INTELLIGENT AGENTS

IT426: Artificial Intelligence Systems

Information Technology Department

*Some slides are by Kris Hauser, IU

OUTLINE

- Definition of agent
- Rational agent
- Properties of environments
- Types of agents

Definition of Agent

- Anything that:
  - **Perceives** its environment through sensors
  - **Acts** upon its environment through actuators
- A.k.a. controller, robot

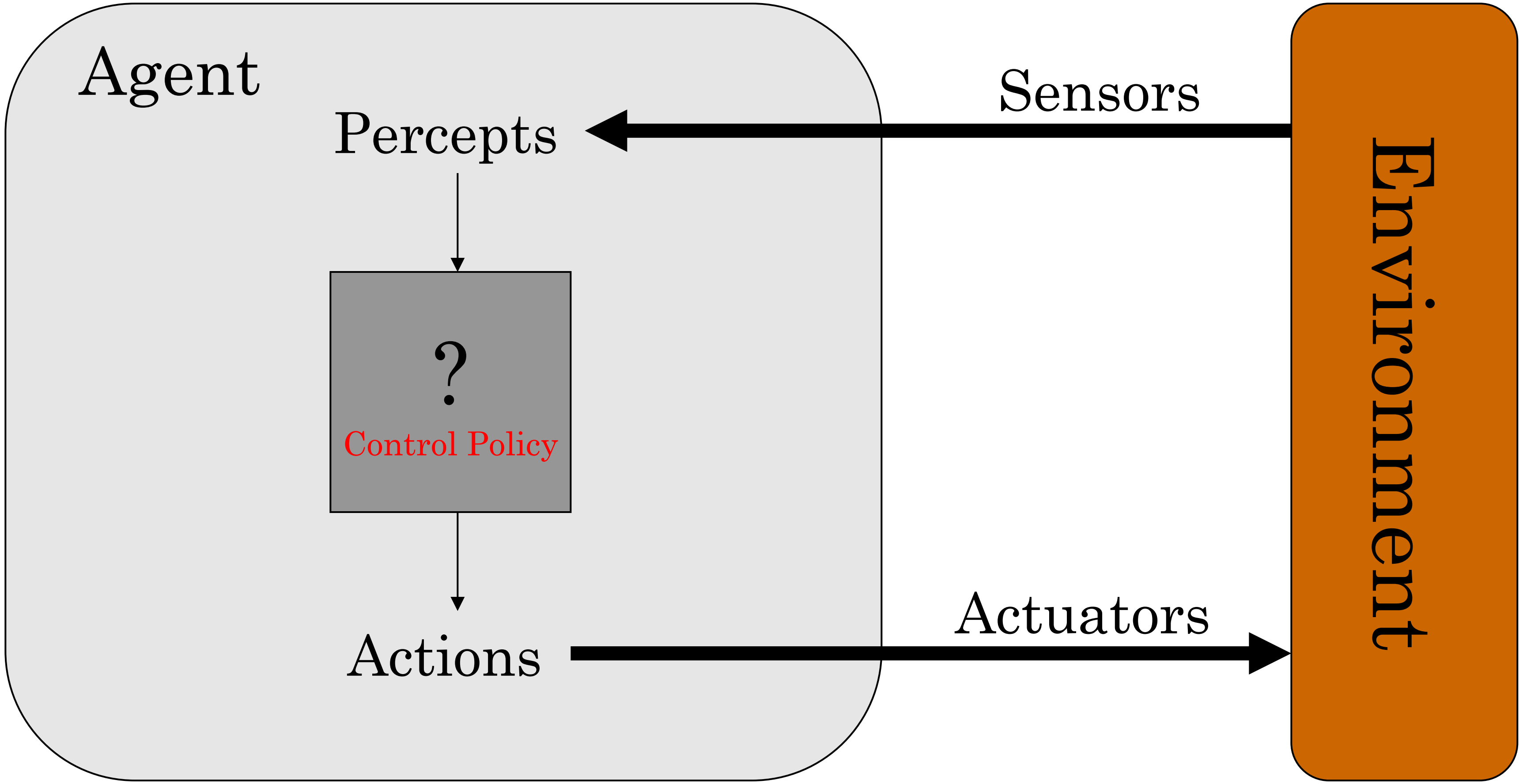
EXAMPLES OF AGENTS

- A human agent:
  - Eyes, ears, and other organs for sensors
  - Hands, legs, vocal tract, and so on for actuators
- A robotic agent:
  - Cameras and infrared range finders for sensors
  - Various motors for actuators
- A software agent:
  - Receives keystrokes, file contents, and network packets as sensory inputs
  - Acts on the environment by displaying on the screen, writing files, and sending network packets

Definition of Environment

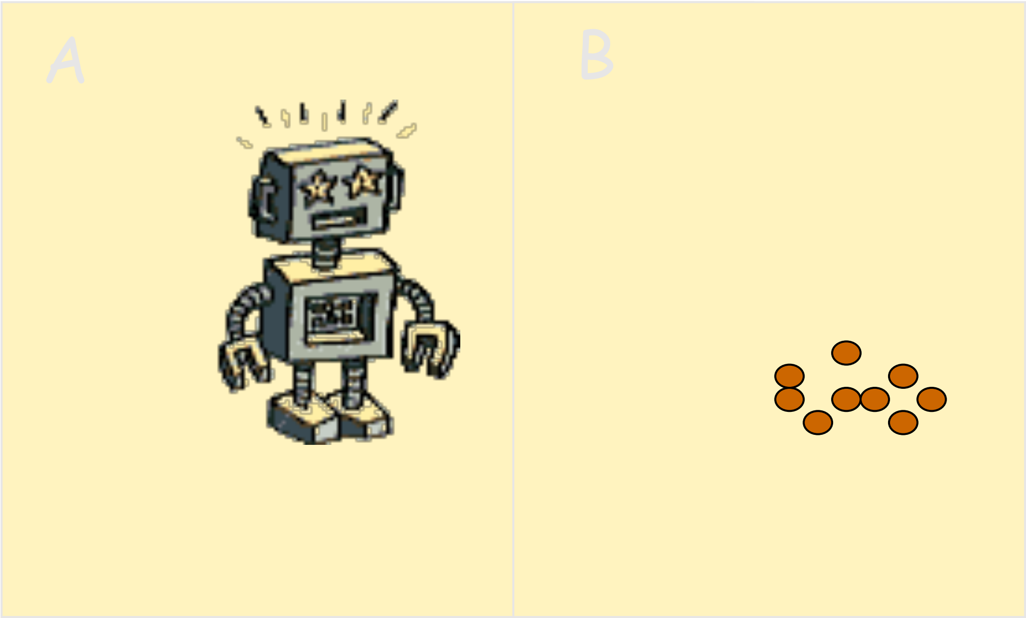
- The real world, or a virtual world
- Rules of math/formal logic
- Rules of a game
- …
- Specific to the **problem domain**

DEFINITION OF AGENT

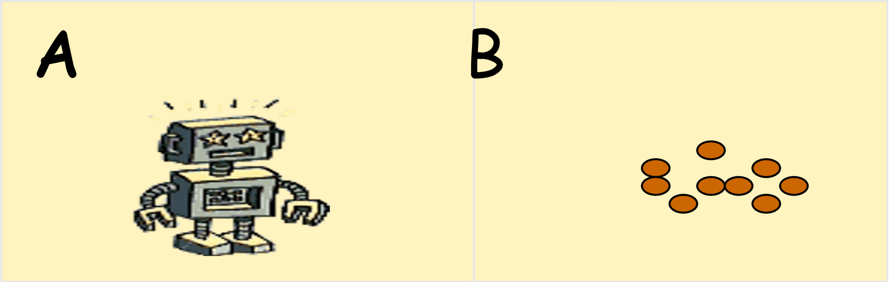


Sense – Plan - Act

The Vacuum-Cleaner World



Agent Function



Percept sequence [Location, status]	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
$\vdots$ history of previous observations	$\vdots$

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Action causes change on status

Good Behavior

- Performance measure (aka reward, merit, cost, loss, error)
- **Part of the problem domain**

Good Behavior

Thinking Humanly	Thinking Rationally
Cognitive Science	**Laws of Thought** logic
Acting Humanly	**Acting Rationally**
Turing Test	**Rational Agents**
human	do the "**right thing**" given what it knows

Measure the success in terms of

Good Behavior

- A **rational agent** is one that does the right thing, i.e. rationale
- When an agent is 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 is desirable, then the agent has performed well.
- This notion of desirability is captured by a **performance measure** that evaluates any given sequence of environment states.

Rational Agent

- What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence to date.

- **Definition of a rational agent**:

  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.*

DESIGN AGENT

- In designing an agent, the first step must always be to specify the **task environment (PEAS)** as fully as possible

PEAS (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Exercise

- Formulate the problem domains for:
  - Tic-tac-toe
  - A taxi driver
  - A bee
  - A student in IT426
  - A doctor diagnosing a patient

What is/are the:

- Environment
- sensors
- Actuators
- Performance measure

Environment

- Environments are characterized by:
  - Observability
  - Determinism
  - Episodism
  - Stasis
  - Discreteness
  - Number of agents

OBSERVABILITY

- Fully vs Partial Observable
- If an agent is operating in a fully observable environment, it can observe the complete state (has a global view) of the environment at any point in time.
- Examples:
  - Tic-Tac-Toe: fully observable
  - Vacuum-Cleaner: Partially observable

DETERMINISM

randomness

- Deterministic vs. stochastic stochastic like taxi driver, tennis
- If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic.
- In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment

and static

EPISODISM

- Episodic vs. sequential
- Episodic environment divides the problem into independent episodes, where an episode consists of an agent sensing the environment followed by its action.
- The next episode does not depend on the actions taken in previous episodes.
- Examples:
  - Classification task: Spotting defective parts on assembly line is episodic
  - Chess-playing and taxi-driving case are sequential

Stasis

- Static vs. dynamic
- If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static
- Examples:
  - Taxi driving is dynamic as the other cars and the taxi itself keep moving
  - Crossword puzzles are static.

DISCRETENESS

- Discrete vs. continuous
- An environment is said to be discrete if there is:
  - a finite number of actions
  - a finite number of percepts
- Examples:
  - Tic-Tac-Toe is discrete
  - Throwing darts is continuous

NUMBER OF AGENTS

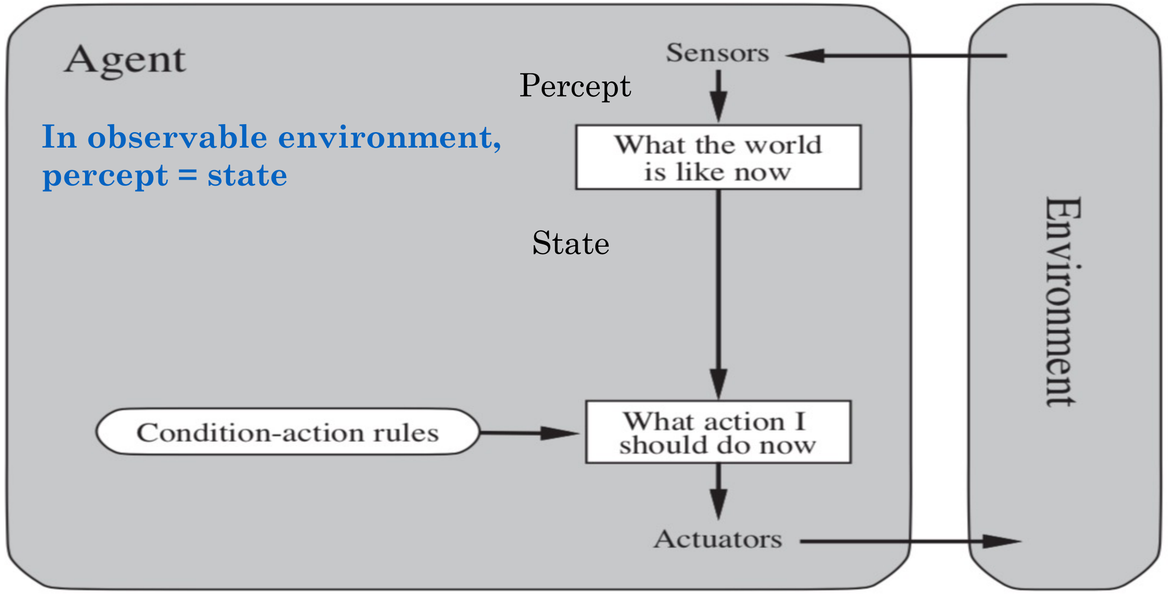
- Single- vs Multi-agent environments
- Examples:
  - Crossword is a single-agent environment
  - Tic-Tac-Toe is a Multi-agent environment
- Multi-agent environments can be competitive or cooperative.

Types of Agents

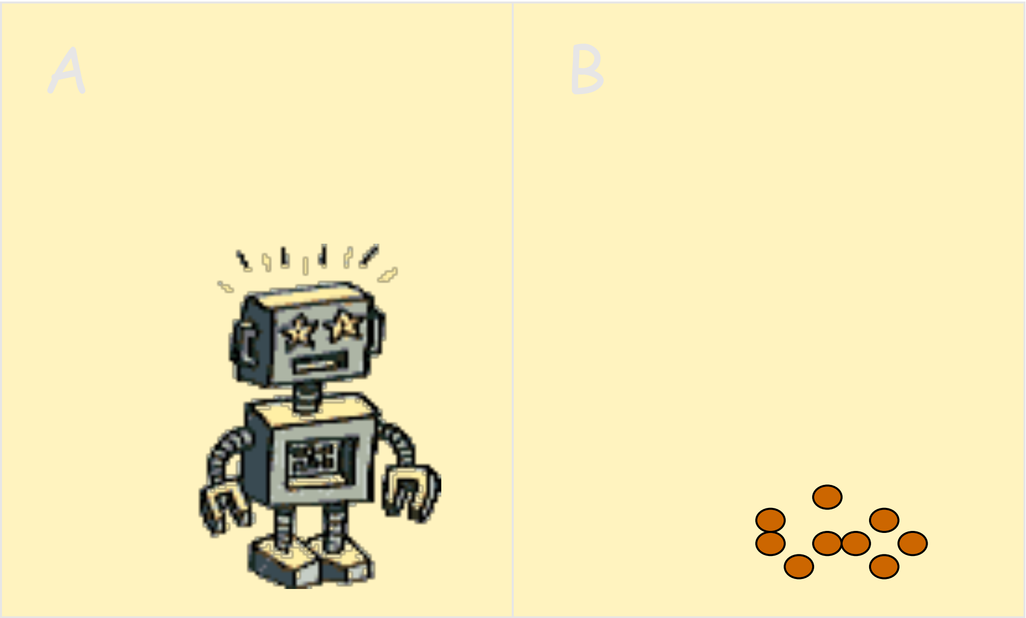
- Simple reflex (aka reactive, rule-based)
- Model-based
- Goal-based
- Utility-based (aka decision-theoretic, game-theoretic)
- Learning (aka adaptive)

SIMPLE REFLEX AGENTS

- Agent selects actions on the basis of the *current* percept, ignoring the rest of the percept history.
- Operates in fully observable environments.



Rule-based Reflex Agent



if DIRTY = TRUE then SUCK

else if LOCATION = A then RIGHT

else if LOCATION = B then LEFT

Building a Simple Reflex Agent

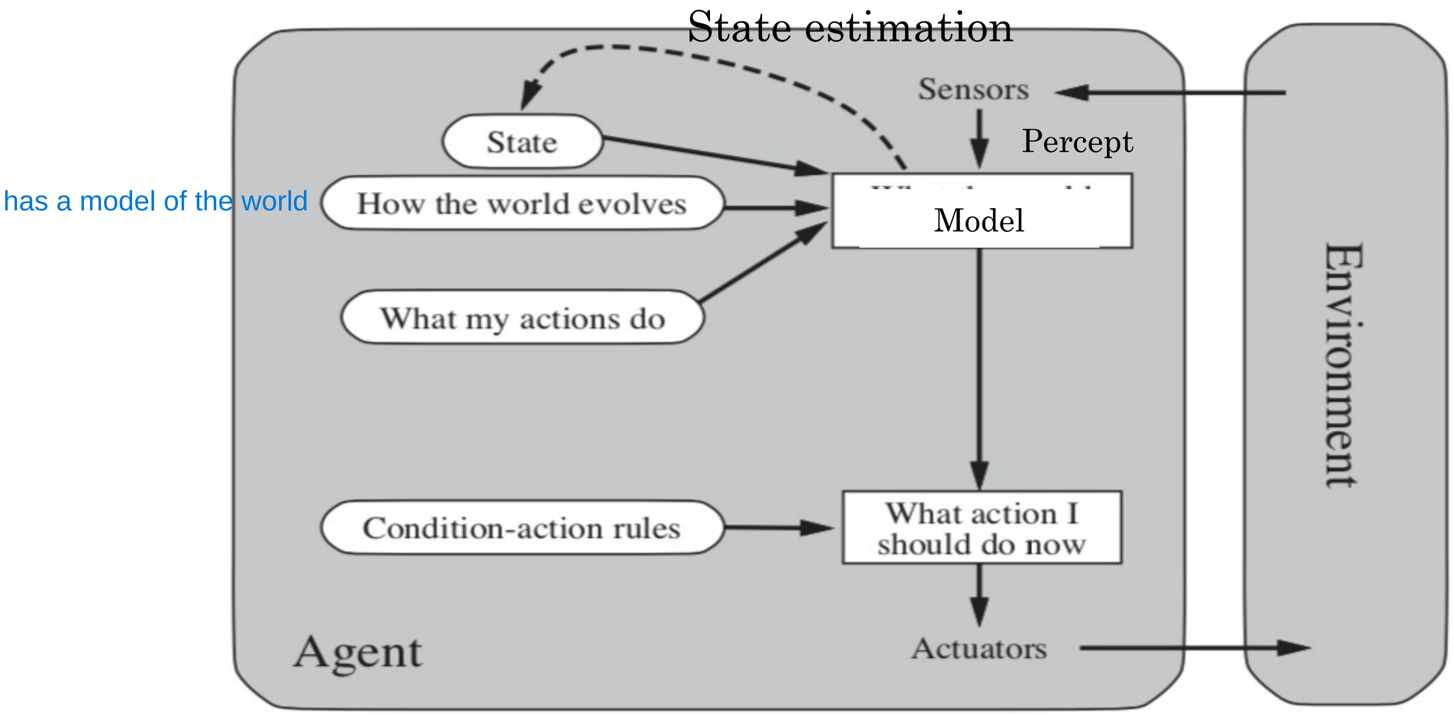
- Rules: a map from states to action
  - $a = \pi(s)$

    a: action, s: state, PI is a function that maps the state onto the performed action.
- Can be:
  - Designed by hand
  - Learned from a "teacher" (e.g., human expert) using ML techniques

Model-based reflex agents

- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see 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

Model-based reflex agents



Model-Based Agent

State:

LOCATION

HOW-DIRTY(A)

HOW-DIRTY(B)

HAS-SEEN(A)

HAS-SEEN(B)

Rules:

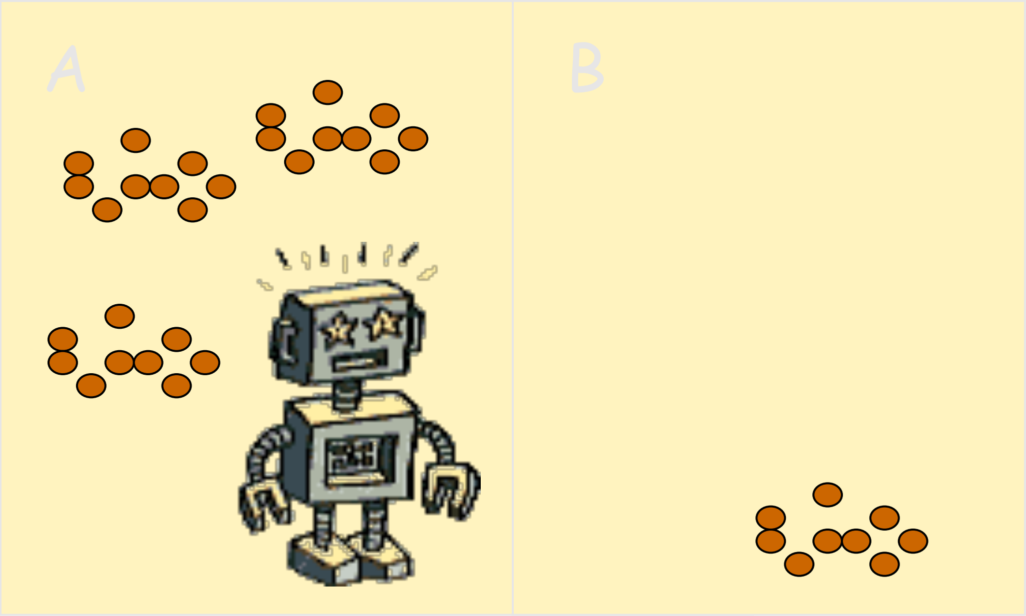
if LOCATION = A then

if HAS-SEEN(B) = FALSE then RIGHT

else if HOW-DIRTY(A) > HOW-DIRTY(B) then SUCK

else RIGHT

…



Model:

HOW-DIRTY(LOCATION) = X continuous number, percentage

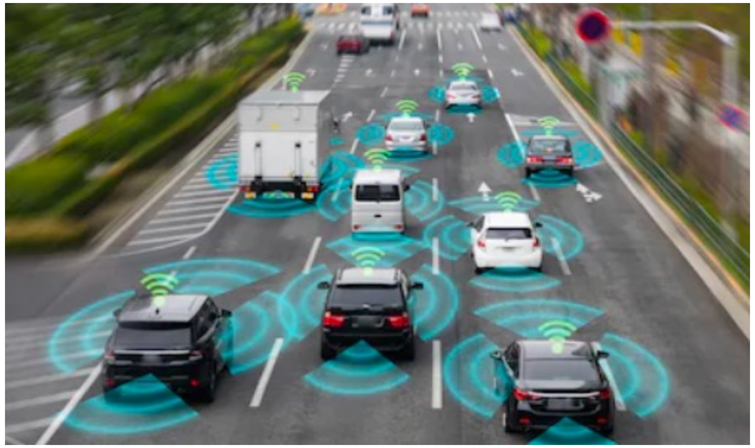
HAS-SEEN(LOCATION) = TRUE

Building a Model-based Reflex Agent

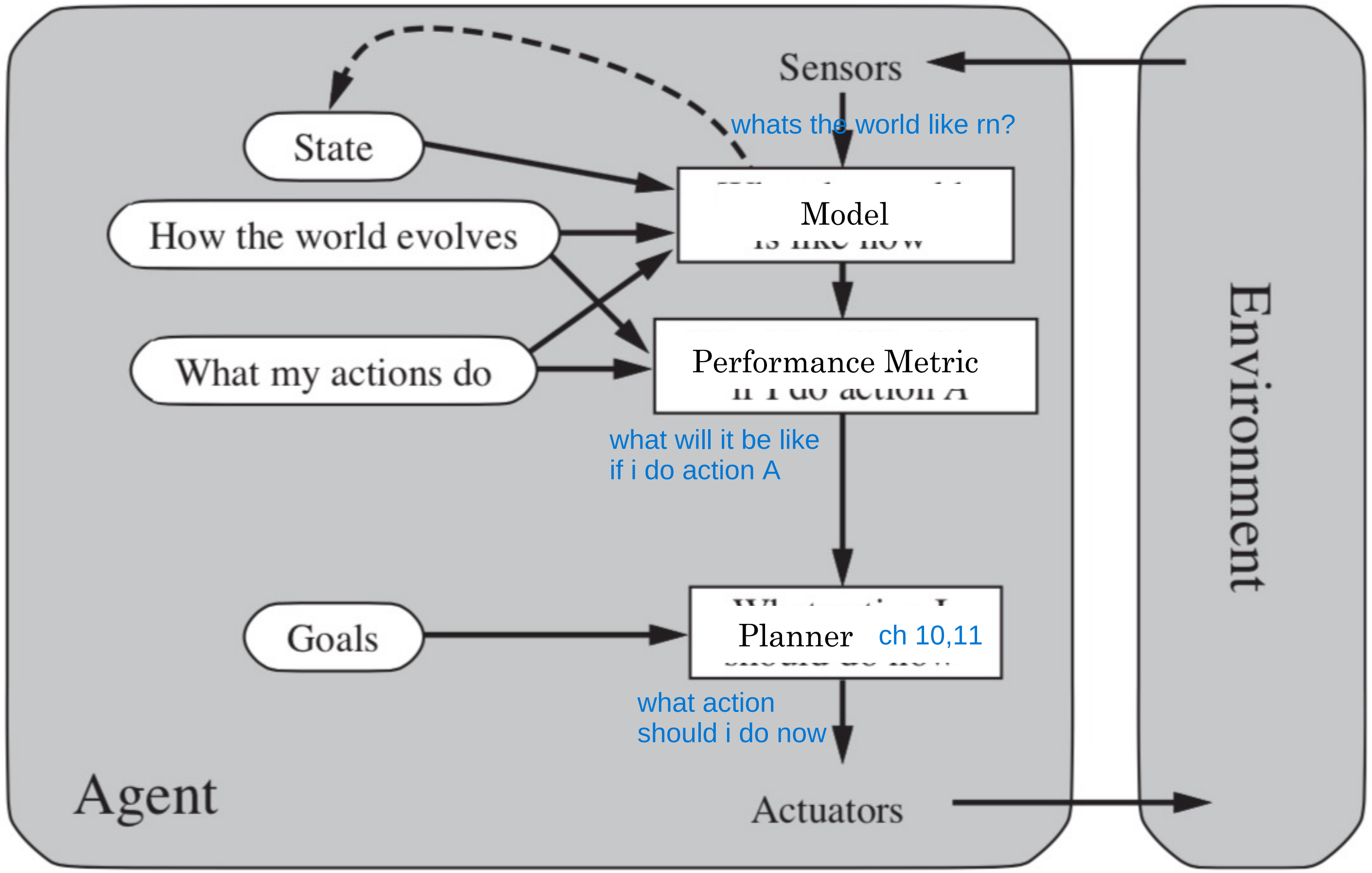
- A model is a map from prior state s, action a, to new state s'
  - s' = T(s,a)
- Can be
  - Constructed through domain knowledge (e.g., rules of a game, state machine of a computer program, a physics simulator for a robot)
  - Learned from watching the system behave (system identification, calibration)
- Rules can be designed or learned as before

Model-based, Goal-Based Agent

- Knowing something about the current state of the environment is not always enough to decide what to do.
- For example:
  - At a road junction, the taxi can turn left, turn right, or go straight on.
  - The correct decision depends on where the taxi is trying to get to.
  - So, as well as a current state description, the agent needs some sort of **goal** information that describes situations that are desirable.



Model-based, Goal-Based Agents



Building a Goal-Based Agent

- Requires:
  - Model of percepts (sensor model)
  - Action generation algorithm (planner)
  - Performance metric
- Planning using search
- Performance metric: does it reach the goal?
- 2 Types:
  - Problem Solving Agents (use atomic rep. of state)
  - Planning Agents (use factored/structured rep. of state)

Different ways for State representations

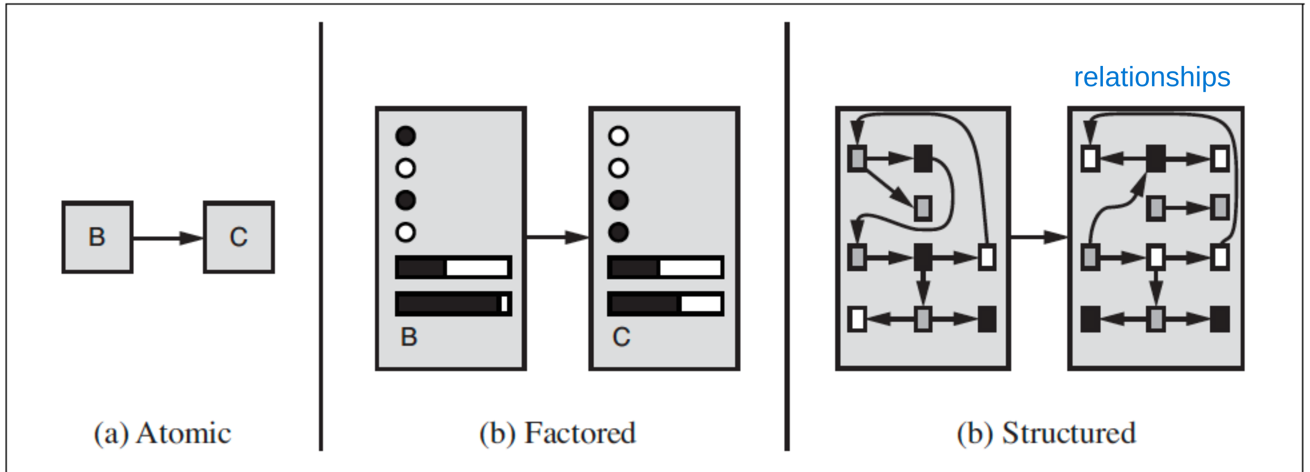
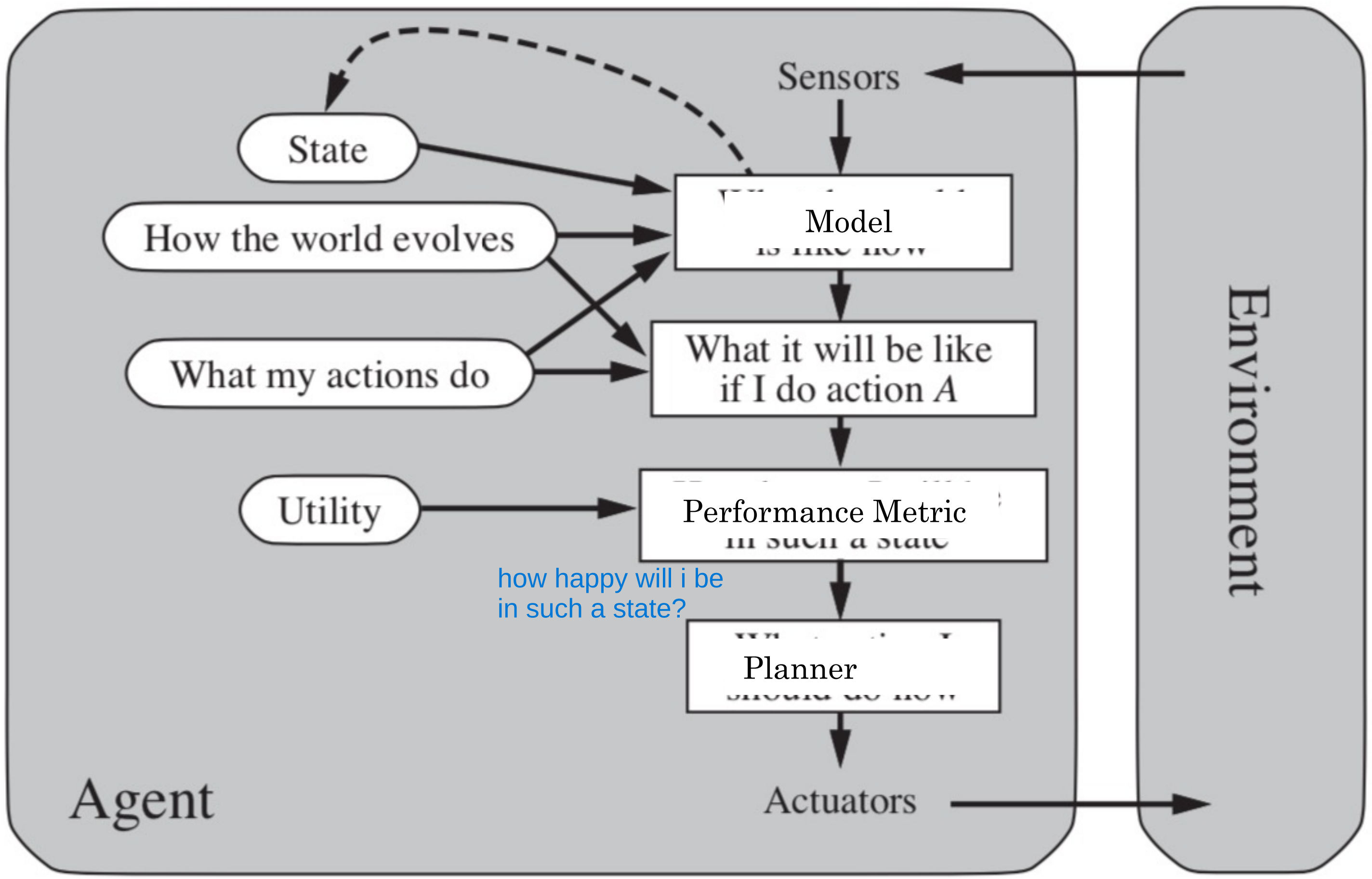


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Utility-Based Agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- For example:
  - Many action sequences will get the taxi to its destination (thereby achieving the goal)
  - But some are quicker, safer, more reliable, or cheaper than others.
  - Goals just provide a crude binary distinction between “happy” and “unhappy” states.
  - A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent. Because “happy” does not sound very scientific, economists and computer scientists use the term **utility** instead.

UTILITY-BASED AGENTS



Building a Utility-Based Agent

- Requires:
  - Model of percepts (sensor model)
  - Action generation algorithm (planner)
  - Performance metric
- Planning using decision theory
- Performance metric: acquire maximum rewards (or minimum cost)

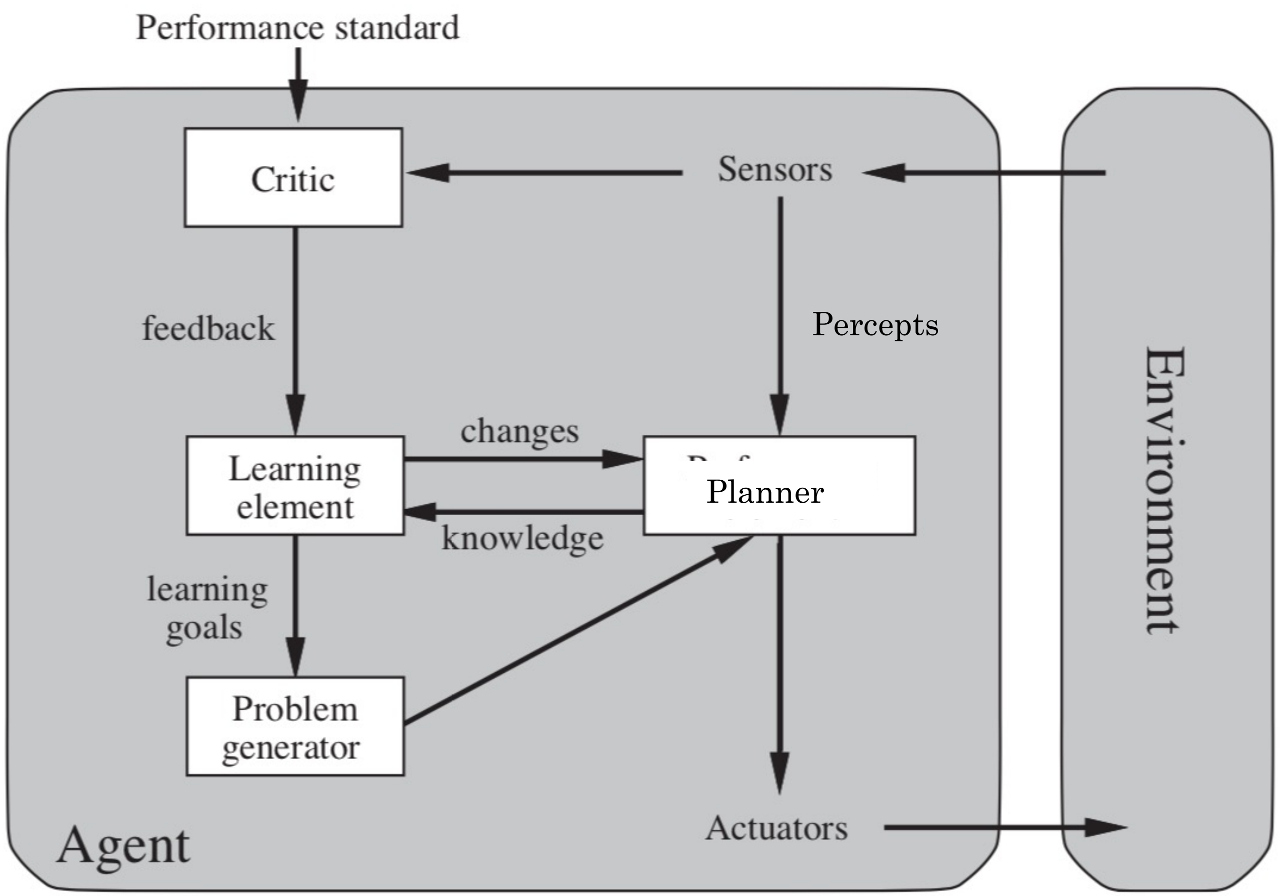
multi objective

LEARNING AGENTS

The **critic** provides feedback on how the agent is doing and determines how the performance element should be modified to do better in the future

Learns a model

exploration exploitation tradeoff:
eg: discover more restaurants other than the exploited ones



Building a Learning Agent

- Need a mechanism for updating models/rules/planners **on-line as it interacts with the environment**
- Need incremental techniques for machine learning