CSC520 - Artificial Intelligence

Lecture 3

Dr. Scott N. Gerard

North Carolina State University

Jan 14, 2025

In The News

'Worst in Show' CES Products Risk Data, Cause Waste

e-commerce site iFixit released its fourth annual 'Worst in Show' awards for products showcased at this year's Consumer Electronics Show. Ultrahuman's Rare Luxury Smart Ring was named "least repairable"; it costs $2,200 and has a battery that cannot be replaced without destroying the device. Bosch's Revol crib was panned for collecting "excessive" data about babies. "We're seeing more and more of these things that have basically surveillance technology built into them," said iFixit's Liz Chamberlain.
[» Read full article]

Associated Press; Sarah Parvini (January 9, 2025)

Agenda

- Agent Function vs Program vs Architecture
- Agent Types
  - Simple reflex
  - Model-based reflex
  - Goal-based
  - Utility-based

Agent Function vs. Program vs Architecture

- *Agent function* maps a percept sequence to an action; external characterization

$$f : p_0, p_1, \ldots p_n \rightarrow a_i$$

Agent Function vs. Program vs Architecture

- *Agent function* maps a percept sequence to an action; external characterization

$$f : p_0, p_1, \ldots p_n \rightarrow a_i$$

- *Agent program* is a concrete implementation; internal characterization

Agent Function vs. Program vs Architecture

- *Agent function* maps a percept sequence to an action; external characterization

$$f : p_0, p_1, \ldots p_n \rightarrow a_i$$

- *Agent program* is a concrete implementation; internal characterization

- *Agent architecture* is the computing device with sensors and actuators

Agent Function vs. Program vs Architecture

- *Agent function* maps a percept sequence to an action; external characterization

$$f : p_0, p_1, \ldots p_n \rightarrow a_i$$

- *Agent program* is a concrete implementation; internal characterization

- *Agent architecture* is the computing device with sensors and actuators
- *Agent* is the combination of architecture and program
  $agent = architecture + program$

Simple Reflex Agent

- Selects action based on the current percept

Simple Reflex Agent

- Selects action based on the current percept
- Does not track history of percepts

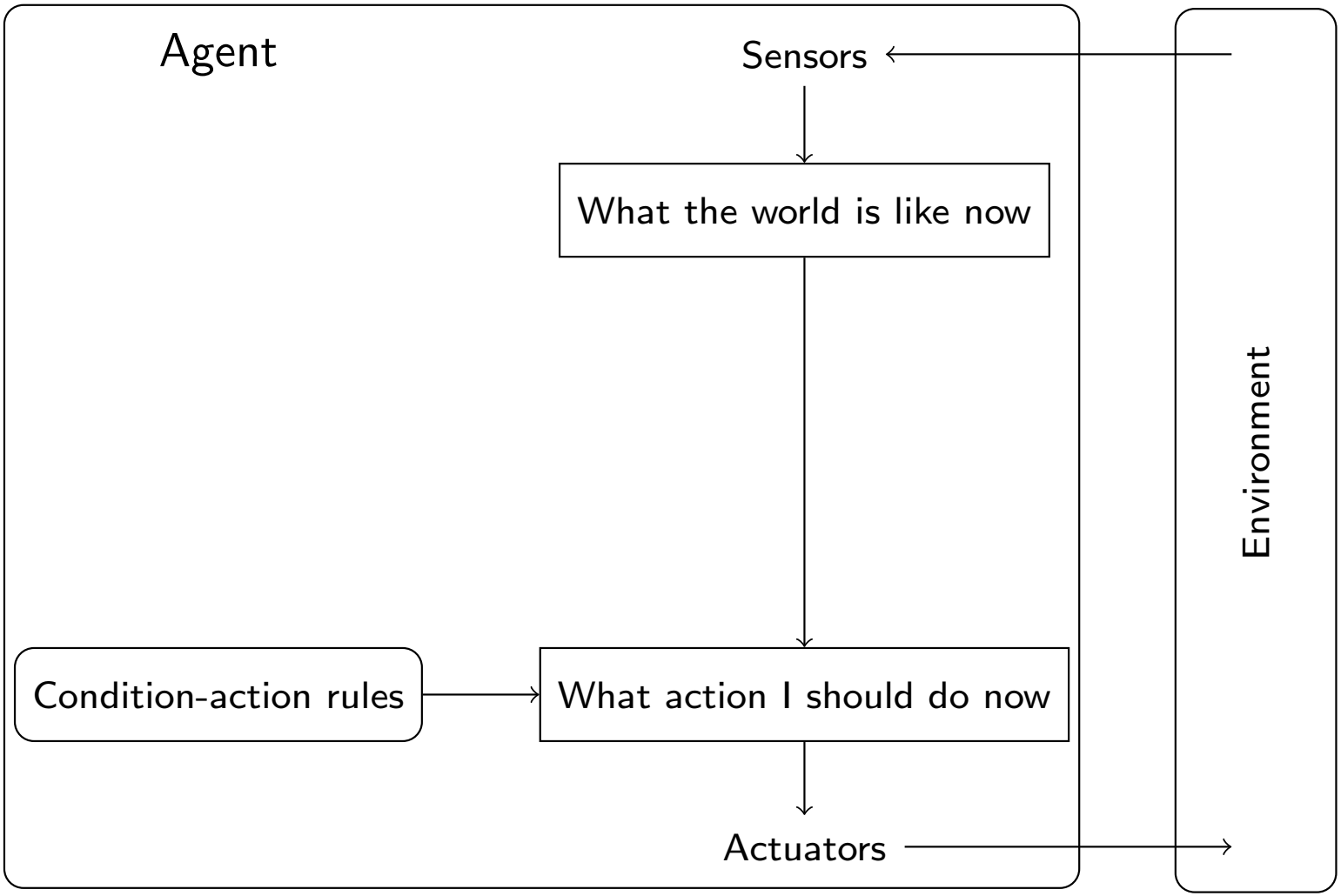
Simple Reflex Agent

- Selects action based on the current percept
- Does not track history of percepts
- Simple but can fail if the environment is not fully observable

Simple Reflex Agent

- Selects action based on the current percept
- Does not track history of percepts
- Simple but can fail if the environment is not fully observable
  - Agent cannot figure out if a car infront is braking by observing only a single image frame

Simple Reflex Agent



Simple Reflex Agent

function SIMPLE-REFLEX-AGENT(*percept*) **return** an action

persistent: *rules*: a set of condition-action rules

state ← INTERPRET-INPUT(*percept*)

rule ← RULE-MATCH(*state*, *rules*)

action ← *rule*.ACTION

return action

Model-based Reflex Agent

- Keeps an internal state of the world to address partial observability

Model-based Reflex Agent

- Keeps an internal state of the world to address partial observability
- Internal state is updated using two kinds of knowledge

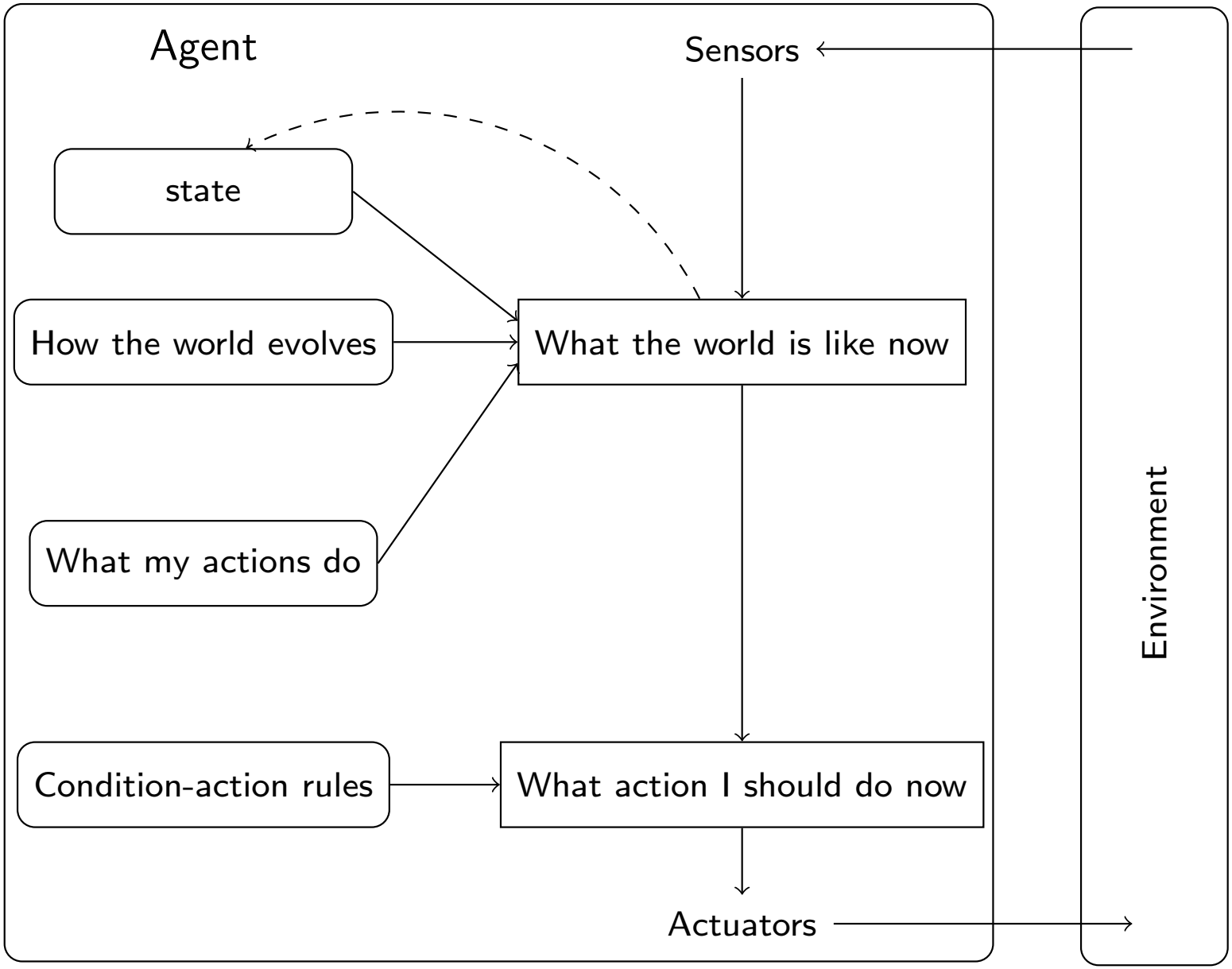
Model-based Reflex Agent

- Keeps an internal state of the world to address partial observability
- Internal state is updated using two kinds of knowledge
  - *Transition model*
    - ★ How state changes based on the agent action?
      e.g. car turns right if the steering wheel is rotated clockwise
    - ★ How state changes independent of the agent action?
      e.g. car's camera gets wet when it rains

Model-based Reflex Agent

- Keeps an internal state of the world to address partial observability
- Internal state is updated using two kinds of knowledge
  - *Transition model*
    - ★ How state changes based on the agent action?
      e.g. car turns right if the steering wheel is rotated clockwise
    - ★ How state changes independent of the agent action?
      e.g. car's camera gets wet when it rains
  - *Sensor model*
    - ★ What is the current state given the agent's percepts?
      e.g. red tail lights appearing in the camera image means the car is braking

Model-based Reflex Agent



Model-based Reflex Agent

function MODEL-BASED-REFLEX-AGENT(*percept*) **return** an action
  persistent:
    state: current world state
    transition_model: next state given the current state and action
    sensor_model: the state given the 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 Agent

- Reflex agents act based on current world state only, which is insufficient
  - A taxi cannot decide which way to turn without knowing the destination, i.e. the goal

Goal-based Agent

- Reflex agents act based on current world state only, which is insufficient
  - A taxi cannot decide which way to turn without knowing the destination, i.e. the goal
- Goal-based agent employs goals with world state, transition and sensor models

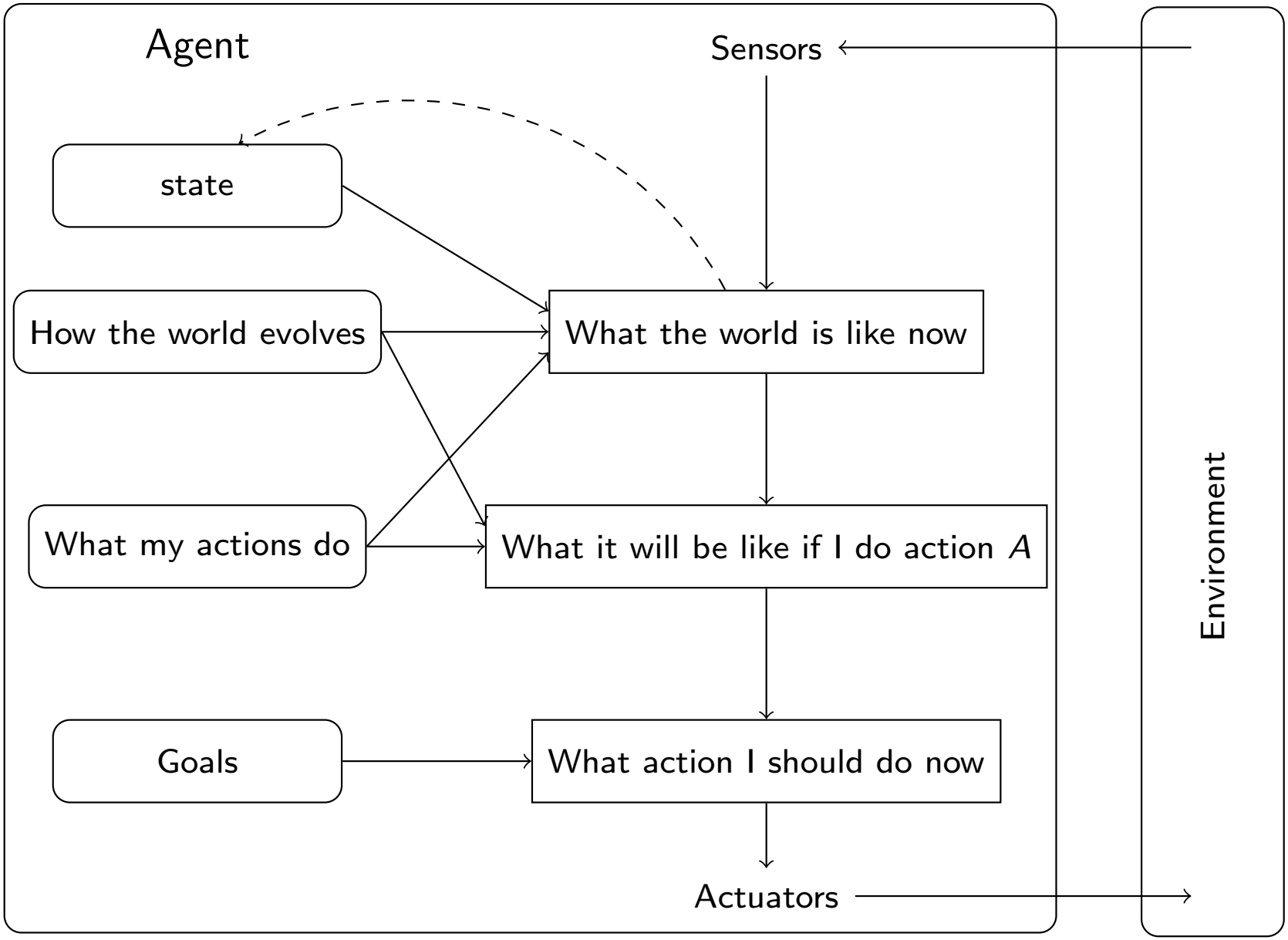
Goal-based Agent

- Reflex agents act based on current world state only, which is insufficient
  - A taxi cannot decide which way to turn without knowing the destination, i.e. the goal
- Goal-based agent employs goals with world state, transition and sensor models
- Compute *sequence* of actions to achieve goals using search and planning techniques

Goal-based Agent

- Reflex agents act based on current world state only, which is insufficient
  - A taxi cannot decide which way to turn without knowing the destination, i.e. the goal
- Goal-based agent employs goals with world state, transition and sensor models
- Compute *sequence* of actions to achieve goals using search and planning techniques
- Goals offer more flexibility in building agents

Goal-based Agent



Goal-based Agent

function GOAL-BASED-AGENT(*percept*) **return** an action

persistent:

state: current world state

transition_model: next state given the current state and action

sensor_model: the state given the percepts

goal: description of the desired goal state

plan: a sequence of actions to take, initially none

action: the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)

if GOAL-ACHIEVED(*state*, *goal*) **then**

return *SUCCESS*

if *plan* is *none* **then**

plan ← PLAN(*state*, *goal*, *transition_model*, *sensor_model*)

action ← FIRST(*plan*)

plan ← REST(*plan*)

return action

Utility-based Agent

- Goals specify desired states but not the quality of the plans
  - E.g. Many paths from city A to city B, some will be preferred since they are shorter or faster

Utility-based Agent

- Goals specify desired states but not the quality of the plans
  - E.g. Many paths from city A to city B, some will be preferred since they are shorter or faster
- Utility-based agent employs *utility function* while computing a plan

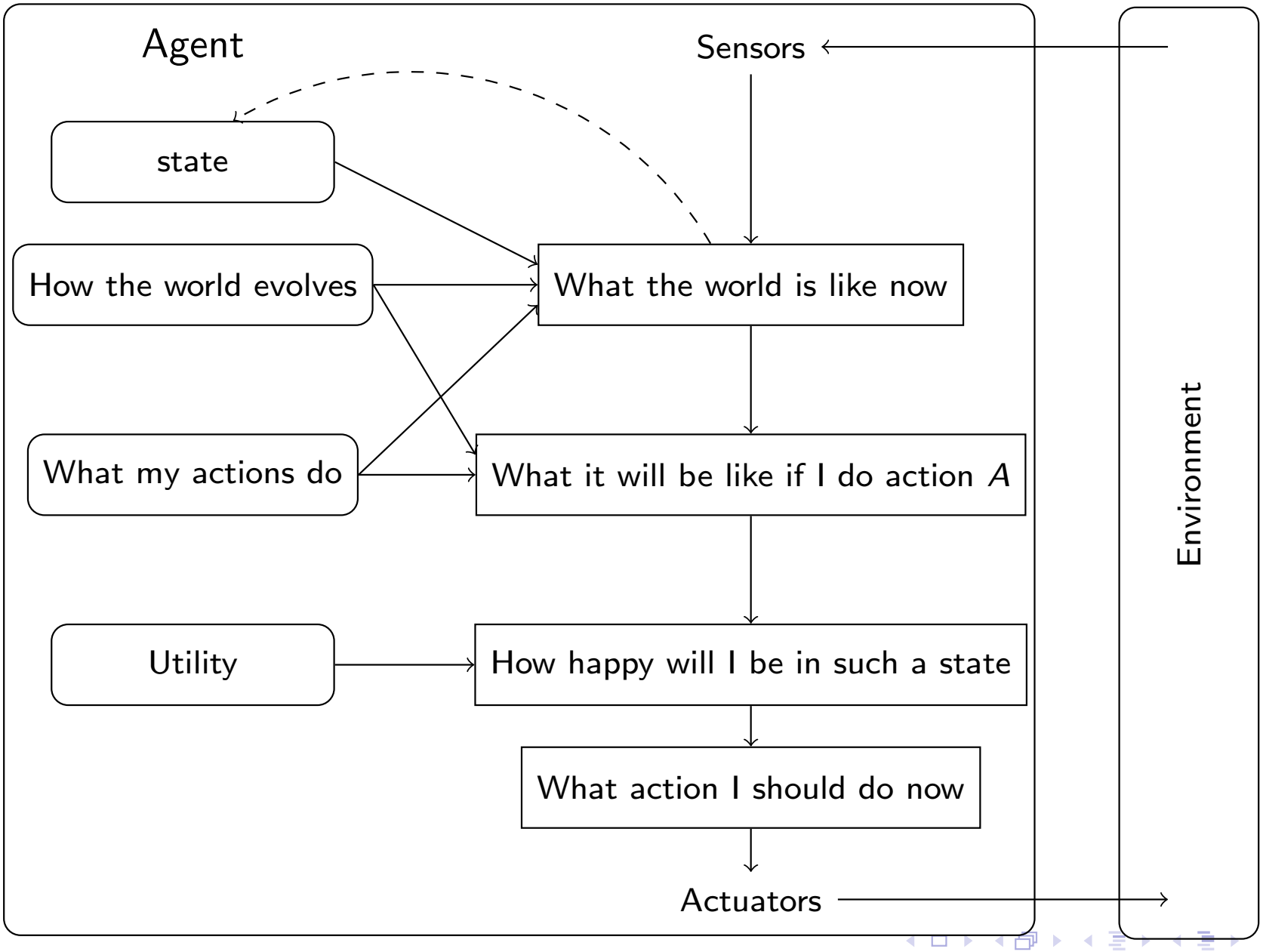
Utility-based Agent

- Goals specify desired states but not the quality of the plans
  - E.g. Many paths from city A to city B, some will be preferred since they are shorter or faster
- Utility-based agent employs *utility function* while computing a plan
- Utility function is an internalization of performance measure
  - A *rational* agent maximizes its expected utility

Utility-based Agent

- Goals specify desired states but not the quality of the plans
  - E.g. Many paths from city A to city B, some will be preferred since they are shorter or faster
- Utility-based agent employs *utility function* while computing a plan
- Utility function is an internalization of performance measure
  - A *rational* agent maximizes its expected utility
- Can handle cases when the agent has multiple goals and when their success is not guaranteed

Utility-based Agent



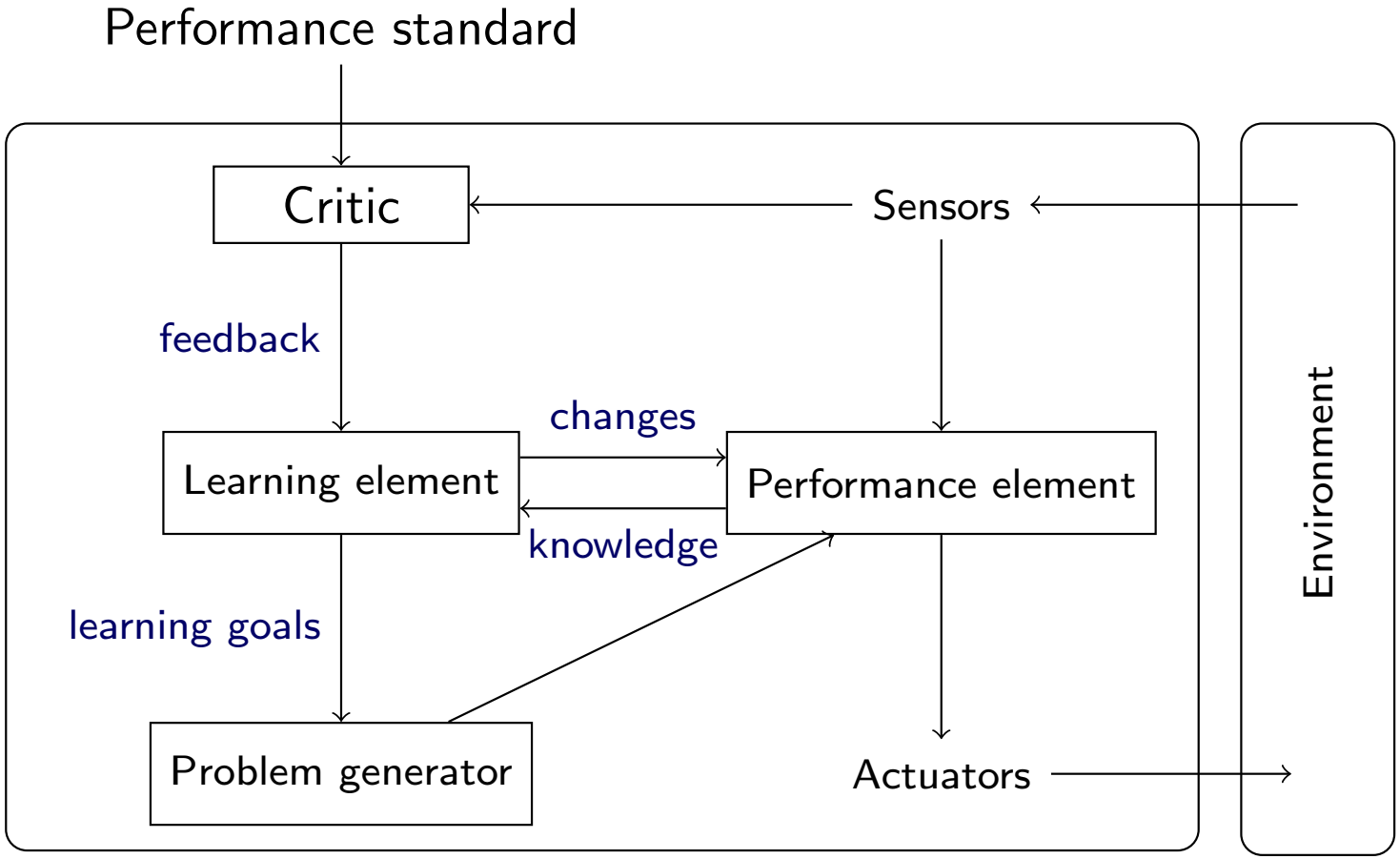
Learning Agent

- Agents that learn from experience
  - What my actions do?
  - How the world evolves?

Learning Agent

- Agents that learn from experience
  - What my actions do?
  - How the world evolves?
- Any agent (model-based, goal-based, utility-based) can be built as a learning agent

Learning Agent



Learning Agent

- Learning element is responsible for making improvements
- Performance element is responsible for selecting actions
  - That is, any of the agents we have studied thus far
- Critic provides feedback on how well the agent is doing with respect to some performance standard
- Problem generator suggests exploratory actions to the agent

Environment Representation

- Model the environment at an appropriate level of abstraction

Environment Representation

- Model the environment at an appropriate level of abstraction
  - E.g. No need to capture the amount of dust on a chess board

Environment Representation

- Model the environment at an appropriate level of abstraction
  - E.g. No need to capture the amount of dust on a chess board
- Environment representations with increasing level of expressiveness

Environment Representation

- Model the environment at an appropriate level of abstraction
  - E.g. No need to capture the amount of dust on a chess board
- Environment representations with increasing level of expressiveness
  - Atomic
    - ★ Black-box without any internal structure
      E.g. state in a city routing task:{CityA, CityB, CityC, CityD, . . . }

Environment Representation

- Model the environment at an appropriate level of abstraction
  - E.g. No need to capture the amount of dust on a chess board
- Environment representations with increasing level of expressiveness
  - Atomic
    - ★ Black-box without any internal structure
      E.g. state in a city routing task:{CityA, CityB, CityC, CityD, . . . }
  - Factored
    - ★ Vector of attributes (Boolean, real, symbol)
      {(CityA, 1.5gal, LIGHT_ON), (CityB, 20gal, LIGHT_OFF), . . . }

Environment Representation

- Model the environment at an appropriate level of abstraction
  - E.g. No need to capture the amount of dust on a chess board
- Environment representations with increasing level of expressiveness
  - Atomic
    - ★ Black-box without any internal structure
      E.g. state in a city routing task:{CityA, CityB, CityC, CityD, . . . }
  - Factored
    - ★ Vector of attributes (Boolean, real, symbol)
      {(CityA, 1.5gal, LIGHT_ON), (CityB, 20gal, LIGHT_OFF), . . . }
  - Structured
    - ★ Objects related to other objects
      {(CarA[16.5gal, LIGHT_ON, 60mph], RoadA[paved, 4lane], CarA-on-RoadA),
      (CarA[10.5gal, LIGHT_ON, 30mph], RoadB[gravel, 1lane], CarA-on-RoadB),
      . . . }

Problem Solving Agent

- An agent that plans ahead

Problem Solving Agent

- An agent that plans ahead
- Employs *search* to compute a plan
  - *Atomic* state representation

Problem Solving Agent

- An agent that plans ahead
- Employs *search* to compute a plan
  - *Atomic* state representation
- Problem solving process
  - Formulate the goal

Problem Solving Agent

- An agent that plans ahead
- Employs *search* to compute a plan
  - *Atomic* state representation
- Problem solving process
  - Formulate the goal
  - Formulate the problem

Problem Solving Agent

- An agent that plans ahead
- Employs *search* to compute a plan
  - *Atomic* state representation
- Problem solving process
  - Formulate the goal
  - Formulate the problem
  - Search to find a plan
    - ★ Agent *simulates* sequences of actions

Problem Solving Agent

- An agent that plans ahead
- Employs *search* to compute a plan
  - *Atomic* state representation
- Problem solving process
  - Formulate the goal
  - Formulate the problem
  - Search to find a plan
    - ★ Agent *simulates* sequences of actions
  - Execute the plan

Problem and Solution

- Problem definition
  - *State space*: set of possible states

Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state

Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds

Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds
  - *Actions*: set of available actions

Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds
  - *Actions*: set of available actions
  - *Transition function*: state resulting from executing an action in a given state. $f : S \times A \rightarrow S$

Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds
  - *Actions*: set of available actions
  - *Transition function*: state resulting from executing an action in a given state. $f : S \times A \rightarrow S$
  - *Cost function*: cost of executing an action

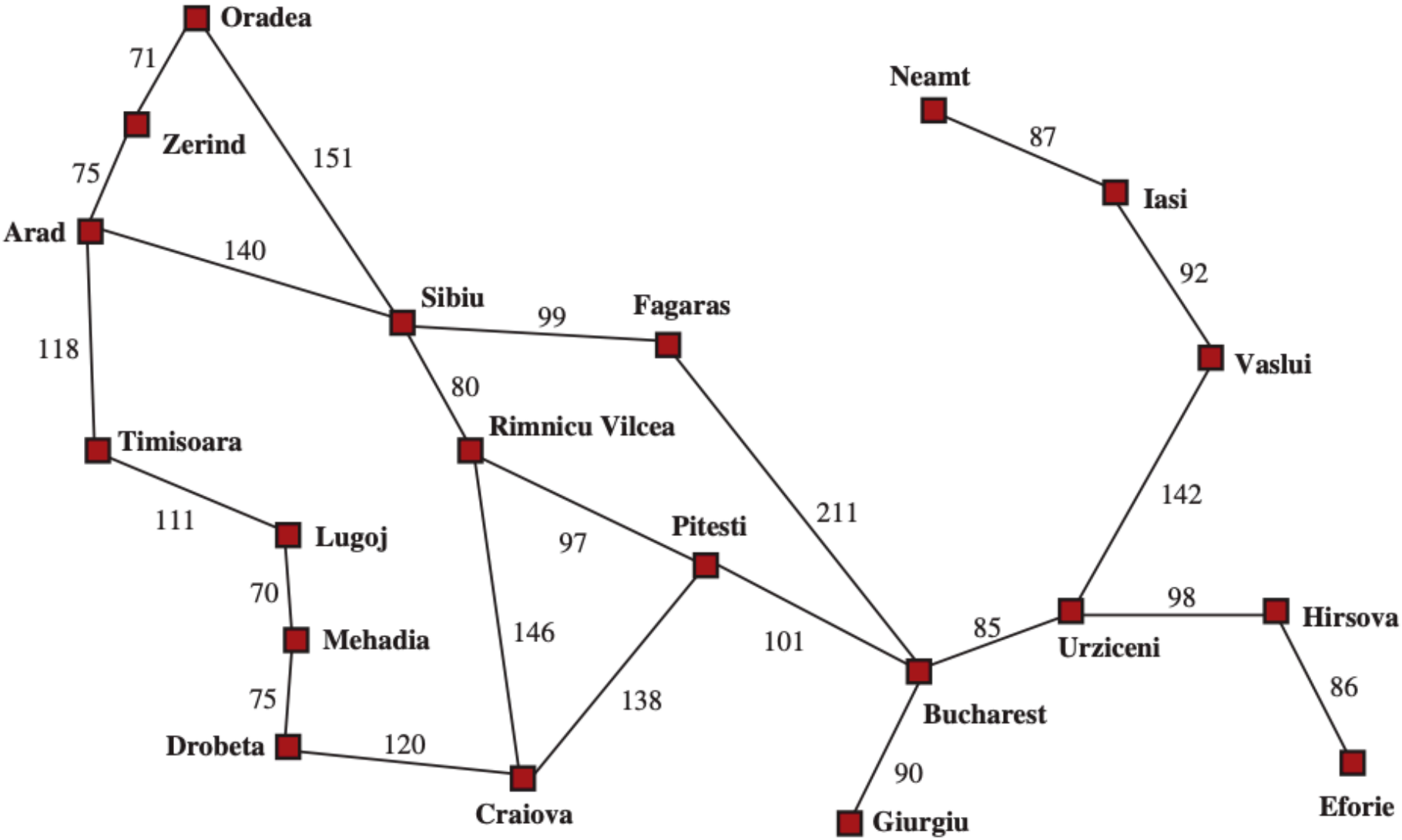
Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds
  - *Actions*: set of available actions
  - *Transition function*: state resulting from executing an action in a given state. $f : S \times A \rightarrow S$
  - *Cost function*: cost of executing an action
- *Solution* is a *path* from the start state to a goal state

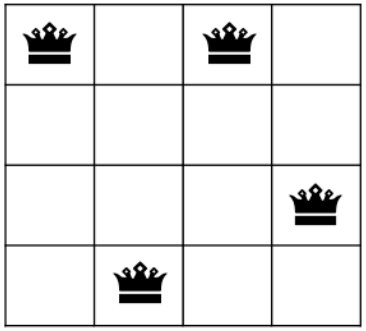
Problem and Solution

- Problem definition
  - *State space*: set of possible states
  - *Initial state*: start state
  - *Goal state(s)*: a state where some property holds
  - *Actions*: set of available actions
  - *Transition function*: state resulting from executing an action in a given state. $f : S \times A \rightarrow S$
  - *Cost function*: cost of executing an action
- *Solution* is a *path* from the start state to a goal state
- *Optimal* solution is the lowest cost solution

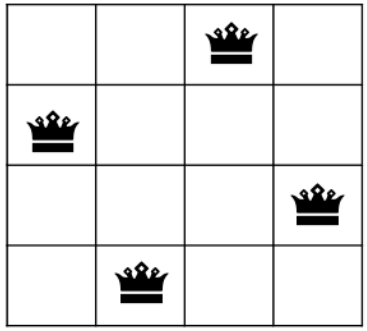
Example Problems



Example Problems

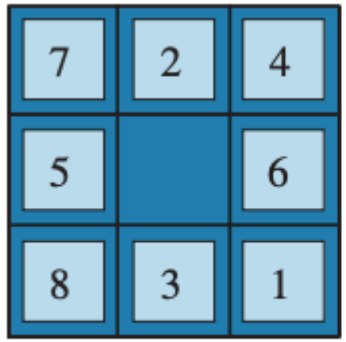


Start State



Goal State

4-Queen



Start State

Goal State

8-Puzzle

Task Environment

- Fully vs. Partially Observable
- Deterministic vs. Stochastic
- Episodic vs. Sequential
- Static vs. Dynamic
- Discrete vs. Continuous
- Known vs. Unknown
- Single vs. Multi-agent

Search Problem

- State Space
- Initial State
- Goal State
- Actions
- Transition Model
- Action Cost Function