Applied AI

Lecture 2

Agents and Environments

Dr Artie Basukoski

[some slides adapted from Artificial Intelligence: A Modern Approach, Russel and Norvigl]

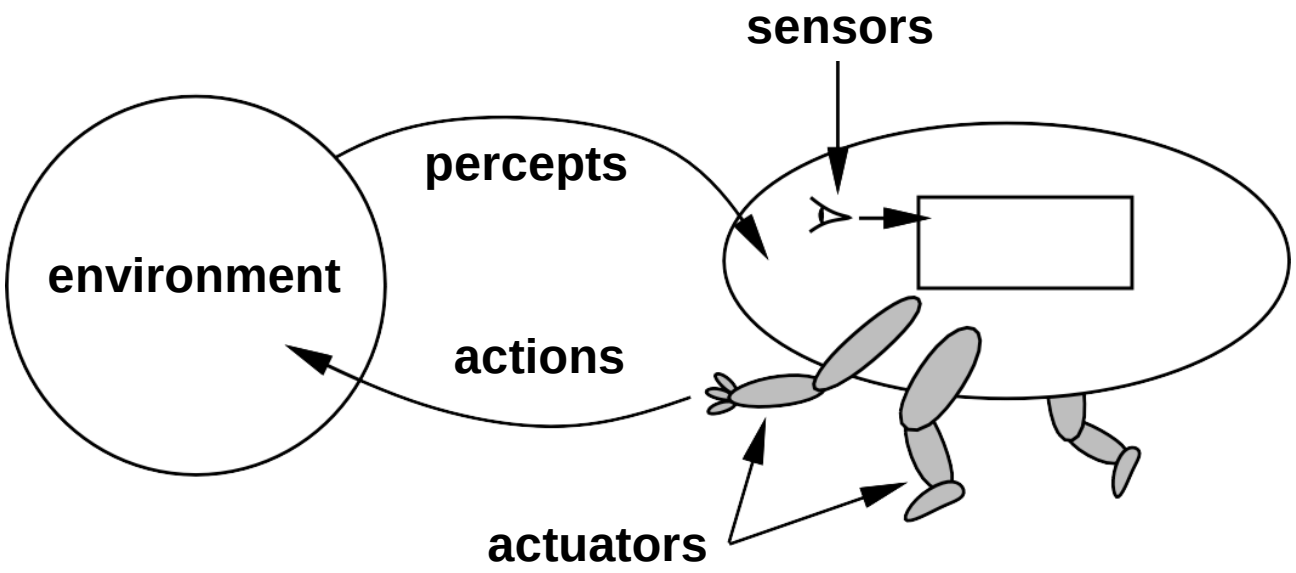
Agenda

- Agents and environments
- Types of agents
- Recent trends
- Representing problems
- Selecting a state space

Intelligent agents

- Identify the concept of an intelligent agent.
- Develop a small set of design principles for building successful agents.
- Agents should be rational – one that does the right thing.
- Behaviour depends on the environment and the goals that we define for the agents.
- We define a number of basic agent designs.
- Agent = Architecture + Program

Agents and environments



Agents include humans, robots, softbots, thermostats, etc.

The agent function maps from percept histories to actions:

$$f : P^* \rightarrow A$$

The agent program runs on the physical architecture to produce f

What is an Intelligent Agent

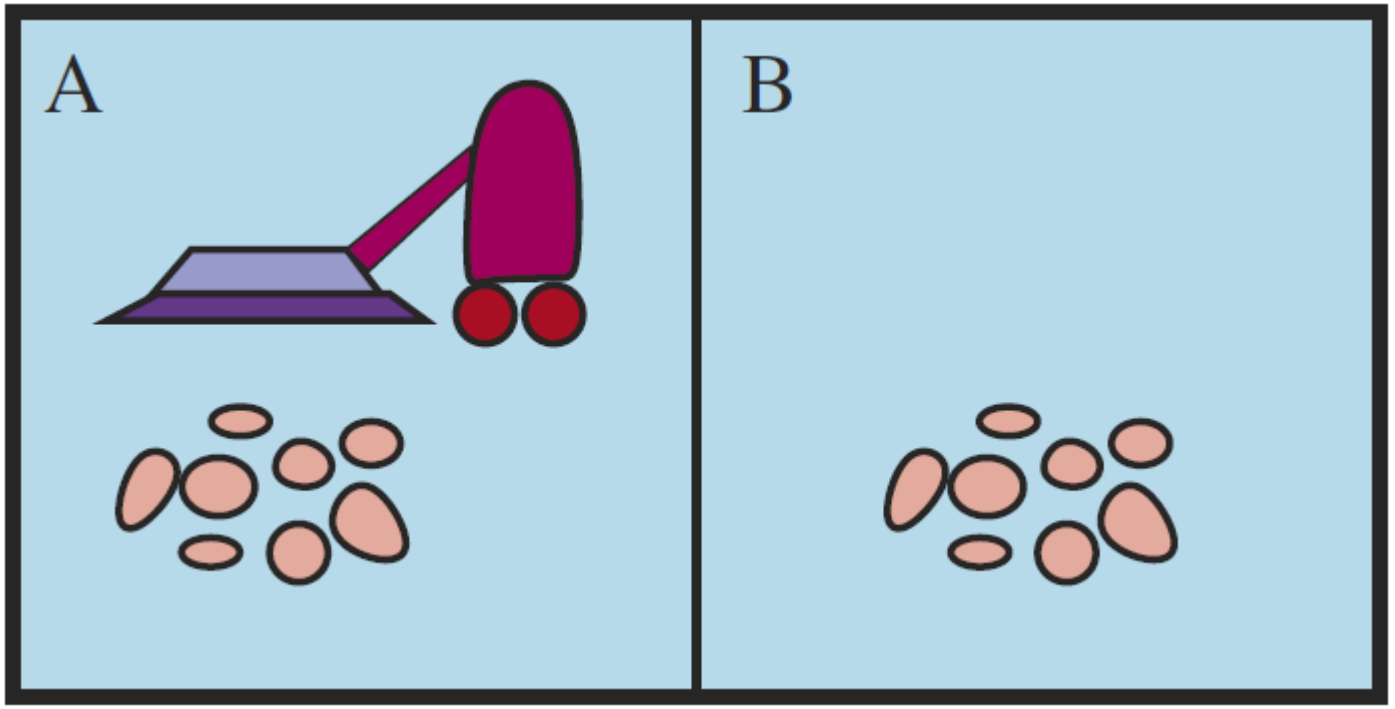
Characteristics

- Autonomy
- Reactivity
- Proactiveness
- Social Ability?

Examples

- Robotics
- Software Agents
- Virtual Assistants

Vacuum-cleaner world



Percepts: location and contents, e.g., [*A, Dirty*]

Actions: *Left*, *Right*, *Suck*, *NoOp*

A vacuum-cleaner agent

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*
.	.

```
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
```

What is the `right` function?
Can it be implemented in a small agent program?

The key challenge for AI is to find out how to write programs that, to the extent possible, produce **rational** behaviour from a smallish program rather than from a vast table.

Rationality

- What is rational at any given time depends on four things:
  - The performance measure that defines the criteria of success
  - The agents prior knowledge of the environment
  - The actions that the agent can perform
  - The agents percept subsequence to date
- This leads to a definition of a 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 the agent has.

Rationality

Fixed performance measure evaluates the environment sequence

- one point per square cleaned up in time T?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rational $=/=$ omniscient

-percepts may not supply all relevant information

Rational $=/=$ clairvoyant

-action outcomes may not be as expected

Hence, rational $=/=$ successful

Rational $\Rightarrow$ exploration, learning, autonomy

Polleverywhere

When poll is active respond at **PollEv.com/artieb757** Send **artieb757** to **07480 781235**

- **Which of the following is a good performance measure for a rational vacuum cleaning agent**
- Maximise the number of squares visited
- **0%**
- Maximise the average amount of dirt on all the squares
- **0%**
- Maximize the number of clean squares while minimizing the amount of energy used and time taken.
- **0%**

PEAS

To design a rational agent, we must specify the task environment

Consider, e.g., the task of designing an automated taxi:

Performance measure: safety, destination, profits, legality, comfort, . . .

Environment: streets/freeways, traffic, pedestrians, weather, . . .

Actuators: steering, accelerator, brake, horn, speaker/display, . . .

Sensors: video, accelerometers, gauges, engine sensors, keyboard, GPS, . . .

Internet shopping agent

Performance measure: ??

Environment: ??

Actuators: ??

Sensors: ??

Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: current and future WWW sites, vendors, shippers

Actuators: display to user, follow URL, fill in form

Sensors: HTML pages (text, graphics, scripts)

ChatGPT agent assistant agent

Performance measure: ??

Environment: ??

Actuators: ??

Sensors: ??

ChatGPT assistant agent

Performance measure: accuracy, relevance, response time, user satisfaction

Environment: current and future WWW sites

Actuators: display to user, plugins allow to interact with databases, the web, etc

Sensors: text, graphics, voice

Environment types

- **Fully Observable vs. Partially Observable**
  - **Fully Observable**: In this environment, an agent can see all the relevant aspects at once. For example, a chess game is fully observable because you can see the entire board.
  - **Partially Observable**: Here, the agent can't see everything. For instance, in a card game like Poker, you can't see the other players' cards.
- **Deterministic vs. Stochastic**
  - **Deterministic**: The next state is completely determined by the current state and the action taken. Chess is deterministic.
  - **Stochastic**: The next state is not entirely predictable, like the stock market.
- **Episodic vs. Sequential**
  - **Episodic**: The agent's experience is divided into episodes. Each episode is independent of others, like in a quiz game.
  - **Sequential**: Decisions are interdependent, and one decision leads to another, like in a game of soccer.
- **Static vs. Dynamic**
  - **Static**: The environment doesn't change while the agent is deliberating. Chess is static.
  - **Dynamic**: The environment can change even if the agent does nothing, like in a self-driving car scenario.
- **Discrete vs. Continuous**
  - **Discrete**: There are a finite number of actions and states, like in a tic-tac-toe game.
  - **Continuous**: The range of possible states and actions is infinite, like steering a car.
- **Single-Agent vs. Multi-Agent**
  - **Single-Agent**: Only one agent is acting in the environment, like a Sudoku solver.
  - **Multi-Agent**: Multiple agents are interacting, like in a stock trading system

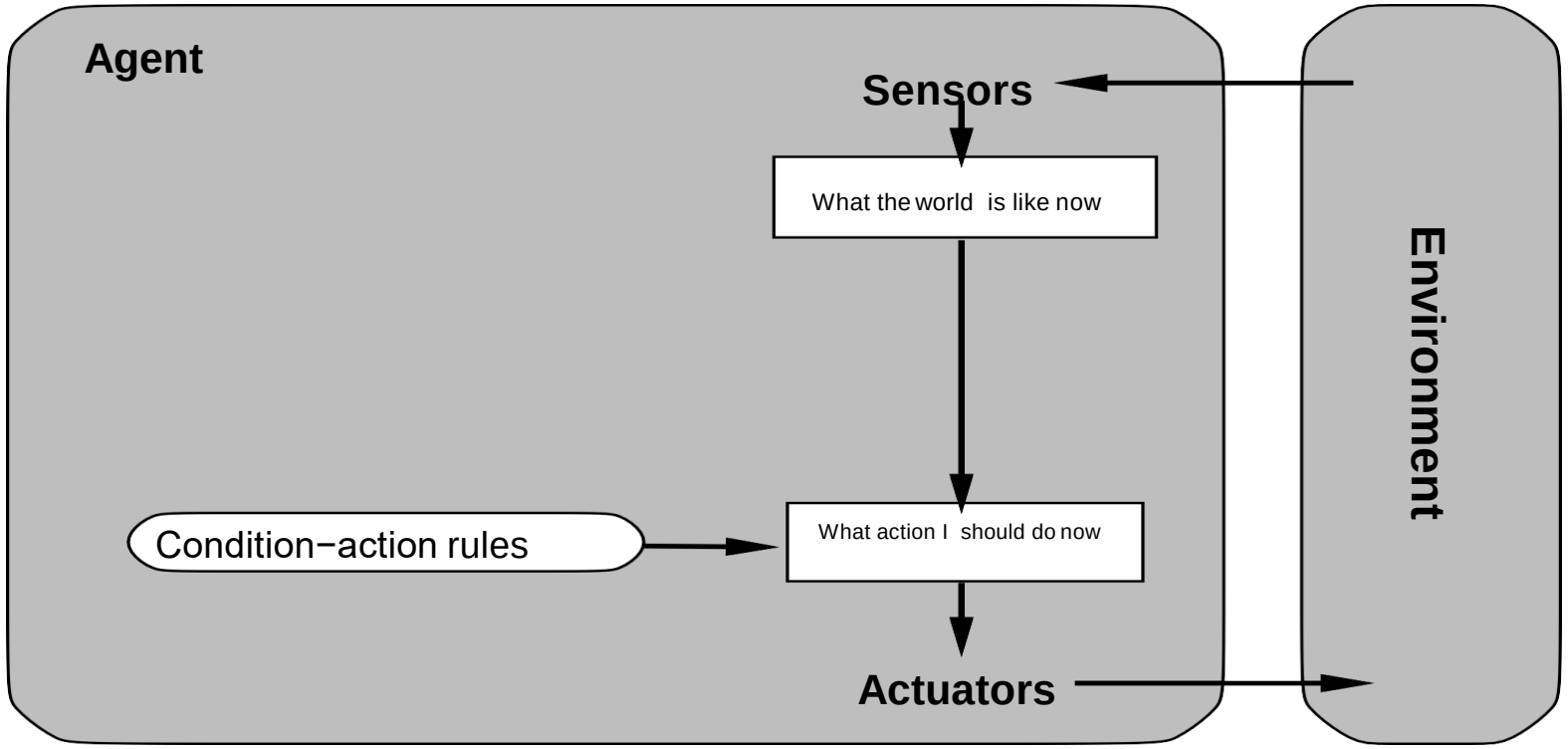
Environment types

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

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

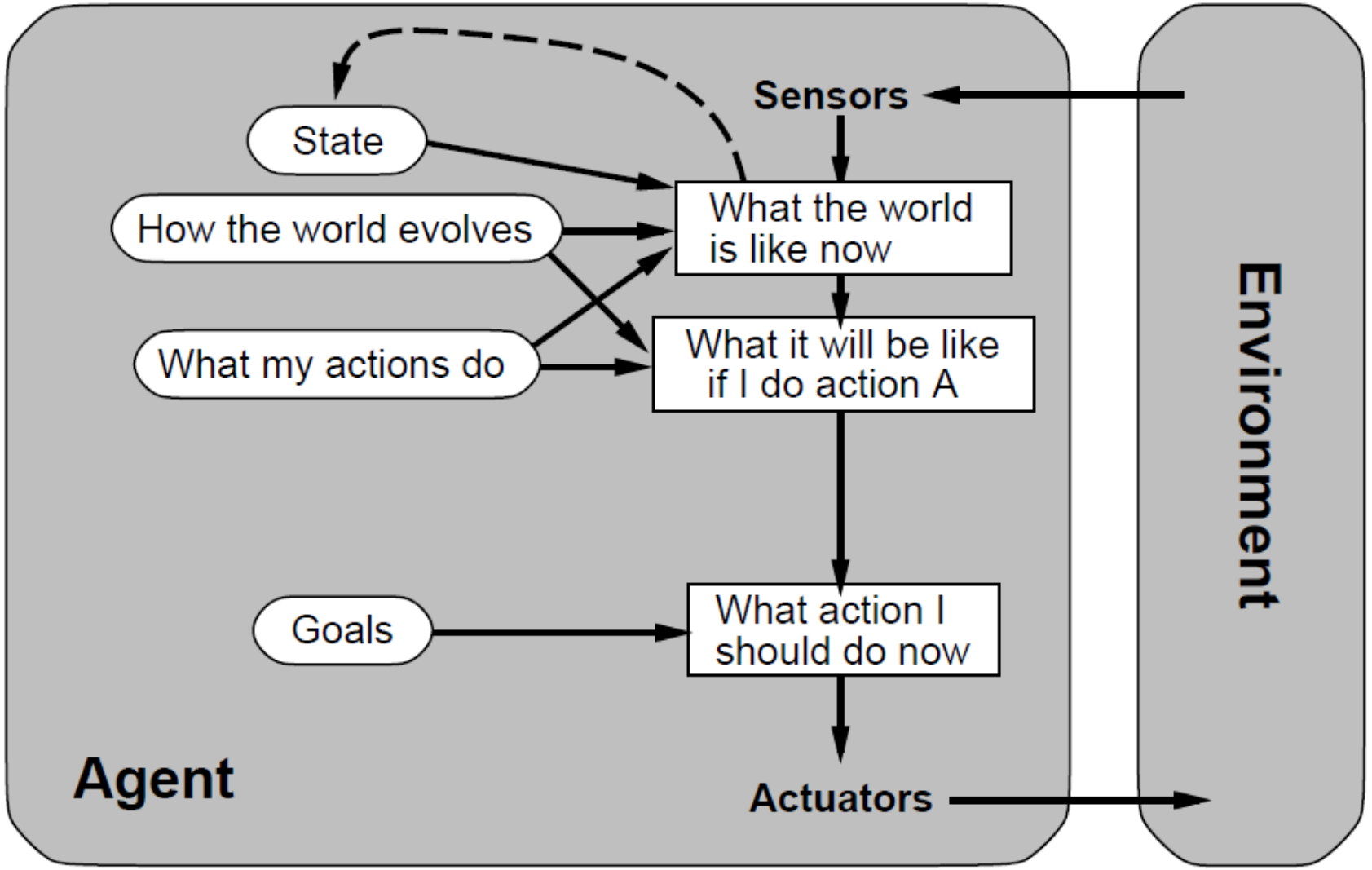
Simple reflex agents




```
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
```

Goal-based agents



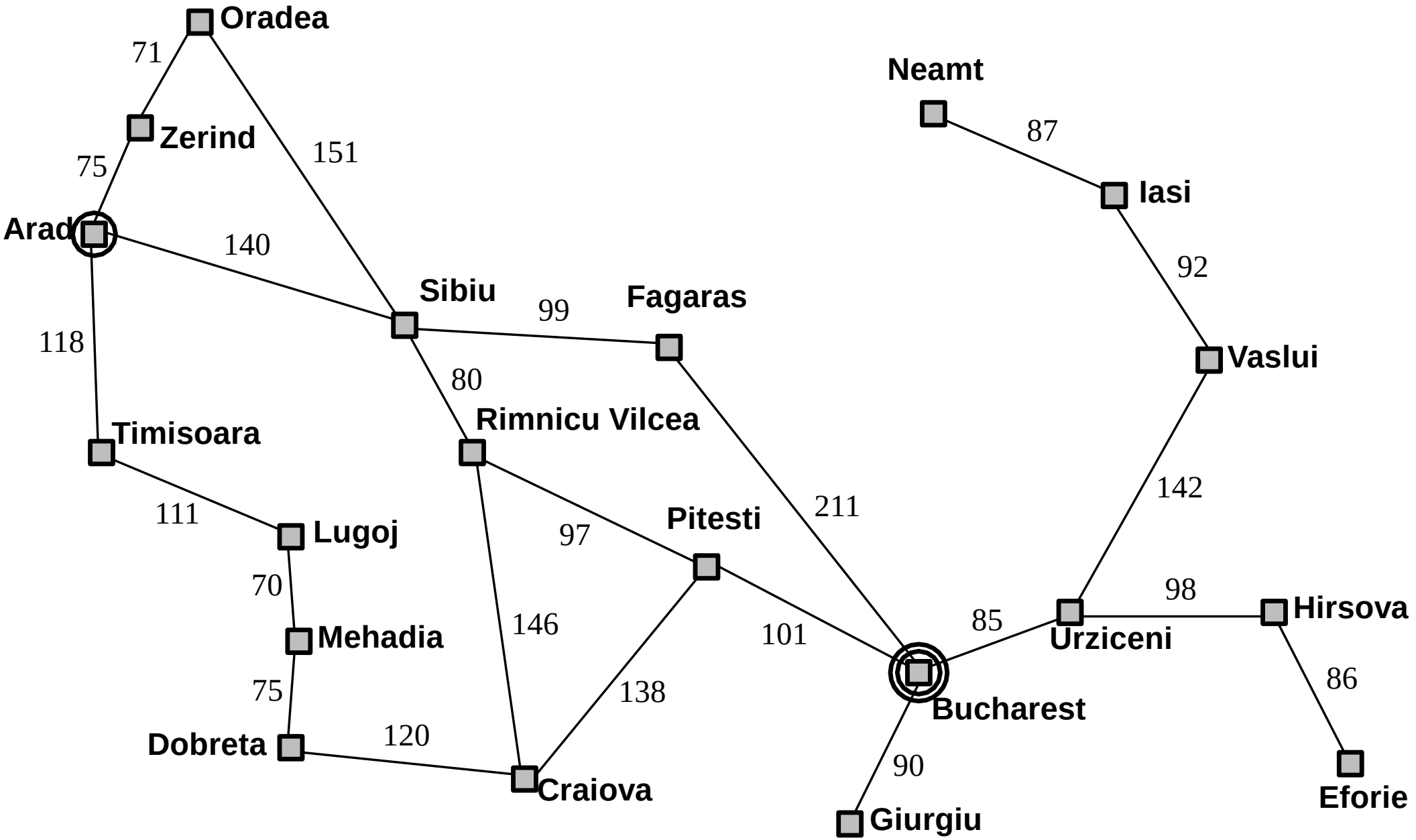
Example: Romania

On holiday in Romania; currently in Arad.
Flight leaves tomorrow from Bucharest

Formulate goal:
be in Bucharest

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

Find solution:
sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest


Oradea
71
Neamt
Zerind
87
151
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
211
111
Pitesti
Lugoj
97
98
70
Hirsova
146
85
Mehadia
101
Urziceni
75
86
138
Bucharest
Dobreta
120
90
Craiova
Eforie
Giurgiu

Problem formulation

A problem is defined by four items:

initial state e.g., "at Arad"

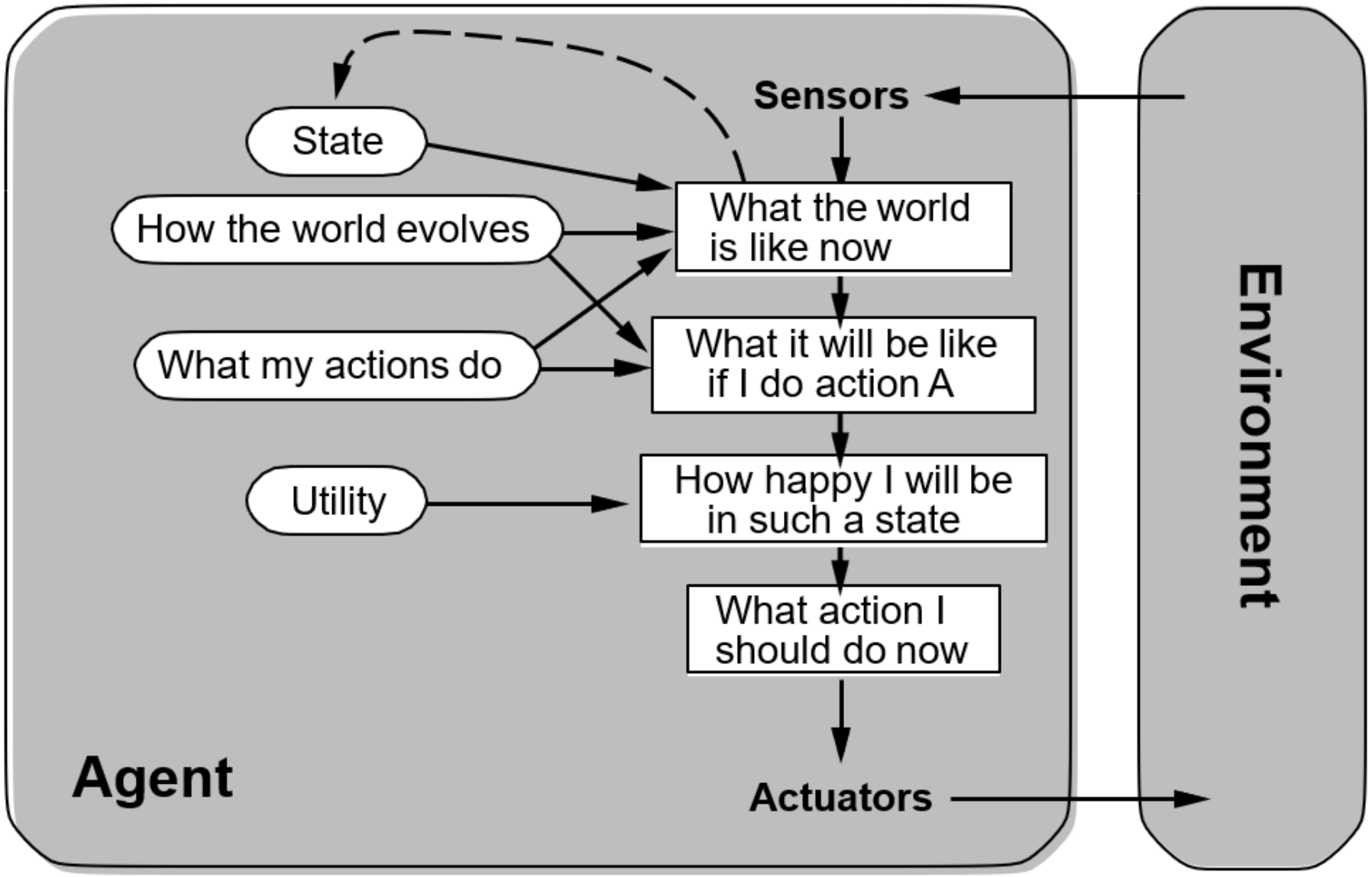
successor function $S(x)$ = set of action–state pairs
e.g., $S(Arad) = \{(Arad \rightarrow Zerind, Zerind), \ldots\}$

goal test, can be
explicit, e.g., x = "at Bucharest" implicit, e.g., $NoDirt(x)$

path cost (additive)
e.g., sum of distances, number of actions executed, etc.
$c(x, a, y)$ is the step cost, assumed to be ≥ 0
Cost of applying action 'a' in state 'x' to reach state 'y'

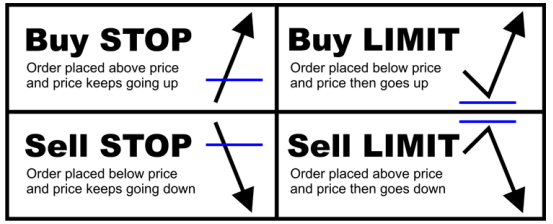
A solution is a sequence of actions from initial to a goal state

Utility based agents

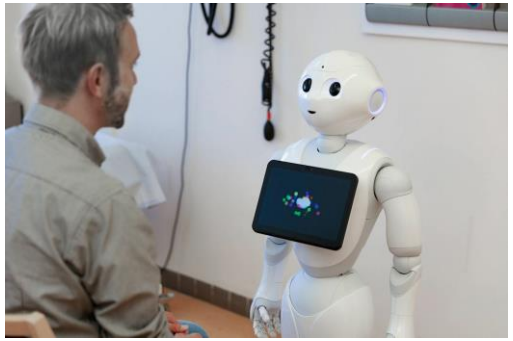


Examples of Utility Based Agents

- Autonomous Vehicles
  - Use utility functions to determine best driving by balancing factors like speed, safety and fuel efficiency. U = w1 * Safety + w2 * TravelTime + w3 * FuelEfficiency + w4 * PassengerComfort
- Financial Trading Systems
  - Weigh potential profits against risks to make investment decisions. U = ExpectedReturn – RiskAversion * Risk



- Recommendation Systems
  - Use user preferences and past behaviour to suggest products, movies to maximise user satisfaction. U = PredictedUserRating - w * ItemPopularity
- Healthcare Management
  - Optimise resource allocation, patient scheduling and treatment plans to improve overall healthcare outcomes.
    U = w1 * PatientOutcomes + w2 * ResourceUtilization + w3 * WaitingTimes

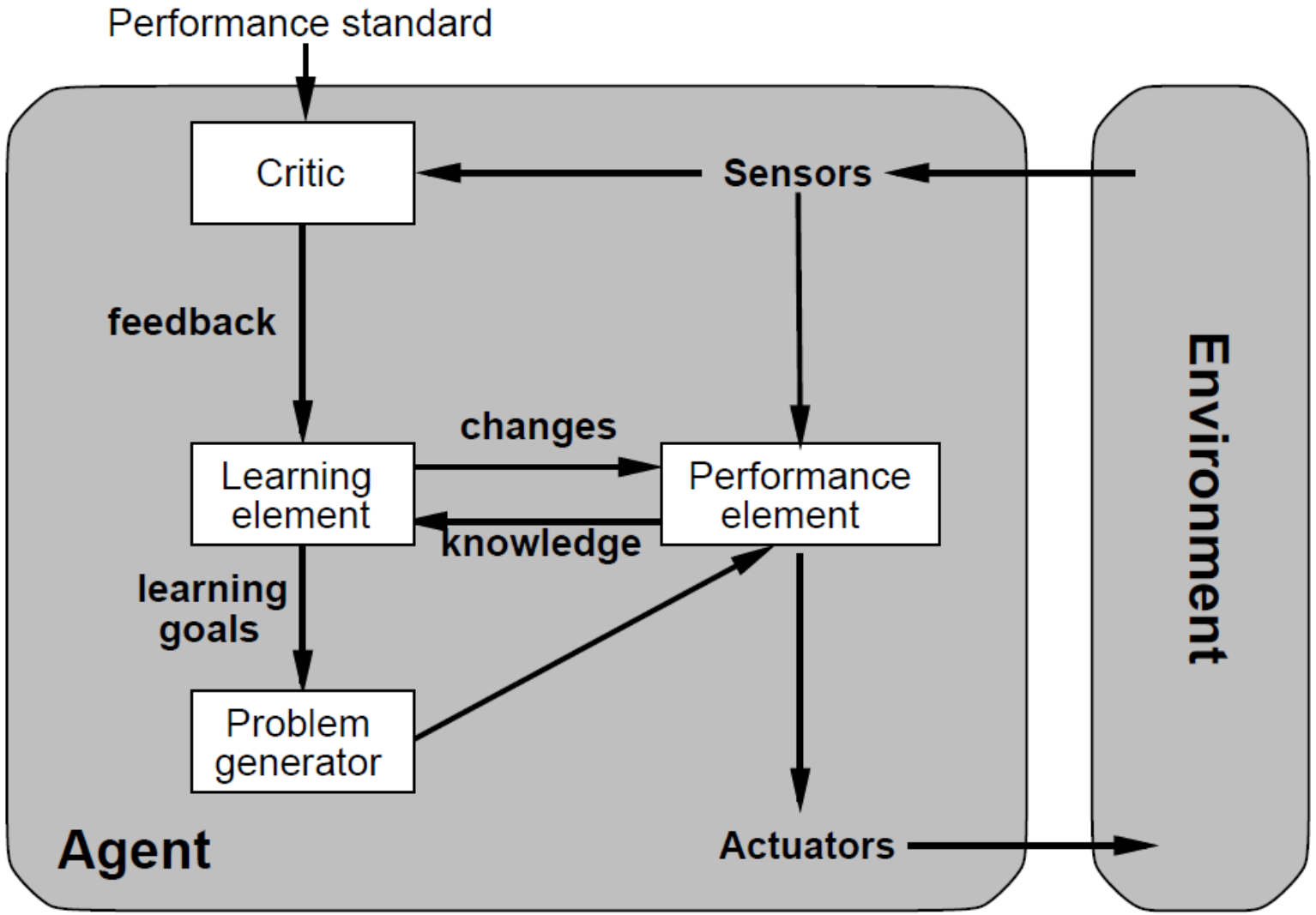


Polleverywhere

When poll is active respond at **PollEv.com/artieb757** Send **artieb757** to **07480 781235**

- Q: A rational agent:
- A) Always achieves the optimal outcome.
- B) Maximises expected performance based on its knowledge.
- C) Has complete information about the environment.
- D) Is incapable of making mistakes

Learning Agents



Examples of Learning Agents

Recommendation Systems (Amazon)

- Performance element
  - Suggests movies, books, or products based on user preferences
- Learning element
  - Learns from user feedback, e.g. ratings, clicks, purchase history
- Critic
  - Evaluates the quality of recommendations based on user engagement and satisfaction e.g. engaging with recommendation: movie, product
- Problem Generator
  - Introduce new or less commonly recommended items to gather data on user interests and preferences in less explored areas

Game Playing AI (AlphaGo)

- Performance element
  - Plays Go by making decisions on moves
- Learning element
  - Learns from past games (reinforcement learning) and improves its strategies
- Critic
  - Compares moves against winning strategies and evaluates performance based on game outcomes
- Problem Generator
  - Experiments with alternative moves or strategies in less explored parts of the game tree to discover new tactics

Problem types

Deterministic, fully observable $\Rightarrow$ single-state problem

Agent knows exactly which state it will be in; solution is a sequence

Non-observable $\Rightarrow$ sensorless problem

Agent may have no idea where it is; solution (if any) is a sequence

Nondeterministic and/or partially observable $\Rightarrow$ contingency problem

percepts provide new information about current state

solution is a contingent plan or a policy

often interleave search, execution

Unknown state space $\Rightarrow$ exploration problem ("online")

Selecting a state space

A set of possible states that the environment can be in.

Real world is absurdly complex

⇒ state space must be abstracted for problem solving

(Abstract) state = set of relevant real states

(Abstract) action = complex combination of real actions

e.g., "Arad → Zerind" represents a complex set of possible routes, detours, rest stops, etc.

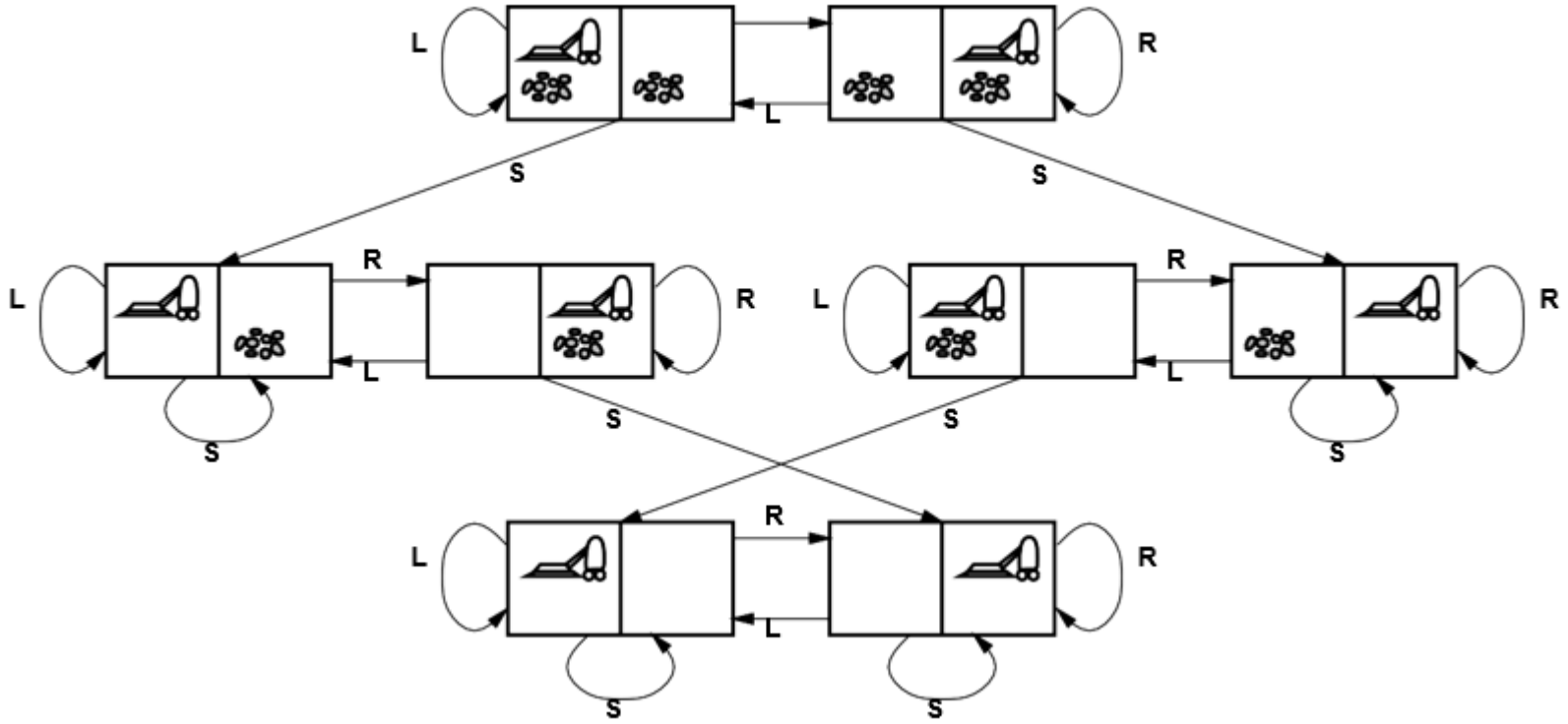
For guaranteed realizability, any real state "in Arad" must get to some real state "in Zerind"

(Abstract) solution =

set of real paths that are solutions in the real world

Each abstract action should be "easier" than the original problem!

Example: Vacuum world state space graph



states: dirt and robot locations (ignore dirt amounts etc.)

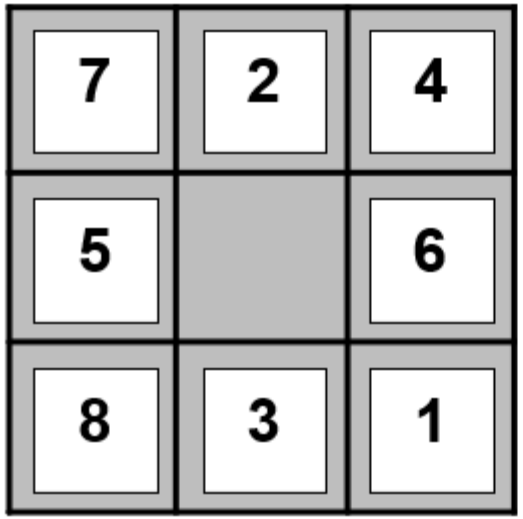
actions: *Left*, *Right*, *Suck*, *NoOp*

goal test: no dirt

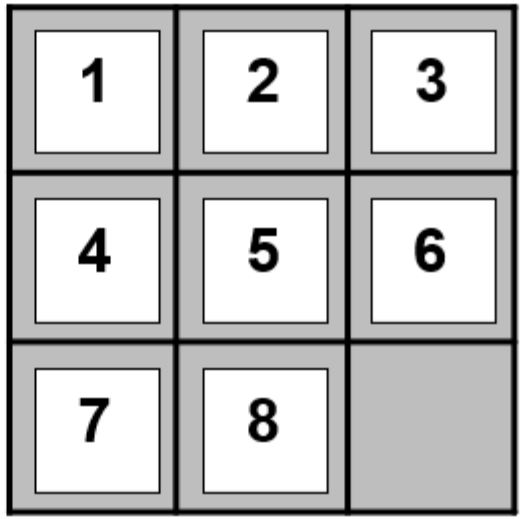
path cost: 1 per action (0 for *NoOp*)

Example: The 8 puzzle

What are the possible next states?



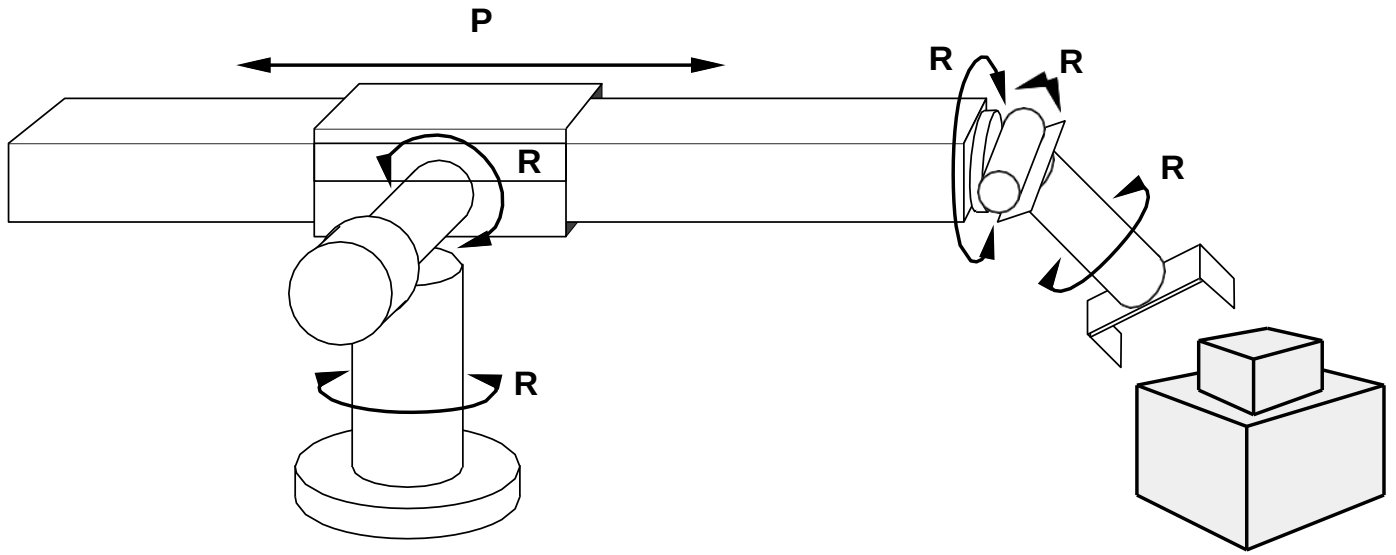
Start State



Goal State

states: integer locations of tiles (ignore intermediate positions)
actions: move blank left, right, up, down (ignore unjamming etc.)
goal test = goal state (given)
path cost: 1 per move

Example: Robotic assembly



states: real-valued coordinates of robot joint angles
parts of the object to be assembled

actions: continuous motions of robot joints

goal test: complete assembly with no robot included!

path cost: time to execute

Polleverywhere

When poll is active respond at **PollEv.com/artieb757** Send **artieb757** to **07480 781235**

Q: What does the successor function describe in a problem formulation?

a) Possible goal states

b) Actions the agent can take

c) Path cost calculation

d) Initial state of the problem

Current research and future directions

- Active research areas - multi-agent systems, human-agent interaction, agent learning
- Open problems - scalability, robustness, commonsense reasoning
- Promising future applications - space exploration, nanotechnology, human augmentation

Multi-Agent Systems

Systems comprising multiple interacting intelligent agents, each with its own goals and capabilities.

Applications

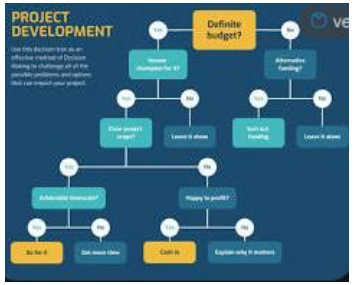
- Traffic management: Coordinating autonomous vehicles to optimize traffic flow and reduce congestion
- Smart grids: Balancing energy production and consumption across a network of distributed energy resources
- Robotics: Collaborative robots working together to complete complex tasks in manufacturing or logistics
- Gaming: Creating realistic and engaging virtual environments with multiple AI-controlled characters

Challenges

- Communication and coordination: Ensuring effective communication and coordination among agents with potentially conflicting goals
- Decision-making: Developing algorithms for agents to make decisions in a dynamic and uncertain environment
- Learning: Enabling agents to learn from their interactions with other agents and the environment

Ethical and societal Implications

- Algorithmic bias and fairness
- Transparency and explainability of agent decisions
- Privacy and security concerns with agent systems
- Accountability and legal frameworks
- Societal impact and workforce displacement



Agents summary

Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances $f : P^* \rightarrow A$

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance

Agent programs implement (some) agent functions

PEAS descriptions define task environments

Environments are categorized along several dimensions:

observable? deterministic? episodic? static? discrete? single-agent?

Several basic agent architectures exist:

reflex, reflex with state, goal-based, utility-based, learning

Questions or comments