Lecture 3

CS 6380 Artificial Intelligence

Structure of Agents

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India

Some material based on lecture slides for CSE 571: Artificial Intelligence by Siddharth Srivastava, ASU.

Logistics and Recap

Resources

- Course Website: https://hails-group.github.io/courses/cs6380/jn26.html
  - Slides posted here
  - Schedule already available

- Announcements: Moodle

- Course Email: pulkitv@cse.iitm.ac.in
  - Subject: [CS6380] <Subject>
  - No URGENT email responses. Please **plan** ahead.
  - We will respond to each email within 48 hours.
  - Any course-related query should be posted on Moodle so that others can also benefit from the answer.

Exams

- 3 Exams:
  1. [Quiz 1] Aug 31
  2. [Quiz 2] Oct 12
  3. [End Sem] Nov 11

- More details closer to exams.

Grading Breakup

- Exams: 70%
  1. Quiz 1 : 20%
  2. Quiz 2 : 20%
  3. End Sem: 30%
- Assignment-Quizzes: 25%
- In-class Quizzes: 5%
- Relative Grading

PEAS: Specifying a Task Environment

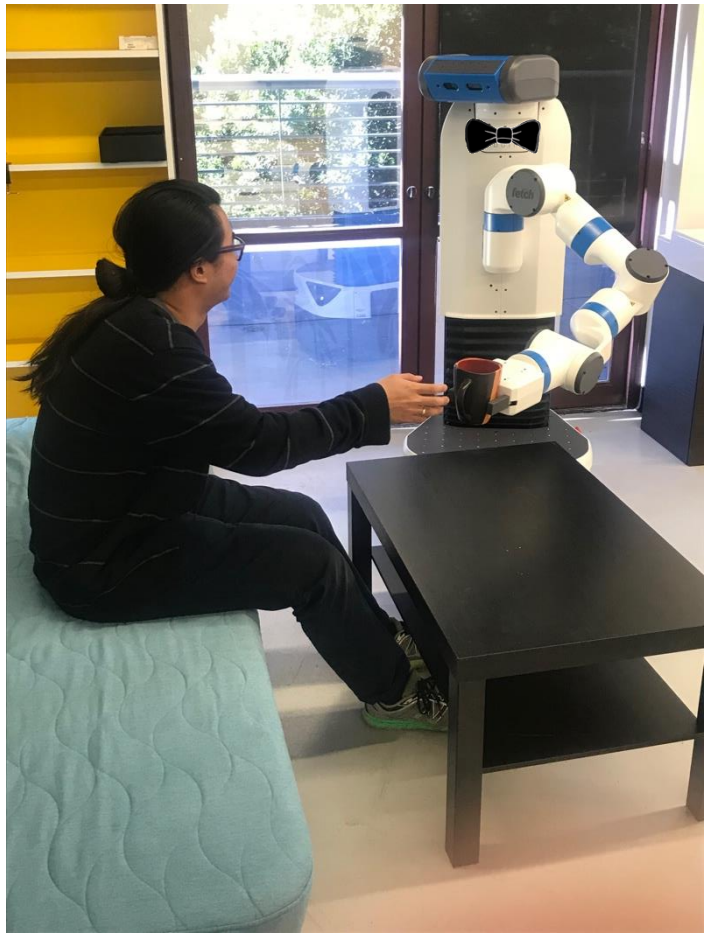
- Before designing any agent, specify the **task environment** as fully as possible

- **P: Performance measure**
  - What counts as success, stated in terms of environment states
- **E: Environment**
  - Everything relevant to the agent's decision-making
- **A: Actuators**
  - What the agent can do to the environment
- **S: Sensors**
  - What the agent can perceive of the environment

- PEAS is the design contract for the whole problem
  - Get it wrong and no amount of algorithmic cleverness will rescue the agent
  - Every algorithm in the rest of this course assumes a PEAS specification has already been fixed

Household Robot

- Actuators
  - Joint motors, wifi antennas
- Sensors
  - RGBD cameras, LiDAR, joint values, mic
  - Object detection algorithms, localization algorithms, user-command interface
- Environment
  - Location of humans, locations of all objects in the household, lighting conditions
- Performance measure
  - Number of questions asked, time taken to serve, cost of resources used



Types of Environments

Types of Environments

	Solitaire	Chess	Internet Shopping	Household Robot
Observable	No	Yes	Partial	Partial
Deterministic	Yes	Yes*	No	No
Static	Yes	No*	No	No
Discrete	Yes	Yes	Yes	No
Single Agent	Yes	No*	No [Communication?]	Yes

- Observable: Agent knows the state of environment at all times
- Deterministic: Effects of actions are expressible as functions: $S \times A \rightarrow S$
- Static: Environment does not change if agent does not act
- Discrete: Environment modeled using discrete variables, discrete timesteps
- Single agent: Problem includes a single decision-maker

Episodic vs. Sequential Environments

- **Episodic**: experience divides into atomic, independent episodes
  - Each episode is one percept and one action
  - The action taken in one episode does not affect any later episode
  - Spotting defective parts on an assembly line; classifying an image

- **Sequential**: the current decision affects all future decisions
  - Chess, driving, robot navigation, a conversation
  - Short-term actions have long-term consequences, so the agent must think ahead

- Why this axis matters more than it looks
  - Episodic problems need no lookahead at all; they are enormously simpler
  - This is the difference between classification and planning

- **Common confusion:** episodic is about **dependence between episodes**, not about whether the task repeats. A part-picking robot repeats its task forever and is still episodic.

Deterministic, Stochastic, and Nondeterministic

- **Deterministic:** the next state is completely determined by the current state and the action executed

- **Stochastic:** outcomes are uncertain, and the uncertainty is **quantified by probabilities**
  - The wheels slip 10% of the time; the sensor is wrong 2% of the time
  - The agent can reason about likelihoods and maximise expected utility

- **Nondeterministic:** outcomes are listed as possibilities, with **no probabilities attached**
  - A plan must succeed for **every** possible outcome, not merely the likely ones
  - This is the right model when probabilities are unknown or when you need a worst-case guarantee

- An environment is **uncertain** if it is not fully observable **or** not deterministic
  - This distinction decides the **form of the solution**: a fixed sequence, a contingent plan, or a policy, which is exactly where this lecture is headed

Known vs. Unknown Is Not the Same as Observable

- **Known** / **unknown** refers to the agent's knowledge of the **rules**: the transition model
- **Observable** / **partially observable** refers to what the **sensors** report about the current state
  - These are different questions, and all four combinations occur

- **Known + fully observable**
  - Solitaire with the rules in front of you
- **Known + partially observable**
  - A card game whose rules you know, but whose opponent's hand you cannot see
- **Unknown + fully observable**
  - A new video game: the screen shows you everything, but you do not yet know what the buttons do
- **Unknown + partially observable**
  - A robot switched on in an unmapped building

- Unknown environments force the agent to **explore** in order to learn the model.

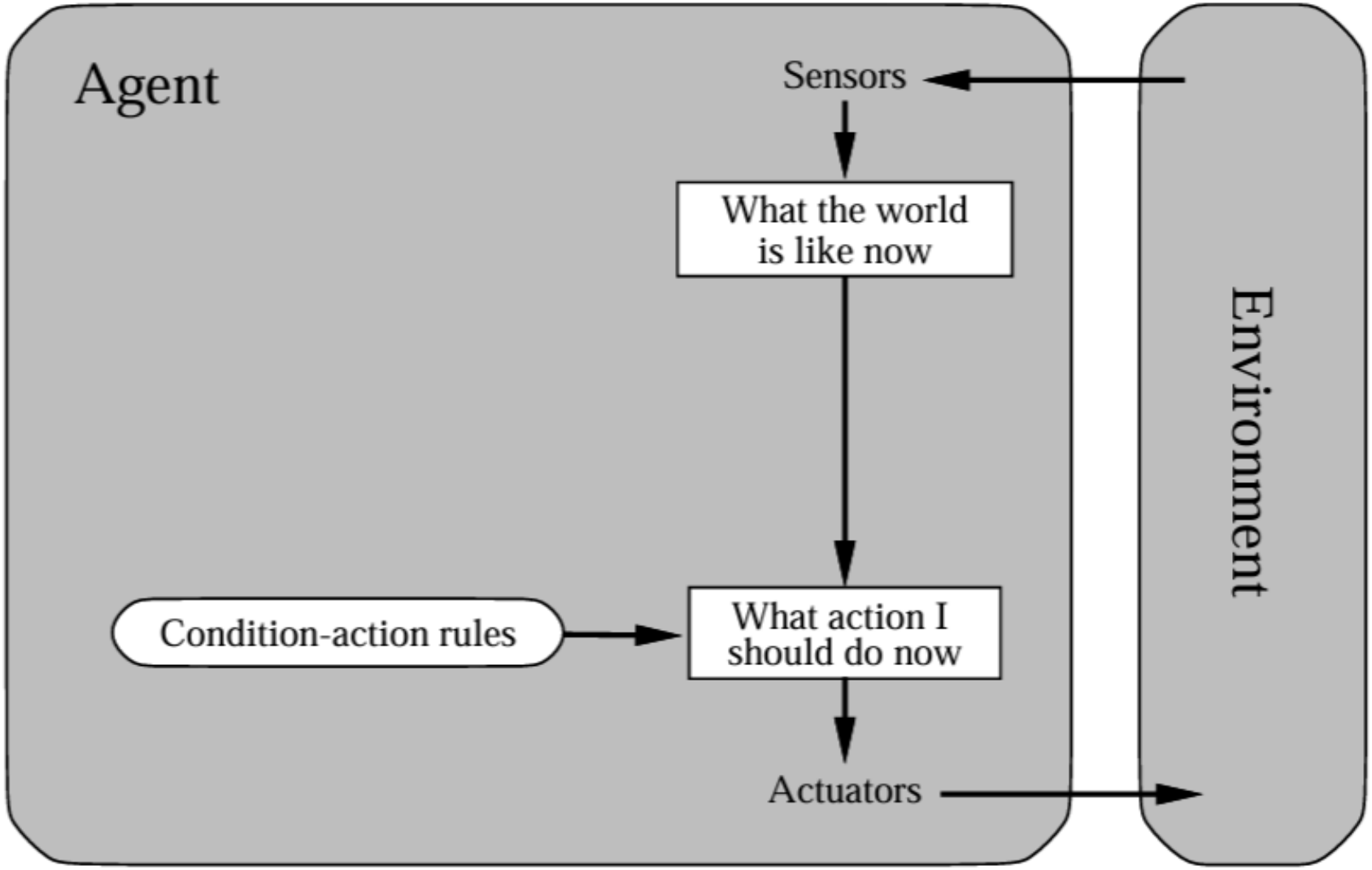
Types of Environments: All Seven Axes

	Solitaire	Chess (clock)	Internet Shopping	Household Robot	Part-Picking Robot
Fully observable	No	Yes	Partial	Partial	Partial
Deterministic	Yes	Yes	No	No	No
Episodic	No	No	No	No	Yes
Static	Yes	Semi	No	No	No
Discrete	Yes	Yes	Yes	No	No
Single agent	Yes	No	No	Yes	Yes
Known	Yes	Yes	Partly	No	Partly

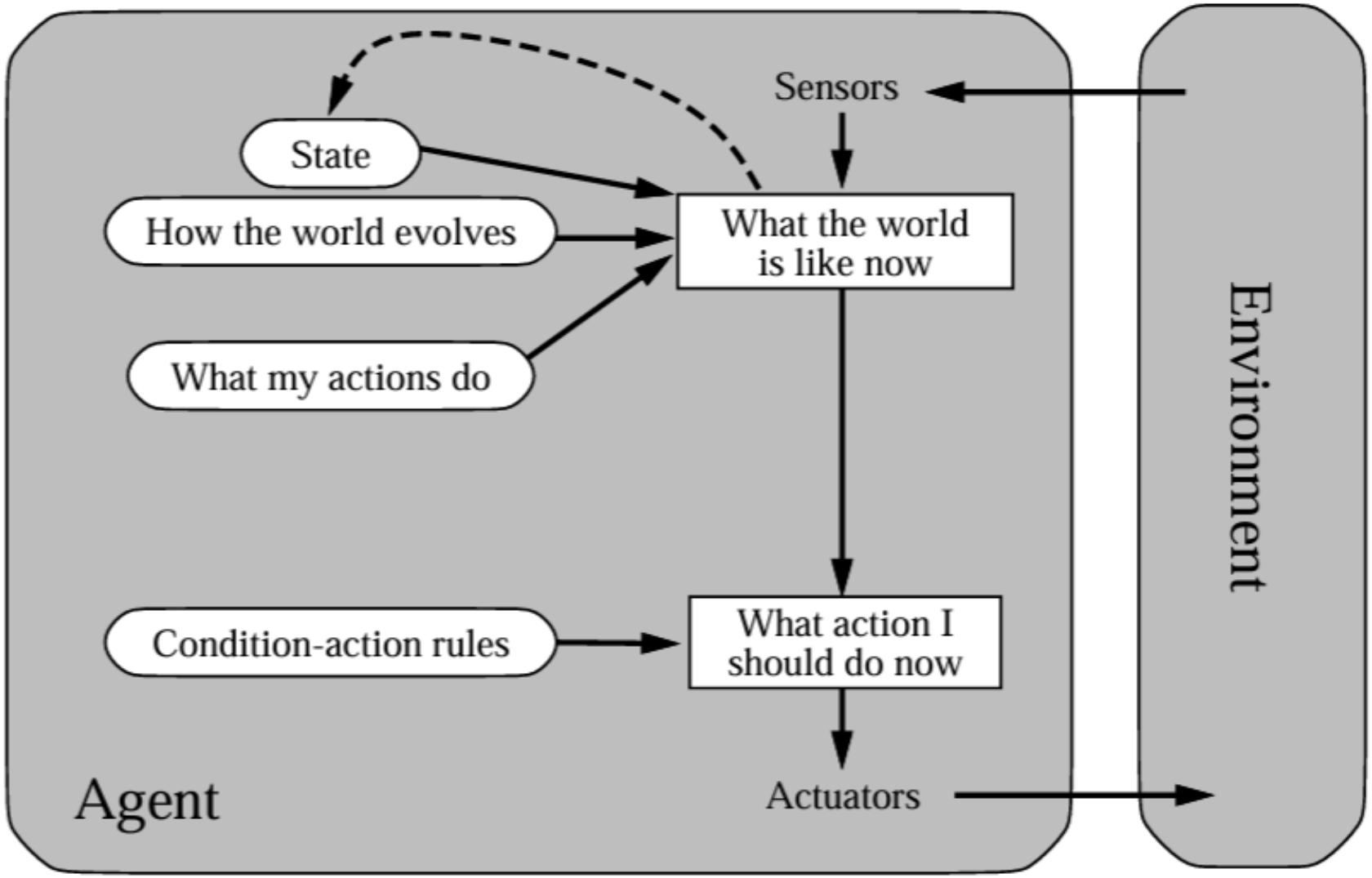
The two rows your earlier table did not have are Episodic and Known, and they are the two that most often decide which algorithm applies.

Structure of Agents

Reflex Agents

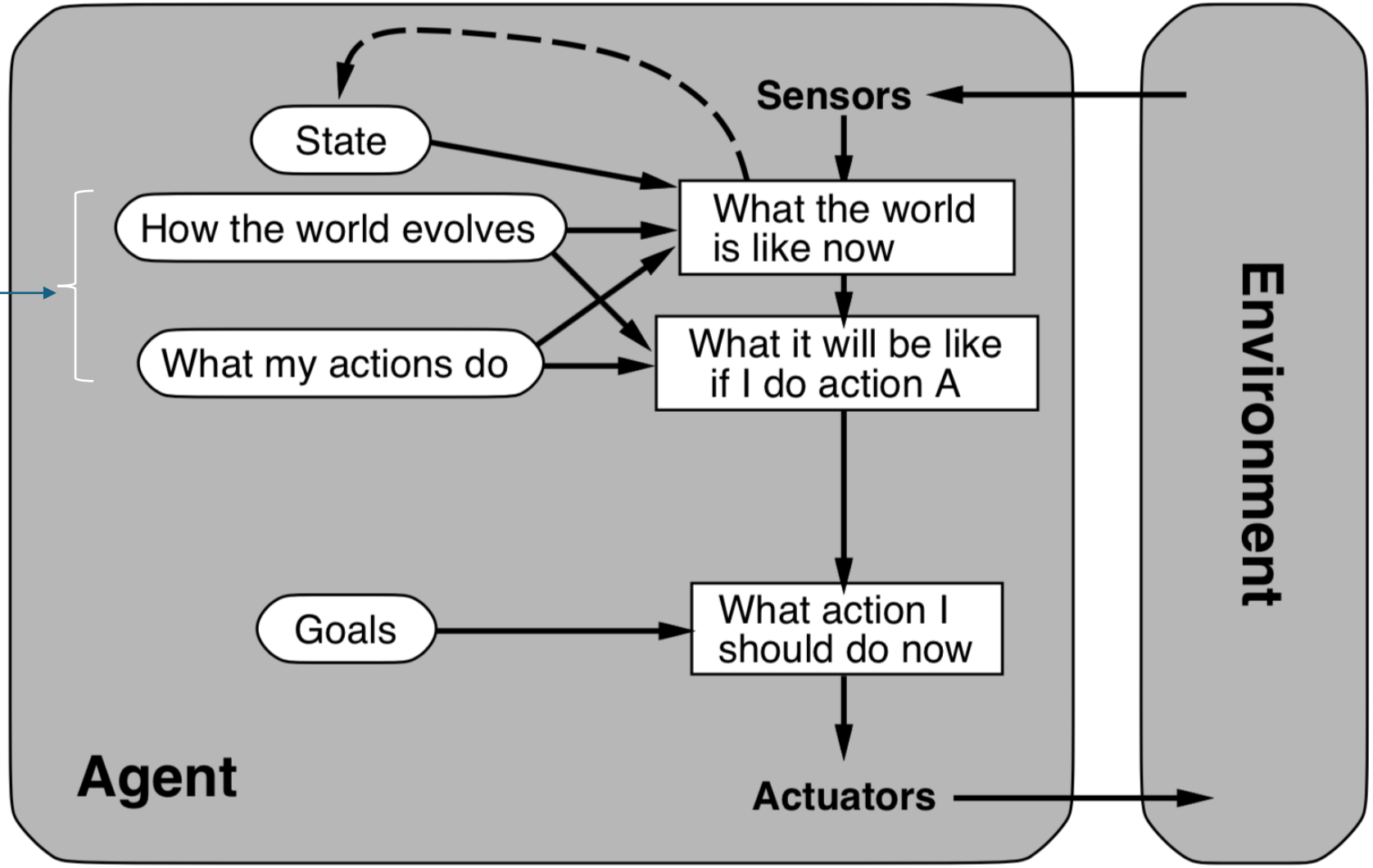


Model-Based Reflex Agents



Goal-based Agents

Models:
Simulators or
Analytical specs



Why Utility, When We Already Have Goals?

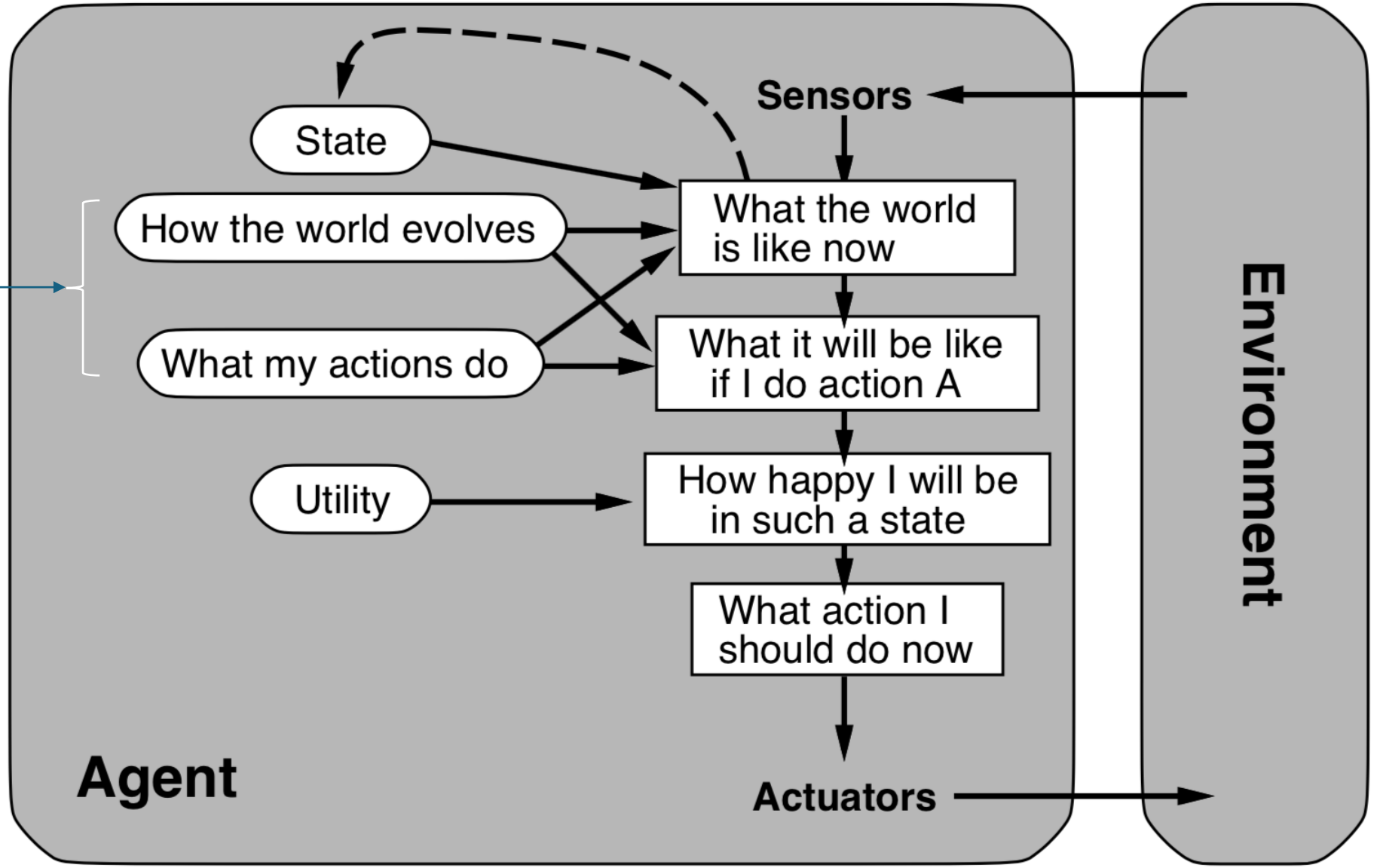
- A **goal** is binary: a state either satisfies it or it does not
  - That is enough to say whether the agent succeeded
  - It is not enough to say how **well** it succeeded

- Goals are insufficient in three common situations:
  - **Many routes to the goal, differing in quality.** Faster, safer, cheaper: goals cannot distinguish them
  - **Conflicting goals.** Speed and safety cannot both be maximised; something must specify the trade-off
  - **Uncertain achievement.** When no action guarantees the goal, likelihood of success must be weighed against the importance of the goal

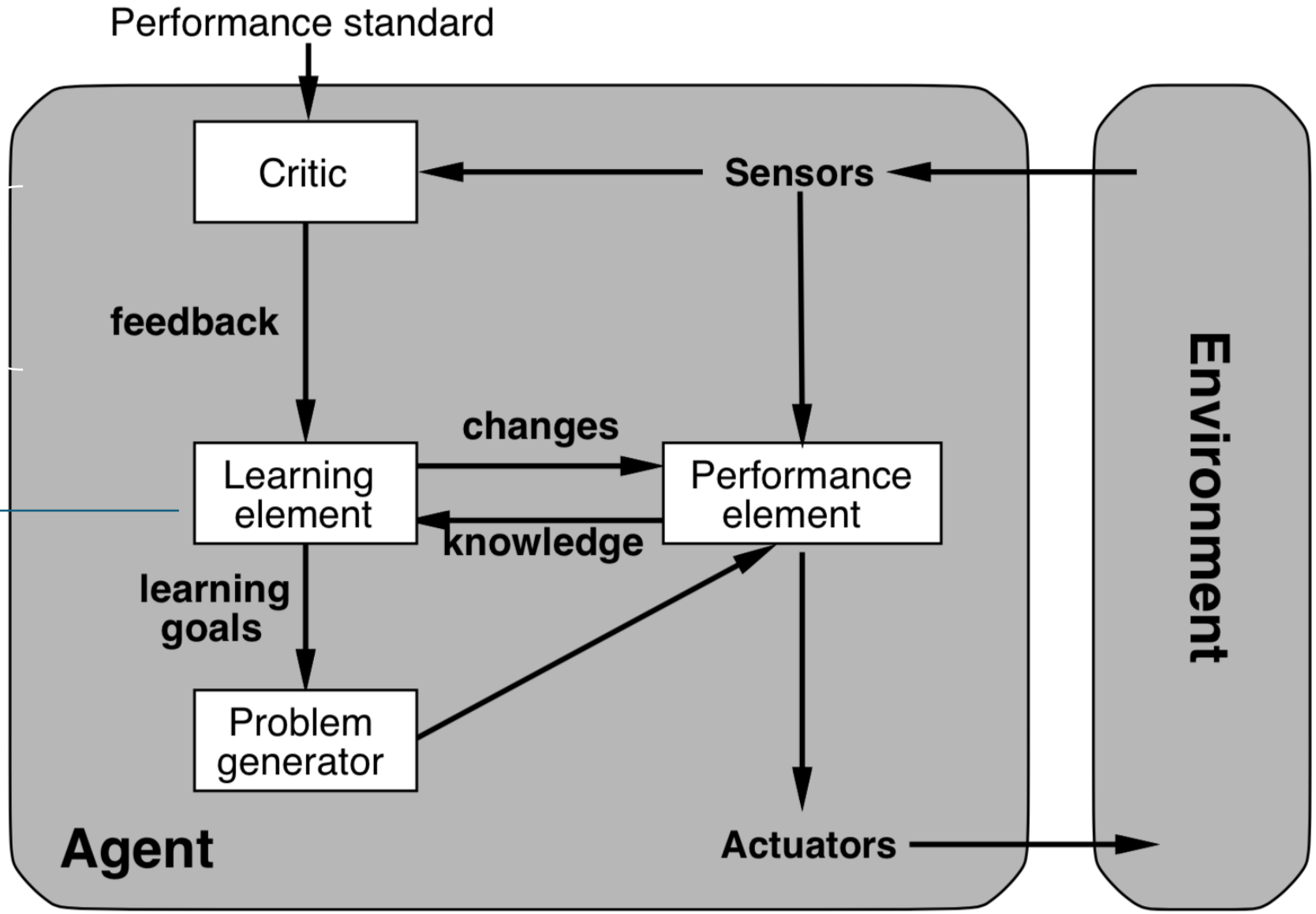
- A **utility function** maps a state (or a sequence of states) to a real number
  - It is the agent's internalisation of the performance measure
  - A rational utility-based agent maximises **expected utility**: the utility of each outcome, weighted by its probability
  - This is what lets an agent act sensibly when it cannot guarantee anything

Utility-based Agents

Models:
Simulators or
Analytical specs



Learning Agents



May learn or modify models

Inside a Learning Agent: Four Components

- **Performance element**
  - Everything we have called "the agent" so far; it takes in percepts and decides on actions
- **Learning element**
  - Makes improvements to the performance element, using feedback from the critic
  - Its design depends entirely on how the performance element is represented
- **Critic**
  - Tells the learning element how well the agent is doing, against a **fixed external performance standard**
  - Necessary because percepts alone do not indicate success; nothing in a chess percept says "that was a blunder"
- **Problem generator**
  - Suggests exploratory actions that are suboptimal now but lead to informative experience
  - Without it, the agent keeps doing what already works and never improves

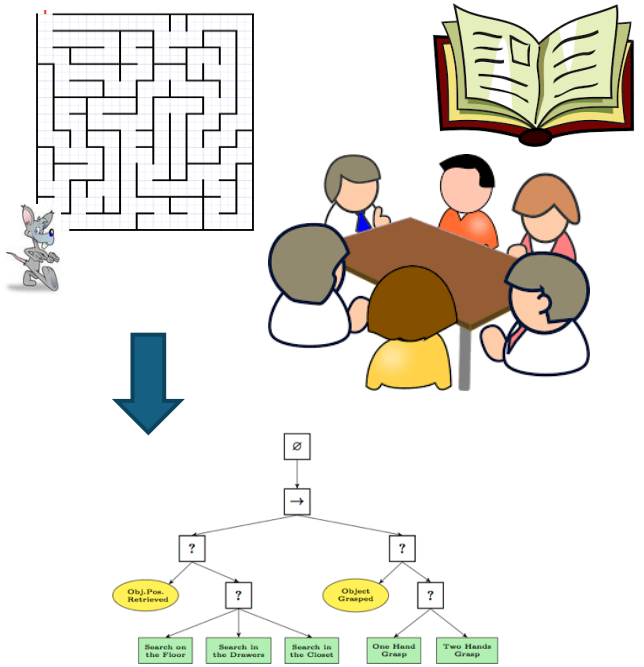
- **Why the standard must be external and fixed:** an agent that could adjust its own standard would "improve" by lowering the bar.

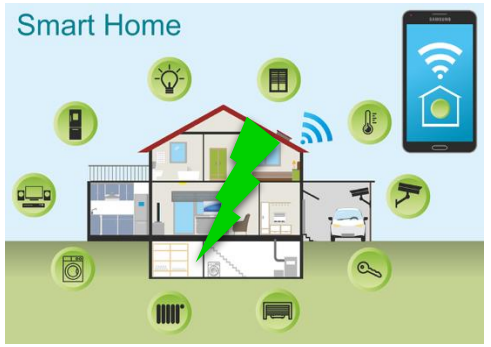
The Typical Agent Design Process

1. **Represent** background information and objective
   - Multi-step agents require states, actions and their effects on states
   - Learned using simulator, modeled using literature/experts, or partially modeled with scope for online learning

2. **Compute** "behavior" that would allow agent to achieve objective
   - Behavior may include contingencies depending on observed system state during execution

3. **Execute** this behavior
   - The bottomline, drives all of the above





Model-based Agents and Model-free Agents

- Model-free agents evaluate courses of action directly through interaction in the real world
  - Rare, due to safety and quality concerns
- Model-based agents evaluate courses of action "in their heads"
  - May employ different types of models: simulators (arbitrary generative models), or analytical models that provide precise sets of possible results
  - E.g., Atari-game-playing agents, AlphaGo etc.

- Ideal situation: agents initialized with available models that improve through learning on-the-fly

How Agents Represent the World

- Every component of an agent must represent the environment somehow. Three levels, in increasing order of expressiveness:

- **Atomic**: a state is a black box with no internal structure
  - The only operation available is: are these two states identical?
  - Used by search, game-playing, hidden Markov models, Markov decision processes
- **Factored**: a state is a fixed set of variables, each with a value
  - Two different states can now share some values, and you can say what changed
  - Used by constraint satisfaction, propositional logic, planning, Bayesian networks, most machine learning
- **Structured**: a state contains objects, each with attributes and relationships to other objects
  - Used by relational databases, first-order logic, first-order probability models, natural language understanding

- **The trade-off:** more expressive representations describe complex worlds far more concisely, but reasoning with them is correspondingly harder.

Representation Drives the Rest of the Course

- **Atomic representations**
  - Uninformed and informed search; search in complex environments; adversarial search and games
  - Also: Markov decision processes and reinforcement learning

- **Factored representations**
  - Constraint satisfaction problems; propositional logic and satisfiability
  - Classical planning; probabilistic reasoning and Bayesian networks; most of machine learning

- **Structured representations**
  - First-order logic and inference; knowledge representation
  - Relational probability models; natural language understanding

- **How to use this when you design an agent**
  - Choose the **coarsest** representation that still captures what your performance measure depends on
  - Every step up in expressiveness costs you computationally; pay for it only when the problem demands it