Agents and Enviroments

CSC 480 - Artificial Intelligence

Rodrigo Canaan
Assistant Professor

Cal Poly, April 3, 2025

Goals

- Define agents and environments
- Define what it means for an agent to be rational
- Build a model for agent-environment interaction
- Describe the different types of agents

Motivation

What is AI?

- AI can be defined as the effort to make agents that act rationally
  - Russel & Norvig's "Standard model": "AI (…) the study and construction of agents that **do the right thing"**.

- But what does it mean to act rationally?

What is Rationality?

Rationality

From Wikipedia, the free encyclopedia

Not to be confused with rationalism.
"Rational" redirects here. For the mathematical property of some numbers, see Rational number. For other uses, see Rational (disambiguation).

Rationality is the quality of being guided by or based on reasons. In this regard, a person acts rationally if they have a good reason for what they do or a belief is rational if it is based on strong evidence. This quality can apply to an ability, as in rational animal, to a psychological process, like reasoning, to mental states, such as beliefs and intentions, or to persons who possess these other forms of rationality. A thing that lacks rationality is either *arational*, if it is outside the domain of rational evaluation, or *irrational*, if it belongs to this domain but does not fulfill its standards.

What is Rationality?

From Russel & Norvig (fourth edition):

"For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has"

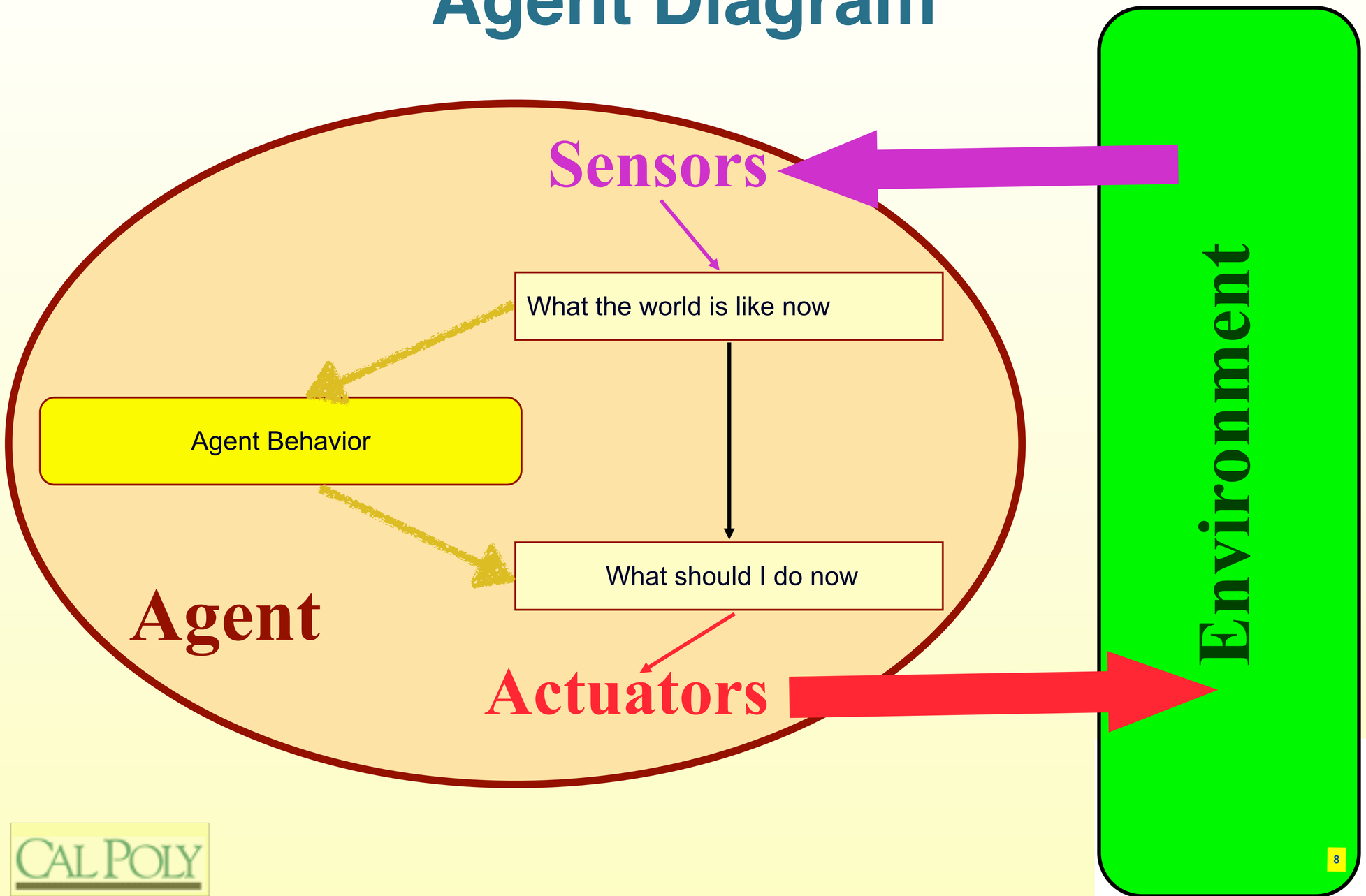
Therefore rationality depends:

- On the performance measure
- On the percept sequence
- On the agent's prior knowledge
- On the actions available to the agent

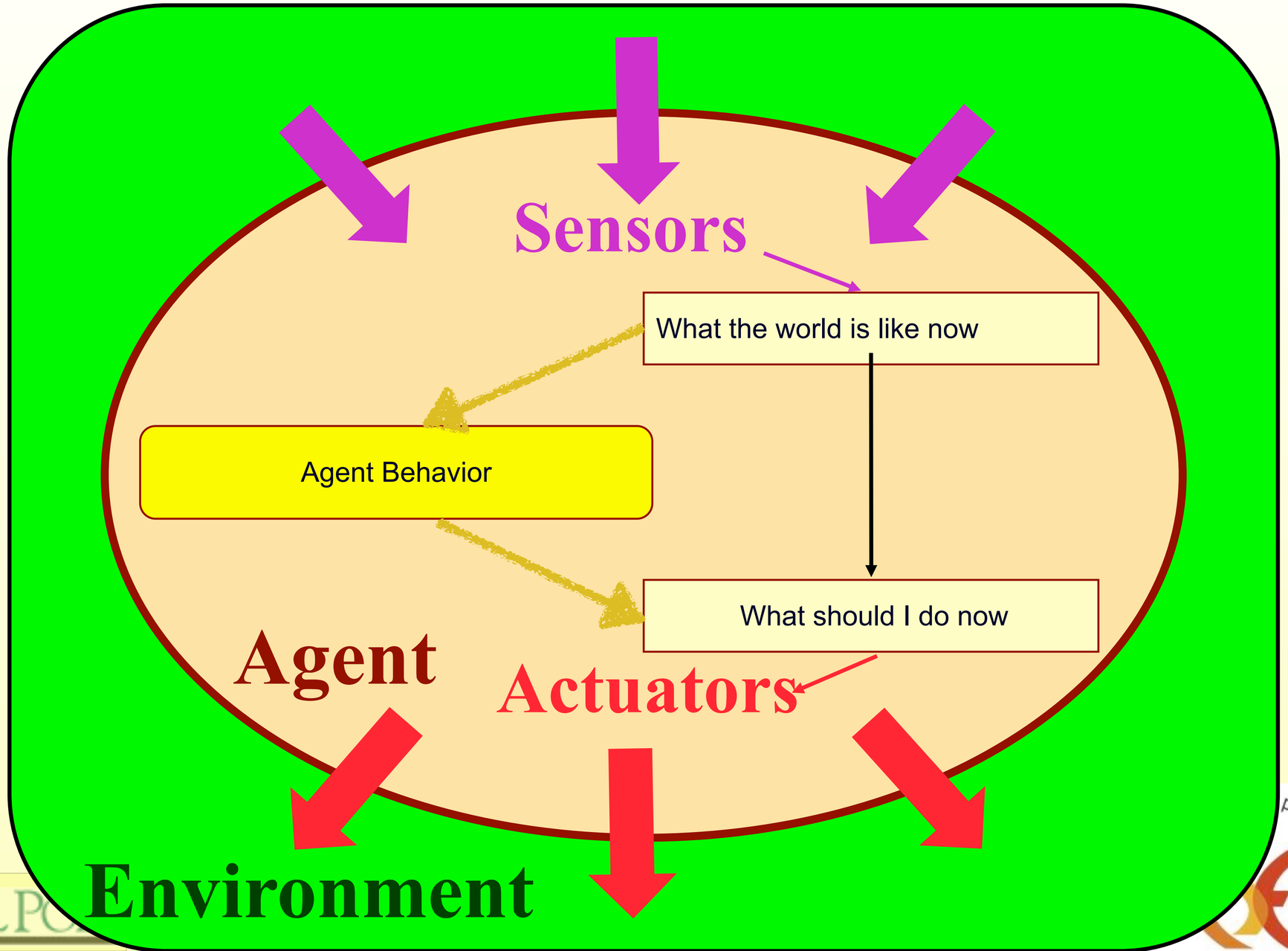
Agents, Environments and Rationality

Agents and environments are useful abstractions to model an entity (the agent) perceiving the world (the environment) and trying to act rationally in it.

Agent Diagram



Agent Diagram 2



Agents and Environments

A simple agent-environment model

- At a given time *t*, the environment can be in one of many states

$$E = \{e_1, e_2, \cdots, \varnothing\}$$

- Each agent has a selection of actions to choose from

$$A = \{a_1, a_2, \cdots\}$$

- A run is a sequence of interleaved environment states and agent actions

$$\mathrm{H} : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

note: let's assume for now that a run always ends in a state (different from the book)
note2: the book uses R for the set of all runs, let's use H instead

A simple environment-agent model

- A **transition function** takes a run and an agent action and returns the next state

$$T : H \times A \to E$$

$$(r, a) \mapsto e$$

- The transition function describes the environment's behavior

Example: thermostat

- Simple behavior:
  - Temperature below *t* -> **heat on**
  - Temperature above *t* -> **heat off**

- How would you define E (states), A (actions) and T (transitions) for this example?

Randomness in an environment

- In general, the transition function may be random, so instead of mapping directly to a state, it maps to a probability over states
- In general, it also may also depend on the whole run (history)
- If it happens to depend only on the last state, we say the system has the **Markov Property** and we can write more succinctly

$$T : E \times A \to E$$

$$(e_1, a) \mapsto e_2$$

(See also: Markov Decision Process (Wiki))

Example: Markov Property

- The Markov property is typically not intrinsic to an envioronment, but to a specific *representation* (or description, model) of its states.

- Example: suppose we're trying to determine the trajectory of a thrown baseball under the action of no forces except gravity
  - State represented as (x, y, z) position: non-markovian
  - State represented as (x, y, z) and a velocity vector: markovian

- In practice: try to model your environment such that the Markov holds. If not possible, mitigate with heuristics, observation windows….

PEAS description

PEAS Description of Task Environments

A task environment is the "problem" the agent is trying to solve

It is usually described by PEAS

- **Performance Measures**
  - used to evaluate how well an agent solves the task at hand
- **Environment**
  - surroundings beyond the control of the agent
- **Actuators**
  - determine the actions the agent can perform
- **Sensors**
  - provide information about the current state of the environment

Environment Properties

Environments

- **determine to a large degree the interaction between the “outside world” and the agent**
  - the “outside world” is not necessarily the “real world” as we perceive it
    - it may be a real or virtual environment the agent lives in
- **in many cases, environments are implemented within computers**
  - they may or may not have a close correspondence to the “real world”
  - simulations or emulations of the real world
    - e.g., flight simulators, car racing
  - artificial worlds with little or no resemblance to the real world

Environment Properties

- **fully observable vs. partially observable**
  - Do sensors capture all relevant information from the environment?
  - Example: Chess vs Poker
- **deterministic vs. stochastic (non-deterministic)**
  - Are changes in the environment predictable?
  - Example: Chess vs Risk
- **episodic vs. sequential (non-episodic)**
  - Is the environment divided in atomic (single percept-action) episodes?
  - Episodes may be of a certain length
  - Example: Image classification (episodic) vs chess (length ~50) vs stock trading (fully sequential / arbitrary length)
-

Environment Properties (cont.)

- **static vs. dynamic**
  - Does the environment change while the agent is “thinking”?
  - Example: chess vs cooking/driving/football
- **discrete vs. continuous**
  - What is the granularity of percepts, actions and time?
  - Example: Chess vs cooking/driving/football
- **single vs. multiple agents**
  - How many agents interact with the environment?
  - Are these agents competing, cooperation
  - Example: Solitaire (single-agent) vs Chess (2 agents) vs Football (a few dozen agents) vs World of Warcraft (arbitrary # of agents)

https://i.redd.it/txok0df6wnyz.jpg

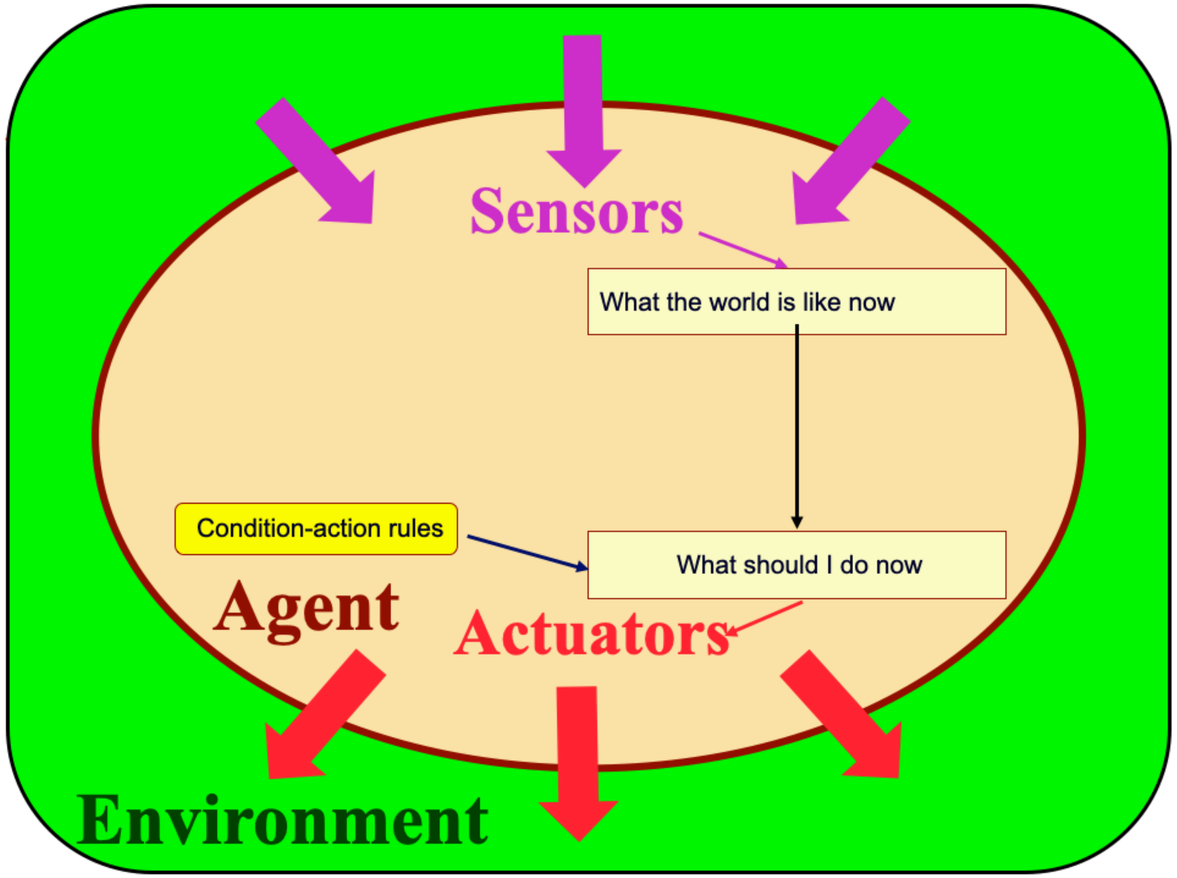
Exercise: properties of environment (Fig 2.6 of book)

- Assign to each task environment below its characteristics:
  - Task environments
    - Crosswords
    - Chess with a clock
    - Poker
    - Backgammon
    - Taxi driver
    - Image Analysis
    - Medical Diagnosis
    - Customer service chatbot
  - Characteristics
    - Fully vs Partially Observable
    - Number of agents
    - Deterministic vs Stochastic
    - Episode length
    - Static vs Dynamic
    - Discrete vs Continuous

Types of agents

Reflex agents

- The thermostat is an example of a simple reflex agent



Example: complex thermostat

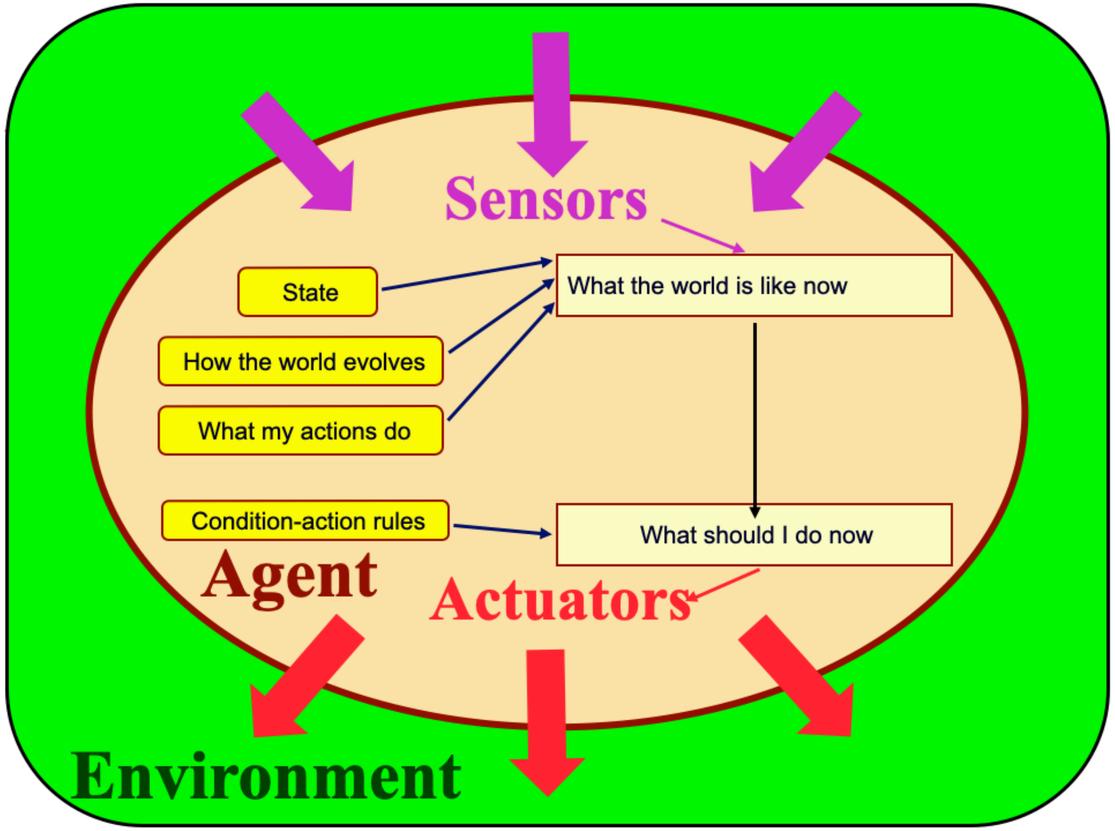
What if the thermostat also needs to:

- Balance temperature against energy usage
- Heat the room only if someone is present
- Anticipate the user's routine to pre-heat the room before user's arrival?

Model-based reflex agents

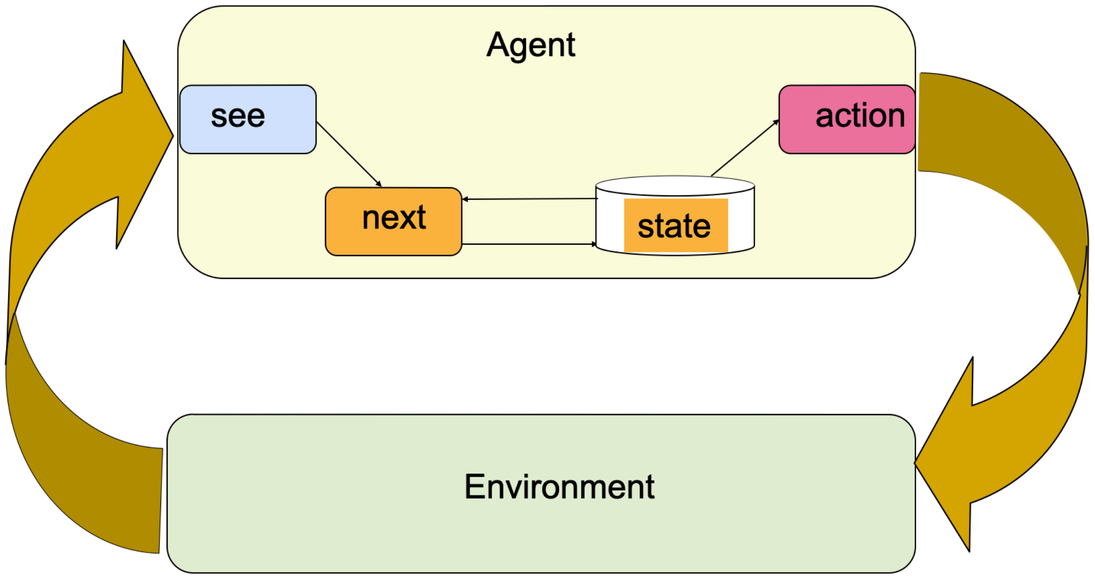
- Agent maintains a model of environment

- Encodes prior knowledge:
  - How long does it take for room to heat?
  - How much energy will it cost?
  - What time does Alice get home on Tuesdays?

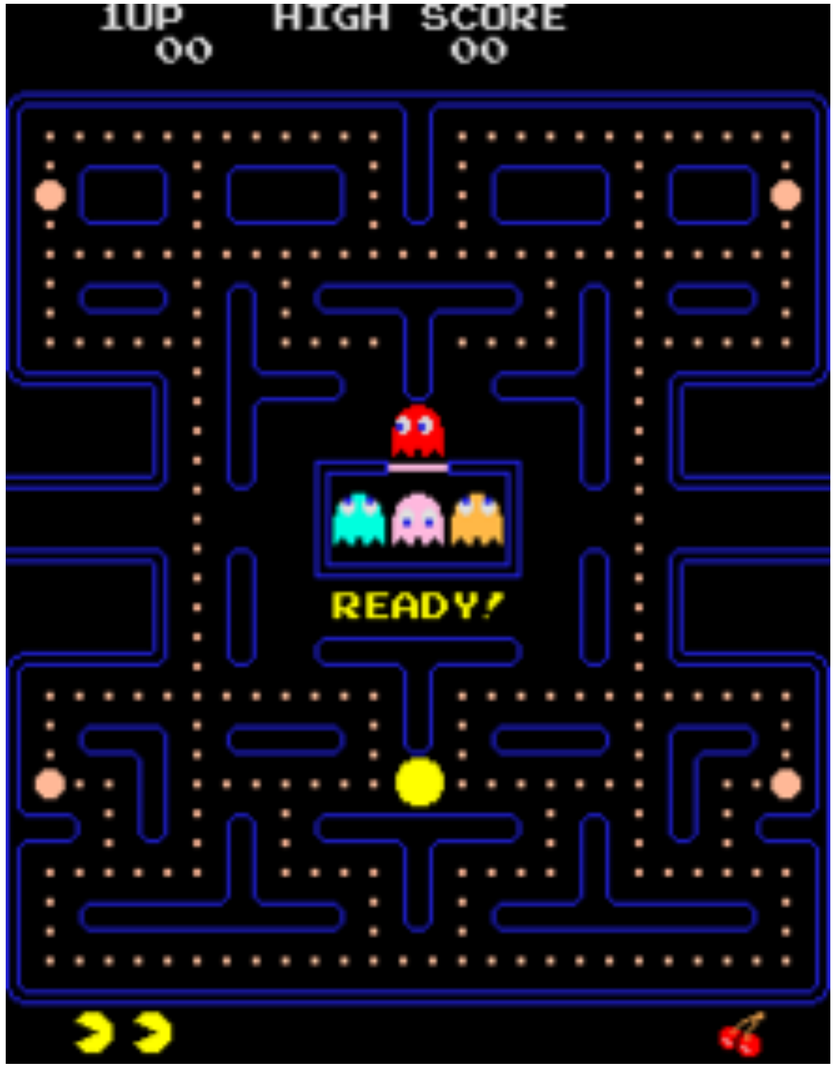


Agents with internal state

- It may also be useful for the agent to maintain an internal state
- Keeps track of information that is not explicit in the environment



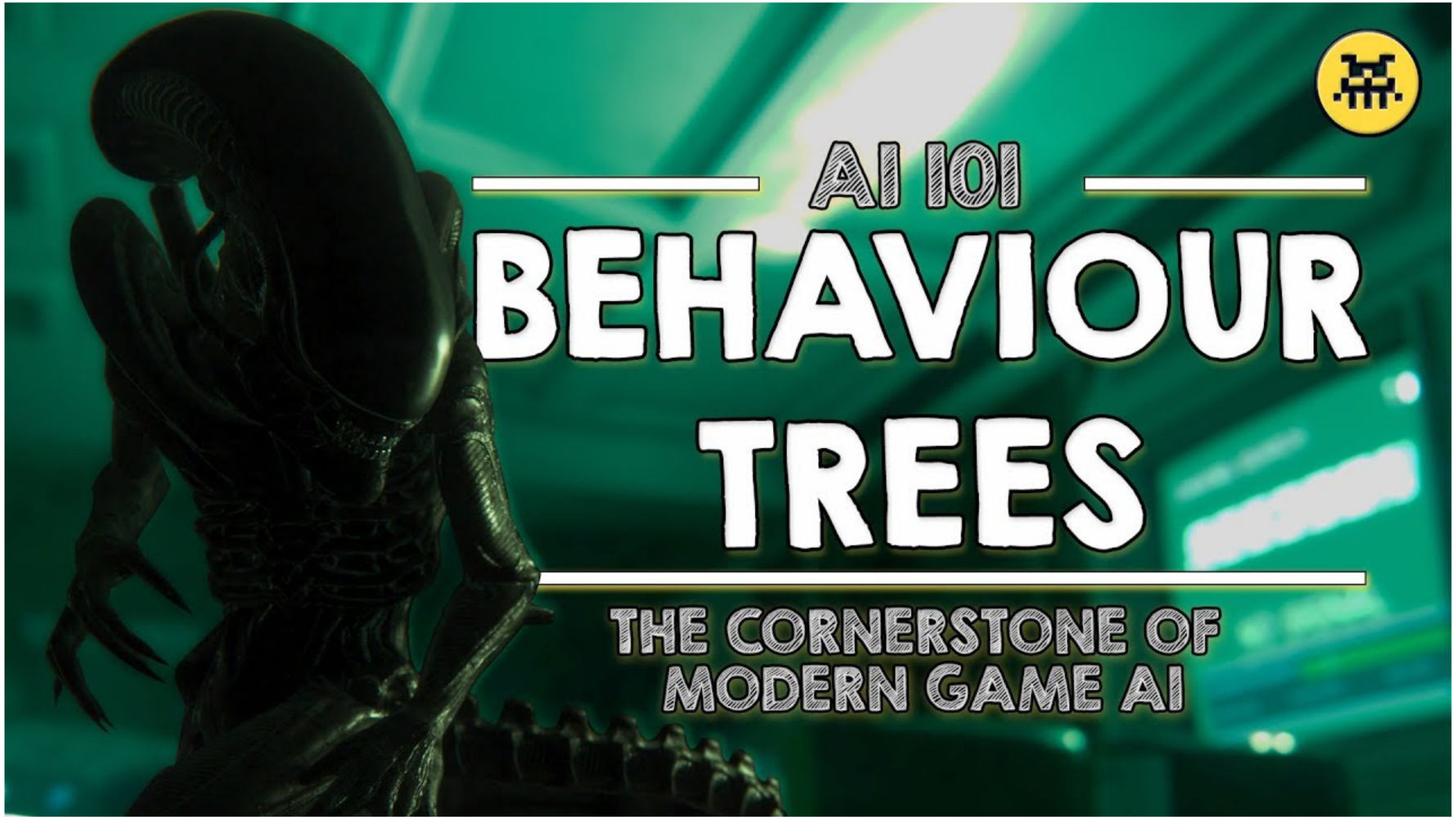
Example: Pacman ghosts



- Ghosts may be in a “fleeing” state or a “chasing” internal state
- In Pacman, it’s easy to derive the internal state from the environment
  - Not necessarily the case in complex environments

Suggested video: Finite State Machines

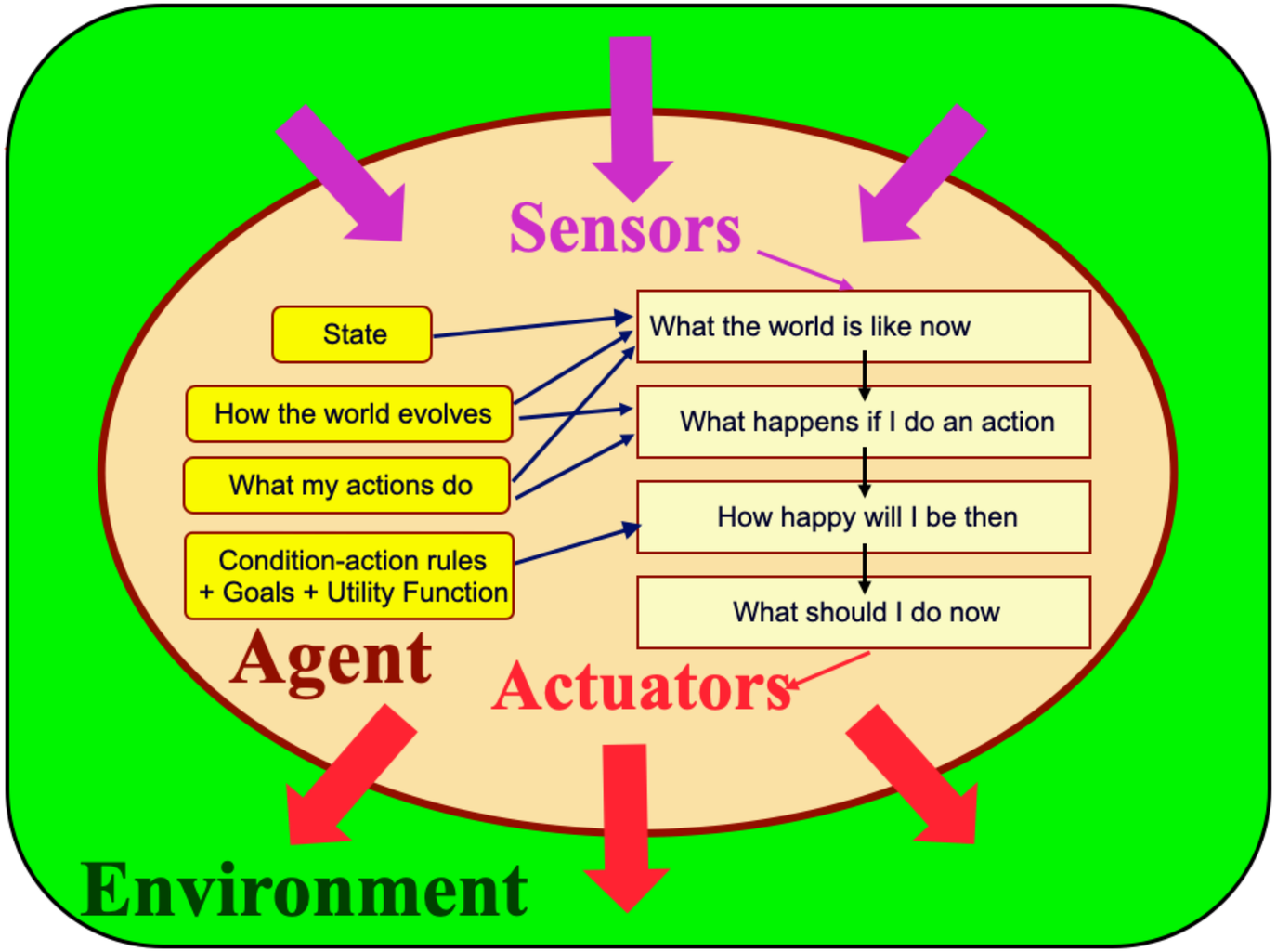
Suggested video: Behavior Trees



Utility-based agents

- In practice, we don’t want to have to write how the agent should act in every situation
- We might not even know the answer ourselves
- Solution
  - Encode what we want with a **utility function** (sometimes **cost**)
  - Tell the agent to maximize (or minimize) utility/cost over relevant timeframe
  - Let the agent decide how to act!

Utility-based agents



Performance X utility

- Most of the time, we build agents to perform a certain task
- The **performance measure** describes how satisfied we (the designers) are with a given run

$$P : H \rightarrow \mathbb{R}$$

- Important: the performance measure is **external** to the agent
- What's a good performance measure for the thermostat environment?

Performance X utility

- For most agents of reasonable complexity, it is important fr the agent to model how well it is performance

- A typical way to do this is to assign a **utility** to each run:

$$U : H \to \mathbb{R}$$

- The utility is **internal** to the agent

- Since the agent doesn’t have access to its true performance, it tries instead to maximize the internal utility.

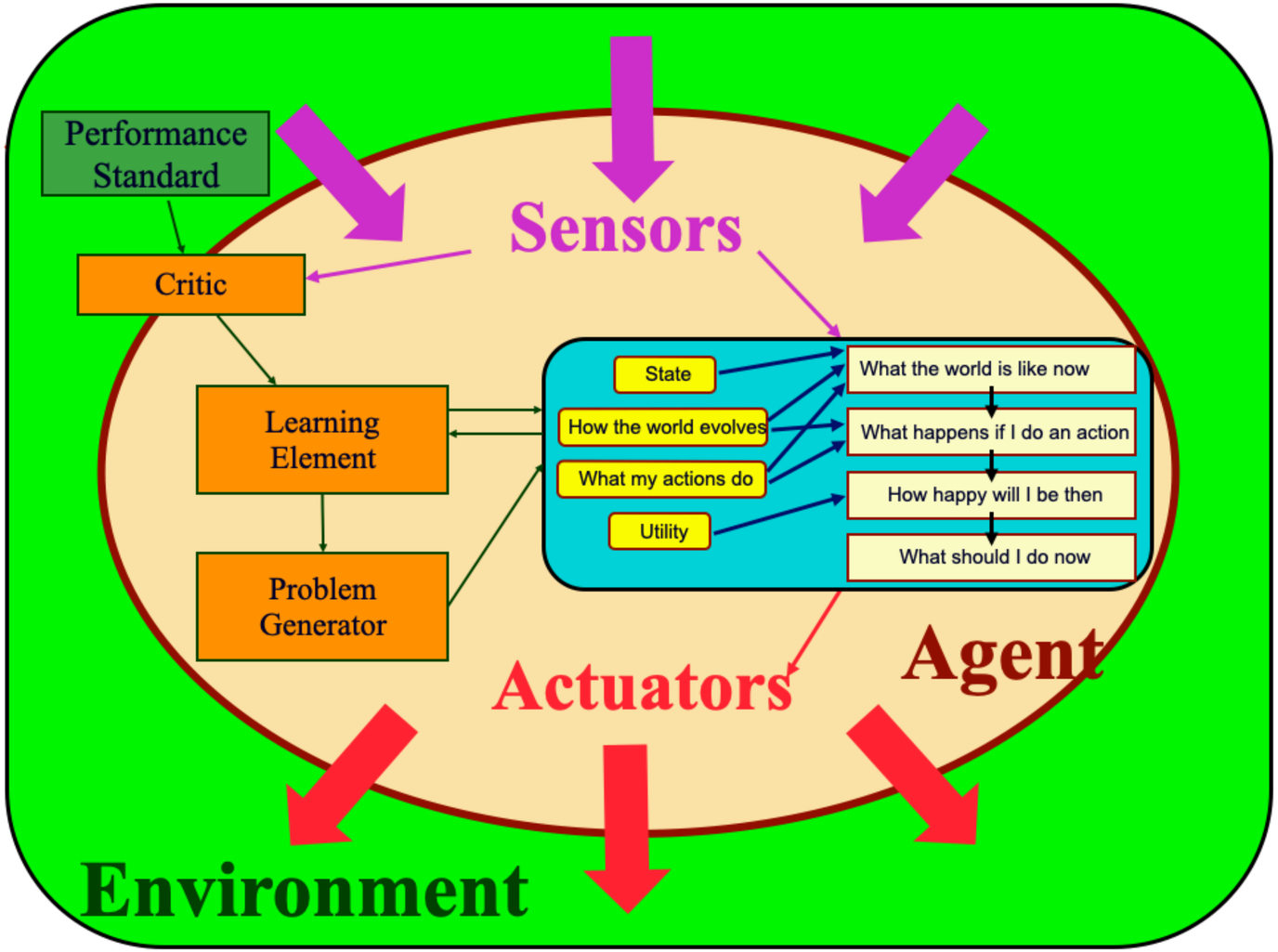
Example: DeepBlue

- Highly optimized minimax tree search algorithm
- Highly customized heuristic evaluation function w/ input by chess experts
- Search & evaluation distributed over customized hardware
- First system to beat a chess world champion (match vs Kasparov, 1997)

Learning-Based Agents

- Sometimes we lack information to make a good utility agent
  - Utility function may be expensive (e.g. require human experts) to create
  - We don’t know a good utility function (e.g. Go)
  - We don’t know the dynamics of the environment
- We may also want the agent to improve over time
- Solution: let the agent learn
  - A good utility function and/or
  - An approximation to the environment’s transition function and/or
  - A policy (map from state or state history to action)

Learning Agents



- Selects moves based on a Monte-Carlo Tree Search (MCTS) procedure
- Two heuristics inform MCTS node expansion and value estimation
- Heuristics are initialized by learning from human demonstrations (Supervised Learning)
- Heuristics are improved via self-play (Reinforcement Learning)

Summary

- An agent-environment system can be characterized by the possible states of the environment (E), the possible actions of the agent (A) and a transition function (T)

- Performance can be captured **externally** by a **performance measure** and **internally** by a **utility function / evaluation function / heuristic**

- Agents may be designed to:
  - Act via simple reflex rules
  - Act to maximize its utility over a time frame
  - Maintain an internal state and/or an internal model of the environment
  - Learn the utility, environment and/or a policy over time