CS 480

Introduction to Artificial Intelligence

September 1, 2022

Announcements / Reminders

- **Please follow the Week 02 To Do List instructions**

- **My office hours:**
  - **Tuesday/Thursday 12:50 PM - 01:50 PM CST Stuart Building 217E or by appointment (online or in person)**

Plan for Today

- Intelligent Agents
- Problem Solving: Searching

Task Environment | PEAS

In order to start the agent design process we need to specify / define:

- The Performance measure
- The Environment in which the agent will operate
- The Actuators that the agent will use to affect the environment
- The Sensors that the agent will use to perceive the environment

Task Environment Properties

Key dimensions by which task environments can be categorized:

- **Fully vs partially observable (can be unobservable too)**
- **Single agent vs multiagent**
  - **multiagent: competitive vs. cooperative**
- **Deterministic vs. nonderministic (stochastic)**
- **Episodic vs. sequential**
- **Static vs. dynamic**
- **Discrete vs. continuous**
- **Known vs. unknown (to the agent)**

Fully Observable Environment

Source: Pixabay (www.pexels.com)

Partially Observable Environment

Source: https://en.wikipedia.org/wiki/Fog_of_war

Partially Observable Environment

Source: https://en.wikipedia.org/wiki/Fog_of_war

Single-agent System

Source: cottonbro (www.pexels.com)

Multiagent System

Source: Vlada Karpovich (www.pexels.com)

Multiagent System

Source: Vlada Karpovich (www.pexels.com)

Deterministic vs. Nondeterministic

- Deterministic environment:
  - next state is completely determined by the current state and agent action
  - deterministic AND fully observable environment: no need to worry about uncertainty
  - deterministic AND partially observable ***may*** appear nondeterministic
- Nondeterministic (stochastic) environment:
  - next state is NOT completely determined by the current state and agent action

Episodic vs. Sequential

- **Episodic environment:**
  - **agent experience is divided into individual, independent, and atomic episodes**
  - **one percept - one action.**
  - **next action is not a function of previous action: not necessary to memorize it**
- **Sequential environment:**
  - **current decision / action COULD affect all future decisions / actions**
  - **better keep track of it**

Static vs. Dynamic

- **Static environment:**
  - **environment CANNOT change while the agent is taking its time to decide**
- **Dynamic environment:**
  - **environment CAN change while the agent is taking its time to decide -> decision / action may be dated**
  - **speed is important**

Discrete vs. Continuous

- **Discrete environment:**
  - state changes are discrete
  - time changes are discrete
  - percepts are discrete
- **Continuous environment:**
  - state changes are continuous (“fluid”)
  - time changes are continuous
  - percepts / actions can be continuous

Known vs. Unknown (to Agent)

- **Known environment:**
  - agent knows all outcomes to its actions (or their probabilities)
  - agent "knows how the environment works"
- **Unknown environment:**
  - agent "doesn't know all the details about the inner workings of the environment"
  - learning and exploration can be necessary

Task Environment Characteristics

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Hardest Case / Problem

- Partially observable (incomplete information, uncertainty)
- Multiagent (complex interactions)
- Nondeterministic (uncertainty)
- Sequential (planning usually necessary)
- Dynamic (changing environment, uncertainty)
- Continuous (infinite number of states)
- Unknown (agent needs to learn / explore, uncertainty)

Designing the Agent for the Task

Analyze the Problem / Task (PEAS)	Select Agent Architecture	Select Internal Representations	Apply Corresponding Algorithms

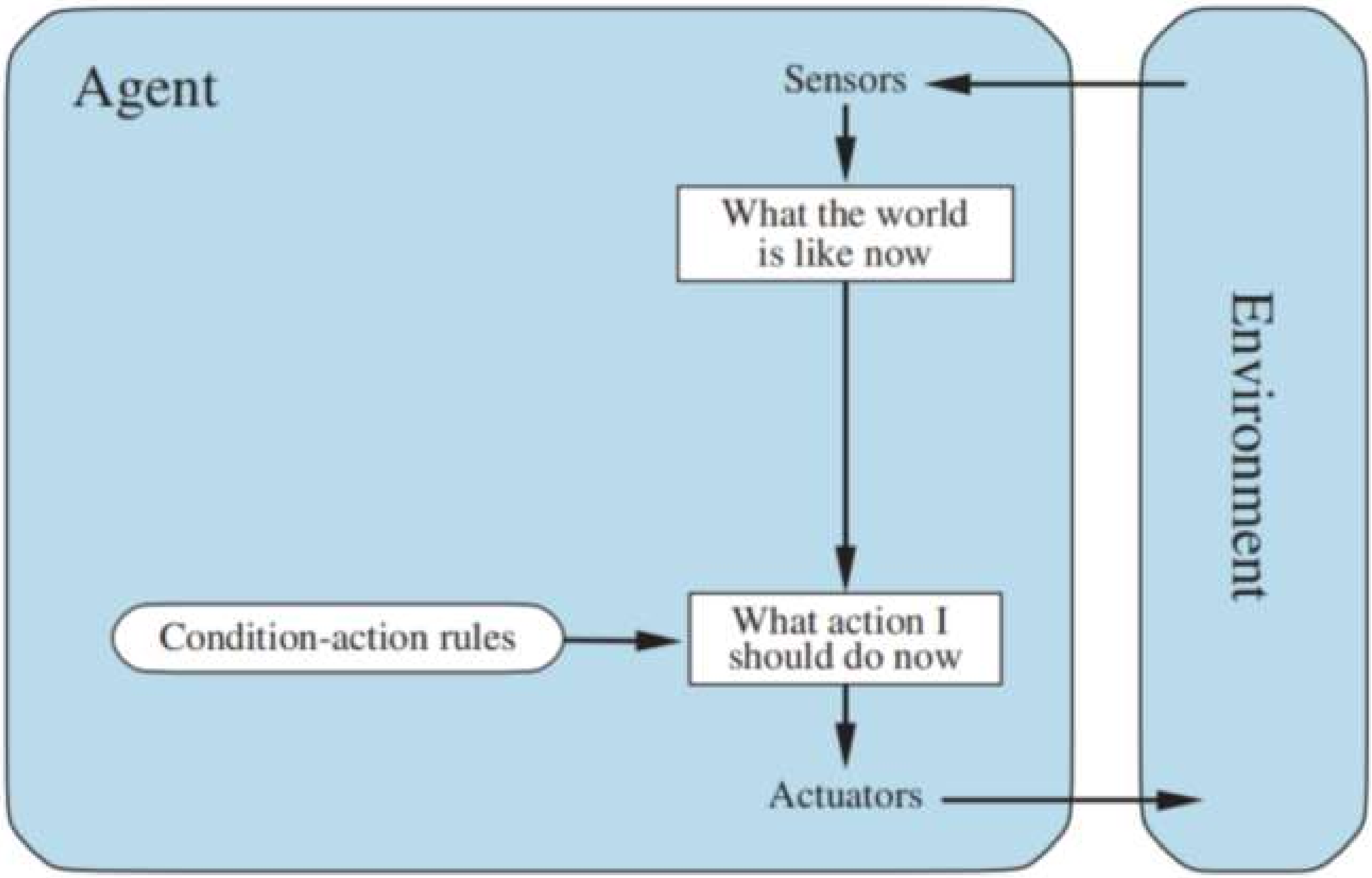
Agent Structure / Architecture

Agent = Architecture + Program

Typical Agent Architectures

- **Simple reflex agent**
- **Model-based reflex agent:**
- **Goal-based reflex agent**
- **Utility-based reflex agent**

Simple Reflex Agent

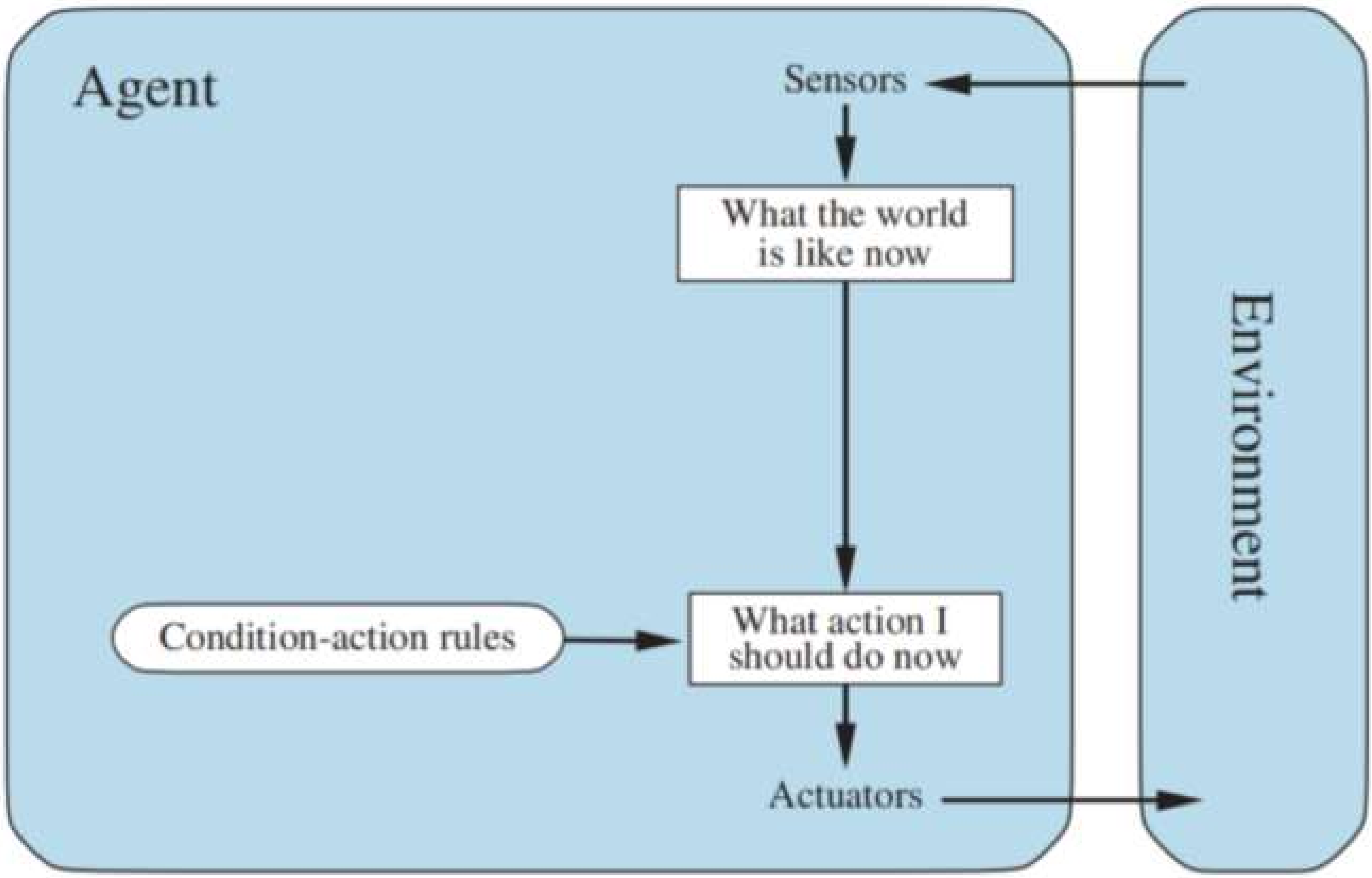


Simple Reflex Agent

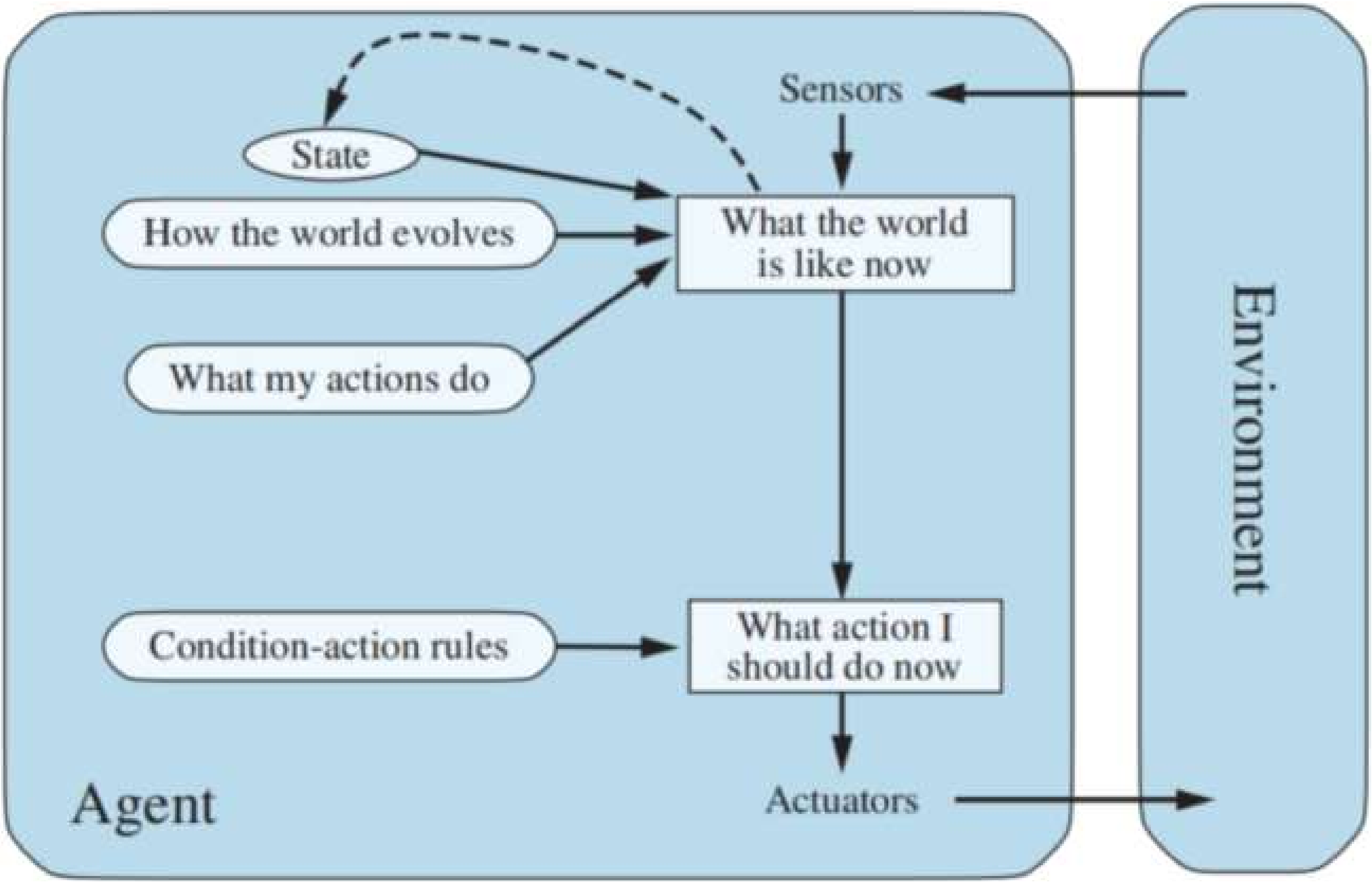
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

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

Simple Reflex Agent: Challenges?



Model-based Reflex Agent

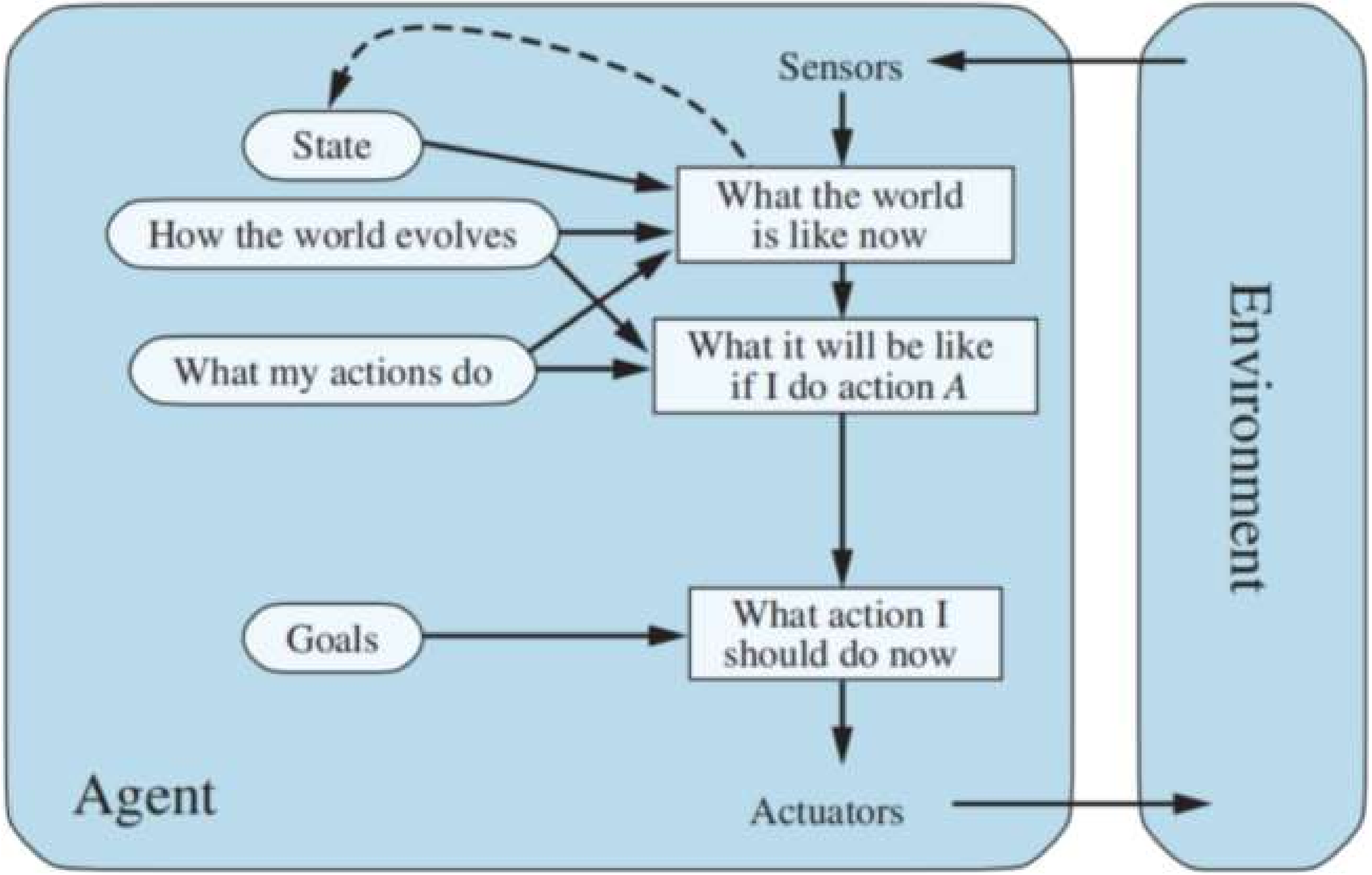


Model-based Reflex Agent

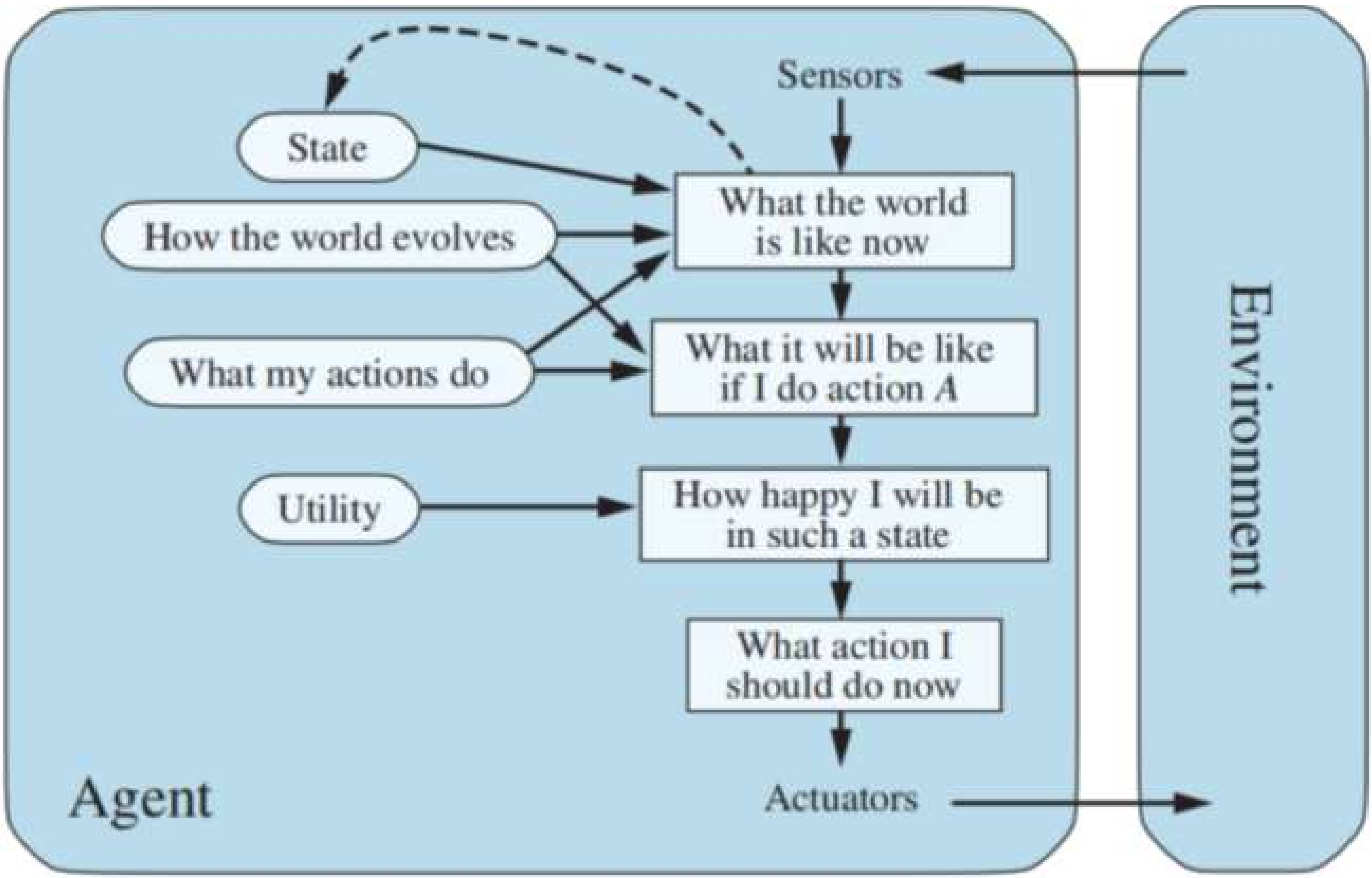
function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on
      the current state and action
    sensor_model, a description of how the current world 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*

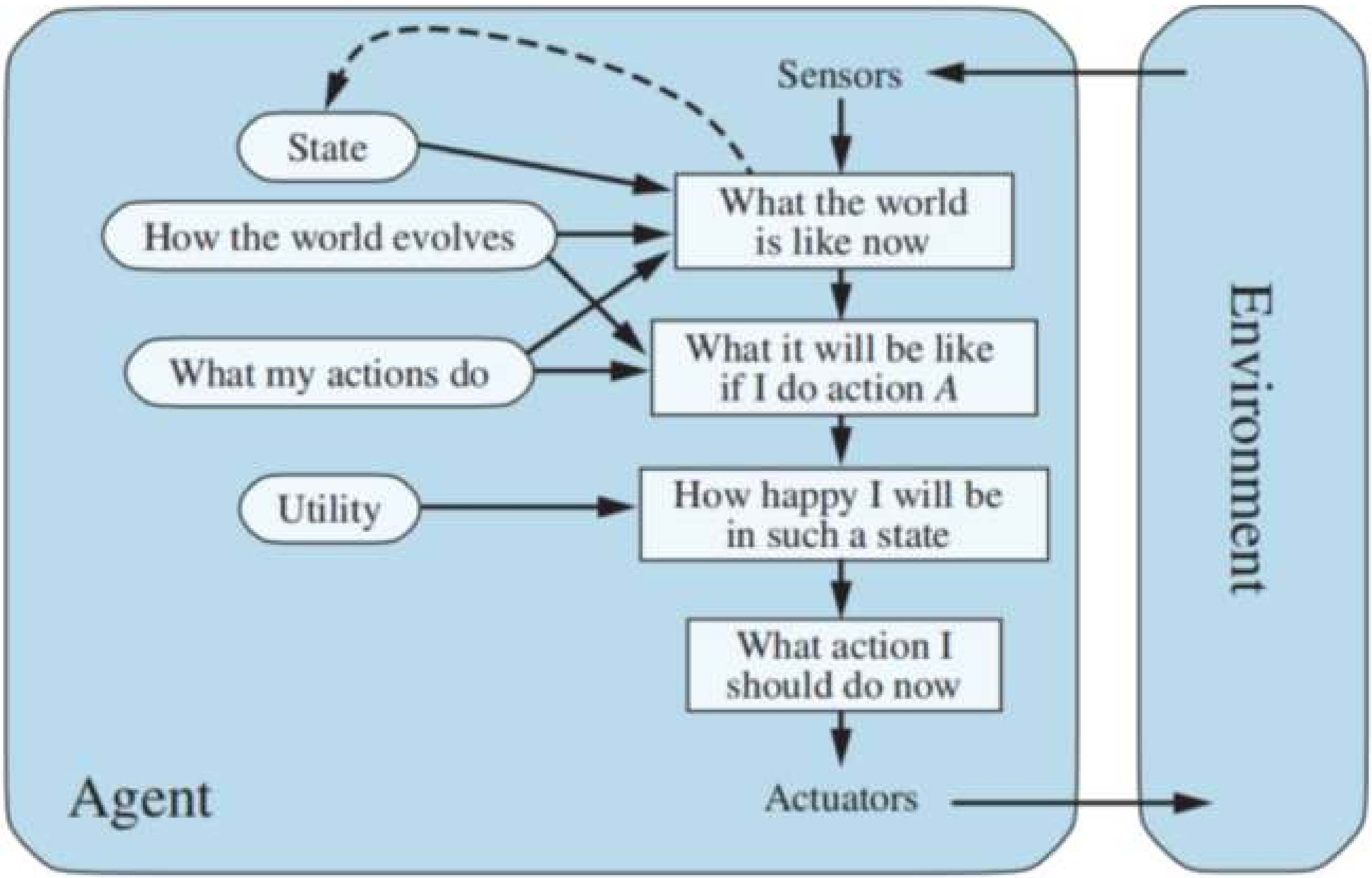
Model-based Goal-based Agent



Model-based Utility-based Agent



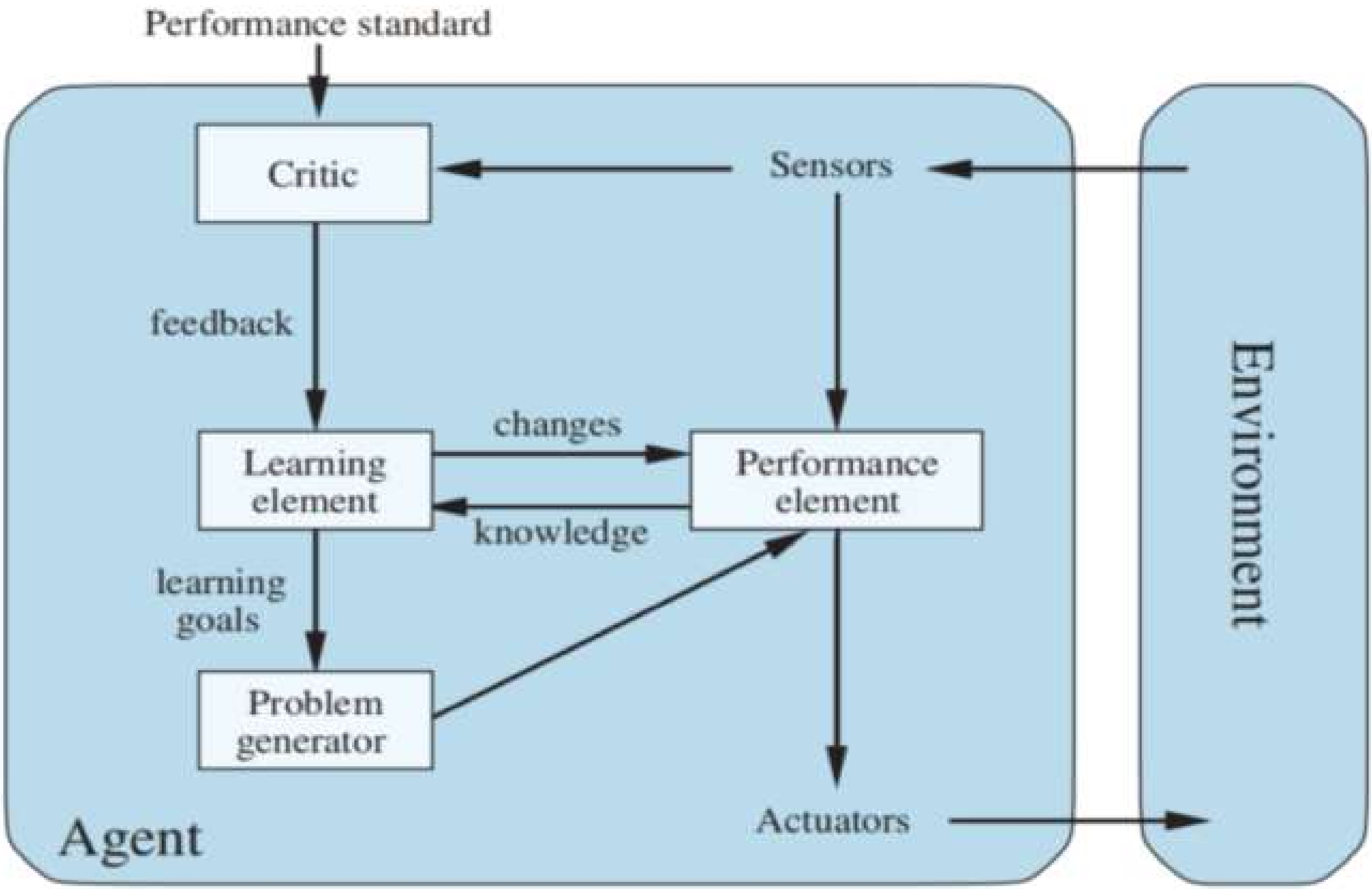
Model-based Agents: Challenges?



Typical Agent Architectures

- Simple reflex agent: uses condition-action rules
- Model-based reflex agent: keeps track of the unobserved parts of the environment by maintaing internal state:
  - "how the world works": state transition model
  - how percepts and environment is related: sensor model
- Goal-based reflex agent: maintains the model of the world and goals to select decisions (that lead to goal)
- Utility-based reflex agent: maintains the model of the world and utility function to select PREFERRED decisions (that lead to the best expected utility: avg (EU * p))

Learning Agent



Designing the Agent for the Task

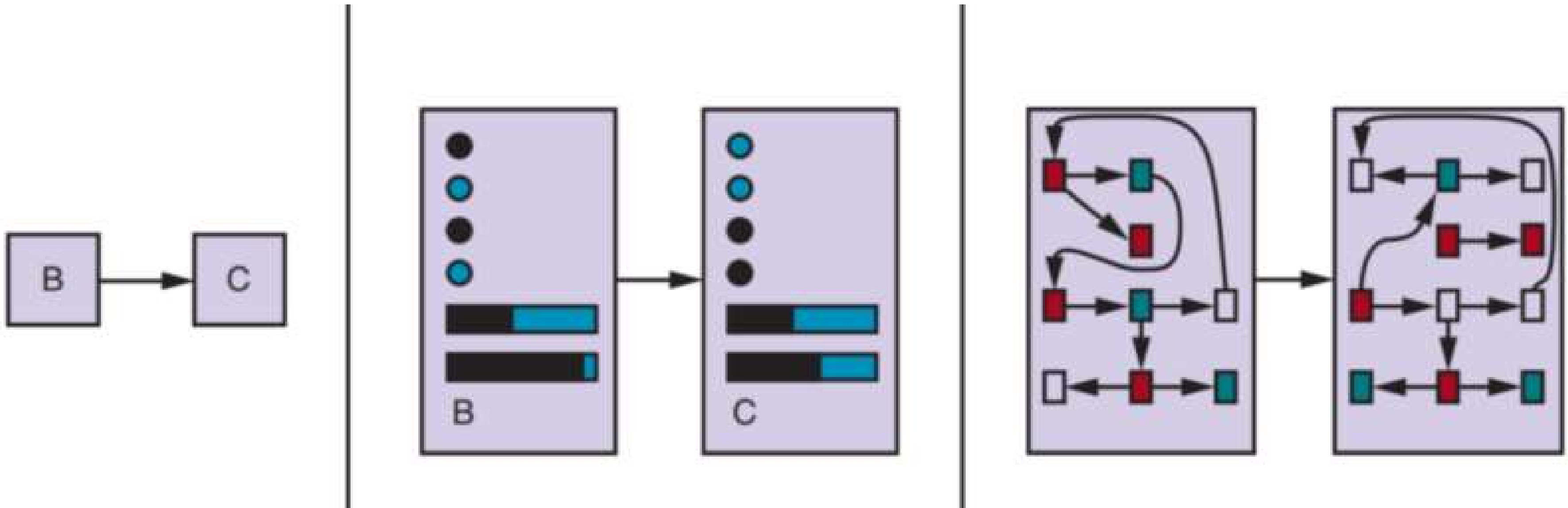
Analyze the Problem / Task (PEAS)

Select Agent Architecture

Select Internal Representations

Apply Corresponding Algorithms

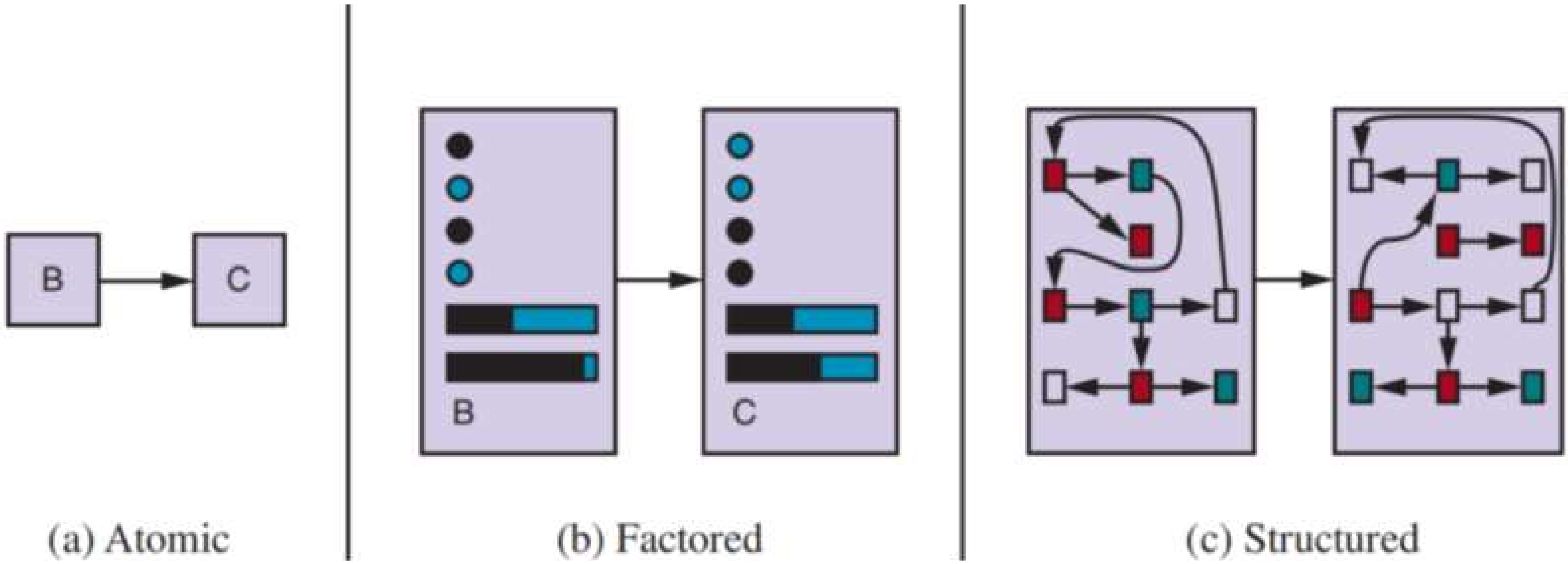
State and Transition Representations



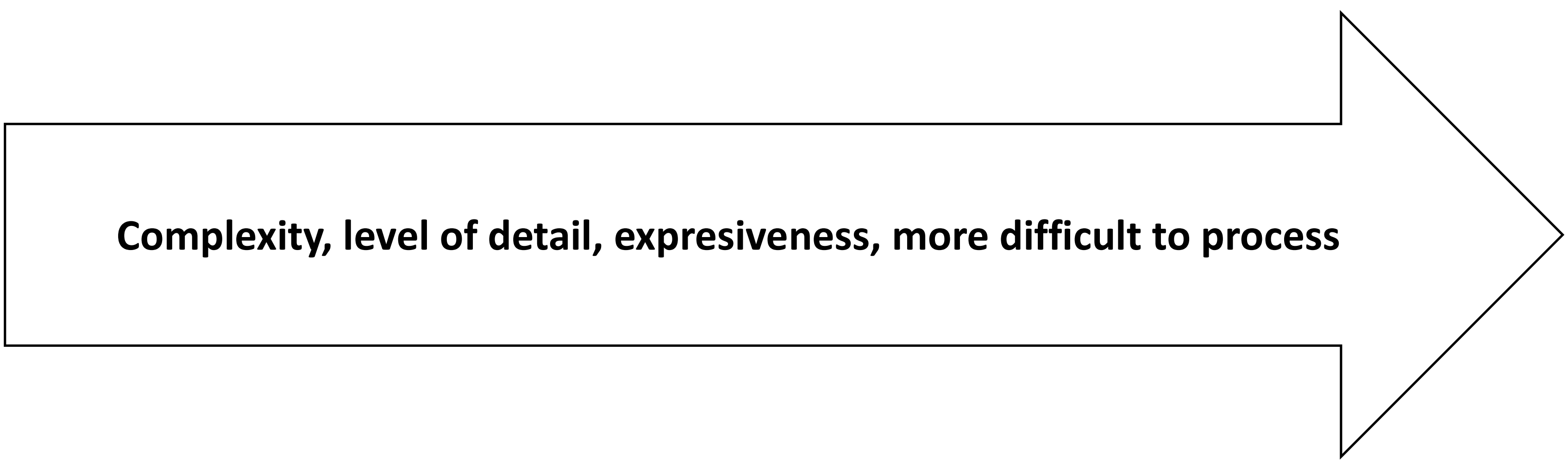
(a) Atomic (b) Factored (c) Structured

- **Atomic: state representation has NO internal structure**
- **Factored: state representation includes fixed attributes (which can have values)**
- **Structured: state representation includes objects and their relationships**

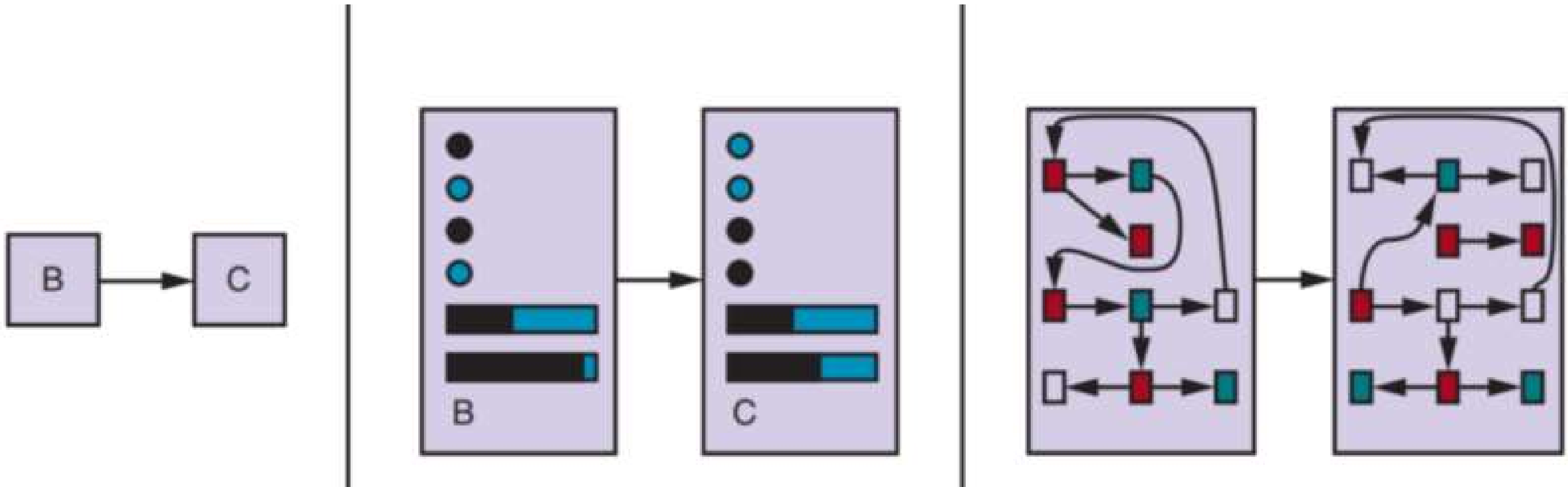
State and Transition Representations



(a) Atomic (b) Factored (c) Structured



Representations and Algorithms



(a) Atomic (b) Factored (c) Structured

- Searching
- Hidden Markov models
- Markov decision process
- Finite state machines

- Constraint satisfaction algorithms
- Propositional logic
- Planning
- Bayesian algorithms
- Some machine learning algorithms

- Relational database algorithms
- First-order logic
- First-order probability models
- Natural language understanding (some)

Designing the Agent for the Task

- Analyze the Problem / Task (PEAS)
- Select Agent Architecture
- Select Internal Representations
- Apply Corresponding Algorithms

Finite State Machine: A Turnstile

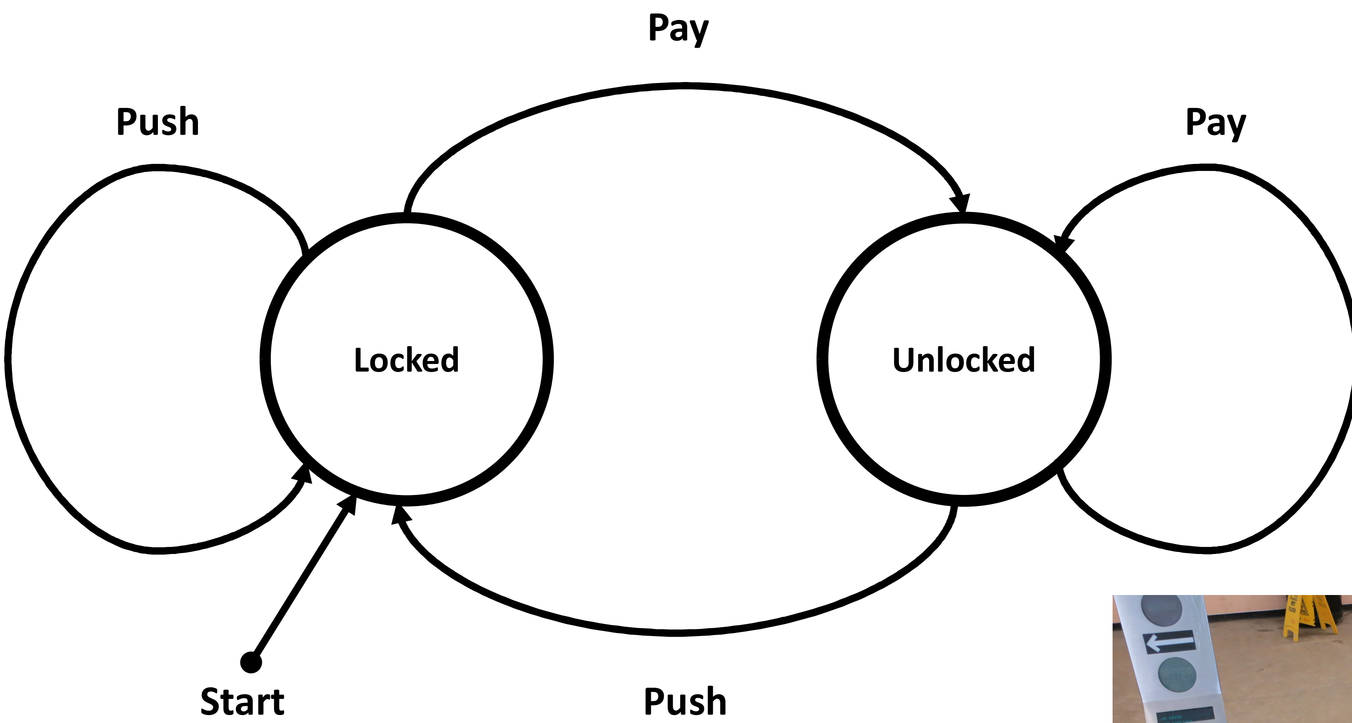
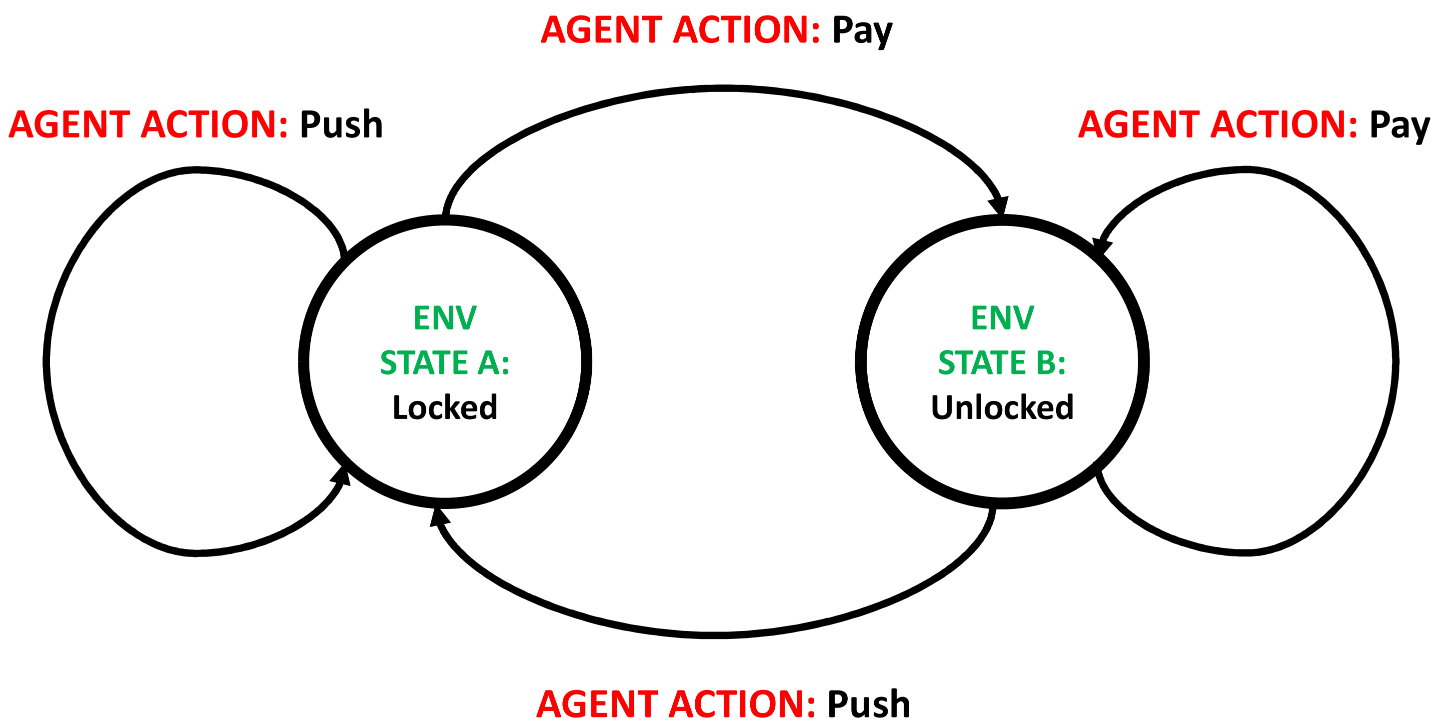
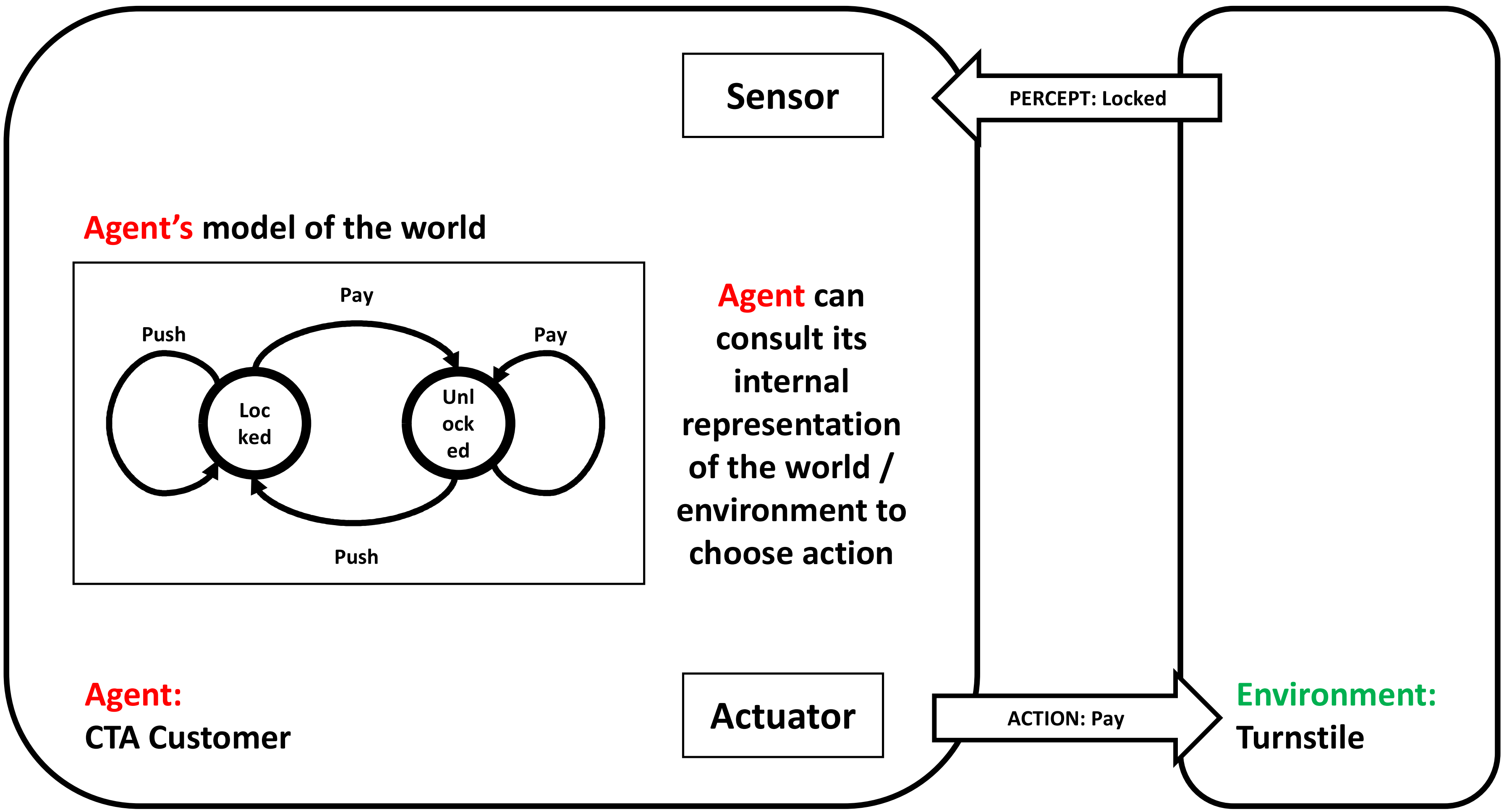


Image source: Wikipedia

Finite State Machine: A Turnstile

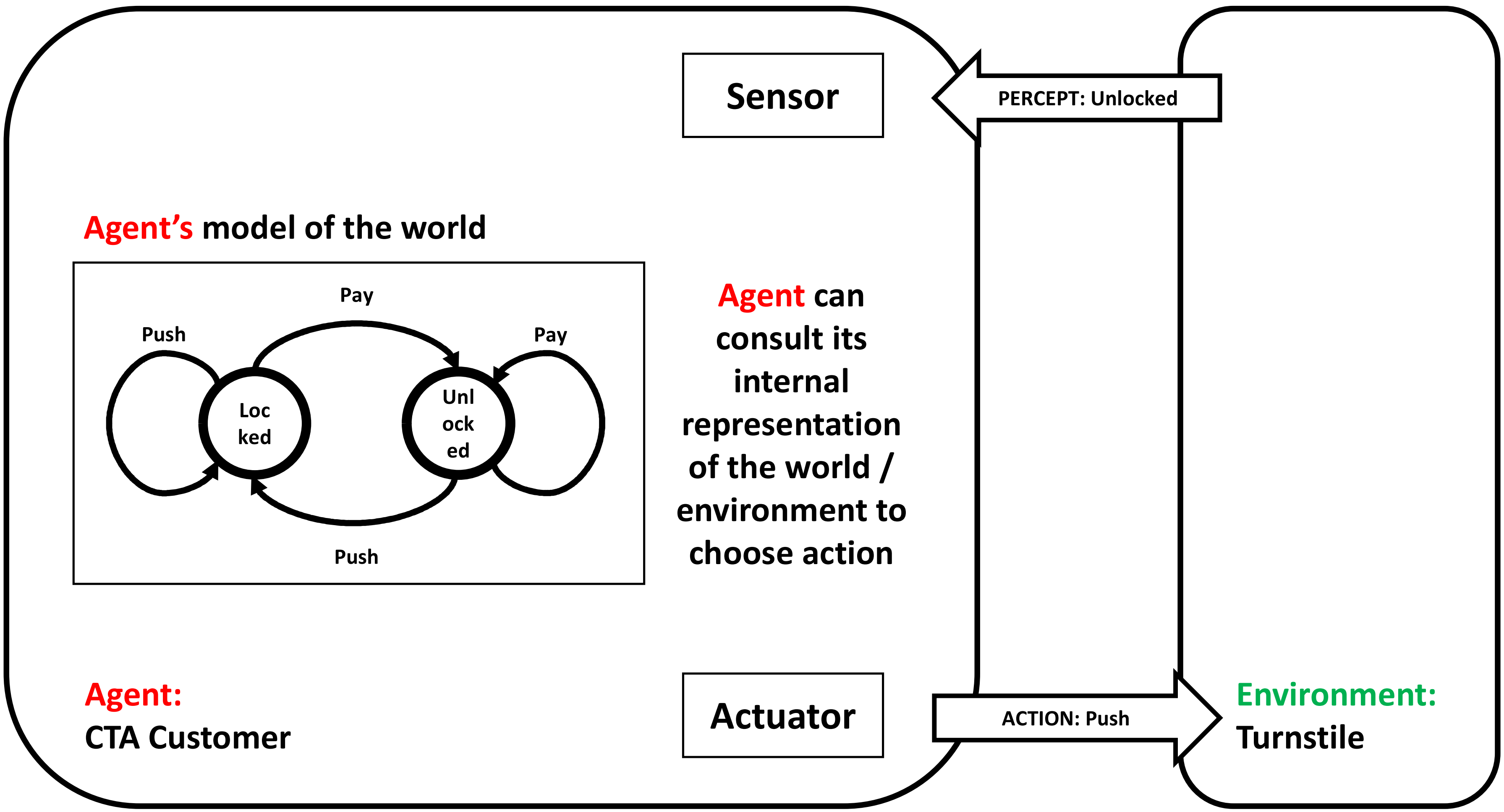


Model-based Reflex Agent Example



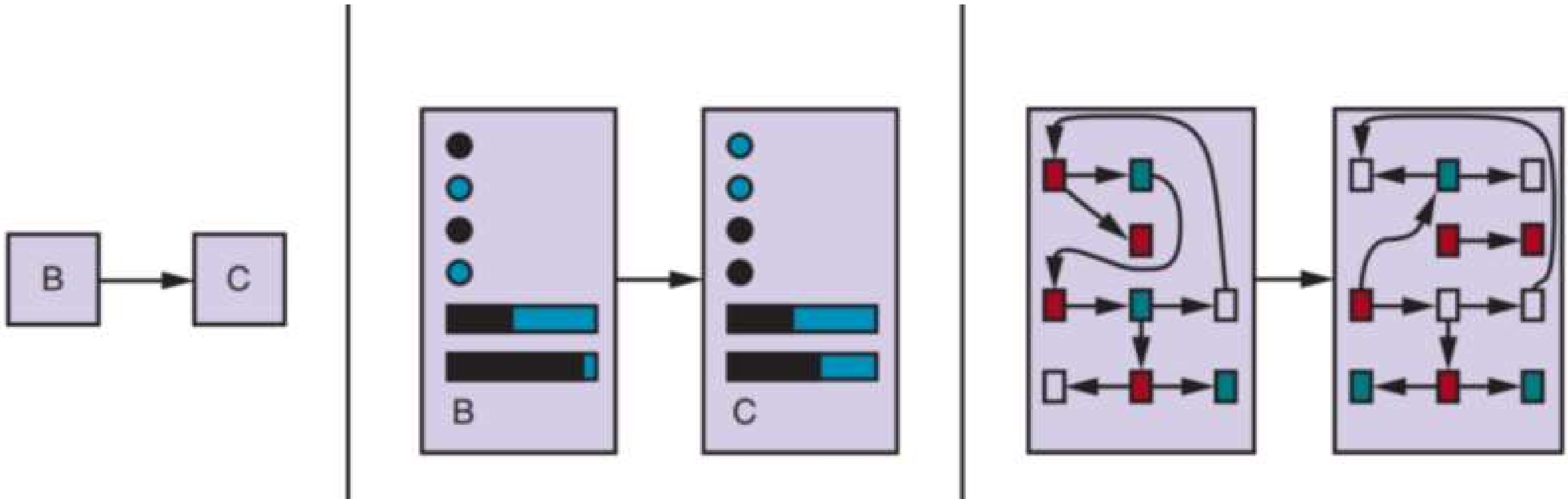
Note: This problem could be easily solved with a simple (without internal model) reflex agent.

Model-based Reflex Agent Example

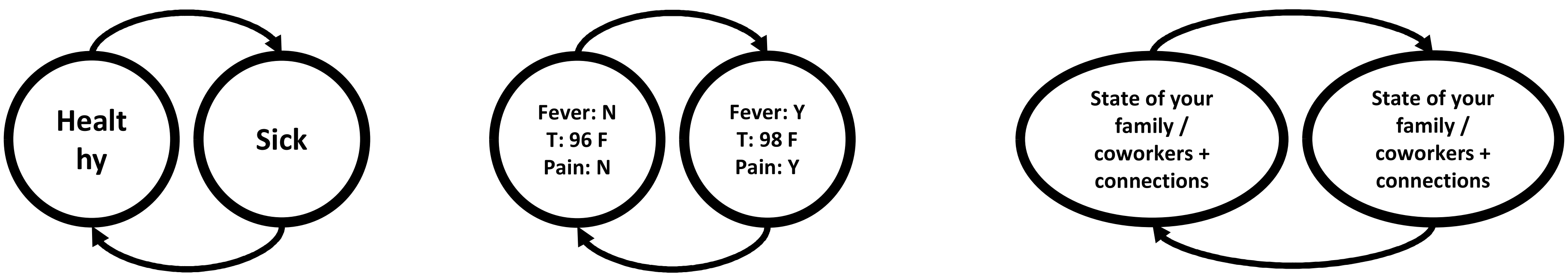


Note: This problem could be easily solved with a simple (without internal model) reflex agent.

Representations: Examples



(a) Atomic (b) Factored (c) Structured



Designing the Agent for the Task

Analyze the Problem / Task (PEAS)	Select Agent Architecture	Select Internal Representations	Apply Corresponding Algorithms

BTW: How Would you Program it All?

Problem-Solving / Planning Agent

- **Context / Problem:**
  - **correct action is NOT immediately obvious**
  - **a plan (a sequence of actions leading to a goal) may be necessary**
- **Solution / Agent:**
  - **come up with a computational process that will <u>search</u> for that plan**
- **Planning Agent:**
  - **uses factored or structured representations of states**
  - **uses searching algorithms**

Planning: Environment Assumptions

Works with a “Simple Environment”:

- Fully observable
- Single agent (for now -> it can be multiagent)
- Deterministic
- Static
- Episodic
- Discrete
- Known to the agent

Problem-Solving Process

- Goal formulation:
  - adopt a goal (think: desirable state)
  - a concrete goal should help you reduce the amount of searching
- Problem formulation:
  - an abstract representation of states and actions
- Search:
  - search for solutions within the abstract world model
- Execute actions in the solution

Planning: Environment Assumptions

Works with a "Simple Environment":

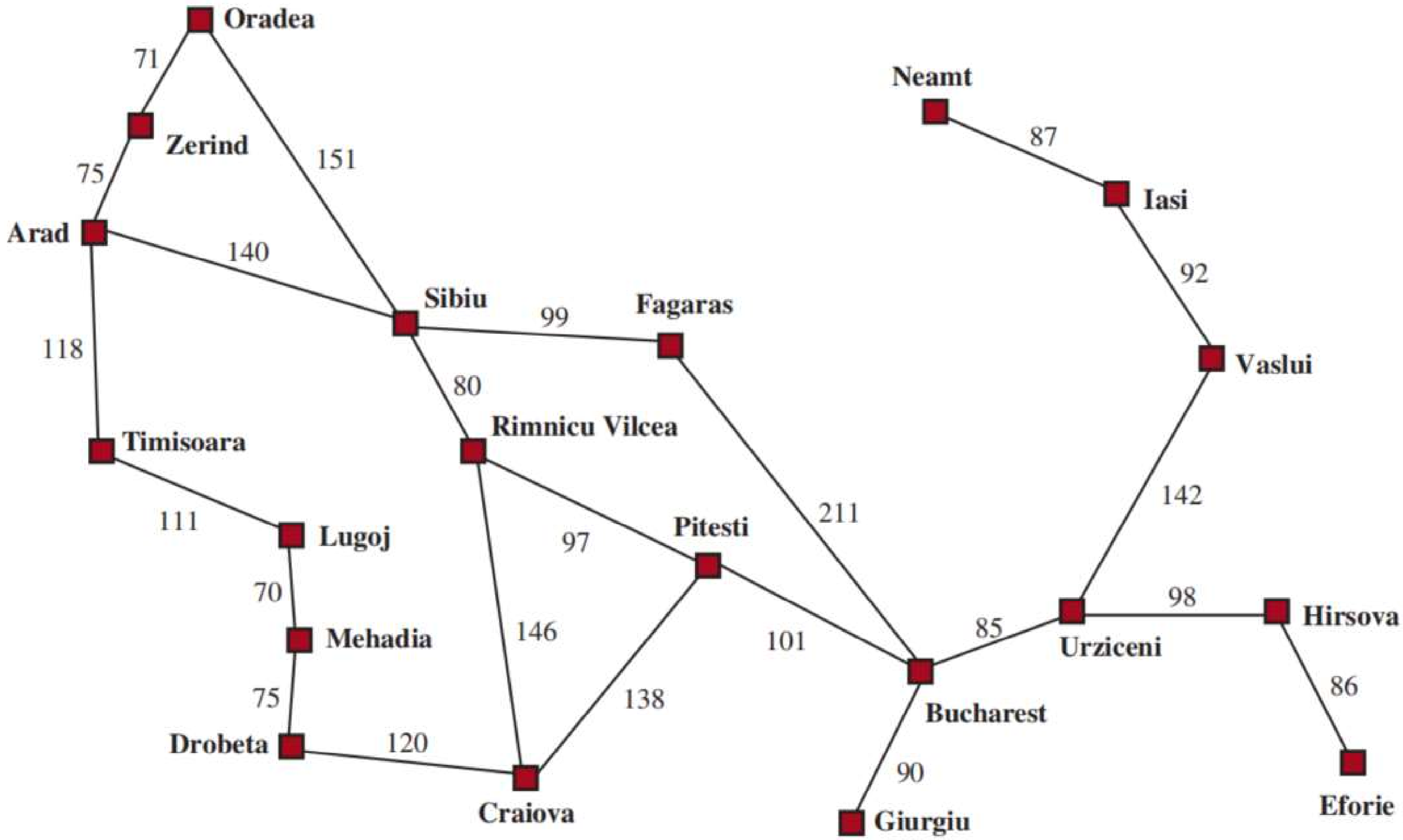
- **Fully observable**
- **Single agent (for now -> it can be multiagent)**
- **Deterministic**
- **Static**
- **Episodic**
- **Discrete**
- **Known to the agent**

Important and helpful:
Such assumptions **GUARANTEE** a **FIXED** sequence of actions as a solution
What does it mean?
You can execute the "plan" without worrying about incoming percepts (open-loop control)

Defining Search Problem

- **Define a set of possible states: State Space**
- **Specify Initial State**
- **Specify Goal State(s) (there can be multiple)**
- **Define a FINITE set of possible Actions for EACH state in the State Space**
- **Come up with a Transition Model which describes what each action does**
- **Specify the Action Cost Function: a function that gives the cost of applying action *a* in state *s***

Sample Problem: Dracula’s Roadtrip

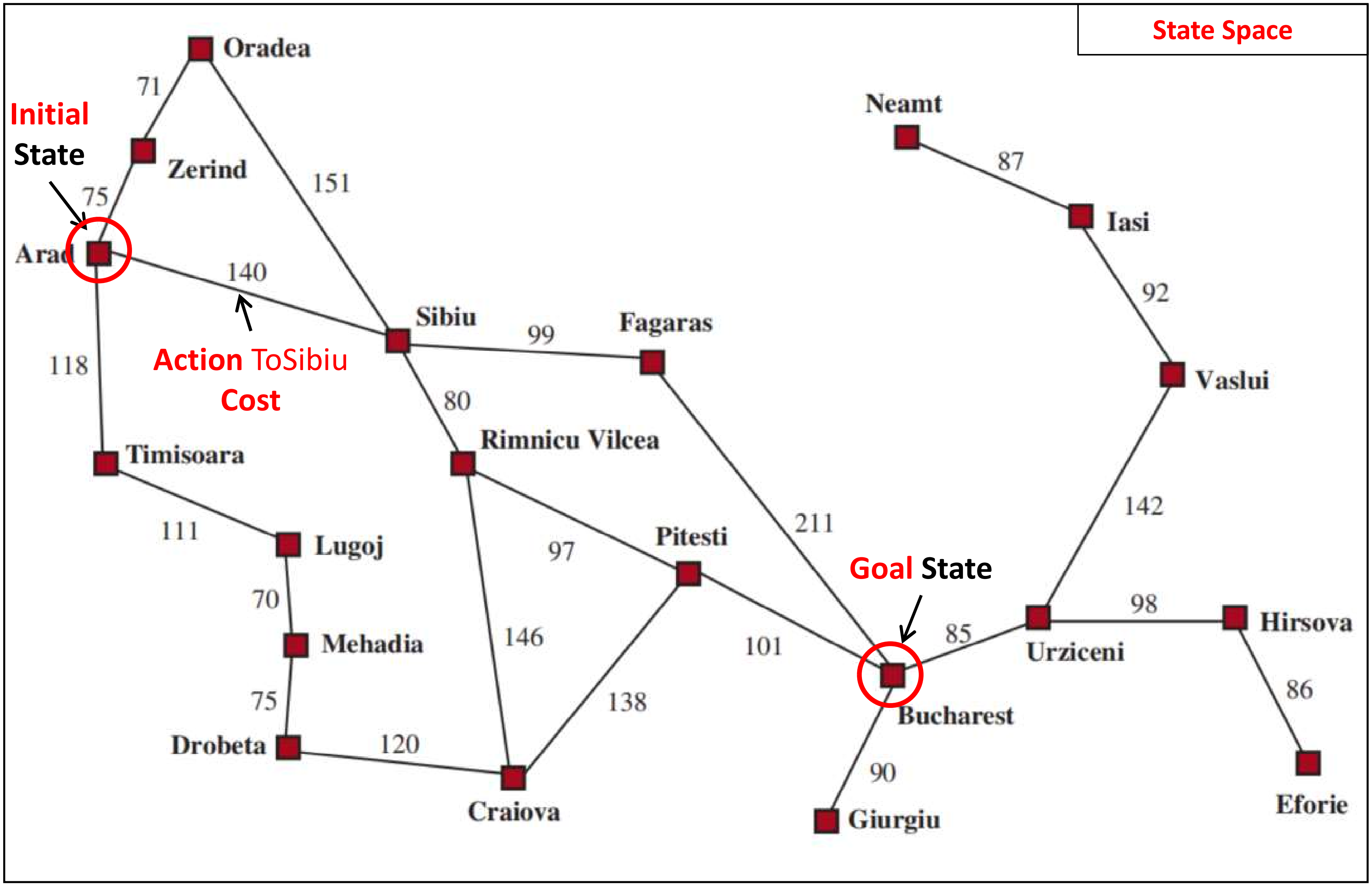


Problem: Get from Arad to Bucharest efficiently (for example: quickly or cheaply).

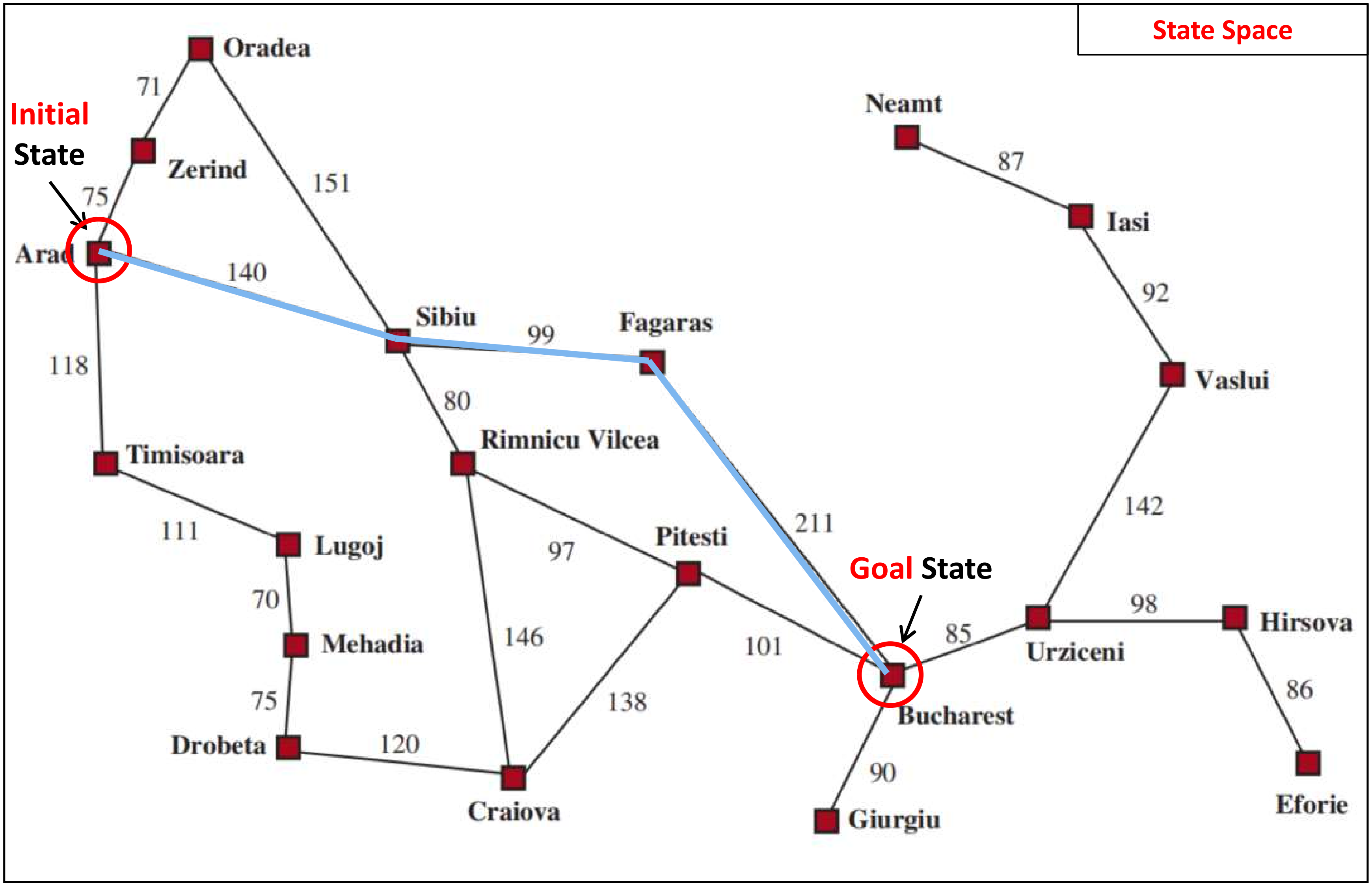
Search Problem: Dracula's Roadtrip

- State Space: a map of Romania
- Initial State: Arad
- Goal State: Bucharest
- Actions:
  - for example: ACTIONS(Arad) = {ToSibiu,ToTimisoara,ToZerind}
- Transition Model:
  - for example: RESULT(Arad, ToZerind) = Zerind
- Action Cost Function [ActionCost($S_{current}$, a, S_{next})]
  - for example: ActionCost(Arad, ToSibiu, Sibiu) = 140

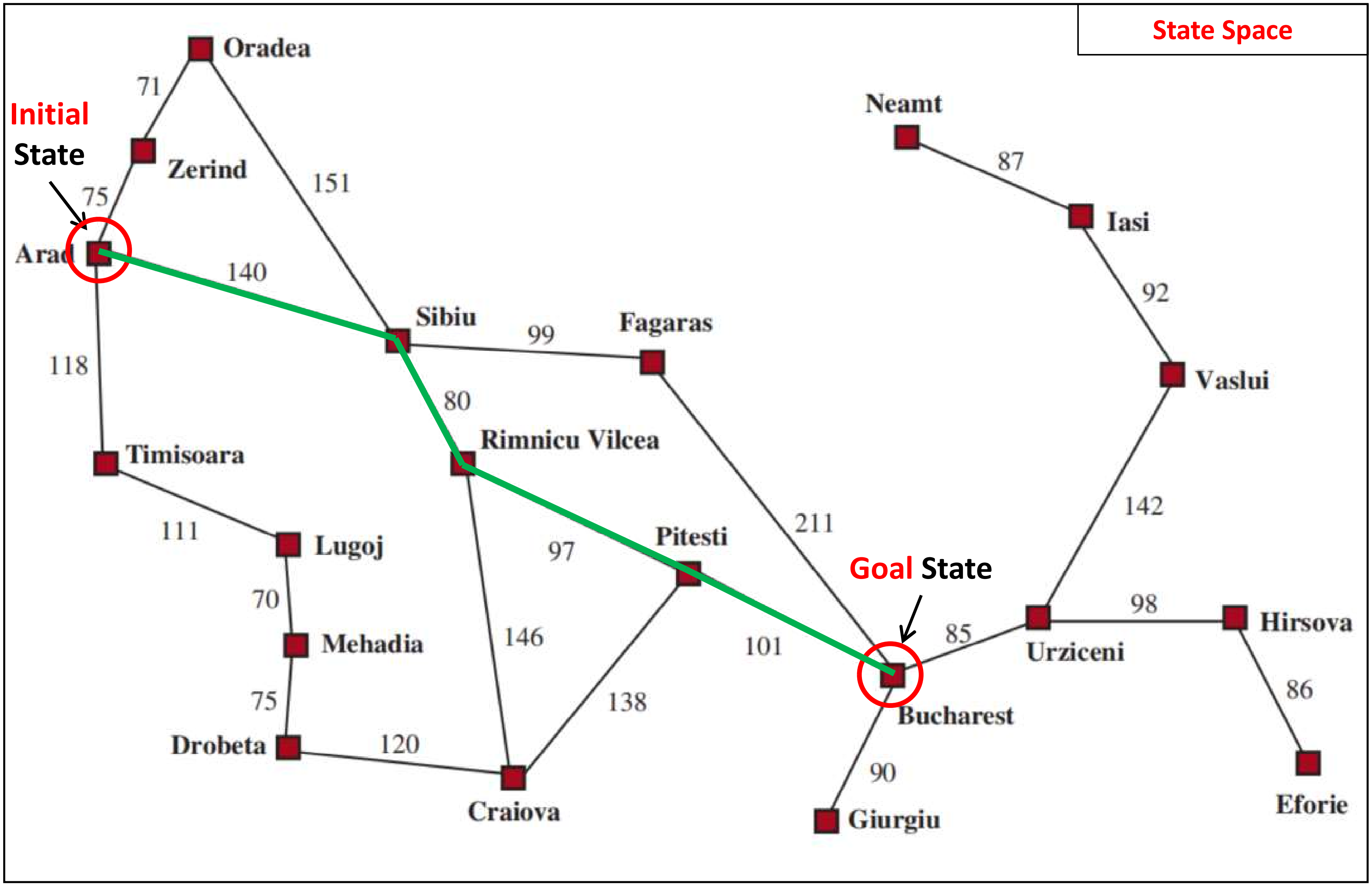
Sample Problem: Dracula's Roadtrip



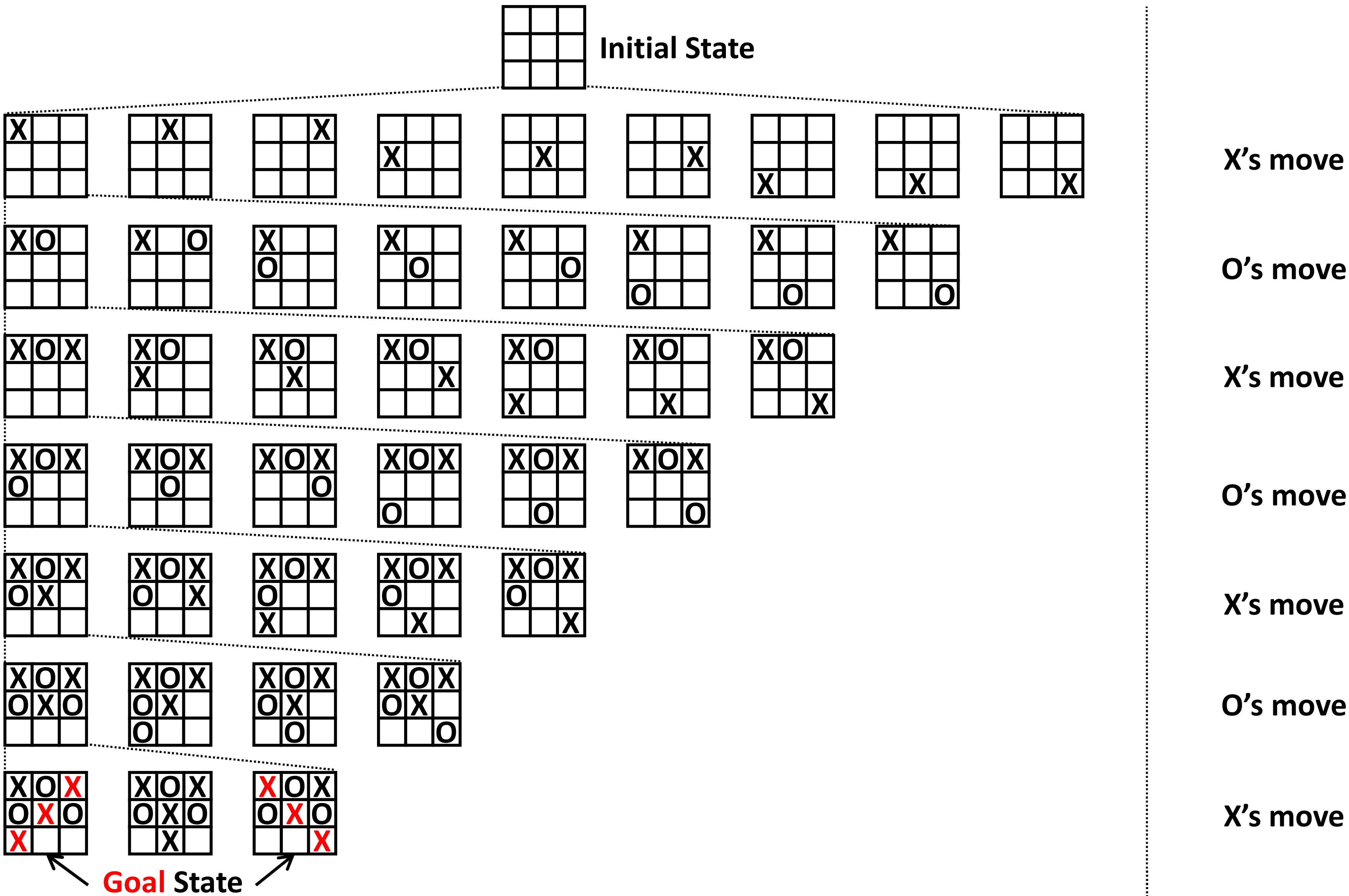
Dracula's Roadtrip: Potential Solution



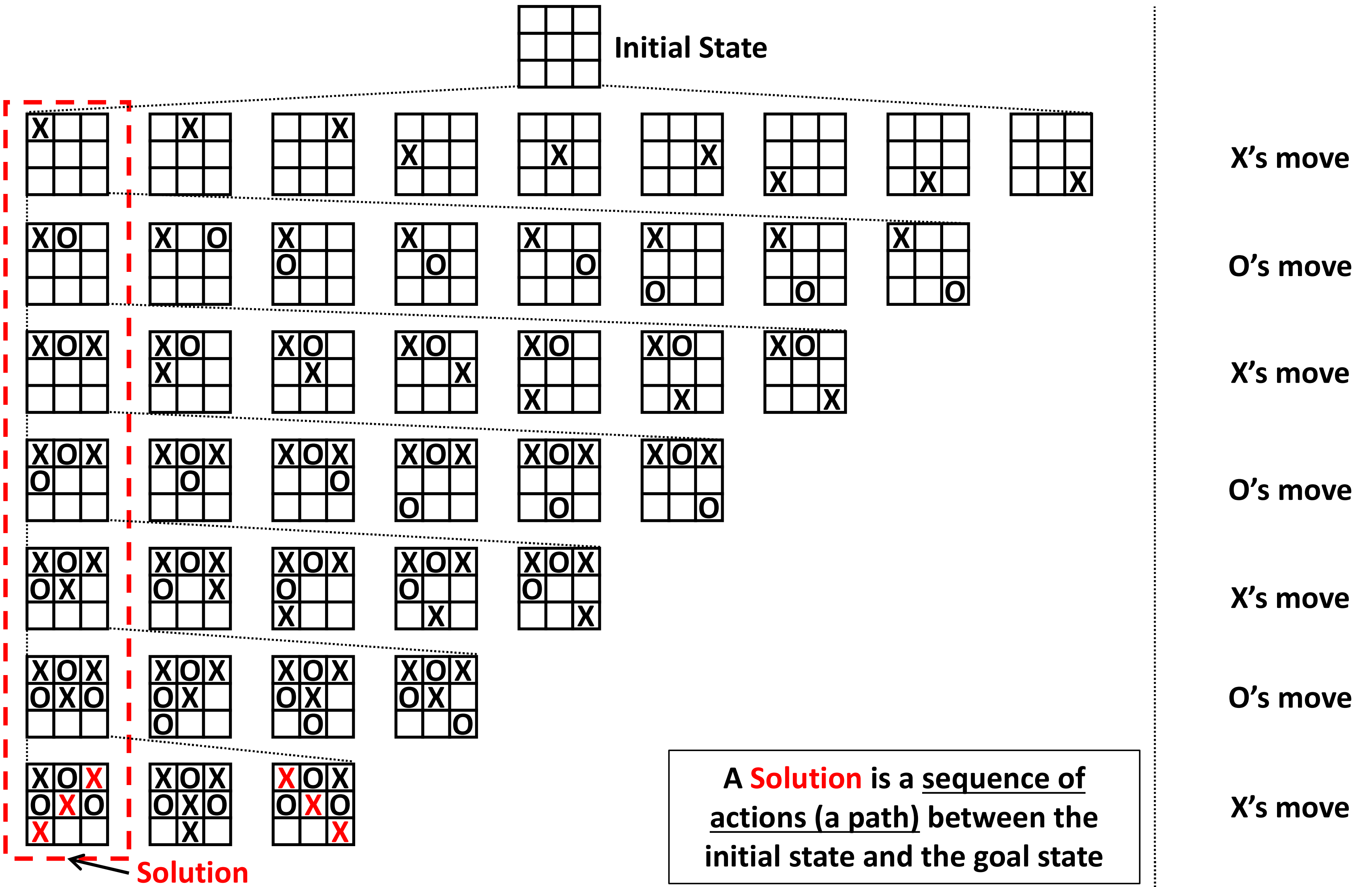
Dracula’s Roadtrip: Potential Solution



Tic Tac Toe: (Partial) State Space



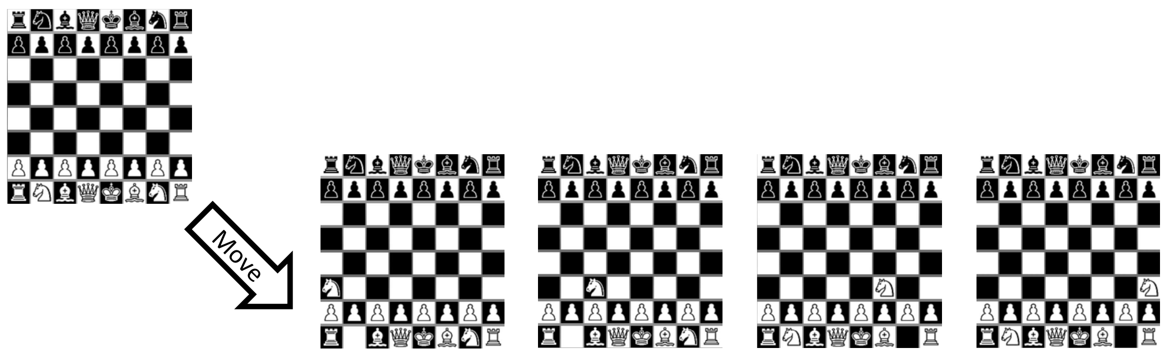
Tic Tac Toe: Solution



A **Solution** is a <u>sequence of actions (a path)</u> between the initial state and the goal state

Chess: (First Move) State Space

Initial State



20 Possible legal first moves:
16 pawn moves
4 knight moves