Artificial Intelligence: Introduction

Sunil SURVE, Professor
Fr. Conceicao Rodrigues college of engineering

What is AI?

- Simulation of human intelligence in machines that are programmed to think like humans and mimic their actions
- Carel Capek – First person to introduce idea of humanoid
- Most people hear the term artificial intelligence, the first thing they usually think of is robots
- First example of AI system is MYCIN
- IBM Watson, Deep Blue, AlphaGo etc.

Artificial Intelligence

- Based on the principle that human intelligence can be defined in a way that a machine can easily mimic it and execute tasks, from the most simple to those that are even more complex.
- The goals of artificial intelligence include learning, reasoning, and perception.
- As technology advances, previous benchmarks that defined artificial intelligence become outdated

Definition of AI

- "The exciting new effort to make computers think . . . *machines with minds,* in the full and literal sense" (Haugeland, 1985)
- "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978)

Systems that think like humans

- "The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985)
- "The study of the computations that make it possible to perceive, reason, and act" (Winston, 1992)

Systems that think rationally

- "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)
- "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991)

Systems that act like humans

- "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990)
- "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

Systems that act rationally

Definition of AI

- "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." **– Stanford**
- "Artificial Intelligence is the study of man-made computational devices and systems which can be made to act in a manner which we would be inclined to call intelligent." **– The University of Louisiana at Lafayette**
- "Defining artificial intelligence isn't just difficult; it's impossible, not the least because we don't really understand human intelligence. Paradoxically, advances in AI will help more to define what human intelligence isn't than what artificial intelligence is." **– OReilly**

Definition of AI

- "The ability of a machine communicating using natural language over a teletype to fool a person into believing it was a human. "AGI" or "artificial general intelligence" extends this idea to require machines to do everything that humans can do, such as understand images, navigate a robot, recognize and respond appropriately to facial expressions, distinguish music genres, and so on." **Matt Mahoney, PhD, Data Compression Expert**
- "The scientific understanding of the mechanisms underlying thought and intelligent behaviour and their embodiment in machines." **AITopics.org**
- "Artificial Intelligence is the science of building artificial minds by understanding how natural minds work and understanding how natural minds work by building artificial minds." **Dr. Ashok Goel, Georgia Institute of Technology**

Definition of AI

- Intelligence is a property of some entity or agent that interacts with some form of environment
- Intelligence is generally indicative of an entity's ability to succeed (by a given set of criteria) at a particular task or achieving a stated goal
- When speaking of an "authentic" intelligence, there is an emphasis on learning, adaptation, and flexibility within a wide range of environments and scenarios

"Artificial intelligence is an entity (or collective set of cooperative entities), able to receive inputs from the environment, interpret and learn from such inputs, and exhibit related and flexible behaviours and actions that help the entity achieve a particular goal or objective over a period of time." **Emerj AI Research and Advisory Company**

Limitations in Defining AI

- No one understood fully how our biological system function, especially brain. As we understand new concepts about human system functioning, the definition will change
- As technology advances, previous benchmarks that defined artificial intelligence become outdated

Strong AI versus Weak AI

- **Strong AI** – Also known as deep AI or what some might deep Artificial General Intelligence (AGI); the idea that a computer can be made or raised to intelligence levels that match human beings'
- **Weak AI** – Otherwise known as narrow AI; the idea that computers can be endowed with features that mirror or mimic thought or thinking processes, making them useful tools for figuring out how our own mind works. Narrow AI systems also enhance or augments human "intelligence" by delivering calculations, patterns and analyses more efficiently than can be done by a human brain.

AI Solutions

- **Simple** – Solutions and platforms for narrow commercial needs, such as eCommerce, network integration or resource management.
  - *Examples: Customer Relationship Management (CRM) software, Content Management System (CMS) software, automated agent technology*
- **Complex** – Involves the management and analysis of specific functions of a system (domain of predictive analytics); could include optimization of work systems, predictions of events or scenarios based on historical data; security monitoring and management; etc.
  - *Examples: Financial services, risk management, intelligent traffic management in telecommunication and energy*
- **Very Complex** – Working through the entire information collection, analysis, and management processes; the system needs to know where to look for data, how to collect, and how to analyze, and then propose suggested solutions for near and mid-term futures.
  - *Examples: Global climate analysis, military simulations, coordination and control of multi-agent systems*

AI Continuum

- **Assisted Intelligence** – Involves the taking over of monotonous, mundane tasks that machines can do more efficiently
  - *Example: Robotic Shelf Picking from IAM Robotics*
- **Augmented Intelligence** – A step up in a more authentic collaboration of "intelligence", in which machines and humans learn from the other and in turn refine parallel processes
  - *Example: Editor from NyTimes*
- **Autonomous Intelligence** – System that can both adapt over time (learn on its own) and take over whole processes within a particular system or entity.
  - *Example: NASA's Mars Curiosity Rover*

Approaches to Achieving AI

- Artificial neural networks
- Reinforcement learning
- Self-supervised learning
- Multi-agent learning
- Machine learning

Acting humanly: The Turing Test approach

- Turing Test
  - Turing defined intelligent behavior as the ability to achieve human-level performance in all cognitive tasks, sufficient to fool an interrogator
  - The computer would need to possess the following capabilities:
    - **natural language processing** to enable it to communicate successfully in English (or some other human language);
    - **knowledge representation** to store information provided before or during the interrogation;
    - **automated reasoning** to use the stored information to answer questions and to draw new conclusions;
    - **machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

Acting humanly: The Turing Test approach

- Total Turing Test
  - includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch."
  - To pass the total Turing Test, the computer will need
    - **computer vision** to perceive objects, and
    - **robotics** to move them about.

Thinking humanly: The cognitive modelling approach

- Must have some way of determining how humans think
  - through introspection
  - through psychological experiments
- Cognitive science
  - Honey bee,
  - Flock of birds,
  - Fish schools, etc.

Thinking rationally: The laws of thought approach

- Greek philosopher Aristotle – “right thinking"
- **Syllogisms** provided patterns for argument structures that always gave correct conclusions given correct premises
- For example, "Socrates is a man; all men are mortal; therefore Socrates is mortal."
- Logic
- There are two main obstacles
  - First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain.
  - Second, there is a big difference between being able to solve a problem "in principle" and doing so in practice.

Acting rationally: The rational agent approach

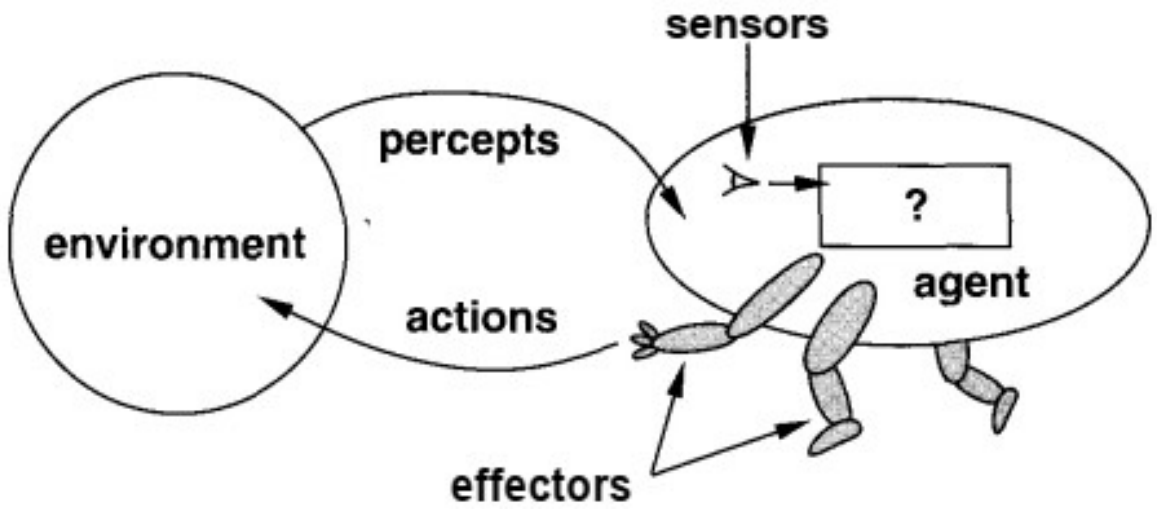
- In the "laws of thought" approach to AI, the whole emphasis was on correct inferences.
- Making correct inferences is sometimes *part* of being a rational agent, because one way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals, and then to act on that conclusion
- Correct inference is not *all* of rationality
- There are also ways of acting rationally that cannot be reasonably said to involve inference
- Rational agent design
  - It is more general than the "laws of thought" approach
  - It is more amenable to scientific development than approaches based on human behavior or human thought

Intelligent Agents

Dr. Sunil Surve
Fr. Conceicao Rodrigues College of Engineering

An Agent

- Anything that can be viewed as **perceiving** its environment through **sensors and acting** upon that environment through **effectors**
- A **rational** agent is one that does right thing
- Performance measures
  - How: the criteria that determine how successful an agent is
    - For example: amount of dirt clean, electricity consumed, consistency
  - When: duration



An Agent

- An omniscient agent knows the *actual* outcome of its actions, and can act accordingly
- An agent can not be blamed for failing to take into account something it could not perceive, or for failing to take an action that it is incapable of taking
- Rationality at any given time depends on four things:
  - The performance measure that defines degree of success
  - The **percept sequence**
  - What the agent knows about the environment
  - The actions that the agent can perform

Ideal Rational Agent

- *For each possible percept sequence, an ideal rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has*
- **Mapping** from percept sequences to actions
- *Specifying which action an agent ought to take in response to any given percept sequence provides a design for an ideal agent*
- A system is autonomous to the extent that its behavior is determined by its own experience

Structure of Intelligent Agents

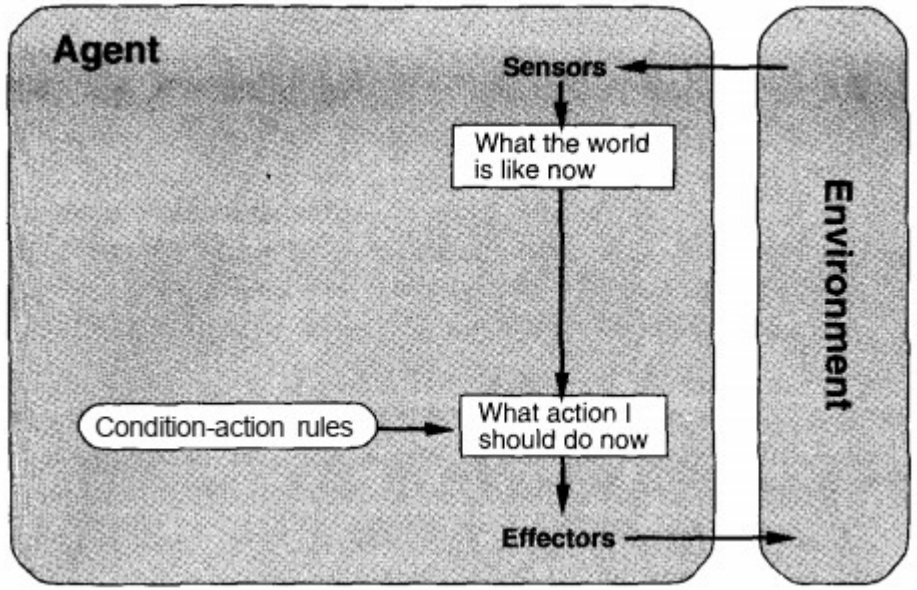
- *agent = architecture + program*
- A skeleton agent

function *SKELETON-AGENT(percept)* **returns** action
static: *memory,* the agent's memory of the world
memory — UPDATE-MEMORY(memory, *percept)*
action <— CHOOSE-BEST-ACTION(memory)
memory — UPDATE-MEMORY(memory, *action)*
return *action*

Type of Agents

- **Simple Reflex Agents**
- **Agents that keep track of the world**
- **Goal-based agents**
- **Utility-based agents**

Simple Reflex Agents

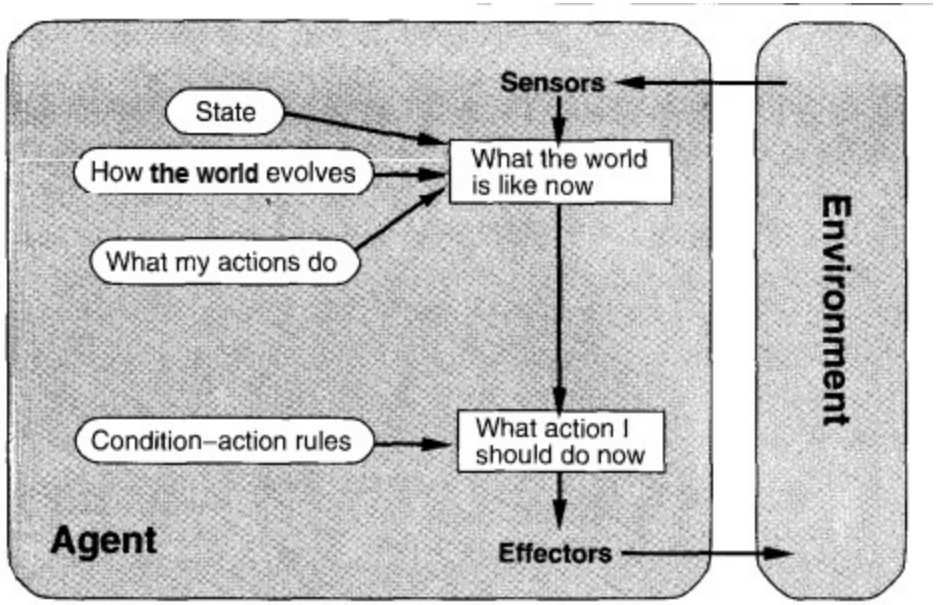


function SIMPLE-REFLEX-AGENT (percept)**returns** *action*

static: *rules,* a set of condition-action rules

state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, *rules)*
action ← RULE-ACTION(rule)
return *action*

Agents that keep track of the world

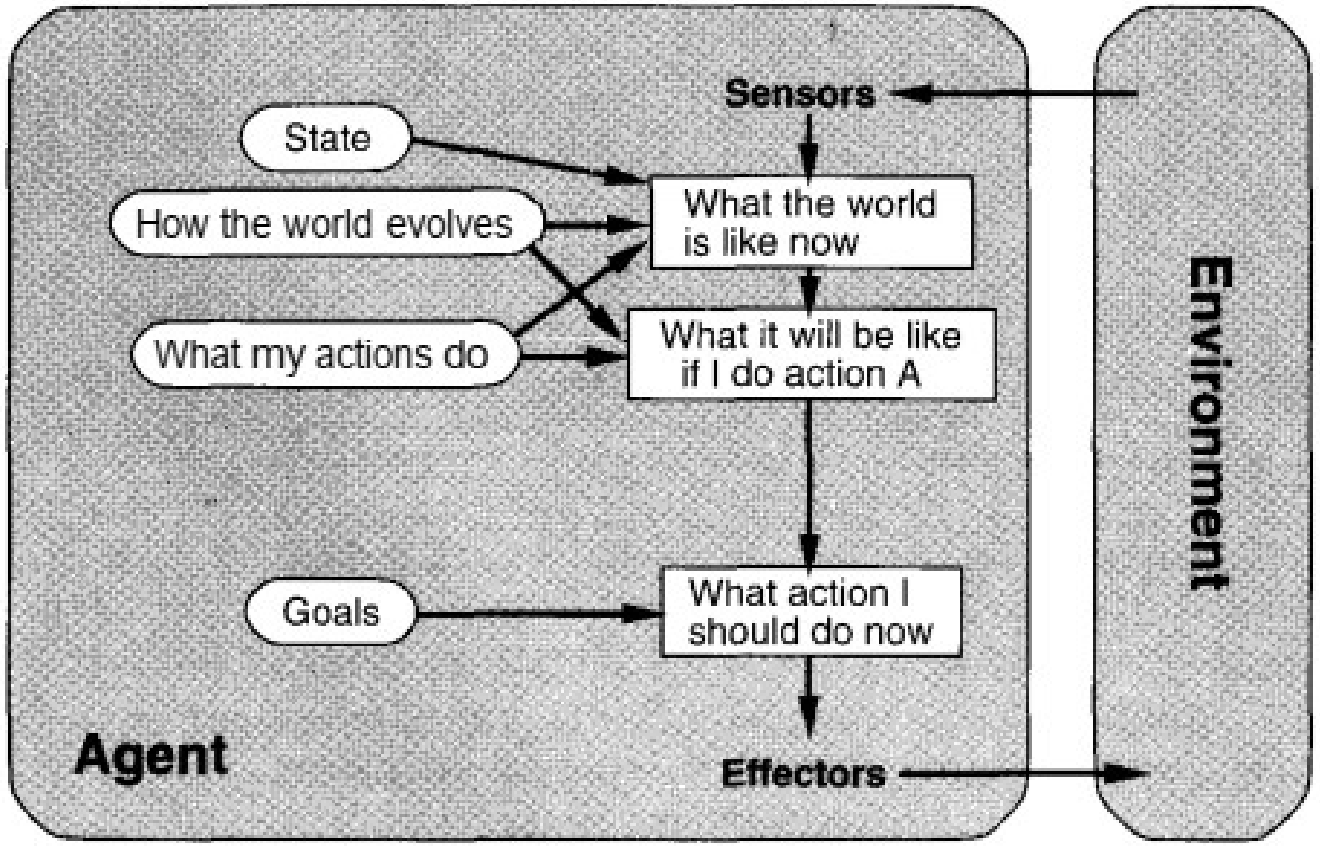


function REFLEX-AGENT-WITH-STATE(percept) **returns** action

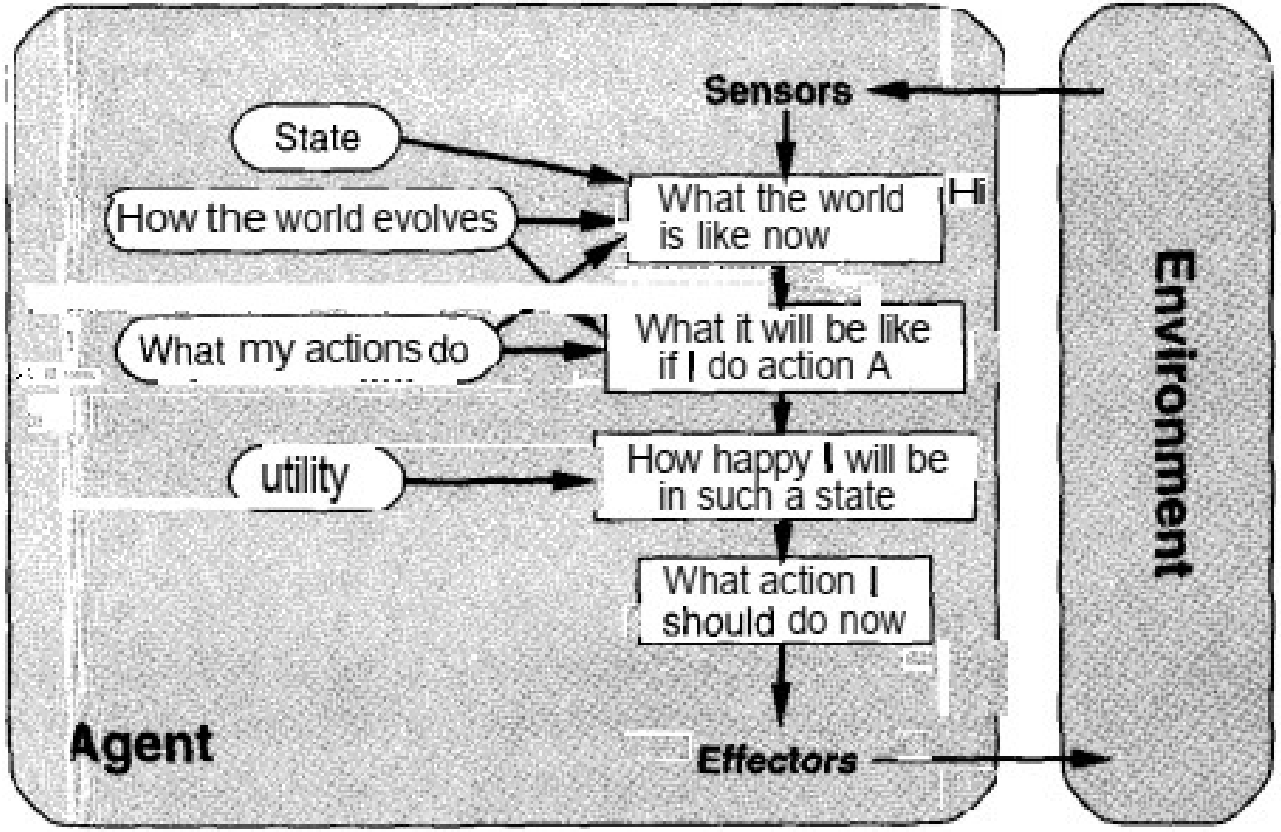
static: *state,* a description of the current world state
rules, a set of condition-action rules

state ← UPDATE-STATE(state, *percept)*
rule — RULE-MATCH(state, *rules)*
action — RULE-ACTION[rule]
state ← UPDATE-STATE(state, *action)*
return *action*

Goal-based agents



Utility-based agents



Environments - Properties

- **Accessible** vs. **inaccessible**
  - Sensors gives complete information about state of agent
  - Sensors detects all aspects that are relevant to the choice of action
  - Agent need not maintain any internal state to keep track of the world
- **Deterministic** vs. **nondeterministic**
  - Next state of the environment is completely determined by the current state and the actions selected
  - No worry about uncertainty in an accessible, deterministic environment
  - Inaccessible → nondeterministic
  - Complex environment → hard to keep track of all the inaccessible aspects

Environments - Properties

- **Episodic** vs. **non-episodic**
  - In each episode, agent acts based on perception
  - Actions in next episodes does not depend on actions in previous episodes
  - Agent does not think ahead in episodic environment
- **Static** vs. **dynamic**
  - State changes with time irrespective of the agent's actions → dynamic environment
  - State does not change with time → static environment
  - State changes with action of an agent → semi-dynamic environment

Environments - Properties

- **Discrete** vs. **continuous**
  - Chess
  - Taxi Driver
- Examples:

Environment	Accessible	Deterministic	Episodic	Static	Discrete
Chess with a clock	Yes	Yes	No	Semi	Yes
Chess without a clock	Yes	Yes	No	Yes	Yes
Taxi driving	No	No	No	No	No
Part-picking robot	No	No	Yes	No	No
Image-analysis system	Yes	Yes	Yes	Semi	No

Environment programs

function RUN-EVAL-ENVIRONMENT(state, UPDATE-FN, *agents, termination,* PERFORMANCE-FN)
returns *scores*

local variables: *scores,* a vector the same size as *agents,* all 0

    repeat
        for each *agent* **in** *agents* **do**
        PERCEPT[agent] ← GET-PERCEPT(agent, *state)*
    end
    for each *agent* **in** *agents* **do**
        ACTTON[agent] ← *PROGRAM[agent](PERCEPT[agent])*
    end
state ← UPDATE-FN(action, *agents, state)*
scores ← PERFORMANCE-FN(actions, *agents, state)*
until *termination(state)*
return *scores*

Summary

- **An agent** is something that perceives and acts in an environment.
- **An ideal agent** is one that always takes the action that is expected to maximize its performance measure, given the percept sequence it has seen so far
- An agent is **autonomous** to the extent that its action choices depend on its own experience, rather than on knowledge of the environment that has been built-in by the designer.
- **An agent program** maps from a percept to an action, while updating an internal state
- **Reflex agents** respond immediately to percepts, **goal-based agents** act so that they will achieve their goal(s), and **utility-based agents** try to maximize their own "happiness."

Solving Problems by Search

Sunil Surve
Fr. Conceicao Rodrigues College of Engineering

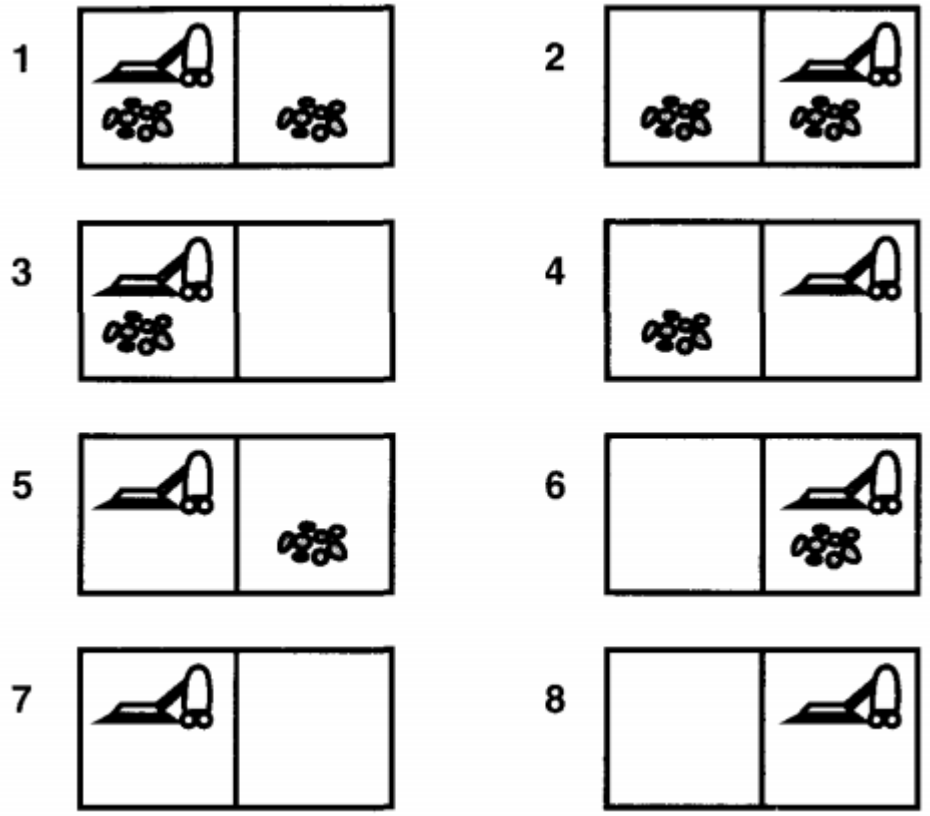
A simple problem-solving agent

function SIMPLE-PROBLEM-SOLVING-AGENT(P) **returns** an action
inputs: *p, a* percept
static: *s,* an action sequence, initially empty
state, some description of the current world state
g, a goal, initially null
problem, a problem formulation

state ← UPDATE-STATE(state, *p)*
if *s* is empty then
g — FORMULATE-GOAL(state)
problem ← FORMULATE-PROBLEM(state, *g)*
s ← *SEARCH(problem)*
action — RECOMMENDATION^, *state)*
s ← REMAINDER(s, *state)*
return *action*

Knowledge and problem types

- Actions: *Left, Right,* and *Suck*
- **Single-state problem**: Accessible and effect of actions
- **Multiple-state problem**: Non-accessible and effect of actions
- **Contingency problem**



Well-defined problems

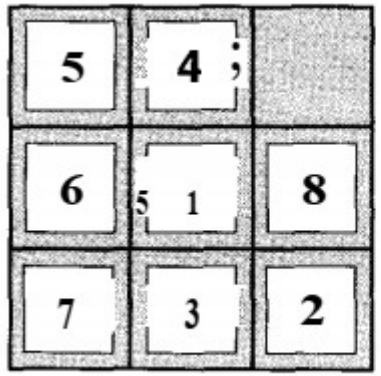
- State Space
  - **The initial state**
  - The set of possible actions available to the agent.
- A **path** in the state space is simply any sequence of actions leading from one state to another
- **The goal test**
- **A path cost** function
- Together, the initial state, operator set, goal test, and path cost function define a problem

Measuring problem-solving performance

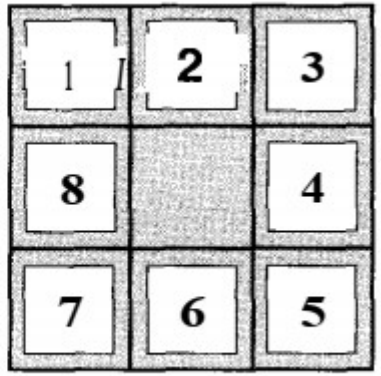
- Does it find a solution at all?
- Is it a good solution (one with a low path cost)?
- What is the **search cost** associated with the time and memory required to find a solution?
- The **total cost** of the search is the sum of the path cost and the search cost.

The 8-puzzle

- States: a state description specifies the location of each of the eight tiles in one of the nine squares
- **Operators:** blank moves left, right, up, or down
- **Goal test:** state matches the goal configuration
- **Path cost:** each step costs 1, so the path cost is just the length of the path



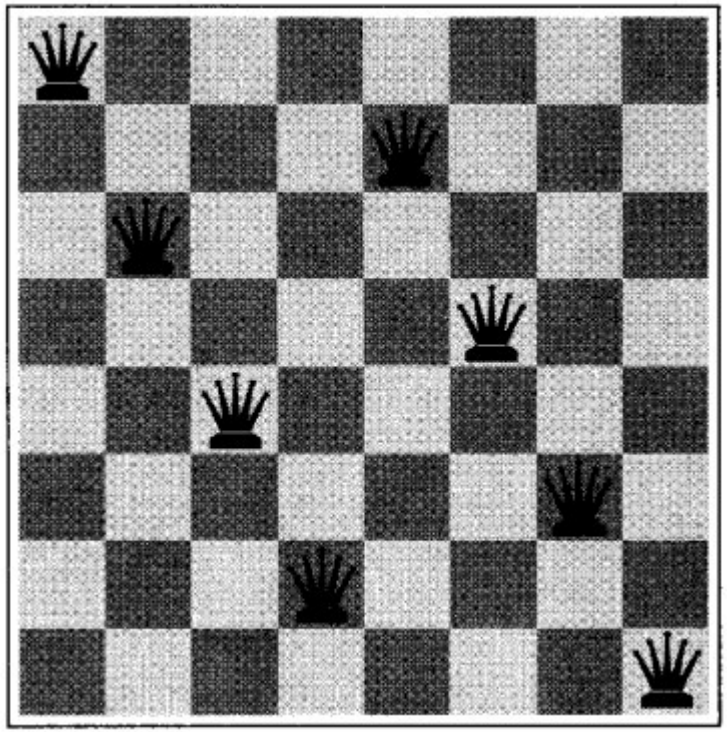
Start State



Goal State

The 8-queens problem

- **States:** any arrangement of 0 to 8 queens on board
- **Operators:** add a queen to any square
- **Path** cost: zero
- **Goal** test: 8 queens on board, none attacked



Searching for solutions

Function GENERAL-SEARCH(*problem*, *strategy* r**eturns** a solution, or failure

initialize the search tree using the initial state of *problem*

loop do

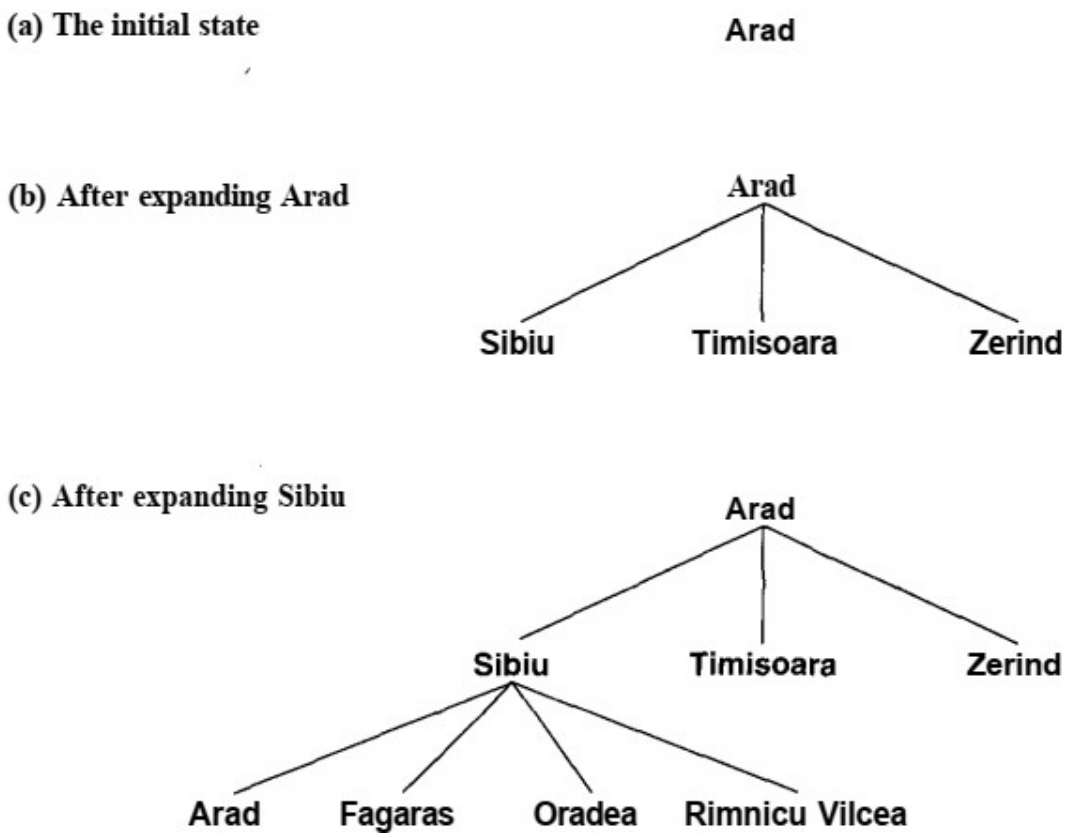
if there are no candidates for expansion **then return** failure

choose a leaf node for expansion according to *strategy*

if the node contains a goal state **then return** the corresponding solution

else expand the node and add the resulting nodes to the search tree

end



Data structures for search trees

- Data structure have five components:
- the state in the state space to which the node corresponds;
- the node in the search tree that generated this node (this is called the **parent node);**
- the operator that was applied to generate the node;
- the number of nodes on the path from the root to this node (the **depth** of the node);
- the path cost of the path from the initial state to the node.

function GENERAL-SEARCH(problem, QUEUING-FN) **returns** a solution, or failure
nodes ← MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[problem]))
loop do
if *nodes* is empty **then return** failure
node ← REMOVE-FRONT(nodes)
if GOAL-TEST[problem] applied to STATE(node) succeeds **then return** *node*
nodes ← QUEUING-FN(nodes, *EXPAND(node, OPERATORS[problem]))*
end

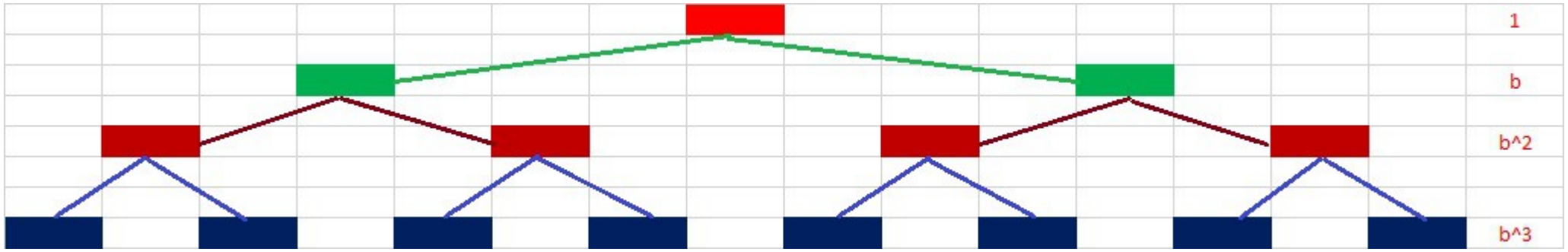
The general search algorithm

Search Strategy

- **Completeness:** is the strategy guaranteed to find a solution when there is one?
- **Time complexity:** how long does it take to find a solution?
- **Space complexity:** how much memory does it need to perform the search?
- **Optimality:** does the strategy find the highest-quality solution when there are several different solutions?

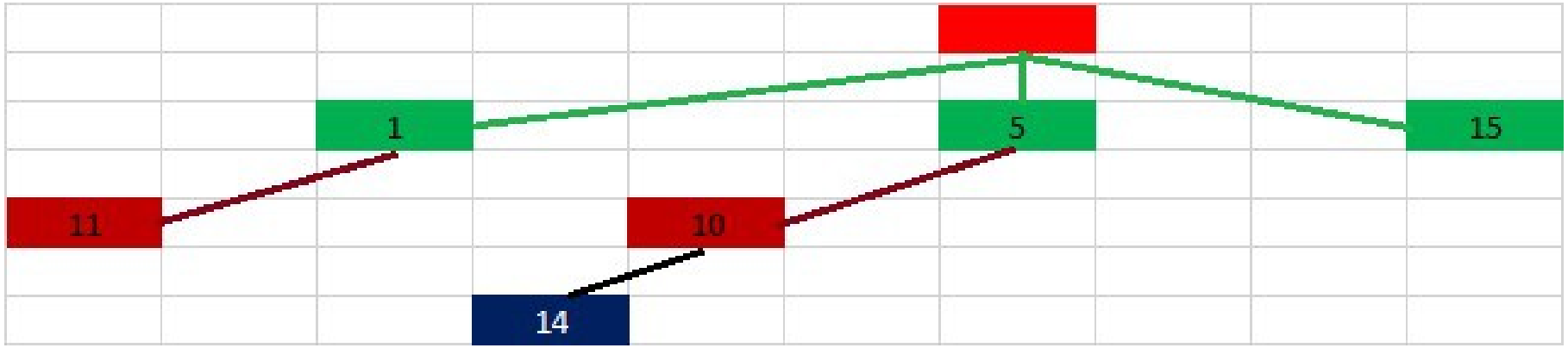
Breadth-first search

- Let branching factor b and depth d
- Total number of nodes:
  1 + b + b2 + b3 + ….. + bd
- Complete, optimal
- High time and space complexity



Uniform cost search

- Expands lowest cost node
- Complete, optimal
- Memory and space complexity low

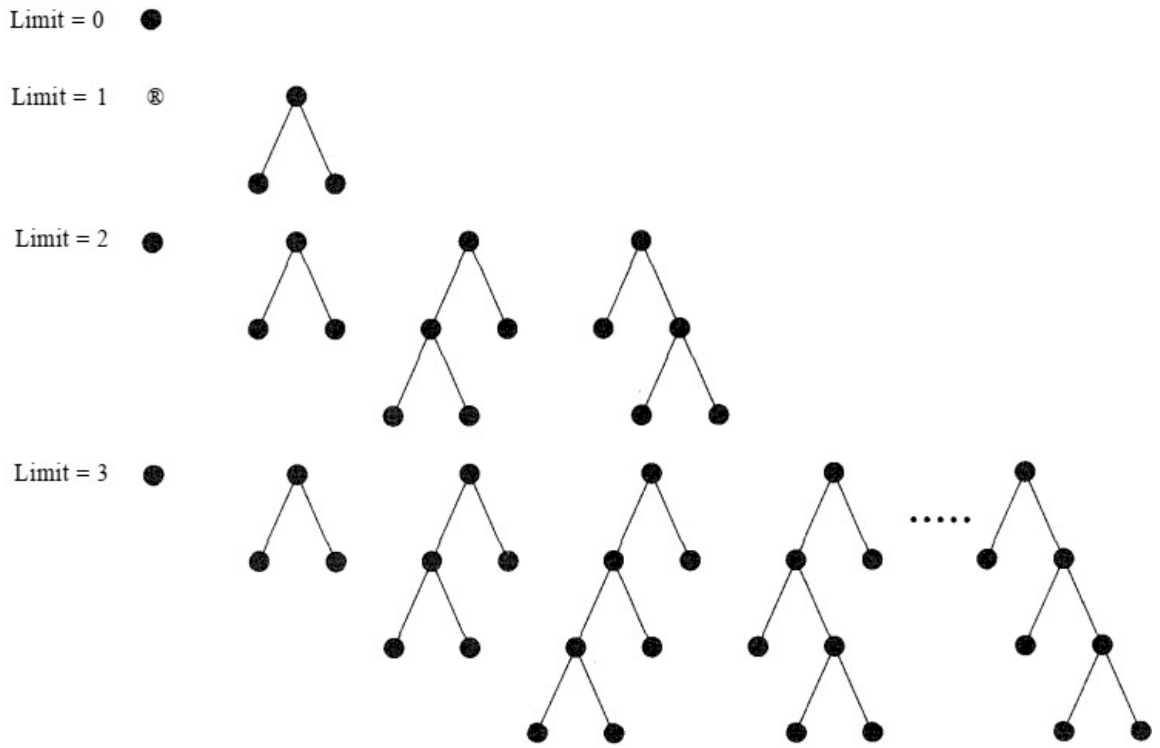


Depth-first search

- **Depth-first search** always expands one of the nodes at the deepest level of the tree.
- Only when the search hits a dead end (a non-goal node with no expansion) does the search go back and expand nodes at shallower levels.
- The time complexity for depth-first search is *O(bm)*.
- Complete, optimal

Variants of DFS

- **Depth-limited search**
- **Iterative deepening search**



Comparison

Criterion	Breadth-first	UCS	Depth-first	Depth Limited	Iterative deepening
Time	bd	bd	bm	bl	bd
Space	bd	bd	bm	bl	Bd/2
Optimal	Yes	Yes	No	No	Yes
Complete	Yes	Yes	No	Yes if l > d	Yes

Summary

- Before an agent can start searching for solutions, it must formulate a goal and then use the goal to formulate a problem
- **A problem** consists of four parts: the **initial state, a set** of **operators, a goal test** function, and a **path cost** function. The environment of the problem is represented by a **state space. A path** through the state space from the initial state to a goal state is a **solution**
- Search algorithms are judged on the basis of **completeness, optimality, time complexity, and space complexity.** Complexity depends on *b,* the branching factor in the state space, and *d,* the depth of the shallowest solution
- **Breadth-first search** expands the shallowest node in the search tree first. It is complete, optimal for unit-cost operators, and has time and space complexity of *O(bm)*. The space complexity makes it impractical in most cases

Summary

- **Uniform-cost search** expands the least-cost leaf node first. It is complete, and unlike breadth-first search is optimal even when operators have differing costs. Its space and time complexity are the same as for breadth-first search
- **Depth-first search** expands the deepest node in the search tree first. It is neither complete nor optimal, and has time complexity of *0(bm)* and space complexity of *O(bm),* where *m* is the maximum depth. In search trees of large or infinite depth, the time complexity makes this impractical.

Summary

- **Depth-limited search** places a limit on how deep a depth-first search can go. If the limit happens to be equal to the depth of shallowest goal state, then time and space complexity are minimized
- **Iterative deepening search** calls depth-limited search with increasing limits until a goal is found. It is complete and optimal, and has time complexity of $O(bd)$ and space complexity of $O(bd)$.

INFORMESD SEARCH METHODS

Sunil Surve
Fr. Conceicao Rodrigues College of ENgineering

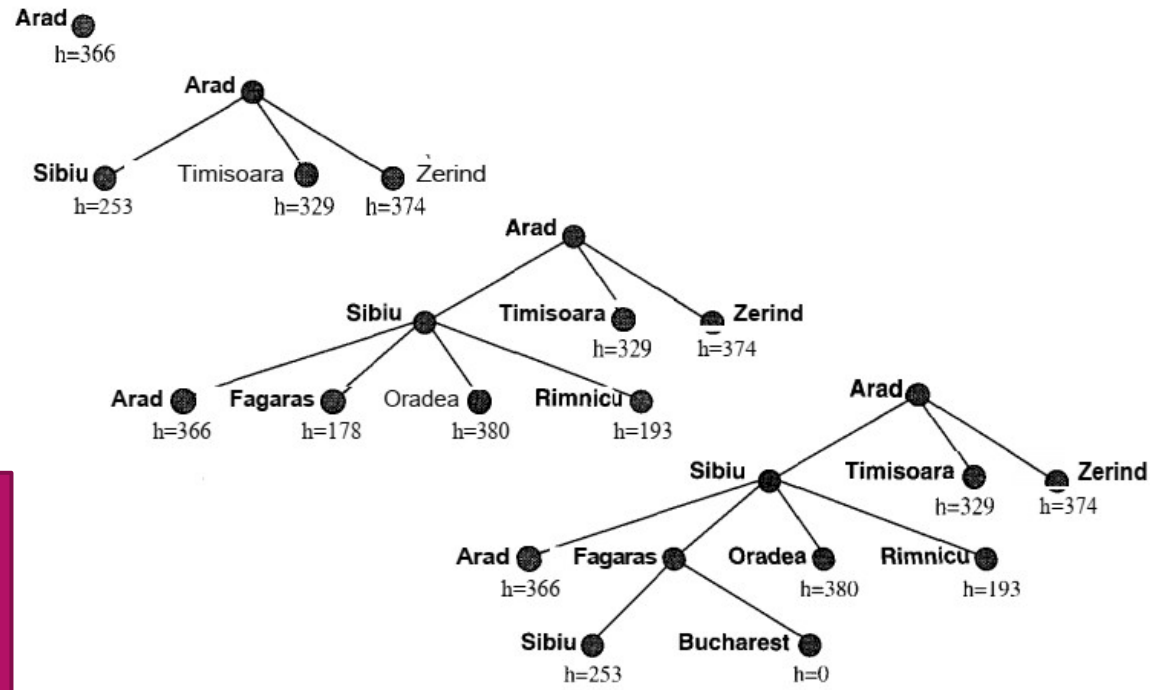
BEST-FIRST SEARCH

- Nodes are ordered so that the one with the best evaluation is expanded first

function BEST-FIRST-SEARCH(problem, EVAL-FN)
returns a solution sequence
    inputs: *problem,* a problem
        Eval-Fn, an evaluation function
    Queueing-Fn ← a function that orders nodes by

Greedy search

- **Heuristic function**
  h(n) = estimated cost of the cheapest path from the state at node *n* to a goal state
- The worst-case time and space complexity for greedy search is *O(bm)*, where *m* is the maximum depth of the search space



function GREEDY-SEARCH(problem)
returns a solution or failure
return BEST-FIRST-SEARCH(problem, *h)*