# Formal languages and computation models

Guy Perrier

# Bibliography


- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman - *Introduction to Automata Theory, Languages, and Computation* - Addison Wesley, 2006.

# 1 - General ideas

1. Introduction
2. Formal languages
3. Computation models

# 1.1 - Introduction

- The course aims at giving the theoretical foundations to understand the notions of **programming language**, **interpretation** and **compilation** of programming and data languages, **complexity** of a computation.

- The general framework is the **theory of formal languages**. Linguistics greatly contributed to its first developments (Chomsky 56) because of the proximity of formal languages with natural languages.

- Natural languages are essentially **ambiguous** and **evolutionary,** whereas formal languages are essentially **non ambiguous** and **frozen**.

# 1.1 - Introduction

- By analogy with linguistics, a computer program can be viewed as an utterance in a given language. As for an utterance in a natural language, a program must be **parsed** to be **interpreted** or to be **translated** into another language. Usually, the translation is performed to a lower level language; in this case, it is called **compilation.**

- Parsing is related to the **syntax** of the program whereas interpretation and compilation are related to its **semantics**.

# 1.1 - Introduction

- Parsing is performed by an abstract machine according to a **computation model**. This model depends on the concerned language. Generally, the **complexity of parsing** increases with the **expressivity of the language**.

- Parsing concerns not only programming languages but various kinds of formal languages (HTML, XML, Postscript, Tex, Latex…).

# 1.2 - Formal languages

- A **formal language** L over a finite alphabet $\Sigma$ of symbols is a part of the set $\Sigma^*$ of **words** composed of symbols from $\Sigma$. The class of languages defined over $\Sigma$ is equipped with the following operations : intersection, union, concatenation, Kleene closure, complementation.

- If L is infinite, it is important to have a computation procedure for **recognizing** L , that is for deciding if any word from $\Sigma^*$ belongs to L. If such a procedure exists, L is said to be **recursive**.

- If there exists only a computation procedure for **enumerating** L, L is said to be **recursively enumerable**.

# 1.3 - Computation models

- The foundation of the computation theory is due to **Turing** , who has formalized the concept of computation by designing general abstract machines (1936).

- A **Turing Machine** (TM) is composed of two parts:
  - ✓ an infinite **tape** in one-to-one correspondence with $\mathbb{Z}$ and a pointer at the current position which can be read and written;
  - ✓ A **control unit** which controls the forward and backward movements of the pointer as the actions of reading and writing on the tape.

# 1.3 - Computation models

- Formally, a TM is defined as a 5-uple ($Q$, $\Sigma$, $q_0$, $F$, $\tau$) such that :

  ✓ $Q$ is the finite set of states of the unit control.

  ✓ $\Sigma$ is the finite tape alphabet of symbols plus a blank symbol $\perp$ which is not in $\Sigma$.

  ✓ $q_0$ is a particular element of $Q$, the start state of the TM.

  ✓ $F$ is a subset of $Q$, the accepting states of the TM.

  ✓ $\tau$ is a transition relation which associates a source state $q_1$ from $Q$, a tape input symbol *a* from $\Sigma \cup \{\perp\}$, with an output symbol *b* from $\Sigma \cup \{\perp\}$, a direction *d* of movement, which can takes the values L or R, and a target state $q_2$ from $Q$. This is denoted : $\tau\ (q_1, a, b, d, q_2)$.

# 1.3 - Computation models

- The **principle of a Turing Machine** :

  ✓ **Initialisation**: Initially, a word *s* from $\sum^*$ is written on the tape. The other positions are filled with the blank symbol $\perp$. A pointer indicates the position of the first symbol of *s*. (if *s* is empty, the initial position of the pointer does not matter).
  The control unit is initially in the state $q_0$.

  ✓ **Transition step** : a transition can occur if the control unit is in a state $q_1$, if the pointer indicates the symbol *a* on the tape, and if the TM has a transition $(q_1, a, b, d, q_2)$ in its transition relation $\tau$. In this case, the symbol *a* is replaced by *b* on the tape, the pointer moves to the next position on the left if $d = L$ and on the right if $d = R$. Finally, the control unit moves to state $q_2$.

  ✓ **Termination** :If in a configuration of the TM, no transition is possible, the TM stops. At this moment, if the unit control is in accepting state, the **input word** is said to be **accepted** by the TM. The word on the tape at this moment constitutes the **output word** of the computation.

# 1.3 - Computation models

- The **language recognized** by a TM (Q, ∑, $q_0$, F, d) is the set of words from ∑,* that are recognized by this TM.
- A TM is **deterministic** (DTM) if its transition relation does not include two distinct transitions with the same source state and the same input symbol.
- A language is **recursively enumerable** if it is recognized by a TM.
- A language is **recursive** or decidable if there exists a TM that stops whatever input is and that recognizes this language.
- For a TM recognizing a recursive language, the **function** that associates any input of a DTM to the word read on the tape when the DMT stops in an accepting state is said to be a **recursive (computable) function**.

# 1.3 - Computation models : exercises

1. A TM is defined by the following transition table :

| τ | 0 | 1 | X | Y | ⊥ |
|---|---|---|---|---|---|
| $s_0$ | $(s_1, X, R)$ | | | | |
| $s_1$ | $(s_1, 0, R)$ | $(s_2, Y, L)$ | | $(s_1, Y, R)$ | |
| $s_2$ | $(s_4, 0, L)$ | | $(s_3, X, R)$ | $(s_2, Y, L)$ | |
| $s_3$ | | | | $(s_3, Y, R)$ | $(s_5, Y, R)$ |
| $s_4$ | $(s_4, 0, L)$ | | $(s_0, X, R)$ | | |

The initial state is $s_0$ and there is one accepting state $s_5$.

a) What are the computations and possibly the output words produced by the MT with the following input words : 01 0101 0011 00011

b) What are the words built with 0 and 1 recognized by the MT ?

# 1.3 - Computation models : exercises

2. Build a TM that recognizes the following languages :
   a) The set of even binary numbers.
   b) The set of sentences in French that contain the word “la” (we assume that there is no hyphens and no dots in such sentences except one full stop).
   c) The set of palindromes built with the letters “*a*” and “*b*”.
   d) The set of strings composed of an equal number of symbols “*a*”, “*b*” and “*c*”.
   e) The set of strings in the form *w w* such that *w* is any string composed of symbols “*a*” or “*b*”.

3. Build a TM that realizes the following recursive functions :
   a) A function that doubles any binary number.
   b) A function that removes all zeros on the right of any binary number.
   c) A function that removes all “*b*” from a string composed of symbols “*a*” or “*b*” and that removes the spaces between the remaining “*a*”.
   d) A function that reverses a string composed of symbols “*a*” or “*b*”