Unit-I
Fundamentals of Artificial Intelligence

-Dr. Radhika V. Kulkarni

Associate Professor, Dept. of Computer Engineering,

Vishwakarma Institute of Technology, Pune.

Sources:

1. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill
2. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.
3. Deepak Khemani, “A First Course in Artificial Intelligence”, McGraw Hill
4. Saroj Kaushik, “Artificial Intelligence”, Cengage Publication.

DISCLAIMER

This presentation is created as a reference material for the students of TY-CS, VIT (AY 2023-24 Sem-1).

It is restricted only for the internal use and any circulation is strictly prohibited.

Syllabus

Unit-I Fundamentals of Artificial Intelligence

Introduction: A.I. Representation, Non-AI & AI Techniques, Representation of Knowledge, Knowledge Base Systems, State Space Search, Production Systems, Problem Characteristics, Types of production systems, Turing Test.

Intelligent Agents: Agents and Environments, concept of rationality, the nature of environments, structure of agents, problem solving agents, problem formulation.

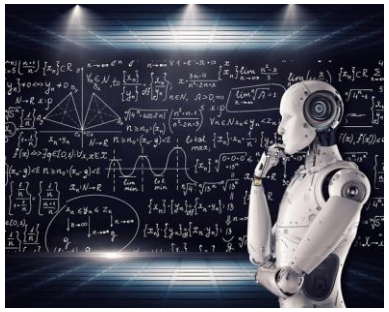
Formulation of problems: Vacuum world, 8 queens, Route finding, robot navigation.

Introduction

Sources:

1. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill
2. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.

What is AI?



- Intelligence: “ability to learn, understand and think” (Oxford dictionary)

- Artificial Intelligence (AI) is a branch of Computer Science that pursues creating the computers or machines as intelligent as human beings.

- **Definition**: Artificial Intelligence is the study of how to make computers do things, which, at the moment, people do better. According to the father of Artificial Intelligence, John McCarthy, it is “The science and engineering of making intelligent machines, especially intelligent computer programs”.

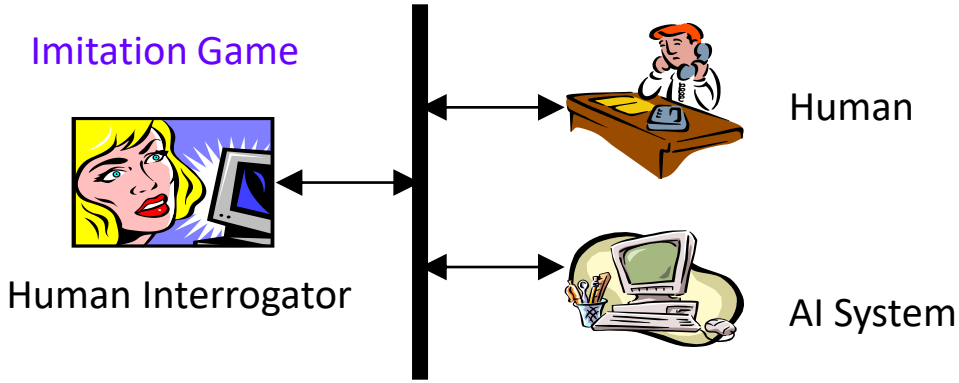
- AI is accomplished by studying how human brain thinks and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

Thinking humanly	Thinking rationally
Acting humanly	Acting rationally

What is AI? (cont..)

- Thinking Humanly:
  - It applies cognitive modelling approach.
  - It is more concerned with comparing reasoning steps of problem solver to traces of human solving the same problem.
  - It requires testable theories of the workings of the human mind- cognitive science.

- Acting Humanly:
  - It applies the Turing test approach.
  - “Can machines think?" → "Can machines behave intelligently?” Alan Turing (1912-1954) proposed an operational test for intelligent behavior: the Imitation Game.
  - “Computing Machinery and Intelligence” (1950) by Turing.
  - Turing Test: A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.
  - Predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes.
  - Anticipated all major arguments against AI in following 50 years. Suggested major components of AI: knowledge, reasoning, language understanding, learning.



What is AI? (cont..)

- Thinking Rationally:
  - It applies laws of thought approach. It is supposed to govern the operation of the mind.
  - Aristotle was one of the first to attempt to codify “right thinking”, i.e., irrefutable reasoning processes.
  - Formal logic provides a precise notation and rules for representing and reasoning with all kinds of things in the world.
  - Obstacles:
    - Informal knowledge representation,
    - Computational complexity and resources.

- Acting Rationally:
  - It applies the rational agent approach.
  - A rational agent is one that acts so as to achieve one’s goals, given one’s beliefs. The agent acts to achieve the best outcome or, when there is uncertainty, the best expected outcome. Acting so as to achieve.
  - The computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals.
  - Does not necessarily involve thinking.
  - Advantages:
    - More general than the “laws of thought” approach because correct inference is just one of several possible mechanisms for achieving rationality.
    - More amenable to scientific development than human-based approaches.

What is AI? (cont..)

- Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think.

- It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting.

- AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.

- From a business perspective AI is a set of very powerful tools, and methodologies for using those tools to solve business problems.

- From a programming perspective, AI includes the study of symbolic programming, problem solving, and search.

Advantages of AI

- High Accuracy with less errors: AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
- High-Speed: AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.
- High reliability: AI machines are highly reliable and can perform the same action multiple times with high accuracy.
- Useful for risky areas: AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
- Digital Assistant: AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.
- Useful as a public utility: AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security purpose, Natural language processing to communicate with the human in human-language, etc.

Disadvantages of AI

- High Cost: The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
- Can't think out of the box: Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
- No feelings and emotions: AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
- Increase dependency on machines: With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
- No Original Creativity: As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

https://www.moralmachine.net/

AI Vocabulary

- Intelligence relates to tasks involving higher mental processes, e.g. creativity, solving problems, pattern recognition, classification, learning, induction, deduction, building analogies, optimization, language processing, knowledge and many more. Intelligence is the computational part of the ability to achieve goals.

- Intelligent behavior is depicted by perceiving one's environment, acting in complex environments, learning and understanding from experience, reasoning to solve problems and discover hidden knowledge, applying knowledge successfully in new situations, thinking abstractly, using analogies, communicating with others and more.

- Science based goals of AI pertain to developing concepts, mechanisms and understanding biological intelligent behavior. The emphasis is on understanding intelligent behavior.

- Engineering based goals of AI relate to developing concepts, theory and practice of building intelligent machines. The emphasis is on system building.

AI Vocabulary (cont..)

- AI Techniques depict how we represent, manipulate and reason with knowledge in order to solve problems. Knowledge is a collection of ‘facts’. To manipulate these facts by a program, a suitable representation is required. A good representation facilitates problem solving.

- Intelligent Agents are able to reason and make decisions on their own, and they are a key component of many AI applications. They take the best possible action in a situation.

- Learning means that programs learn from what facts or behaviour can represent. Learning denotes changes in the systems that are adaptive in other words, it enables the system to do the same task(s) more efficiently next time.

- Applications of AI refers to problem solving, search and control strategies, speech recognition, natural language understanding, computer vision, expert systems, game playing etc.

Non-AI and AI Techniques

- Non-AI techniques involve a series of instructions that are followed when a trigger is encountered. An information system that is built using decision-making techniques other than AI that include procedural or declarative knowledge. Procedural experience maps the process of decision making into the processes by which the issues are solved, or decisions are made.

- AI techniques are models built from sophisticated elements of the computational and mathematical models. Such models allow a computer or machine to measure tasks that are supposed to be performed by humans. E.g., Heuristics, Support Vector Machines, Artificial Neural Networks, Markov Decision Process, Natural Language Processing etc.

- In technical terms, An algorithm (Non-AI) is a set of instructions — a hardcoded recipe that the system executes line by line when it is triggered.

- AI on the other hand — which is trying to give “common sense” to computers that we humans use in our day to day life — is a group of algorithms (smart) that modify the present algorithm based on the previous inputs and the data processed using the inputs in an attempt to reduce the space and time complexity. Here the term “intelligence” is used to show the ability of the computer to change, adapt and grow based on new data.

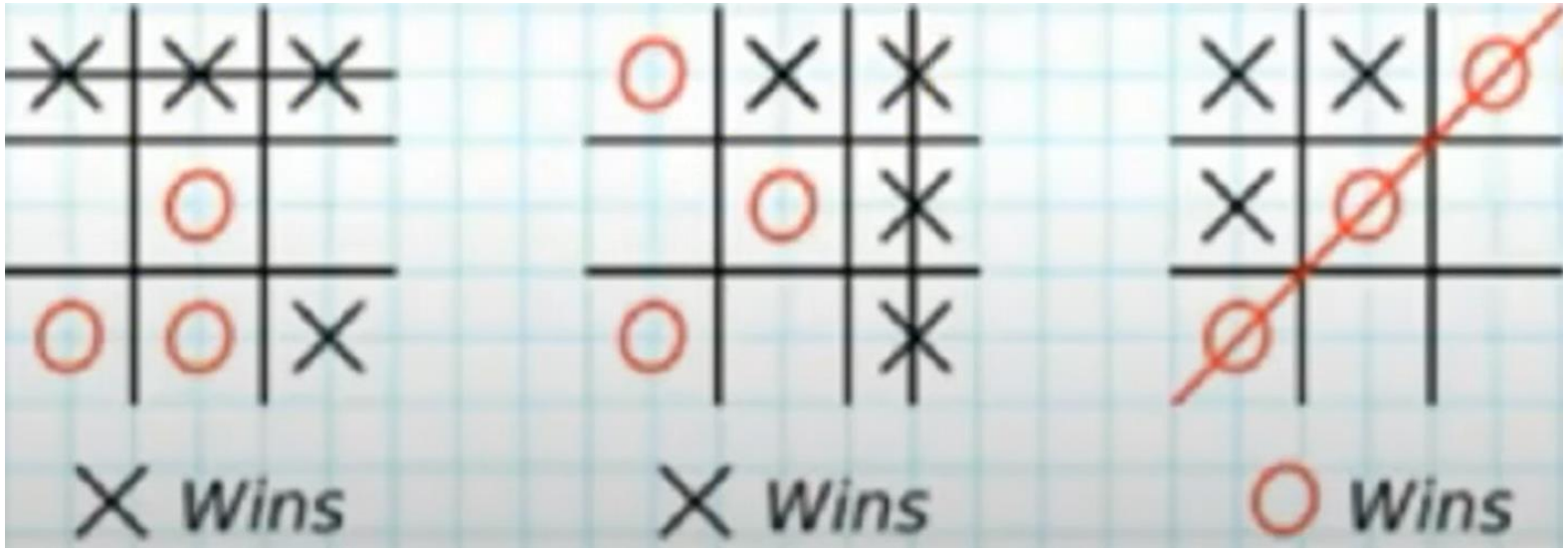
- The bottom line, non-AI technique is when the system keeps on following the defined set of rules to reach the solution and AI is when the system learns from its past, overcomes its mistakes and gives more optimal solutions to the problems.

Non-AI and AI Techniques (cont..)

Non-AI Machines	AI Machines
Smart machines which are not AI, do not require training data, they work on algorithms only.	AI machines are trained with data and algorithm.
Smart machines work on fixed algorithms and they always work with the same level of efficiency, which is programmed into them.	AI machines learn from mistakes and experience. They try to improvise on their next iterations.
Machines which are not AI cannot take decisions on their own.	AI machines can analyze the situation and can take decisions accordingly.
An automatic door in a shopping mall, seems to be AI-enabled, but it is built with only sensor technology	AI based drones capture the real-time data during the flight, processes it in real-time, and makes a humanin dependent decision based on the processed data.

Example: Tic-Tac-Toe

- Tic-Tac-Toe is a paper-pencil game of two-players **X** and **O**, who selects to marks a spaces on 3x3 grid.
- A player who succeeds in putting a 3 of their marks in a horizontal, vertical or diagonal line is a winner of the game.



Tic-Tac-Toe: Program 1 (Non-AI Technique)

- Data Structure BOARD: The Tic-Tac-Toe game consists of a nine-element vector called BOARD; it represents the numbers 1 to 9 in three rows. An element contains the value 0 for blank, 1 for X and 2 for O.

1	2	3
4	5	6
7	8	9

2-D Board

1	2	3	4	5	6	7	8	9

1-D Vector BOARD representing a board

- Data Structure MOVETABLE: A MOVETABLE vector consists of 19,683 elements (3^9) and is needed where each element is a nine-element vector. The contents of the vector are especially chosen to help the algorithm.

Index	Current Board Position	Next Board Position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010
:		
19682		

Tic-Tac-Toe: Program 1 (cont..)

- The algorithm: (Assumption: 1st player is X.)
  1. View the vector as a ternary number. Convert it to a decimal number.
  2. Use the decimal number as an index in MOVETABLE and access the vector.
  3. Set BOARD to this vector indicating how the board looks after the move.

- This approach is capable in time, but it has several disadvantages.
- Disadvantages:
  - It takes more space to store MOVETABLE.
  - It has a lot of work to specify all entries in MOVETABLE.
  - It requires stunning effort to calculate the decimal numbers.
  - It is highly error-prone due to huge volume of data.
  - This method is specific to this 2-D game and cannot be completed for more dimensions like 3-D for which 3^{27} board positions would have to be stored.

Tic-Tac-Toe: Program 2

- Data Structure BOARD: The structure of the data is as before but we use 2 for a blank, 3 for an X and 5 for an O.
- Data Structure TURN: An integer variable called TURN indicates 1 for the first move and 9 for the last.

- The algorithm consists of three sub procedures:
  - MAKE2: Returns 5 if the center square is blank(i.e., if BOARD[5]=2, then return 5); otherwise, it returns any blank non corner square, i.e., 2, 4, 6 or 8.

1	2	3
4	5	6
7	8	9

  - POSSWIN(p): Returns 0 if player p cannot win on the next move and otherwise returns the number of the square that gives a winning move. It operates by checking, one at a time, each of the rows, columns and diagonals. It checks each line using products of values of its squares together. A product 3*3*2 = 18 gives a win for X, 5*5*2=50 gives a win for O, and the winning move is the holder of the blank.

  - GO(n): Makes a move to square n. It sets BOARD[n] to 3 if TURN is odd or 5 if TURN is even. It also increments TURN by 1.

Tic-Tac-Toe: Program 2 (cont..)

- The algorithm has a built-in-strategy for each move it may have to make. It makes odd-numbered moves if it is playing X and even-numbered moves if it is playing O. The strategy for each turn is as follows:
  - TURN=1: Go(1) (Upper left corner).
  - TURN=2: If Board[5] =2(blank), Go(5), else Go(1).
  - TURN=3: If Board[9] =2(blank), Go(9), else Go(3).
  - TURN=4: If POSSWIN(X)≠0, then Go(POSSWIN(X)) [i.e. block opponent's win], else Go(Make2).
  - TURN=5: If POSSWIN(X)≠0, then Go(POSSWIN(X)) [i.e. win], else if POSSWIN(O)≠0, then Go(POSSWIN(O)) [i.e. block opponent's win], else if BOARD[7] =2(blank), Go(7), else Go(3). [Here, program is trying to make a fork.]
  - TURN=6: If POSSWIN(O)≠0, then Go(POSSWIN(O)), else if POSSWIN(X)≠0, then Go(POSSWIN(X)), else Go(Make2).
  - TURN=7: If POSSWIN(X)≠0, then Go(POSSWIN(X)), else if POSSWIN(O)≠0, then Go(POSSWIN(O)), else go anywhere that is blank.
  - TURN=8: If POSSWIN(O)≠0, then Go(POSSWIN(O)), else if POSSWIN(X)≠0, then Go(POSSWIN(X)), else go anywhere that is blank.
  - TURN=9: Same as TURN 7.
- This algorithm checks several conditions before making each move so, takes longer time but it is more efficient in storage which compensates for its longer time. It depends on the programmer's skill. Also, it cannot be generalized to different domains like 3-D Tic-Tac-Toe.

Tic-Tac-Toe: Program 2’

- It is identical to Program 2 except for the one change in the board representation.
- Data Structure BOARD: The numbering of the BOARD produces a magic square: all row, columns and diagonals sum up to 15.

8	3	4
1	5	9
6	7	2

- It simplifies the process of checking of a possible win.
- The algorithm:
  - In addition to marking the BOARD as moves are made, keep a list for each player, of squares in which he/she has played.
  - To check a possible win for one player, consider each pair of squares owned by that player and compute the difference between 15 and sum of the two squares. If the difference is < 0 (negative) or > 9, then the original two squares are not collinear and so can be ignored.
  - Otherwise, if the square representing the difference is blank, a move there will produce win
  - or check if the opponent is winning, block his/her win.
- Since, no player can have more than 4 squares at a time, there will be many fewer squares examined using this scheme than using approach in Program 2.
- This shows how the choice of representation can have impact on the efficiency on problem-solving program.

Tic-Tac-Toe: Program 2’

- E.g. BORAD

8	3	4
1	5	9
6	7	2

TURN1

	X	

TURN2

O		
	X	

Turn3

O		X
	X	

TURN4

O		X
	X	
O		

TURN5

O		X
X	X	
O		

TURN6

O		X
X	X	
O		

TURN7

O		X
X	X	O
O		

and so on….

Tic-Tac-Toe: Program 3 (AI Technique)

- Data Structure BOARDPOSITION: The structure contains a nine-element vector, a list of board positions that could result from the next move and a number representing an estimation of how the board position leads to an ultimate win for the player to move.

- The algorithm looks ahead to make a decision on the next move by deciding which the most promising move or the most suitable move at any stage would be, selects the same and assign the rating of the best move to the current position.
- To decide which of a set of BOARD positions is best, do the following for each of them:
  - See if it is a win, If so, call the best by giving it the highest possible rating.
  - Otherwise, consider all the moves the opponent could make next. See which of them is the worst for us(by recursively calling this procedure). Assume the opponent will make that move. Whatever rating that move has , assign it to the node we are considering.
  - The best node is then the one with the highest rating.
- The algorithm will look ahead all possible moves to find out a sequence that leads to a win , if possible, in the shortest time.
- It attempts to maximize a likelihood of winning, while assuming that opponent will try to minimize that likelihood.

Tic-Tac-Toe: Program 3 (cont..)

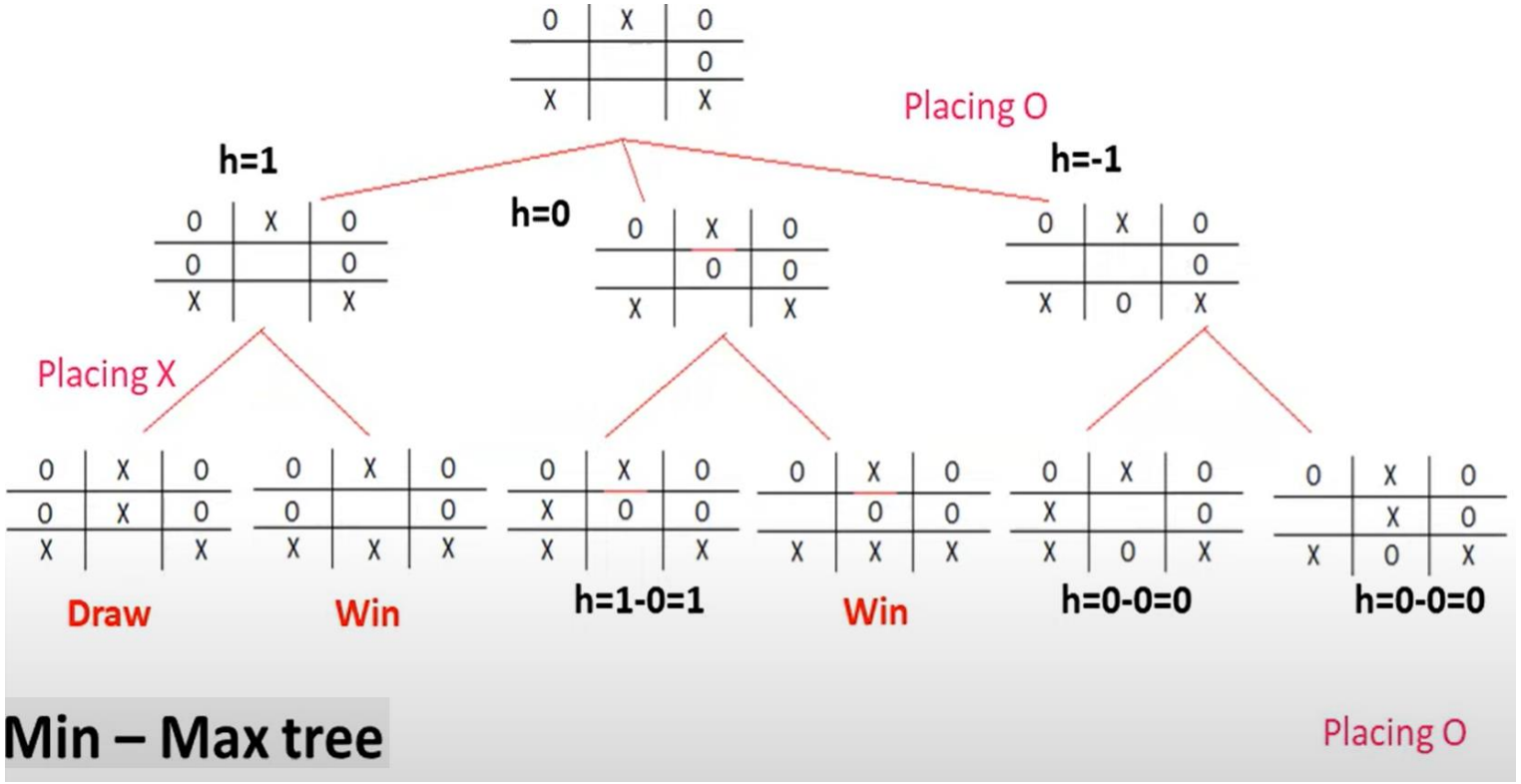
E.g.

Heuristics Function H= P(X wins) - P(O wins)

H=0 is DRAW.

X chooses Maximum value of H.

O chooses Minimum value of H.



- Actually, this is most difficult to program by a good limit as it will need more time and computations to search a tree representing all possible move sequences prior to each move, but it is superior to other programs as this technique can be extended to in any game.
- It can be augmented by a variety of specific kinds of knowledge of games and reduce number searches by applying simple and reasonable algorithms.
- This method makes relatively fewer loads on the programmer in terms of the game technique, but the overall game strategy must be known to the adviser.

Knowledge Representation

- Humans are best at understanding, reasoning, interpreting knowledge and performing various actions in the real world as per their knowledge. But how machines do all these things comes under knowledge representation and reasoning.

- Knowledge representation can be described as following:
  - Knowledge representation and reasoning is the part of AI which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
  - It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real-world problems such as diagnosis a medical condition or communicating with humans in natural language.
  - It is also a way which describes how we can represent knowledge in AI. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Knowledge Representation (cont..)

- Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. It is typically represented in the form of rules or facts, which can be used to draw conclusions or make decisions.
- It plays an important role in demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input.
- Following are the types of knowledge in AI:
  1. **Declarative Knowledge:** It is to know about something. It includes concepts, facts, and objects. It is also called descriptive knowledge and expressed in declarative sentences. It is simpler than procedural language.
  2. **Procedural Knowledge:** It is also known as imperative knowledge. It is responsible for knowing how to do something. It can be directly applied to any task. It includes rules, strategies, procedures, agendas, etc. It depends on the task on which it can be applied.
  3. **Meta-knowledge:** The knowledge about the other types of knowledge is called Meta-knowledge.
  4. **Heuristic knowledge:** It is representing knowledge of some experts in a subject. It is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.
  5. **Structural knowledge:** It is basic knowledge to problem-solving. It describes relationships between various concepts such as kind of, part of, and grouping of something. It describes the relationship that exists between concepts or objects.

Knowledge Based Systems

- A knowledge-based system is a system that uses AI techniques to store and retrieve knowledge.
- The system stores knowledge in the form of rules, which are then used to generate solutions to specific problems. The rules are typically written in a formal language, which allows the system to reason about the knowledge and apply it to new situations.
- The system's ability to reason about the knowledge is what allows it to generate new solutions to problems. The system can also learn new knowledge by observing how humans solve problems.
- Knowledge-based systems are a type of artificial intelligence and have been used in a variety of applications including medical diagnosis, expert systems, and decision support systems.

- The components of a knowledge-based system:
  1. A knowledge base: This is a database of facts and rules that the system can use to make decisions.
  2. An inference engine: This is the part of the system that uses the knowledge base to make deductions and reach conclusions.
  3. A user interface: This is the part of the system that allows humans to interact with the system, usually through natural language.

Knowledge Based Systems (cont..)

- Advantages of a knowledge-based system
  - It can help to automate decision-making processes. For example, a knowledge-based system could be used to diagnose a medical condition, by reasoning over a set of rules that describe the symptoms and possible causes of the condition.
  - It can be used to explain their decisions to humans. This can be useful, for example, in a customer service setting, where a knowledge-based system can help a human agent to understand why a particular decision was made.
  - It can help you to create more intelligent and efficient systems.
  - It can also help you to create systems that are more robust and easier to maintain.
  - It can help you to create systems that are more flexible and adaptable to change.

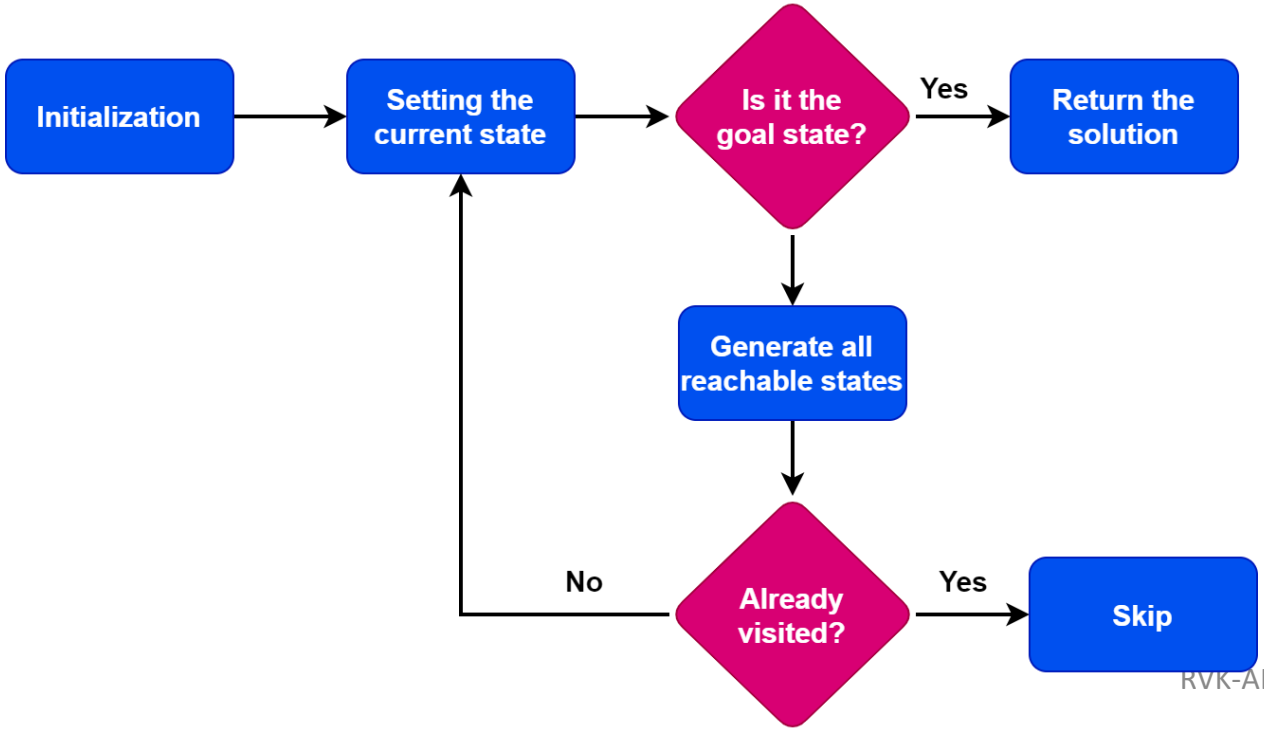
- Challenges associated with knowledge-based systems
  - Frame problem: This is the problem of how to represent the world in a way that makes it possible for the system to reason about changes.
  - Combinatorial explosion problem: This is the problem of how to deal with the fact that the number of possible states of the world grows exponentially as the number of objects in the world increases.
  - There is the challenge of how to deal with incomplete or uncertain information.

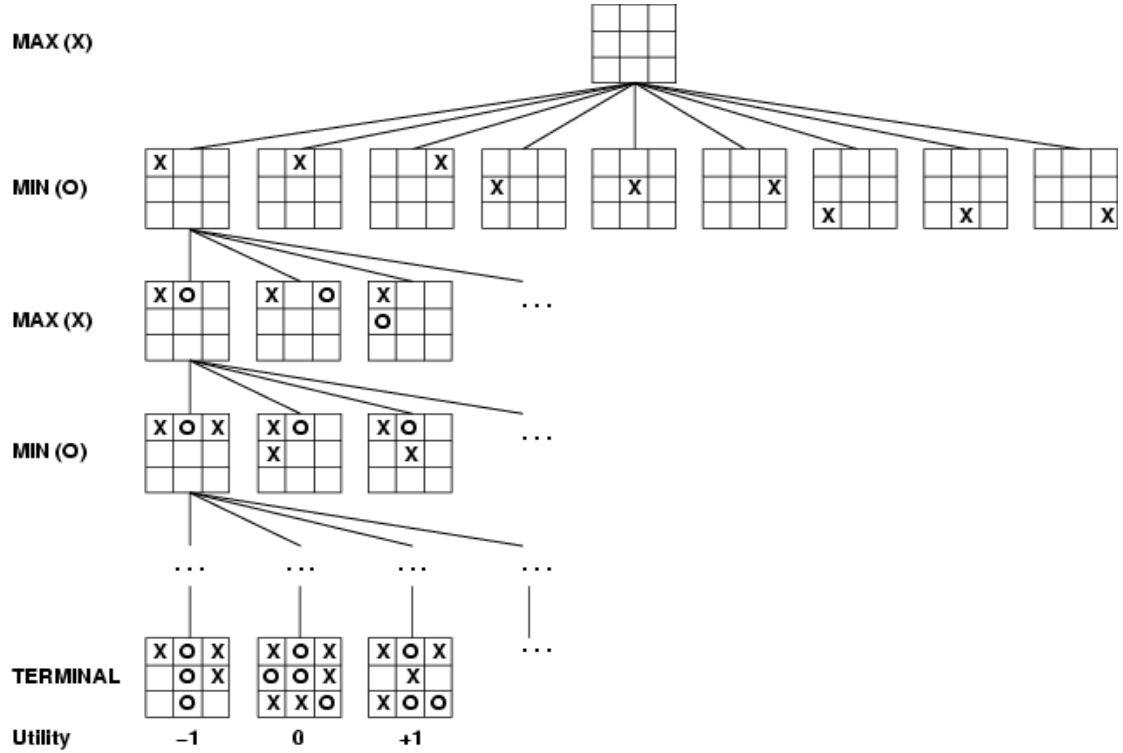
State Space

- A state space represents a problem in terms of states and operators that change states.
- It is the set of all states reachable from the initial state.
- A set of operators in a state space can change one state into another state. In a board game, the operators are the legal moves from any given state. Often the operators are represented as programs that change a state representation to represent the new state.
- It has a set of final states; some of these may be desirable, others undesirable. This set is often represented implicitly by a program that detects terminal states.
- It is a representation of the states the system can be in. For example, in a board game, the board represents the current state of the game.
- In a state space, a path is a sequence of states connected by a sequence of actions.
- The solution of a problem is part of the map formed by the state space.
- The structures of a state space are trees and graphs.
  - A tree is a hierarchical structure in a graphical form. It has only one path to a given node; i.e., a tree has one and only one path from any point to any other point.
  - A graph is a non-hierarchical structure consisting of a set of nodes (vertices) and a set of edges (arcs). Arcs establish relationships (connections) between the nodes; i.e., a graph has several paths to a given node. The *Operators* are directed *arcs* between nodes.

State Space Search

- State space search is a method used widely in artificial intelligence and computer science to find a solution to a problem by searching through the set of possible states of the problem.
- Furthermore, a state space search algorithm uses the state space to navigate from the initial state to the goal state. Additionally, it generates and explores possible successors of the current state until we find a solution.
- In the worst case, the search explores all possible paths between the initial state and the goal state.
- In the state space, a solution to a problem is a path from the initial state to a goal state or, sometimes, just a goal state.
- The state space size can greatly affect a search algorithm's efficiency. Hence, it's important to choose an appropriate representation and search strategy to efficiently search the state space.





Problem Solving

- Problem solving is a process of generating solutions from observed data. A problem is characterized by a set of goals, a set of objects, and a set of operations.
- To solve the problem of building a system you should take the following steps:
  1. Define the problem accurately including detailed specifications and what constitutes a suitable solution.
  2. Scrutinize the problem carefully, for some features may have a central affect on the chosen method of solution.
  3. Segregate and represent the background knowledge needed in the solution of the problem.
  4. Choose the best solving techniques for the problem to solve a solution.
- A problem space is an abstract space.
  - It encompasses all valid states that can be generated by the application of any combination of operators on any combination of objects.
  - The problem space may contain one or more solutions. A solution is a combination of operations and objects that achieve the goals.
- A search refers to the search for a solution in a problem space.
  - It proceeds with different types of search control strategies.
  - The depth-first search and breadth-first search are the two common search strategies.
- Problem solution
  - A solution cost function assigns a numeric cost to each path; it also gives the cost of applying the operators to the states.
  - A solution quality is measured by the path cost function; and an optimal solution has the lowest path cost among all solutions.
  - The solutions can be any or optimal or all.
  - The importance of cost depends on the problem and the type of solution asked.

Problem Characteristics

- Since most AI problems make use of knowledge and guided search through the knowledge, AI can be described as the study of techniques for solving exponentially hard problems in polynomial time by exploiting knowledge about problem domain.

- To use the heuristic search for problem solving, we suggest analysis of the problem for the following considerations:
  - Decomposability of the problem into a set of independent smaller subproblems
  - Possibility of undoing solution steps, if they are found to be unwise
  - Predictability of the problem universe
  - Possibility of obtaining an obvious solution to a problem without comparison of all other possible solutions
  - Type of the solution: whether it is a state or a path to the goal state
  - Role of knowledge in problem solving
  - Nature of solution process: with or without interacting with the user

- The general classes of engineering problems such as planning, classification, diagnosis, monitoring and design are generally knowledge intensive and use a large amount of heuristics.

Problem Characteristics (cont..)

- Depending on the type of problem, the knowledge representation schemes and control strategies for search are to be adopted.
- There are a number of other general-purpose search techniques which are essentially heuristics based. Their efficiency primarily depends on how they exploit the domain specific knowledge to abolish undesirable paths. Such search methods are called 'weak methods', since the progress of the search depends heavily on the way the domain knowledge is exploited.
- Three important classes of problems :
  1. Ignorable, in which solution steps can be ignored. These problems involve creating new things rather than changing old ones. E.g.: Theorem Proving, Making deductions from some facts.
  2. Recoverable, in which solution steps can be undone. E.g.: 8-Puzzle
  3. Irrecoverable, in which solution steps cannot be undone. E.g.: Chemical process

- Predictable Vs Unpredictable Problems:
  - In predictable problems as the outcome is very certain it is possible to plan an entire sequence of moves and be confident what the resulting state will be. We can backtrack to earlier moves if they prove unwise. E.g. 8-Puzzle.
  - In unpredictable problems as the outcome is uncertain, we follow the process of plan revision as the plan is carried out and the necessary feedback is provided. The disadvantage is that the planning in this case is often very expensive. E.g. Bridge, Controlling a robot arm.

Problem Characteristics (cont..)

- Absolute Vs Relative Problem:
- Consider the following problem of answering questions based on a database of simple facts:
    1. Siva was a man.
    2. Siva was a worker in a company.
    3. Siva was born in 1915.
    4. All men are mortal.
    5. All workers in a factory died when there was an accident in 1952.
    6. No mortal lives longer than 100 years.
- Suppose we ask a question: **'Is Siva alive?'** By representing these facts in a formal language, such as predicate logic, and then using formal inference methods we can derive an answer to this question easily. There are two ways to answer the question shown below:
    - **Method I:** 1. Siva was a man. 2. Siva was born in 1915. 3. All men are mortal. 4. Now it is 2023, so Siva's age is 108 years. 5. No mortal lives longer than 100 years.
    - **Method II:** 1. Siva is a worker in the company. 2. All workers in the company died in 1952.
- **Answer:** So, **Siva is not alive**. It is the answer from the above methods.
- We are interested to answer the question; it does not matter which path we follow. If we follow one path successfully to the correct answer, then there is no reason to go back and check another path to lead the solution.

Production Systems

- In Artificial Intelligence, a production system serves as a cognitive architecture. It encompasses rules representing declarative knowledge, allowing machines to make decisions and act based on different conditions.
- It helps in structuring AI programs in such a way that facilitates describing and performing a search process.
- Many expert systems and automation methodologies rely on the rules defined in production systems to guide their behavior.
- The production rules operate on the knowledge database. Each rule has a precondition—that is, either satisfied or not by the knowledge database. If the precondition is satisfied, the rule can be applied.
- Application of the rule changes the knowledge database. The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the knowledge database is satisfied.

- A production system has four basic components as enumerated below.
  1. A set of rules each consisting of a left side that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.
  2. A database of current facts established during the process of inference.
  3. A control strategy that specifies the order in which the rules will be compared with facts in the database and also specifies how to resolve conflicts in selection of several rules or selection of more facts.
  4. A rule firing module which checks the capability of rule by matching the content state with the left hand side of the rule and finds the appropriate rule from database of rules.

Production Systems (cont..)

- E.g. **Eight puzzle (8-Puzzle):** The 8-puzzle is a 3 × 3 array containing eight square pieces, numbered 1 through 8, and one empty space. A piece can be moved horizontally or vertically into the empty space, in effect exchanging the positions of the piece and the empty space. There are four possible moves, UP (move the blank space up), DOWN, LEFT and RIGHT. The aim of the game is to make a sequence of moves that will convert the board from the start state into the goal state. This example can be solved by the operator sequence UP, RIGHT, UP, LEFT, DOWN.

Initial State

8	3	4
1	5	
6	7	2

Goal State

7	3	6
1		8
4	5	2

- Characteristics/ Advantages of a Production System
  - Simplicity: The structure of each sentence in a production system is unique and uniform as they use “IF-THEN” structure. It provides simplicity in knowledge representation. It improves the readability of production rules.
  - Modularity: This means production rule code the knowledge available in discrete pieces. Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deleterious side effects.
  - Modifiability: This means the facility of modifying rules. It allows the development of production rules in a skeletal form first and then it is accurate to suit a specific application.
  - Knowledge intensive: The knowledge base of production system stores pure knowledge. This part does not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation.

Production Systems (cont..)

- Disadvantages of production system
  - Opacity: This problem is generated by the combination of production rules. The opacity is generated because of less prioritization of rules. More priority to a rule has the less opacity.
  - Inefficiency: During execution of a program several rules may active. A well devised control strategy reduces this problem. As the rules of the production system are large in number and they are hardly written in hierarchical manner, it requires some forms of complex search through all the production rules for each cycle of control program.
  - Absence of learning: Rule based production systems do not store the result of the problem for future use. Hence, it does not exhibit any type of learning capabilities. So, for each time for a particular problem, some new solutions may come.
  - Conflict resolution: The rules in a production system should not have any type of conflict operations. When a new rule is added to a database, it should ensure that it does not have any conflicts with the existing rules.

Types of Production Systems

1. Monotonic Production System: In this type of a production system, the rules can be applied simultaneously as the use of one rule does not prevent the involvement of another rule that is selected at the same time. Rules are independent in nature.

2. Non-monotonic Production System: This type of a production system increases efficiency in solving problems. The implementation of these systems does not require backtracking to correct the previous incorrect moves. The non-monotonic production systems are necessary from the implementation point of view to find an efficient solution.

3. Partially Commutative Production System: This class helps to create a production system that can give the results even by interchanging the states of rules. If using a set of rules transforms State A into State B, then multiple combinations of those rules will be capable to convert State A into State B.

4. Commutative System: Commutative systems are helpful where the order of an operation is not important. It is both monotonic and partially commutative. Also, problems where the changes are reversible use commutative systems. On the other hand, partially commutative production systems help in working on problems, where the changes are irreversible such as a chemical process. When dealing with partially commutative systems, the order of processes is important to get the correct results.

Types of Production Systems (cont..)

Four Categories of Production Systems

Production System	Monotonic	Non-monotonic
Partially Commutative	Theorem Proving	Robot Navigation
Non-partially Commutative	Chemical Synthesis	Bridge

- In the formal sense, there is no relationship between kinds of problems and kinds of production systems as all problems can be solved by all kinds of systems. But in the practical sense, there is definitely such a relationship between the kinds of problems and the kinds of systems that lend themselves to describing those problems.
  - Partially commutative, monotonic productions systems are useful for solving ignorable problems. These are important from an implementation point of view without the ability to backtrack to previous states when it is discovered that an incorrect path has been followed.
  - Both types of partially commutative production systems are significant from an implementation point; they tend to lead to many duplications of individual states during the search process.
  - Production systems that are not partially commutative are useful for many problems in which permanent changes occur.
- E.g.
  - Theorem proving: Partially commutative, monotonic, ignorable,
  - Robot Navigation: Partially commutative, non-monotonic, recoverable
  - Chemical synthesis: Non-partially commutative, monotonic, order is important, irrecoverable
  - Bridge: Non-partially commutative, non-monotonic, less chances of producing the same node many times in search process.

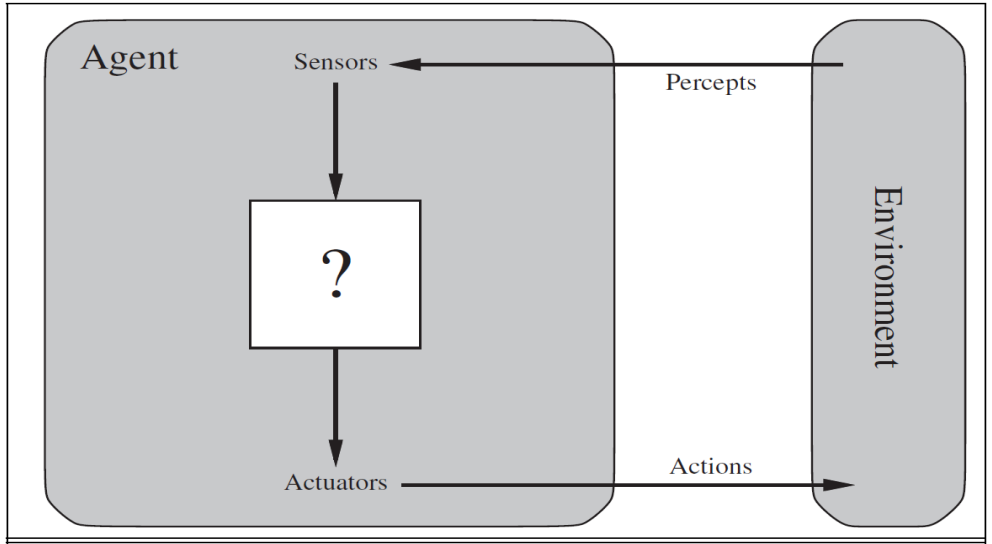
Intelligent Agents

Source:

1. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill
2. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.

Agents and Environments

- An agent is something that perceives and acts in an environment.
- The agent function for an agent specifies the action taken by the agent in response to any percept sequence.
- The performance measure evaluates the behavior of the agent in an environment.
- A task environment specifies the performance measure, the external environment, the actuators, and the sensors **(PEAS)**.
  - In designing an agent, the first step must always be to specify the task environment as fully as possible.
  - Task environments vary along several significant dimensions.
  - They can be fully or partially observable, single-agent or multiagent, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.
- A rational agent acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far. What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence to date.



Agents interact with environments through sensors and actuators.

Agents and Environments (cont..)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Examples of agent types and their PEAS descriptions.

The Nature of Task Environments

- The design of an agent heavily depends on the type of environment they act upon.
  - In partially observable environments, the agent does not have full information about the state and thus the agent must have an internal estimate of the state of the world. This is in contrast to fully observable environments, where the agent has full information about their state.
  - Stochastic environments have uncertainty in the transition model, i.e., taking an action in a specific state may have multiple possible outcomes with different probabilities. This is in contrast to deterministic environments, where taking an action in a state has a single outcome that is guaranteed to happen.
  - In multi-agent environments the agent acts in the environments along with other agents. For this reason, the agent might need to randomize its actions in order to avoid being "predictable" by other agents. However in a single-agent system has only one agent.
  - If the environment does not change as the agent acts on it, then this environment is called static. This is contrast to dynamic environments that change as the agent interacts with it.
  - In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. In sequential environments, on the other hand, the current decision could affect all future decisions.
  - The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
  - In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given. Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.

The Nature of Task Environments (cont..)

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

Examples of task environments and their characteristics.

The Structure of Agents

- The job of AI is to design an agent program that implements the agent function—the mapping from percepts to actions.
- There exists a variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process.
- The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.
- All agents can improve their performance through learning.
- We assume the agent program will run on some sort of computing device with physical sensors and actuators—we call this the architecture: agent = architecture + program
- The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table.

Types of Agents

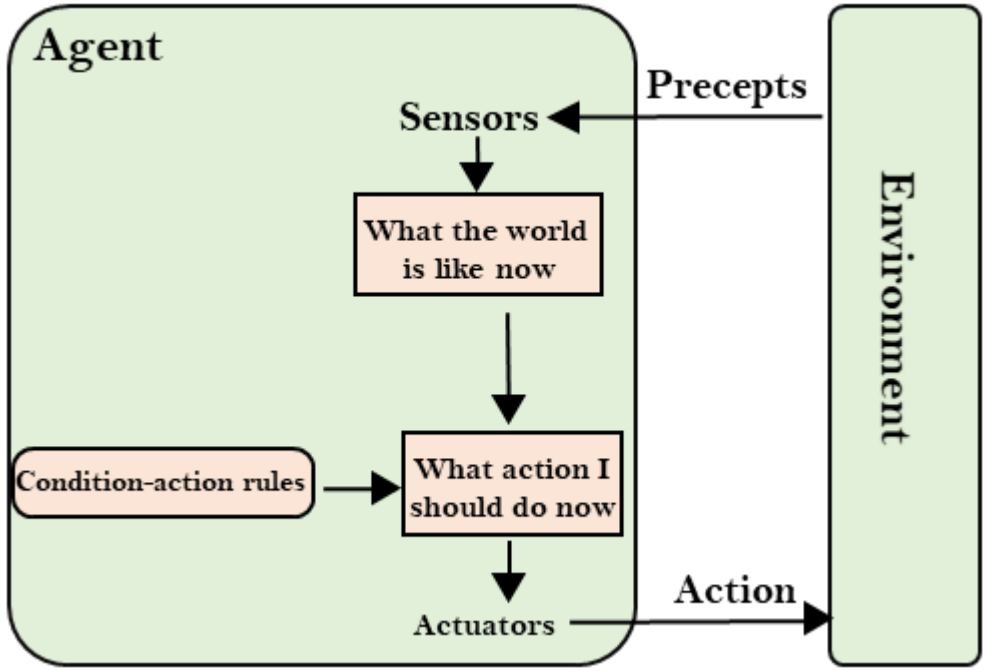
1. Simple reflex agents:

- They take decisions on the basis of the current percepts and ignore the rest of the percept history.
- They only succeed in the fully observable environment.
- They work on **condition-action rules / situation–action rules/ productions/if–then rules**, which means they maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:
  - They have very limited intelligence
  - They do not have knowledge of non-perceptual parts of the current state
  - Mostly too big to generate and to store.
  - Not adaptive to changes in the environment.

```
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
```

A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.



Types of Agents (cont..)

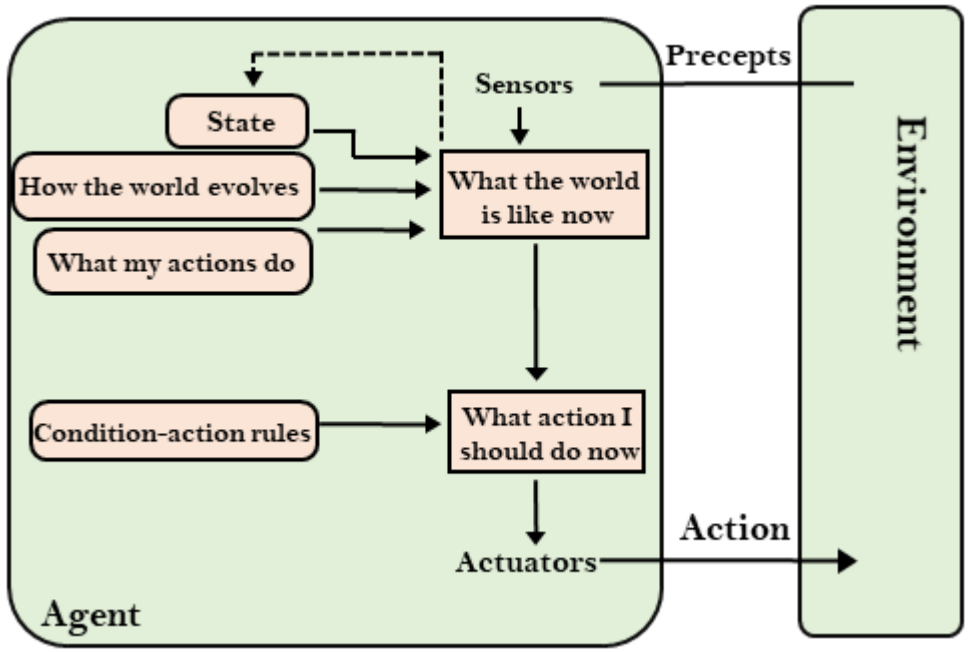
2. Model-based reflex agents:

- They maintain internal state to track aspects of the world that are not evident in the current percept.
- They can work in a partially observable environment and track the situation.
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - Internal State: It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
persistent: *state*, the agent's current conception of the world state
model, a description of how the next state depends on current state and action
rules, a set of condition–action rules
action, the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

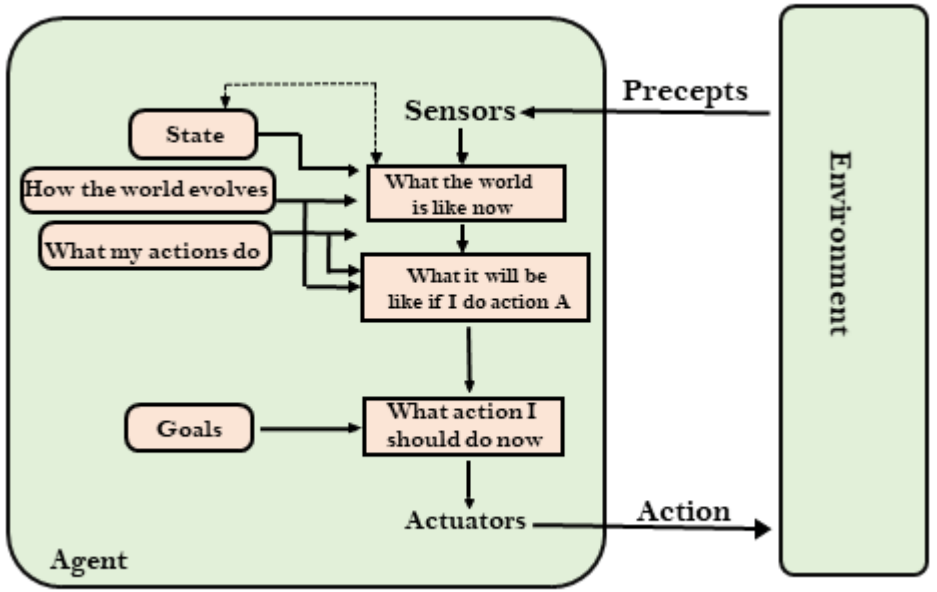
A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.



Types of Agents (cont..)

3. Goal-based agents:

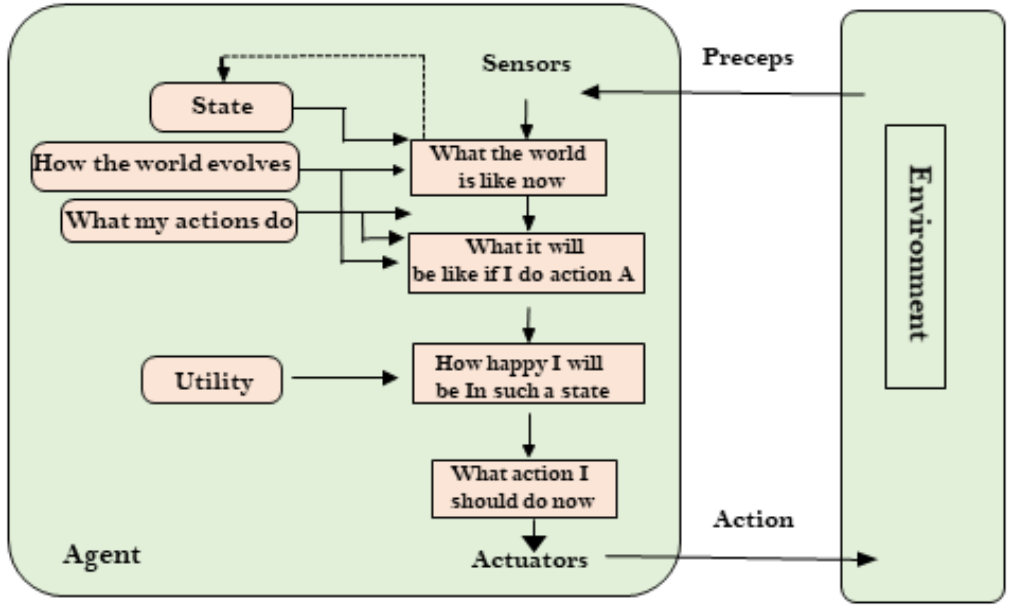
- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations. They choose an action, so that they can achieve the goal.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.



Types of Agents (cont..)

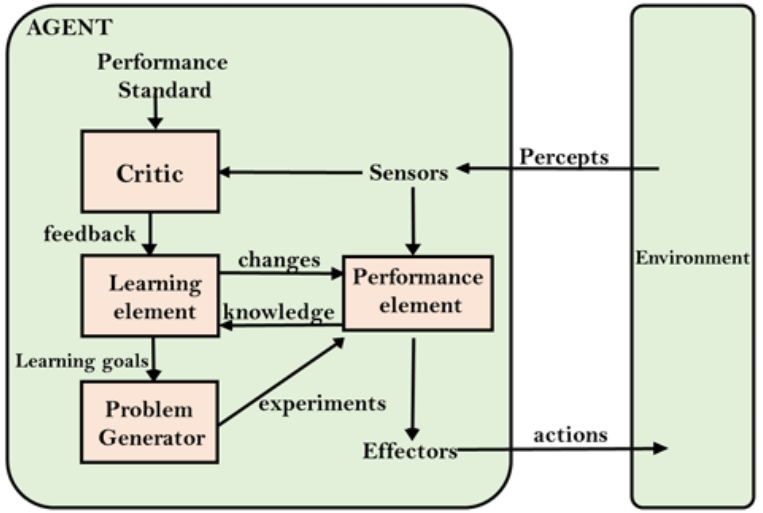
4. Utility-based agents:

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.



A General Learning Agent

- A learning agent in AI can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.
- A learning agent has mainly four conceptual components, which are:
  - **Learning element:** It is responsible for making improvements by learning from environment
  - **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  - **Performance element:** It is responsible for selecting external action
  - **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.
- Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.



Problem Solving Agents

- The **problem-solving agents** are one kind of goal-based agents. They use **atomic** representations, that is, states of the world are considered as wholes, with no internal structure visible to the problem-solving algorithms.
- Goal-based agents that use more advanced **factored** or **structured** representations are usually called **planning agents.**

```
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 ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

Problem Formulation

Source:

1. Elaine Rich and Kevin Knight, "Artificial Intelligence" Tata McGraw Hill
2. Stuart Russell & Peter Norvig, "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.

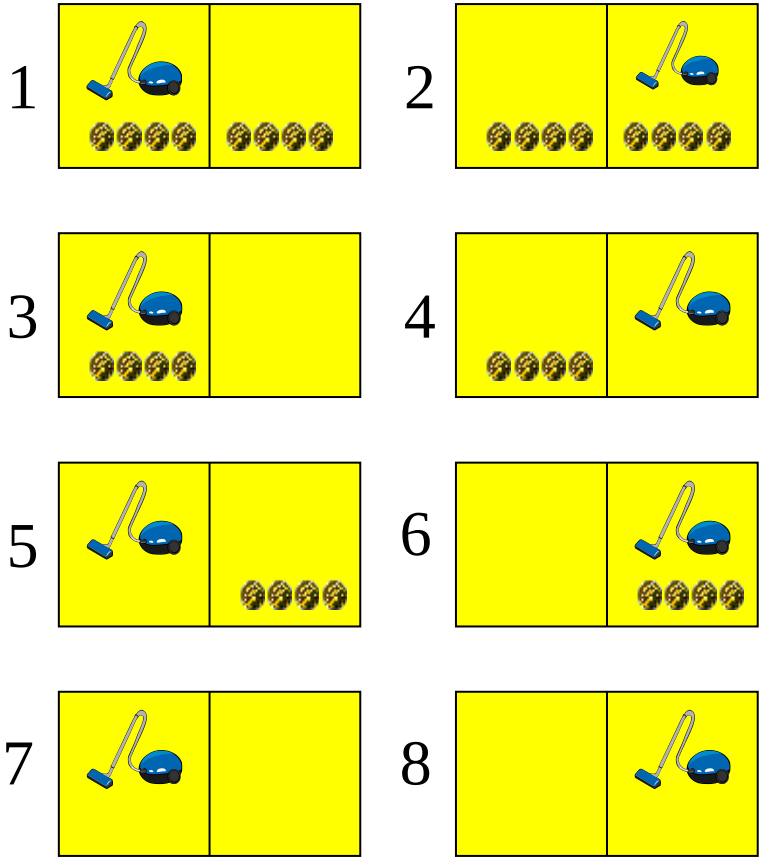
Example Problems

- Toy problems
  - Illustrate/test various problem-solving methods
  - Concise, exact description
  - Can be used to compare performance
  - *Examples*: 8-puzzle, 8-queens problem, Cryptarithmetic, Vacuum world, Missionaries and cannibals, simple route finding

- Real-world problem
  - More difficult
  - No single, agreed-upon specification (state, successor function, edgecost)
  - *Examples*: Route finding, VLSI layout, Robot navigation, Assembly sequencing

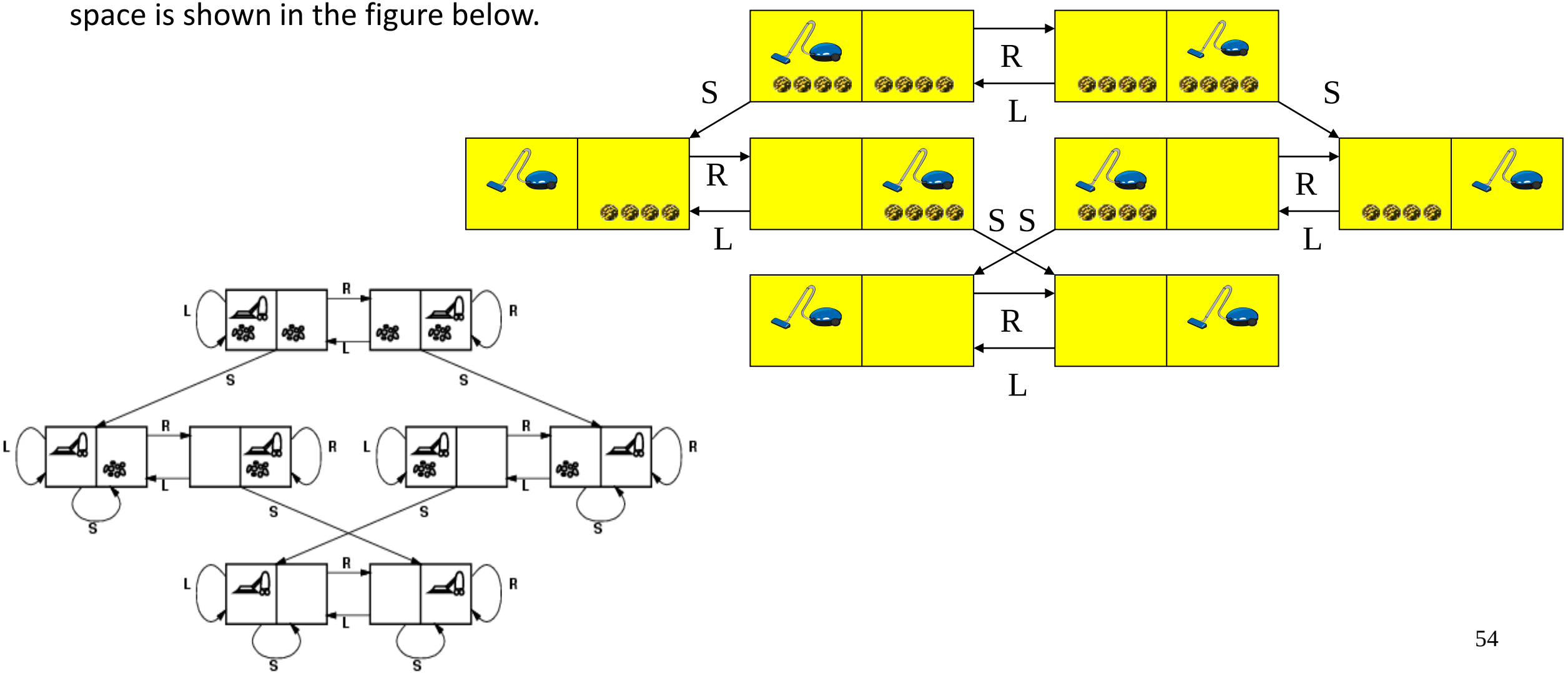
Toy Problems: The Vacuum World

- The vacuum world has only two locations. Each location may or may not contain dirt.
- States: The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus, there are $2 \times 2^2 = 8$ possible world states. A larger environment with n locations has $n \times 2^n$ states.
- Initial state: Any state can be designated as the initial state.
- Actions: In this simple environment, each state has just three actions: Left, Right, and Suck. Larger environments might also include Up and Down.
- Goal test: This checks whether all the squares are clean.
- Path cost: Each step costs 1, so the path cost is the number of steps in the path. Compared with the real world, this toy problem has discrete locations, discrete dirt, reliable cleaning, and it never gets any dirtier.



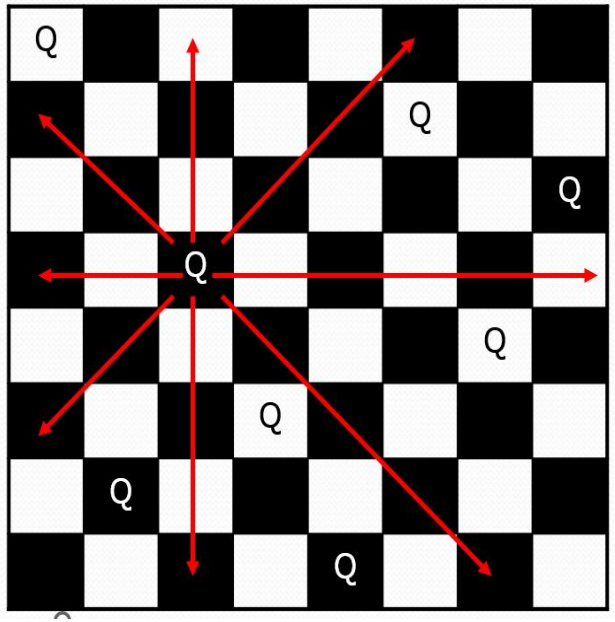
Toy Problems: The Vacuum World (cont..)

- Transition model: The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and Sucking in a clean square have no effect. The complete state space is shown in the figure below.



Toy Problems: 8-Queens

- Consider the problem of trying to place 8 queens on a chess board such that no queen can attack another queen.
- States: Any arrangement of 0 to 8 queens on the 8 x 8 board.
- Initial state: No queen on the board.
- Goal state: Placement of 8 non-attacking queens on the board.



- Mathematically, it can be expressed as below:
- If 2 queens are placed at position (i, j) and (k, l) where i and k are the row indices and j and l are the column indices then,

  $i \neq k$ ....(No same row),

  $j \neq l$ ....(No same column) and

  $| i - k | \neq | j - l |$ ....(No same diagonal) , where i, j, k, l $\in \{1, 2, \ldots, 8\}$.
- Solutions by Brute-force method, Backtracking method

Toy Problems: 8-Queens (cont..)

STEPS REVISITED – BACKTRACKING

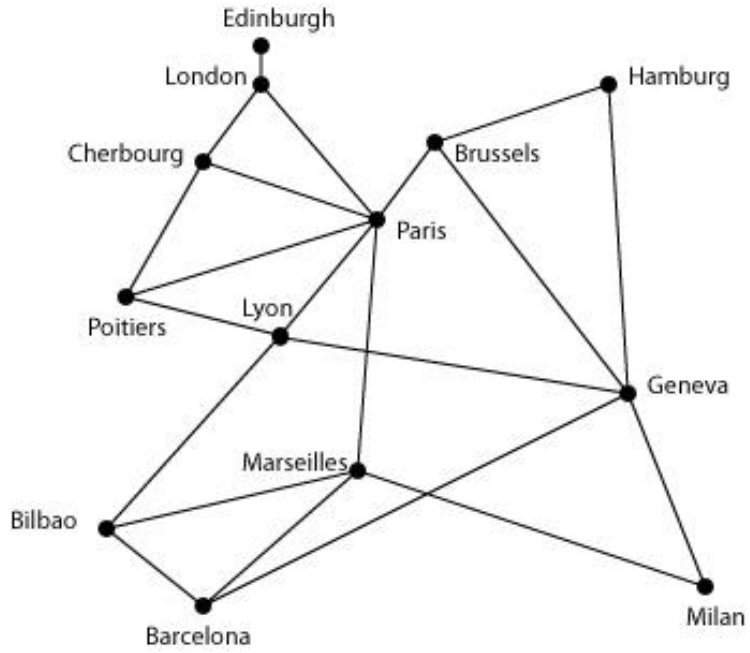
1. Place the first queen in the left upper corner of the table.
2. Save the attacked positions.
3. Move to the next queen (which can only be placed to the next line).
4. Search for a valid position. If there is one go to step 8.
5. There is not a valid position for the queen. Delete it (the x coordinate is 0).
6. Move to the previous queen.
7. Go to step 4.
8. Place it to the first valid position.
9. Save the attacked positions.
10. If the queen processed is the last stop otherwise go to step 3.

Real-World Problems

- Route finding: Defined in terms of locations and transitions along links between them.
  - *Applications*: routing in computer networks, automated travel advisory systems, airline travel planning systems
- Touring and traveling salesperson problems: Needs information about the visited cities.
  - "Visit every city on the map at least once and end in Bucharest"
  - Goal: Find the shortest tour that visits all cities
  - NP-hard, but a lot of effort has been spent on improving the capabilities of TSP algorithms
  - Applications: planning movements of automatic circuit board drills
- VLSI layout: Place cells on a chip so they don't overlap and there is room for connecting wires to be placed between the cells
- Robot navigation: Generalization of the route finding problem.
  - No discrete set of routes,
  - Robot can move in a continuous space
  - Infinite set of possible actions and states
- Assembly sequencing: Automatic assembly of complex objects.
  - The problem is to find an order in which to assemble the parts of some object.

Real-World Problems: Route Finding

- Given knowledge of direct connections, what's the shortest route which gets you from source A to destination B?
- Initial State: The starting location.
- Goal State: The final destination.
- States: The total states in the problem.
- Transition Model: How one can shift from one state to another?
- Actions: Action set, to move from one state to another.
- Path Cost: What is total efforts (cost/ distance/ time) from initial state to final goal state
- In this problem, action sequences form a tree structure.
- At some given point, certain actions are possible. These actions take you to new points.
- At each of those new points, more actions are possible. And so on.
- At each point the possible actions form a branch. Joining up the branches gives you a tree.

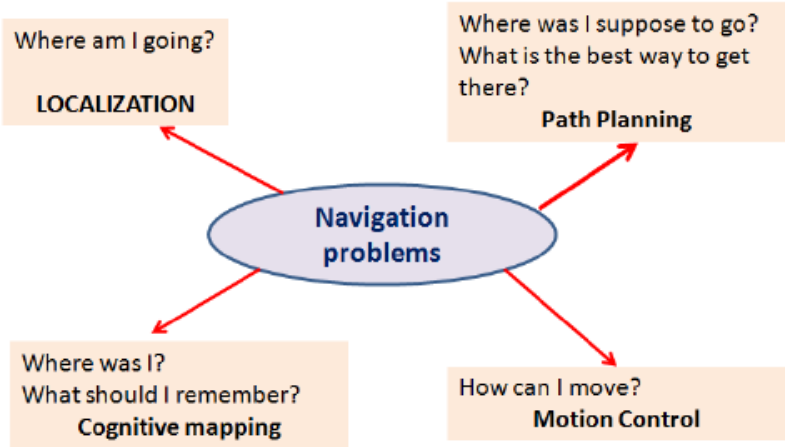


Real-World Problems: Route Finding (cont..)

- To find a solution, we need to search the tree of possible action sequences looking for one with the right start and finish. But since the tree doesn't actually exist to begin with, we will have to generate it first.
- If we are going to do this, we may as well inspect nodes as we are going along. So, in practice, “tree generation” and “search” are merged into one process.
- The two basic methods of search:
  - Depth-first search (DFS) always expand nodes at a deeper level of the tree whenever there is a choice.
  - Breadth-first search (BFS) always expand every node at the present level of the tree before moving to any deeper level.
- Other search methods like heuristics search can be also applied.

Real-World Problems: Robot Navigation

- Robot Navigation Problem aims to enable a robot to move around on the ground plane by visually detecting and avoiding obstacles.
- Robot navigation problems can be generalized into four categories:
  1. localization,
  2. path planning
  3. motion control
  4. cognitive mapping.



- Among these problems, it can be argued that path planning is the most important issue in the navigation process. It enables the selection and identification of a suitable path for the robot to traverse in the workspace area.
- Stages in Robot Navigation:
  - Camera Calibration
  - Edge & Corner Detection
  - Finding Correspondences
  - 3-D Reconstruction
  - Path Planning
  - Navigating the Path

Real-World Problems: Robot Navigation

- Graph Model of Robot Navigation:
  - Occupy one vertex of G = (V,E) at a time.
  - G usually is a grid graph where V= set of cells and Adjacency is {N,S,E,W} or chess king.
  - Robot has compass.
  - Tactile sensors detect neighbors of current vertex v; other sensors may detect more.
- Localization Problem:
  - Know graph G (input).
  - Robot is at some vertex of G.
  - Problem: determine location by moving about, making observations at each vertex visited.
  - Might conclude that robot cannot uniquely determine its location.
- Goal-directed Search Problem:
  - Know graph G=(V,E).
  - Know initial location s and goal t
  - Robot has compass, or on general graphs, can distinguish among vertices
  - Check for blocked vertices and find unblocked path from s to t.
  - There may be no unblocked s to t path

Thank you!