Lecture 2

Agents and Problem Solving

Lusi Li
Department of Computer Science
ODU

Reading for This Class:
Chapter 3, Russell and Norvig

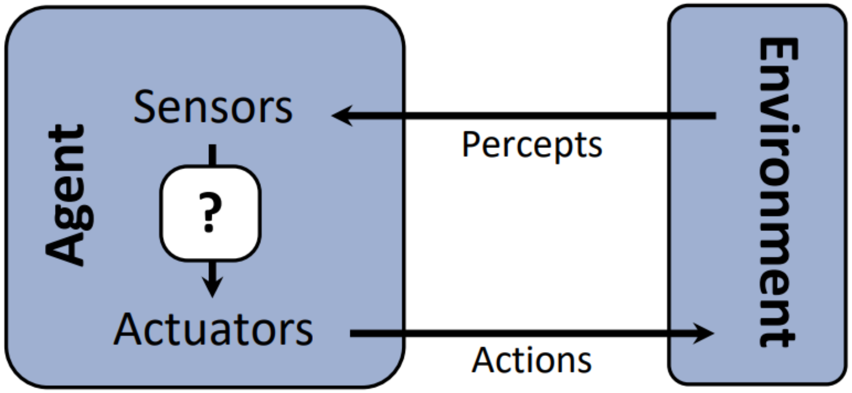
Special thanks to Dr. Yaohang Li for sharing the slides

Review

- Last Class
  - agents perceive and act in an environment
    - Description of the task environment (PEAS)
  - ideal agents maximize their performance measure
    - Rationality
  - basic agent types
    - simple reflex
    - model-based reflex
    - goal-based
    - utility-based
    - learning
- This Class
  - Problem Solving Agent
  - Problem Formulation
- Next Class
  - Problem Solving Algorithms

Review: Intelligent Agent

- Agent: Definition
  - Any entity that perceives its environment through sensors and acts upon that environment through actuators
  - Examples: human, robotic, software agents
- Perception
  - Signal from environment
- Sensors
  - Acquire percepts
  - Possible limitations
- Action
  - Attempts to affect environment
- Actuators
  - Transmits actions
  - Possible limitations



Agents

- **Rational Agents**
- **Agent PEAS Description**
- **Agent Environment**
  - fully observable vs. partially observable
  - deterministic vs. stochastic (non-deterministic)
  - episodic vs. sequential (non-episodic)
  - static vs. dynamic
  - discrete vs. continuous
  - single vs. multiple agents
- **Agent Types**
  - simple reflex
  - model-based reflex
  - goal-based
  - utility-based
  - learning

Problem-Solving Agents: Definition

- Problem-solving Agent
  - A kind of goal-based agent
  - It solves a problem by finding a sequence of actions that lead to desirable states (goals)

- Problem solving process has four steps
  - Formulate a "goal"
  - Formulate a "problem"
  - Search for a "solution"
  - Execute the solution

Problem-solving Agent
Preliminary Design

- **Goal Formulation**
  - Goals organize the agent behavior by limiting the objectives and hence the actions to be considered
  - A goal is formulated as a set of world states based on the current situation and performance measure, where the goal test is satisfied
  - Here we consider environments that are observable, discrete and deterministic

- **Problem Formulation**
  - To decide what actions and states to consider, given a goal
  - States: various world states (initial, goal, intermediate, etc.)
  - Actions: causing transitions between world states
- **Search**
  - There are many paths to achieve the same goal
  - The process of finding a sequence of actions that reaches the goal
- **Execution**
  - Execute the actions in the solution, one at a time

A Simple Problem-Solving Agent

function SIMPLE-PROBLEM-SOLVING-AGENT(*percept*) **returns** an action
  persistent: *seq*, an action sequence, initially empty
    state, some description of the current world state
    goal, a goal, initially null
    problem, a problem formulation

  $state \leftarrow$ UPDATE-STATE($state$, $percept$)
  if *seq* is empty **then**
    $goal \leftarrow \boxed{\text{FORMULATE-GOAL}(state)}$
    $problem \leftarrow \boxed{\text{FORMULATE-PROBLEM}(state, goal)}$
    $seq \leftarrow \boxed{\text{SEARCH}(problem)}$
    if *seq* = *failure* **then return** a null action
  $action \leftarrow$ FIRST(seq)
  $seq \leftarrow$ REST(seq)
  return *action*

Example: Romania Touring

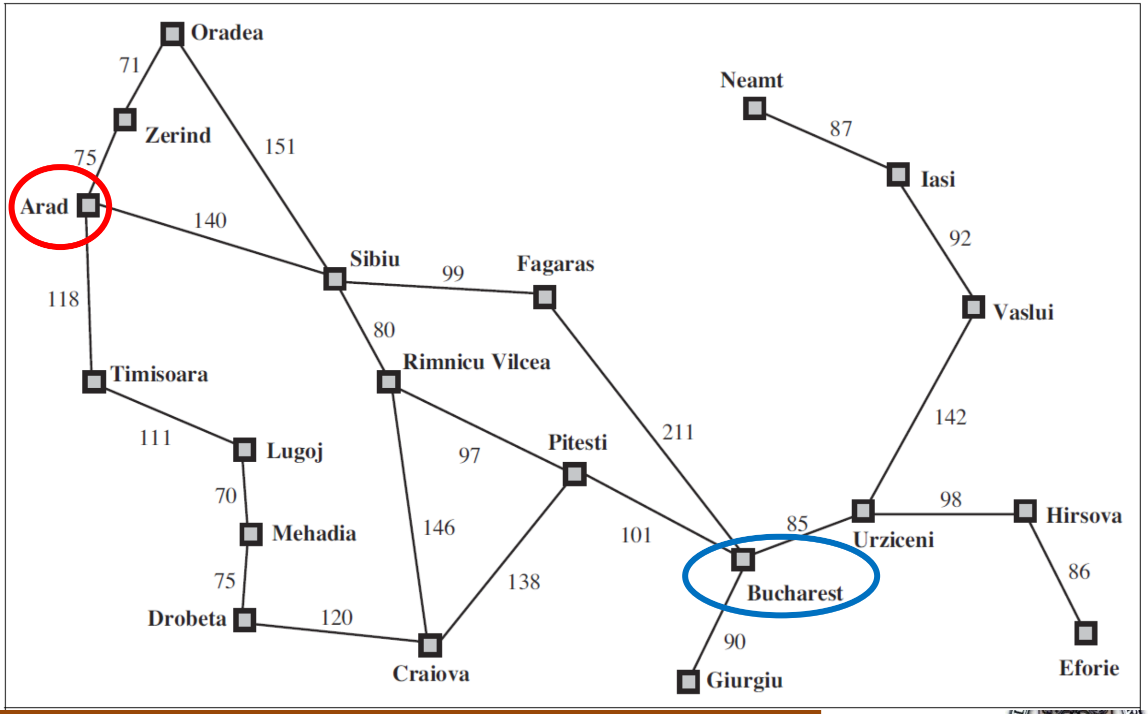
- **On holiday in Romania; currently in Arad with a car**
- **Non-refundable ticket to fly out of Bucharest tomorrow**

- **Formulate goal (perf. evaluation):**
  - **be in Bucharest before the flight**

- **Formulate problem:**
  - **states: various cities**
  - **actions: drive between cities**

- **Search:**
  - **a sequence of cities**

Road Map of Romania



A Well-defined Problem

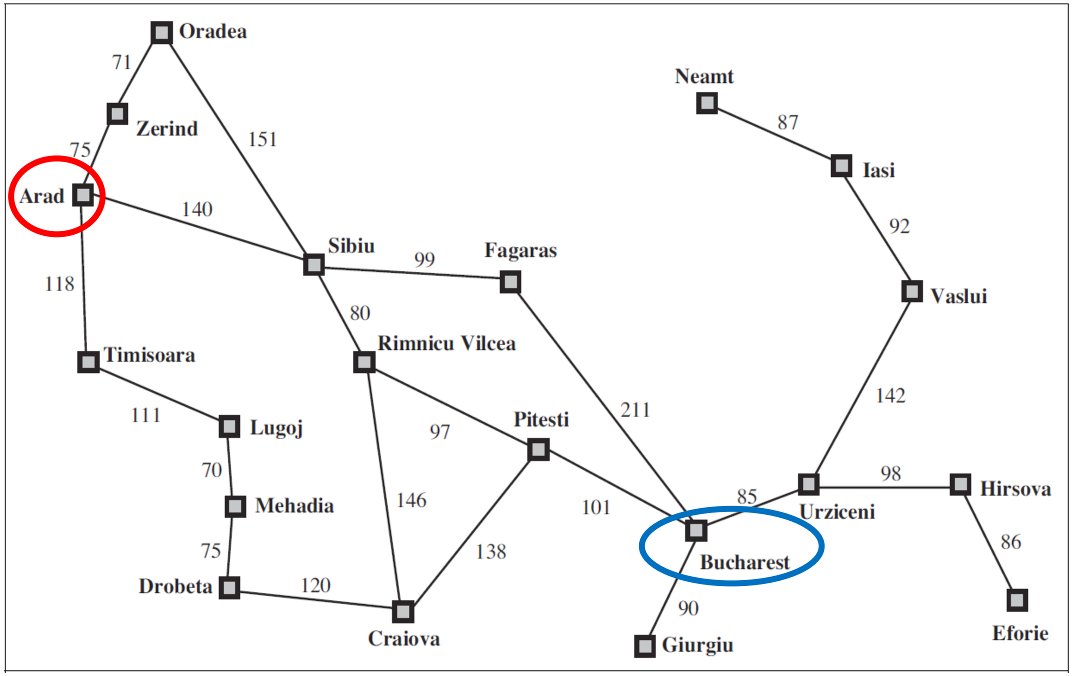
- A problem can be defined formally by five components
  - Initial state
    - The agent starts in
  - Actions
    - A description of the possible actions available to the agent
    - E.g., Given a state *s*, ACTION(s) returns a set of actions that can be executed in *s*
  - Transition model (successor function)
    - A description of what each action does (returns successor)
    - E.g., RESULT(*s, a*) returns the state that results from doing *a* in *s*
  - Goal test
    - Determines whether a given state is a goal state
    - Can be explicit (goal state) or implicit (described by some properties)
  - Path cost function
    - Assigns a numeric cost to each path (a sequence of actions)
    - Step cost (cost of each action)

- Initial state, actions and transition model define the state space
  - the set of all states reachable from the initial state by any sequence of actions
  - forms a graph where nodes are states and the links are actions

Problem Formulation – Example: Romania Touring

- **Initial state:** **In(Arad)**
- **Actions:**
  - **If current state is In(Arad), actions = {Go(Sibiu), Go(Timisoara), Go(Zerind)}**
- **Transition model:**
  - **e.g., Results(In(Arad), Go(Sibiu)) = In(Sibiu)**

Road Map of Romania

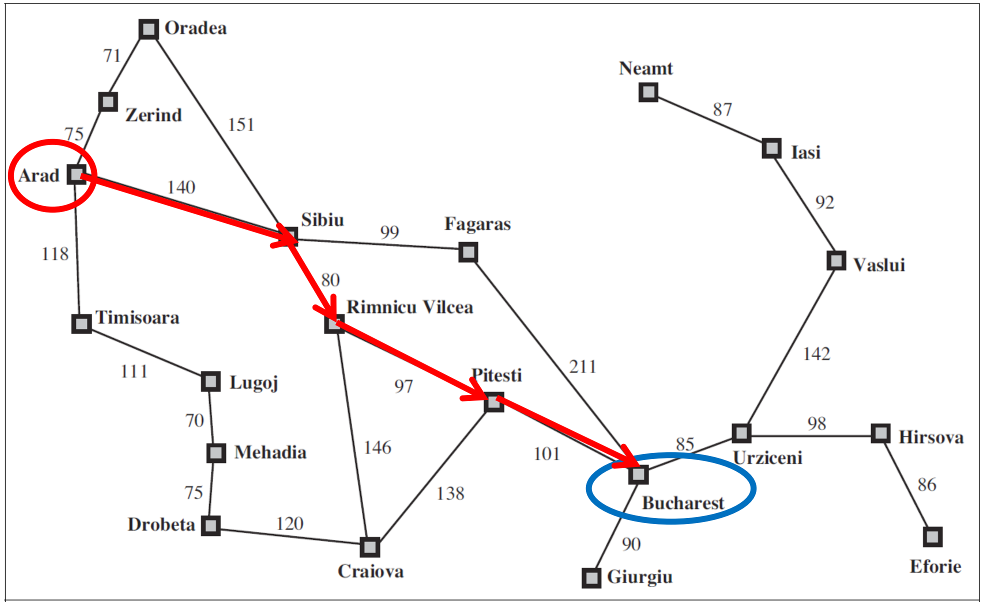


- **Goal test:**
  - **explicit, e.g. In(Bucharest)**
- **Path cost function**
  - **e.g., distance traveled**

Solution

- A sequence of actions forms a path
- Solution
  - A path from the initial state to a goal state
  - A problem may have more than one solutions
- Optimal Solution
  - Has the lowest path cost among all solutions
  - E.g., Arad, Sibiu, Rimnicu Vilcea, Bucharest

Road Map of Romania



Representation of States and Actions

- **States:**
  - The state of the world includes so many things, e.g., weather, radio, scenery
  - What information is necessary to encode to solving the goal?
  - The size of a problem is usually described in terms of the number of states that are possible.
- **Actions:**
  - The number of actions/operators depends on the representation used in describing a state.
  - E.g., turn left VS turn left 30 degree
- **Good representation for both states and actions can greatly simplify a problem!**

Problem Abstraction

- Real world is complex and has more details
- Abstraction
  - the process to take out the irrelevant information
  - leave the most essential parts to the description of the states
- What's the appropriate level of abstraction?
  - the abstraction is valid, if we can expand it into a solution in a more detailed world
  - the abstraction is useful, if carrying out the actions in the solution is easier than the original problem
  - remove as much detail as possible while retaining validity and usefulness
- Note that every problem formulation involves abstractions

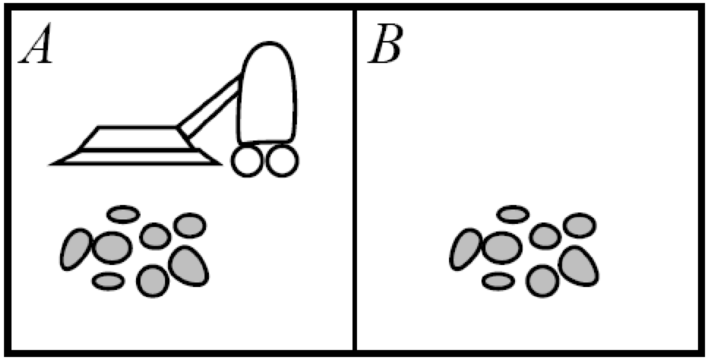
Problem-Solving Agents: Problem Formulation

- **Decide**
  - **States?**
  - **Initial State?**
  - **Actions?**
  - **Transition model?**
  - **Goal test?**
  - **Path cost?**

Review: initial state, actions and transition model define the state space

- **the set of all states reachable from the initial state by any sequence of actions**
- **forms a graph where nodes are states and the links are actions**

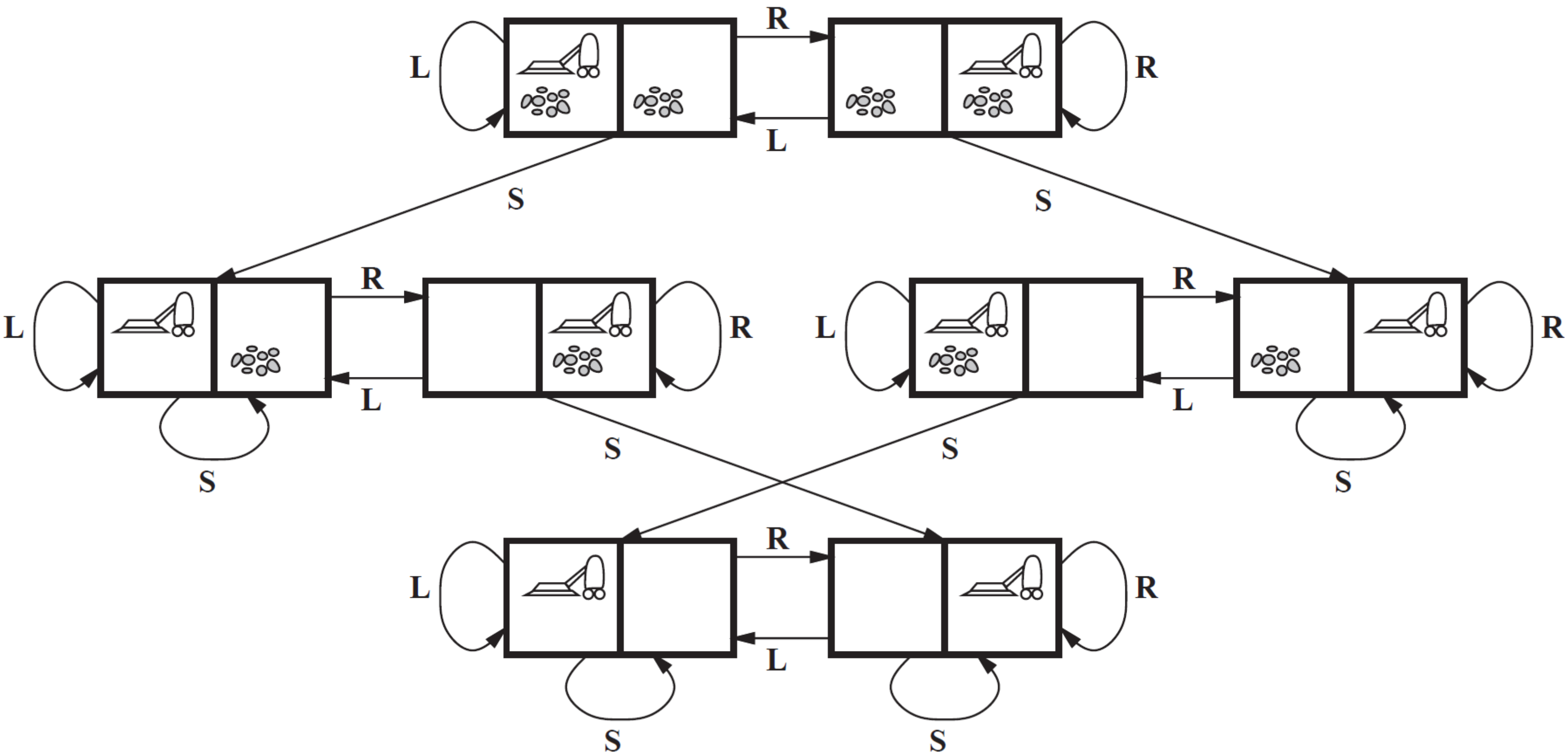
Example: Vacuum-Cleaner World



- **States**
  - agent location and dirt location
  - number of states: $n * 2^n$ (n squares)
- **Initial state**
  - any state
- **Actions**
  - Left, Right, and Suck [simplified: left out No-op]
- **Transition model**
  - the actions have their expected effects
  - complete state space, see next page
- **Goal test**
  - whether both squares are clean
- **Path cost**
  - each step costs 1
  - # of steps in the path

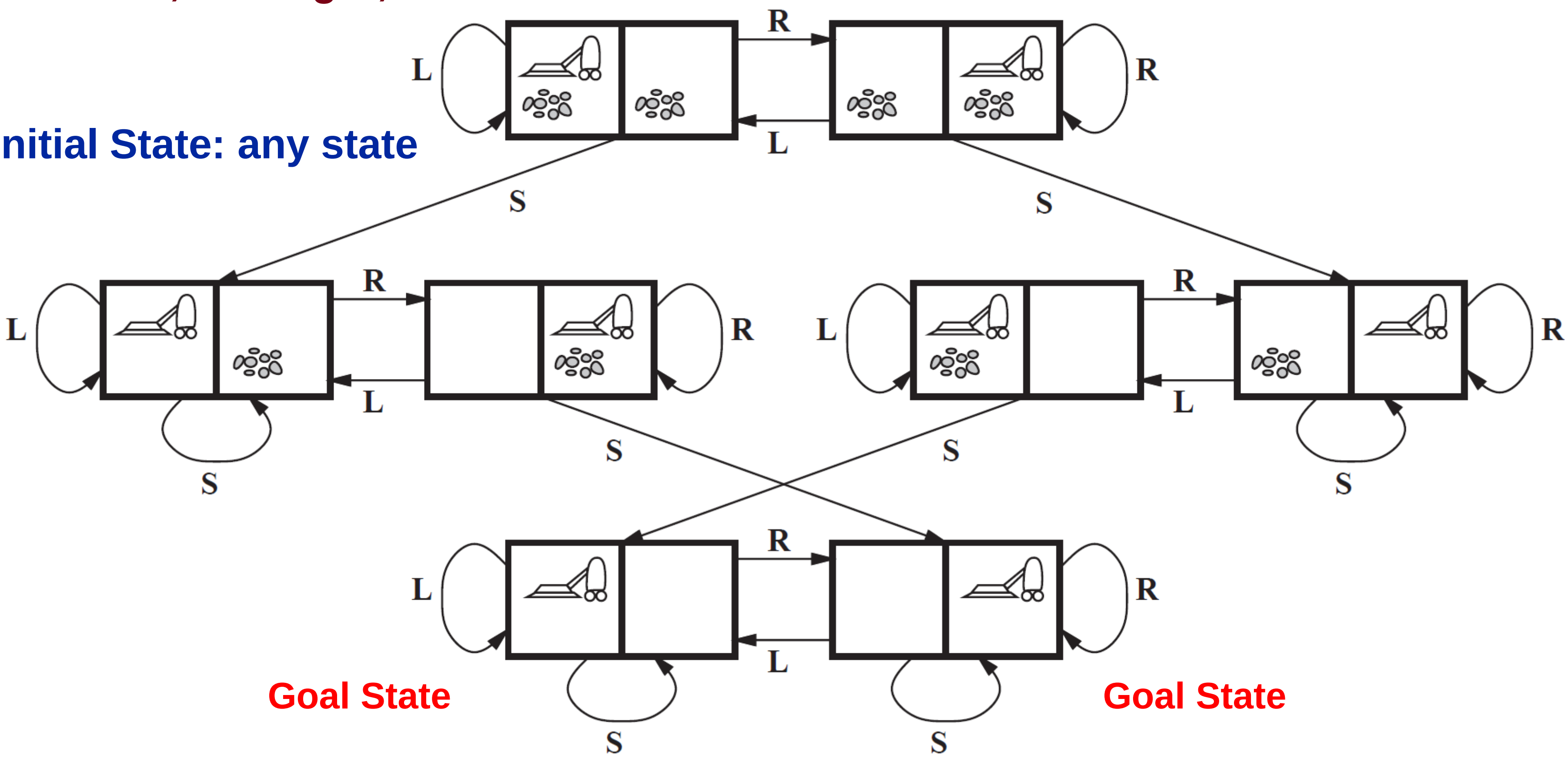
Example: Vacuum-Cleaner World

- The state space graph for the vacuum world.
  - Each node represents a state and each edge denotes an action
- L = Left, R = Right, S = Suck.



Example: Vacuum-Cleaner World

- The state space graph for the vacuum world.
  - Each node represents a state and each edge denotes an action
- L = Left, R = Right, S = Suck.



Initial State: any state

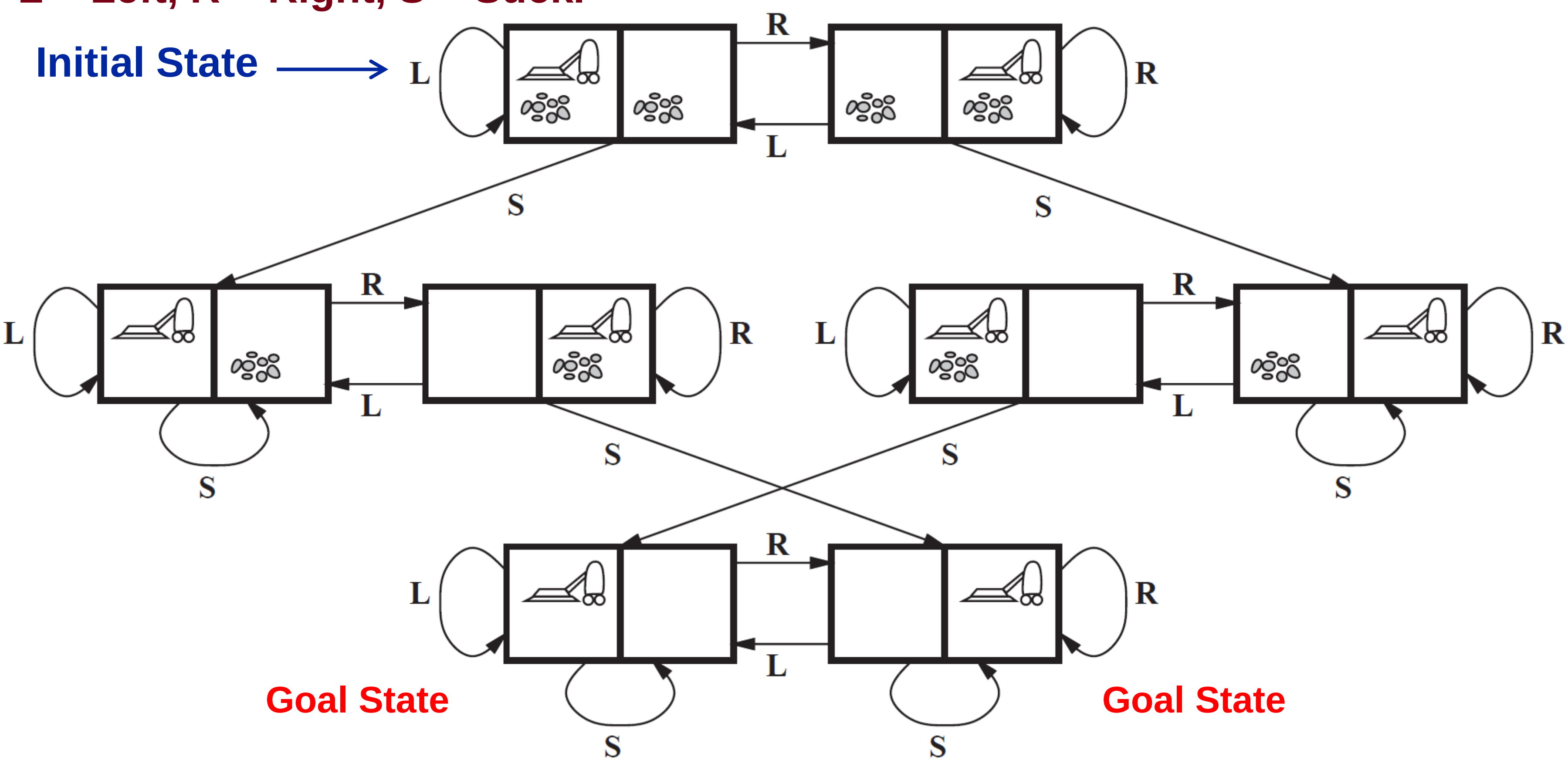
Goal State

Goal State

Example: Vacuum-Cleaner World

- **The state space graph for the vacuum world.**
  - **Each node represents a state and each edge denotes an action**
- **L = Left, R = Right, S = Suck.**

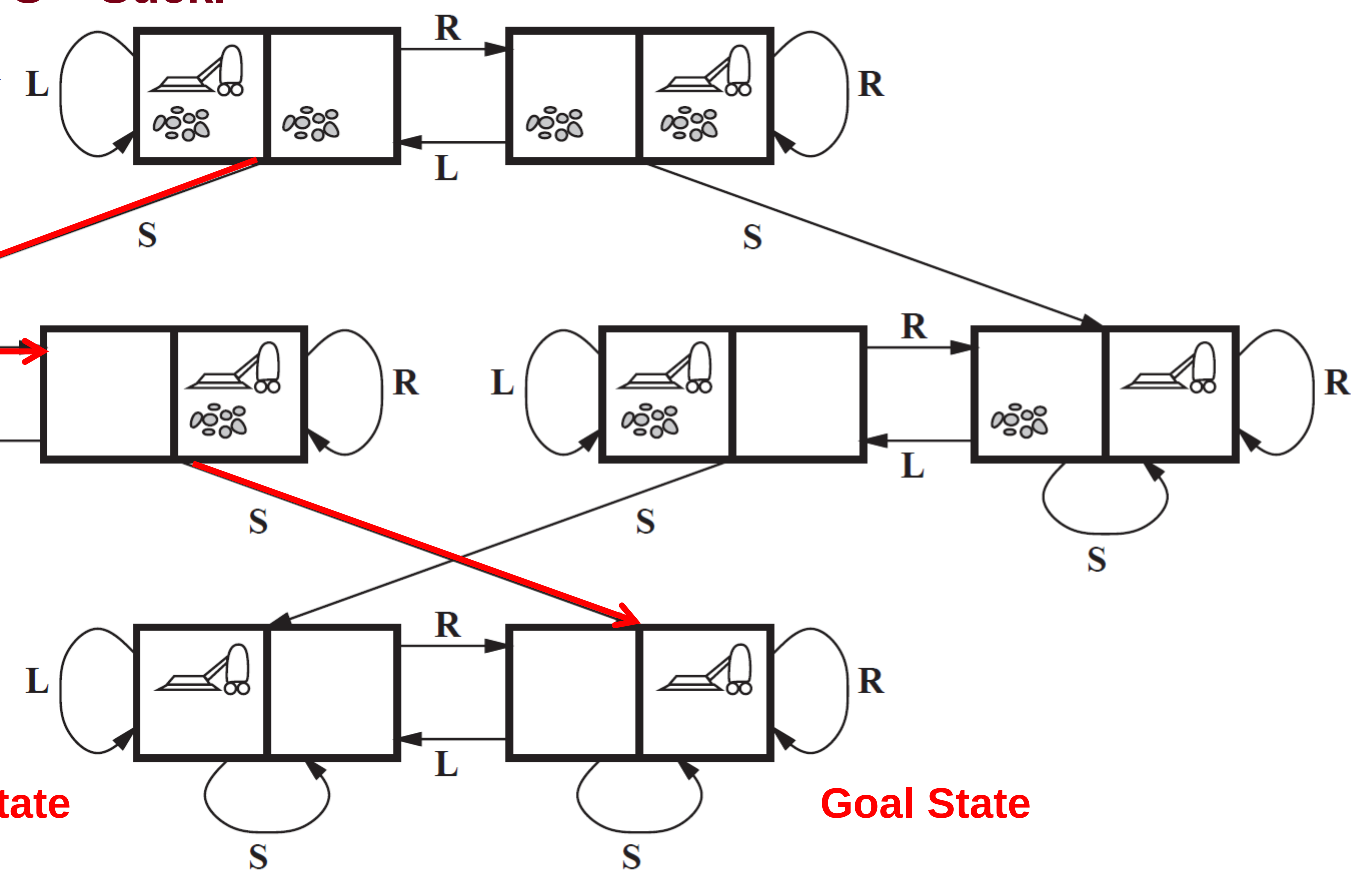
Initial State



Example: Vacuum-Cleaner World

- The state space graph for the vacuum world.
  - Each node represents a state and each edge denotes an action
- L = Left, R = Right, S = Suck.

Initial State



Goal State

Goal State

Solution: S -> R -> S

Example: Farmer Problem

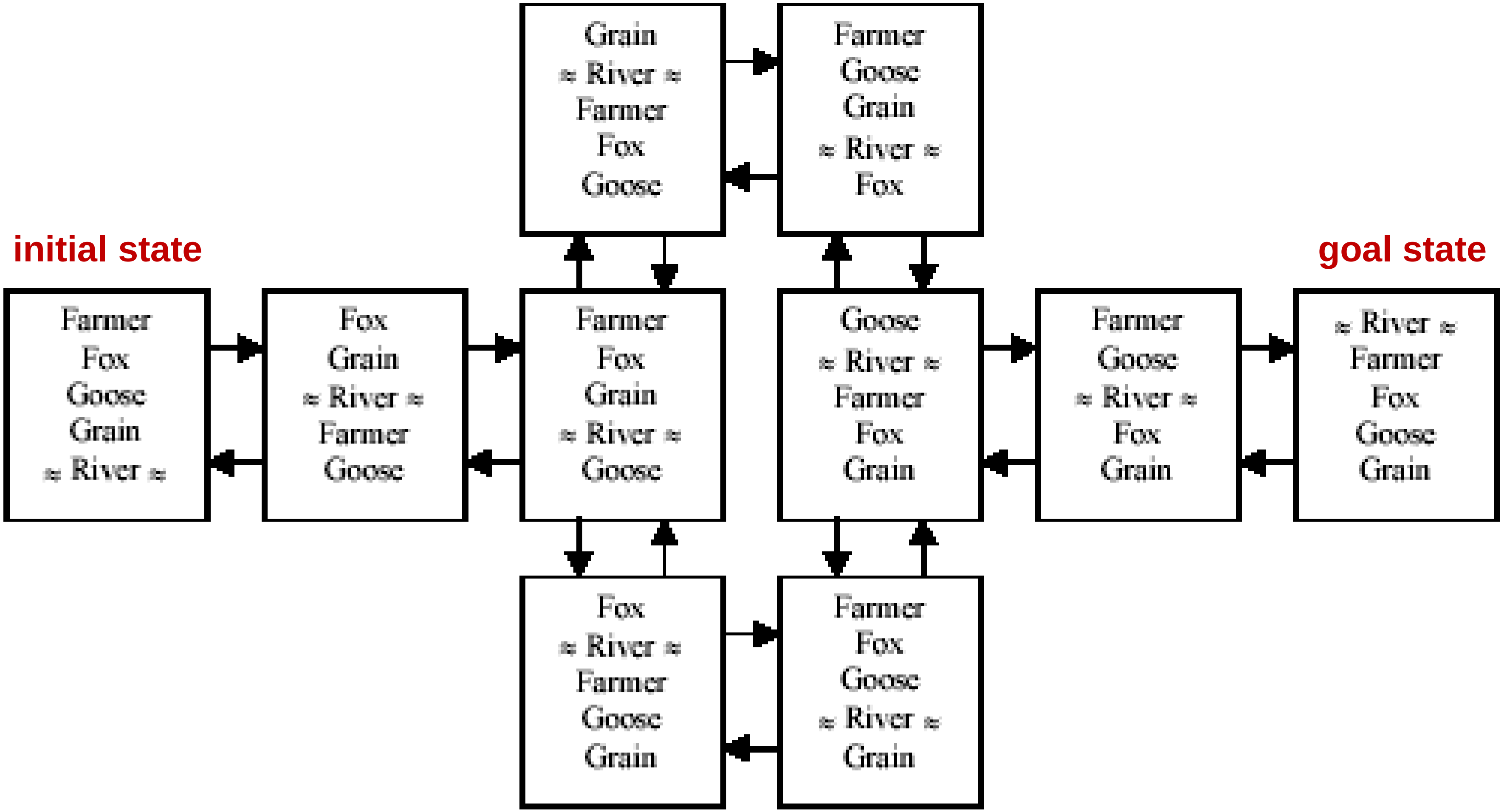
- Good Representations to a Problem are the Key to Good Problem Solving
- Example
  - Items: Farmer, Fox, Goose, and Grain
  - Farmer wants to cross the river with all items by driving a boat
    - Fox will eat goose
    - Goose will eat grain
    - Boat will take max of two and one is the farmer

- States?
- Initial State?
- Actions?
- Transition model?
- Goal test?
- Path cost?

Problem Formulation for the Farmer Problem

- **States**
  - The locations of the farmer, fox, goose, and grain
- **Initial State**
  - Farmer, fox, goose, and grain are at one side of the river
- **Actions**
  - Farmer moves along with one of the three (fox, goose, grain) to the other side of the river
- **Transition model**
  - Returns a resulting state after an action
- **Goal test**
  - Farmer, fox, goose, and grain are at the other side of the river
- **Path cost**
  - Number of crossing

State Space


initial state
goal state
Farmer Fox Goose Grain ≈ River ≈
Fox Grain ≈ River ≈ Farmer Goose
Farmer Fox Grain ≈ River ≈ Goose
Grain ≈ River ≈ Farmer Fox Goose
Farmer Goose Grain ≈ River ≈ Fox
Fox ≈ River ≈ Farmer Goose Grain
Farmer Fox Goose ≈ River ≈ Grain
Goose ≈ River ≈ Farmer Fox Grain
Farmer Goose ≈ River ≈ Fox Grain
≈ River ≈ Farmer Fox Goose Grain

Exercise: Missionaries and Cannibals Problem

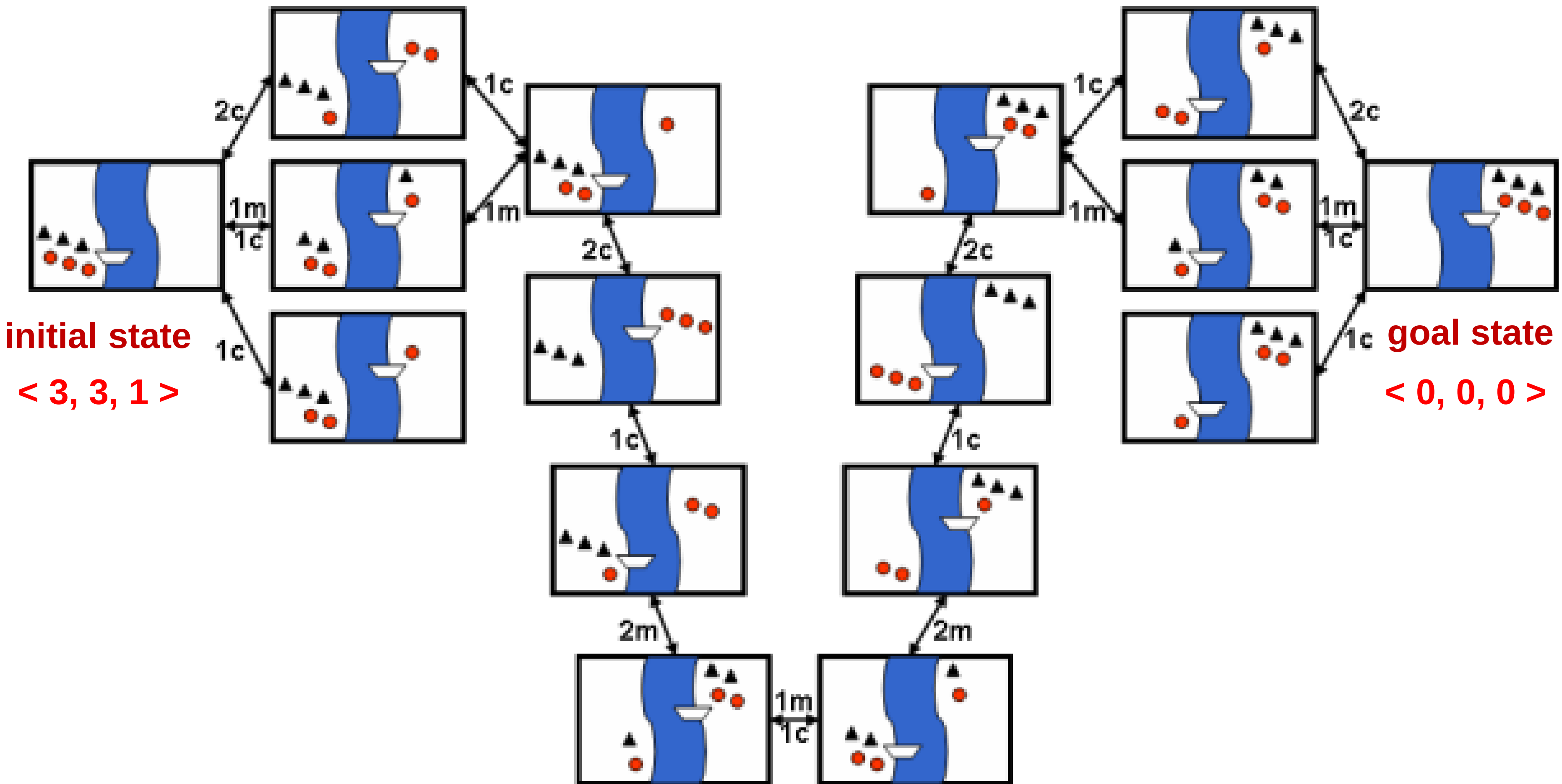
- Example
  - People: 3 missionaries (black triangles) and 3 cannibals (red dots)
  - All 6 people want to cross the river by driving a boat
    - The missionaries in one place cannot be outnumbered by cannibals (otherwise the cannibals would eat the missionaries)
    - Boat will take one or two people

- States?
- Initial State?
- Actions?
- Transition model?
- Goal test?
- Path cost?

Problem Formulation

- **States**
  - <m, c, b> representing the # of missionaries and the # of cannibals, and the # of the boat at the left side of the river
- **Initial State**
  - < 3, 3, 1 >
- **Actions**
  - Take 1 missionary, 1 cannibal, 2 missionaries, 2 cannibals, or 1 missionary and 1 cannibal across the river
- **Transition model**
  - Returns a resulting state after an action
- **Goal test**
  - < 0, 0, 0 >
- **Path cost**
  - Number of crossing

State Space



initial state
< 3, 3, 1 >
goal state
< 0, 0, 0 >
2c
1m
1c
1c
1c
1m
2c
1c
2m
1m
1c
2m
1c
2c
1m
1c
2c
1m
1c
1c

Representation Principle

Once a problem is formulated using an appropriate representation, the problem is almost solved.

Summary

- Problem-solving Agent
- Goal Formulation
- Problem Formulation

What I want you to do

- **Review Chapter 3**