Intelligent Agents

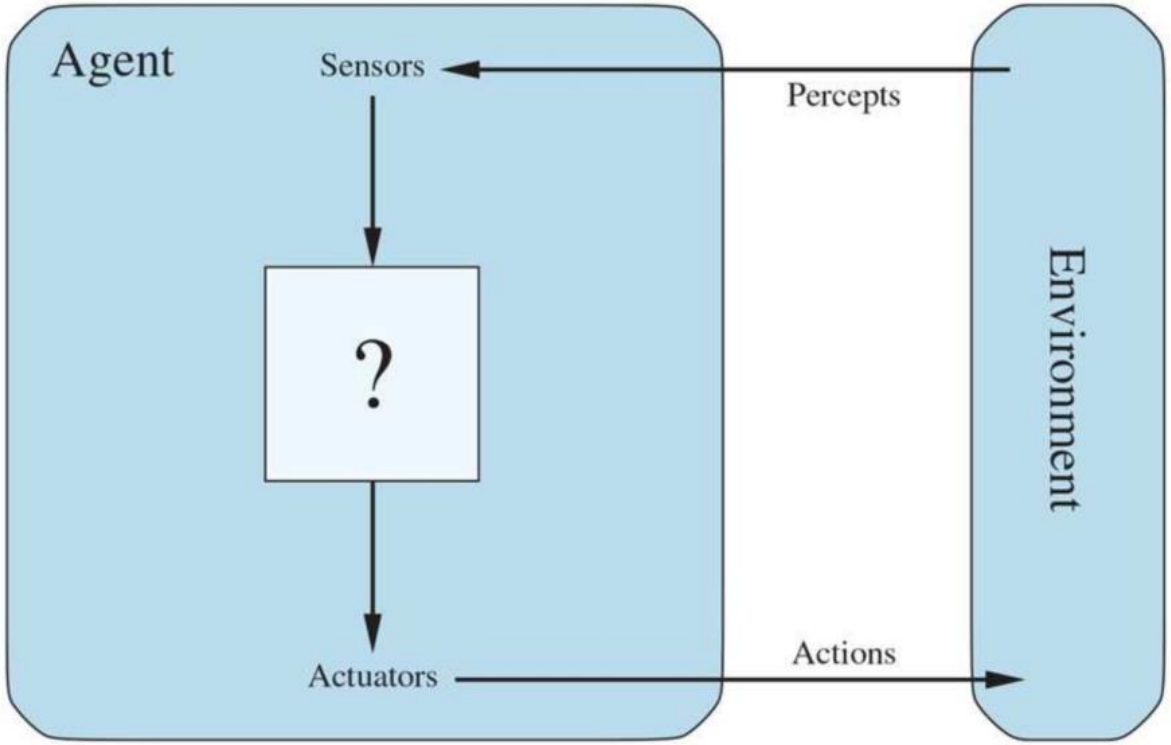
Dr. Sandareka Wickramanayake

Outline

- Definition of an agent
- Rationality
- Performance measures
- Definition of a Rational Agent
- The Task Environment
- Environment Types
- Agent Types

What is an Agent?

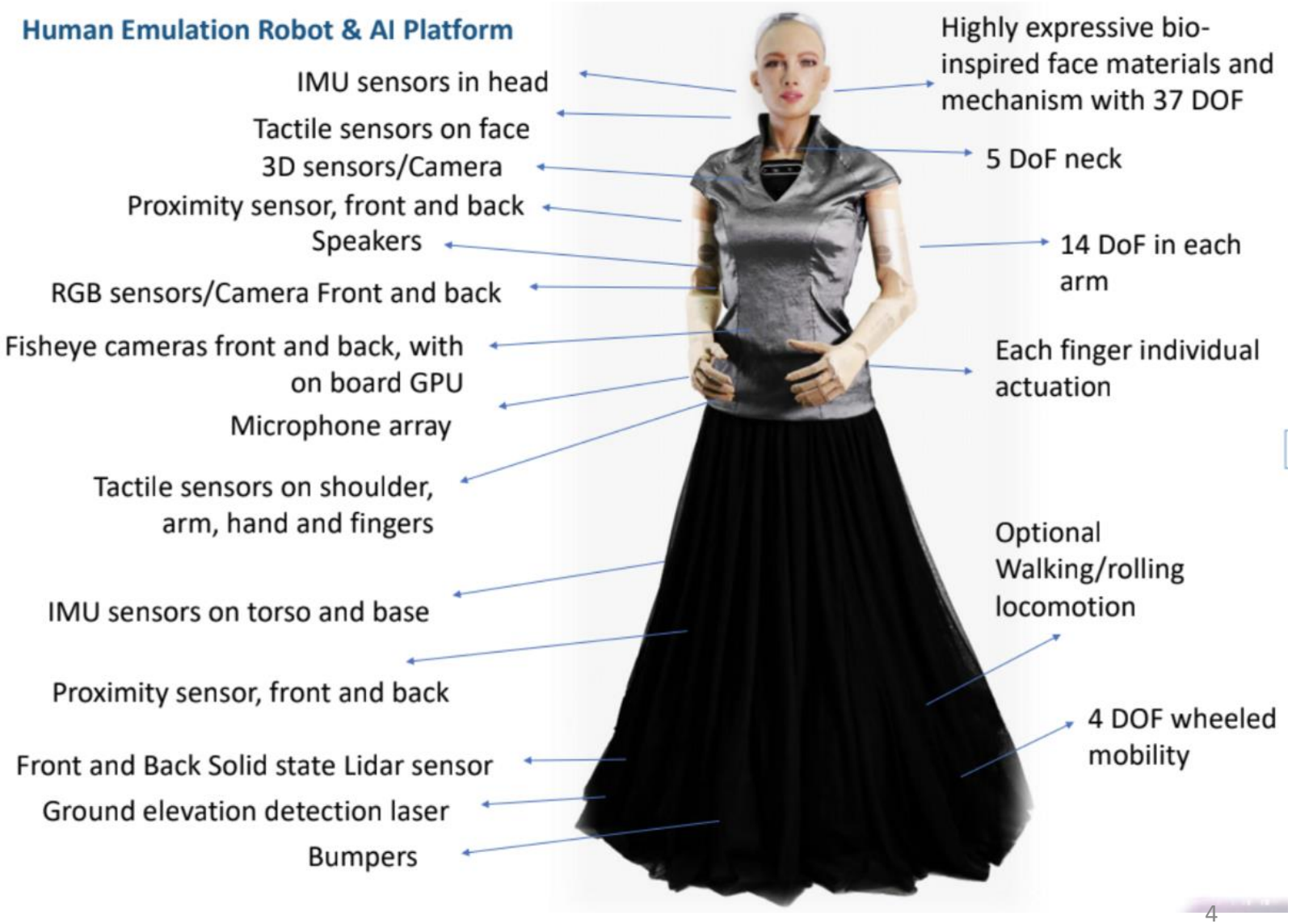
- "Anything" that can **PERCEIVE** its **ENVIRONMENT** through **SENSORS** and acts upon the environment through **ACTUATORS**
- Analogy
  - Human
  - Cleaning robot
  - Software agents



Agents interact with environments through sensors and actuators.

Examples of Agents

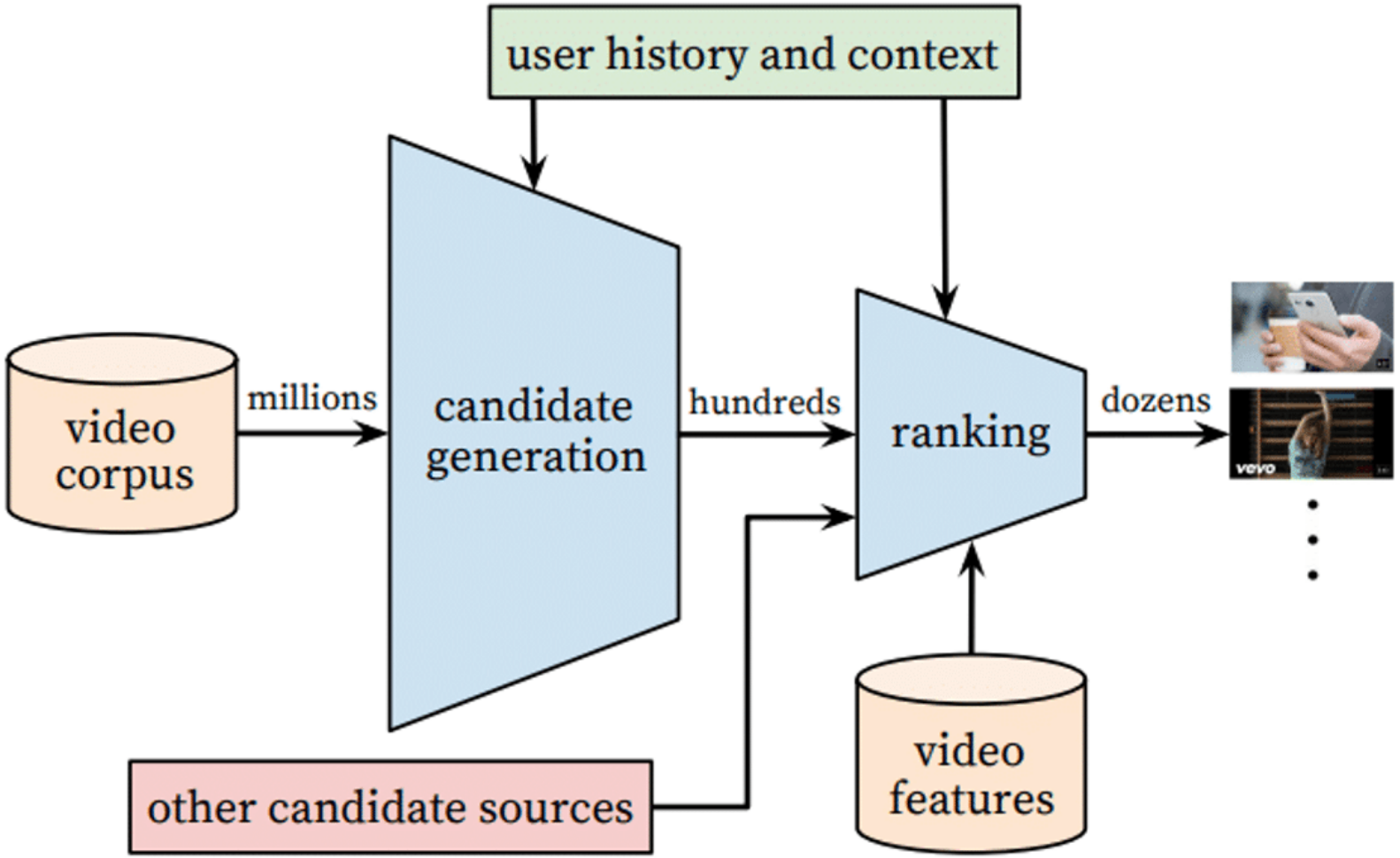
- Sophia



https://www.hansonrobotics.com/building-out-sophia-2020-an-integrative-platform-for-embodied-cognition/

Examples of Agents

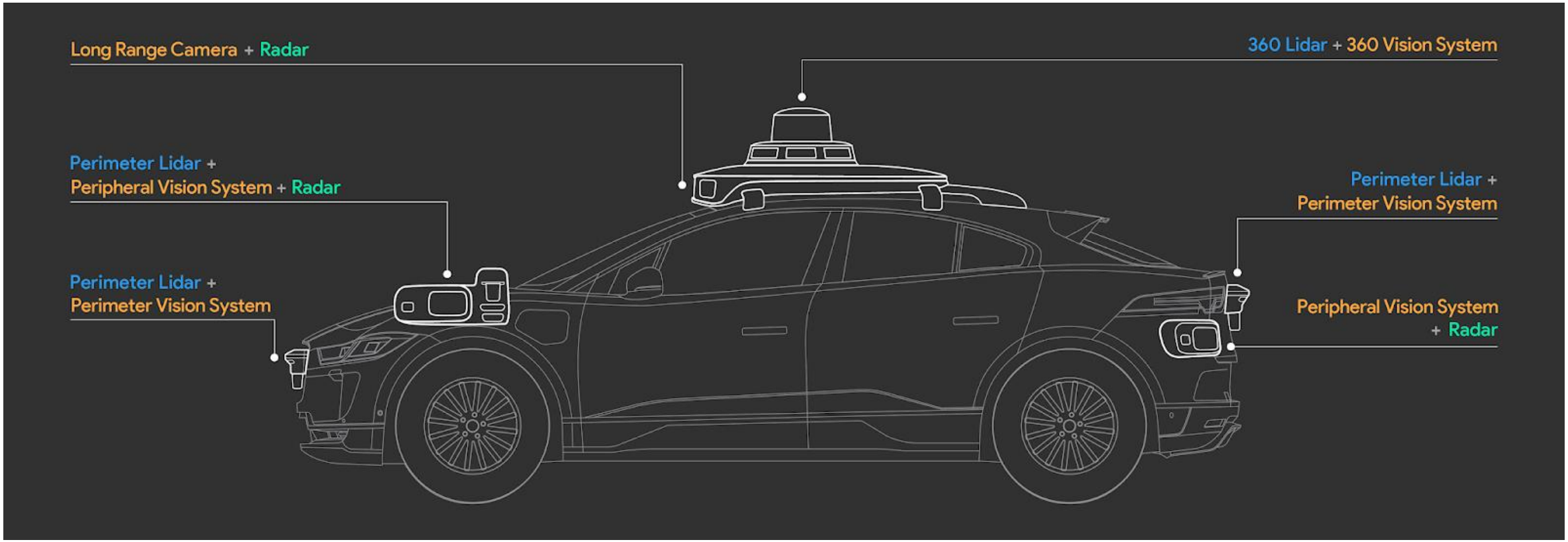
- Youtube



https://blog.hootsuite.com/how-the-youtube-algorithm-works/

Examples of Agents

- Waymo

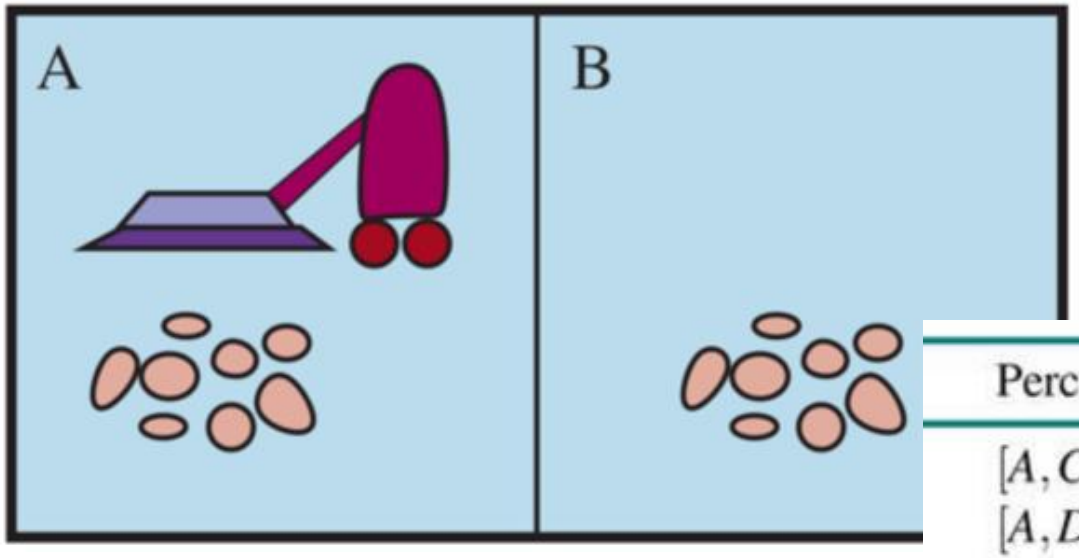


https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html

The Agent Function

- **Maps the agent's percept sequence to an action**
- **Percept sequence -** The complete history of everything an agent has perceived so far.
- An ideal mapping specifies an agent's actions for any given percept sequence.
- **Agent Program** - The implementation of the agent function.
- The effectiveness of the agent function is measured through a **performance measure**.

Example of an Agent Function



A vacuum-cleaner world with just two locations.

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

Partial tabulation of a simple agent function for the vacuum-cleaner world

Good Behavior: The Concept of Rationality

- **Rational Agent -** An agent who does the right thing
- The right thing at a given time depends on:
  - The performance measure
  - Everything the agent has perceived so far: The percept sequence
  - What the agent knows about the Environment
  - The actions the agent can do

Performance Measures

- **Consequentialism**: Evaluating an agent's behavior by its consequences.
- Evaluates how successful the agent's behavior is
- E.g.
  - For the cleaning robot, the performance criterion would be having a clean floor.
  - Award points for each clean square at each time step ☺.
  - Minus points for electricity consumed and noise generated ☹.
  - The robot should take actions that would maximize its points.
- Design the performance measure according to what we want in the environment and not how we think the agent should behave.

Definition of 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 an agent has.*

- Suppose an agent crosses a road and it can look only sideways (not up at the sky). After seeing that the road is clear, it starts to cross the road but something heavy fell from a tall building and crushed it. Was the agent rational?
- According to this definition, yes. Because it didn't perceive what was coming from the top.

Definition of Rational Agent

- Vacuum-cleaner agent
  - The performance measure awards one point for each clean square at each time step, over a “lifetime” of 1000-time steps.
  - The “geography” of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The $Right$ and $Left$ actions move the agent one square except when this would take the agent outside the environment, in which case the agent remains where it is.
  - The only available actions are $Right$, $Left$, and $Suck$.
  - The agent correctly perceives its location and whether that location contains dirt.

Building Rational Agents

- AI is about building rational agents
- A rational agent always does the right thing
- **Perfect Rationality**
  - Assumes the agent knows everything
  - Will always take the action that **maximizes its utility**
- **Bounded Rationality**
  - Proposed by Herbert Simon in 1958
  - Limited by the information it has
  - Use **approximate** methods to handle many tasks
  - Like the way a human mind works

Rationality

- Rational Action
  - The action that maximizes the expected value of the performance measure given the percept sequence to date.

- Questions
  1. Does it mean that it’s the best action?
  2. Does a rational action mean that it’s optimal?

Omniscience, Learning, and Autonomy

- Omniscience
  - The agent who knows about the ACTUAL outcome of its actions and acts accordingly
- Rationality
  - The agent who maximizes the EXPECTED outcome
  - Does not have to be omniscient.
  - The rational choice depends only on the percept sequence to date.
- This requires a rational agent to:
  - Gather information
  - Learn from perceptions

Omniscience, Learning, and Autonomy

- Agent's actions depend more on its own experience (gathered through sensors) - **More autonomous**
- Agents' actions depend more on knowledge of the environment that has been built in by the designer - **Less autonomous**

The Task Environment

- Defines the problems to which the rational agents attempt to provide solutions
- Consists of PEAS
  - **P**erformance Measure
  - **E**nvironment
  - **A**ctuators
  - **S**ensors
- In designing an agent, the first step must always be to specify the task environment as fully as possible.

Example: Automated Taxi Driver

- Performance Measurements:
  - Safe, fast, legal, comfortable trip, maximize profits, minimize the impact on other road users
- Environment:
  - Roads, other traffic, pedestrians, customers, police, weather
- Actuators:
  - Steering wheel, accelerator, brake, signal, horn, display, speech
- Sensors:
  - Cameras, Speedometer, radar, GPS, engine sensors, accelerometer, microphones, touchscreen

More Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Properties of Task Environments

- Observed from the agent's point of view
- **Fully observable** vs. **partially observable**
  - Fully observable: The agent's sensors give access to the complete state of the environment at each point in time.
  - Fully observable environments are convenient.
  - An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data
  - E.g.,
    - A vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares.
    - An automated taxi cannot see what other drivers are thinking.

Properties of Task Environments

- **Single-Agent vs Multi-Agent**
  - Single-agent – An agent solving a crossword puzzle
  - Multi-agent – An agent playing chess
  - Whether an entity in the environment is an agent or not depends.
    - The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior.

One agent trying to maximize its performance measure causes to minimize another agent's performance measure.
E.g., Chess

One agent trying to maximize its performance measure also causes to maximize another agent's performance measure.
E.g., Taxi-driving environment avoiding collisions.

Properties of Task Environments

- **Deterministic vs. Nondeterministic vs. Stochastic**
  - Whether the next state of the environment is completely determined by the current state and the action executed by the agent(s) or not.
  - Most real situations are nondeterministic.
    - E.g., Taxi driving
  - **Stochastic** – When the environment explicitly deals with probabilities.
    - There's a 25% chance of rain tomorrow

Properties of Task Environments

- **Episodic vs. Sequential**
  - **Episodic environment**
    - Agent's experience is divided into atomic episodes
    - Each episode consists of the agent perceiving and performing a single action
    - Next episode does not depend on the actions taken in the previous
    - E.g., An agent that has to spot defective parts on an assembly line
  - **Sequential environment**
    - The current decision could affect all future decisions
    - , E.g., Chess and taxi driving
  - Episodic actions are simpler because the agent does not THINK AHEAD

Properties of Task Environments

- **Dynamic Vs. Static**
  - Whether the environment can change while the agent's deliberating or not.
  - E.g., Taxi driving – Dynamic, Crossword puzzles – Static

- **Discrete Vs. Continuous**
  - Apply to the *state* of the environment, the way *time* is handled, and the *percepts* and *actions* of the agent.
  - E.g.,
    - Taxi driving is a continuous state and continuous-time problem.
    - Taxi-driving actions are also continuous.
    - Chess: Fixed number of moves

Pop Quiz

- Environment types are, fully/partially observable, deterministic/stochastic/strategic, episodic/non-episodic, dynamic/static, discrete/continuous , single/multi-agent.
- Pick which of these properties correctly represent the following environments.

	Fully/ Partially Observable	**Deterministic/ Stochastic**	**Discrete/ continuous**	**Episodic/ Non-Episodic**	**Static/ Dynamic**
Checkers					
Driving a robot car					

Pop Quiz

- Environment types are, fully/partially observable, deterministic/stochastic/strategic, episodic/non-episodic, dynamic/static, discrete/continuous , single/multi-agent.
- Pick which of these properties correctly represent the following environments.

	Fully/ Partially Observable	**Deterministic/ Stochastic**	**Discrete/ continuous**	**Episodic/ Non-Episodic**	**Static/ Dynamic**
Checkers	Fully	Deterministic	Discrete	Sequential	Static
Driving a robot car	Partially	Stochastic	Continuous	Sequential	Dynamic

The Structure of Agents

- **Agent program** – The mapping from percepts to actions
  - Implements the agent function.
- **Agent architecture** – The computing device with physical sensors and actuators on which the agent program runs.
  - The architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.

$$Agent = Architecture + Program$$

Agent Program

```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

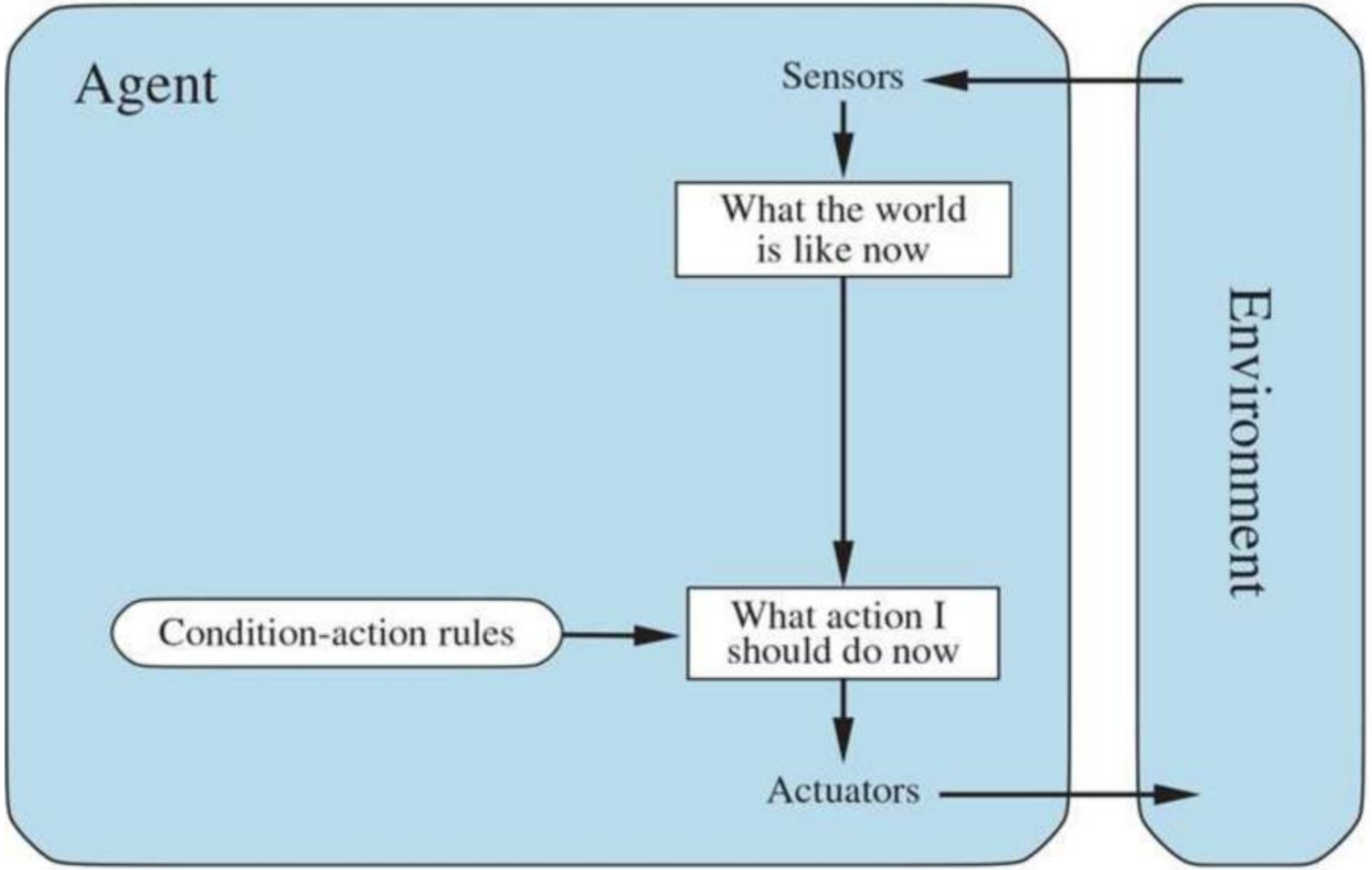
Agent Program Types

- Simple reflex agents
- Model-based agents
- Goal-based agents
- Utility-based agents

Simple Reflex Agents

- Simplest kind of agent
- Actions depend only on present percepts ignoring the history
- Limited Intelligence
- Works best in an observable environment
- E.g., Vacuum agent

Simple Reflex Agents



Simple Reflex Agents

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Specific program to one vacuum environment.

```
function REFLEX-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
```

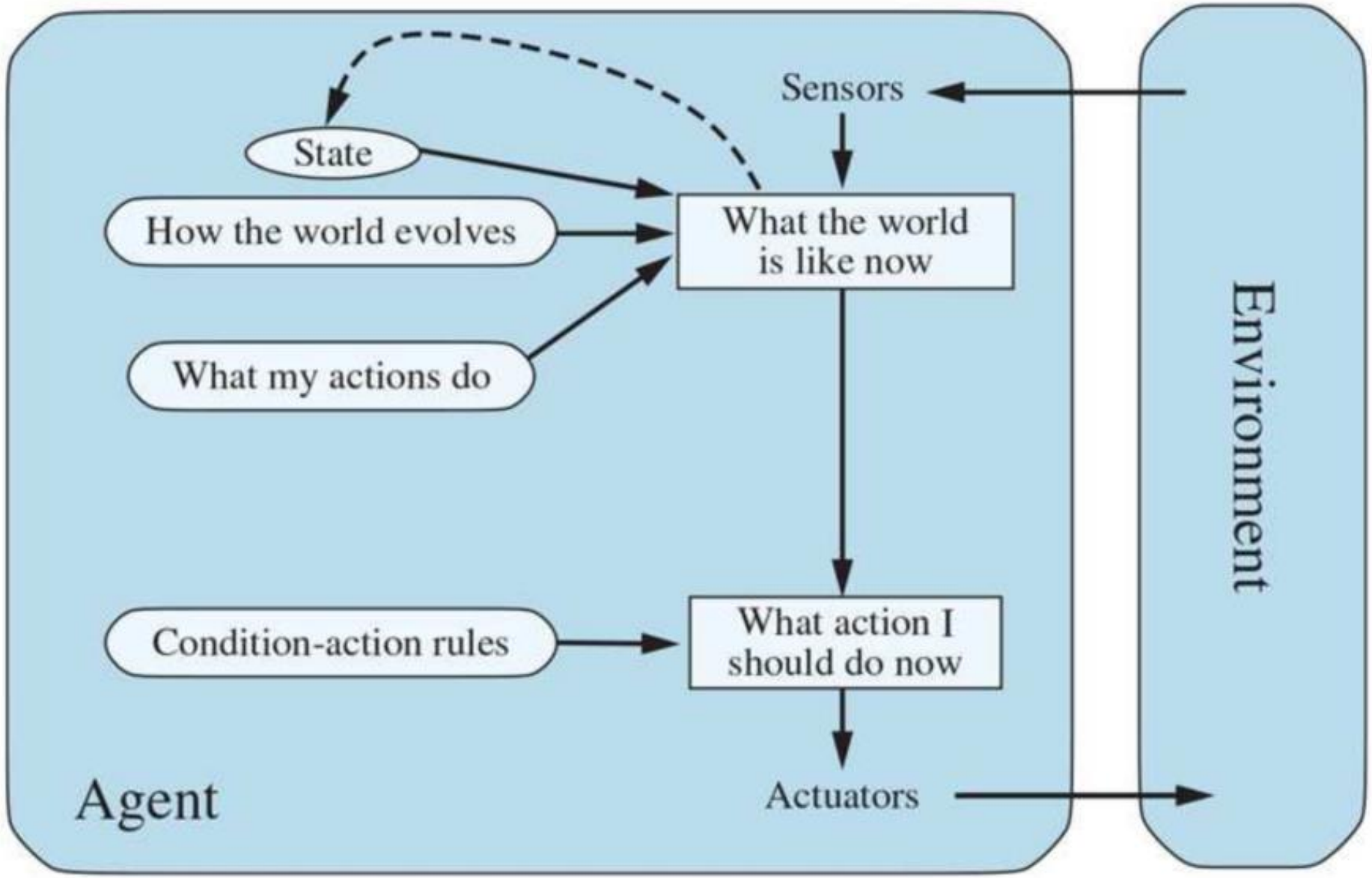
Model-based Reflex Agents

- Maintains an **internal state** that depends on the percept history.
- Keeps track of the **perception history**
  - Useful in partially observable environments
  - If an agent wants to cross the road, it'll look right, then left. It doesn't see what's on the right when it's looking left: Keeping history will be useful here.
  - Applying brakes when the vehicle in the front brakes.
    - Internal state – Previous frame from the camera
  - Changing lanes
    - Internal state - Where the other cars are

Model-based Reflex Agents

- How to maintain the internal state
- Required to store two kinds of knowledge
  - **Transition model** of the world – How the world works
    - The effects of the agent's actions
    - How the world evolves independently of the agent
  - **Sensor model** - how the state of the world is reflected in the agent's percepts.

Model-based Reflex Agents



Model-based Reflex Agents

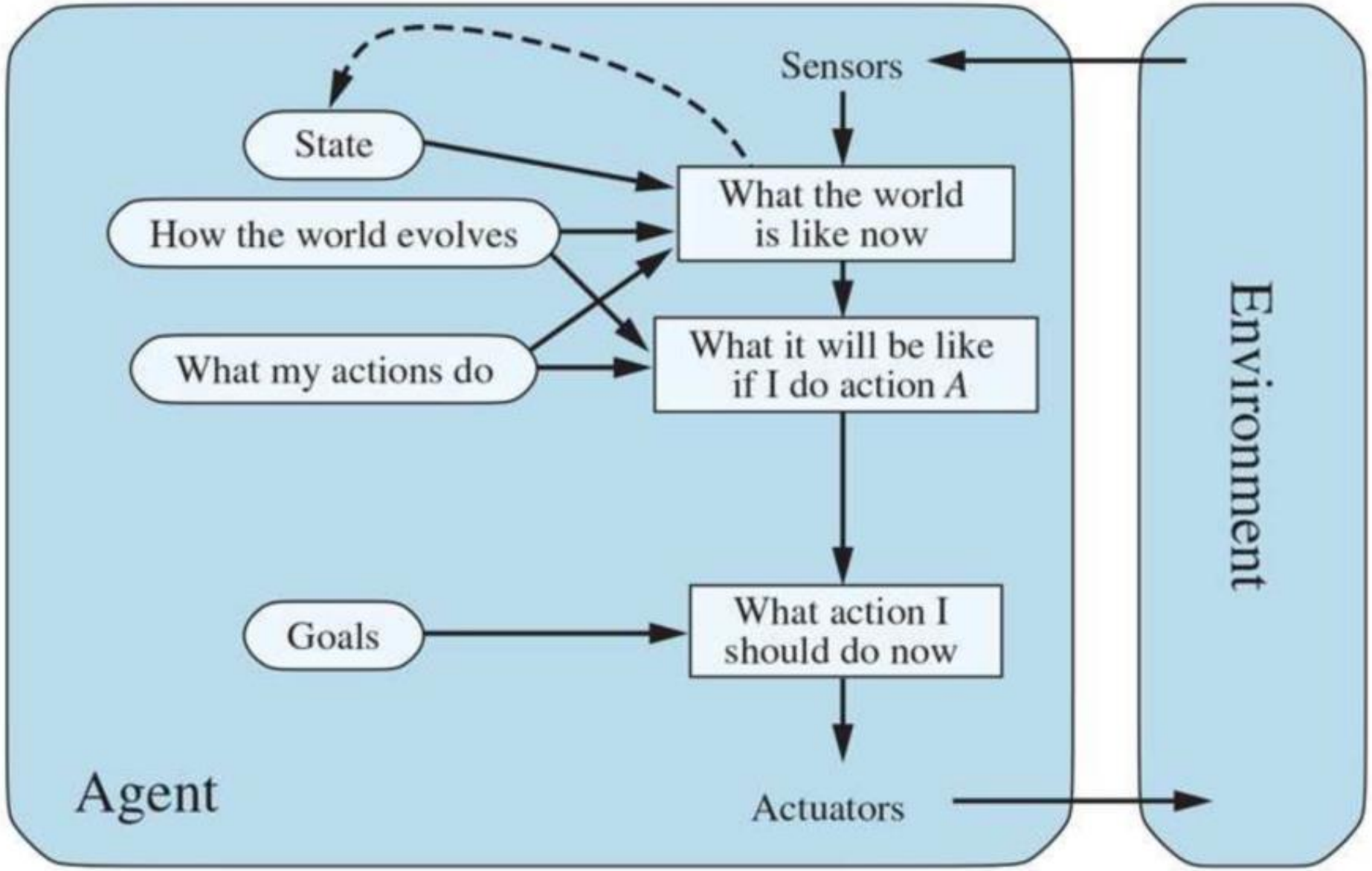
function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on the current state and action
    sensor_model, a description of how the current world state is reflected in the agent's percepts
    rules, a set of condition–action rules
    action, the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

Goal-based Agents

- Combines prior knowledge and perceptions and takes actions **to achieve a goal.**
- E.g., Robot car
  - Goal: Customers Destination (Say Ratmalana)
  - Actions: at a junction, turn left/right/go straight
- More flexible than reflex agents
  - Changing the goal will change the actions appropriately. E.g., Robot Taxi driver

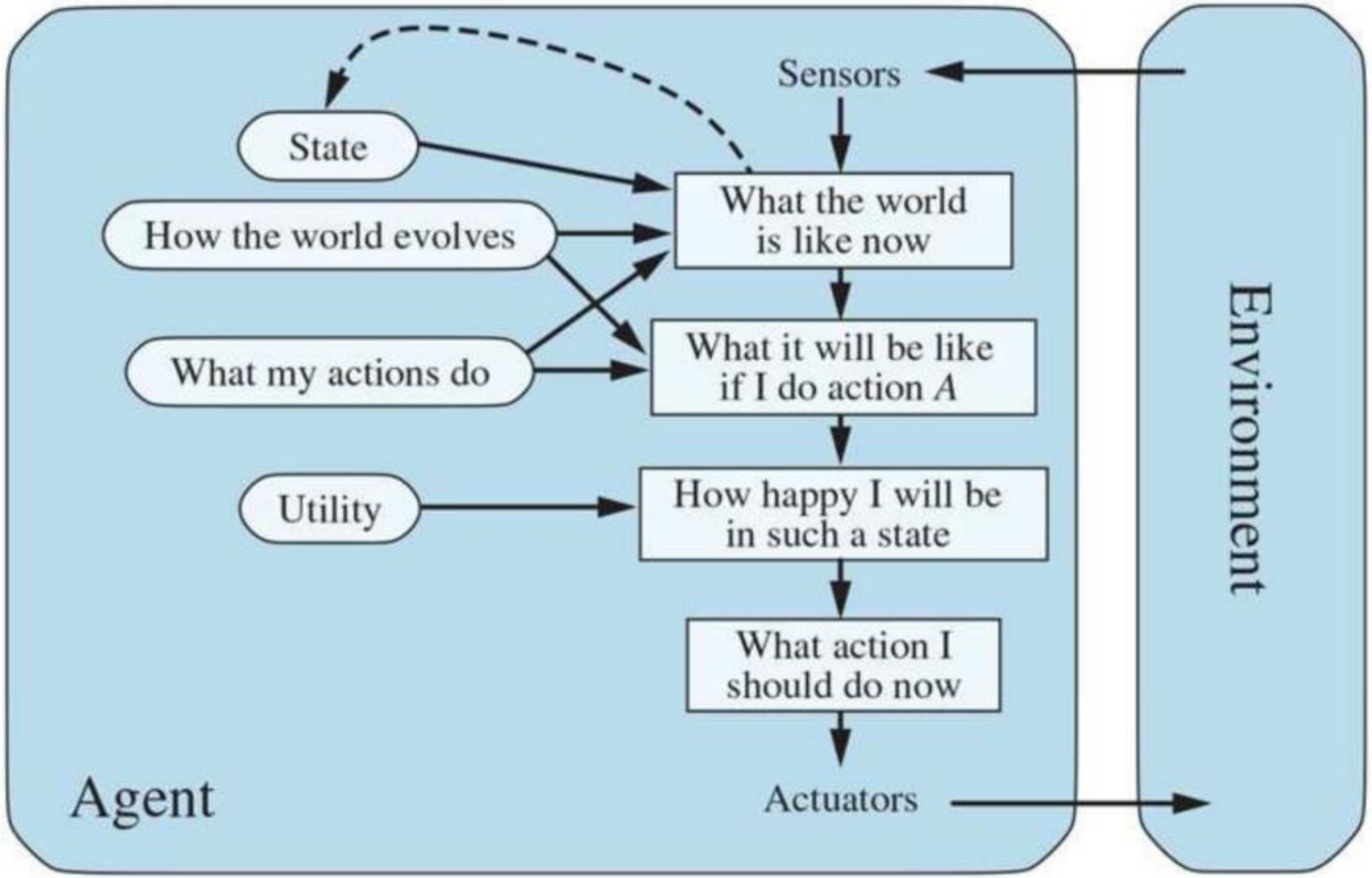
Goal-based Agents



Utility-based Agents

- A UTILITY FUNCTION maps a state or a sequence of states onto a real number.
- Describes the degree of *happiness.*
- Is used in cases where goals are inadequate
  - Conflicting goals. E.g., Speed and safety
    - Utility function specifies a tradeoff
  - Choosing the best fit out of many options
    - Taxi driver: Multiple routes/actions to get to the same destination. Which is best?

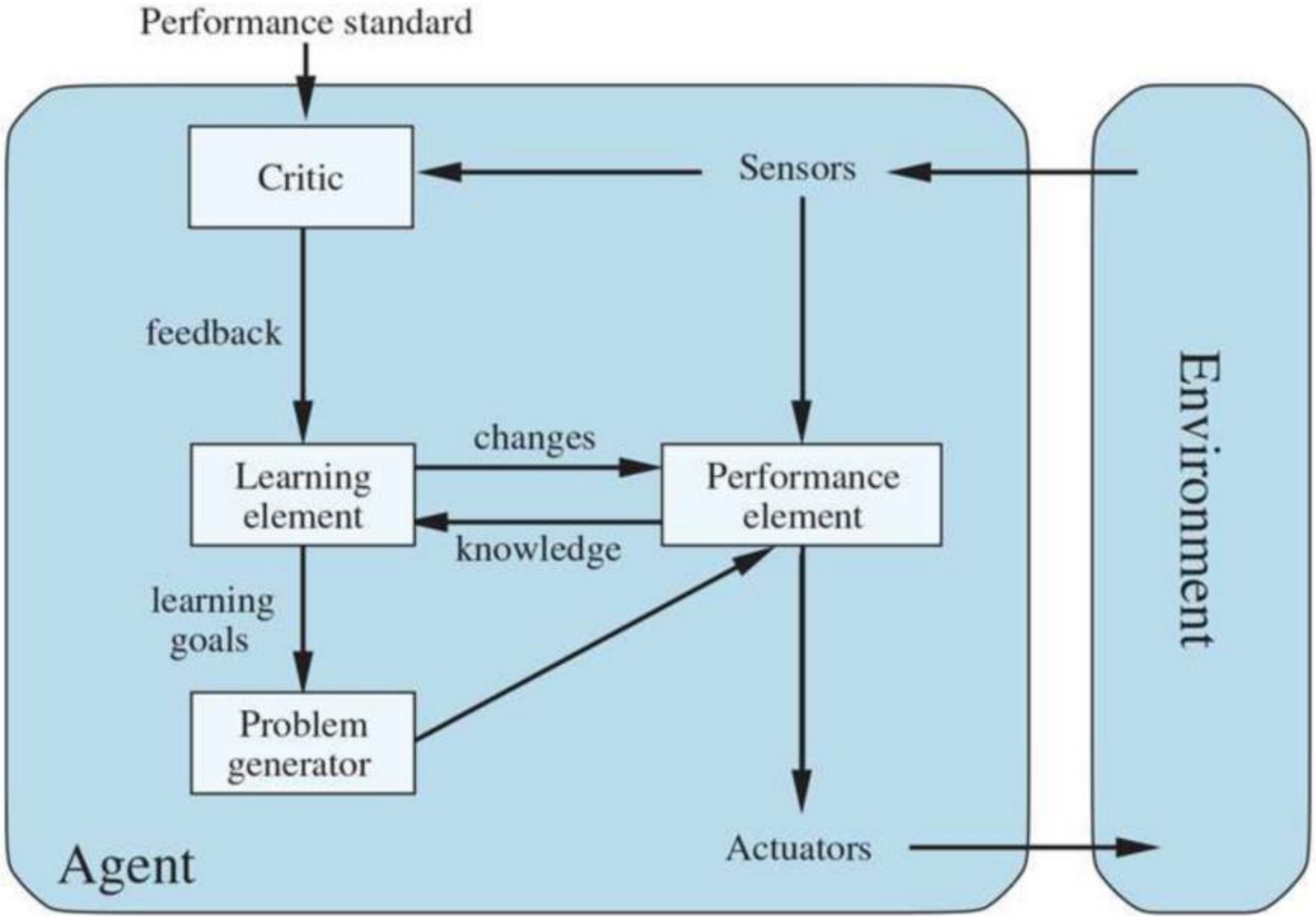
Utility-based Agents



Learning Agents

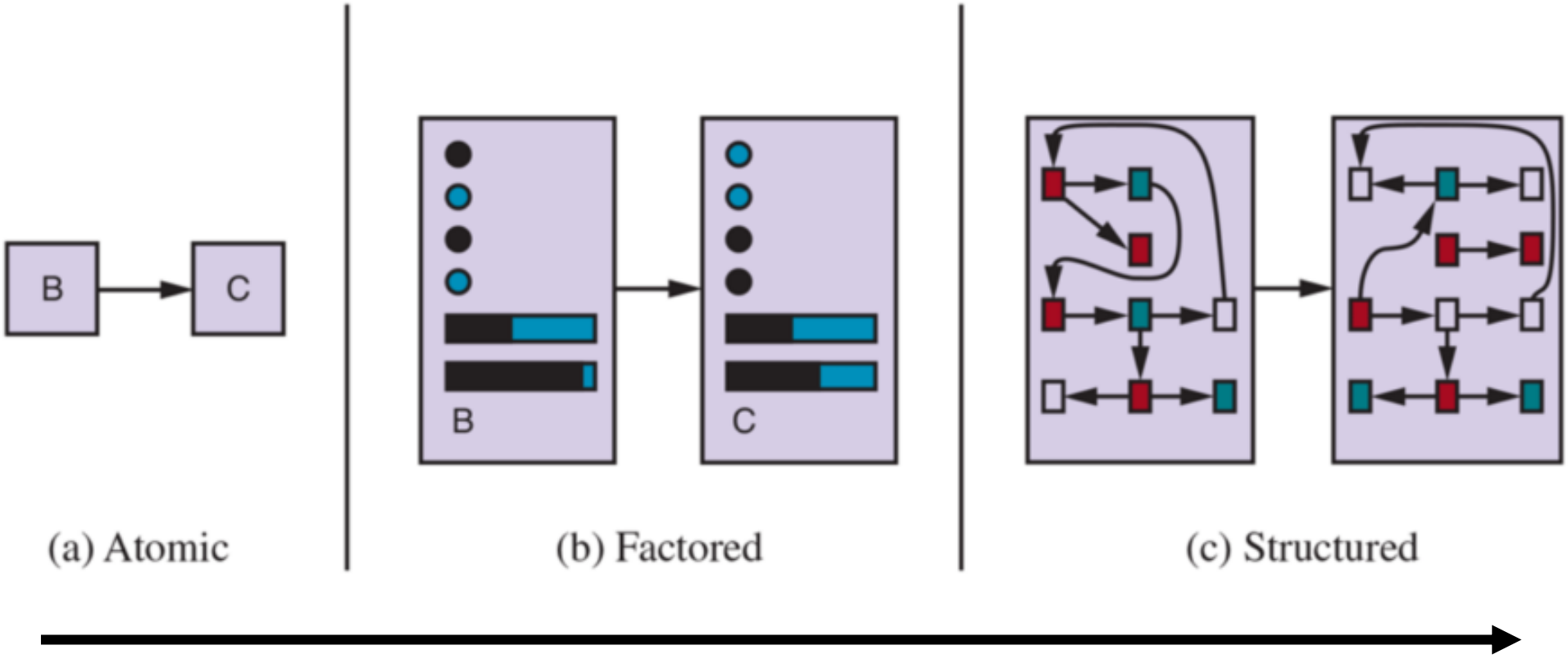
- Performance Element
  - Takes in percepts and decides on actions: Up to this slide, this is what we called the entire agent
- Critic
  - Evaluates how the agent is performing and gives feedback
- Learning Element
  - Takes feedback from the *Critic* and modifies the *performance element* to do better in the future
- Problem Generator
  - Suggests exploratory actions leading to new and informative experiences
    - These actions may be suboptimal, but will lead to the discovery of better actions: Learn thorough mistakes

Learning Agents



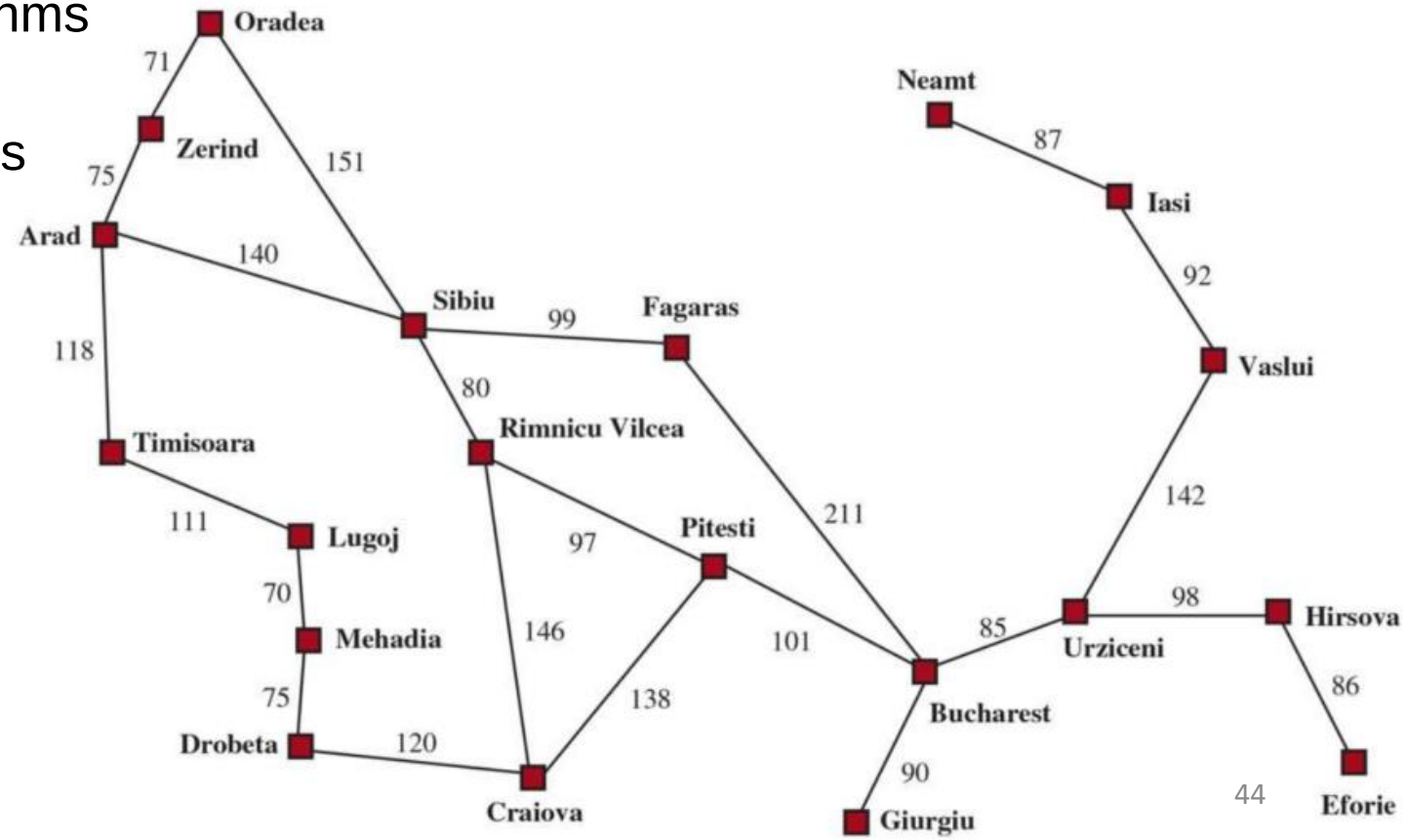
How the Components of Agent Programs Work?

- The representations in the order of increasing complexity and expressive power—atomic, factored, and structured.



How the Components of Agent Programs Work?

- Atomic representation
  - A state is a black box with no internal structure
  - Algorithms
    - Standard searching algorithms
    - Hidden Markov Models
    - Markov Decision Processes



How the Components of Agent Programs Work?

- Factored representation
  - A state consists of a set of **variables** or **attributes**, each of which can have a value.
  - Pay attention to
    - How much gas is in the tank
    - Our current GPS coordinates
    - Whether or not the oil warning light is working
    - How much money do we have for tolls
    - What station is on the radio
    - Etc.

How the Components of Agent Programs Work?

- Factored representation
  - Two different factored states can share some attributes (such as being at some GPS location) and not others (such as having lots of gas or having no gas)
  - Makes it much easier to work out how to turn one state into another.
  - Algorithms
    - Constraint satisfaction algorithms
    - Propositional logic
    - Bayesian networks
    - Various machine learning algorithms

How the Components of Agent Programs Work?

- Structured representation
  - A state includes objects, each of which may have attributes of its own as well as relationships to other objects.
  - Algorithms
    - First-order logic
    - First-order probability models