

# Languages and Algorithms for Artificial Intelligence (Module 3)

Last update: 11 March 2024

Academic Year 2023 – 2024  
Alma Mater Studiorum · University of Bologna

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Notations . . . . .	1
1.1.1	Strings . . . . .	1
1.1.2	Tasks encoding . . . . .	1
1.1.3	Asymptotic notation . . . . .	2
<b>2</b>	<b>Turing Machine</b>	<b>3</b>
2.1	$k$ -tape Turing Machine . . . . .	3
2.2	Computation . . . . .	4
2.3	Universal Turing Machine . . . . .	4
2.4	Computability . . . . .	5
2.4.1	Undecidable functions . . . . .	5
2.4.2	Rice's theorem . . . . .	6
<b>3</b>	<b>Complexity</b>	<b>7</b>
3.1	Polynomial time . . . . .	7
3.2	Exponential time . . . . .	7
3.3	<b>NP</b> class . . . . .	8

# 1 Introduction

**Computational task** Description of a problem.

Computational task

**Computational process** Algorithm to solve a task.

Computational  
process

**Algorithm (informal)** A finite description of elementary and deterministic computation steps.

## 1.1 Notations

**Set of the first  $n$  natural numbers** Given  $n \in \mathbb{N}$ , we have that  $[n] = \{1, \dots, n\}$ .

### 1.1.1 Strings

**Alphabet** Finite set of symbols.

Alphabet

**String** Finite, ordered, and possibly empty tuple of elements of an alphabet.

String

The empty string is denoted as  $\varepsilon$ .

**Strings of given length** Given an alphabet  $S$  and  $n \in \mathbb{N}$ , we denote with  $S^n$  the set of all the strings over  $S$  of length  $n$ .

**Kleene star** Given an alphabet  $S$ , we denote with  $S^* = \bigcup_{n=0}^{\infty} S^n$  the set of all the strings over  $S$ .

Kleene star

**Language** Given an alphabet  $S$ , a language  $\mathcal{L}$  is a subset of  $S^*$ .

Language

### 1.1.2 Tasks encoding

**Encoding** Given a set  $A$ , any element  $x \in A$  can be encoded into a string of the language  $\{0, 1\}^*$ . The encoding of  $x$  is denoted as  $\lfloor x \rfloor$  or simply  $x$ .

Encoding

**Task function** Given two countable sets  $A$  and  $B$  representing the domain, a task can be represented as a function  $f : A \rightarrow B$ .

Task

When not stated,  $A$  and  $B$  are implicitly encoded into  $\{0, 1\}^*$ .

**Characteristic function** Boolean function of form  $f : \{0, 1\}^* \rightarrow \{0, 1\}$ .

Characteristic  
function

Given a characteristic function  $f$ , the language  $\mathcal{L}_f = \{x \in \{0, 1\}^* \mid f(x) = 1\}$  can be defined.

**Decision problem** Given a language  $\mathcal{M}$ , a decision problem is the task of computing a boolean function  $f$  able to determine if a string belongs to  $\mathcal{M}$  (i.e.  $\mathcal{L}_f = \mathcal{M}$ ).

Decision problem

### 1.1.3 Asymptotic notation

**Big O** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is  $O(g)$  if  $g$  is an upper bound of  $f$ .

Big O

$$f \in O(g) \iff \exists \bar{n} \in \mathbb{N} \text{ such that } \forall n > \bar{n}, \exists c \in \mathbb{R}^+ : f(n) \leq c \cdot g(n)$$

**Big Omega** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is  $\Omega(g)$  if  $g$  is a lower bound of  $f$ .

Big Omega

$$f \in \Omega(g) \iff \exists \bar{n} \in \mathbb{N} \text{ such that } \forall n > \bar{n}, \exists c \in \mathbb{R}^+ : f(n) \geq c \cdot g(n)$$

**Big Theta** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is  $\Theta(g)$  if  $g$  is both an upper and lower bound of  $f$ .

Big Theta

$$f \in \Theta(g) \iff f \in O(g) \text{ and } f \in \Omega(g)$$

## 2 Turing Machine

### 2.1 $k$ -tape Turing Machine

**Tape** Infinite one-directional line of cells. Each cell can hold a symbol from a finite alphabet  $\Gamma$ . Tape

**Tape head** A tape head reads or writes one symbol at a time and can move left or right on the tape.

**Input tape** Read-only tape where the input will be loaded.

**Work tape** Read-write auxiliary tape used during computation.

**Output tape** Read-write tape that will contain the output of the computation.

**Remark.** Sometimes the output tape is not necessary and the final state of the computation can be used to determine a boolean outcome.

**Instructions** Given a finite set of states  $Q$ , at each step, a machine can: Instructions

**Read** from the  $k$  tape heads.

**Replace** the symbols under the writable tape heads, or leave them unchanged.

**Change** state.

**Move** each of the  $k$  tape heads to the left or right, or leave unchanged.

**$k$ -tape Turing Machine (TM)** A Turing Machine working on  $k$  tapes (one of which is the input tape) is a triple  $(\Gamma, Q, \delta)$ :  $k$ -tape Turing Machine (TM)

- $\Gamma$  is a finite set of tape symbols. We assume that it contains a blank symbol ( $\square$ ), a start symbol ( $\triangleright$ ), and the digits 0, 1.
- $Q$  is a finite set of states. The initial state is  $q_{\text{init}}$  and the final state is  $q_{\text{halt}}$ .
- $\delta$  is the transition function that describes the instructions allowed at each step. It is defined as:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^{k-1} \times \{\text{L}, \text{S}, \text{R}\}^k$$

By convention, when the state is  $q_{\text{halt}}$ , the machine is stuck (i.e. it cannot change state or operate on the tapes):

$$\delta(q_{\text{halt}}, \{\sigma_1, \dots, \sigma_k\}) = (q_{\text{halt}}, \{\sigma_1, \dots, \sigma_k\}, (\text{S}, \dots, \text{S}))$$

**Theorem 2.1.1** (Turing Machine equivalence). The following computational models have, with at most a polynomial overhead, the same expressive power: 1-tape TMs,  $k$ -tape TMs, non-deterministic TMs, random access machines,  $\lambda$ -calculus, unlimited register machines, programming languages (Böhm-Jacopini theorem), ...

## 2.2 Computation

**Configuration** Given a TM  $\mathcal{M} = (\Gamma, Q, \delta)$ , a configuration  $C$  is described by:

Configuration

- The current state  $q$ .
- The content of the tapes.
- The position of the tape heads.

**Initial configuration** Given the input  $x \in \{0, 1\}^*$ , the initial configuration  $\mathcal{I}_x$  is described as follows:

- The current state is  $q_{\text{init}}$ .
- The first (input) tape contains  $\triangleright x \square \dots$ . The other tapes contain  $\triangleright \square \dots$ .
- The tape heads are positioned on the first symbol of each tape.

**Final configuration** Given an output  $y \in \{0, 1\}^*$ , the final configuration is described as follows:

- The current state is  $q_{\text{halt}}$ .
- The output tape contains  $\triangleright y \square \dots$ .

**Computation (string)** Given a TM  $\mathcal{M} = (\Gamma, Q, \delta)$ ,  $\mathcal{M}$  returns  $y \in \{0, 1\}^*$  on input  $x \in \{0, 1\}^*$  (i.e.  $\mathcal{M}(x) = y$ ) in  $t$  steps if:

Computation  
(string)

$$\mathcal{I}_x \xrightarrow{\delta} C_1 \xrightarrow{\delta} \dots \xrightarrow{\delta} C_t$$

where  $C_t$  is a final configuration for  $y$ .

**Computation (function)** Given a TM  $\mathcal{M} = (\Gamma, Q, \delta)$  and a function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ ,  $\mathcal{M}$  computes  $f$  iff:

Computation  
(function)

$$\forall x \in \{0, 1\}^* : \mathcal{M}(x) = f(x)$$

If this holds,  $f$  is a computable function.

**Computation in time  $T$**  Given a TM  $\mathcal{M}$  and the functions  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and  $T : \mathbb{N} \rightarrow \mathbb{N}$ ,  $\mathcal{M}$  computes  $f$  in time  $T$  iff:

Computation in time  
 $T$

$$\forall x \in \{0, 1\}^* : \mathcal{M}(x) \text{ returns } f(x) \text{ in at most } T(|x|) \text{ steps}$$

**Decidability in time  $T$**  Given a function  $f : \{0, 1\}^* \rightarrow \{0, 1\}$ , the language  $\mathcal{L}_f$  is decidable in time  $T$  iff  $f$  is computable in time  $T$ .

Decidability in time  
 $T$

## 2.3 Universal Turing Machine

**Turing Machine encoding** Given a TM  $\mathcal{M} = (\Gamma, Q, \delta)$ , the entire machine can be described by  $\delta$  through tuples of form:

$$Q \times \Gamma^k \times Q \times \Gamma^{k-1} \times \{L, S, R\}^k$$

It is therefore possible to encode  $\delta$  into a binary string and consequently create an encoding  $\sqcup \mathcal{M} \sqcup$  of  $\mathcal{M}$ .

The encoding should satisfy the following conditions:

1. For every  $x \in \{0, 1\}^*$ , there exists a TM  $\mathcal{M}$  such that  $x = \sqcup \mathcal{M} \sqcup$ .

2. Every TM is represented by an infinite number of strings. One of them is the canonical representation.

**Theorem 2.3.1** (Universal Turing Machine (UTM)). There exists a TM  $\mathcal{U}$  such that, for every binary strings  $x$  and  $\alpha$ , it emulates the TM defined by  $\alpha$  on input  $x$ :

Universal Turing Machine (UTM)

$$\mathcal{U}(x, \alpha) = \mathcal{M}_\alpha(x)$$

where  $\mathcal{M}_\alpha$  is the TM defined by  $\alpha$ .

Moreover,  $\mathcal{U}$  simulates  $\mathcal{M}_\alpha$  with at most  $CT \log(T)$  time overhead, where  $C$  only depends on  $\mathcal{M}_\alpha$ .

## 2.4 Computability

### 2.4.1 Undecidable functions

**Theorem 2.4.1** (Existence of uncomputable functions). There exists a function  $uc : \{0, 1\}^* \rightarrow \{0, 1\}^*$  that is not computable by any TM.

Uncomputable functions

*Proof.* Consider the following function:

$$uc(\alpha) = \begin{cases} 0 & \text{if } \mathcal{M}_\alpha(\alpha) = 1 \\ 1 & \text{if } \mathcal{M}_\alpha(\alpha) \neq 1 \end{cases}$$

If  $uc$  was computable, there would be a TM  $\mathcal{M}$  that computes it (i.e.  $\forall \alpha \in \{0, 1\}^* : \mathcal{M}(\alpha) = uc(\alpha)$ ). This will result in a contradiction:

$$uc(\perp \mathcal{M} \perp) = 0 \iff \mathcal{M}(\perp \mathcal{M} \perp) = 1 \iff uc(\perp \mathcal{M} \perp) = 1$$

Therefore,  $uc$  cannot be computed.  $\square$

**Halting problem** Given an encoded TM  $\alpha$  and a string  $x$ , the halting problem aims to determine if  $\mathcal{M}_\alpha$  terminates on input  $x$ . In other words:

Halting problem

$$\text{halt}(\perp(\alpha, x) \perp) = \begin{cases} 1 & \text{if } \mathcal{M}_\alpha \text{ stops on input } x \\ 0 & \text{otherwise} \end{cases}$$

**Theorem 2.4.2.** The halting problem is undecidable.

*Proof.* Note: this proof is slightly different from the traditional proof of the halting problem.

Assume that  $\text{halt}$  is decidable. Therefore, there exists a TM  $\mathcal{M}_{\text{halt}}$  that decides it.

We can define a new TM  $\mathcal{M}_{uc}$  that uses  $\mathcal{M}_{\text{halt}}$  such that:

$$\mathcal{M}_{uc}(\alpha) = \begin{cases} 1 & \text{if } \mathcal{M}_{\text{halt}}(\alpha, \alpha) = 0 \text{ (i.e. } \mathcal{M}_\alpha(\alpha) \text{ diverges)} \\ \begin{cases} 0 & \text{if } \mathcal{M}_\alpha(\alpha) = 1 \\ 1 & \text{if } \mathcal{M}_\alpha(\alpha) \neq 1 \end{cases} & \text{if } \mathcal{M}_{\text{halt}}(\alpha, \alpha) = 1 \text{ (i.e. } \mathcal{M}_\alpha(\alpha) \text{ converges)} \end{cases}$$

This results in a contradiction:

- $\mathcal{M}_{uc}(\perp \mathcal{M}_{uc} \perp) = 1 \iff \mathcal{M}_{\text{halt}}(\perp \mathcal{M}_{uc} \perp, \perp \mathcal{M}_{uc} \perp) = 0 \iff \mathcal{M}_{uc}(\perp \mathcal{M}_{uc} \perp) \text{ diverges}$
- $\mathcal{M}_{\text{halt}}(\perp \mathcal{M}_{uc} \perp, \perp \mathcal{M}_{uc} \perp) = 1 \Rightarrow \mathcal{M}_{uc} \text{ is not computable by Theorem 2.4.1.}$

□

**Diophantine equation** Polynomial equality with integer coefficients and a finite number of unknowns.

Diophantine equation

**Theorem 2.4.3** (MDPR). Determining if an arbitrary diophantine equation has a solution is undecidable.

## 2.4.2 Rice's theorem

**Semantic language** Given a language  $\mathcal{L} \subseteq \{0, 1\}^*$ ,  $\mathcal{L}$  is semantic if:

Semantic language

- Any string in  $\mathcal{L}$  is an encoding of a TM.
- If  $\llbracket \mathcal{M} \rrbracket \in \mathcal{L}$  and the TM  $\mathcal{N}$  computes the same function of  $\mathcal{M}$ , then  $\llbracket \mathcal{N} \rrbracket \in \mathcal{L}$ .

A semantic language can be seen as a set of TMs that have the same property.

**Trivial language** A language  $\mathcal{L}$  is trivial iff  $\mathcal{L} = \emptyset$  or  $\mathcal{L} = \{0, 1\}^*$

**Theorem 2.4.4** (Rice's theorem). If a semantic language is non-trivial, then it is undecidable (i.e. any decidable semantic language is trivial).

Rice's theorem

*Proof idea.* Assuming that there exists a non-trivial decidable semantic language  $\mathcal{L}$ , it is possible to prove that the halting problem is decidable. Therefore,  $\mathcal{L}$  is undecidable. □



## 3 Complexity

**Complexity class** Set of tasks that can be computed within some fixed resource bounds. Complexity class

### 3.1 Polynomial time

**Deterministic time (DTIME)** Let  $T : \mathbb{N} \rightarrow \mathbb{N}$  and  $\mathcal{L}$  be a language.  $\mathcal{L}$  is in **DTIME**( $T(n)$ ) iff there exists a TM that decides  $\mathcal{L}$  in time  $O(T(n))$ . Deterministic time (**DTIME**)

**Polynomial time (P)** The class **P** contains all the tasks computable in polynomial time: Polynomial time (**P**)

$$\mathbf{P} = \bigcup_{c \geq 1} \mathbf{DTIME}(n^c)$$

**Remark.** **P** is closed to various operations on programs (e.g. composition of programs)

**Remark.** In practice, the exponent is often small.

**Remark.** **P** considers the worst case and is not always realistic. Other alternative computational models exist.

**Church-Turing thesis** Any physically realizable computer can be simulated by a TM with an arbitrary time overhead. Church-Turing thesis

**Strong Church-Turing thesis** Any physically realizable computer can be simulated by a TM with a polynomial time overhead. Strong Church-Turing thesis

**Remark.** If this thesis holds, the class **P** is robust (i.e. does not depend on the computational device) and is therefore the smallest class of bounds.

**Deterministic time for functions (FDTIME)** Let  $T : \mathbb{N} \rightarrow \mathbb{N}$  and  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ .  $f$  is in **FDTIME**( $T(n)$ ) iff there exists a TM that computes it in time  $O(T(n))$ . Deterministic time for functions (**FDTIME**)

**Polynomial time for functions (FP)** The class **FP** is defined as: Polynomial time for functions (**FP**)

$$\mathbf{FP} = \bigcup_{c \geq 1} \mathbf{FDTIME}(n^c)$$

**Remark.** It holds that  $\forall \mathcal{L} \in \mathbf{P} \Rightarrow f_{\mathcal{L}} \in \mathbf{FP}$ , where  $f_{\mathcal{L}}$  is the characteristic function of  $\mathcal{L}$ . Generally, the contrary does not hold.

### 3.2 Exponential time

**Exponential time (EXP/FEXP)** The **EXP** and **FEXP** classes are defined as: Exponential time (**EXP/FEXP**)

$$\mathbf{EXP} = \bigcup_{c \geq 1} \mathbf{DTIME}(2^{n^c}) \quad \mathbf{FEXP} = \bigcup_{c \geq 1} \mathbf{FDTIME}(2^{n^c})$$

**Theorem 3.2.1.** The following hold:

$$\mathbf{P} \subset \mathbf{EXP} \quad \mathbf{FP} \subset \mathbf{FEXP}$$

### 3.3 NP class

**Certificate** Given a set of pairs  $\mathcal{C}_{\mathcal{L}}$  and a polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$ , we can define the language  $\mathcal{L}$  such that:

$$\mathcal{L} = \{x \in \{0, 1\}^* \mid \exists y \in \{0, 1\}^{p(|x|)} : (x, y) \in \mathcal{C}_{\mathcal{L}}\}$$

Given a string  $w$  and a certificate  $y$ , we can exploit  $\mathcal{C}_{\mathcal{L}}$  as a test to check whether  $y$  is a certificate for  $w$ :

$$w \in \mathcal{L} \iff (w, y) \in \mathcal{C}_{\mathcal{L}}$$

**Nondeterministic TM (NDTM)** TM that has two transition functions  $\delta_0, \delta_1$  and, at each step, non-deterministically chooses which one to follow. A state  $q_{\text{accept}}$  is always present:

- A NDTM accepts a string iff one of the possible computations reaches  $q_{\text{accept}}$ .
- A NDTM rejects a string iff none of the possible computations reach  $q_{\text{accept}}$ .

**Nondeterministic time (NDTIME)** Let  $T : \mathbb{N} \rightarrow \mathbb{N}$  and  $\mathcal{L}$  be a language.  $\mathcal{L}$  is in **NDTIME**( $T(n)$ ) iff there exists a NDTM that decides  $\mathcal{L}$  in time  $O(T(n))$ .

**Remark.** A NDTM  $\mathcal{M}$  runs in time  $T : \mathbb{N} \rightarrow \mathbb{N}$  iff for every input, any possible computation terminates in time  $O(T(n))$ .

#### Complexity class NP

**NDTM formulation** The class **NP** contains all the tasks computable in polynomial time by a nondeterministic TM:

$$\mathbf{NP} = \bigcup_{c \geq 1} \mathbf{NDTIME}(n^c)$$

**Verifier formulation** Let  $\mathcal{L} \subseteq \{0, 1\}^*$  be a language.  $\mathcal{L}$  is in **NP** iff there exists a polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$  and a polynomial TM  $\mathcal{M}$  (verifier) such that:

$$\mathcal{L} = \{x \in \{0, 1\}^* \mid \exists y \in \{0, 1\}^{p(|x|)} : \mathcal{M}(\perp(x, y)) = 1\}$$

In other words,  $\mathcal{L}$  is the language of the strings that can be verified by  $\mathcal{M}$  in polynomial time using a certificate  $y$  of polynomial length.

**Theorem 3.3.1.**  $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$