CCES – College of Computer Engineering and Sciences

Prince Sattam Bin Abdulaziz University

CS600: Advanced AI

Introduction to Artificial Intelligence

Chapter 1 — Agents, Reasoning, and Learning

Dr. Khalil Chebil

Artificial Intelligence Course

Chapter Outline | What we will cover

Learning Objectives

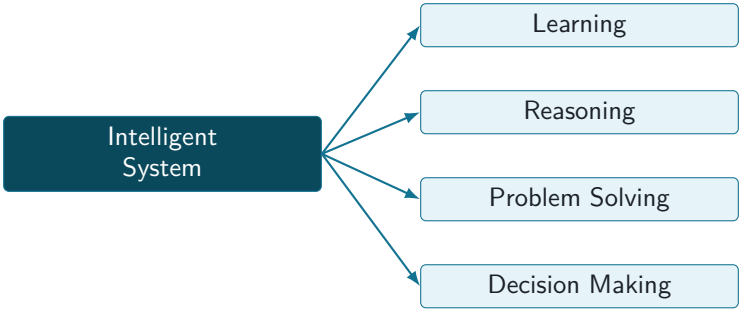
- Distinguish **deductive reasoning** from **inductive learning**
- Understand the concept of **agents** and **environments**
- Explore **supervised**, **unsupervised**, and **reinforcement** learning
- Reflect on the **Turing Test** and the goal of AGI

Section 1

What is Artificial Intelligence?

Definition

A branch of computer science concerned with the **automation of intelligent behaviors**.

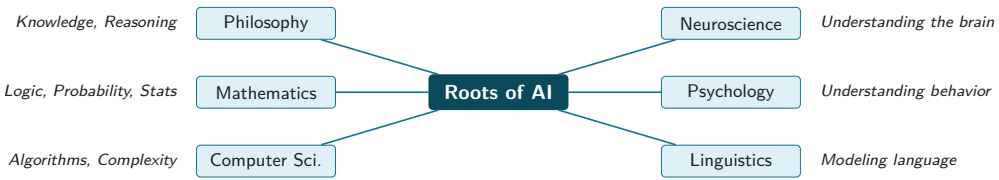


Examples: speech recognition, visual perception, language translation.

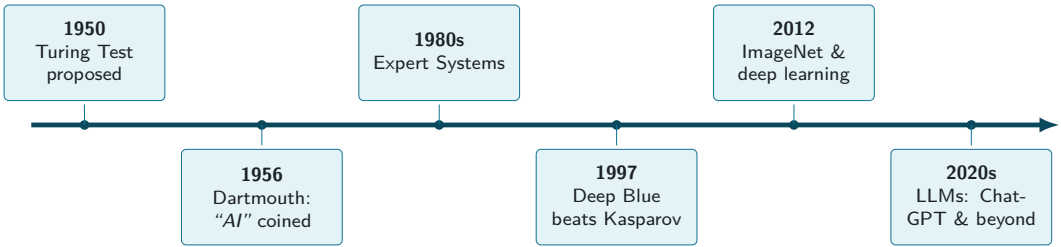
The Evolution of AI | Six defining phases

1. **Early Excitement** (1950s–60s)
   optimism about reasoning machines
2. **First AI Winter** (1970s)
   limited computational power
3. **Expert Systems Era** (1980s)
   knowledge-based systems
4. **Second AI Winter** (late 80s–90s)
   LISP market collapse, funding cuts
5. **ML Renaissance** (2000s–10s)
   statistical methods
6. **Deep Learning Revolution** (2010s–)
   neural nets + massive data

Roots of Artificial Intelligence | An interdisciplinary field



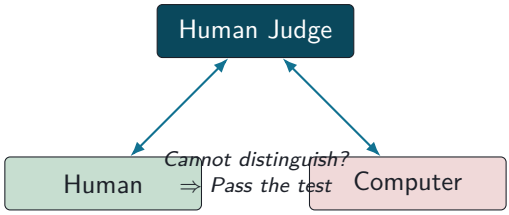
Timeline of AI History | Milestones



The Turing Test | Can a machine think? (Turing, 1950)

Procedure

1. All communication is text-based
2. A judge questions two hidden respondents
3. One is human, one is a computer
4. The machine *passes* if the judge cannot reliably tell them apart



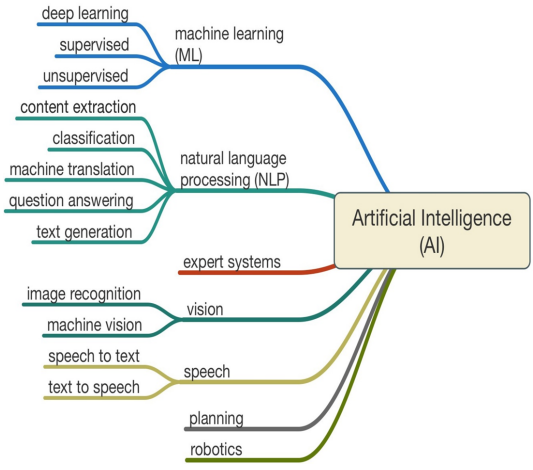
Criticisms

Tests *deception* rather than intelligence; linguistic-only; needs no understanding or consciousness; anthropocentric.

Status

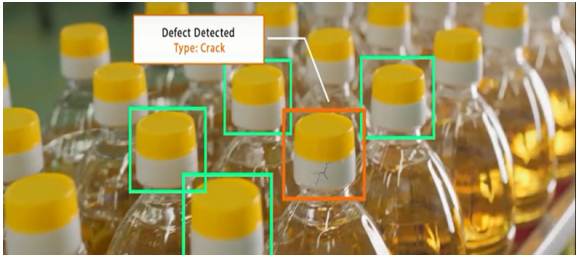
No system has convincingly passed the *full* test. AGI remains aspirational, not a present reality.

Subfields of AI | The AI landscape



AI in Industry | Where value is created

- **Anomaly detection** in processes and equipment
- **Optimize processes** to improve yield
- **Smarter decisions** that minimize risk
- **Predict future scenarios** with neural networks



Is Artificial Intelligence Dangerous? | Responsible AI

AI can be dangerous if **misused** or **poorly designed**. Key risks include:

- Job displacement
- Privacy concerns
- Bias and discrimination
- Autonomous weapons

!

Safety
Fairness
Accountability
Governance

Takeaway

Ethical development, transparency, and regulation are essential.

Section 2

How to Achieve AI

Two Roads to Artificial Intelligence | Phenomenal vs. Biological

1. Phenomenal Approach

- **Knowledge Representation**
- **Expert Systems & Planning**
- **Search**
  - Uninformed: BFS, DFS
  - Informed: A*, Greedy

Deductive reasoning

2. Biological Approach

- **Neural Networks** (ANN)
- **Evolutionary Algorithms**
- **Learning**: supervised, unsupervised, reinforcement
- **Agents**: NLP, vision, robotics

Inductive learning

Deductive vs. Inductive Reasoning | The two schools of thought

Deductive Reasoning

Start from a **knowledge base** of facts; apply logic to make inferences.

$$\text{Rules} + \text{Facts} \Rightarrow \text{Conclusions}$$

Search and logic-based methods.

Inductive Reasoning

Start from **observations**; generalize to broader rules or patterns.

$$\text{Examples} \Rightarrow \text{General Rule}$$

The foundation of machine learning.

The greatest strength of deductive reasoning — interpretability — is also its greatest limitation.

- **Constraint Satisfaction (CSP)**
  variables, domains, constraints
- **Search Algorithms**
  BFS, DFS, UCS, A*, local search
- **Logic-Based Systems**
  propositional & first-order logic, resolution

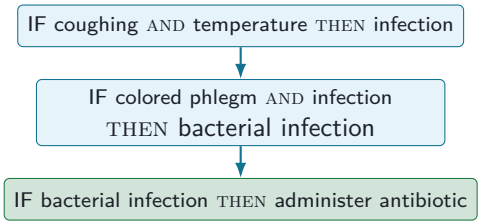
Expert Systems

- **Knowledge base**: facts + rules
- **Inference engine**: applies rules
- **User interface**

Forward chaining (data-driven) ▪ Backward chaining (goal-driven)

Facts about John

- Running a temperature
- Coughing
- Colored phlegm



In practice, thousands of rules and cases are needed for a system to work well.

Section 3

Constraint Satisfaction Problems

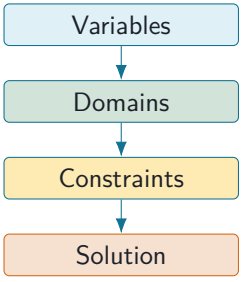
A **CSP** is defined by three ingredients:

- **Variables** $X_1, \ldots, X_n$
- **Domains** — allowed values for each variable
- **Constraints** — legal combinations of values

A *solution* assigns a value to every variable so that **all constraints hold**.

Everyday CSPs

Sudoku ▪ N-Queens ▪ Traveling Salesperson ▪ map / graph coloring ▪ scheduling.



Example 1 — Sudoku as a CSP | Fill the 9×9 grid

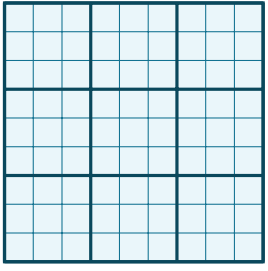
- **Variables**: $X_{i,j}$ for each cell (i, j)
- **Domains**: $\{1, 2, \ldots, 9\}$
- **Constraints** — all different within each unit:

Row / column / sub-grid

Rows: $\forall i,\ j \neq k:\ X_{i,j} \neq X_{i,k}$

Columns: $\forall j,\ i \neq k:\ X_{i,j} \neq X_{k,j}$

Sub-grids: within each 3×3 block,
$X_{3m+i,\,3n+j} \neq X_{3m+k,\,3n+l}$ for $(i, j) \neq (k, l)$



81 variables, 27 *all-different* constraints.

Sudoku — Python Implementation | Using the python-constraint library

```
from constraint import Problem, AllDifferentConstraint

def solve_sudoku(puzzle):              # puzzle: 9x9 list, 0 = blank
    p = Problem()
    for r in range(9):
        for c in range(9):
            v = puzzle[r][c]
            p.addVariable((r, c), range(1, 10) if v == 0 else [v])
    for i in range(9):                 # rows and columns all-different
        p.addConstraint(AllDifferentConstraint(), [(i, c) for c in range(9)])
        p.addConstraint(AllDifferentConstraint(), [(r, i) for r in range(9)])
    for br in range(3):                # 3x3 sub-grids all-different
        for bc in range(3):
            cells = [(br*3 + i, bc*3 + j) for i in range(3) for j in range(3)
                ]
            p.addConstraint(AllDifferentConstraint(), cells)
    sols = p.getSolutions()
    return sols[0] if sols else None
```

Example 2 — Eight Queens as a CSP | No two queens attack each other

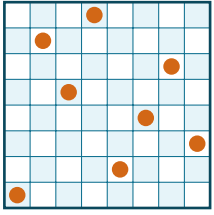
- **Variables**: $Q_1, \ldots, Q_8$ — column of the queen in row i
- **Domains**: $\{1, \ldots, 8\}$

Constraints

Columns differ: $\forall i \neq j : \; Q_i \neq Q_j$

Diagonals differ: $|Q_i - Q_j| \neq |i - j|$

Rows are handled implicitly — one queen per row by construction.



One valid placement of 8 queens.

Example 3 — Traveling Salesperson as a CSP | Shortest tour of all cities

Cities $\{1, \ldots, n\}$ with distances $d(i, j)$; find a tour of cost at most C.

- **Variables**: $X_{i,j} \in \{0, 1\}$ — 1 if the tour goes directly $i \to j$
- **Domains**: $\{0, 1\}$

Constraints

Enter each city once: $\sum_{i \neq j} X_{i,j} = 1$

Leave each city once: $\sum_{j \neq i} X_{i,j} = 1$

Bounded cost: $\sum_{i} \sum_{j \neq i} d(i, j)\, X_{i,j} \leq C$

A tour with n cities has $(n-1)!$ possibilities — **NP-hard** as n grows.

Example 4 — Graph / Map Coloring as a CSP | Adjacent regions differ

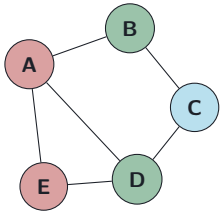
Graph $G = (V, E)$ with colors $\{1, \ldots, k\}$.

- **Variables**: $X_{i,c} \in \{0, 1\}$ — 1 if vertex i has color c
- **Domains**: $\{0, 1\}$

Constraints

One color per node: $\sum_{c=1}^{k} X_{i,c} = 1, \ \forall i$

Adjacent differ: $X_{i,c} + X_{j,c} \leq 1, \ \forall (i, j) \in E, \ \forall c$



No edge joins two same-colored nodes.

Section 4

Intelligent Agents

The Agent Abstraction | Perceive — Decide — Act

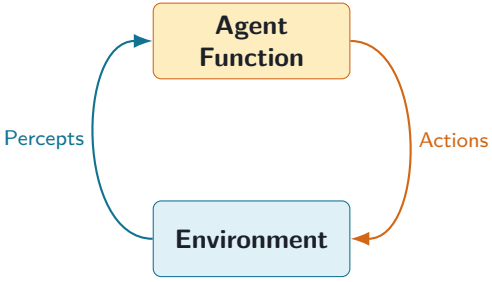
An **agent**:

1. **Perceives** the environment via *sensors*
2. **Acts** on it via *actuators*
3. Makes **decisions** to achieve goals

Formal definition

An agent function maps percept histories to actions:

$$f : P^* \to A$$

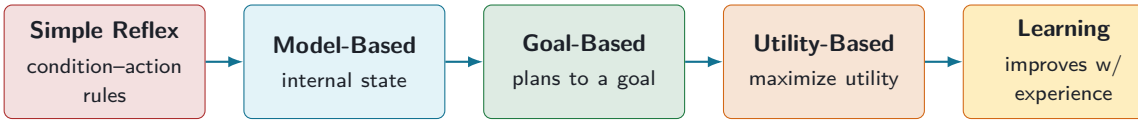


PEAS Description | Characterizing an agent

Agent	Performance	Environment	Actuators	Sensors
Cleaning robot	Cleanliness, battery	Room, dirt	Wheels, vacuum	Camera, dirt sensor
Chess program	Win/loss/draw	Board, opponent	Move selection	Board position
Self-driving car	Safety, speed	Roads, traffic	Steering, throttle	Cameras, LIDAR, GPS
Medical diagnosis	Patient health	Symptoms	Treatment plan	Lab tests, imaging
Chatbot	User satisfaction	Conversation	Text response	User input text

Performance ▪ **E**nvironment ▪ **A**ctuators ▪ **S**ensors

Types of Agents | From reflexes to learning

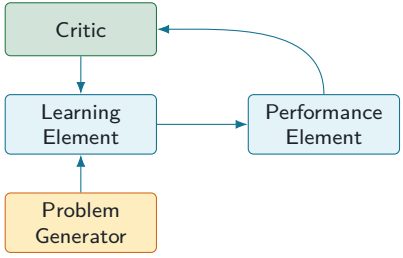


Thermostat → simple reflex **Vacuum w/ memory** → model-based
GPS navigation → goal-based **Trading bot** → utility-based **AlphaGo** → learning

▶ **Open the Agent Lab in Google Colab**

The Learning Agent | Four interacting components

- **Performance Element** — selects actions
- **Learning Element** — improves the performance element
- **Critic** — provides feedback on performance
- **Problem Generator** — suggests exploratory actions



Example

AlphaGo learned to master Go through self-play.

Types of Environments | Design depends on the world

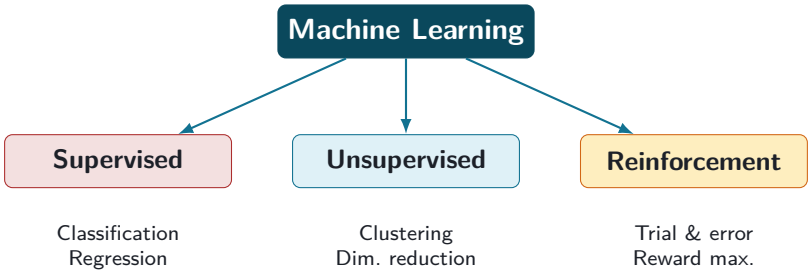
Dimension	Type 1	Type 2	*Example*
Observability	Fully observable	Partially observable	*Chess vs. Poker*
Determinism	Deterministic	Stochastic	*Chess vs. Backgammon*
Episodes	Episodic	Sequential	*Image class. vs. Chess*
Dynamics	Static	Dynamic	*Crossword vs. Driving*
State space	Discrete	Continuous	*Chess vs. Robot control*
Agents	Single-agent	Multi-agent	*Puzzle vs. Poker*

Simplest: fully observable, deterministic, episodic, static, discrete, single-agent. Hardest: real-world robotics (the opposite of every dimension).

Section 5

Learning Paradigms

Three Types of Learning | Inductive learning from data



Learn a function from **labeled** examples.

- Training data: $(X_1, y_1), \ldots, (X_n, y_n)$
- Learn f such that $y_i \approx f(X_i)$
- Minimize loss: $L = \sum_{i=1}^{n} (y_i - f(X_i))^2$

Two tasks

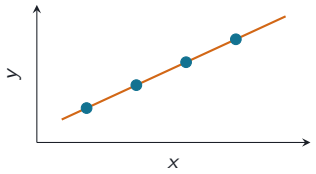
- **Classification**: categorical (spam/not spam)
- **Regression**: numerical (house prices)

Example

Data $(1, 3), (2, 5), (3, 7), (4, 9)$

$$y = 2x + 1$$

Predict $x = 5 \Rightarrow y = 11$.



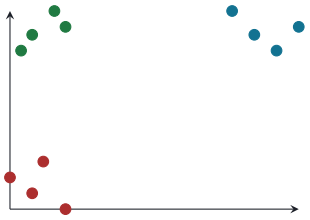
Unsupervised Learning | Discovering hidden structure

Find structure in **unlabeled** data.

- **Clustering**: K-means, hierarchical, DBSCAN
- **Dimensionality reduction**: PCA, t-SNE
- **Anomaly detection**: fraud, intrusion

K-Means (sketch)

1. Initialize k random centroids
2. Repeat until convergence:
   a. Assign each point to nearest centroid
   b. Update centroids to cluster means



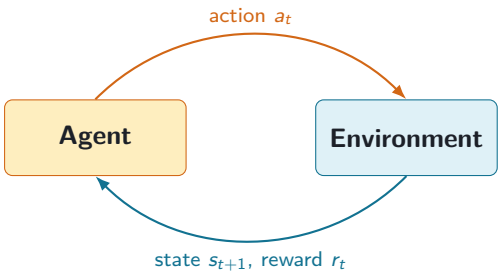
Three clusters discovered

An **agent** learns which actions to take by observing **rewards**.

- State s, action a, reward r, policy $\pi(s)$
- Goal: maximize cumulative reward

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \quad \gamma \in [0, 1]$$

Algorithms: Q-Learning, DQN (value-based); Policy Gradient, Actor–Critic (policy-based); AlphaZero (model-based).



Applications

AlphaGo, robotics, autonomous vehicles, data-center cooling.

Comparing Learning Types | At a glance

Aspect	Supervised	Unsupervised	Reinforcement
Data	Labeled examples	Unlabeled data	Interaction w/ environment
Feedback	Correct answers	None (implicit)	Rewards / penalties
Goal	Predict outputs	Discover patterns	Maximize reward
Training	(X, y) pairs	X only	(s, a, r, s') tuples
Example	Spam detection	Segmentation	Game playing

Section 6

Summary

Chapter Summary | Key takeaways

Reasoning

- **Deductive**: rules + facts $\Rightarrow$ conclusions
- **Inductive**: examples $\Rightarrow$ general rules

Agents

Perceive–decide–act; PEAS; five agent types; environment dimensions.

Learning

- Supervised — labeled data
- Unsupervised — hidden structure
- Reinforcement — reward-driven

Looking ahead

Ch. 2: Search ▪ Ch. 3: Multiagent Search ▪ Ch. 4–5: Logic.

Thank you!

Questions & Discussion

Next: Chapter 2 — Searching State Spaces