EC-350 AI and Decision Support Systems

Week 2
Intelligent Agents

Dr. Arslan Shaukat

NUST

Acknowledgement: Lecture slides material from Stuart Russell

Rationality

- What is rational at given time depends upon four things
  - *The performance measure that defines criterion for success*
  - *The agent's prior knowledge of the environment*
  - *The actions that agent can perform*
  - *The agent's percept sequence to-date*

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.
- Vacuum-cleaner agent?

Vacuum-Cleaner Agent

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known a priori but the dirt distribution and the initial location of the agent are not.
- The only available actions are Left, Right, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.

Omniscience

- An omniscience agent knows the actual outcomes of its actions and can act accordingly
- Omniscience is impossible in reality
- Road crossing example

Omniscience, Perfection

- Rationality is NOT the same as Perfection
- Rationality maximizes expected performance
- Perfection maximizes actual performance

Exploration, Learning

- Doing actions in order to modify future percepts, sometimes called *information gathering*, is an important part of rationality.
- It performs such actions to increase its perception
- This is also called Exploration
  - *as taken by vacuum cleaner agent*
- A rational agent not only gather information but also learn as much as possible from what it perceives

Agent Autonomy

- The capacity to compensate for partial or incorrect prior knowledge by learning
- An agent is called autonomous if its behavior is determined by its own experience (with ability to learn and adopt)
- A truly autonomous agent should be able to operate successfully in a wide variety of environments

Task Environment

- *Problems* to which rational agents are the *solutions*.
- PEAS
  - *P – Performance Measure*
  - *E – Environment*
  - *A – Actuators*
  - *S – Sensors*
- First step in designing an agent must be to define the task environment

PEAS - Example

- Automated Taxi Driver Agent
  - *Performance measure: Safe, correct destination, minimizing fuel consumption, min wear and tear, fast, legal, comfortable trip, maximize profit*
  - *Environment: Roads, other traffic, pedestrians, customers, stray animals, police cars, signals, potholes*
  - *Actuators: Steering, accelerator, brake, signal, horn, display*
  - *Sensors: TV Cameras, sonar, speedometer, accelerometer, GPS, odometer, engine sensors, keyboard, mic*

Agent Type and PEAS

Agent Type	Performance Measures	Environment	Actuators	Sensors
Medical Diagnostic	Healthy patients, minimize costs,	Patients, hospital, staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patients' answers
Satellite image analysis system	Correct image characterization	Downlink from orbiting satellite	Display categorization of scene	Color pixel arrays
Part picking robot	Percentage of parts in correct bins	Conveyor belt with parts, bins	Jointed arm and hand	Cameras, joint angle sensors
Refinery controller	Maximize purity, yield safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors

Environment Types

- Fully observable vs. partially observable:
  - *An agent's sensors give it access to the complete state of the environment at each point in time.*
  - *Partially observable because of noisy and inaccurate sensors*
- Deterministic vs. stochastic:
  - *The next state of the environment is completely determined by the current state and the action executed by the agent.*
  - *If the environment is partially observable, then it could appear to be stochastic.*
- Episodic vs. sequential:
  - *The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action)*
  - *choice of action in each episode depends on the episode itself.*

Environment Types

- Static vs. dynamic:
  - *The environment is unchanged while an agent is deliberating.*
  - *The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does*
- Discrete vs. continuous:
  - *A limited number of distinct states*
  - *Clearly defined percepts and actions.*
- Single agent vs. multiagent:
  - *An agent operating by itself in an environment.*
  - *Chess is a* ***competitive*** *multiagent environment.*
  - *Taxi-driving is a partially* ***cooperative*** *multiagent environment.*

Examples of Task Environments

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
Interactive. English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Structure of Agents

- Agent Program
  - *Implements the agent function— the mapping from percepts to actions*
  - *Runs on some sort of computing device, we may call it the architecture*
  - *It may be a plain computer or may include special hardware*

Architecture

- The architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.
- The relationship
  - *Agent = architecture + program*

Table Driven Agent

function TABLE-DRIVEN-AGENT(*percept*) **returns** an action
  static: *percepts*, a sequence, initially empty
    table, a table of actions, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← LOOKUP (*percepts*, *table*)
  return *action*

Figure: The TABLE DRIVEN AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

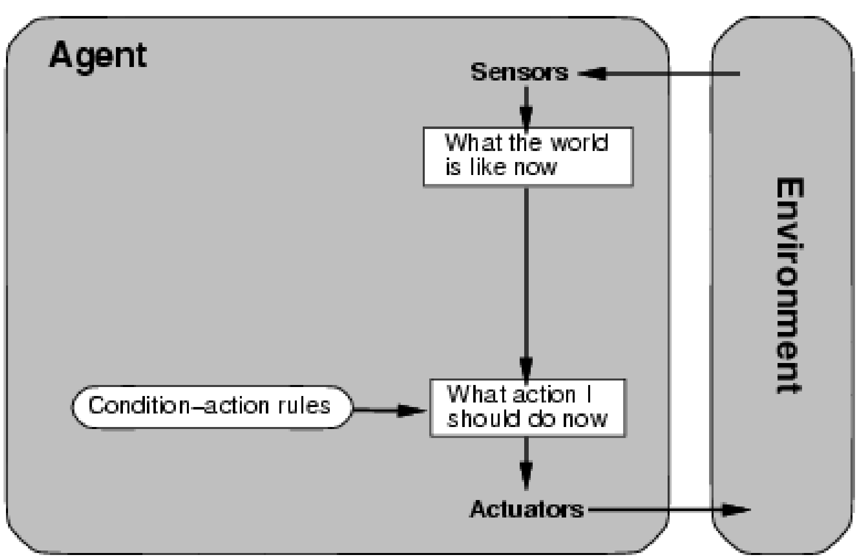
Table Driven Agent

- Disadvantages:
  - *(a) no physical agent in this universe will have the space to store the table,*
  - *(b) the designer would not have time to create the table,*
  - *(c) no agent could ever learn all the right table entries from its experience, and*
  - *(d) even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries.*

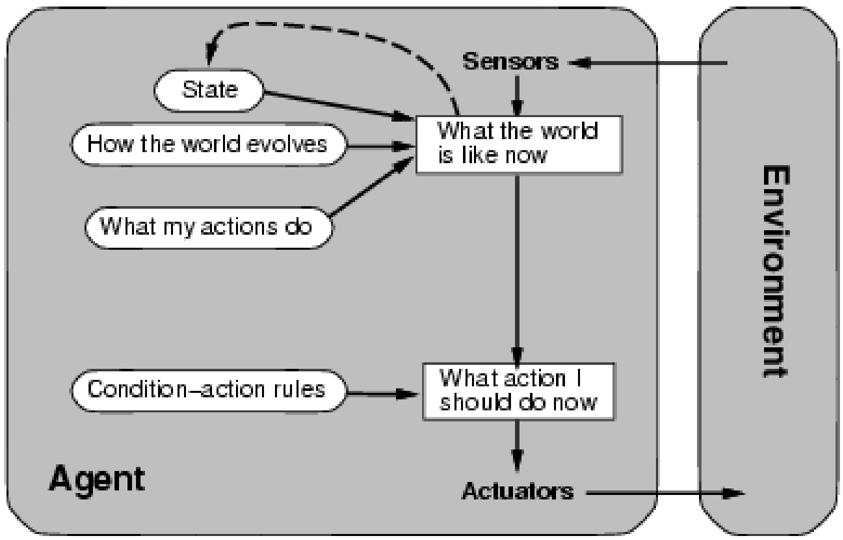
Types of Agents

- 5 types:
- Simple reflex agents
  - *respond directly to percepts*
- Model-based reflex agents
  - *maintain internal state to track aspects of the world that are not evident in the current percept.*
- Goal-based agents
  - *act to achieve their goals, and*
- Utility-based agents
  - *try to maximize their own expected "happiness" or utility*
  - *Utility function*
- Learning Agents
  - *Agents can improve their performance through learning*

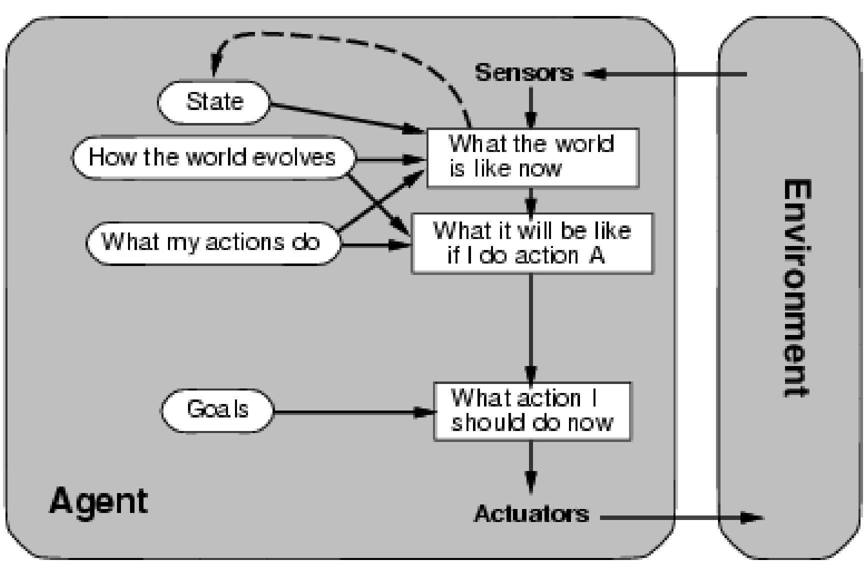
Simple Reflex Agents



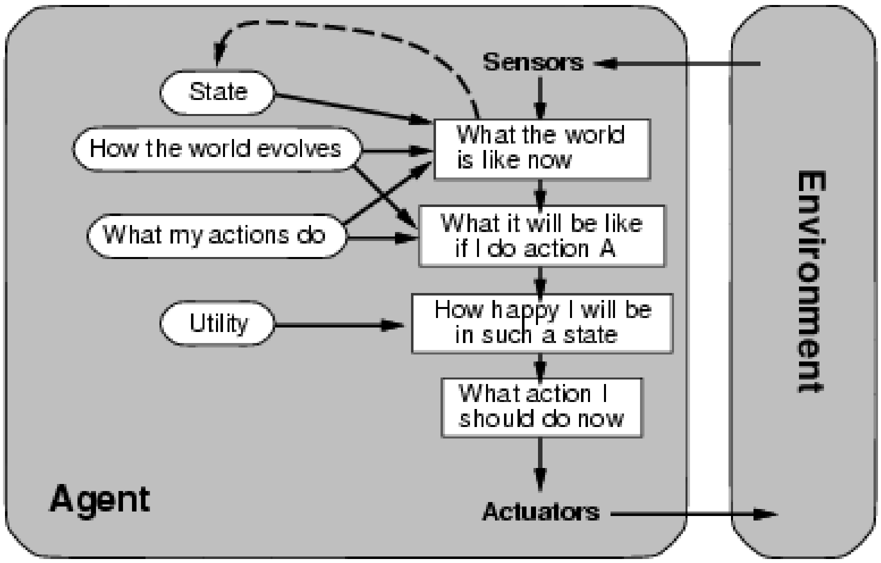
Model-based Reflex Agents



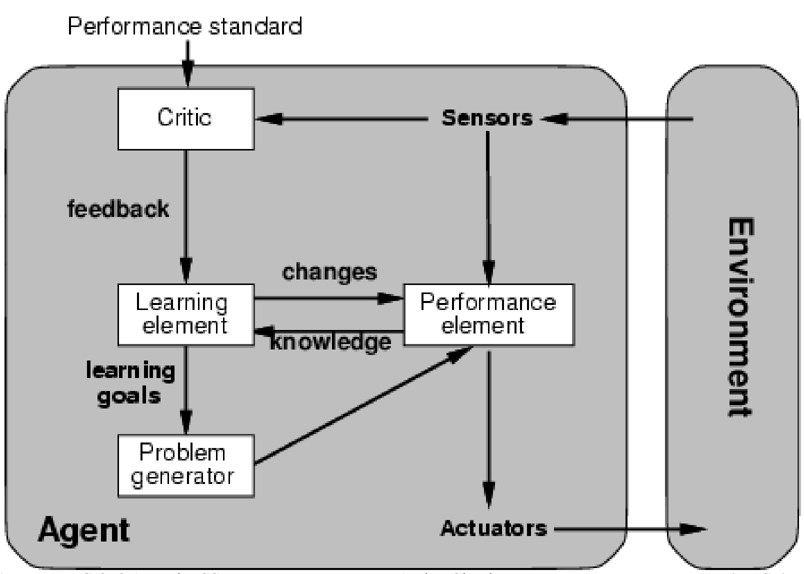
Goal-based Agents



Utility-based Agents



Learning Agents



Solving Problems by Searching

Problem Solving Agent

- Problem solving agent decides what to do by finding sequence of actions that lead to desirable states and hence solution
- 2 types of search algorithms:
  - *Uninformed search algorithms: that are given no information about the problem other than its definition.*
  - *Informed search algorithms: can do quite well given some guidance on where to look for solutions*
- Intelligent agents are supposed to maximize their performance measure.
- Achieving this is sometimes simplified if the agent can adopt a **goal** and aim at satisfying it

Example

- 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*

Goal & Problem Formulation

- Goals help organize behaviour by limiting the objectives that the agent is trying to achieve and hence the actions it needs to consider

- Goal formulation, based on current situation and the agent's performance measure, is the first step in problem solving

- Problem Formulation is the process of deciding what actions and states to consider, given a goal

- In general, an agent with several immediate options of unknown value can decide what to do by first examining different possible sequences of actions that leads to a state of known value, and then choosing the best sequence

Example

One possible route
Arad -> Sibiu -> Ramnincu Valcea -> Pitesti -> Bucharest

Search and Solution

- The process of looking for sequence of actions to arrive at a goal is called search
- Search algorithm takes problem as input and returns solution in form of action sequence
- Once a solution is found, the recommended actions can be carried out. This is called execution.
- Formulate, search, execute design for agent

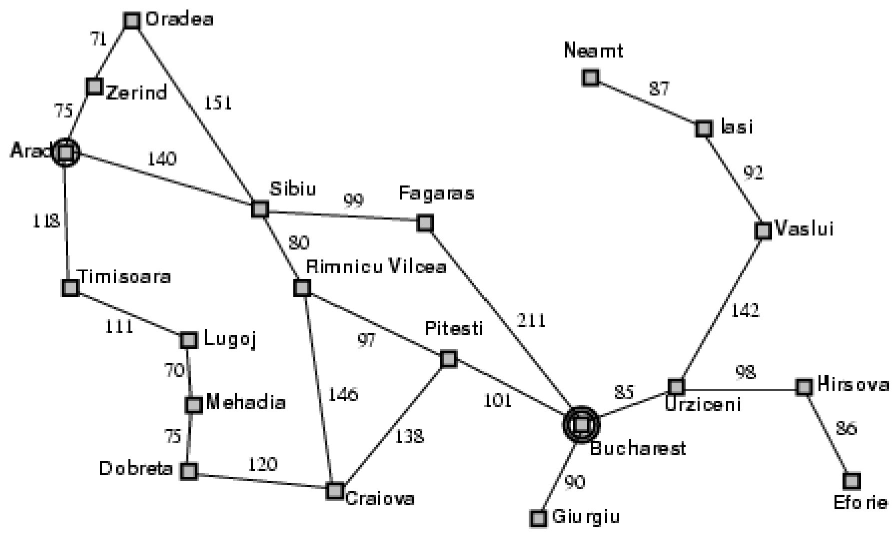
Problem Definition

- A Problem can be defined by the following 5 components:
  - *Initial state defines the start state*
  - *Actions (s) A description of the possible actions available to the agent. Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s*
  - *Transition model/Result (s, a) returns the state that results from doing action a in state s*

Problem Definition (cont'd)

  - *Goal Test (s) a function, when a state is passed to it, returns True or False value if it is a goal or not*
  - *Path Cost an additive function which assigns a numeric cost to each path. This function also reflect agent's own performance measure.*
    - There may be step cost also if a path contains more than one steps. It may be denoted by c(s, a, s'), where a is action and s & s' are the current and new states respectively.
- A path or solution in the state space is the sequence of states connected by sequence of actions
- Together, the initial state, actions and new states implicitly defines the State space of the problem

Find a Route – Arid to Bucharest

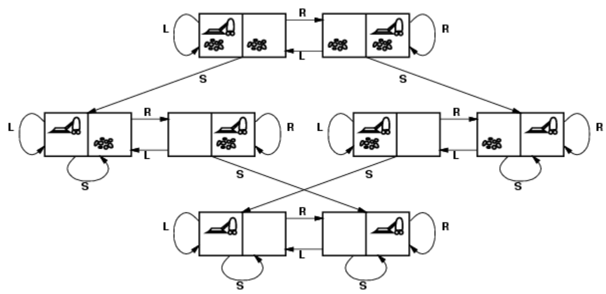


Problem Formulation - Example

- Problem Description: find an optimal path from Arad to Bucharest
- Initial State= *In(Arad)*
- Actions (s) = set of possible actions agent can take

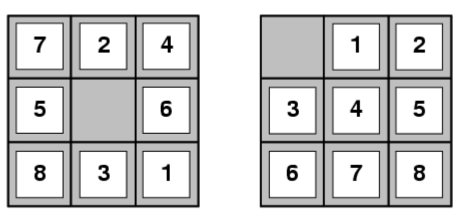
  {goto(Zerind), goto(Sibiu), goto(Timisora)}
- Result (s, a): Result (*In(Arad)*, *Go(Zerind))* = *In(Zerind)* .
- Goal Test: determine if at goal
  - *can be explicit, e.g., In(Bucharest)*
- Path Cost: cost of each step added together
  - *e.g., sum of distances, number of actions executed, etc.*
  - *The step cost is assumed to be* ≥ 0
- A solution is a sequence of actions leading from the initial state to a goal state, solution quality is measured by path cost

Vacuum World State Space Graph



- states? The agent is in one of two locations, 2*2*2 = 8 possible world states
- initial state: any state can be designated as initial state
- Actions (a): {<left>, <right>, <suck>, <noop>}
- Result(s,a): <right clean>, <left clean>
- goal test: no dirt at all locations
- path cost: 1 per action

Example: The 8-puzzle



Start State Goal State

- States: locations of each tile and blank,
  9!/2=181,440 reachable world states
- Initial state: any state can be designated
- Action(s): {<left>, <right>, <up>, <down>}
- Result(s,a): new state after taking any of above actions
- goal test: matches the goal state (given)
- path cost:1 per move

Example: 8-Queen Problem

States: any arrangement of 0 to 8 queens on the board
Initial State: no queen on the board
Actions: add a queen at any square
Result: new board state with queen added
Goal test: 8 queens on the board – none attacked
Path Cost: 1 per move