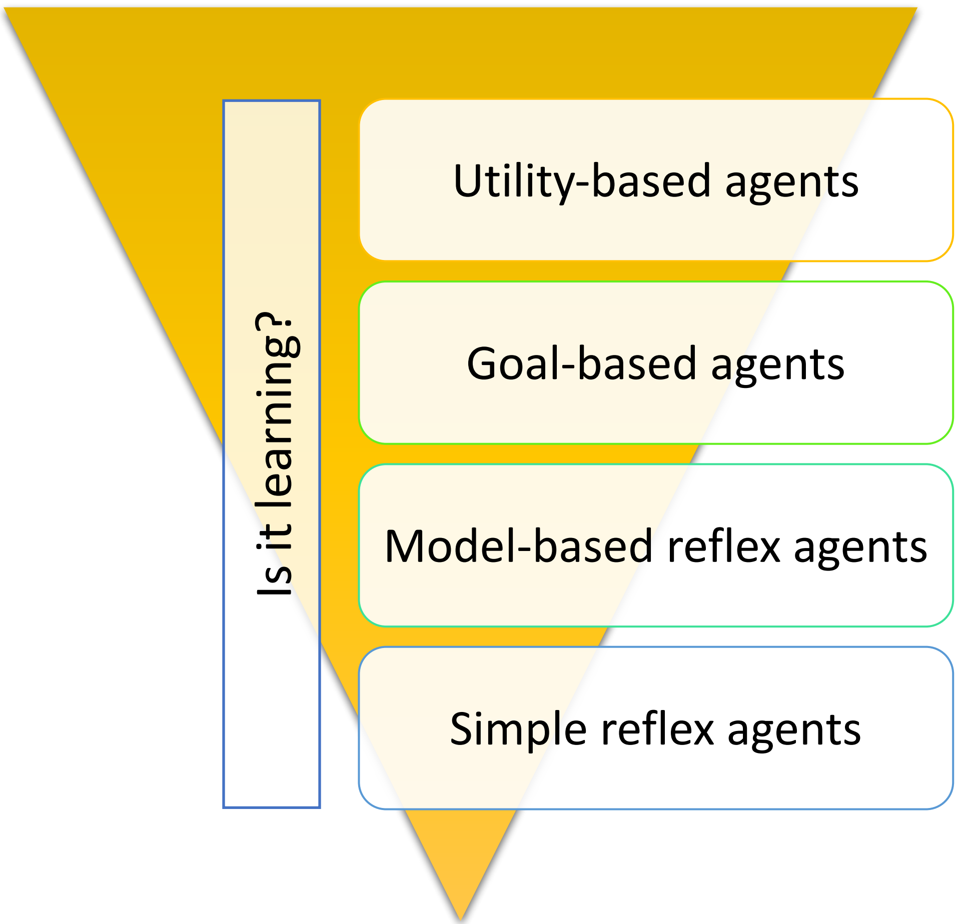
PEAS Description of a Modern Robot Vacuum

Performance measure	Environment	Actuators	Sensors

PEAS Description of a Modern Robot Vacuum

Performance measure	Environment	Actuators	Sensors
• Time to clean 95% • Does it get stuck?	• Rooms • Obstacles • Dirt • People/pets • …	• Wheels • Brushes • Blower • Sound • Communicate to server/app	• Bumper • Cameras/dirt sensor • Laser • Motor sensor (overheating) • Cliff detection • Home base locator

What Type of Intelligent Agent is a Modern Robot Vacuum?



Does it collect utility over time? How would the utility for each state be defined?

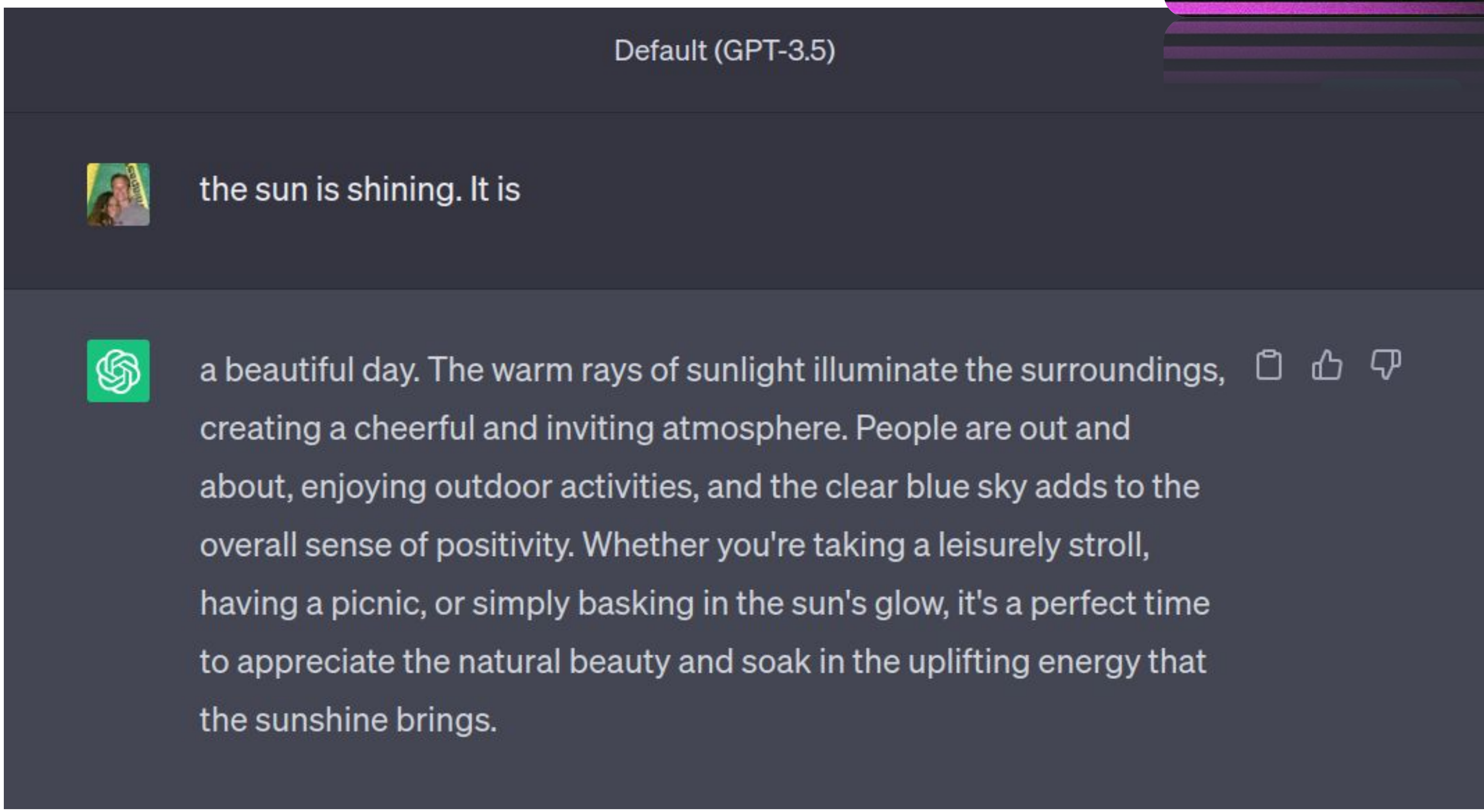
Does it have a goal state?

Does it store state information. How would they be defined (atomic/factored)?

Does it use simple rules based on the current percepts?

☑ Check what applies

Example: Large Language Models



PEAS Description of ChatGPT

Performance measure

- Goal
- Metrics
- Avoid

Environment

- Interaction environment
- Inputs
- Context

Actuators

- Text Generation
- Response Options

Sensors

- Input Detection
- Context Monitoring
- Language Understanding

What Type of Intelligent Agent is ChatGPT?

Is it learning?

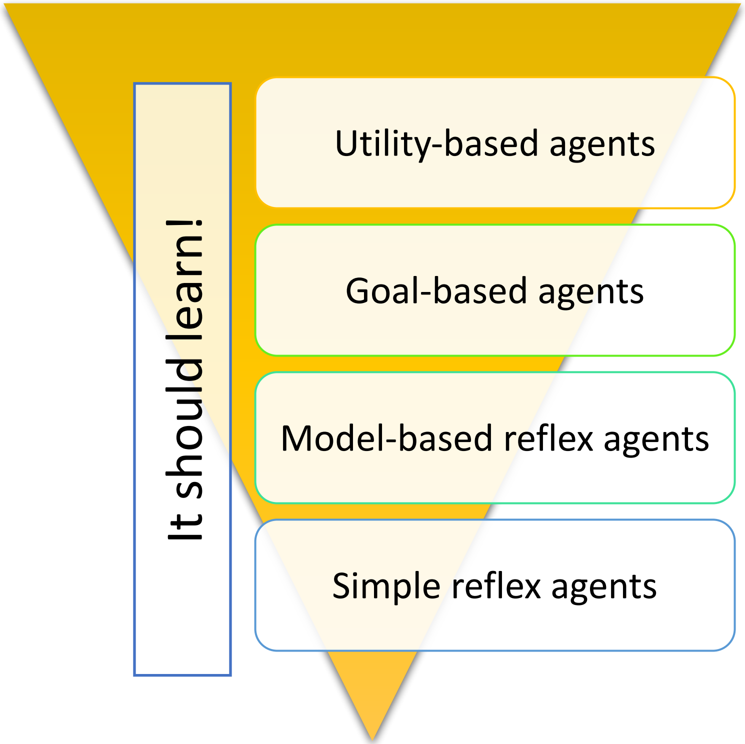
Agent type	Question
Utility-based agents	Does it collect utility over time? How would the utility for each state be defined?
Goal-based agents	Does it have a goal state?
Model-based reflex agents	Does it store state information. How would they be defined (atomic/factored)?
Simple reflex agents	Does it use simple rules based on the current percepts?

☑ Check what applies

Answer the following questions:

- Does ChatGPT pass the Touring test?
- Is ChatGPT a rational agent? Why?

Intelligent Systems a Sets of Agents: Self-driving Car



Make sure the passenger has a pleasant drive (not too much sudden breaking = utility)

High-level planning

Plan the route to the destination.

Remember where every other car is and calculate where they will be in the next few seconds.

React to unforeseen issues like a child running in front of the car quickly.

Low-level planning

Some Environment Types Revisited

Fully observable: The agent's sensors always show the whole **state**.

vs.

Partially observable: The agent only perceives part of the **state** and needs to remember or infer the test.

Deterministic:

a) **Percepts** are 100% reliable
b) Changes in the environment are completely determined by the current **state** of the environment and the agent's **action**.

vs.

Stochastic:

a) **Percepts** are unreliable (noise distribution, sensor failure probability, etc.). This is called a stochastic sensor model.
b) The **transition function** is stochastic leading to transition probabilities and a Markov process.

Known: The agent knows the **transition function**.

vs.

Unknown: The needs to **learn the transition function** by trying actions.

We will spend the whole semester on discussing algorithms that can deal with environments that have different combinations of these three properties.

Research Areas Inspired by Intelligent Agents

Intelligent agents inspire the research areas of modern AI

Search for a goal (e.g., navigation).	**Optimize** functions (e.g., utility).	Stay within given **constraints** (constraint satisfaction problem; e.g., reach the goal without running out of power)
Deal with **uncertainty** (e.g., current traffic on the road).	**Learn** a good agent program from data and improve over time (machine learning).	**Sensing** (e.g, natural language processing, vision)

Summary

- We explored
  - **KBS (Knowledge-Based Systems)**
    - Purpose: use stored knowledge to solve problems.
    - Components: Knowledge Base + Inference Engine + User Interface.
    - Framework: acquisition → representation → reasoning → explanation.
    - Reasoning: forward chaining (data-driven) & backward chaining (goal-driven).
    - Applications in medicine, business, education; limited by knowledge capture and maintenance.
  - **Intelligent Agents**
    - Definition: entities that perceive, reason, and act in an environment.
    - Rationality: act to maximize performance.
    - PEAS model: defines agent tasks by performance measure, environment, actuators, sensors.
    - Types of environments: deterministic vs. stochastic, static vs. dynamic, etc.
    - Types of agents: simple reflex, model-based, goal-based, utility-based, learning agents.
- *“KBS taught us how machines can reason like experts; Intelligent Agents show us how they can also perceive, act, and adapt — the foundation of modern AI.”*