CS 480

Introduction to Artificial Intelligence

September 2nd, 2021

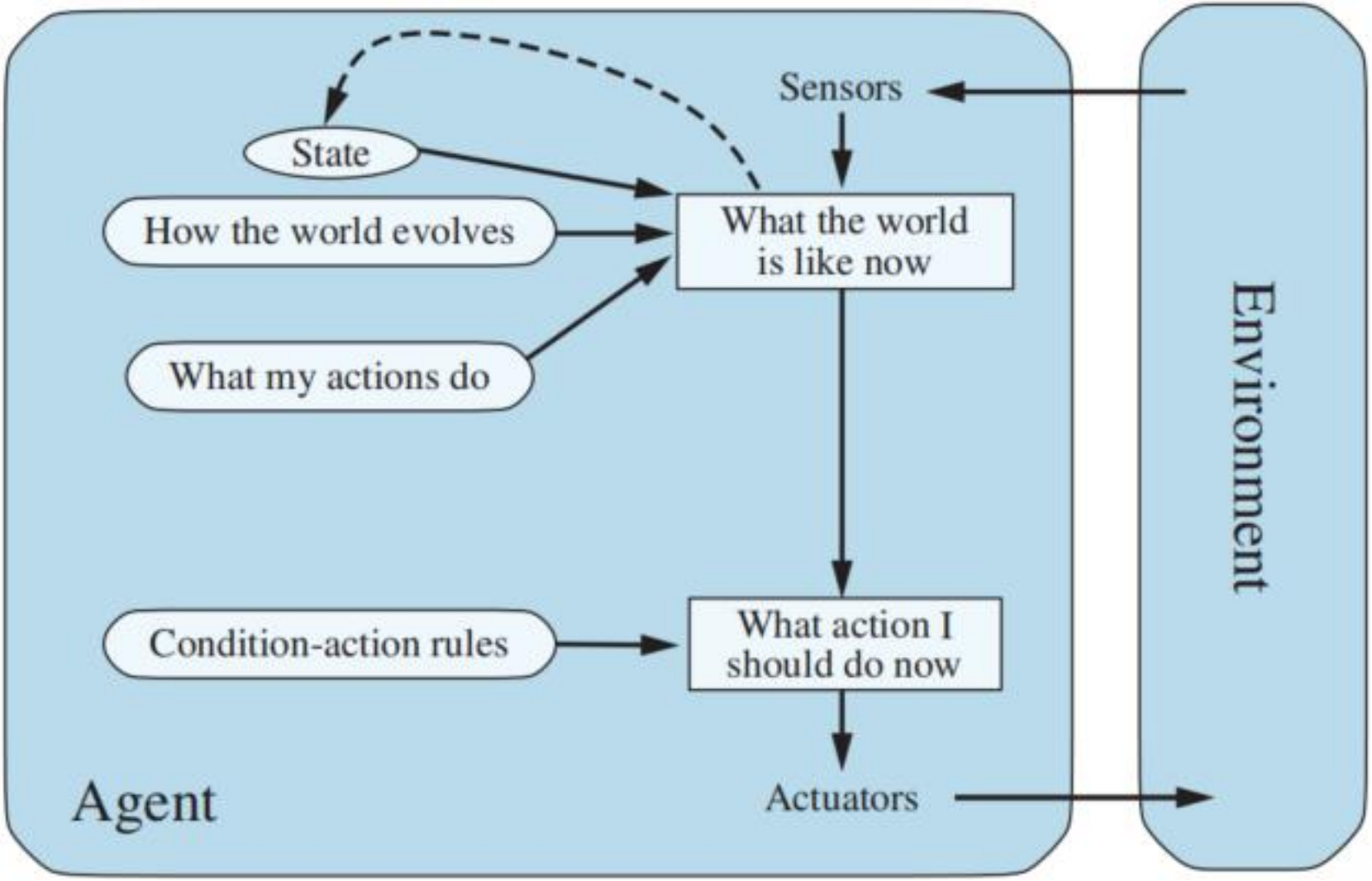
Announcements / Reminders

- Contribute to the discussion on Blackboard, please
- Please follow the Week 02 To Do List instructions

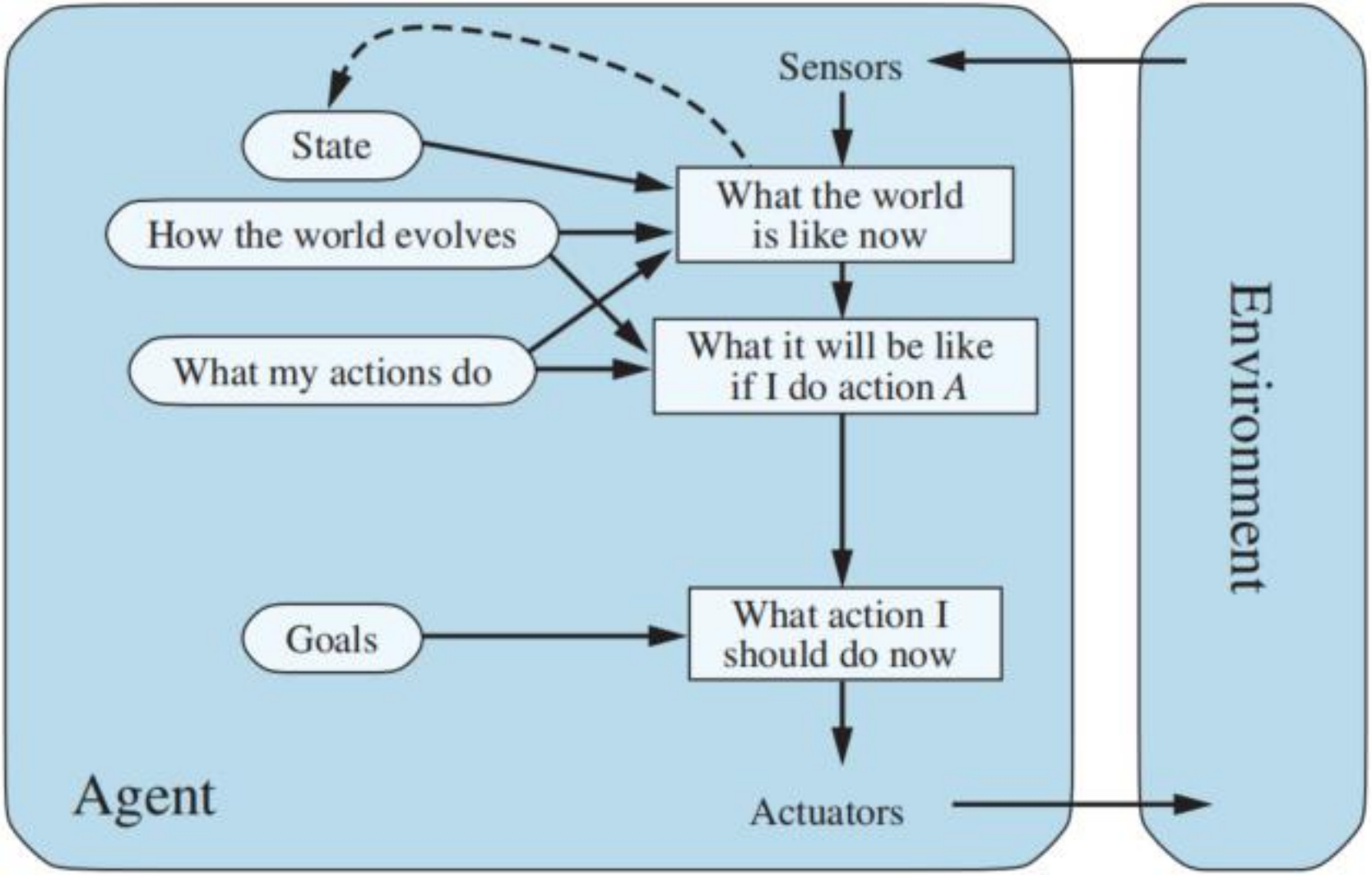
Plan for Today

- **Intelligent Agents**
- **Problem Solving: Searching**

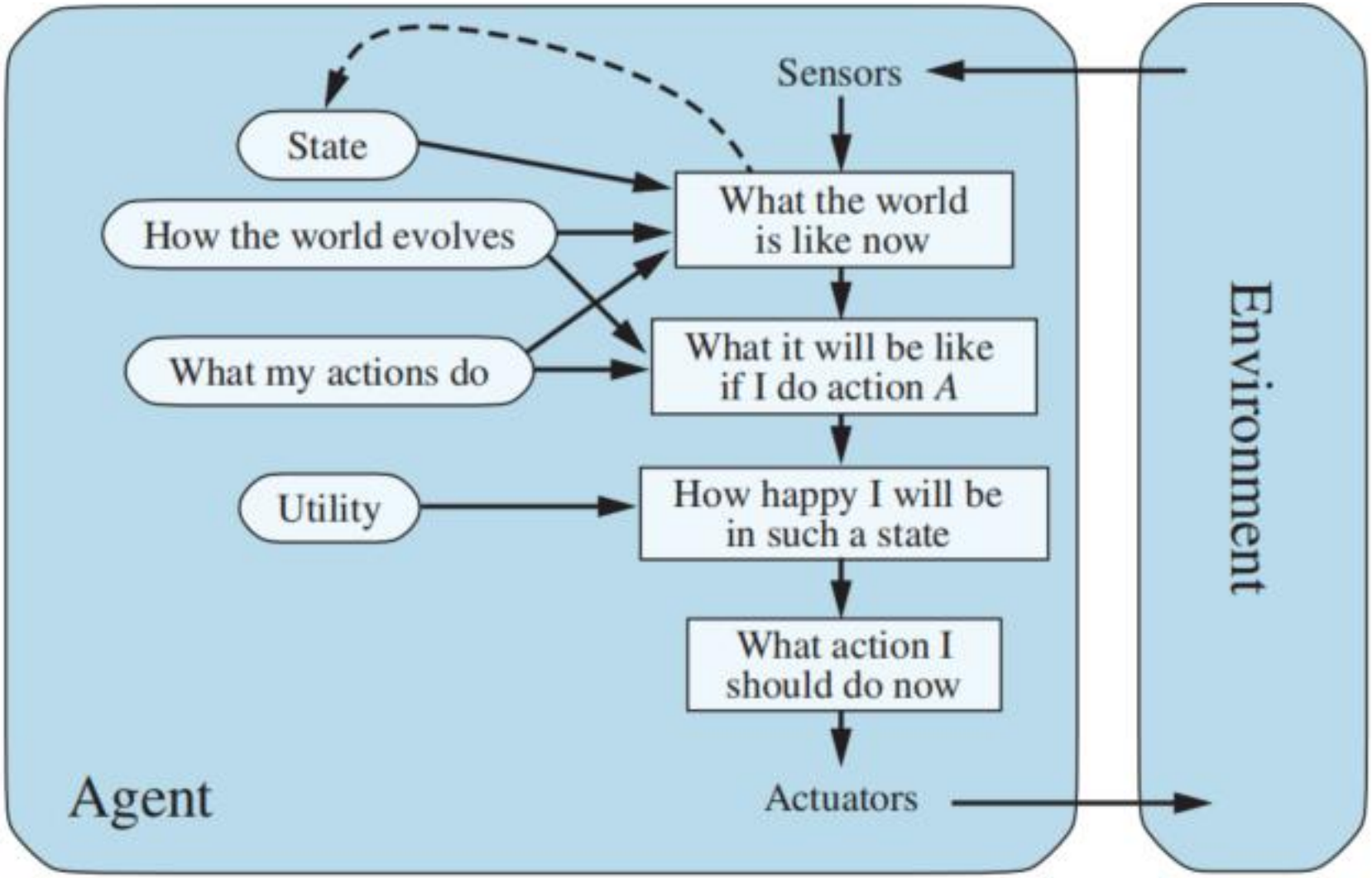
Model-based Reflex Agent



Model-based Goal-based Agent



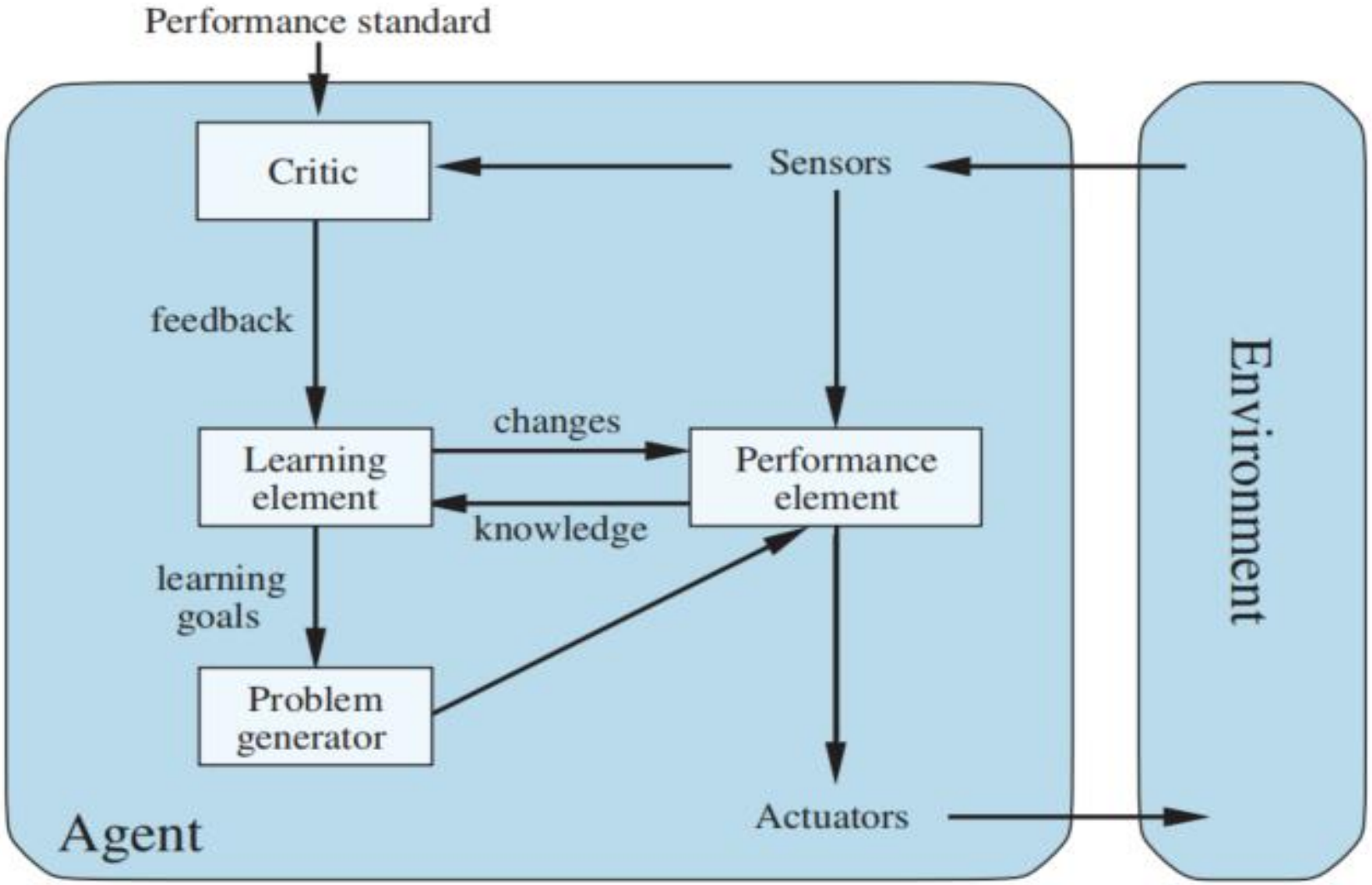
Model-based Utility-based Agent



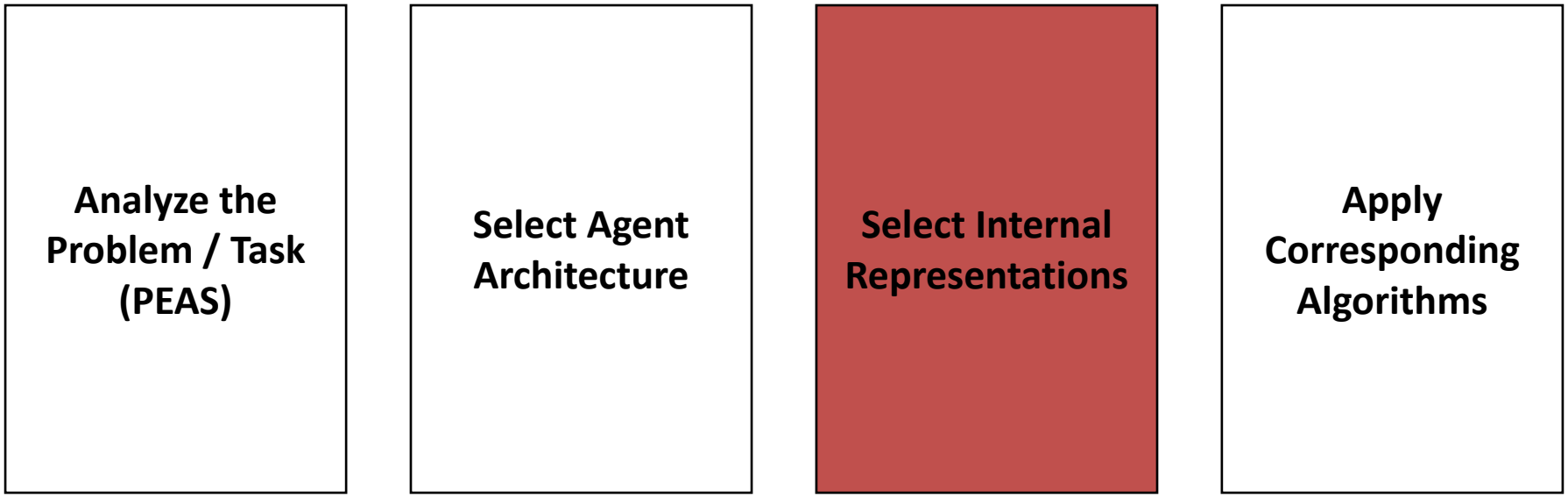
Typical Agent Architectures

- 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))

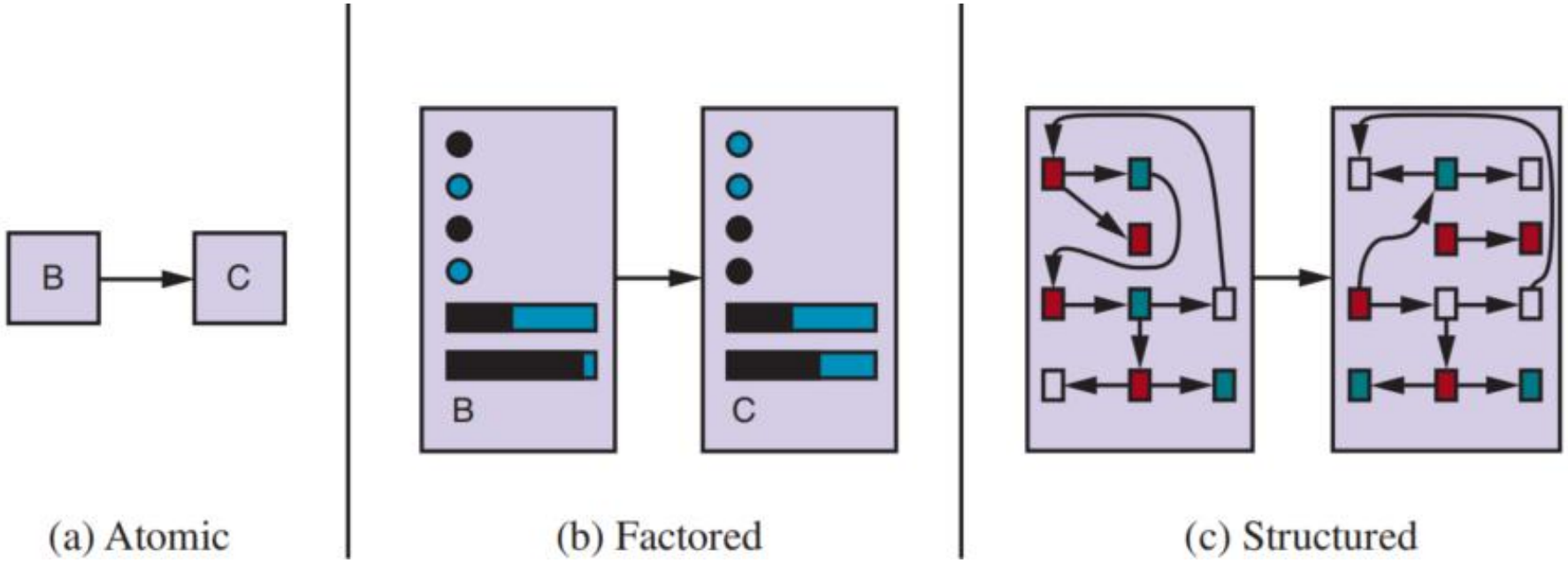
Learning Agent



Designing the Agent for the Task

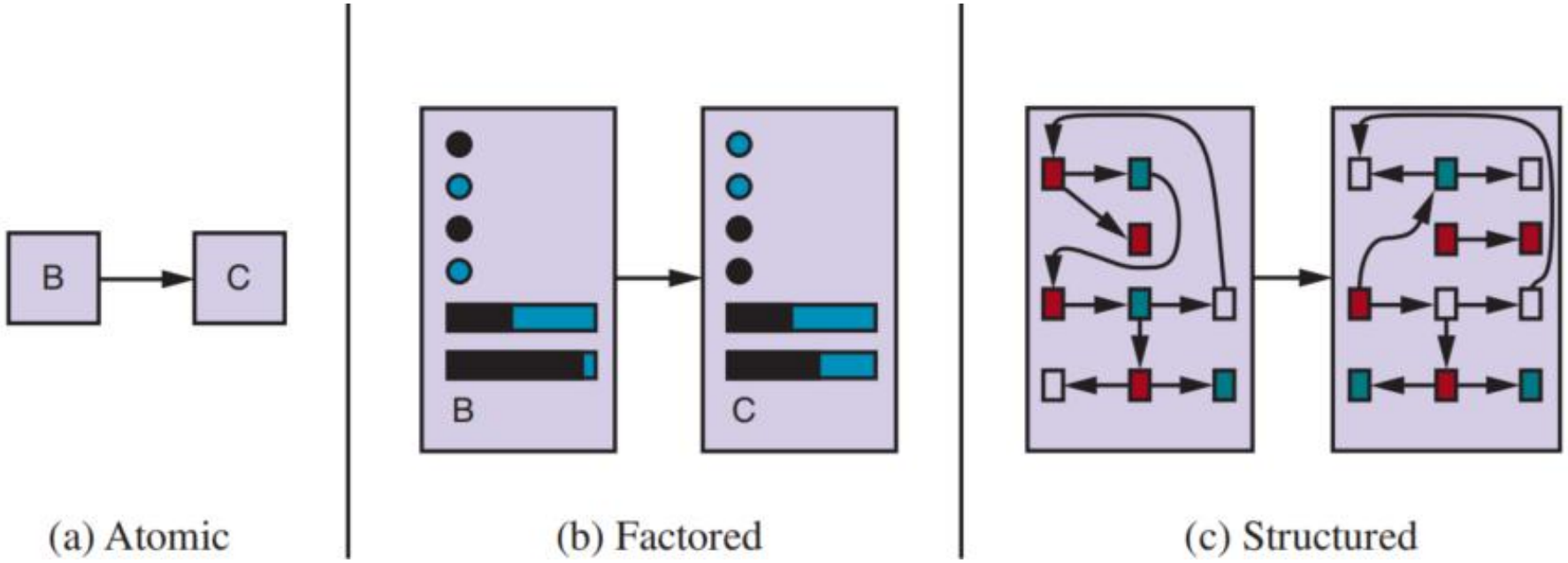


State and Transition Representations



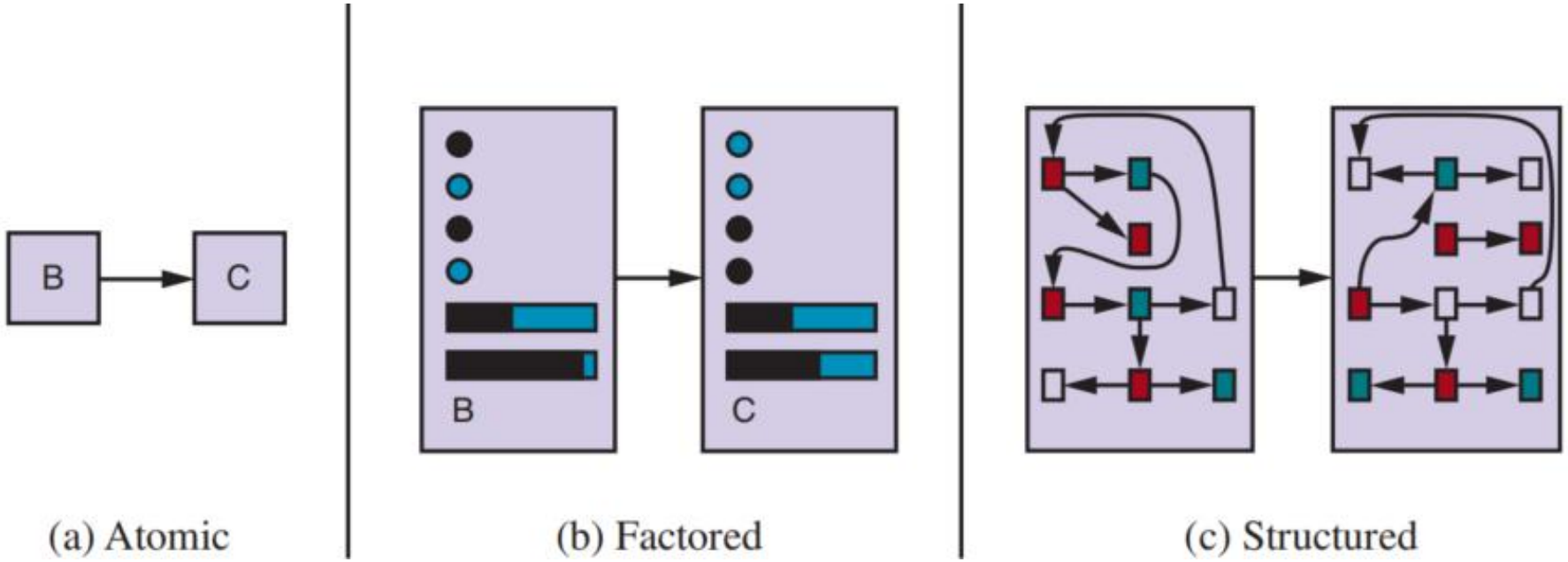
- **Atomic: state representation has NO internal structure**
- **Factored: state representation includes fixed attributes (which can have values)**
- **Structured: state representation includes objects and their relationships**

State and Transition Representations



Complexity, level of detail, expresiveness, more difficult to process

Representations and Algorithms



- 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)

Finite State Machine: A Turnstile

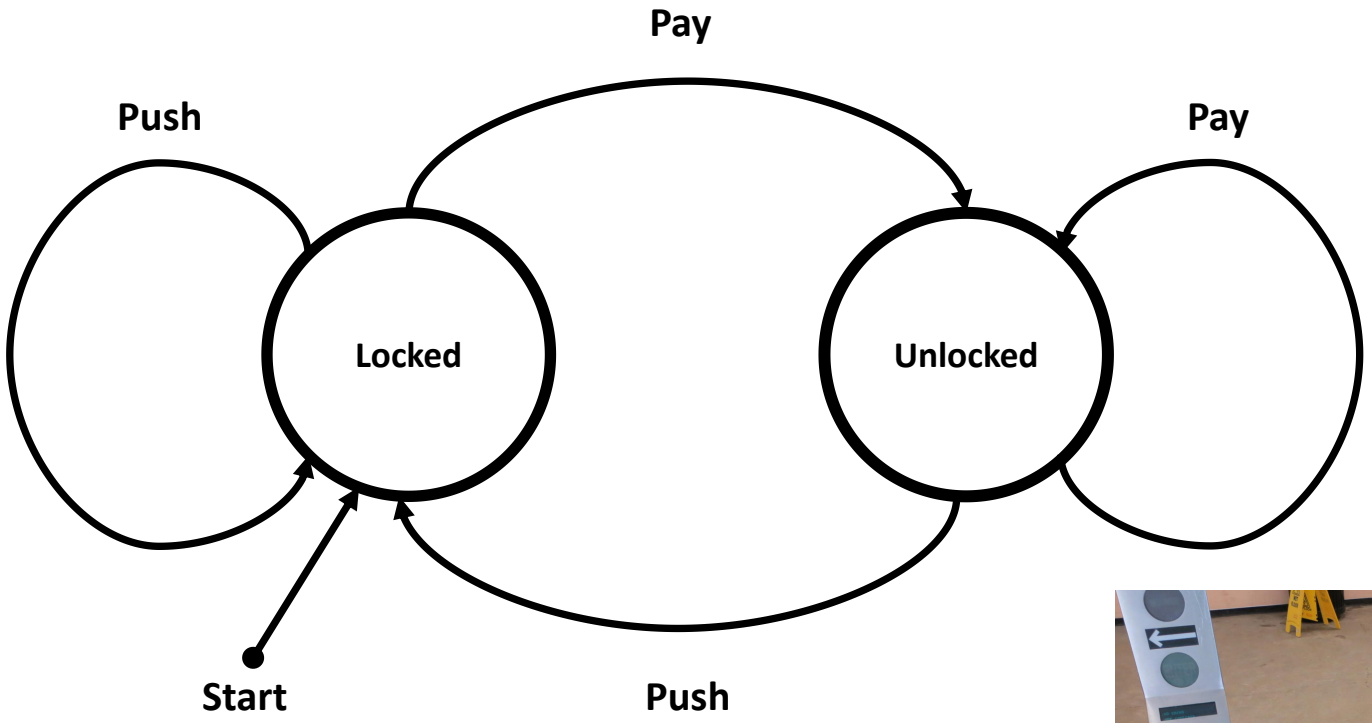
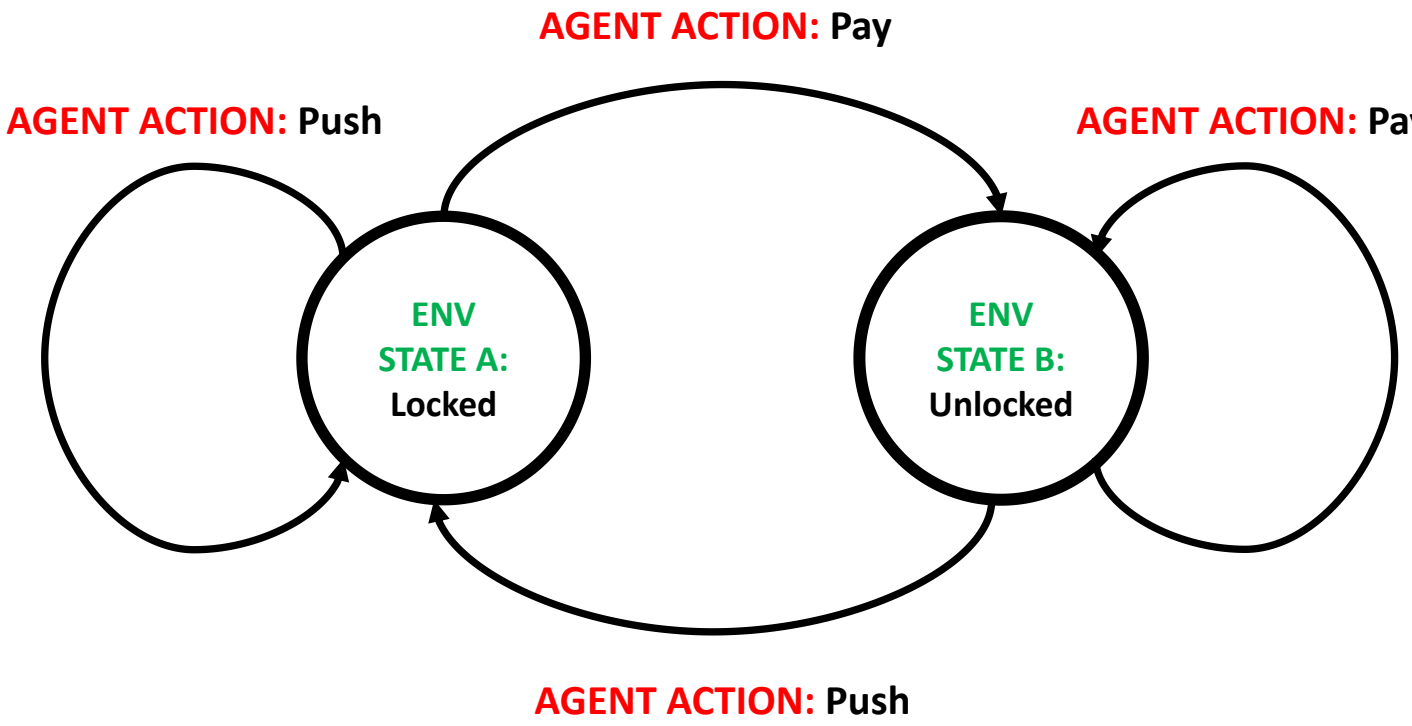
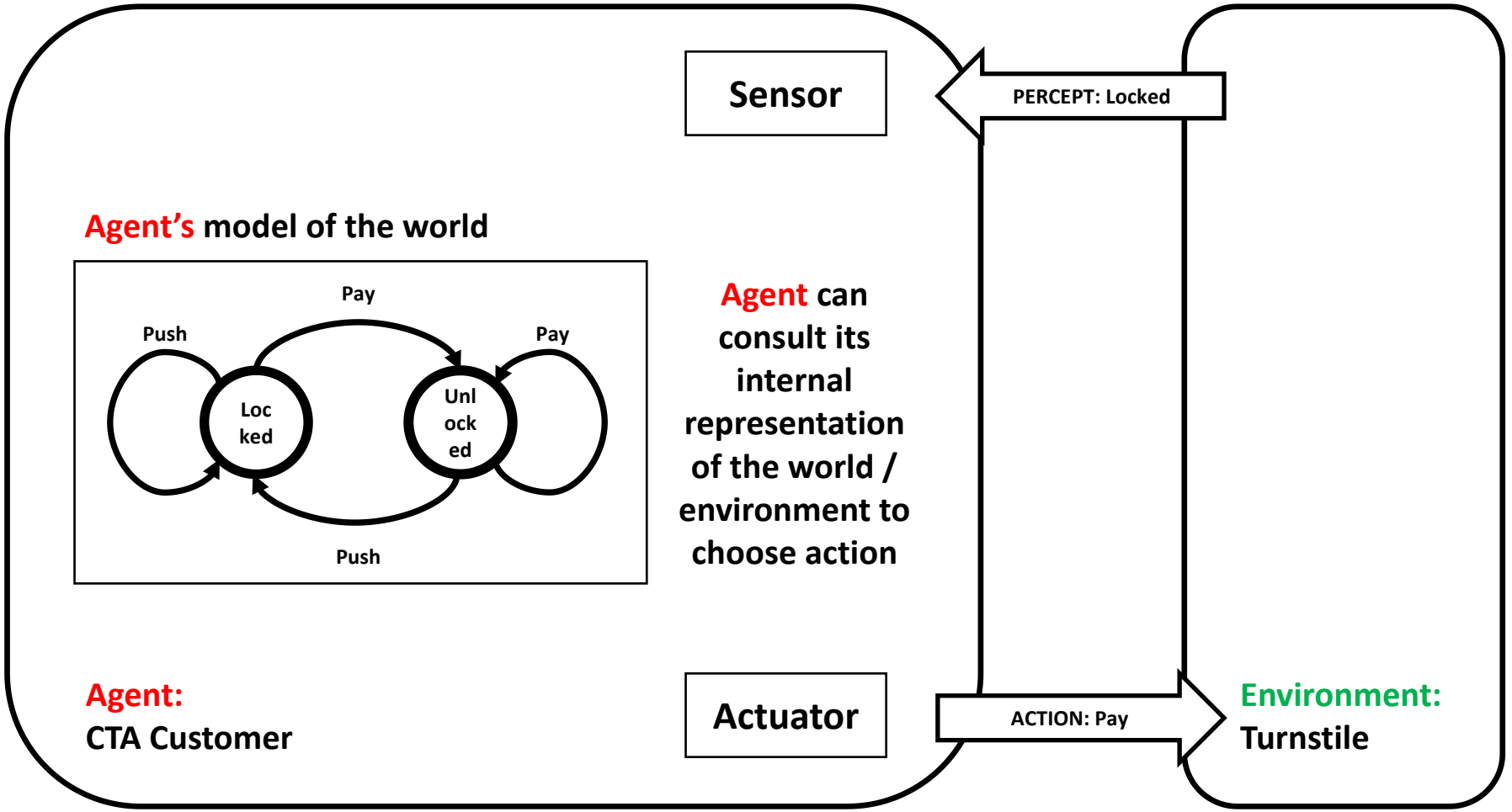


Image source: Wikipedia

Finite State Machine: A Turnstile

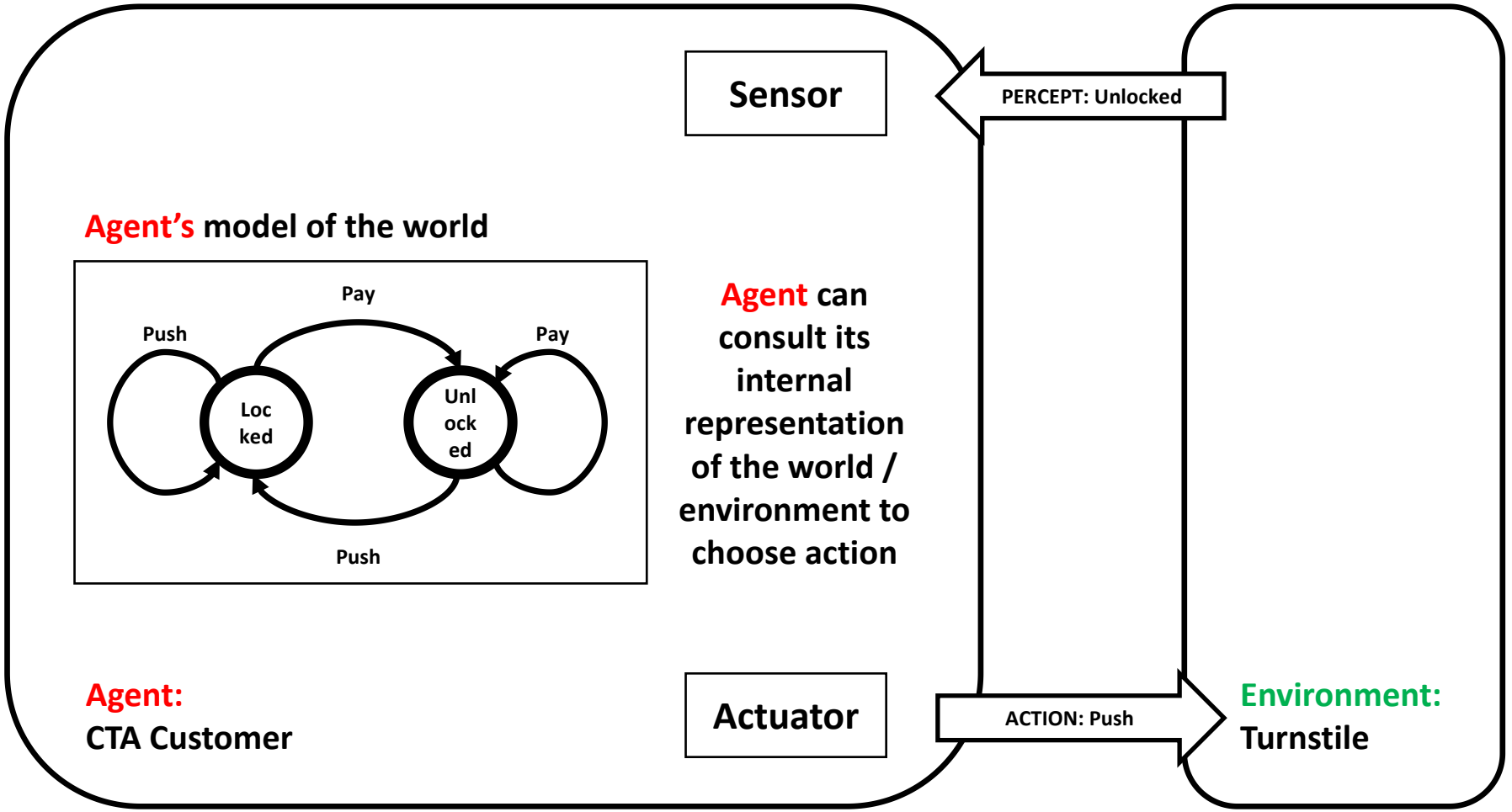


Model-based Reflex Agent Example



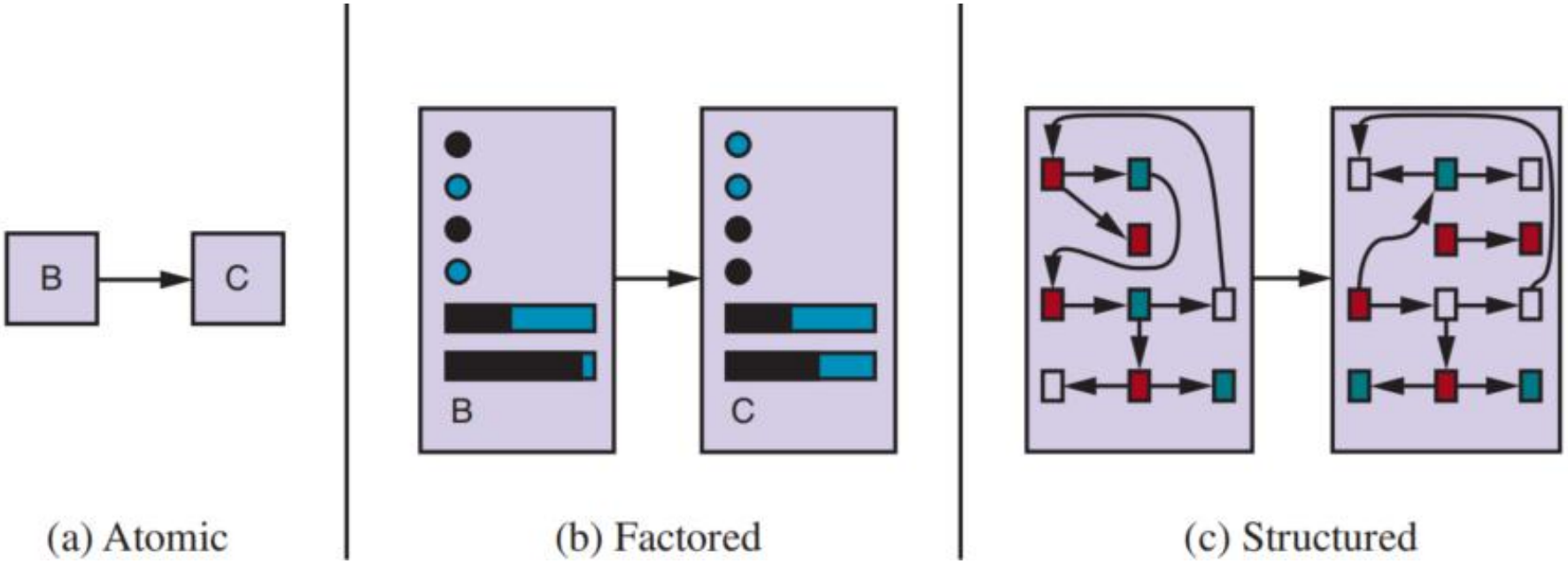
Note: This problem could be easily solved with a simple (without internal model) reflex agent.

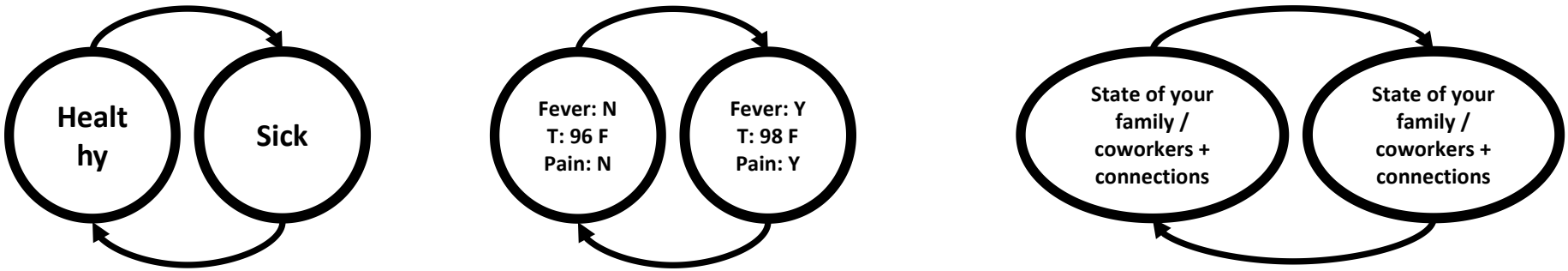
Model-based Reflex Agent Example



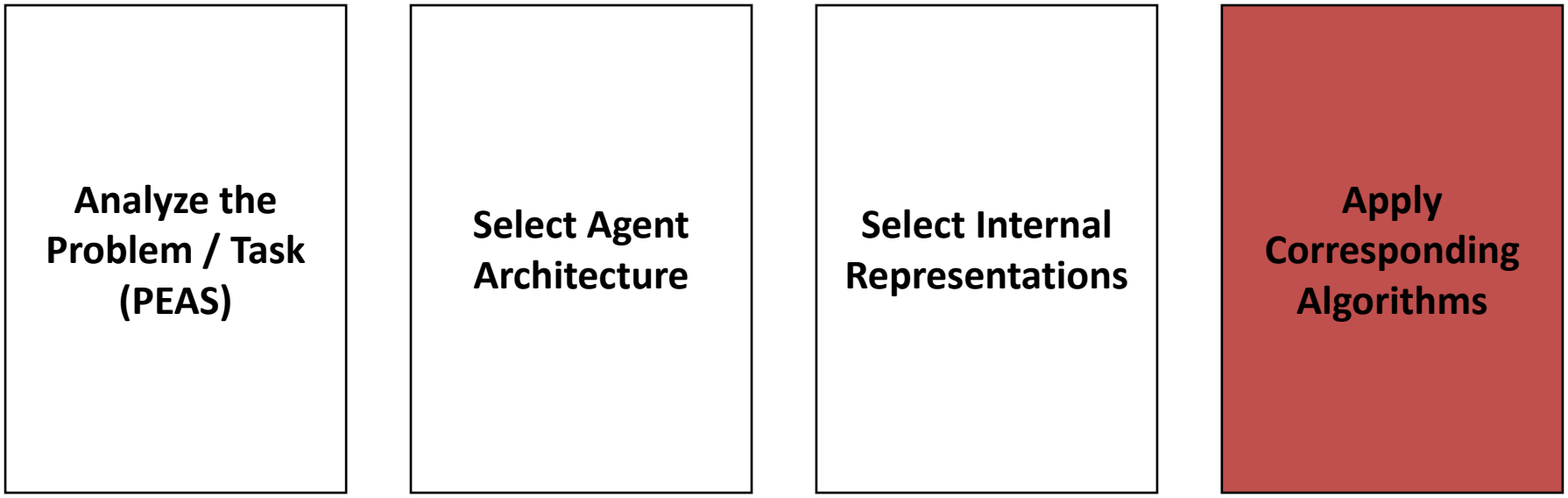
Note: This problem could be easily solved with a simple (without internal model) reflex agent.

Representations: Examples





Designing the Agent for the Task



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 search 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":

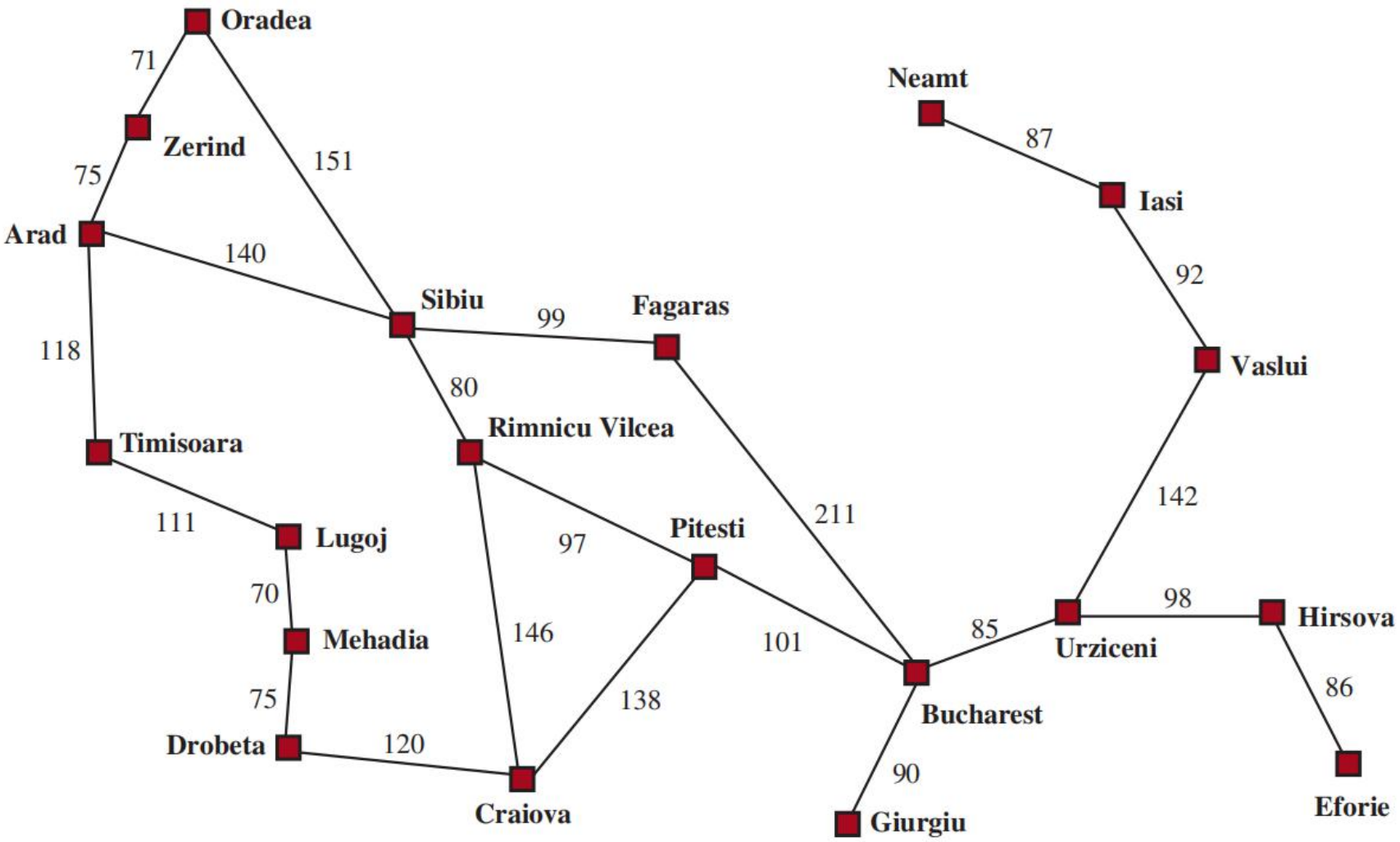
- **Fully observable**
- **Single agent (for now -> it can be multiagent)**
- **Deterministic**
- **Static**
- **Episodic**
- **Discrete**
- **Known to the agent**

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

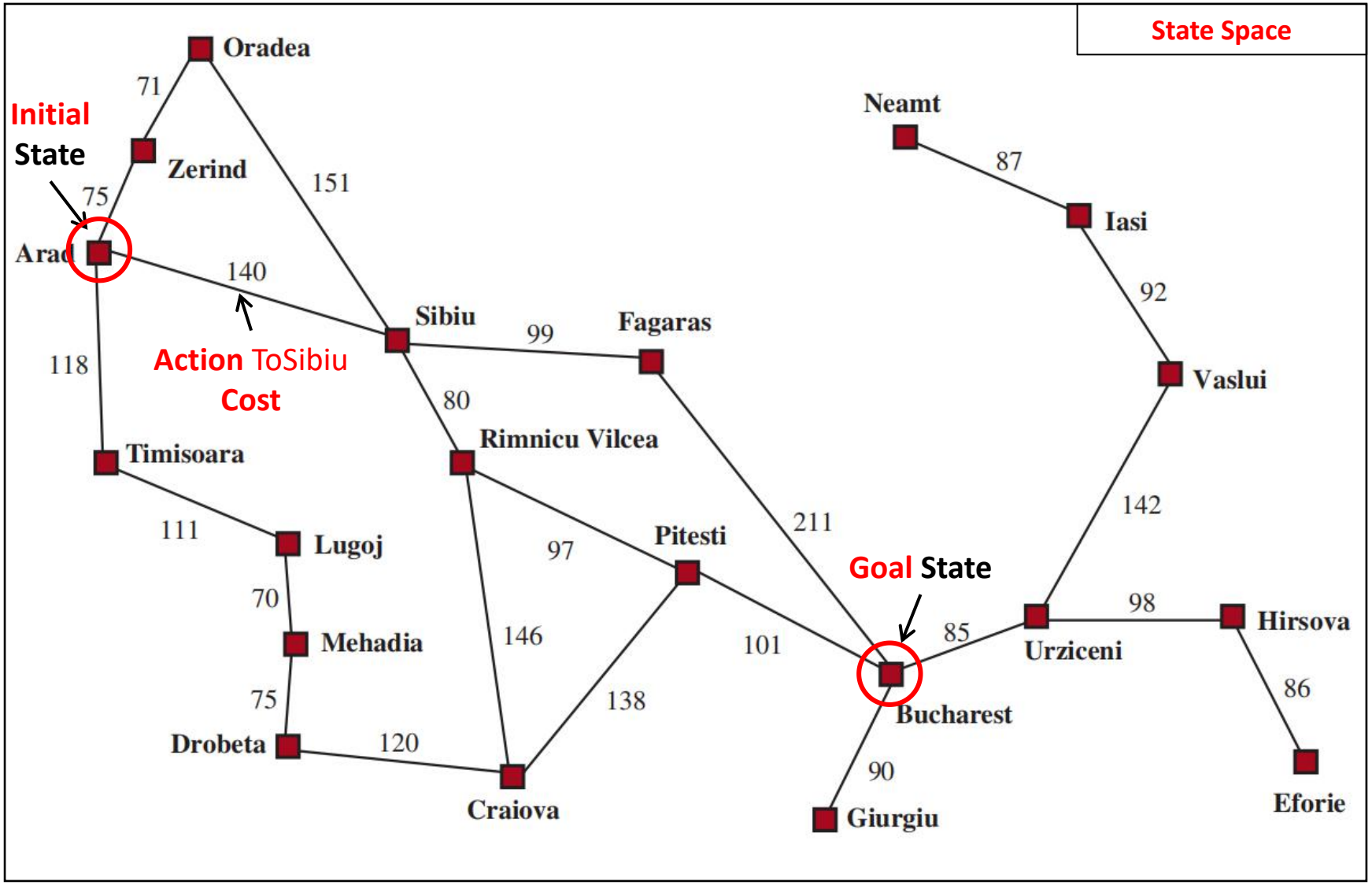


Problem: Get from Arad to Bucharest efficiently (for example: quickly or cheaply).

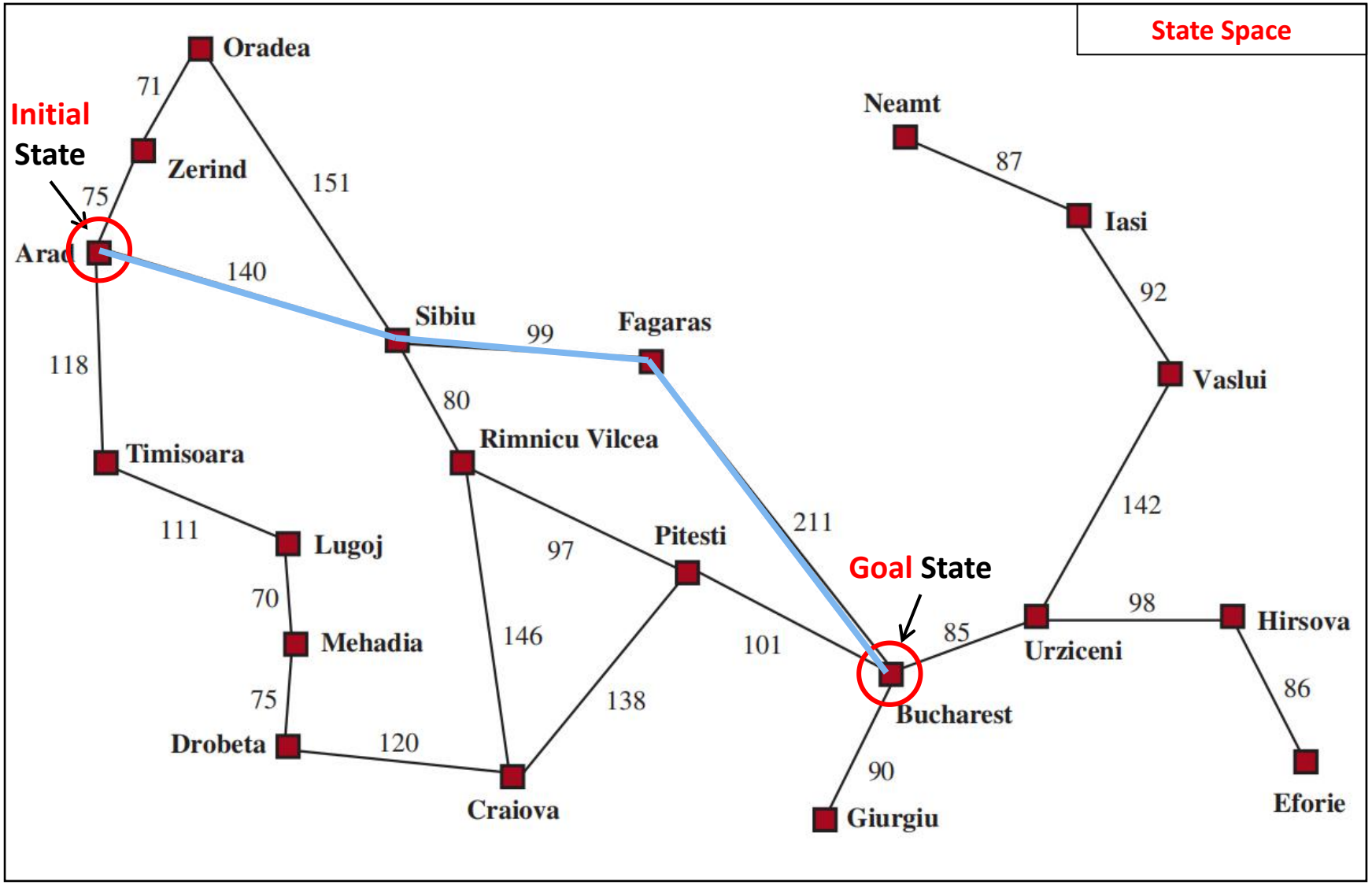
Search Problem: Dracula's Roadtrip

- 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

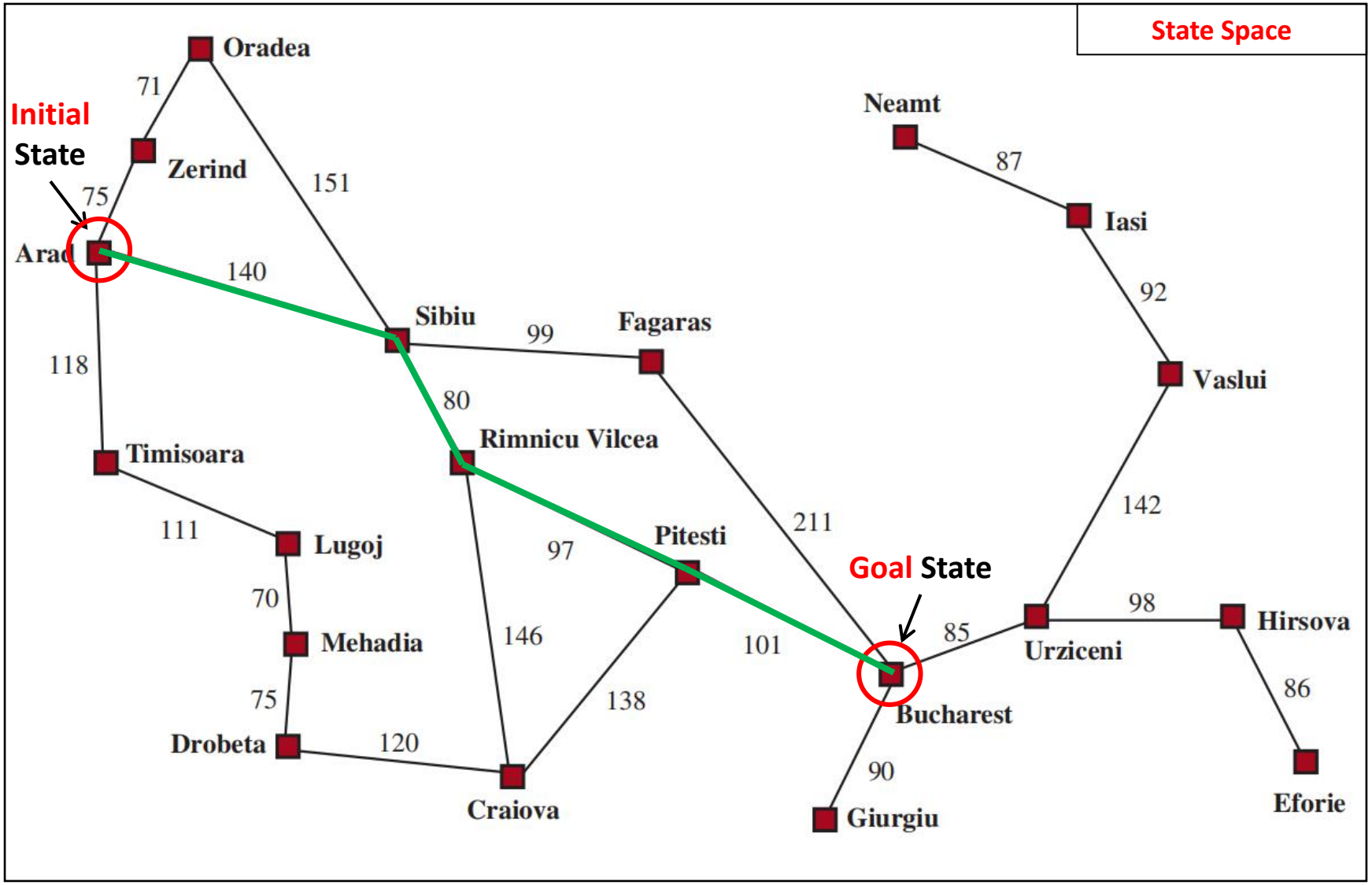
Sample Problem: Dracula's Roadtrip



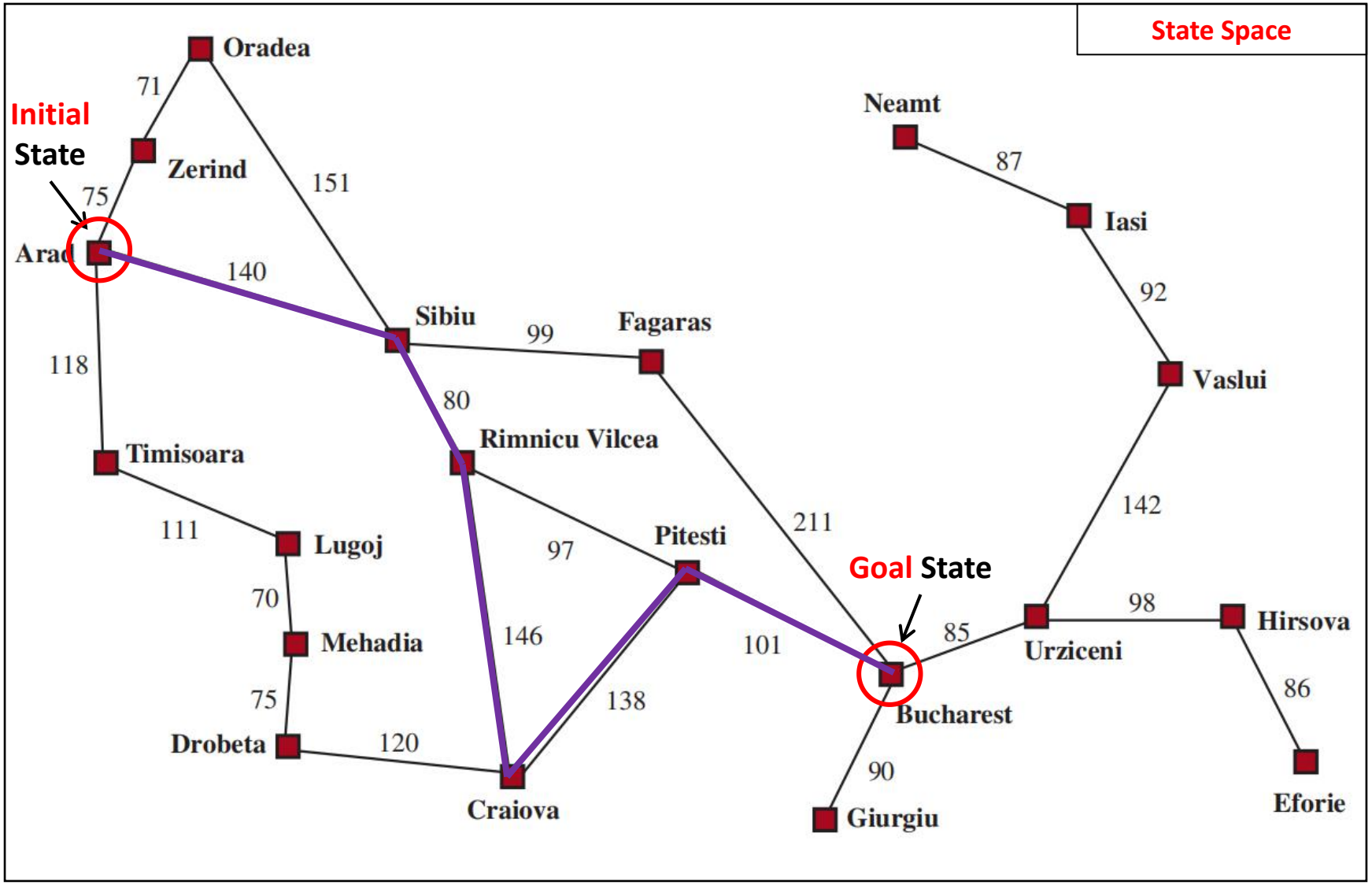
Dracula's Roadtrip: Potential Solution



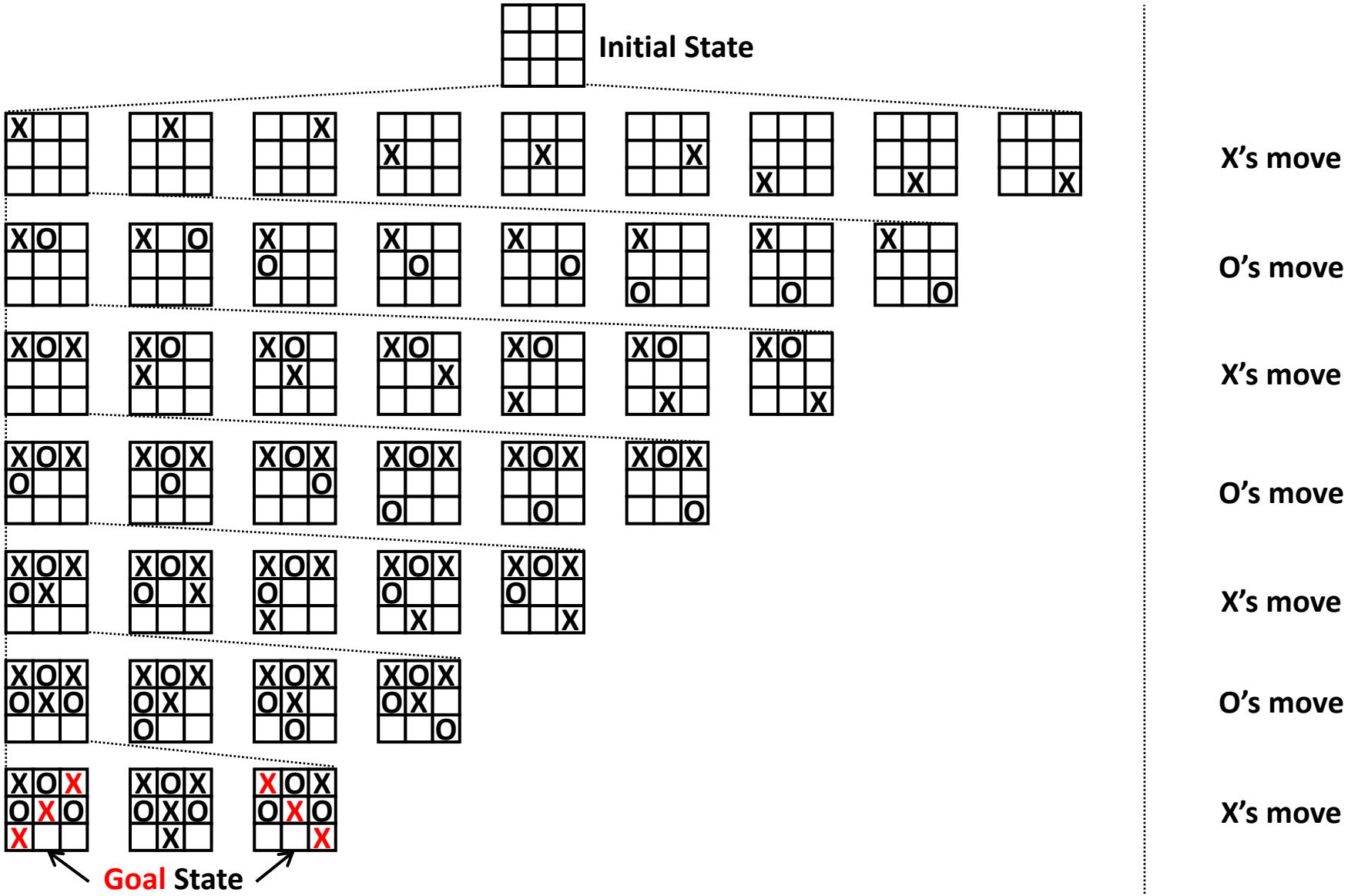
Dracula's Roadtrip: Potential Solution



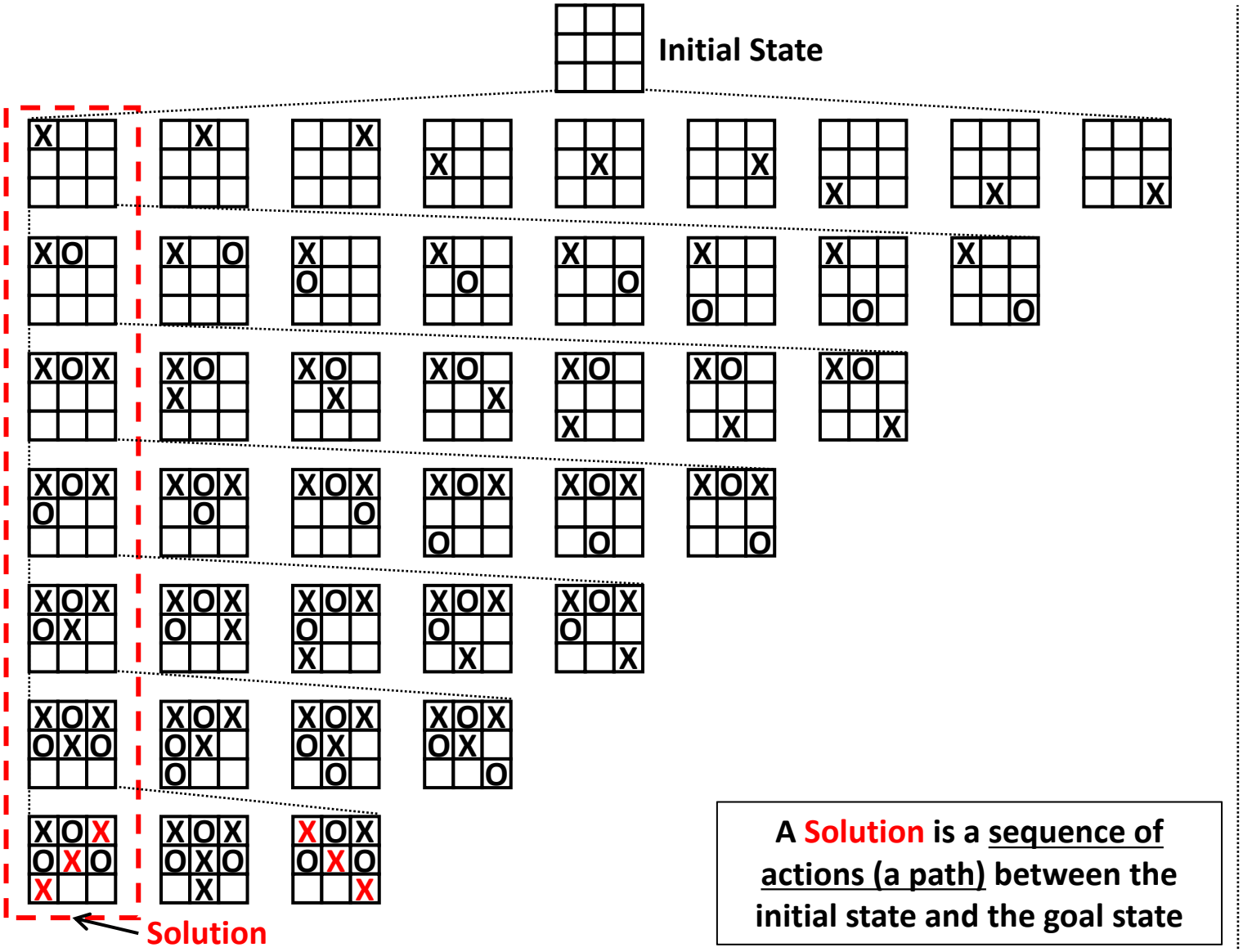
Dracula's Roadtrip: Potential Solution



Tic Tac Toe: (Partial) State Space



Tic Tac Toe: Solution

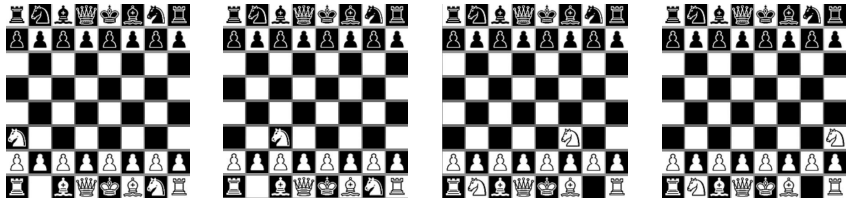


A Solution is a sequence of actions (a path) between the initial state and the goal state

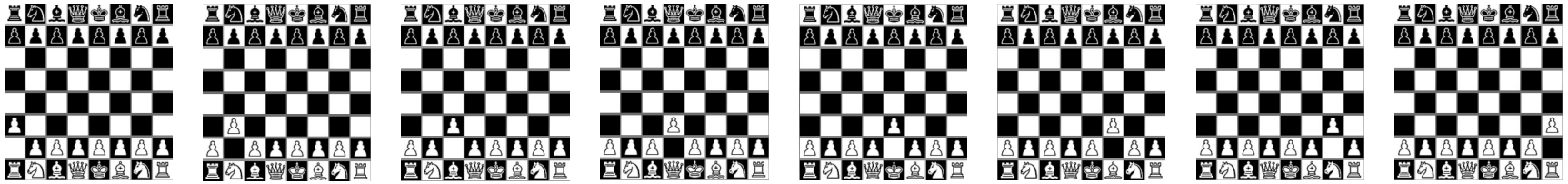
Chess: (First Move) State Space

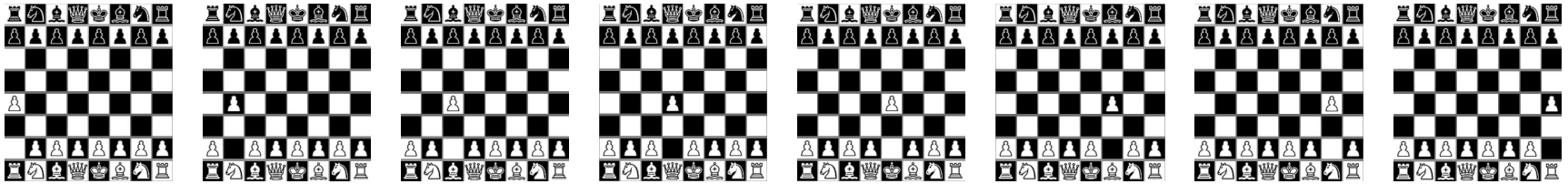
Initial State

Move

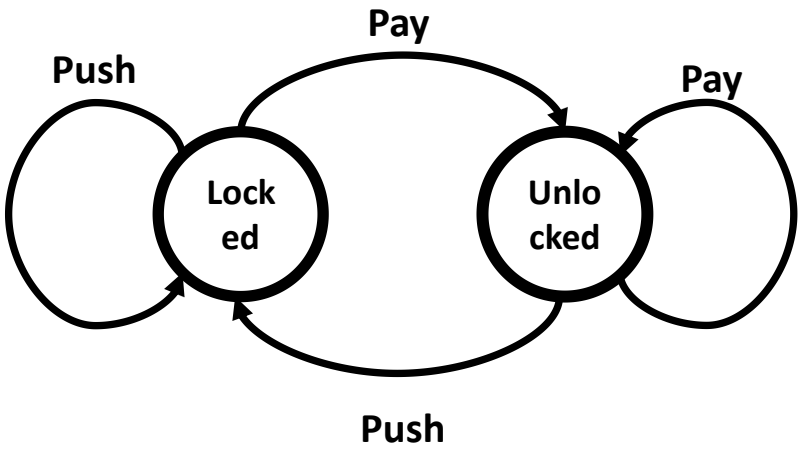


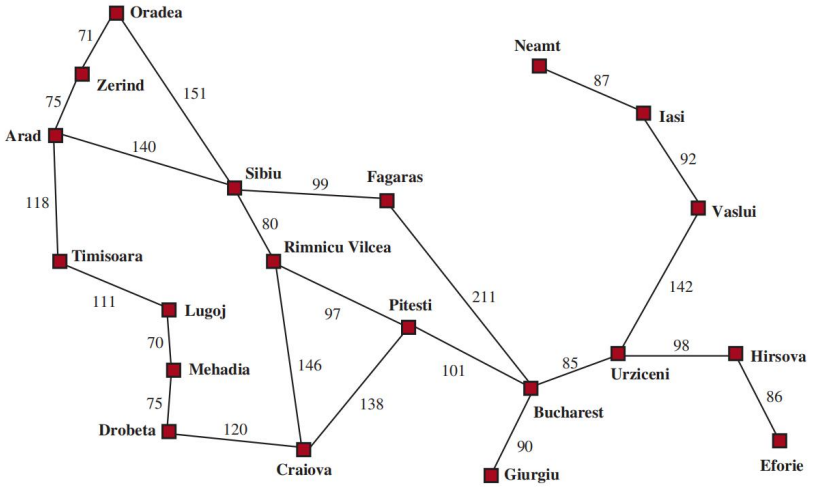
20 Possible legal first moves:
16 pawn moves
4 knight moves

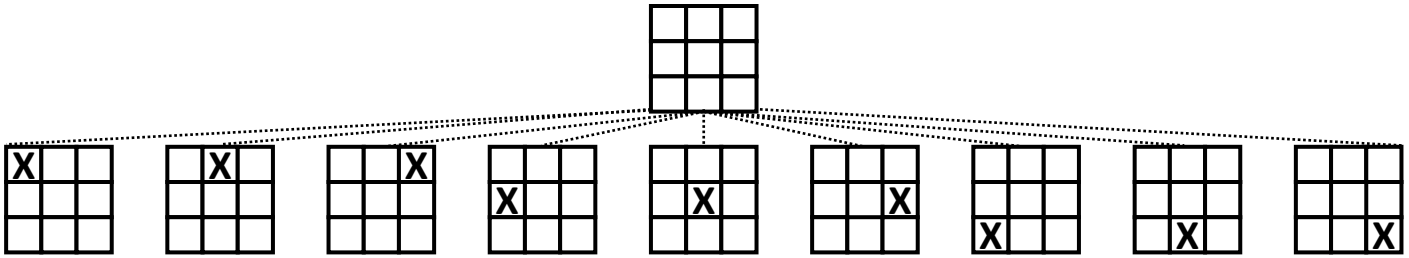




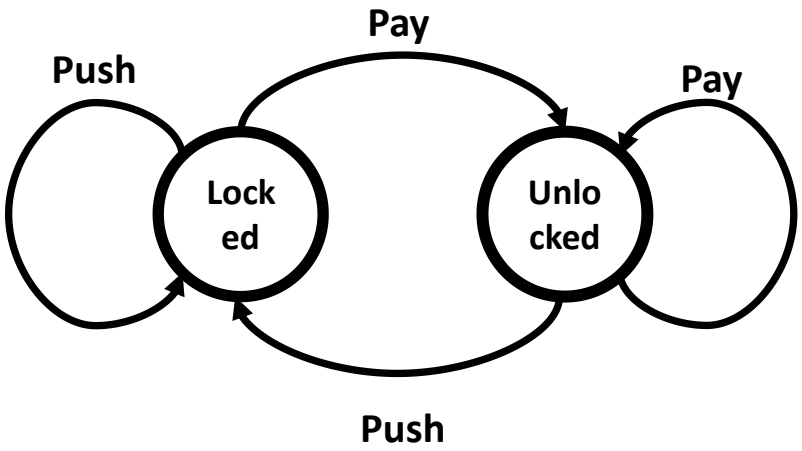
Does This Look Familiar?

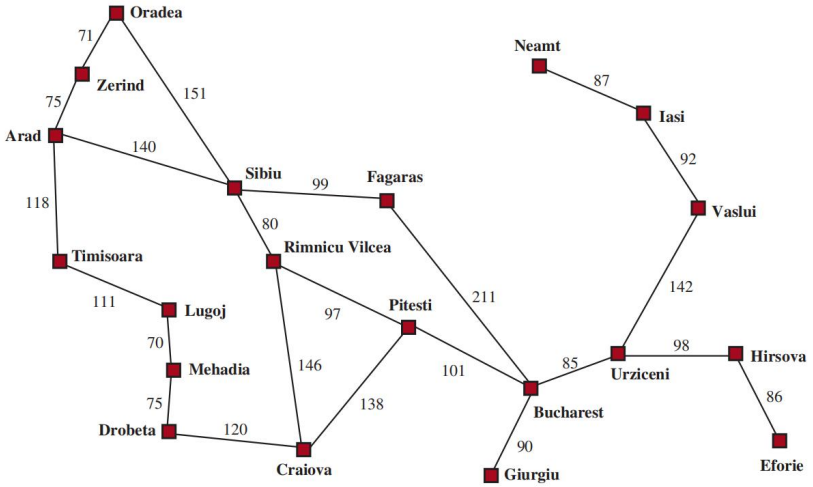


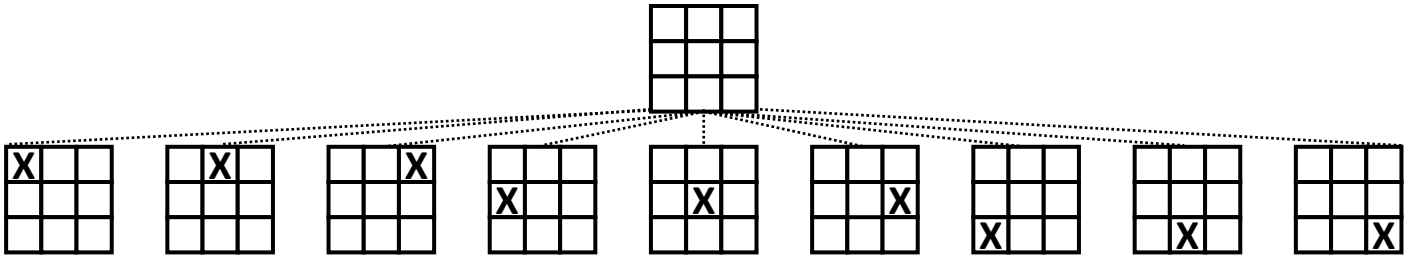




Does This Look Familiar?

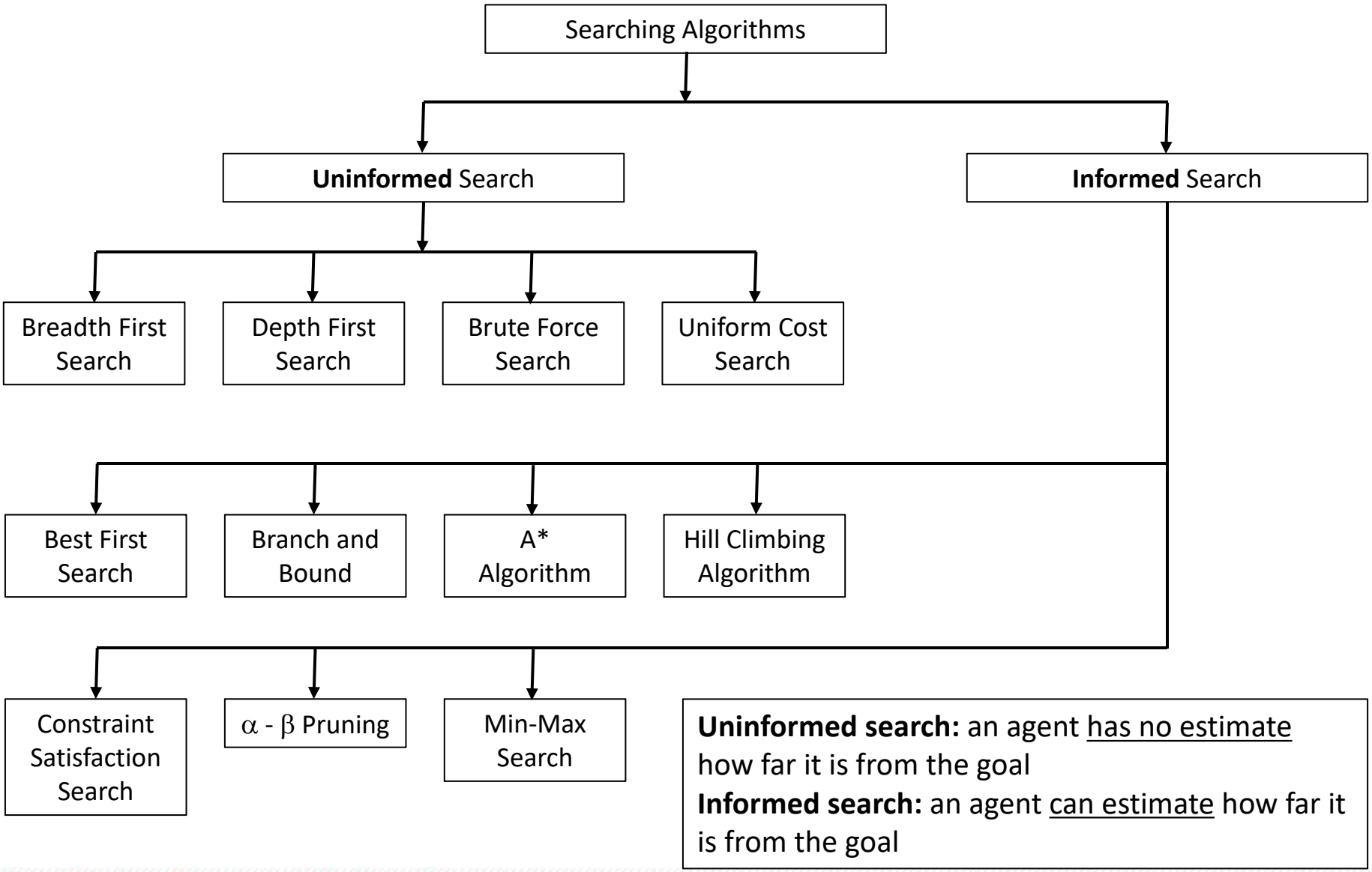






They are all graphs (some will be trees) with states as nodes and actions as links / edges.

Selected Searching Algorithms



Uninformed search: an agent has no estimate how far it is from the goal
Informed search: an agent can estimate how far it is from the goal