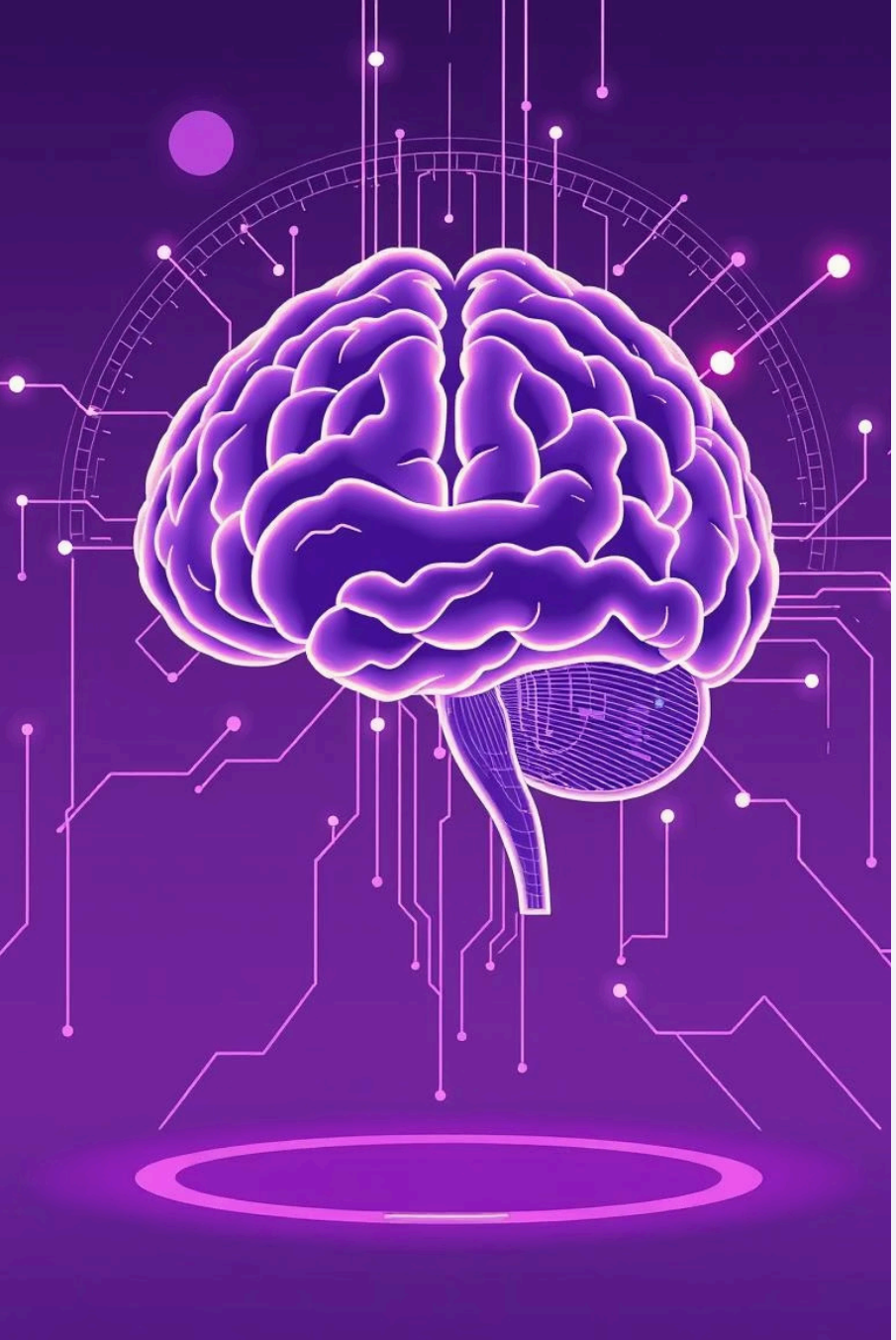


The Foundations of Artificial Intelligence

Welcome to an in-depth exploration of Artificial Intelligence, a field dedicated to creating systems that can reason, learn, and act intelligently. This presentation covers core concepts, from fundamental definitions and historical milestones to the algorithms that drive modern AI systems.

Defining AI and Its Core Goals

Artificial Intelligence (AI) is the science and engineering of making intelligent machines, especially intelligent computer programs. The ultimate goal is to enable machines to perform tasks that typically require human intelligence, such as learning, reasoning, perception, and problem-solving.

Historically, the goals of AI have been categorized into four main approaches:

Thinking Humanly

Modeling cognitive processes (Cognitive Science).

Acting Humanly

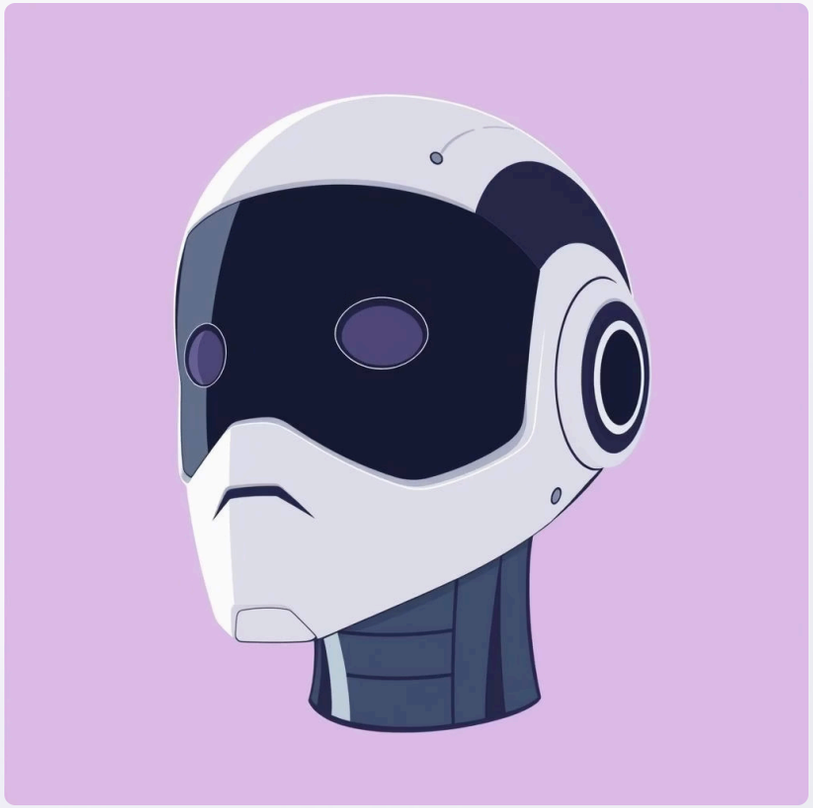
Passing the Turing Test successfully.

Thinking Rationally

Using logic and precise reasoning to reach conclusions.

Acting Rationally

Designing intelligent agents that achieve the best outcome.



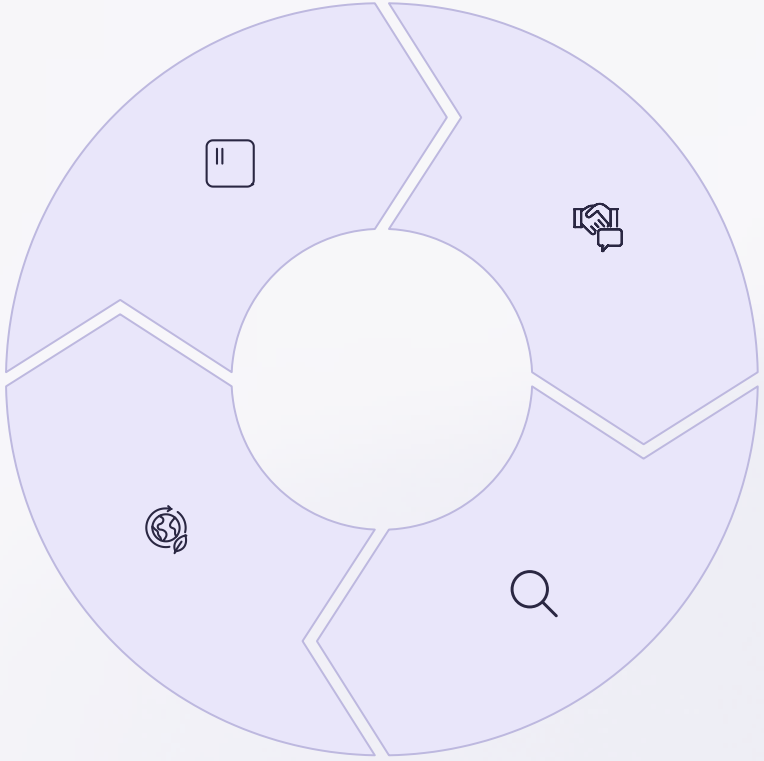
> The Turing Test, proposed by Alan Turing, defines intelligence as the ability of a machine to exhibit behavior equivalent to, or indistinguishable from, that of a human being.

Intelligent Agents: The Core of Rational Action

An AI agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. Rationality dictates that an agent should choose the action that maximizes its expected performance measure, given the percept sequence to date.

Sensors

Perceive environment (e.g., camera, microphone).



Processing

Decision making and reasoning.

Environment

The world the agent interacts with.

Actuators

Take actions (e.g., motors, display screen).

Different types of agents are designed based on how much knowledge they have of the world and their goals.

Architectures of AI Agents

Agents vary in complexity depending on how they map percepts to actions. The four primary agent types build upon each other in terms of sophistication:

Simple Reflex Agent

Acts based only on the current percept, ignoring the history. Uses **Condition-Action rules**.

Model-Based Reflex Agent

Maintains an internal state (a model of the world) to track unobservable aspects of the environment.

Goal-Based Agent

Uses goal information alongside the world model to choose actions that lead to a desired state.

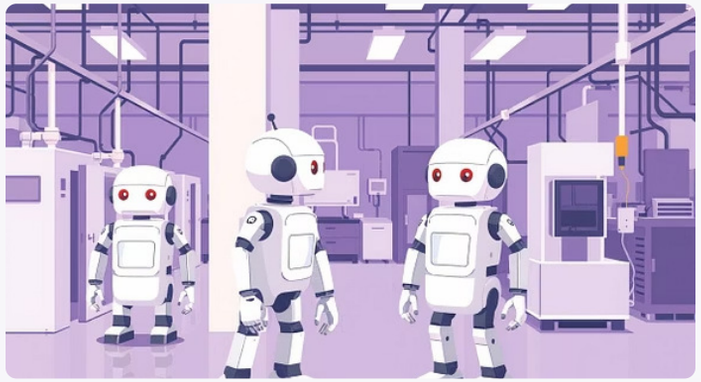
Utility-Based Agent

Decides based on maximizing a utility function, which provides a measure of how good a state is and balances competing goals.

A fifth crucial type, the **Learning Agent**, includes a **Critic** to evaluate performance and a **Learning Element** to modify the agent's knowledge over time, ensuring continuous improvement.

Understanding the Agent's Environment

The design and complexity of a rational agent heavily depend on the nature of its environment. Environments can be classified along several critical dimensions:



Fully vs. Partially Observable

Can the agent sense the entire state of the environment?

Deterministic vs. Stochastic

Is the next state entirely determined by the current state and the agent's action?

Episodic vs. Sequential

Do actions in one episode affect future episodes?

Static vs. Dynamic

Can the environment change while the agent is deliberating?

Discrete vs. Continuous

Are the states and actions limited to a finite set or are they continuous?

Known vs. Unknown

Does the agent know the rules and outcome probabilities of the environment?

The Problem-Solving Paradigm: State Space Search

Many AI problems are solved by searching through a state space—a collection of possible states the problem can be in. A well-defined problem has four key components:

Initial State

The starting configuration of the problem.

Successor Function

A description of the possible actions available in any state.

Goal Test

The condition that determines if a given state is the solution.

Path Cost

A function that assigns a cost to a sequence of actions.

Uninformed Search Algorithms explore the state space without using domain-specific knowledge, including Breadth-First Search (BFS), Depth-First Search (DFS), Depth-Limited Search, and Iterative Deepening Search.

Informed Search and Heuristics

Informed search strategies use problem-specific knowledge, known as a **Heuristic**, to estimate the cost from the current state to the goal state. This allows the search to be more efficient by prioritizing promising paths.

Best-First Search

Uses a heuristic function $h(n)$ to expand the node closest to the goal first.

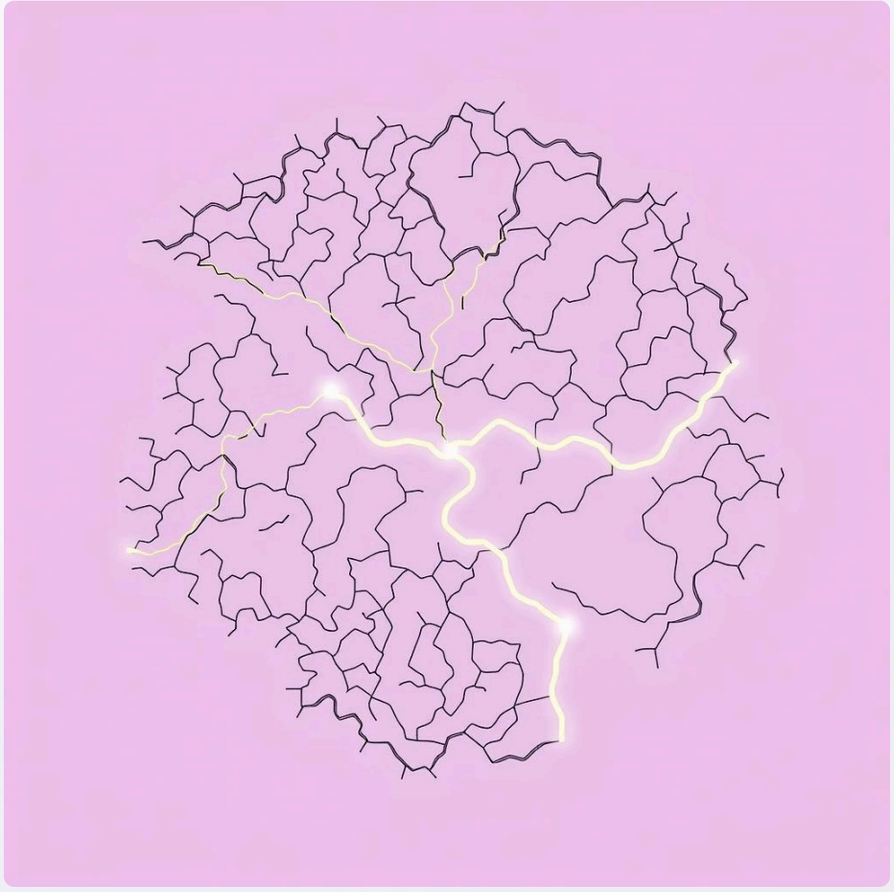
A* Search

Combines cost-so-far $g(n)$ and heuristic $h(n)$ to find the path with the minimum total estimated cost $f(n) = g(n) + h(n)$.

Hill Climbing

A local search that moves continuously in the direction of increasing value/performance (the 'uphill' direction).

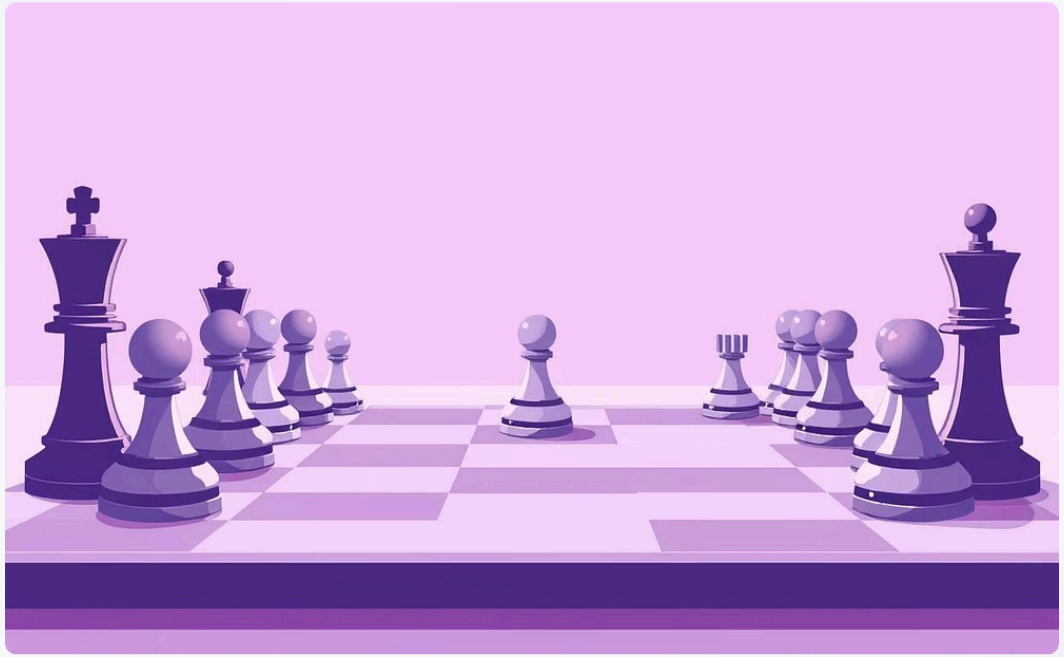
The power of the A* search lies in the concept of **Admissibility**: a heuristic is admissible if it never overestimates the cost to reach the goal. This guarantees that A* finds the optimal solution.



Example: Solving the 8-puzzle problem or the N-Queens problem often relies on well-designed heuristics to manage the vast state space.

Adversarial Search: Games and Decision Making

Adversarial search, or game theory, deals with environments where the agent's success depends on the actions of an opponent. Games like Chess, Tic-Tac-Toe, and Nim are classic examples solved using these techniques.



Minimax Algorithm

A recursive algorithm used to determine the optimal move for a player assuming the opponent also plays optimally (maximizing their win, minimizing ours).

Alpha-Beta Pruning

An optimization technique for the Minimax algorithm that eliminates branches in the search tree that are guaranteed not to lead to the optimal solution, greatly enhancing efficiency.

These algorithms are essential for developing strong AI players in two-player, zero-sum games. Demonstrating these concepts often involves creating GUI versions of games like Chess or Tic-Tac-Toe where the AI uses Minimax and Alpha-Beta Pruning to select moves.

Constraint Satisfaction Problems (CSP)

CSPs are a special class of problems where the solution must satisfy a set of hard constraints. They are defined by variables, domains, and constraints.

Solving CSPs involves search and powerful techniques for reducing the search space, known as **Constraint Propagation** (e.g., Arc Consistency, Path Consistency). Backtracking search is often employed, enhanced by heuristics like **Minimum Remaining Value (MRV)** and the **Degree Heuristic** to determine the best variable and value assignments.

- Common CSP examples include Map Coloring, Cryptarithmetic puzzles, and scheduling problems.
- Global Consistency, like **K-consistency**, ensures consistency among larger subsets of variables.

Conclusion & Next Steps in AI

Key Takeaways from the Foundations of AI

AI seeks to build **rational agents** that act optimally in complex environments.

Problem-solving is rooted in **State Space Search**, transitioning from uninformed to informed methods using **heuristics**.

Adversarial environments require advanced techniques like **Minimax** and **Alpha-Beta Pruning**.

Constraint Satisfaction Problems require specialized methods like **Constraint Propagation** for efficient solutions.

The concepts covered here form the bedrock for advanced AI disciplines like Machine Learning and Robotics. For educational demos, consider using these foundational algorithms in practical, visual game environments.