What is Artificial Intelligence?

→ A branch of computer science dealing with the simulation of intelligent behavior in computers.
→ The capability of a machine to imitate intelligent human behavior.

Examples:

→ Google's AI-Powered Predictions.

Using anonymized location data from smart-phones, Google Maps (Maps) can analyze the speed of movement of traffic at any given time. And, with its acquisition of crowdsourced traffic app Waze in 2013, Maps can more easily incorporate user-reported traffic incidents like construction and accidents. Access to vast amounts of data being fed to its proprietary algorithms means Maps can reduce commutes by suggesting the fastest routes to and from work.

→ Spam Filters

Your email inbox seems like an unlikely place for AI, but the technology is largely powering one of its most important features: the spam filter. Simple rules-based filters (i.e. "filter out messages with the words 'online pharmacy' and 'Nigerian prince' that come from unknown addresses") aren't effective against spam, because spammers can quickly update their messages to work around them. Instead, spam filters must continuously *learn* from a variety of signals, such as the words in the message, message metadata (where it's sent from, who sent it, etc.).

→ Smart Email Categorization

Gmail uses a similar approach to categorize your emails into primary, social, and promotion inboxes, as well as labeling emails as important. In a research paper titled, "The Learning Behind Gmail Priority Inbox", Google outlines its machine learning approach and notes "a huge variation between user preferences for volume of important mail. Thus, we need some manual intervention from users to tune their threshold. When a user marks messages in a consistent direction, we perform a real-time increment to their threshold.**"** Every time you mark an email as important, Gmail learns.

→ Plagiarism Checkers

Many high school and college students are familiar with services like *Turnitin*, a popular tool used by instructors to analyze students' writing for plagiarism. While *Turnitin* doesn't reveal precisely how it detects plagiarism, research demonstrates how ML can be used to develop a plagiarism detector.

→ Fraud Prevention

How can a financial institution determine if a transaction is fraudulent? In most cases, the daily transaction volume is far too high for humans to manually review each transaction. Instead, AI is used to create systems that learn what types of transactions are fraudulent.

→ Facebook

When you upload photos to Facebook, the service automatically highlights faces and suggests friends to tag. How can it instantly identify which of your friends is in the photo? Facebook uses AI to recognize faces. In a short video highlighting their AI research (below), Facebook discusses the use of artificial neural networks—ML algorithms that mimic the structure of the human brain—to power facial recognition software.

School of Thoughts:

Acting Humanly	Acting Rationally
Thinking Humanly	Thinking Rationally

→ Acting Humanly:

One of the most popular approaches to the definition of AI is the Turing Test, proposed by Alan Turing (1950). It explains whether a machine acts humanly. Imagine a large box in which a man or a machine could hide. You can exchange written messages via a letter slot with the "system" behind the box. Now your task is to write questions and messages to the secret system in the box and determine if it is a human or an AI. The AI would pass the test if the human questioner cannot tell whether he is communicating with a machine or a human being. According to the definition of Turing, a computer that is capable of passing this test, possesses an artificial intelligence. According to Russel and Norvig an AI would need the following capabilities to process the messages and return answers:

- **Natural Language Processing**: in order to understand and use language to communicate,
- **Knowledge Representation**: so that the AI can store what it knows and hears,
- **Automated Reasoning**: which means the AI is able to use the stored information and derive knowledge from it,
- **Machine Learning**: in order to learn from past input and actions, so the AI can adapt to a changing environment and detect patterns.

Moreover the **Total Turing test** additionally requires the human interrogator to see the AI system through video and give it mechanical tasks. In order to pass this far more complicated challenge, the AI would need:

- **Computer Vision**: the ability to perceive objects in the environment,
- **Robotics**: in order to fulfil mechanical tasks.

This illustrative definition approach is still relevant today and led to the creation of the six main disciplines in the field of AI till today (As elaborated Above).

→Thinking Humanly:

The basic idea is the following: If we understand how the human brain works, we can simulate or rebuild it. Through psychological experiments, introspection and brain imaging we can try to gain insights about the mechanisms and patterns in the human mind. Mostly cognitive scientists follow this approach, but also psychologists and neuroscientists contribute on that field.

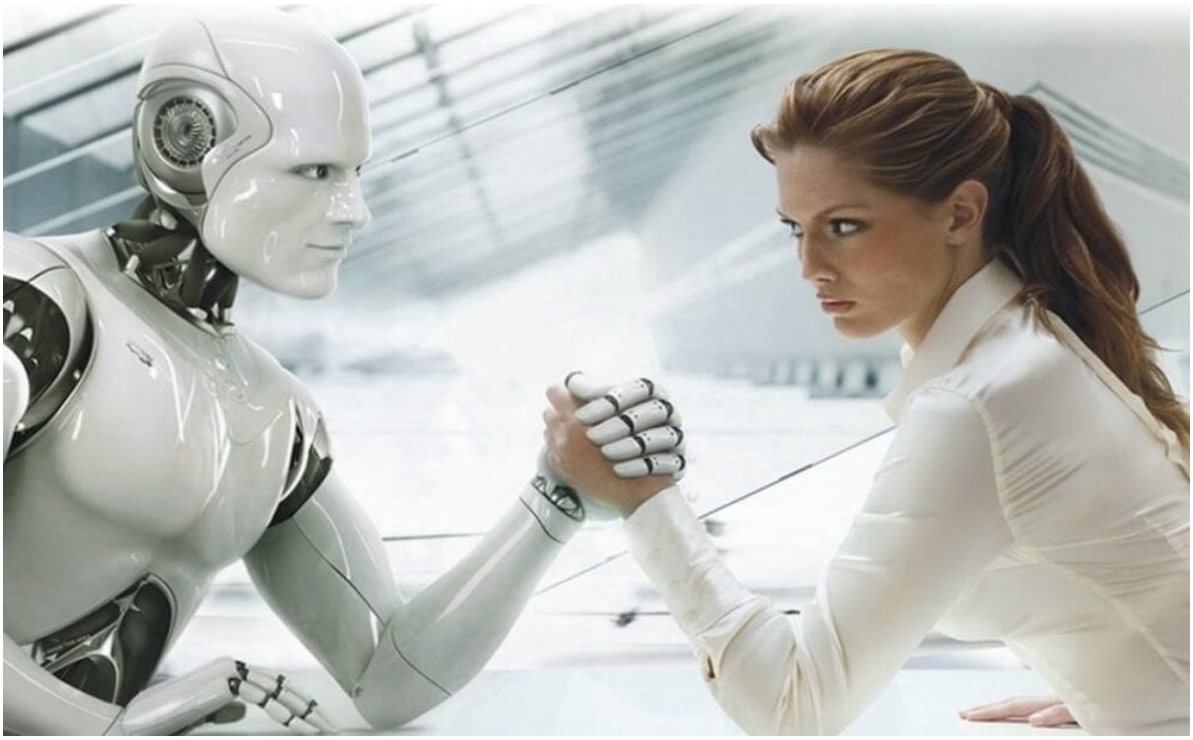
→ Acting Rationally:

The last school of thought is the most modern approach in AI. It tries to define AI, with the concept of so called **rational agents**. An agent is just something that acts and a rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome (Russel and Norvig, 2016). This approach tends to be better than the previous schools of thought, because it is more inclusive and general, since it allows logical inference to be used as a conclusion drawing method but also opens the door for a variety of other mechanisms to achieve rationality.

→ Thinking Rationally:

This approach is mainly based on logic and has its routes in the Greece philosophy. A logic-based AI system uses a set out of rules, so called syllogisms, which it uses to draw conclusions. For example: "Socrates is a man; all men are mortal; therefore, Socrates is mortal." But this school of thought comes with its limits in practice. First, it is difficult to know all the rules in a complex world from which to draw logical conclusions. Second, the translation of perceived informal knowledge into logical rules is not as easy, as one might think, especially if one cannot be certain if the knowledge is true.

Strong AI vs Weak AI:



Strong AI	Weak AI
Self Improvement	No or minor Self Improvement
Decision Making	No or minor Decision Making
Less Manual	More Manual

Weak AI

Siri and Alexa could be considered AI, but generally, they are weak AI programs. Even advanced chess programs are considered weak AI. Voice-activated assistance and chess programs often have a programmed response. They are sensing for things similar to what they know, and classifying them accordingly. This presents a human-like experience, but that is all it is a simulation. If you ask Alexa to turn on the TV, the programming understands key words like On and TV. The algorithm will respond by turning on the TV, but it is only responding to its programming. In other words, it does not comprehend any of the meaning of what you said.

Strong AI

Featured in many movies, strong AI acts more like a brain. It does not classify, but uses clustering and association to process data. In short, it means there isn’t a set answer to your keywords. The function will mimic the result, but in this case, we aren’t certain of the result. Like talking to a human, you can assume what someone would reply to a question with, but you don’t know. For example, a machine might hear “good morning” and start to associate that with the coffee maker

turning on. If the computer has the ability, it theoretically could hear “good morning” and decide to turn on the coffee maker.

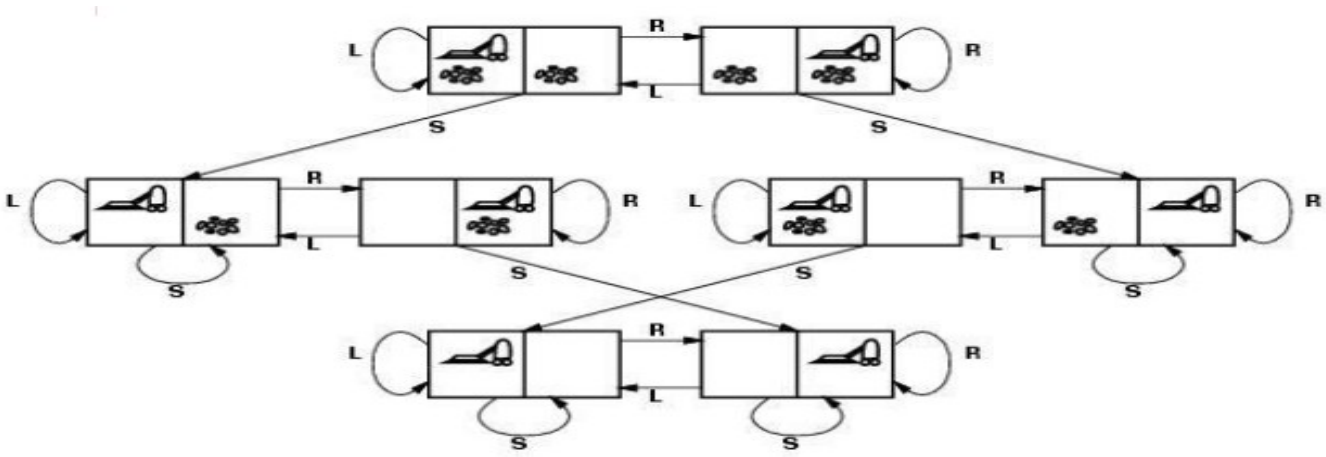
Another example is AI in games. In one example, an AI program taught itself to play 49 classic Atari games. When the program was instructed to obtain the highest score it could in the game Breakout, it was able to outperform humans in just 2.5 hours. Researchers let the program continue and to their surprise, the program developed a strategy that was not programmed into the system.

Foundations of A.I

→ Philosphy
→ Math-e-matics
→ Economics
→ Neuroscience
→ Psychology
→ Computer Engineering
→ Control theory and Cybernetics
→ Linguistics

Intelligent Agents:

Agent?
What it does?
Types?
Vacuum World Problem:



Nature of Environment (PEAS)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

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

Properties of Environment:

→ Discrete / Continuous
→ Observable / Partially Observable
→ Static / Dynamic
→ Single agent / Multiple agents
→ Accessible / Inaccessible
→ Deterministic / Non-deterministic
→ Episodic / Non-episodic

Problem Solving :

→ Representing/defining a problem.

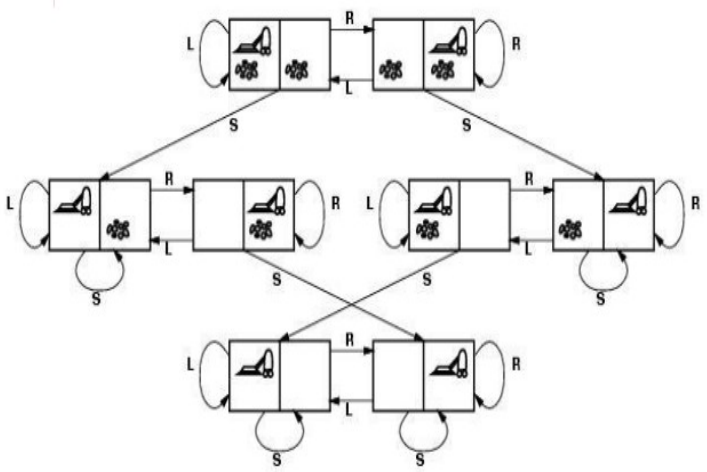
A problem can be defined using following components:

→ Initial State
→ Actions
→ State Space
→ Transitions
→ Goal State
→ Solution
→ Goal Test
→ Path
→ Path Cost
→ Optimal Solution
→ Search

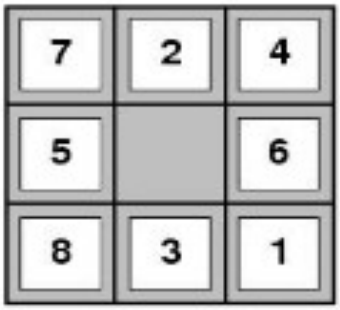
State:

→ A Problem is defined by its elements and relations.
→ A State is representation of those elements in a given moment.

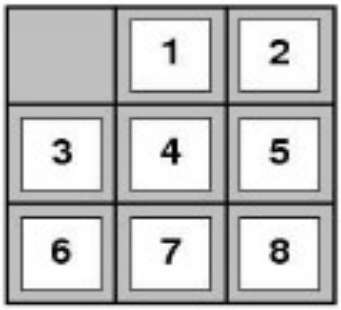
Vacuum Cleaner Problem:



States: Two locations with or without dirt: $2x2^2 = 8$
Initial State: Any state can be initial.
Actions: { Left, Right, Suck }
Transition model: Left, Right, and Suck, moves left, moves right, and suctions respectively. If in rightmost square, Right has no effect. In leftmost square, Left has no effect. Suck in a clean square has no effect.
Goal State: Both rooms clean.
Path: Series of Transitions adopted at a particular moment.
Solution: Series of Transitions that let to clean rooms.
Goal test: Check whether squares are clean.
Path cost: Each step costs 1, so path cost is the number of steps.
8-Puzzle Game:



Start State



Goal State

States: Integer location of each tile
Initial State: Any state can be initial.
Actions: Move blank { Left, Right, Up, Down}
Transition model: given a state and an action, return the resulting state.
Goal Test: Check whether goal configuration is reached.
Path cost: Number of steps to reach goal.
Water Jug Problem:
A Water Jug Problem: You are given two jugs, a 4-gallon (A) one and a 3-gallon (B) one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the jug 'A'?

Do it yourself.

Missionaries, Cannibal Problem.
Tower of Hanoi.
8-Queen Problem.

Search Strategies

Why needed?

Parameters to analyze Search Strategies:

Strategies are evaluated along the following criteria:

→ **Completeness:** does it always find a solution if one exists?
→ **Optimality:** does it always find a least cost solution?
→ **Time complexity:** number of nodes generated
→ **Space complexity:** maximum number of nodes in memory

Un-informed (Blind Searches) vs. informed Searches

Un-informed Searches:

→ Depth First Search (DFS).
→ Breadth First Search (BFS).
→ Iterative Deepening Search (IDS).
→ Uniform Cost Search (UCS).
→ Bidirectional Search (BS).

Informed Searches:

→ Greedy Best First Search
→ A* Search

Depth First Search:

Complete	No in case of tree, Yes in case of graph
Optimal	No, returns the first solution it finds
Time Complexity	$O(b^m)$
Space Complexity	O(bm)

Breadth First Search:

Complete	Yes
Optimal	Yes
Time Complexity	O(V+E)
Space Complexity	O(V)

Iterative Deepening Search:

Complete	Yes
Optimal	Yes
Time Complexity	O(bd)
Space Complexity	O(bd)

Uniform Cost Search:

Complete	Yes, If Cost of every step is a positive number
Optimal	Yes

Time Complexity	O(b^[1+(C*/ e)])
Space Complexity	O(b^[1+(*C**/*e*)])

Bi-Directional Search:

Complete	Yes, in case of BFS
Optimal	Yes, in case of BFS
Time Complexity	$O(b^{d/2})$
Space Complexity	$O(b^{d/2})$

Informed Searches:

→ Greedy Best First Search
→ A* Search

Greedy Best First Search:

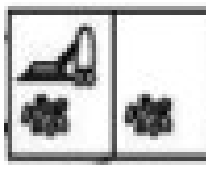
Complete	No, can get Stuck in loops
Optimal	No
Time Complexity	$O(b^m)$, but a good heuristic can give dramatic improvement
Space Complexity	$O(b^m)$, keeps all nodes in memory

A* Search:

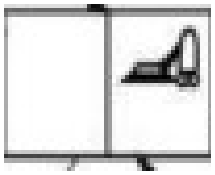
Complete	Yes
Optimal	Yes, (depending upon search algo and heuristic property)
Time Complexity	$O(b^d)$
Space Complexity	$O(b^d)$, keeps all nodes in memory

Assignment:

Solve Vaccume cleaner Problem using Depth First Search if:



Drawing 1: Start State



Drawing 2: Goal State

Constraint Satisfaction Problem:

A constraint satisfaction problem (CSP) is a problem that requires its solution within some limitations/conditions also known as constraints.
It consists of the following:

- A finite set of **variables** which stores the solution. (V = {V1, V2, V3,....., Vn})
- A set of **discrete** values known as **domain** from which the solution is picked. (D = {D1, D2, D3,.....,Dn})
- A finite set of **constraints.** (C = {C1, C2, C3,....., Cn})

Converting problems to CSPs

A problem to be converted to CSP requires the following steps:

- **Step 1:** Create a variable set.
- **Step 2:** Create a domain set.
- **Step 3:** Create a constraint set with variables and domains (if possible) after considering the constraints.
- **Step 4:** Find an optimal solution.

Examples:

1. Map Coloring Problem

2. Graph Coloring Problem

3. n-Queen Problem

4. Scheduling Problems

Varieties of constraints:

Unary constraints involve a single variable, e.g., SA ≠ green

Binary constraints involve pairs of variables, e.g., SA ≠ WA

Higher-order constraints involve 3 or more variables.
e.g., SA≠WA≠NT

Backtracking Search:

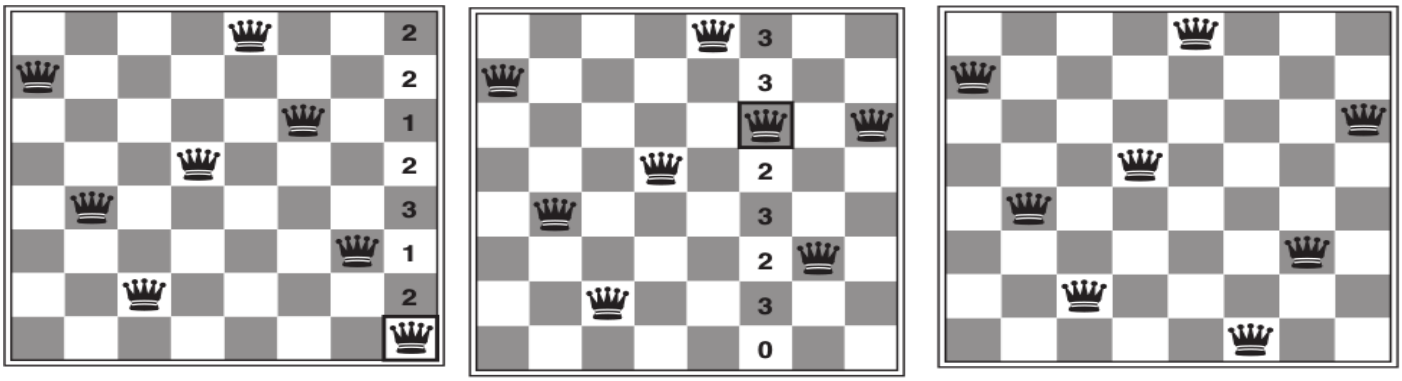
Algorithm repeatedly chooses an unassigned variable, and then tries all values in the domain of that variable in turn, trying to find a solution. If an inconsistency is detected, then algorithm returns failure, causing the previous call to try another value.

Brute Force vs. Backtracking.

Local Search:

Local search is a method for finding a solution to a CSP instance.
It's based on the idea of starting with a complete assignment of variables, usually randomlly generated, and working step by step toward a solution by changing the variable assignments.
The steps usually involve changing one variable at a time, often using some heuristic such as the number of unsatisfied constraints.

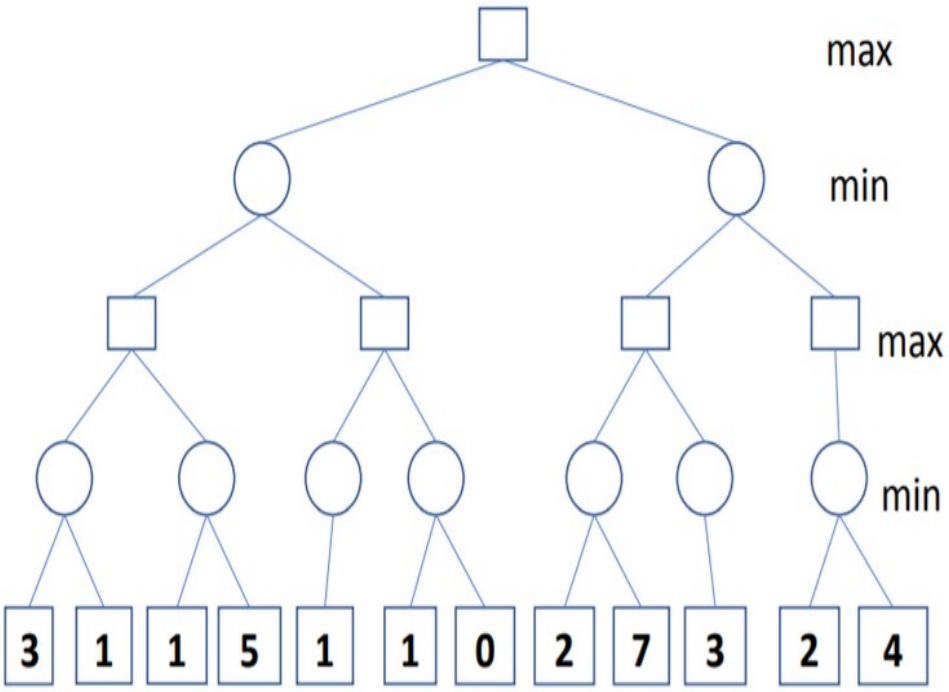
8-Queen Example:



Min-Max Algorithm

→ Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the other player is also playing optimally.
→ It is used in games such as tic-tac-toe, go, chess, isola, checkers, and many other two-player games.
→ Such games are called games of perfect information because it is possible to see all the possible moves of a particular game.
→ There can be two-player games which are not of perfect information such as Scrabble because the opponent's move cannot be predicted.

- **Game Tree:** It is a structure in the form of a tree consisting of all the possible moves which allow you to move from a state of the game to the next state.
- **Initial state:** It comprises the position of the board and showing whose move it is.
- **Successor function:** It defines what the legal moves a player can make are.
- **Terminal state:** It is the position of the board when the game gets over.
- **Utility function:** It is a function which assigns a numeric value for the outcome of a game. For instance, in chess or tic-tac-toe, the outcome is either a win, a loss, or a draw, and these can be represented by the values +1, -1, or 0, respectively.



Alpha-Beta Pruning

Purpose: To prune unnecessary branches from a fully grown tree.
Advantage over min-max: Pruning makes algorithm less time consuming.
In which case Alpha-Beta Pruning will have overhead?
Conditions:
1. $\alpha \geq \beta$ for pruning.
2. α, β values can not move upward.
3. α, β values can move downward.
4. node value can move upward.
5. node value can not move downward.

Knowledge Representation and Learning

Knowledge: Understanding of subject area.

AI Cycle:

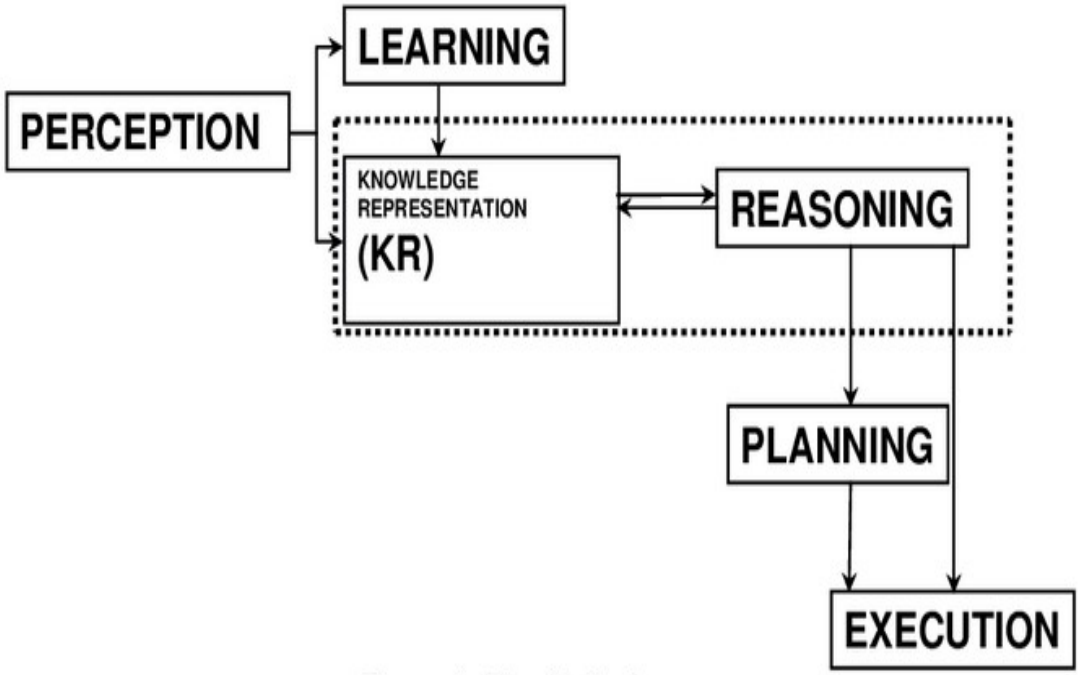


Figure 1: The AI Cycle

→ Descriptive Knowledge
→ Meta-Knowledge
→ Heuristic Knowledge
→ Structural Knowledge

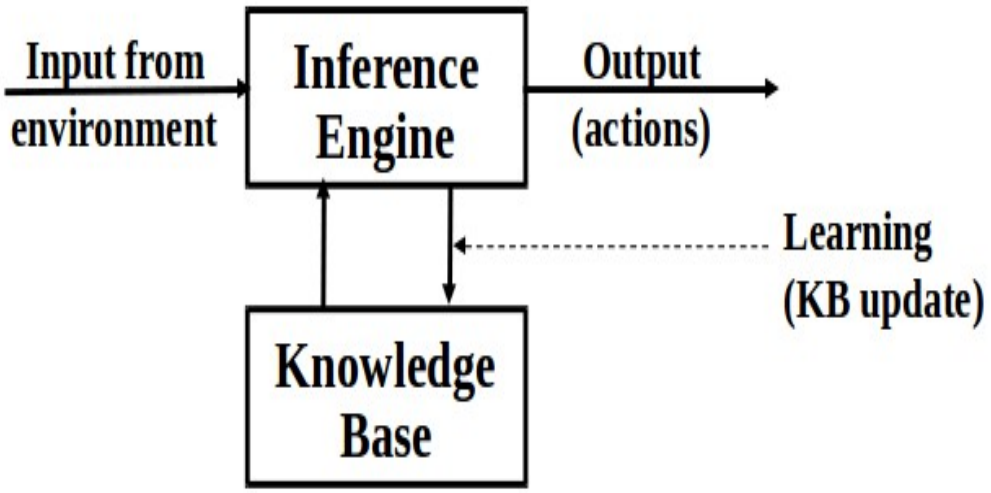
Representation of knowledge:

In General: Symbols

Specific to Computer Sciences: Data Structures for problem at hand.

Graph Representation?????

Knowledge Base Agent:



The **Inference engine derives new sentences from the input and KB**

The inference mechanism depends on representation in KB

The agent operates as follows:

1. receives percepts from environment

2.It computes what action it should perform (by IE and KB)

3. It performs the chosen action (some actions are simply inserting inferred new facts into KB).

→Facts

→Propositions

→Rules

→Logical Operators:

AND [^]

OR [v]

NOT [~, !, ¬, ¯]

Implication [→]

Bi-Conditional [↔]

P	**Q**	**P^Q**	**P v Q**	**¬P**	**P → Q**	**P ↔ Q**
T	T	T	T	F	T	T
T	F	F	T	F	F	F
F	T	F	T	T	T	F
F	F	F	F	T	T	T

Rules of Inference

Rule of Inference	*Tautology*	*Name*
p $p \rightarrow q$ $\therefore q$	$[p \wedge (p \rightarrow q)] \rightarrow q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$	Modus tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$[(p \vee q) \wedge \neg p] \rightarrow q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore p \wedge q$	$[(p) \wedge (q)] \rightarrow (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$	Resolution

Logical Equivalences:

$P \rightarrow Q \equiv \neg P \text{ v } Q$

$P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

Wumpus World Problem

First Order Predicate Logic / Predicate Logic

→ Proposition: “All tapirs are cuddly” would result in true or false.
Here “tapirs” is “Subject” and “are cuddly” is predicate.
→ Predicate logic/First Order Predicate Logic helps in quantifications such that “scope of predicate over subject”
i.e. ∀x(Tx ⇒ Cx)

Limits of Propositional Logic:

“If a person is rich then they have a nice car.”

We could encode this knowledge in propositional knowledge by creating a large set of rules, one rule for each person:

BobIsRich→ BobHasNiceCar
TonyIsRich→ TonyHasNiceCar
JonIsRich→ JonHasNiceCar
.
.
.

Syntax of FO Logic:

→ **Logical connectives(⇒,∧,∨, and⇐⇒)**, **negation (¬)**, and parentheses. These will be used torecursively build complex formulas, just as was done for propositional logic.
→ **Constants symbols** are strings that will be interpreted as representing objects, e.g.Bobmightbe a constant.
→ **Variable symbols** will be used as “place holders” for quantifying over objects.
→ **Predicate symbols** Bob is rich ==== > Rich(Bob)
→ **Function symbols** Jhon is the father of Ellen ====> fatherOf(Ellen)=Jhon
→ **Universal and existential quantifier symbols** will be used to quantify over objects.

Quantifiers:

→ **Universal quantification**
(∀x)P(x) means that P holds for all values of x in the domain associated with that variable
E.g., (∀x) dolphin(x) → mammal(x)
→ **Existential quantification**
(∃ x)P(x) means that P holds for some value of x in the domain associated with that variable
E.g., (∃ x) mammal(x) ∧ lays-eggs(x)
Permits one to make a statement about some object without naming it.

Formulas:

→ Formulas are used in FOL to state properties and will be interpreted as true or false.
→ **primitive formulas**
TallerThan(Bob,FatherOf(Bob))
→ **a logical combination of formula**
TallerThan(Bob,FatherOf(Bob))∧TallerThan(FatherOf(FatherOf(Bob)),Bob)
→ **an expression of the form “quantifier variable formula”**
∃xTallerThan(FatherOf(x),x)

Examples:

Herschel is a cat and Orfy is a cat. Cat(herschel) ∧ Cat(orfy)

Heidi loves Gary.	Loves (Heidi, Gary)
Some cats are pets.	∃x (Pet(x) ∧Cat(x))
No dog is a cat.	∀x (Dog(x) →¬Cat(x))
Willy is larger than every cat.	∀x (Cat(x) →Larger(willy, x))
There is a cat that every dog chases.	∃y (Cat(y)∧∀x (Dog(x) → Chases(x, y)))

Rules of Inference for Quantifiers

Rule of Inference	Name
$\forall x P(x)$ $\therefore P(c)$	Universal instantiation
P(c) for an arbitrary c $\therefore \forall x P(x)$	Universal generalization
$\exists x P(x)$ $\therefore P(c)\ for\ some\ c$	Existential instantiation
P(c) for some c $\therefore \exists x P(x)$	Existential generalization

Example:

"All human beings are mortal."
"Sachin is a human being."
conclusion: Does it follow that "Sachin is mortal?"
Translation:

$$\frac{\forall x\,(H(x) \rightarrow M(x)) \qquad H(Sachin)}{\therefore M(Sachin)}$$

Solution:

Step	Valid Argument	Reason
(1)	∀x (H(x) → M(x))	Premise
(2)	H(Sachin) → M(Sachin)	Universal instantiation from (1)
(3)	H(Sachin)	Premise
(4)	M(Sachin)	Modus ponens from (2) and (3)

Example:
"All lions are fierce."
"Some lions do not drink coffee."
Conclusion: Does it follow that: "Some fierce creatures do not drink coffee."
Translation:

1. $\forall x\,(L(x) \rightarrow F(x))$
2. $\exists x\,(L(x) \wedge \neg C(x))$
3. $\exists x\,(F(x) \wedge \neg C(x))$

Solution:

1.	$\exists x\,(L(x) \wedge \neg C(x))$	Premise
2.	$L(Foo) \wedge \neg C(Foo)$	Existential Instantiation from (1)
3.	$L(Foo)$	Simplification from (2)
4.	$\neg C(Foo)$	Simplification from (2)
5.	$\forall x\,(L(x) \rightarrow F(x))$	Premise
6.	$L(Foo) \rightarrow F(Foo)$	Universal instantiation from (5)
7.	$F(Foo)$	Modus ponens from (3) and (6)
8.	$F(Foo) \wedge \neg C(Foo)$	Conjunction from (4) and (7)
9.	$\exists x\,(F(x) \wedge \neg C(x))$	Existential generalization from (8)

Genetic Algorithm

→ **Evolution vs Revolution**
→ **Deterministic vs Stochastic**
→ **Darwin Theory**
→ **Concept of Survival of Fittest**
→ **Optimization**
→ **Optimization Problems (Traveling Sales Person Problem, Graph Coloring Problem, Function Optimization)**

Traditional Approaches:

1. Enumerative:

Within a finite search space or discretized infinite search space, look at objective function value at **every** point in space **one** at a time.

Analysis:

→ Attractive as mimicking human search.
→ Works when number of possibilities are small.
→ NOT efficient as practical spaces are too large.

2. Random:

→ Became popular because of shortcomings of calculus based and enumerative schemes. But random techniques that search and save the best must also be discontinued because of inefficiency.
→ In the long run do no better than enumerative.
→ Note that GA use random choice as a tool in directed search process.

→ A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of **stochastic operators**.
→ A biologically inspired model of intelligence and the principles of *biological evolution* are applied to find solutions to difficult problems
→ The problems are not solved by reasoning logically about them; rather populations of competing candidate solutions are spawned and then evolved to become better solutions through a process patterned after ***biological evolution***
→ Less worthy candidate solutions tend to die out, while those that show promise of solving a problem survive and reproduce by constructing new solutions out of their components.
→ GA begin with a population of candidate problem solutions
→ Candidate solutions are evaluated according to their ability to solve problem instances: only the fittest survive and combine with each other to produce the next generation of possible solutions.
→ Thus increasingly powerful solutions emerge in a Darwinian universe
→ Learning is viewed as a competition among a population of evolving candidate problem solutions
→ This method is heuristic in nature and it was introduced by John Holland in 1975

Terminologies (Representation of GA):

Gene: A basic unit, which represents one characteristic of the individual. The value of each gene is called an ***allele***

Chromosome: A string of genes; it represents an individual i.e. a possible solution of a problem. Each chromosome represents a point in the search space

Population: A collection of chromosomes

An appropriate chromosome representation is important for the efficiency and complexity of the GA.

Population Size:
→ Number of individuals present and competing in an iteration (generation).
→ If the population size is too large, the processing time is high and the GA tends to take longer to converge upon a solution (because less fit members have to be selected to make up the required population)
→ If the population size is too small, the GA is in danger of premature convergence upon a sub-optimal solution (all chromosomes will soon have identical traits). This is primarily because there may not be enough diversity in the population to allow the GA to escape local optima

GAs differ from more normal optimization and search procedures in 4 ways:
→ GAs work with a coding of the parameter set, not the parameters themselves.
→ GAs search from a population of points, not a single point.
→ GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
→ GAs use probabilistic transition rules, not deterministic rules.

Stochastic Operators:
→ Selection
→ Recombination / Crossover
→ Mutation

Selection:
→ They are used to select parents from the current population.
→ The selection is primarily based on the fitness. The better the fitness of a chromosome, the greater its chance of being selected to be a parent.
→ The rate at which a selection algorithm selects individuals with above average fitness is ***selective pressure***.
→ If there is not enough *selective pressure*, the population will fail to converge upon a solution. If there is too much, the population may not have enough diversity and converge prematurely

Types of Selection:
→ Random Selection
→ Roulette Wheel Selection
→ Tournament Selection
→ Rank Based Selection

Random Selection:
→ Individuals are selected randomly with no reference to fitness at all.
→ All the individuals, good or bad, have an equal chance of being selected.

Roulette Wheel Selection / Proportional Selection:
→ Chromosomes are selected based on their fitness relative to the fitness of all other chromosomes
→ For this all the fitness are added to form a sum S and each chromosome is assigned a relative fitness (which is its fitness divided by the total fitness S)
→ A process similar to spinning a roulette wheel is adopted to choose a parent; the better a chromosome's relative fitness, the higher its chances of selection
→ The selection of only the most fittest chromosomes may result in the loss of a correct gene value which may be present in a less fit member (and then the only chance of getting it back is by mutation)
→ One way to overcome this risk is to assign probability of selection to each chromosome based on its fitness
→ In this way even the less fit members have some chance of surviving into the next generation
Chromosomes are selected based on their fitness relative to the fitness of all other chromosomes

→ For this all the fitness are added to form a sum S and each chromosome is assigned a relative fitness (which is its fitness divided by the total fitness S)
→ A process similar to spinning a roulette wheel is adopted to choose a parent; the better a chromosome's relative fitness, the higher its chances of selection
→ The probability of selection of a chromosome "i" may be calculated as

$p_i = \text{fitness}_i / \sum_j fitness_j$

Example

Chromosome	**Fitness**	**Selection Probability/Relative Fitness**
1	7	7/14
2	4	4/14
3	2	2/14
4	1	1/14

Tournament Selection:
→ One parent is selected by comparing a subset *b* of the available chromosomes, and selecting the fittest; a second parent may be selected by repeating the process
→ Its advantage is that the worse individuals of the population will have very little probability of selection, whereas the best individuals will not dominate the selection process, thus ensuring diversity
Rank based Selection:
→ Rank based selection uses the rank ordering of the fitness values to determine the probability of selection and not the fitness values themselves.
→ This means that the selection probability is independent of the actual fitness value.
→ Ranking therefore has the advantage that a highly fit individual will not dominate in the selection process as a function of the magnitude of its fitness.
→ The population is sorted from best to worst according to the fitness
→ Each chromosome is then assigned a new fitness based on a *linear* ranking function

New Fitness = (P – r) + 1

→ where **P** = population size, **r** = fitness rank of the chromosome
If P = 11, then a chromosome of rank 1 will have a New
Fitness of 10 + 1 = 11 & a chromosome of rank 6 will have 6
→ Once the new fitness is assigned, parents are selected by the same roulette wheel procedure used in proportionate selection
Disadvantage:
- Population must be sorted
- Chromosomes with the same fitness will not have the same probability of being selected.

GA Metaphor to Nature:

Nature	**Genetic Algorithms**
Environment	Optimization problem
Individuals living in that environment	Feasible solutions
Individual's degree of adaptation to its surrounding environment	Solutions quality (fitness function)

Nature	Genetic Algorithms
A population of organisms (species)	A set of feasible solutions
Selection, recombination and mutation in nature's evolutionary process	Stochastic operators
Evolution of populations to suit their environment	Iteratively applying a set of stochastic operators on a set of feasible solutions

Crossover/Recombination:

→ This proceeds in two steps:

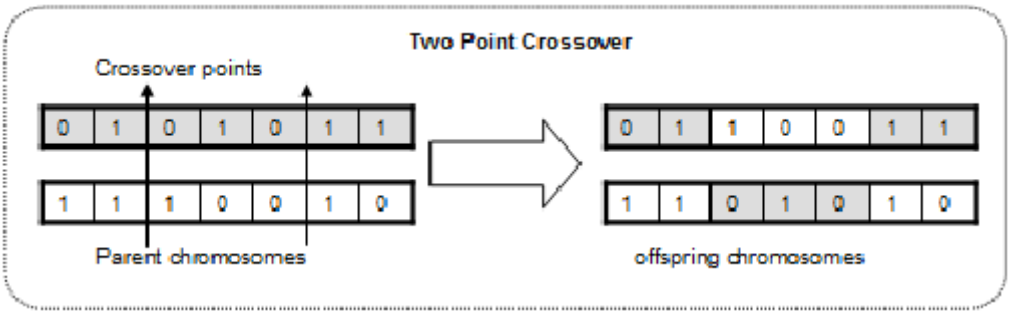
1. Members of the copied strings are combined at random.
2. Each pair of strings undergoes crossover (as follows):

→ An integer position is selected uniformly at random |1, l – 1| where l is string length.

→ Two new strings are created by swapping all characters between positions k + 1 and l inclusively

Chromosome1	11011|00100110110
Chromosome2	11011|11000011110
Offspring1	11011|11000011110
Offspring2	11011|00100110110

Single Point Crossover

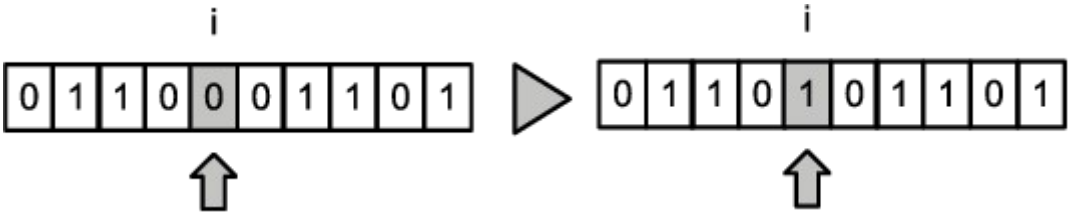


Mutation:

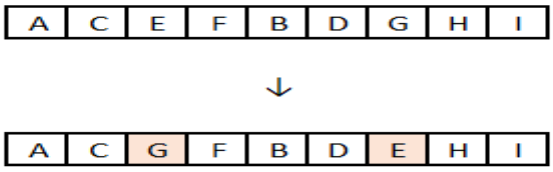
→ Although selection and crossover generate bulk of next generation, but these operations may miss-out potentially useful material i.e. 1's and 0's at particular locations.

→ In artificial genetic systems, mutation protects against such irrecoverable loss.

→ In GA mutation is the random (with small probability) alteration of the value of a string position. Mutation in GA is used sparingly i.e. one in a thousand bit (position) transfers.



Flip Mutation



Swap/Interchange Mutation

→ **Example on Function Optimization**