**Engineering and Applied Science Programs for Professionals**
**Whiting School of Engineering**
**Johns Hopkins University**
**685.621 Algorithms for Data Science**

# Artificial Intelligence

The lecture content in this document complements the lecture notes from Russell, S. and Norvig, P., http://aima.cs.berkeley.edu/.

This document provides a roll-up of Artificial Intelligence (AI) Data Analytics and the methods associated in the theory of AI. At the start of this document, AI Fundamentals are the definition of what is AI, PEAS (Performance measure, Environment, Actuators, Sensors), a brief history, and ends with the state-of-the-art technologies in AI. This then leads to a funny story that is used in various examples on how to rephrase a questions/dialog for better communication, the story is then enhanced with the use of the LLM ChatGPT to provide an example of the use of AI technology. The third area of the document covers the concept of agents and environments, an example, different types of environments, and various types of agents. This introduction will provide the necessary knowledge to start learning about AI.

# Contents

**1 AI Fundamentals** **4**

1.1 **What is AI** . . . 5

1.1.1 **Acting humanly: The Turing test** . . . 6

1.1.2 **Thinking humanly: Cognitive Science** . . . 7

1.1.3 **Thinking rationally: Laws of Thought** . . . 7

1.1.4 **Acting rationally** . . . 8

1.2 **PEAS (Performance measure, Environment, Actuators, Sensors)** . . . 10

1.2.1 Example: Chatbot . . . 10

1.2.2 Example: Self-driving car . . . 11

1.3 **Potted History of AI** . . . 13

1.4 **State-of-the-Art** . . . 15

**2 Unintentional Funny Story** **16**

2.1 **A twist to the funny story of Joe Bear using AI** . . . 17

**3 Agents and Environments** **19**

3.1 **Vacuum-cleaner World Example** . . . 21

3.1.1 **A vacuum-cleaner agent** . . . 21

3.2 **Environment Types** . . . 24

3.3 **Agent Types** . . . 26

3.3.1 **Simple reflex agents** . . . 26

3.3.2 **Reflex agents with state** . . . 28

3.3.3 **Goal-based and Utility-based agents** . . . 29

3.3.4 **Goal-based agents** . . . 29

3.3.5 **Utility-based agents** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

3.3.6 **Learning agents** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

# 1 AI Fundamentals

At the start of this lesson, a broad overview of the fundamentals of Artificial Intelligence (AI) and related methods in the field of AI will be discussed. This will include the definition of AI and the components of intelligent systems acting and thinking humanly as well as thinking and acting rational. This is followed by PEAS which is an acronym for Performance measures, environments, actuators and sensors with a couple of examples. Next, the plotted history of AI is discussed and ends with some technologies that are current state-of-the-art models. This introduces some basic components of AI fundamentals which will provide the necessary knowledge in AI to assist in formulating solutions for problems in AI Data Analytics. [4, 1].

## 1.1 What is AI

Artificial Intelligence (AI) refers to the development of computer systems that can perform tasks typically requiring human intelligence. These tasks include problem-solving, speech recognition, planning, learning, natural language processing, perception, and the ability to move and manipulate objects. Artificial Intelligence can be divided into two main categories: Narrow AI, also known as Weak AI, which refers to AI systems that are programmed and trained to carry out a specific task. Examples of this type of AI include voice recognition systems and chess-playing computer programs. The other type is General AI, also known as Strong AI, which is a theoretical concept that has not yet been achieved. This type of AI would possess intelligence similar to that of a human, being able to understand, learn, and apply knowledge in various areas.

The construction of AI systems requires a theoretical understanding of how to create intelligent entities that can effectively act in various situations. This involves four key components: (1) architecture, (2) development, (3) analysis, and (4) implementation. Developing AI systems necessitates a theoretical understanding of how to create intelligent entities to think and act like humans, as well as to think and act rationally, a combination of multiple disciplines is necessary, making AI a complex and far-reaching field. The field of AI involves a variety of sub-disciplines, including game theory and machine learning (see Figure 1).

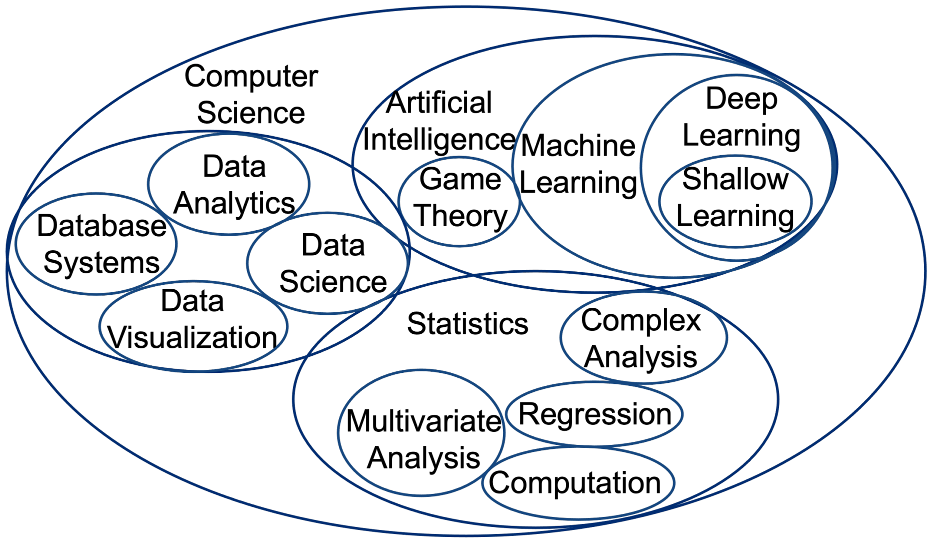

Figure 1: Representation of how AI fits into the field of Data Science and how other components such as Game Theory, Machine Learning, Deep Learning fit into the field of Artificial Intelligence.

Artificial Intelligence (AI) technologies make use of a variety of techniques, such as machine learning (a subset of AI where systems learn from data), deep learning (a more complex form of machine learning involving neural networks), and other computational methods to enable systems to carry out tasks similar to those of humans. AI has been the focus of extensive research and development, and its applications are far-reaching, from healthcare and finance to transportation and entertainment. It has become an integral part of modern technology and continues to have a major impact on many aspects of our lives.

### 1.1.1 Acting humanly: The Turing test

The Turing test is a method of inquiry in artificial intelligence (AI) to determine whether or not a computer is capable of human-like intelligence. Proposed by Alan Turing in 1950, this test is one of the most influential ideas in the study of artificial intelligence [3].

Is it possible for machines to act intelligently? To answer this question, an operational test must be performed to determine if they can demonstrate intelligent behavior as shown in Figure 2.

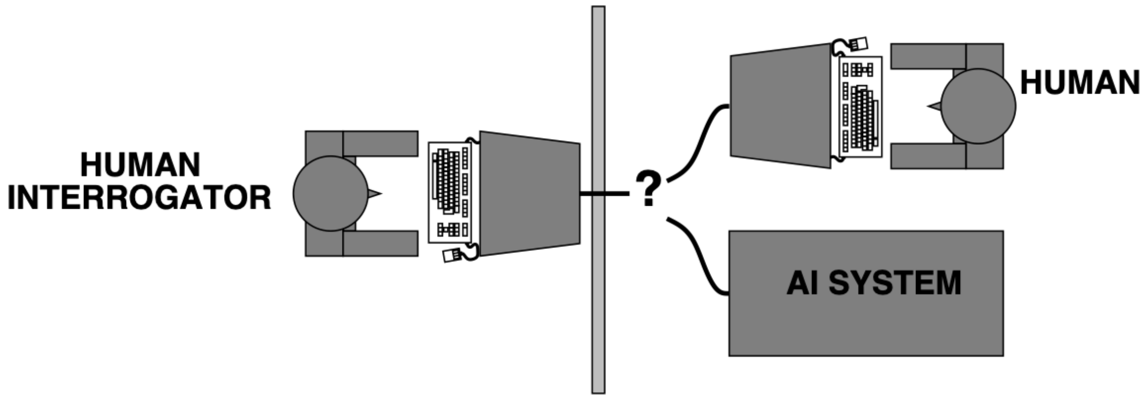

Figure 2: Human interrogator thinking a human is being interviewed while an AI system answers a series of questions.

The Turing Test is conducted as follows:

- Human Evaluator: A human judge engages in a natural language conversation with an unseen interlocutor, which might be either a human or a machine.
- The Test: Conversation is typically conducted through a computer interface so that the judge cannot hear or see their interlocutor. The judge's job is to determine which of the two they are conversing with: the human or the machine.
- Evaluation: If the judge cannot reliably tell the machine from the human, the machine is said to have passed the test.

It was predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes. The anticipation is that all major arguments against AI in the next 50 years will be contested in favor of the AI field. The suggested major components of AI are as follows:

- knowledge
- reasoning
- language understanding
- learning

The concept behind the Turing Test is that if a machine can interact in a manner that is indistinguishable from a human, it can be seen as a sign of intelligence. Despite this, the Turing test has been the subject of criticism and debate as it is not reproducible, constructive, or able to be mathematically analyzed. Some people believe that passing the Turing Test does not necessarily mean that the machine has consciousness, understanding, or even true intelligence, and may just be imitating human-like responses without any real comprehension of the conversation.

Other researchers have created different versions of the Turing test or completely different tests to measure machine intelligence more precisely. Despite the debates, the Turing Test has had a major effect on the field of AI, encouraging the advancement of technologies that strive to imitate human-like conversational skills.

### 1.1.2 Thinking humanly: Cognitive Science

This shift in thinking was driven by advances in computer technology, which allowed researchers to model and simulate cognitive processes. It also led to the development of new theories and models of cognition, such as artificial intelligence, neural networks, and cognitive architectures. The Cognitive Revolution has had a profound impact on the field of psychology and has led to the emergence of new research areas such as cognitive neuroscience and cognitive robotics. In the 1950s and 1960s, the cognitive revolution occurred, resulting in the emergence of cognitive science. This shift in thinking moved away from behaviorism, which only focused on observable behaviors, and towards understanding the internal psychological processes that control human thinking, learning, and behavior. This change was driven by advances in computer technology, which allowed researchers to model and simulate cognitive processes. This led to the development of new theories and models of cognition, such as artificial intelligence, neural networks, and cognitive architectures. The Cognitive Revolution has had a major influence on the field of psychology and has given rise to new research areas such as cognitive neuroscience and cognitive robotics.

The Cognitive Revolution placed a greater emphasis on mental processes rather than behavior, which had a significant effect on Artificial Intelligence (AI) research. This shift in focus moved away from simply creating machines that could act intelligently and instead focused on understanding and replicating the human thought process, leading to a more comprehensive and detailed approach to AI.

Cognitive Science has had a major impact on Artificial Intelligence (AI) by offering insights into human cognition that can be used to create more sophisticated and human-like artificial systems. This interdisciplinary approach has been beneficial both for our understanding of the human mind and the advancement of AI technologies that attempt to mimic human cognitive processes.

### 1.1.3 Thinking rationally: Laws of Thought

Rational thinking is the practice of using logical principles to make decisions. It involves clear, objective, and deliberate decision-making based on logical principles. This concept has its origins in philosophy and mathematics and is closely related to the "Laws of Thought," which are fundamental axiomatic rules that form the basis of rational thinking. . These laws are fundamental principles of logic that are used to determine the validity of an argument. They are also used to determine the truth of a statement. The three laws of thought are the law of identity, the law of noncontradiction, and the law of the excluded middle. The "Laws of Thought" are traditionally attributed to the ancient Greek philosopher Aristotle. These laws are essential principles of logic that are used to assess the soundness of an argument. They are also used to determine the accuracy of a statement. The three laws of thought are the law of identity, the law of noncontradiction, and the law of the excluded middle and include:

- The Law of Identity states that an object is identical to itself; if A is A, then A is equal to A.
- The Law of Non-Contradiction states that it is impossible for a statement to be both true and false simultaneously. In other words, no assertion can be both true and false at the same time.
- The Law of Excluded Middle states that a statement must be either true or false, with no other option in between. A proposition is either accurate, or its opposite is accurate.

Aristotle sought to identify what constitutes valid arguments and thought processes. He sought to determine the criteria for determining the correctness of an argument or thought process.

Several Greek educational institutions created different types of logic, *notation*, and *rules of derivation* for reasoning. It is not clear whether these led to the concept of mechanization.

The Law of Identity is a fundamental concept in Artificial Intelligence (AI) and computer science. It is especially important for operations that involve comparisons, equality checks, and logical reasoning. In symbolic AI, which uses rules and symbols to represent knowledge, understanding identity and equivalence is essential for successful problem solving and reasoning.

Despite its seeming straightforwardness, the Law of Identity has far-reaching implications for how we comprehend objects, symbols, and associations in mathematics and logic, and it remains a fundamental notion in a variety of disciplines, including Artificial Intelligence.

### 1.1.4 Acting rationally

Acting rationally refers to the concept of making decisions, performing actions, or behaving in a way that is consistent with reason or logic. Rational behavior is doing the right thing! In the context of AI, it involves creating systems that behave in a manner that is considered intelligent, based on available information, logical reasoning, and given goals. In AI, it is the right thing for a system to maximize an achievable goal, given the necessary information.

The application of Aristotle's "Nicomachean Ethics" to AI is an intersection between philosophy and technology that can lead to rich discussions and thoughtful considerations in the design and implementation of AI systems. It allows us to think about AI not just as a tool but as a part of our moral and social fabric.

Here is how the idea of acting rationally and the principle of "doing the right thing" are connected in

AI:

- Intelligent Agents: In AI, an intelligent agent is a system that perceives its environment, reasons about it, and takes actions to achieve specific goals. Acting rationally in this context means that the agent's actions are appropriate for its goals and the information it has, even if it does not possess complete or perfect knowledge.
- Decision Making: Rational behavior involves making decisions that are logically sound and that achieve specific goals. In AI, this can involve the use of algorithms to optimize choices based on certain criteria, such as maximizing rewards or minimizing costs.
- Rationality vs. Human-Likeness: It is worth noting that acting rationally doesn't necessarily mean acting humanly. An AI system can act rationally without mimicking human behavior or thought processes. Its actions are guided by logical principles and algorithms, rather than attempting to replicate human reasoning.
- Ethics and Morality: Acting rationally also has ethical implications, especially when AI systems are involved in decision-making that affects human lives. The definition of "the right thing" might vary based on ethical frameworks, societal values, or individual beliefs, and it is an area of ongoing debate and research in AI.

The concept of acting rationally has wide-ranging applications in AI, from the development of intelligent agents to the design of algorithms for optimization, planning, and decision making. It is at the heart of the endeavor to create machines that not only think but also act intelligently, balancing logical reasoning with practical constraints and ethical considerations.

## 1.2 PEAS (Performance measure, Environment, Actuators, Sensors)

The PEAS framework is a useful way to describe and analyze intelligent systems.

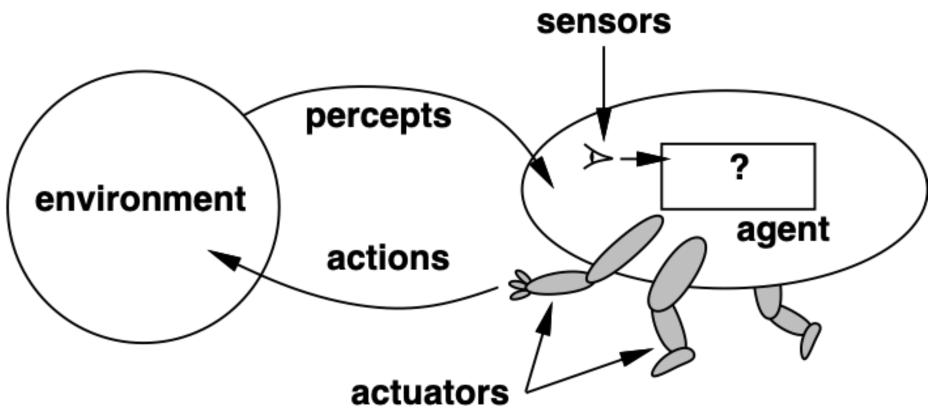

Figure 3: Agent's actions with an environment using sensors to stimulate the actuators.

### 1.2.1 Example: Chatbot

Let us break down the components of the PEAS framework for a chatbot designed to provide customer support on a website:

**Performance Measure**: The performance measure defines how the success of the chatbot system is evaluated. In this case, the performance measure could include metrics such as:

- Response time: How quickly the chatbot responds to user queries.
- Accuracy: How accurately the chatbot provides relevant and correct information.
- User satisfaction: Gathering feedback from users to determine their level of satisfaction with the chatbot's assistance.

**Environment**: The environment represents the external context in which the chatbot operates. In this case, the environment includes:

- User queries: Text-based questions or requests from users seeking information or assistance.
- Website content: The information the chatbot uses to respond to user queries.
- User interactions: The back-and-forth conversation between the user and the chatbot.

**Actuators**: Actuators are the components that perform actions based on the decisions made by the system. For a chatbot, the actuators could include:

- Text generation module: Generating responses to user queries in natural language.
- Display module: Presenting the generated responses to the user in the chat interface.

**Sensors**: Sensors provide the system with information about the environment and the system's current state. For a chatbot, the sensors could include:

- Text input module: Receiving user queries and input.
- User feedback module: Collecting feedback from users about the chatbot's responses.

Using the PEAS framework, we can summarize the example as follows:

- Performance Measure: Response time, accuracy, user satisfaction.
- Environment: User queries, website content, user interactions.
- Actuators: Text generation module, display module.
- Sensors: Text input module, user feedback module.

In this example, the chatbot's objective is to provide accurate and timely customer support on a website. It interacts with users by generating responses based on their queries and aims to achieve high accuracy and user satisfaction. The PEAS framework helps in analyzing the different aspects of the chatbot's operation and its interaction with the environment.

### 1.2.2 Example: Self-driving car

Let us break down the components of the PEAS framework for a self-driving car:

**Performance Measure**: The performance measure defines how the success of the self-driving car is evaluated. In this case, the performance measure could be the successful navigation of the car to its destination while ensuring safety, efficiency, and compliance with traffic rules. The time taken to reach the destination, adherence to traffic regulations, and the number of safety-critical incidents are all factors that contribute to the performance measure.

**Environment**: The environment represents the external context and surroundings in which the self-driving car operates. It includes the road network, traffic conditions, weather conditions, other vehicles, pedestrians, road signs, and traffic signals. The self-driving car needs to interact with and navigate through this environment to reach its destination.

**Actuators**: Actuators are the components responsible for carrying out actions or physical operations based on the decisions made by the system. In the case of a self-driving car, actuators include the steering system, accelerator pedal, brake pedal, and turn signals. These actuators allow the car to change lanes, accelerate, decelerate, and follow the prescribed route.

**Sensors**: Sensors provide the system with information about the environment and the car's current state. For a self-driving car, sensors can include cameras, LiDAR (Light Detection and Ranging), radar, GPS (Global Positioning System), and ultrasonic sensors. These sensors capture data about the car's surroundings, its position, nearby objects, road conditions, and potential obstacles.

We can use the PEAS (Performance, Environment, Actuators, Sensors) framework to summarize the example. This framework looks at the performance of a system, the environment in which it operates, the actuators that control it, and the sensors that measure its performance which can be summarized as follows:

- Performance Measure: Reaching the destination safely and efficiently, following traffic regulations, and having a well-thought-out route.
- Environment: The road system, the amount of traffic, the weather, other cars, people walking, road signs, and traffic lights all have an impact.
- Actuators: The steering wheel, gas pedal, brake pedal, and blinkers are all components of a vehicle control system.
- Sensors: Cameras, LiDAR, radar, GPS, ultrasonic sensors.

In this example, the objective of the self-driving car system is to navigate through the environment while ensuring safety, efficiency, and compliance with traffic regulations. The actuators control the car's movements, and the sensors provide the necessary input for the system to make informed decisions about navigating the environment. The PEAS framework helps to systematically analyze the different aspects of the system and its interactions with the environment.

## 1.3 Potted History of AI

The timeline of Artificial Intelligence (AI) is long and varied. This timeline shows the major milestones and advancements that have been made in AI:

- Pre-1950s (AI Prehistory):
  - Philosophical foundations and logical theories laid by philosophers like Aristotle, Descartes, and Leibniz explored questions about logic, reasoning, and the nature of mind that would later influence AI.
  - 1837: Charles Babbage's Analytical Engine.
  - 1936: Alan Turing's theoretical Turing Machine.
  - 1943: McCulloch & Pitts: Boolean circuit model of the brain.
  - Early research in cybernetics and neural networks.
- 1950s:
  - 1950: Alan Turing proposes the Turing Test, Computing Machinery and Intelligence [3].
  - 1956: The Dartmouth Workshop title *Artificial Intelligence*, marking the birth of AI as a field.
  - Development of early AI programming languages like LISP.
- 1960s:
  - 1965: Robinson's complete algorithm for logical reasoning.
  - AI discovers computational complexity.
  - Expansion of AI research in universities.
  - Early successes in problem solving algorithms and symbolic AI.
  - Development of robotic systems.
  - Neural network research is almost gone.
- 1970s:
  - "First AI Winter" period, marked by reduced funding and skepticism.
  - Focus on expert systems and domain-specific knowledge.
- 1980s:
  - Renewed interest and funding in AI.
  - Growth of expert systems in industries.
  - James Cameron and Gale Anne Hurd start the Terminator franchise by introducing synthetic intelligence.
  - Connectionism and neural networks return to popularity the revival.
- 1990s:
  - Expert systems industry busts: "Second AI Winter"
  - Advances in machine learning and data mining.
  - Improved algorithms for planning and optimization.
- 2000s:

  - Human-level AI back on the agenda.
  - Rise of modern machine learning, including support vector machines and random forests.
  - Success in AI competitions, such as IBM's Deep Blue defeating chess champion Garry Kasparov in 1997.

- 2010s:
  - Breakthroughs in deep learning and neural networks.
  - 2011: IBM's Watson wins Jeopardy!
  - JHU/APL starts the Intelligent Systems Center.
  - JHU/APL create the Intelligent Systems Branch.
  - Advances in natural language processing, computer vision, and autonomous vehicles.
  - Increased discussion of AI ethics and social impact.

- 2020s:
  - 2020: Johns Hopkins University Whiting School of Engineering, Engineering for Professionals starts the Artificial Intelligence Program.
  - Continued advancement in AI technologies across various domains.
  - Growing focus on AI explainability, fairness, and regulation.
  - Integration of AI in various industries and daily life.
  - ChatGPT takes the world by surprise.
  - No more "AI Winters."

The development of AI has been marked by a pattern of optimism, creativity, doubt, and revival. It is a field that has consistently drawn from a variety of areas and adapted to the latest technology and social requirements, leading to the multifaceted and ever-changing field we are familiar with today.

## 1.4 State-of-the-Art

The most recent developments and cutting-edge approaches in Artificial Intelligence are reflected in the current state of technology. Here is a summary of some advances in artificial intelligence:

**Natural Language Processing:** This is the capacity of a computer program to comprehend human language as it is spoken. Transformers and Attention Mechanisms have revolutionized natural language processing and comprehension. These tools have enabled a new level of understanding of the language, allowing for more accurate and efficient processing. This has had a profound impact on the field of natural language processing, leading to improved accuracy and speed in many applications. GPT-3 and BERT models have gained a great deal of prominence.

**Computer Vision:** A computer's capacity to interpret and understand the visual environment is demonstrated. Generative Adversarial Networks (GANs) are used to create realistic images, videos, and other media. GANs are a type of artificial intelligence that can generate new content based on existing data. They are composed of two neural networks, a generator and a discriminator, which work together to create new content. The generator creates new data, while the discriminator evaluates the generated data and provides feedback to the generator. This feedback helps the generator to improve its output and create more realistic content. GANs have been used to generate images, videos, and audio and have been used in a variety of applications, such as image synthesis, image-to-image translation, and text-to-image generation.

**Reinforcement Learning:** DeepMind, a Google-owned artificial intelligence company, has developed AlphaGo and AlphaZero, two AI programs that demonstrate remarkable skill in challenging games of Go and Chess. Both programs have achieved a level of play that surpasses that of any human.

Figure 4: Computer Vision example of an image created with the assistance of DALL-E2.

The field of AI is constantly advancing due to ongoing research, technological advances, and the increasing use of AI in various industries and applications. However, this progress brings with it a number of challenges, such as ethical, privacy, and security issues, which are currently being discussed and explored by the AI community. To further promote growth and diversity in the field, there is a strong emphasis on collaboration, open source initiatives, and interdisciplinary research.

## 2 Unintentional Funny Story

The funny story of Joe Bear.

One day Joe Bear was hungry. He asked his friend Irving Bird where some honey was. Irving told him there was a beehive in the oak tree. Joe threatened to hit Irving if he didn't tell him where some honey was. The End.

Stated differently

Joe Bear was hungry. He asked Irving Bird where some honey was. Irving refused to tell him, so Joe offered to bring him a worm if he'd tell him where some honey was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was . . .

## 2.1 A twist to the funny story of Joe Bear using AI

Incorporating AI into the Joe Bear funny story with the assistance of ChatGPT?

Once upon a time in the enchanted forest of Algorithms, there was a bear named Joe. Joe was no ordinary bear. He was a coding wizard and a data enthusiast who had taught himself artificial intelligence from scavenged textbooks and online tutorials he accessed through a Wi-Fi hotspot near the old ranger's station.

One fine day, Joe decided he needed a break from his machine learning projects and thought, "I could really use a new profile picture for my BearHub account where I share all my code." Joe had a brilliant idea: Why not design an AI-powered drone to take the perfect selfie?

He rummaged through discarded electronics he had collected over the years and pieced together a drone equipped with a camera and facial recognition software. After coding the algorithm, he hit the 'Run' button with his paw. The drone buzzed to life and floated into the air.

"Smile, Joe," the drone's automated voice said, as it positioned itself for the optimal angle.

Just as Joe struck his most dashing pose, a squirrel named Sally leapt out of nowhere onto the drone, mistaking it for a floating acorn. The drone, not programmed for squirrel interference, spun out of control and flew straight into a beehive.

The bees were furious! The AI drone tried its best to identify the bees as 'friendly or hostile' but concluded, "Error: species not recognized," before crashing into a tree.

Meanwhile, Joe Bear dashed through the forest, his AI drone following him haphazardly while still trying to take a picture. Bees and a very confused squirrel trailed them. Finally, Joe jumped into a river, and the swarm dispersed. Sally the squirrel grabbed her acorn and scampered off.

Soaking wet but relieved, Joe found his drone floating in the water. Miraculously, it had taken a picture at the very moment he plunged into the river. It was a mess of water droplets, a frantic bear face, and the tips of bee wings.

Joe uploaded the picture to his BearHub account with a caption that read, "Even AI can't account for the unpredictability of life, squirrels, or bees. #EpicFail #StillLearning."

The picture went viral in the animal kingdom, making Joe Bear a forest celebrity overnight. His failed yet hilarious AI experiment became a lesson in both the capabilities and limitations of artificial intelligence—and the importance of accounting for squirrels and bees in your code.

And so ends the AI adventure of Joe Bear, a bear whose funny escapade served as a reminder that even in a world of algorithms and machine learning, the unpredictability of life makes for the best stories.

## 3 Agents and Environments

The agent needs a set of sensors in order to percept the environment and act upon that environment through actuators. The agent then is able to take a set of correct actions based on the actuators available to the agent. The environment is the context within which an agent operates, and it can be as simple as a grid or as complex as the real world. This is depicted in Figure 5.

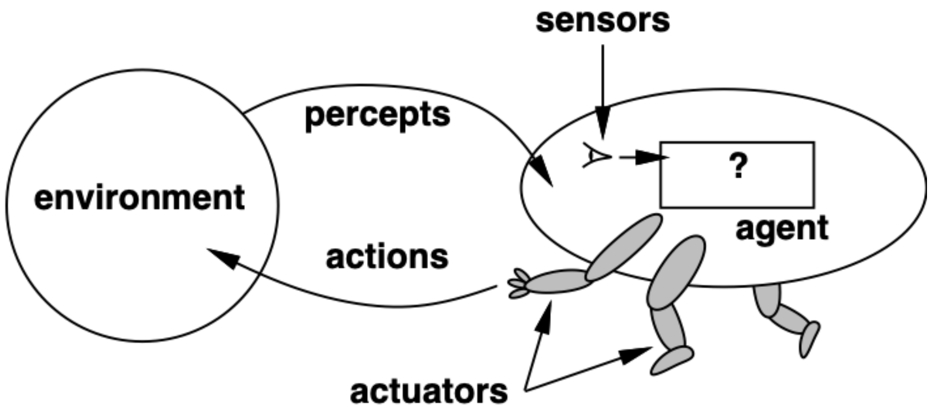

Figure 5: Agent's actions with an environment using sensors to stimulate the actuators.

**Agents**

- Simple Reflex Agents: These agents select actions on the basis of the current percept, ignoring the rest of the percept history. They are useful in simpler environments.
- Model-Based Reflex Agents: A bit more sophisticated, they maintain some kind of internal model that depends on the percept history. This allows them to handle a wider array of situations.
- Goal-Based Agents: These agents take actions to achieve goals, often using a form of search and planning.
- Utility-Based Agents: These are goal-based agents with a "happiness" metric. They try to maximize this utility as they operate.
- Learning Agents: These agents can learn from their environment to improve their performance over time.

**Types of Environments**

- Fully Observable vs Partially Observable: In a fully observable environment, agents can see all aspects at once. In partially observable environments, they might only be able to see part of it.
- Deterministic vs Stochastic: A deterministic environment is one in which any action has a fixed outcome, whereas in a stochastic environment, outcomes have some degree of randomness.
- Episodic vs Sequential: In episodic environments, actions in each episode do not affect future episodes. In sequential environments, current actions can have long-term consequences.

- Static: In a static environment, the state of the world does not change unless the agent takes an action.
- Discrete vs Continuous: Some environments have a finite number of percepts and actions (discrete), while others range over a continuum (continuous).
- Competitive vs Cooperative: Agents might be working against each other (competitive) or with each other (cooperative) to achieve goals.

**Interaction Between Agents and Environments**

- Percept Sequence: The history of everything the agent has ever perceived.
- Agent Function: Maps from percept sequences to actions. Defines the agent's behavior.
- Agent Program: Implements the agent function. Runs on the agent's architecture to produce action from percepts.

Understanding the types of agents and environments is crucial for designing a system where the agent can operate effectively. For instance, a learning agent may be best suited for complex, stochastic, and partially observable environments, while a simple reflex agent might suffice for a fully observable, deterministic environment.

This framework of agents and environments provides a foundational model for thinking about AI problems and designing intelligent systems.

## 3.1 Vacuum-cleaner World Example

The vacuum cleaner world is a classic example used in artificial intelligence (AI) education and research to explain basic agent-environment interactions. This example is often employed to introduce fundamental AI concepts in a simplified manner. In this vacuum cleaner example the vacuum percepts the environment as having a location being defined as dirty and takes a set of actions with the goal of cleaning the location with the available actuators as shown in Figure 6.

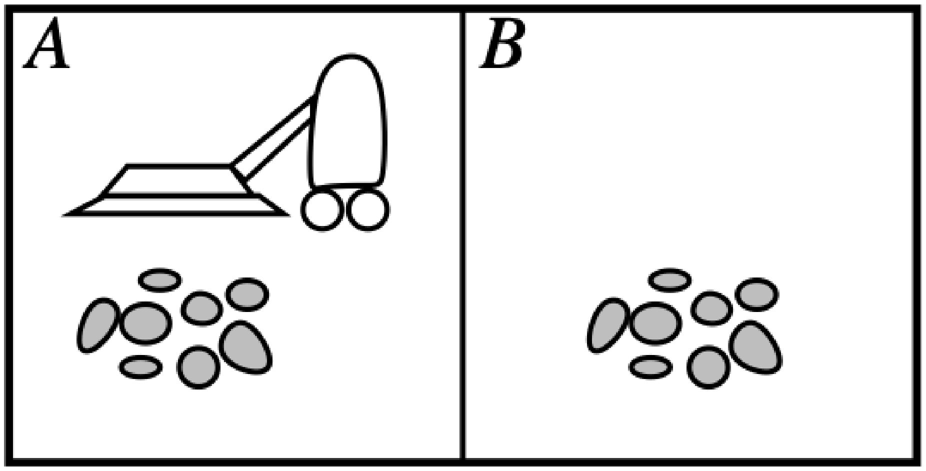

Figure 6: Vacuum cleaner example of taking actions based on the environment. *Figure 2.2 page 38 from the book: A vacuum-cleaner world with just two locations. Each location can be clean or dirty, and the agent can mover left or right and can clean the square that it occupies. Different versions of the vacuum world allow for different rules about what the agent can perceive, whether its actions always succeed, and so on.* [1].

In the environment the world consists of a grid of squares. Each square can either be clean or dirty. The vacuum cleaner is an agent that can occupy a single square at any given time. The percepts are the location and contents, e.g., $[A, Dirty]$ with the actions being $Left$, $Right$, $Suck$, $NoOp$.

### 3.1.1 A vacuum-cleaner agent

The vacuum cleaner agent can perceive its current square as clean or dirty. It has a limited set of actions it can perform, e.g., move left, move right, suck up dirt (clean), or do nothing. The objective of the vacuum cleaner agent is usually to clean all the squares. In some variations, the agent also has to minimize the energy used or time taken to clean the environment.

The types of agents include the following:

- Simple Reflex Agent: Takes an action based solely on the current percept (i.e., the cleanliness of the current square).
  - If the current square is dirty, suck up dirt.
  - If the current square is clean, move to an adjacent square.

- Model-Based Reflex Agent (shown below): Remembers past actions to avoid unnecessary moves. It might maintain an internal state of what squares are clean or dirty.
- Goal-Based Agent: Has a goal such as ”clean all the squares” and takes actions to achieve it, possibly planning multiple steps ahead.
- Utility-Based Agent: Tries to maximize a utility function that could combine several factors like cleanliness, time, and energy consumption.

```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action

if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

When designing your agent consider the following scenarios:

Scenario 1: The agent starts at the leftmost square.

- The agent moves right until it finds a dirty square.
- The agent cleans the dirty square and continues this process.

Scenario 2: The agent uses a more advanced strategy.

- The agent starts by moving right, cleaning any dirty squares it finds.
- When it reaches the end, it moves back to the leftmost square, making sure it didn’t miss any spots.

Scenario 3: The agent tries to minimize energy.

- The agent starts by scanning adjacent squares and only moves if it detects dirt, minimizing unnecessary movements.

| Percept sequence | Action |
|---|---|
| $[A, Clean]$ | $Right$ |
| $[A, Dirty]$ | $Suck$ |
| $[B, Clean]$ | $Left$ |
| $[B, Dirty]$ | $Suck$ |
| $[A, Clean]$, $[A, Clean]$ | $Right$ |
| $[A, Clean]$, $[A, Dirty]$ | $Suck$ |
| $\vdots$ | $\vdots$ |
| $[A, Clean]$, $[A, Clean]$, $[A, Clean]$ | $Right$ |
| $[A, Clean]$, $[A, Clean]$, $[A, Dirty]$ | $Suck$ |
| $\vdots$ | $\vdots$ |

Figure 7: Percept Sequence and Actions. *Figure 2.3 page 38 from the book: Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2. The agent cleans the current square if it is dirty, otherwise it moves to the other square. Note that the table is of unbounded size unless there is a restriction on the length of possible percept sequences.* [1].

Do the scenarios map to the table in Figure 7?
CAn you design a scenario based on the table in Figure 7?
How would you implement the **move right** function?
Can the scenarios be implemented in a small agent program?

The vacuum cleaner world example is an excellent introductory example for studying AI agents and their interactions with the environment. It simplifies many complexities found in real-world scenarios, making it easier to focus on core AI principles like perception, action, goal-setting, and utility maximization.

## 3.2 Environment Types

In the study of Artificial Intelligence (AI), understanding the environment within which an agent operates is crucial for designing effective algorithms and systems. The environment can be characterized in multiple dimensions, each affecting how an agent will behave and what type of problem-solving techniques will be most appropriate. Here's a breakdown of the terms:

**Observable vs. Partially Observable**

Observable: In a fully observable environment, an agent can access all the relevant information to make a decision. It knows the complete state of the environment at all times.

– Example: In a game of chess, both players can see the entire board and the positions of all pieces.

Partially Observable: In such an environment, the agent can only see a part of the world, requiring it to reason about its actions given incomplete information.

– Example: In a game of poker, the players cannot see each other's cards.

**Deterministic vs. Stochastic**

Deterministic: In a deterministic environment, every action has a predetermined outcome. Given a particular state and an action, the result is always the same.

– Example: Tic-tac-toe is deterministic; moving a piece to a particular square always results in the same new state.

Stochastic: In a stochastic environment, actions have probabilistic outcomes. The same action can lead to different states depending on random variables.

– Example: Trading in the stock market, where the outcome of buying or selling a stock is uncertain and influenced by random variables.

**Episodic vs. Sequential**

Episodic: In an episodic environment, the agent's experience is divided into episodes. Each episode is a standalone situation, and actions within it do not affect future episodes.

– Example: Quality control on an assembly line, where each item checked is a separate episode.

Sequential: In such an environment, the current actions can have long-term consequences affecting future decisions.

– Example: A game of chess, where each move affects the state of the board and influences future gameplay.

**Static vs. Dynamic**

Static: In a static environment, the world state does not change by itself. It only changes when the agent performs an action.

– Example: A crossword puzzle remains the same unless the player fills in a word.

Dynamic: In a dynamic environment, the state can change either due to the agent's actions or independently of them.

– Example: A self-driving car navigating in traffic, where other vehicles move independently of the car's actions.

**Discrete vs. Continuous**

Discrete: An environment where the set of possible states and actions are finite and distinct.

– Example: Chess, with a finite set of board configurations and moves.

Continuous: An environment where states or actions can take on a range of continuous values.

– Example: Piloting a drone, where the speed and direction can be any value within a continuous range.

**Single-Agent vs. Multi-Agent**

Single-Agent: An environment where only one agent is acting to achieve its goals. There's no competition or cooperation.

– Example: A Roomba vacuuming a room.

Multi-Agent: An environment where multiple agents interact, either by competing against each other or by cooperating to achieve common goals.

– Example: A game of soccer with multiple players on each team.

Understanding these dimensions helps researchers and practitioners better design, analyze, and implement AI agents suitable for various tasks and conditions.

| | Solitaire | Backgammon | Internet shopping | Taxi |
|---|---|---|---|---|
| Observable | Yes | Yes | No | No |
| Deterministic | Yes | No | Partly | No |
| Episodic | No | No | No | No |
| Static | Yes | Semi | Semi | No |
| Discrete | Yes | Yes | Yes | No |
| Single-agent | Yes | No | Yes (except auctions) | No |

Figure 8: In this table the environment types are mapped to known example problems that are relatable to users. [1].

## 3.3 Agent Types

In the study of artificial intelligence (AI), agents are broadly classified based on their capabilities, functionalities, and the level of complexity involved in their decision-making processes. Here's a list of the types of agents in this subsection:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents
- learning agents

Each of these types of agents has its own set of advantages and disadvantages and is suited for specific kinds of tasks and environments. The choice of agent type largely depends on the specific requirements of the problem you are trying to solve. All these can be turned into learning agents.

### 3.3.1 Simple reflex agents

In simple reflex agents the use of simple rules such as "if then" are used. The agents have the disadvantage of being short sighted when returning an action. No history is accounted for in the percept, only the current percept is used in the decision process to take an action.

```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action

if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

Figure 9: Reflex Vacuum Agent*Figure 2.8 page 49 from the book: The agent program for a simple reflex agent in the two-location vacuum environment. This program implements the agent function tabulated in Figure 2.3 of the book* [1].

**Example**

```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program)))

(defun make-reflex-vacuum-agent-program ()
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (cond ((eq status 'dirty) 'Suck)
              ((eq location 'A) 'Right)
              ((eq location 'B) 'Left)))))
```

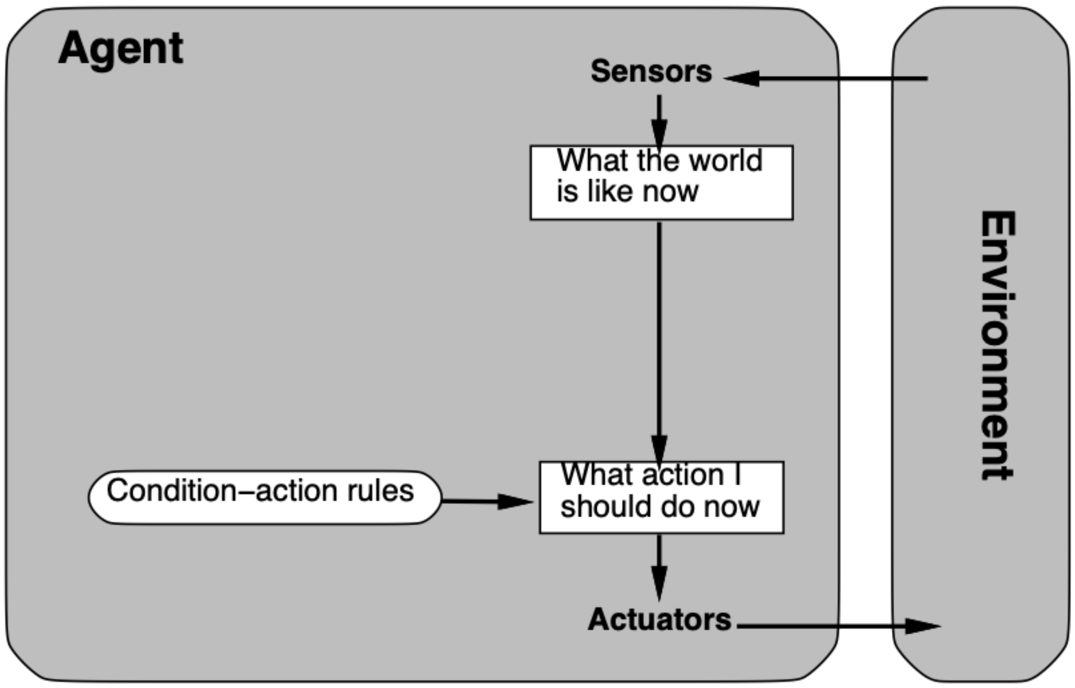

Figure 10: Simple Reflex Agent *Figure 2.9 page 50 from the book: Schematic diagram of a simple reflex agent. We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process* [1].

```
**function** SIMPLE-REFLEX–AGENT( percept) **returns** an action
**persistent**: rules, a set of conditions-action rules
state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, rule)
action ← rule.ACTION
**return** action
```

Figure 11: Simple Reflex Agent *Figure 2.10 page 51 from the book: A simple reflex agent. It acts according to a rule whose condition matched the current state, as defined by the percept* [1].

### 3.3.2 Reflex agents with state

```
**function** REFLEX-AGENT-WITH-STATE( percept) **returns** an action
state ← UPSTATESTATE(state, percept)
rule ← RULE-MATCH(state, rule)
action ← rule.ACTION
state ← UPDATESTATE(state, action)
**return** action
```

Figure 12: A reflex agent with a state algorithm [1].

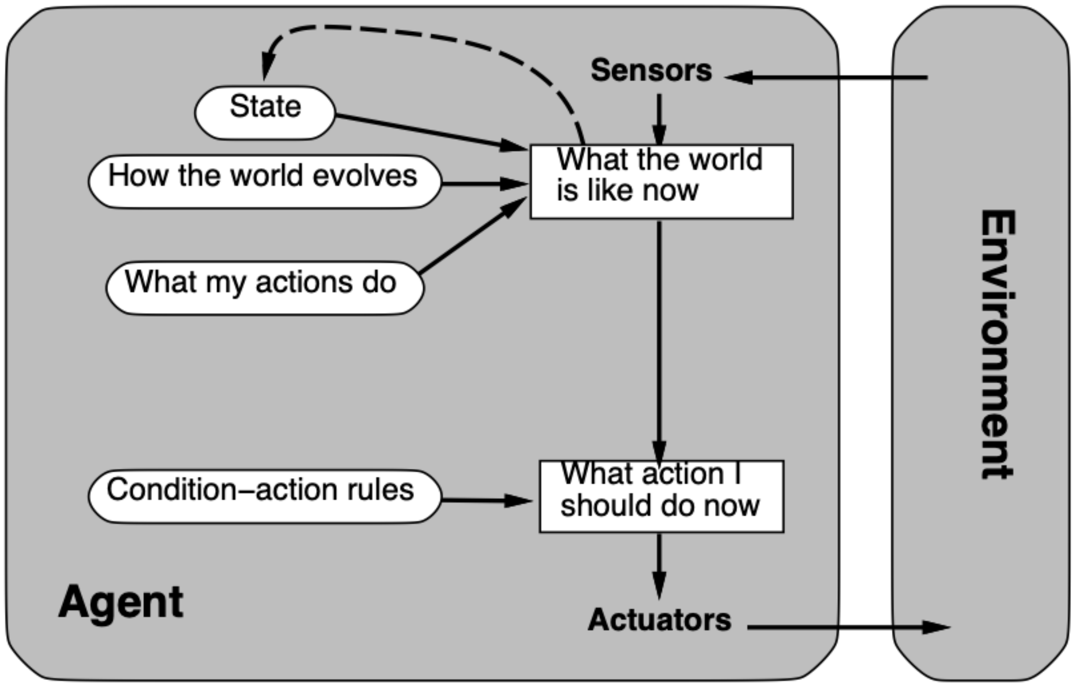

Figure 13: *Figure 2.11 page 52 from the book: A model-baed reflex agent* [1].

**Example**

```
(defun make-reflex-vacuum-agent-with-state-program ()
  (let ((last-A infinity) (last-B infinity))
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (incf last-A) (incf last-B)
        (cond
         ((eq status 'dirty)
          (if (eq location 'A) (setq last-A 0) (setq last-B 0))
          'Suck)
         ((eq location 'A) (if (> last-B 3) 'Right 'NoOp))
         ((eq location 'B) (if (> last-A 3) 'Left 'NoOp)))))))
```

```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action
static: last_A, last_B, numbers, initially $\infty$

if status = Dirty then ...
```

Figure 14: A model-baed reflex agent with a state algorithm [1].

### 3.3.3 Goal-based and Utility-based agents

The design of goal- and utility-based agents depends on the structure of the task environment. The simplest such agents, for example those in chapters 3 and 10 [1], compute the agent's entire future sequence of actions in advance before acting at all. This strategy works for static and deterministic environments which are either fully-known or unobservable.

For fully-observable and fully-known static environments a policy can be computed in advance which gives the action to by taken in any given state.

For partially-observable environments the agent can compute a conditional plan, which specifies the sequence of actions to take as a function of the agent's perception. In the extreme, a conditional plan gives the agent's response to every contingency, and so it is a representation of the entire agent function.

In all cases it may be either intractable or too expensive to compute everything out in advance. Instead of a conditional plan, it may be better to compute a single sequence of actions which is likely to reach the goal, then monitor the environment to check whether the plan is succeeding, repairing or replanning if it is not. It may be even better to compute only the start of this plan before taking the first action, continuing to plan at later time steps.

### 3.3.4 Goal-based agents

Pseudocode for simple goal-based agent is given in Figure 15. GOAL-ACHIEVED tests to see whether the current state satisfies the goal or not, doing nothing if it does. PLAN computes a sequence of actions to take to achieve the goal. This might return only a prefix of the full plan, the rest will be computed after the prefix is executed. This agent will act to maintain the goal: if at any point the goal is not satisfied it will (eventually) replan to achieve the goal again.

```
function GOAL-BASED-AGENT( percept) returns an action
persistent: state, ...
            model, ...
               goal, ...
               plan, ...
               action, ...
state ← ...
plan ← ... Conditional Statements
action ← ...
return action
```

Figure 15: A goal-based agent.

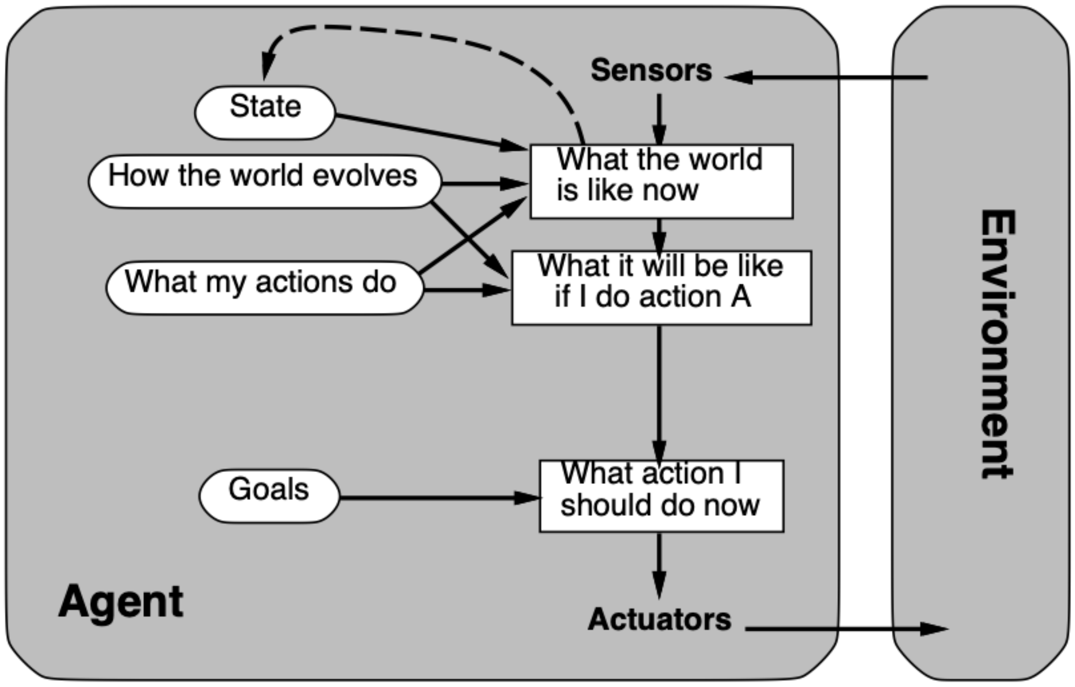
Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Figure 16: Goal-Based Agent *Figure 2.13 page 54 from the book: A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goal* [1].

### 3.3.5 Utility-based agents

At this level of abstraction the utility-based agent is not much different than the goal- based agent, except that action may be continuously required (there is not necessarily a point where the utility function is "satisfied"). Pseudocode is given in Figure 17.

```
function UTILITY-BASED-AGENT( percept) returns an action
persistent: state, ...
            model, ...
               utility-function, ...
               plan, ...
               action, ...
state ← ...
Conditional Statements
plan ← ...
action ← ...
plan ← ...
return action
```

Figure 17: A utility-based agent.

Let consider the game of tic-tac-toe where and agent needs to make a decision on the next location to place a piece on the board using the utility-based agent shown in Figure 17. The game is covered in more detail in the Game Theory module, with this said, the game of Tic-Tac-Toe there are set know rules to help play the game when two-players sit at a table to play on a piece of paper, https://en.wikipedia.org/wiki/Tic-tac-toe. In the following board the locations are represented from 0 to 8 in order to determine where specific pieces lie on the tic-tac-toe board during any point in time in the game. We consider this the current state of the game during the evaluation of the game using the utility-based agent.

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

Let's examine a state where where there is one X along the diagonal represented as locations board-location(2) with an O located at position board-location(0) as follows:

| | | |
|---|---|---|
| O | | X |
| | | |
| | | |

At this point a model for evaluation needs to be selected to determine if a win for the agent is possible, if so take it, determine if the opponent is about to win, if so block it, if a win is not present then the agent needs a model for evaluating the board and where to place the next piece on the board. Now a utility-function is needed. Taking the heuristic evaluation function from Russell and Norvig [1], based on Claude Sahnnon's paper [2], allows a function to evaluate the board based on the direction for each row, column and diagonal using a mathematical manner. Below is a linear evaluation function that is a good complement within the utility-function.

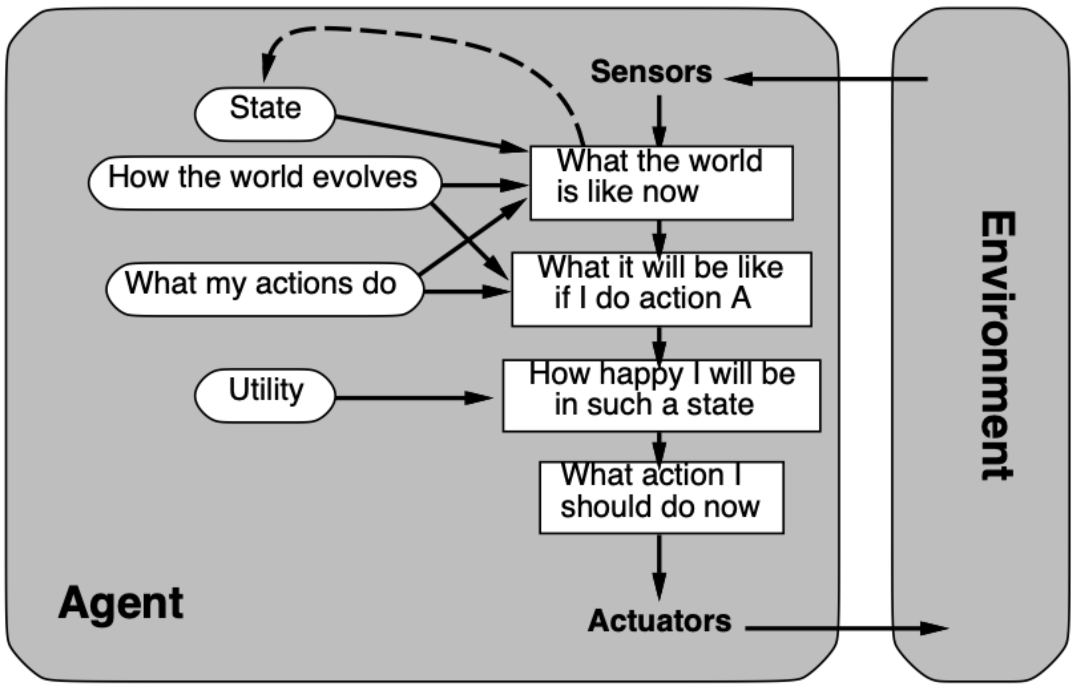
State
How the world evolves
What my actions do
Utility
Sensors
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

Figure 18: Utility-Based Agent *Figure 2.14 page 55 from the book: A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among stats of the world. Then it chooses that action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome* [1].

$$Eval = 3 * X_2 + X_1 - (3 * O_2 + O_1) \tag{1}$$

The variable $X_n$ is defined as the number of rows, columns and diagonals that contain exactly $n$ $X$'s and no $O$'s, where the following variables are defined as:

- $X_2$ is the number of lines with 2 X's and a blank
- $X_1$ is the number of lines with 1 X and 2 blanks
- $O_2$ is the number of lines with 2 O's and a blank
- $O_1$ is the number of lines with 1 O and 2 blanks

Additionally, the number of lines that are summed include the rows, columns and diagonals in the game board at the current state. One way of expanding this evaluation function is shown as follows:

$$Eval(d) = 3 * X(d) - O(d) \tag{2}$$

where $d = 0, 2, \cdots, 8$ with the following definitions for the evaluation of calculating the rows, columns and diagonals:

1. $d = 0$ for the first row
2. $d = 1$ for the second row
3. $d = 2$ for the third row
4. $d = 3$ for the first column
5. $d = 4$ for the second column
6. $d = 5$ for the third column
7. $d = 6$ for the first diagonal
8. $d = 7$ for the second diagonal

The combination of a utility function with an evaluation function allows a function to be developed as shown in Figure 19.

```
function UTILITY-FUNCTION-WITH-EVALUATION-FUNCTION( location) returns list
list ← initialize as an empty list

for d ← 0 to 8 do
        if row = empty
                list[d] ← 0
        if row = full
                list[d] ← -10
else list[d] ← Eval(d)
return list
```

Figure 19: Utility-Function-with-Evaluation-Function.

### 3.3.6 Learning agents

Learning agents improve their performance based on experience, often consisting of four components: learning element, performance element, a critic, and a problem generator. These agents are suitable for complex, dynamic, and stochastic environments where pre-programmed solutions are impractical. Some limitations may require a lot of computational resources and data for effective learning.

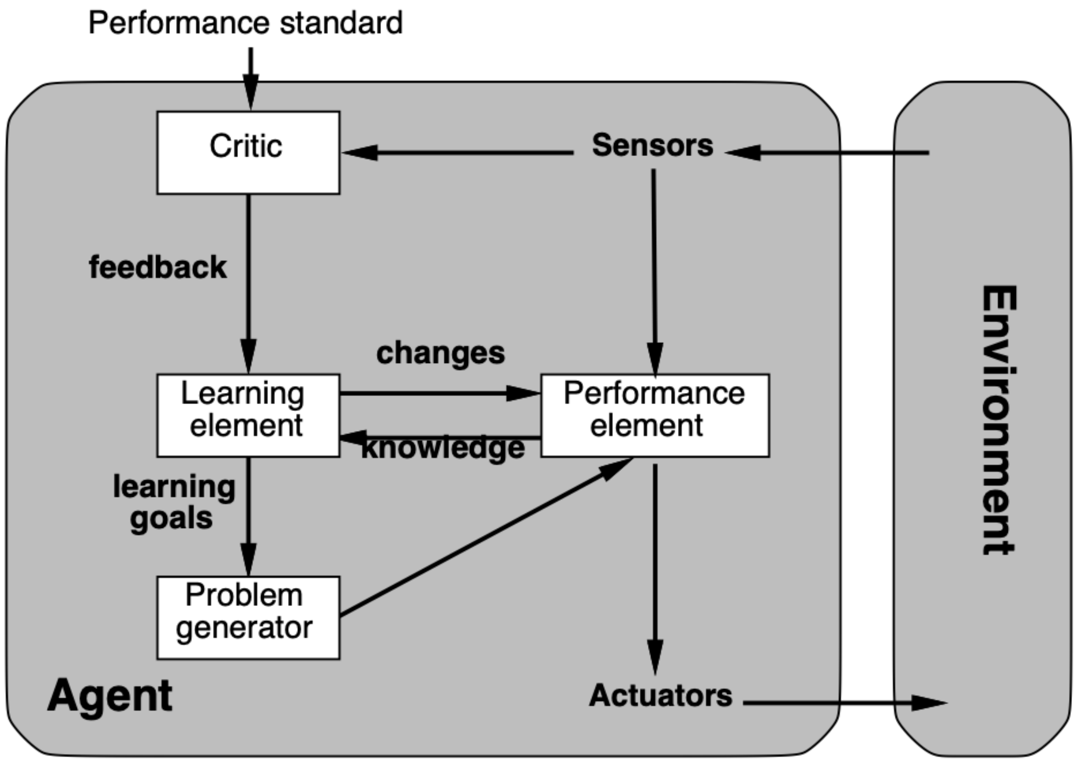

Figure 20: General Learning Agent *Figure 2.15 page 56 from the book: A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance* [1].

## References

[1] Stuart Russell and Peter Norvig. **Artificial Intelligence: A Modern Approach**. 4th. Prentice Hall, 2020.

[2] C. E. Shannon. “Programming a Computer for Playing Chess”. In: **Philosophical Magazine**. 7th ser. 41.314 (1950).

[3] Alan M. Turing. “Computing Machinery and Intelligence”. In: **Mind** 59.236 (1950), pp. 433 –460.

[4] P. Winston. **Artificial Intelligence**. 3rd. Pearson, 1992.