Artificial Intelligence I

Ibrahim Nasser

November 7, 2025

Contents

1 Recap

1.1 Set Theory

Def 1.1. Set: A collection of different things (called **elements** or **members** of the set). We can represent sets by listing the elements within curly brackets, e.g. $\{a, b, c\}$ or by describing the elements via a property, e.g. $\{x \mid x \mod 2 = 0\}$ (the set of even numbers). We can state element-hood ($a \in S$) or not ($b \notin S$)

Def 1.2. Set Equality: $A \equiv B :\equiv \forall x. x \in A \Leftrightarrow x \in B$

Def 1.3. Subset: $A \subseteq B :\equiv \forall x. x \in A \Rightarrow x \in B$

Def 1.4. Proper Subset: $A \subset B :\equiv (A \subseteq B) \wedge (A \not\equiv B)$

Def 1.5. Super Set: $A \supseteq B :\equiv \forall x. x \in B \Rightarrow x \in A$

Def 1.6. Proper Superset: $A \supset B :\equiv (A \supseteq B) \wedge (A \not\equiv B)$

Def 1.7. Set Union: $A \cup B := \{x \mid x \in A \vee x \in B\}$

Def 1.8. Set Intersection: $A \cap B := \{x \mid x \in A \wedge x \in B\}$

Def 1.9. Disjoint: Two sets A, B are **disjoint** iff $A \cap B = \Phi$

Def 1.10. Set Difference: $A \setminus B := \{x \mid x \in A \wedge x \notin B\}$

Def 1.11. Power Set: $\mathcal{P}(A) := \{S \mid S \subseteq A\}$ (the set of all subsets)

Def 1.12. Cartesian Product: $A \times B := \{(a, b) \mid a \in A \wedge b \in B\}$, (a, b) is a pair

Def 1.13. Family: A set of sets

Def 1.14. Topology: A family of open sets

Def 1.15. Indexing Function: Let I and X be sets. A function $f : I \to X$ is called an indexing function. The set I is called the index set. For each $i \in I$, we define $x_i := f(i)$, here i is called the index (or parameter) of x_i.

Def 1.16. Indexed Family: Given an indexing function $f : I \to X$, the set of values $f(I) = \{f(i) : i \in I\}$ is called an indexed family. It is often written as $(x_i)_{i \in I}$, $\langle x_i \rangle_{i \in I}$, or $\{x_i\}_{i \in I}$.

Def 1.17. Union over a Family: Let $(X_i)_{i \in I}$ be an indexed family, then $\bigcup_{i \in I} X_i := \{x \mid \exists i \in I. x \in X_i\}$

Def 1.18. Intersection over a Family: Let $(X_i)_{i \in I}$ be an indexed family, then $\bigcap_{i \in I} X_i := \{x \mid \forall i \in I. x \in X_i\}$

Def 1.19. n-fold Cartesian Product: $\prod_{i=1}^{n} X_i := X_1 \times \cdots \times X_n := \{\langle x_1, \cdots, x_n \rangle \mid \forall i. 1 \leq i \leq n \Rightarrow x_i \in X_i\}$ where $\langle x_1, \cdots, x_n \rangle$ is called an n-tuple.

Def 1.20. n-dim Cartesian Space: $X^n := \{\langle x_1, \cdots, x_n \rangle \mid 1 \leq i \leq n \Rightarrow x_i \in X\}$ where $\langle x_1, \cdots, x_n \rangle$ is called a vector.

Def 1.21. Size of a Set: The size $\Gamma(A)$ of a set A is the number of elements in A.

1.2 Relations

Def 1.22. Relation: $R \subseteq A \times B$ is a (binary) relation between A and B

Def 1.23. Relation on: a relation $R \subseteq A \times B$ where $A = B$ is called a *relation on* A

Def 1.24. Total: A relation $R \subseteq A \times B$ is called *total* (*left total*) iff $\forall x \in A. \exists y \in B. (x, y) \in R$

Def 1.25. Converse Relation: $R^{-1} \subseteq B \times A := \{(y, x) \mid (x, y) \in R\}$ is the converse relation of R

Def 1.26. Relation Composition: The composition of $R \subseteq A \times B$ and $S \subseteq B \times C$ is defined as $R \circ S := \{(a, c) \in A \times C \mid \exists b \in B. (a, b) \in R \wedge (b, c) \in S\}$

A relation on A: $R \subseteq A \times A$ is called:

- **reflexive** on A, iff $\forall a \in A. (a, a) \in R$
- **irreflexive** on A, iff $\forall a \in A. (a, a) \notin R$
- **symmetric** on A, iff $\forall a, b \in A. (a, b) \in R \Rightarrow (b, a) \in R$
- **asymmetric** on A, iff $\forall a, b \in A. (a, b) \in R \Rightarrow (b, a) \notin R$
- **antisymmetric** on A, iff $\forall a, b \in A. (a, b) \in R \wedge (b, a) \in R \Rightarrow a = b$

- **transitive** on A, iff $\forall a, b, c \in A.(a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R$
- **equivalence relation** on A, iff R is reflexive symmetric, and transitive

Def 1.27. Equality Relation: The equality relation is an equivalence relation on any set.

Def 1.28. Divides Relation: The *divides* relation on the integers is defined as $| \subseteq \mathbb{Z} \times \mathbb{Z}$, where $| = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid \exists k \in \mathbb{Z} : b = a \cdot k\}$. We write $a \mid b$ to denote $(a, b) \in |$ (read as *"a divides b"*).

Def 1.29. Congruence Modulo: For a fixed $n \in \mathbb{N}$ with $n \geq 1$, the *congruence modulo n* relation on the integers is defined as

$$\equiv_n \subseteq \mathbb{Z} \times \mathbb{Z}, \qquad \equiv_n = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid n \mid (a - b)\}.$$

We write $a \equiv b \pmod{n}$ to denote $(a, b) \in \equiv_n$ (read as *"a is congruent to b modulo n"*).

Def 1.30. Parital Ordering ($\preceq$): A relation $R \subseteq A \times A$ is called a (non-strict) **partial ordering** on A, iff R is reflexive, antisymmetric, and transitive on A.

Def 1.31. Strict Partial Ordering ($\prec$): A relation $R \subseteq A \times A$ is called a **strict partial ordering** on A, iff R is irreflexive, and transitive on A.

Def 1.32. Comparable: In a non-strict partial ordering, two elements a, b are comparable if either $a \preceq b$ or $b \preceq a$. If neither holds, they are incomparable.

Def 1.33. Linear Order: A partial ordering is called linear (or total) on A, iff all elements in A are comparable, i.e. if $(x, y) \in R$ or $(y, x) \in R$ for all $x, y \in A$. For example, the $\leq$ relation is a linear order on $\mathbb{N}$. However, the "divides" ($|$) relation is a non-strict partial ordering on $\mathbb{N}$ that is not linear (e.g., neither $2 \mid 3$ nor $3 \mid 2$).

1.3 Functions

Def 1.34. Partial function: A relation $R \subseteq X \times Y$ is a *partial function* from X to Y ($f : X \rightharpoonup Y$) iff $\forall x \in X, \forall y_1, y_2 \in Y.\big((x, y_1) \in R \wedge (x, y_2) \in R \Rightarrow y_1 = y_2\big)$ (i.e. there is at most one $y \in Y$ with $(x, y) \in f$)

Def 1.35. dom, codom: We call X the domain ($\mathrm{dom}(f)$), and Y the codomain ($\mathrm{codom}(f)$)

Def 1.36. Function Application: Instead of writing $(x, y) \in f$ we write $f(x) = y$

Def 1.37. Undefined: we call a partial function $f : X \rightharpoonup Y$ undefined at $x \in X$, iff $\forall y \in Y.(x, y) \notin f$ ($f(x) = \bot$).

Def 1.38. Function: A partial function $f : X \rightharpoonup Y$ is called a *function* (or *total function*) from X to Y iff it is a total relation. $(\forall x \in X.\exists^1 y \in Y : (x, y) \in f)$

> **Note**
>
> A partial function guarantees uniqueness, but not existence. So each x has at most one y (some might have none). A total function guarantees both uniqueness and existence (at most and at least $\rightarrow$ exactly one)

A function $f : X \to Y$ is called

- **injective** iff $\forall x_1, x_2 \in X.f(x_1) = f(x_2) \Rightarrow x_1 = x_2$
- **surjective** iff $\forall y \in Y.\exists x \in X.f(x) = y$
- **bijective** iff f is injective and surjective

Def 1.39. Function Composition: If $f : A \to B$ and $g : B \to C$ are functions, then we call $g \circ f : A \to C; x \longmapsto g(f(x))$ the composition of g and f (read g after f).

Def 1.40. Image and Preimage: Let $f : X \to Y$ be a function, $X' \subseteq X$ and $Y' \subseteq Y$, then we call

- $f(X') := \{y \in Y \mid \exists x \in X'.(x, y) \in f\}$ the **image** of X' under f,
- $\mathrm{Im}(f) := f(X)$ the **image** of f, and
- $f^{-1}(Y') := \{x \in X \mid \exists y \in Y'.(x, y) \in f\}$ the **preimage** of Y' under f.

Def 1.41. Cardinality: We say that a set A is **finite** and has **cardinality** $\Gamma(A) \in \mathbb{N}$, iff there is a bijective function $f : A \to \{n \in \mathbb{N} \mid n < \Gamma(A)\}$.

Def 1.42. Countably Infinite: We say that a set A is countably infinite, iff there is a bijective function $f : A \to \mathbb{N}$.

Def 1.43. Countable: A set A is called countable, iff it is finite or countably infinite.

Def 1.44. Curried Function Type: Let X, Y, Z be sets. An *uncurried* function is written as

$$f : X \times Y \to Z, \quad f(x, y) = E(x, y).$$

The same function can equivalently be written in *curried* form as

$$f : X \to Y \to Z, \quad f(x)(y) = E(x, y).$$

Thus in the curried notation the function takes one argument at a time: for $x \in X$, the value $f(x)$ is itself a function $Y \to Z$, and applying it to $y \in Y$ yields $f(x)(y) \in Z$.

Def 1.45. Invertible: Let A and B be sets and $f : A \to B$ a function, then f is called invertible, iff there is a function $g : B \to A$, such that $f \circ g = \mathrm{Id}_B$. g is called the inverse function (or just inverse) of f and is written as f^{-1}.

1.4 Mathematical Structures

Def 1.46. Formulae: A mathematical formula can be a **mathematical statement** (clause) which can be true or false (e.g. $x > 5, 3 + 5 = 7$), or a **mathematical object** (e.g. $3, n, x^2 + y^2 + z^2, \int_1^0 x^{3/2} dx$)

Def 1.47. Mathematical Structure: A mathematical structure combines multiple mathematical objects (the components) into a new object. The components usually have names by which they can be referenced. Given a definition of a mathematical structure S, we say that any object that conforms to that is an instance of S.

Def 1.48. Group: A group is a mathematical structure $\langle G, \circ, e, .^{-1} \rangle$ that consists of:

- a base set G of objects,
- an operation $\circ : G \times G \to G$, such that $\forall a, b \in G. a \circ b \in G$ and $\forall a, b, c \in G. (a \circ b) \circ c = a \circ (b \circ c)$,
- a unit $e \in G$, such that $\forall a \in G. e \circ a = a \circ e = a$, and
- the inverse function $.^{-1} : G \to G$, such that $\forall a \in G. a \circ a^{-1} = a^{-1} \circ a = e$

An example of a group is the set $G = \mathbb{Z}$, with the operation $+ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$; then $e = 0$ and $.^{-1}$ is $\lambda x \in \mathbb{Z}. -x$. We write it as $\langle \mathbb{Z}, +, 0, - \rangle$.

Question: can we do the same for multiplication where $e = 1$? If we think about it, we would be stuck on finding the $.^{-1}$ component of the group. What can we always multiply with any integer and get $e = 1$ back? Take for example 1, we can multiply it with $\frac{1}{1}$ (great!). Take -1, we can multiply it with $\frac{1}{-1}$ (also great!). But take any other integer, e.g. 2, we must multiply it with $\frac{1}{2}$ to get 1 back, however, $\frac{1}{2} \notin \mathbb{Z}$, it is however $\in \mathbb{Q}$! The multiplicative group would only make sense iff $G = \mathbb{Q} \setminus \{0\}$. We write that as $\langle \mathbb{Q} \setminus \{0\}, *, 1, a \mapsto \frac{1}{a} \rangle$.

But we need some multiplication structure that works for $\mathbb{Z}$!

Def 1.49. Monoid: A monoid is a mathematical structure $\langle M, \circ, e \rangle$ that consists of:

- A base set M of objects,
- An operation $\circ : M \times M \to M$, such that for all $a, b \in M$, we have $a \circ b \in M$ (closure),
- Associativity: for all $a, b, c \in M$, $(a \circ b) \circ c = a \circ (b \circ c)$, and
- A unit element $e \in M$, such that for all $a \in M. e \circ a = a \circ e = a$.

Hence, the mathematical structure $\langle \mathbb{Z}, *, 1 \rangle$ is a monoid (a group missing an inverse function).

We saw that $\langle \mathbb{Z}, +, 0, - \rangle$ is a group under addition. And $\langle \mathbb{Z}, *, 1 \rangle$ is a monoid under multiplication. Now, we want to do everyday arithmetics with integers; we want expressions where we have additions and multiplication (e.g. $a * (b + c)$) which evaluates to $(a * b) + (a * c)$.

For that, we need a mathematical structure where both addition and multiplication live together.

Def 1.50. Ring: A ring is a mathematical structure $\langle R, +, 0, -, *, 1 \rangle$ consisting of:

- A base set R.

- An addition operation $+ : R \times R \to R$ such that $\langle R, +, 0, - \rangle$ is a group (called abelian group).
- A multiplication operation $* : R \times R \to R$ such that $\langle R, *, 1 \rangle$ is a monoid.
- Distributivity: for all $a, b, c \in R, a * (b + c) = (a * b) + (a * c), (a + b) * c = (a * c) + (b * c)$

Hence, mixing components from the group $\langle \mathbb{Z}, +, 0, - \rangle$ and the monoid $\langle \mathbb{Z}, *, 1 \rangle$ leaves us with a ring $\langle \mathbb{Z}, +, 0, -, *, 1 \rangle$

Def 1.51. Magma: A magma is a mathematical structure $\langle M, \circ \rangle$ consisting of:

- A base set M, and
- A binary operation $\circ : M \times M \to M$.

Example: $\langle \mathbb{Z}, - \rangle$ (integers under subtraction). $\langle \mathbb{N}, + \rangle$ (natural numbres under addition)

Def 1.52. Semigroup: A semigroup (magma with associativity) is a mathematical structure $\langle S, \circ \rangle$ consisting of:

- A base set S,
- A binary opertation $\circ : S \times S \to S$, and
- Associativity: for all $a, b, c \in S, (a \circ b) \circ c = a \circ (b \circ c)$

Example: $\langle \mathbb{Z}, + \rangle$ (integers under addition).

1.5 Formal Languages and Grammars

Def 1.53. Alphabet: An alphabet is a finite set; we call each element $a \in A$ a **character**, and an n-tuple $s \in A^n$ a **string** (of *length* n over A). We often write a string $\langle c_1, \cdots, c_n \rangle$ as "$c_1 \cdots c_n$", for instance `"abc"` for $\langle a, b, c \rangle$

Def 1.54. Empty String: Let A be an alphabet, then $A^0 = \{\langle\rangle\}$, where $\langle\rangle$ is the unique 0-tuple. We consider $\langle\rangle$ as the string of length 0 and call it the **empty string** and denote it with ϵ.

Def 1.55. String Length: Given a string s, we denote its length with $|s|$.

Def 1.56. String Concatenation: The concatenation $\text{conc}(s, t)$ of two strings $s = \langle s_1, \cdots, s_n \rangle \in A^n$ and $t = \langle t_1, \cdots, t_m \rangle \in A^m$ is defined as $s + t$ or simply `st`

Def 1.57. Kleene Plus/Star: Let A be an alphabet, then we define the sets:

- $A^+ := \bigcup_{i \in \mathbb{N}^+} A^i$ (*nonempty strings*), and
- $A^* := A^+ \cup \{\epsilon\}$ (*strings*).

Def 1.58. Formal Language: A set $L \subseteq A^*$ is called a *formal language* over A. For example, If $A = \{a, b\}$ then:

$$A^* = \bigcup_{n=0}^{\infty} A^n$$

where $A^0 = \epsilon$, $A^1 = \{a, b\}$, $A^2 = \{a, b, ab, ba\}$, and so on ...

$$A^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, \cdots\}$$
$$A^+ = \{a, b, aa, ab, ba, bb, aaa, aab, aba, abb, \cdots\}$$

A formal language might be $L = \{a, b, aab, bbb, \cdots\}$

Def 1.59. Repeating Chars: We use $c^{[n]}$ for the string that consists of the character `c` repeated n times. **Examples:**

- Let $A = \{a, b\}$, $a^{[5]} =$ `"aaaaa"`.
- The set $M := \left\{ \text{ba}^{[n]} \mid n \in \mathbb{N} \right\}$ of strings that start with character b followd by an arbitrary numbers of a's is a formal language over $A = \{a, b\}$

> **Note**
>
> A formal language might be infinite and even undecidable even if the alphabet A is finite. For example, let $A = \{a, b\}$ then the formal language $L = \{a, ab, bba\}$ is finite but the formal language $L = \{ a^{[n]} \mid n \in \mathbb{N} \}$ is infinite.

Since we cannot list all strings in a formal language L because there are infinitely many, we need a **finite description** that can generate all strings in L. We need **Grammars**.

Def 1.60. Phrase Structure Grammar: A phrase structure grammar (*type 0 grammar, unrestricted grammar, grammar*) is a tuple $\langle N, \Sigma, P, S\rangle$ where

- N is a finite set of **nonterminal symbols**,
- Σ is a finite set of **terminal symbols**. (members of $N \cup \Sigma$ are called symbols).
- P is a finite set of **production rules**: pairs $p := h \to b$ (also written as $h \Rightarrow b$), where $h \in (\Sigma \cup N)^* N (\Sigma \cup N)^*$ and $b \in (\Sigma \cup N)^*$. The string h is called the **head** of p and b the **body**.
- $S \in N$ is a distinguished symbol called the **start symbol** (also **sentence symbol**).

The sets N and Σ are assumed to be disjoint. Any word $w \in \Sigma^*$ is called a **terminal word**.

> Note
>
> Production rules map strings with at least one nonterminal to arbitrary other strings

Notation: If we have n production rule $h \to b_i$ sharing a head, we often write $h \to b_1 \mid \cdots \mid b_n$ instead.

Example: A simple grammar for english sentences:

$$\begin{aligned} \text{S} &::= \text{NP Vi} \\ \text{NP} &::= \text{Article N} \\ \text{Article} &::= \text{the} \mid \text{a} \mid \text{an} \\ \text{N} &::= \text{dog} \mid \text{teacher} \mid \cdots \\ \text{Vi} &::= \text{sleeps} \mid \text{smells} \mid \cdots \end{aligned}$$

Def 1.61. Lexical: A production rule whose head is a single nonterminal and whose body consists of a single terminal is called **lexical** or a **lexical insertion rule**.

Def 1.62. Lexicon of a Grammar: The subset of lexical rules of a grammar G is called the **lexicon** of G

Def 1.63. Vocabulary: The set of body symbols are called the vocabulary (or the alphabet).

Def 1.64. Lexical Categories of a Grammar: The nonterminals that appear in the heads of lexical rules are called lexical categories of the grammar.

Def 1.65. Structural Rules: The non lexicon production rules are called structural.

Def 1.66. Phrasal categories: The nonterminals in the heads of production rules that expand into other nonterminals are called phrasal (syntactic) categories.

Def 1.67. G-Derivation: Given a phrase structure grammar $G := \{N, \Sigma, P, S\}$, we say G **derives** $t \in (\Sigma \cup N)^*$ from $s \in (\Sigma \cup N)^*$ in one step, iff there is a production rule $p \in P$ with $p = h \to b$ and there are $u, v \in (\Sigma \cup N)^*$, such that $s = uhv$ and $t = ubv$. We write $s \to_G^p t$ (or $s \to_G t$ if p is clear from the context) and use $\to_G^*$ for the reflexive transitive closure of $\to_G$. We call $s \to_G^* t$ a G-derivation of t from s.

Def 1.68. Sentential Form: Given a phrase structure grammar $G := \{N, \Sigma, P, S\}$, we say that $s \in (\Sigma \cup N)^*$ is a **sentential form** of G, iff $S \to_G^* s$

Def 1.69. Sentence: A sentential form that does not contain nonterminals is called a sentence of G, we also say that G **accepts** s. We say that G **rejects** s, iff it is not a sentence of G.

Def 1.70. Language Generation: The language $L(G)$ of G is the set of its sentences. We say that $L(G)$ is **generated** by G.

Def 1.71. Equivalent Grammars: We call two grammars **equivalent**, iff they have the same languages.

Def 1.72. Universal Grammar: A grammar G is said to be **universal** if $L(G) = \Sigma^*$

Def 1.73. Syntactic Analysis: Syntactic analysis (parsing / syntax analysis) is the process of analyzing a string of symbols, either in a formal or a natural language by means of a grammar.

Note

The shape of the grammar determines the size of its language

Def 1.74. Context-Sensitive: We call a grammar context-sensitive (or type 1), if the bodies of production rules have no less symbols than the heads.

Def 1.75. Context-Free: We call a grammar context-free (or type 2), iff the heads have exactly one symbol.

Def 1.76. Regular: We call a grammar regular (or type 3, or REG), if additionally the bodies are empty or consist of a nonterminal, optionally followed by a terminal symbol.

Note

By extension, a formal language L is called *context-sensitive / context-free / regular*, iff it is the language of a respective grammar. Context-free grammars are sometimes `CFGs` and context-free languages `CFLs`.

2 Introduction

Def 2.1. Artificial intelligence (AI): The capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making

Def 2.2. Symbolic AI: A subfield of Artificial Intelligence (AI) based on the assumption that many aspects of intelligence can be achieved by the manipulation of symbols, combining them into meaningful structures (expressions) and manipulating them (using processes) to produce new expressions.

Def 2.3. Statisitical AI: Remedies the two shortcomings of symbolic AI approaches: that all concepts represented by symbols are crisply defined, and that all aspects of the world are knowable/representable in principle. Statistical AI adopts sophisticated mathematical models of uncertainty and uses them to create more accurate world models and reason about them.

Def 2.4. Subsymbolic AI (a.k.a connectionism/neural AI): A subfield of AI that posits that intelligence is inherently tied to brains, where information is represented by a simple sequence pulses that are processed in parallel via simple calculations realized by neurons, and thus concentrates on neural computing.

Def 2.5. Embodied AI: Posits that intelligence cannot be achieved by reasoning about the state of the world (symbolically, statistically, or sub-symbolically), but must be embodied i.e. situated in the world, equipped with a "body" that can interact with it via sensors and actuators. Here, the main method for realizing intelligent behavior is by learning from the world.

Def 2.6. Reasoning: The process of producing valid arguments and predictive world models. There are three forms of reasoning:

- **deductive reasoning** to produce new knowledge from existing knowledge. *Example:* All humans are mortal. Socrates is a human. Therefore, Socrates is mortal.
- **inductive reasoning** to produce knowledge from perception. *Example:* The sun has risen every morning in recorded history. Therefore, the sun will rise tomorrow.
- **abductive reasoning** to produce explanations for observations and given knowledge. *Example:* The grass is wet this morning. If it rained last night, that would explain it. Therefore, it probably rained.

Def 2.7. Inference: The act or process of reaching a **conclusion** about something from known facts or evidence (jointly called **premises**)

Def 2.8. Formal Logic: The science of deductively valid inferences, meaning arguments whose conclusions necessarily follow from their premises by virtue of their form (their structure), regardless of the topic.

Def 2.9. Agent: An agent is a structure $A := \langle \mathcal{P}, \mathcal{A}, f \rangle$ where:

- $\mathcal{P}$ is a set of percepts
- $\mathcal{A}$ is a set of actions
- f is a function $f : \mathcal{P}^* \to \mathcal{A}$ that maps from percepts to actions.

In other words, an agent is anything that perceives its environment via sensors and acts on it with actuators.

Def 2.10. Performance Measure: A function that evaluates a sequence of environments.

Def 2.11. Rational: An agent is called **rational**, if it chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date.

Def 2.12. PEAS: To design rational agents, we must specify the PEAS components: Performance Measure, Environment, Actuators, and Sensors.

Def 2.13. Environment Types: For an agent a we classify the environment e of a by its **type**, which is one of the following. We call e

1. *fully observable*, iff the a's sensors give it access to the complete state of the environment at any point in time; otherwise, we call it *partially observable*.
2. *deterministic*, iff the next state of the environment is completely determined by the current state and a's actions; otherwise, *stochastic*.
3. *episodic*, iff a's experience is divided into atomic episodes, where it perceives and then performs a single action, and the next episode does not depend on previous ones. Otherwise, *sequential*.
4. *dynamic*, iff the environment can change without an action performed by a; otherwise, *static*. If the environment does not change but a's performance measure does, we call e *semidynamic*.

5. *discrete*, iff the sets of e's state and a's actions are couuntable; otherwise, *continuous*.
6. *single-agent*, iff only a acts on e; otherwise *multi-agent*.

Def 2.14. Reflex: An agent $\langle \mathcal{P}, \mathcal{A}, f \rangle$ is called a **reflex agent**, iff it only takes the last percept into account when choosing an action, i.e. $f(p_1, \cdots, p_k) = f(p_k)\ \forall p_1 \cdots p_k \in \mathcal{P}$

Def 2.15. Model-Based Agent: A model-based agent $\langle \mathcal{P}, \mathcal{A}, \mathcal{S}, \mathcal{T}, s_0, S, a \rangle$ is an agent $\langle \mathcal{P}, \mathcal{A}, f \rangle$ whose actions depend on:

- a **world model**: a set $\mathcal{S}$ of possible *states*, and a start state $s_0 \in \mathcal{S}$.
- a **transition model** $\mathcal{T}$ that predicts a new state $\mathcal{T}(s, a)$ from a state s and an action a.
- a **sensor model** S that given a state s and a percept p determines a new state $S(s, p)$.
- an **action function** $a : \mathcal{S} \to \mathcal{A}$ that given a state $s \in \mathcal{S}$ selects the next action $a \in \mathcal{A}$.

Note

If the agent is in state s then it took action a and now perceives p, then the agent state will become $s' = S(p, \mathcal{T}(s, a))$ and accordingly take action $a' = a(s')$.

Def 2.16. Goal Based Agent: A goal based agent $\langle \mathcal{P}, \mathcal{A}, \mathcal{S}, \mathcal{T}, s_0, S, a, \mathcal{G} \rangle$ is a model based agent with an explicit set of goals. It consists of:

- a set of internal states $\mathcal{S}$ and an initial state $s_0 \in \mathcal{S}$,
- a transition model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$,
- a state update function $S : \mathcal{P} \times \mathcal{S} \to \mathcal{S}$,
- a set of goals $\mathcal{G}$,
- a goal conditioned action function $a : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ selecting an action given a state and the goals.

Def 2.17. Utility-Based Agent: A utility-based agent uses a world model along with a utility function that models its preferences among the states of that world. It chooses the action that maximizes the expected utility.

Def 2.18. Learning Agent: A learning agent is an agent that can improve its own behavior through experience. It is composed of four main components:

- the performance element, which selects actions based on the agent's current percepts and represents its existing knowledge or behavior,
- the learning element, which improves the performance element over time by analyzing feedback and identifying how to make better decisions,
- the critic, which evaluates the agent's performance according to a given performance standard and provides feedback to the learning element,
- the problem generator, which suggests new and informative actions or experiences that help the agent explore and learn more effectively.

Def 2.19. State Representation: We call a state representation **atomic**, iff it has no internal structure (black box). However, iff each state is characterized by attributes and their values then the representation is **factored**. A **structured** state representation is when include representations of objects, their properties and relationships.