



# **Intelligent Agents**

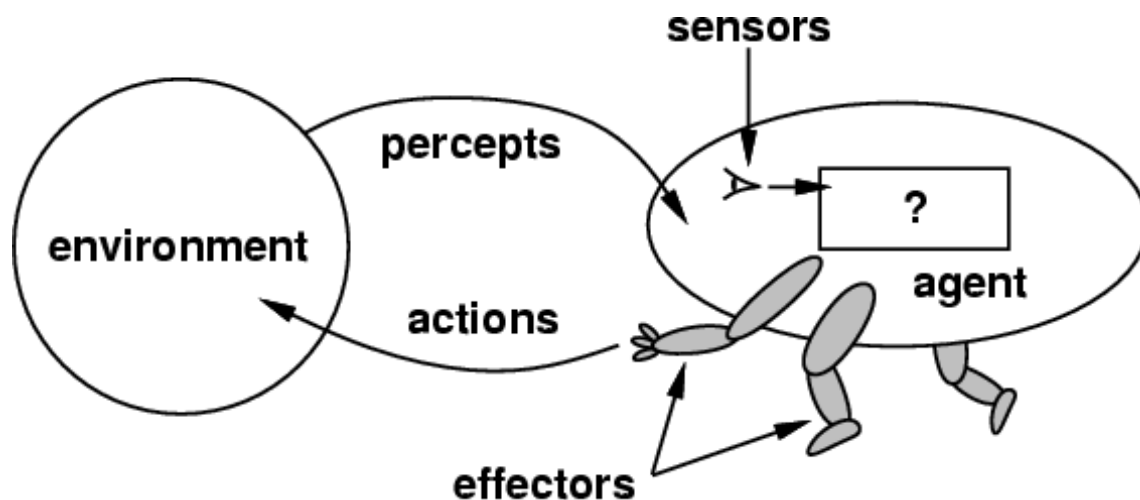
**Dr. Areej Fatemah Meghji**  
**areej.fatemah@faculty.muet.edu.pk**

# What is An Agent?

An “**Agent**” is anything that can be viewed as *perceiving* its environment through sensors and *acting* upon the environment through effectors.

Example:

- A Human Agent
- A Robotic Agent
- A Software Agent





# What is an Intelligent Agent?

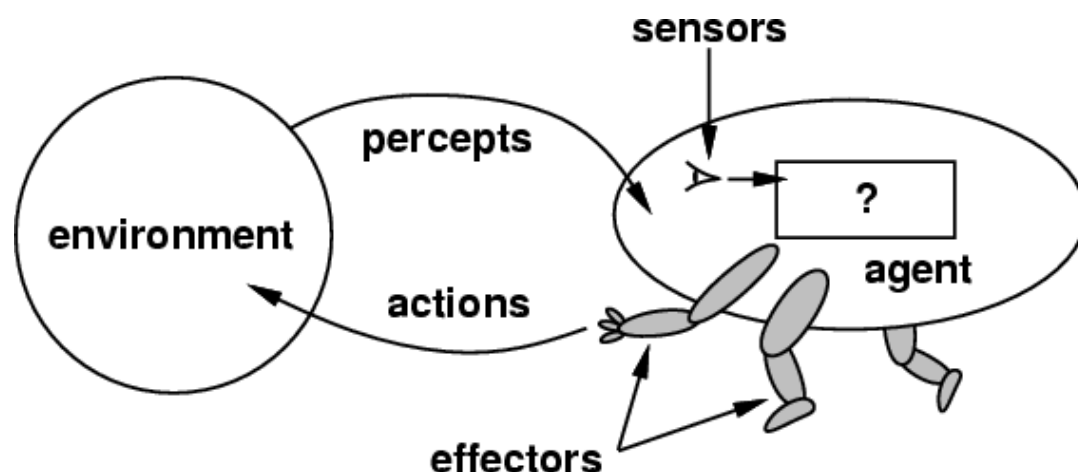
“An intelligent agent is an entity capable of combining *cognition*, *perception*, and *action* in behaving autonomously, purposively, and flexibly in some environment”

It **perceives** its environment, ***reasons*** about its goals, and **acts** upon the environment

## Supported by

- knowledge representation, search, inference, planning, uncertainty, learning, communication....

# Description of an Intelligent Agent



Main components

- Perception (sensors)
- Reasoning/cognition
- Action (actuators)

An **intelligent agent** perceives its environment via **sensors** and acts *rationally* upon that environment with its **effectors**.

Discrete agents receive percepts one at a time, and map them to a sequence of discrete actions.



# Sensors, Effectors Percepts and Actions

- **Humans**

- **Sensors:** Eyes (vision), ears (hearing), skin (touch), tongue (gustation), nose (olfaction), neuromuscular system (proprioception)
- **Effectors:** limbs, eyes, tongue, ...
- **Percepts:**
  - At the lowest level – electrical signals from these sensors
  - After preprocessing – objects in the visual field (location, textures, colors, ...), auditory streams (pitch, loudness, direction), ...
- **Actions:** lift a finger, turn left, walk, run, carry an object, ...



# Sensors, Effectors Percepts and Actions

- **Robot**

- **Sensors:** camera, infrared range finders, ...
- **Effectors:** motors

Percepts and actions need to be carefully defined, possibly at different levels of abstraction.

- **Software Agent**

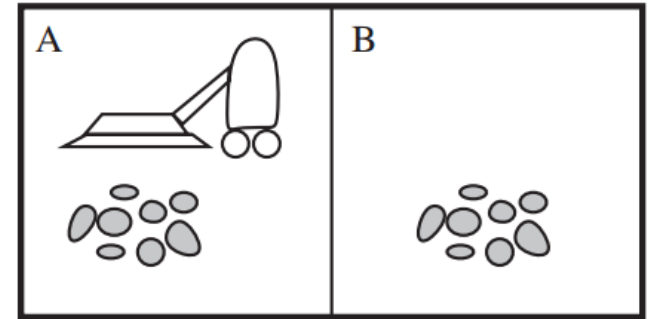
- a computer program designed to carry out some task on behalf of the user.
- receives keystrokes, file contents and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, printing documents and sending network packets.

- We can make the assumption that every agent can perceive its own actions (but not always the effects)

- **Percept:** Refers to the agent's perceptual input at any given instant.
- **Percept Sequence:** complete history of everything that the agent has ever perceived.
- *An agent's choice of action at any given instant can depend on the entire percept sequence observed to date.*
- **Agent Function:** this describes the agent behavior as it maps a given percept (percept sequence) to an action.  $[f = P \rightarrow A]$
- **Agent Program:** A concrete implementation running on the agent architecture. The agent program runs on the physical architecture to produce  $f$ .

Agent = Architecture + Program

## Example: Vacuum Cleaner



- Percept: Location and status (e.g. [A, Dirty])
- Actions: Move Left, Move Right, Vacuum
- Agent Function: **(determined based on the agent program)**
  - If current block is dirty, vacuum; else move to the next block.

### Percept

[A, Dirty]

[A, Clean]

[B, Dirty]

[B, Clean]

### Action

Vacuum

Move Right

Vacuum


Move Left





# Considerations:

- How should an agent act?
- How does an agent know it is doing what it is supposed to do?
- How does it know it has done its job well?
- What happens if an agent is put in a different environment?

- 
- **Performance Measure** embodies an objective criterion for success of an agent's behavior.
    - When the agent is placed in an environment it generates a sequence of actions according to the percepts it receives.
    - This sequence of actions causes the environment to go through a sequence of states.
    - If the sequence of states is “desirable”, then the agent has performed well.
  - **Utility Function** maps a set of states to the set of real numbers.
    - In other words, given a particular state of the world, an agent is able to use its utility function to derive a score, or utility value, that figure out how “**happy**” it is in that state or how successful it has been if upon reaching that state.



# Goal of Intelligent Agents

- *Maximize expected performance*

An ideal agent should, for each possible percept sequence, do whatever actions will maximize its expected performance measure based on

1. the percept sequence, and
2. its built-in and acquired knowledge.

It maximizes the likelihood of success, given its information

- How is “the right thing” chosen?
  - Possible actions (from which to choose)
  - Percept sequence (current and past)
  - Knowledge (static or modifiable)
  - Performance measure (*based on* goals – defines success)

**AI strives to build RATIONAL AGENTS**



# Properties of Environments

## 1. Observability: (Fully Observable – Partially Observable)

- If an agent's sensors give it access to the complete state of the environment needed to choose an action, the environment is **Fully Observable**, agent doesn't keep track of the other changes in the environment. E.g., chess.
- In **Partially observable** environment, only the related states and features are accessed and observed. E.g., Bridge, Solitaire.

## 2. Determinism: (Deterministic – Stochastic)

- An environment is **Deterministic** if the next state of the environment is completely determined by the current state of the environment and the action of the agent. If the environment is deterministic except for the actions of other agents, then the environment is **strategic**.
- In a **Stochastic** environment, there are multiple, unpredictable outcomes. A partially observable, deterministic environment will appear stochastic to the agent.



# Properties of Environments

## 3. Episodicity: (Episodic – Sequential)

- In an **Episodic** environment the agent's experience is divided into a set of episodes such that the subsequent episodes do not depend on what actions occurred in previous episodes.
- Each episode consists on agent perceiving and then performing a single action. E.g: finding defective parts on an assembly line.
- Episodic environments do not require the agent to plan ahead.
- In a **Sequential** environment, the agent engages in a series of connected episodes. The current decision could effect all future decisions.
- Playing chess, Othello, taxi driving are all sequential as short term actions can have long term consequences .

## 4. Dynamism: (Static – Dynamic)

- A **Static** environment does not change while the agent is thinking.
- The passage of time as an agent deliberates is irrelevant. The agent doesn't need to observe the world during deliberation.
- A **Dynamic** environment changes over time independent of the actions by the agent, timely action is required to make effective action.



# Properties of Environments

## 5. Continuity: (Discrete – Continuous)

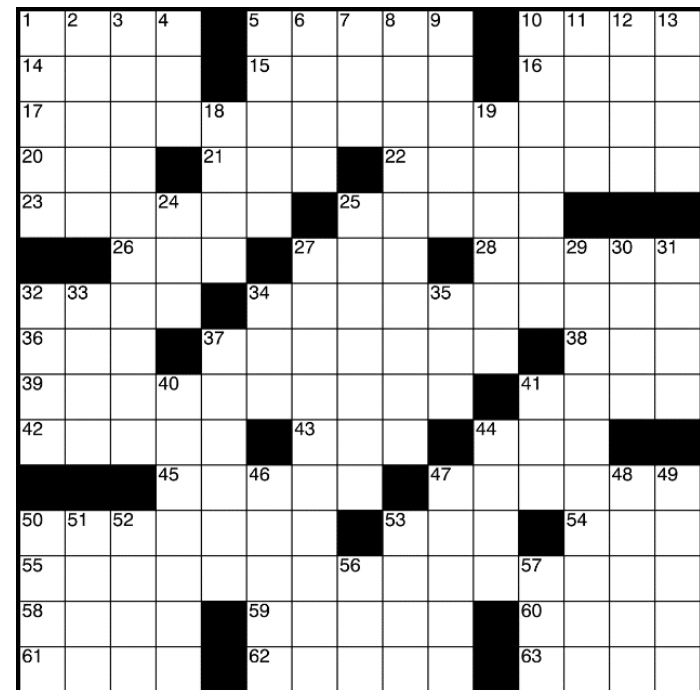
- If the number of distinct percepts and actions is limited, the environment is **Discrete**, otherwise it is **Continuous**.
- Discrete state environment may be a board game as chess or Othello as it has a finite number of distinct states. Taxi driving is a continuous time - continuous state problem.

## 6. Presence of Other Agents:

- If the environment contains **Multi-Agents**, the agent needs to be concerned about strategic, game-theoretic aspects of the environment (for either cooperative or competitive agents)
- Most engineering environments works with **Single Agent** properties, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.

# Task Environment: Cross World Puzzle

- Observable
  - Fully Observable
- Determinism
  - Deterministic
- Episodicity
  - Sequential
- Dynamism
  - Static
- Continuity
  - Discrete
- Presence of Agents
  - Single



# Task Environment: Taxi Driving



- Observable
  - Partially Observable
- Determinism
  - Stochastic
- Episodicity
  - Sequential
- Dynamism
  - Dynamic
- Continuity
  - Continuous
- Presence of Agents
  - Multi Agent



# Task Environment: Part Picking Robot



- Observable
  - Partially Observable
- Determinism
  - Stochastic
- Episodicity
  - Episodic
- Dynamism
  - Dynamic
- Continuity
  - Continuous
- Presence of Agents
  - Single Agent



# Task Environment: Poker

- Observable
  - Partially Observable
- Determinism
  - Stochastic
- Episodicity
  - Sequential
- Dynamism
  - Static
- Continuity
  - Discrete
- Presence of Agents
  - Multi Agent



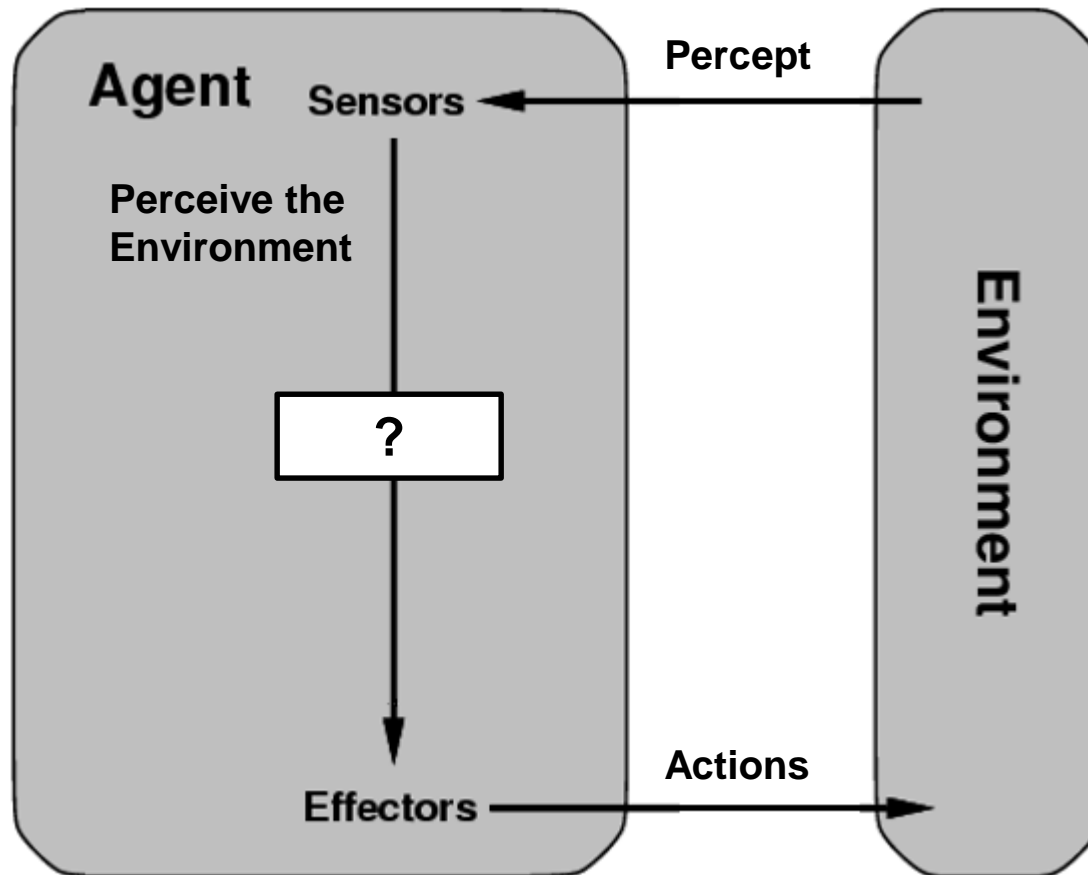
# Task Environment: Image Analysis

- Observable
  - Fully Observable
- Determinism
  - Deterministic
- Episodicity
  - Episodic
- Dynamism
  - Static
- Continuity
  - Discrete
- Presence of Agents
  - Single

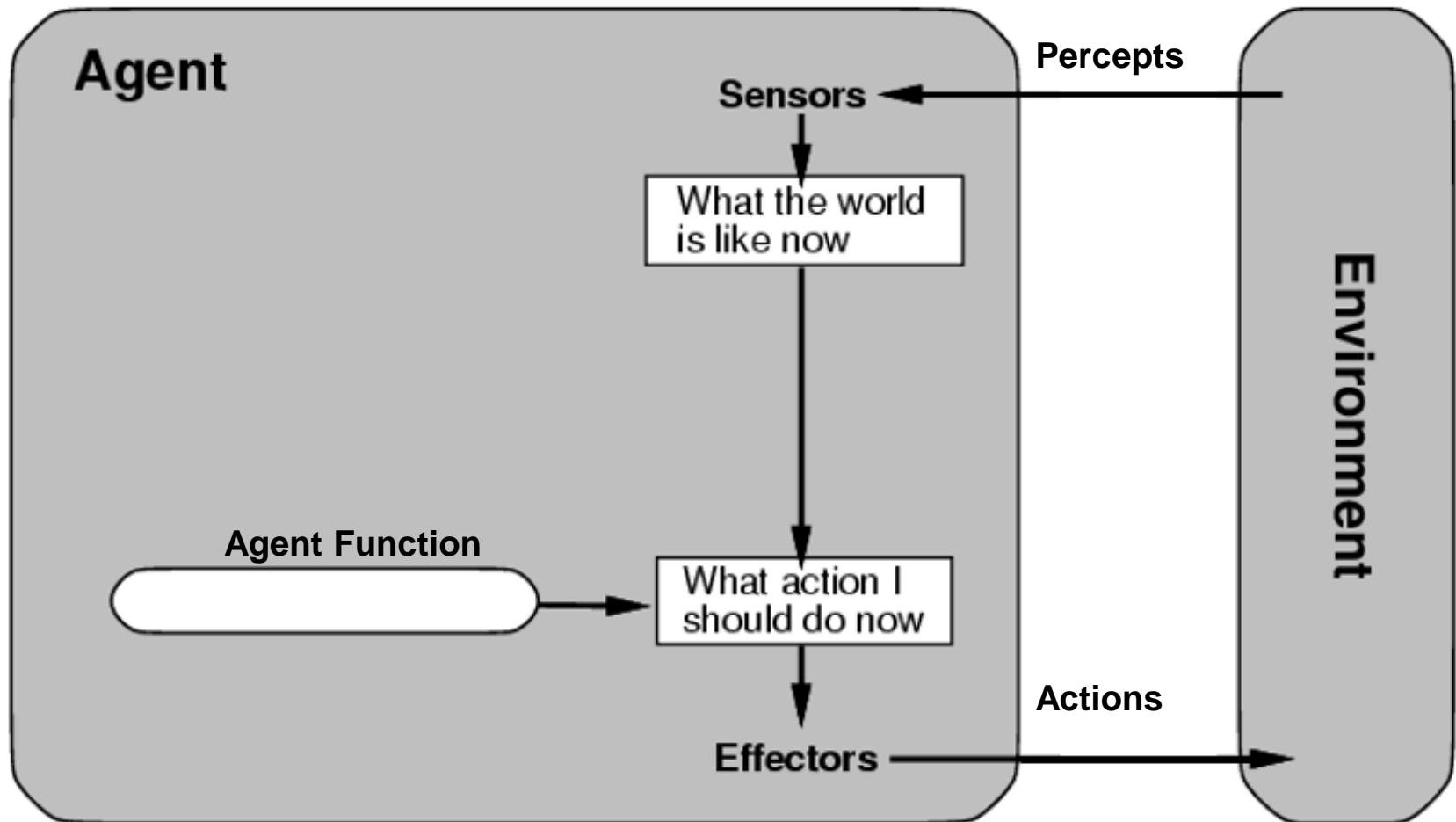
# Example of Task Environment and Characteristics

Task Env.	Observab.	Determ. / Stochastic	Episodic / Sequent.	Static / dynamic	Discrete / Cont.	Agents
<b>Crossword puzzle</b>	fully	determ.	sequential	static	discrete	single
<b>Taxi driv.</b>	partial	stochastic	sequential	dynamic	cont.	multi
<b>Part Picking Robot</b>	partial	stochastic	episodic	dynamic	cont.	single
<b>Poker</b>	partial	stochastic	sequential	static	discrete	multi
<b>Image Analysis</b>	fully	determ.	episodic	static	discrete	single

# Agent:



# Agent:





# Basic Agent Types

## 1- Table-driven agents

use a percept sequence/action table in memory to find the next action. They are implemented by a (large) **lookup table**.

## 2- Simple reflex agents

are based on **condition-action rules**, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.

## 3- Agents with memory

have **internal state**, which is used to keep track of past states of the world.



# Basic Agent Types

## 4- Agents with goals

are agents that, in addition to state information, have **goal information** that describe desirable situations. Agents of this kind take future events into consideration.

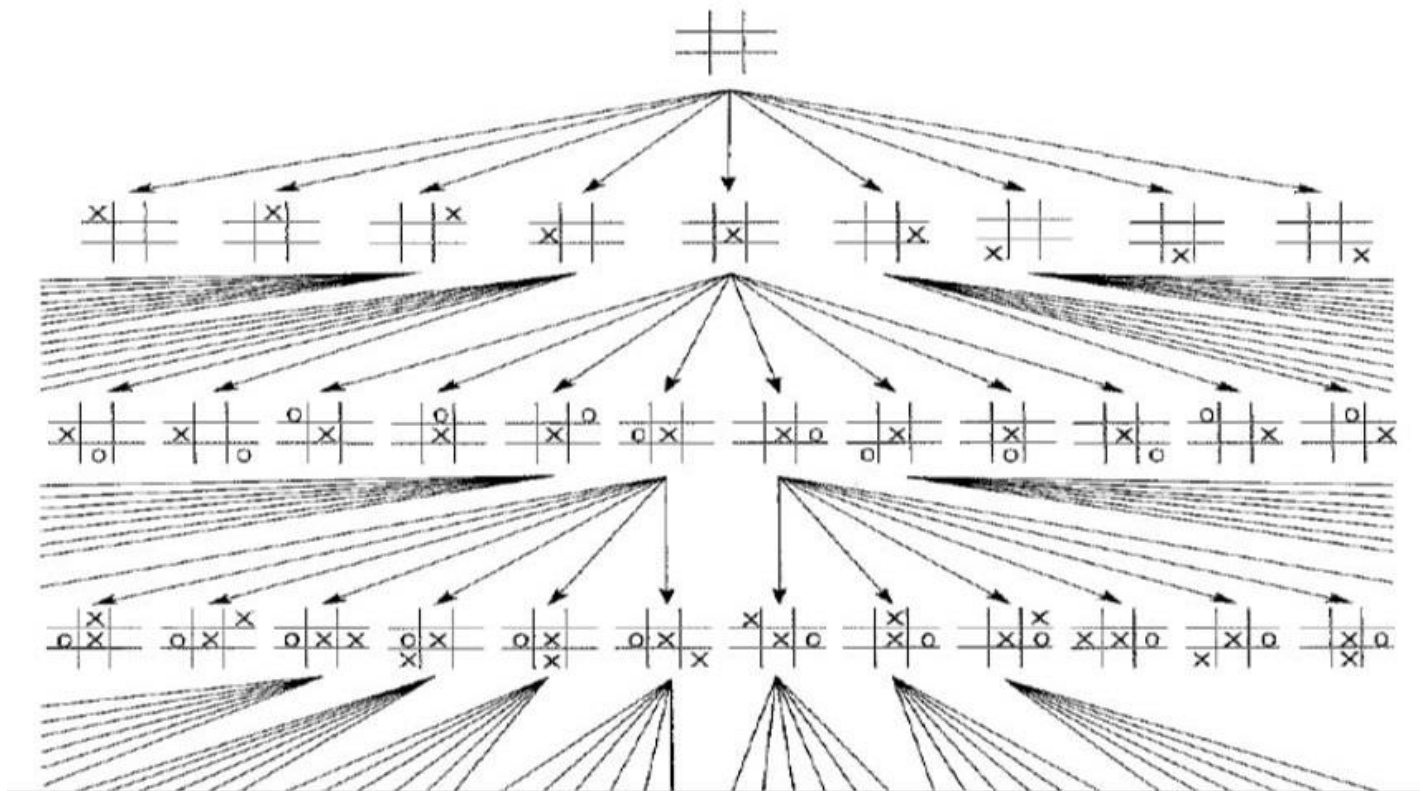
## 5- Utility-based agents

base their decisions on **classic axiomatic utility theory** in order to act rationally.



# 1. Table-driven agents

- **Table lookup of percept-action pairs mapping** from every possible perceived state to the optimal action for that state





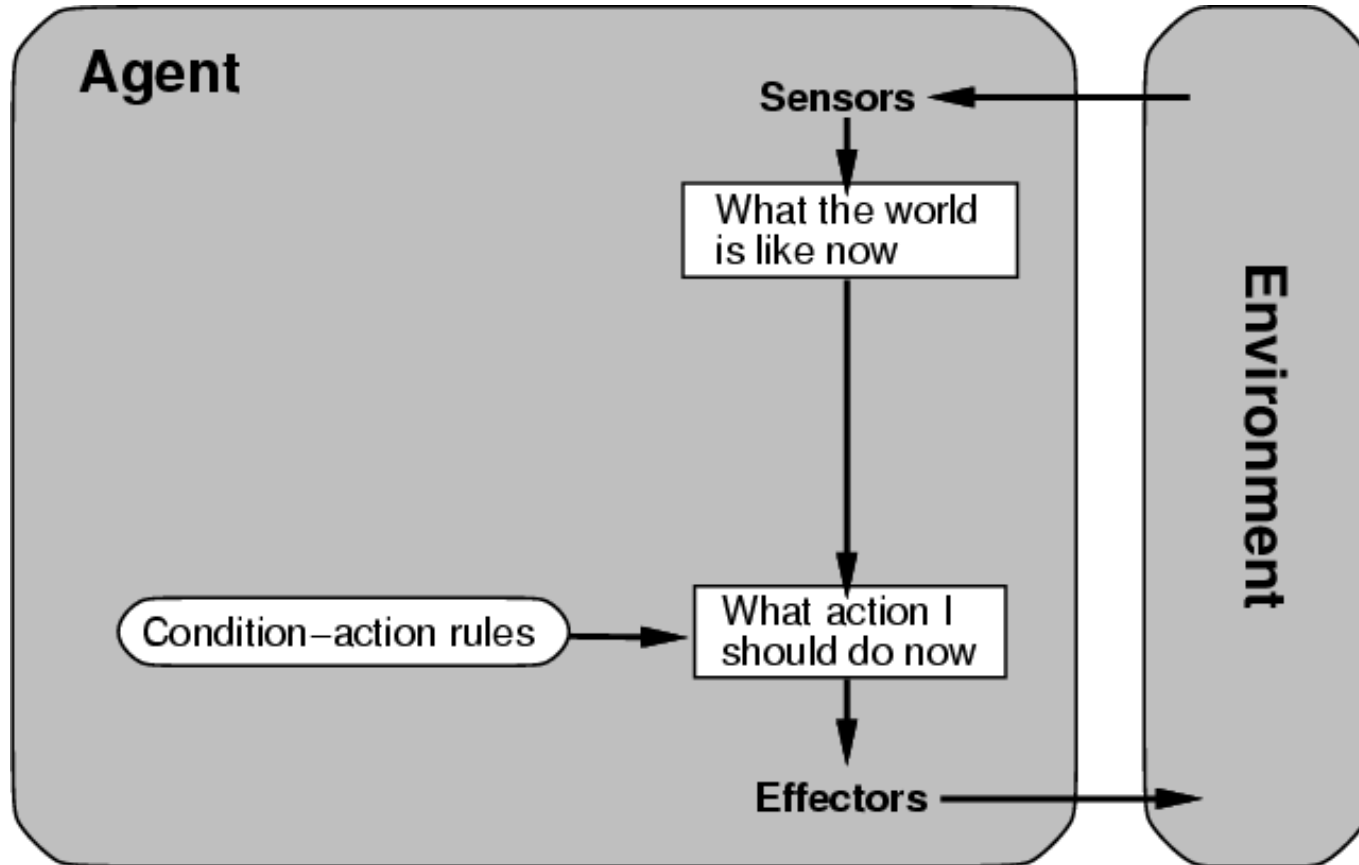
# 1. Table-driven agents

- **Problems**

- Too big to generate and to store (Chess has about  $10^{150}$  states)
- Non adaptive to changes in the environment.
- Looping: Can't make actions conditional on previous actions/states.

## 2. Simple reflex agents

- Possibly the simplest kind of agent. These agents select actions on the basis of current percept, ignoring the rest of the percept history.





## 2. Simple reflex agents

- **Problems**

- Still usually too big to generate and to store
- No knowledge of non-perceptual parts of state
- Can't adapt to changes in the environment; require collection of rules to be updated if change occurs
- Can't make actions conditional on previous state



## 2. Simple reflex agents

- A reactive agent does not tend to perform well when its environment changes or when something happens that it has not been told about.
- New rules can of course be written to deal with such situations, but it might be more desirable to have an agent that can learn to adapt to new situations.



### 3. Agents with Memory (Model based Agent)

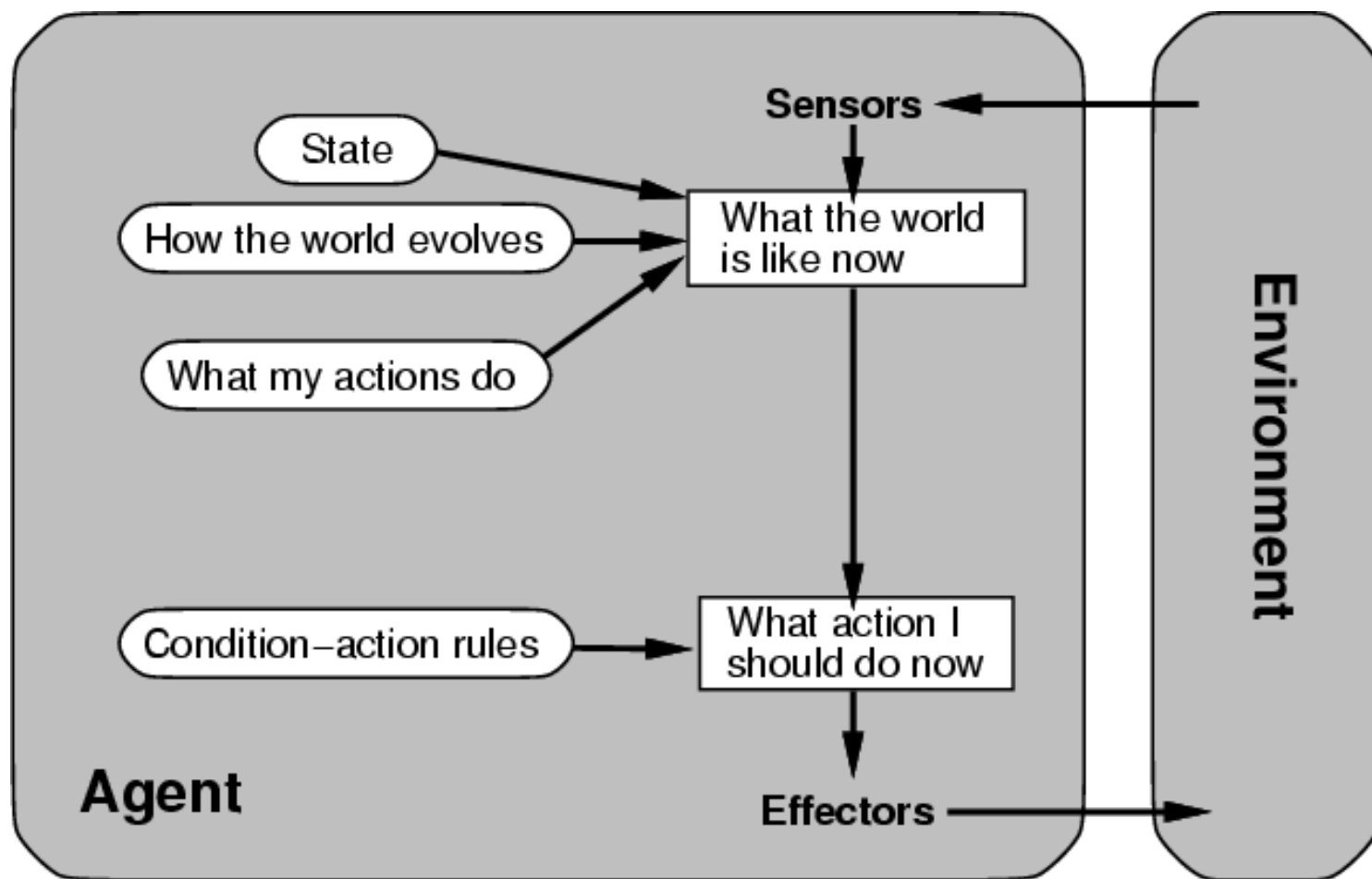
- An effective way to handle the shortcomings of the previous two agents is to fashion an agent that ***keeps track of the parts of the world that it can't see right now.***
- The agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.



### 3. Agents with Memory (Model based Agent)

- Encode “internal state” of the world to remember the past as contained in earlier percepts.
- Needed because sensors do not usually give the entire state of the world at each input, so perception of the environment is captured over time. “State” is used to encode different "world states" that generate the same immediate percept.
- Requires ability to represent change in the world; one possibility is to represent just the latest state, but then can't reason about hypothetical courses of action.

# Architecture for an Agent with memory



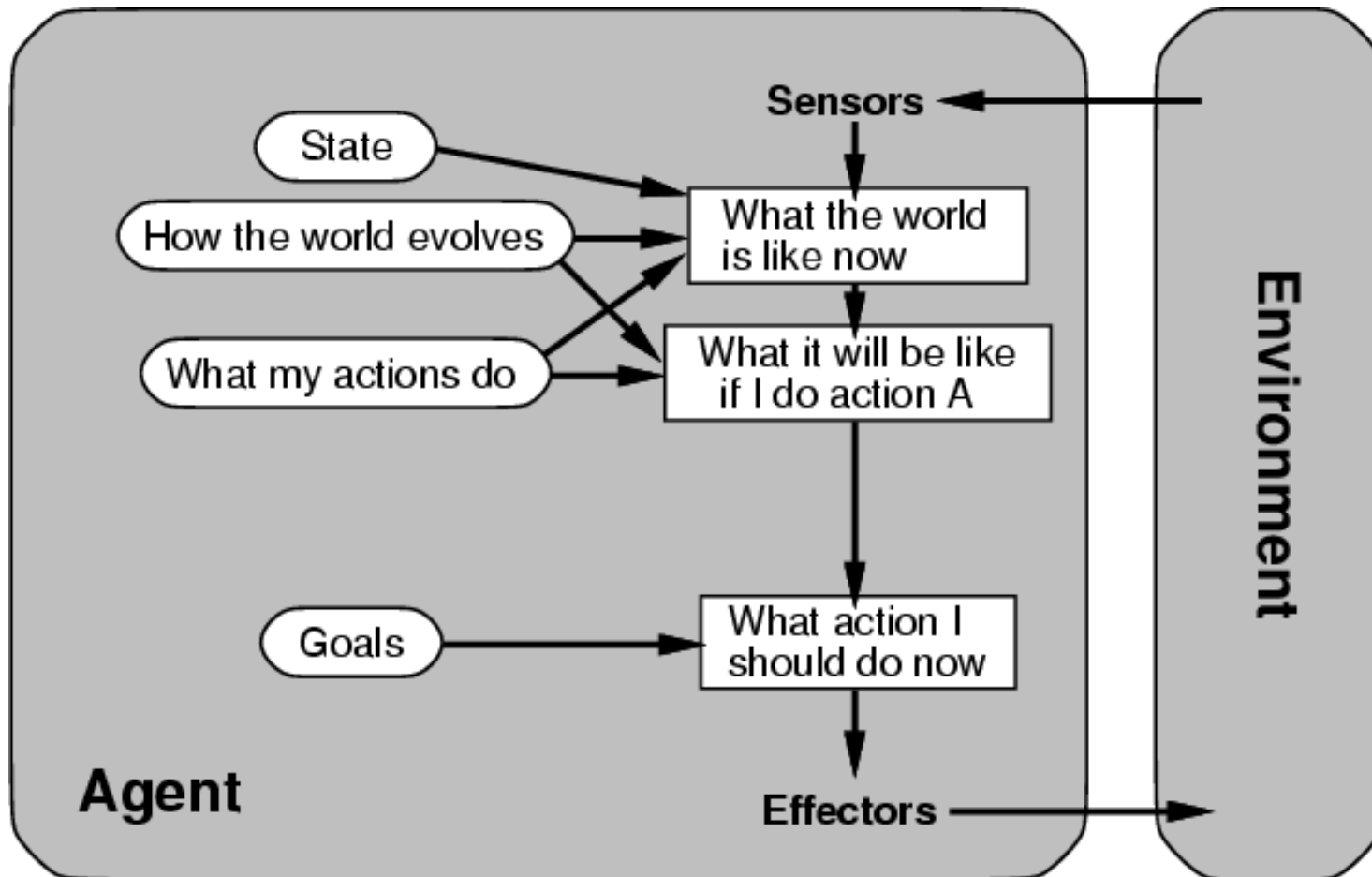




## 4. Goal-based agents

- Choose actions so as to achieve a (given or computed) goal.
- A goal is a description of a desirable situation.
- Keeping track of the current state is often not enough – need to add goals to decide which action/outcome is desirable.
- **Deliberative** instead of **reactive**.
- May have to consider long sequences of possible actions before deciding if goal is achieved – involves consideration of the future, “*what will happen if I do...?*”

# Architecture for goal-based agent





## 5. Utility-based Agents

**When there are multiple possible alternatives,  
how to decide which one is best?**

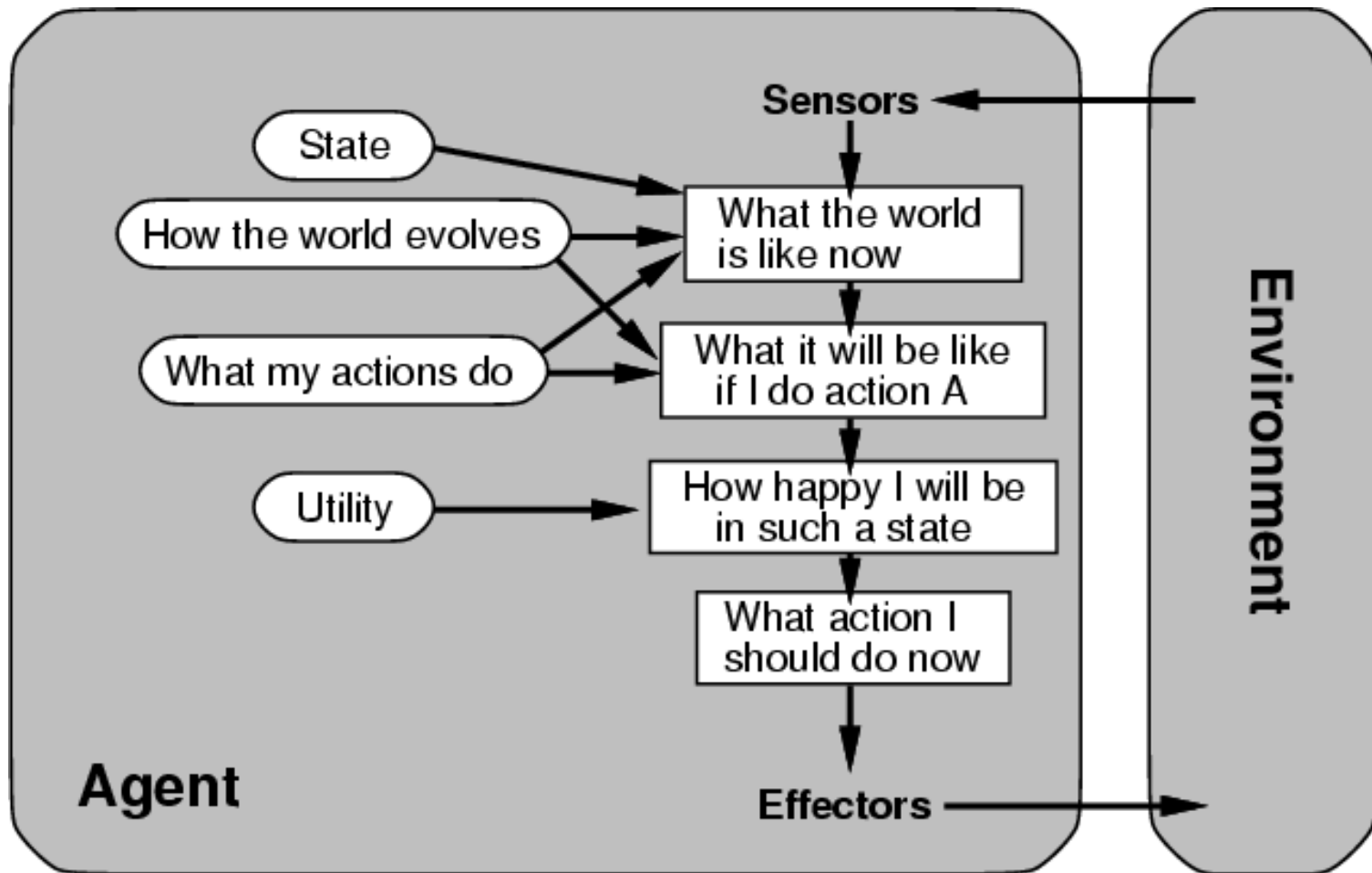
- A goal specifies a crude distinction between a happy and unhappy state
- Often, a more general performance measure that describes “degree of happiness” is required,
- Utility function **U: State → Real** indicating a measure of success or happiness when at a given state.
- Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).



## 5. Utility-based Agents

- A utility-based agent is similar to a goal-based agent, but in addition to attempting to achieve a set of goals, the utility-based agent is also trying to maximize some utility value.
- The utility value can be thought of as the happiness of the agent, or how successful it is being. It may also take into account how much work the agent needs to do to achieve its goals.

# Architecture for Utility-Based Agent





# Possible Properties

- Agents are **autonomous** – they act on behalf of the user
- Agents can **adapt** to changes in the environment
- Agents don't only act **reactively**, but sometimes also **proactively**
- Agents have **social ability** – they communicate with the user, the system, and other agents as required
- Agents also **cooperate** with other agents to carry out more complex tasks than they themselves can handle
- Agents **migrate** from one system to another to access remote resources or even to meet other agents



# Task Environment

- Before designing an intelligent agent its “task environment” needs to be specified:
- **PEAS** description of an agent
  - Performance measure
  - Environment (Properties)
  - Actuators
  - Sensors

# PEAS

Agent = Automated taxi driving system



- **Performance Measure:**

- Safe, fast, comfortable trip, no rule violations, less time and cost..

- **Environment:**

- Urban streets, freeways, traffic, pedestrians, weather, customers,..
- Properties?

- **Actuators/Actions:**

- Steer, accelerate, brake, horn, signal, speak/display, ...

- **Sensors:**

- Cameras/video, speedometer, GPS/navigation, sonar, odometer,..



# PEAS

Agent = Part Picking Robot



- **Performance Measure:** Percentage of parts in correct bins ...
- **Environment:** Conveyor belt with parts, bins, ...
  - Properties?
- **Actuators/Actions:** Jointed arm and hand , ...
- **Sensors:** Camera, joint angle sensors , ...





# Class Task

1. Generate the PEAS description for a chess playing agent. Keep in consideration that the game is being played with a clock.
2. Generate the PEAS description for an Automatic Door Operator.



# PEAS

## Agent = Chess with a Clock

- **Performance Measure:** How well the agent plays the game (winning, losing, drawing) and how it manages time.
- **Environment:** chessboard state, game rules, the clock, and the opponent's actions
- **Actuators/Actions:** The possible moves the agent can make, and how it interacts with the clock (Robotic Arm for physical chessboard, Software Move Execution, Clock Control Effector)
- **Sensors:** The perception of the board, the opponent's moves, and the clock status. (Camera or Vision System, Piece Detection, Clock Sensor)

# Task Environment: Chess with a Clock

- Observable
  - Fully Observable
- Determinism
  - Strategic
- Episodicity
  - Sequential
- Dynamism
  - Semi Static
- Continuity
  - Discrete
- Presence of Agents
  - Multi Agent





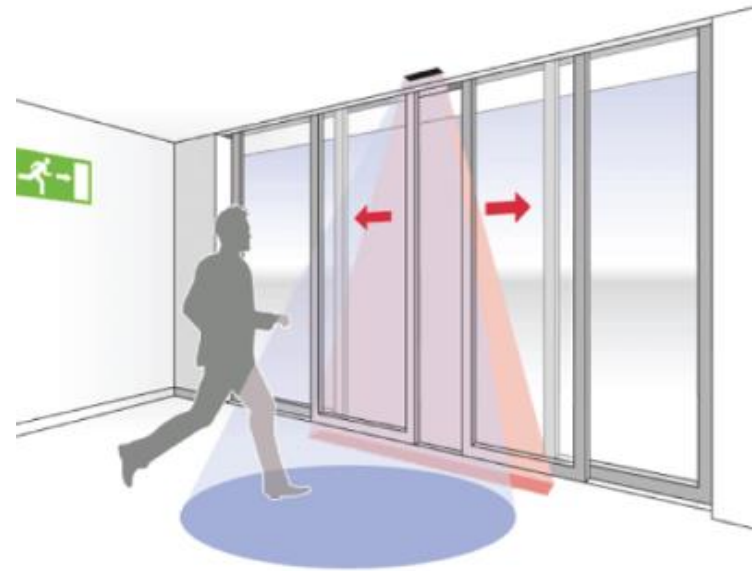
# PEAS

## Agent = Automatic Door Opener

- **Performance Measure:** Efficiency in Door Operation, Accuracy of Detection, Energy Efficiency, Safety, Responsiveness to Time
- **Environment:** Physical Space around the door, Presence of people or objects, Other moving objects
- **Actuators/Actions:** Motor: A motor to open and close the door, Safety Mechanism (Fail-Safe)
- **Sensors:** Motion Sensors: These sensors (e.g., infrared sensors), Pressure Sensors, Proximity sensor, safety sensors, Door Position Sensors)

# Task Environment: Automatic Door Operator


- Observable
  - Partially Observable
- Determinism
  - Deterministic/**Stochastic**
- Episodicity
  - Episodic
- Dynamism
  - Dynamic
- Continuity
  - Continuous
- Presence of Agents
  - Single





# Summary

- An **Agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- An **Ideal agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An **Autonomous agent** uses its own experience rather than built-in knowledge of the environment by the designer.

- 
- The most challenging environments are:
    - partially observable, stochastic, sequential, dynamic, and continuous, and environments with multiple agents.
  - **Reflex agents** respond immediately to percepts.
  - **Goal-based agents** act in order to achieve their goal(s).
  - **Utility-based agents** maximize their own utility function.
  - **Representing knowledge** is important for successful agent design.