# Introduction to computability

*Pierre Wolper*

Email: Pierre.Wolper@ulg.ac.be

URL: http: //www.montefiore.ulg.ac.be/~pw/
http: //www.montefiore.ulg.ac.be/
~pw/cours/calc.html

*References*

Pierre Wolper, *Introduction à la calculabilité - 3ième édition*, Dunod, 2006.

Michael Sipser, Introduction to the Theory of Computation, Second Edition, Course Technology, 2005

# Chapter 1

# Introduction

## 1.1 Motivation

- To understand the limits of computer science.

- To distinguish problems that are solvable by algorithms from those that are not.

- To obtain results that are independent of the technology used to build computers.

# 1.2 Problems and Languages

- Which problems can be solved by a program executed by a computer?

  We need to be more precise about:

  – the concept of problem,

  – the concept of program executed by a computer.

# The concept of problem

Problem: *generic* question.

**Examples :**

- to sort an array of numbers;
- to decide if a program written in C stops for all possible input values; (halting problem) ;
- to decide if an equation with integer coefficients has integer solutions (Hilbert's 10th problem).

# The concept of program

Effective procedure: program that can be executed by a computer.

**Examples :**

- Effective procedure : program written in JAVA ;

- Not an effective procedure: "to solve the halting problem, one must just check that the program has no infinite loops or recursive call sequences."

# The halting problem

```
recursive function threen (n: integer):integer;
begin
if (n = 1) then 1
           else if even(n) then threen(n ÷ 2)
                           else threen(3 × n + 1);
end;
```

## 1.3 Formalizing problems

How could one represent problem instances?

# Alphabets and words

Alphabet : finite set of symbols.

**Examples**

- $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$
- $\{\alpha, \beta, \gamma\}$
- $\{1, 2, 3\}$
- $\{\clubsuit, \diamondsuit, \heartsuit\}$

Word on an alphabet : *finite* sequence of elements of the alphabet.

**Examples**

- *a*, *abs*, *zt*, *bbbssnbnzzyyyyyddtrra*, *grosseguindaille* are words on the alphabet $\{a, \ldots, z\}$.

- 4♣3♢5♡2♠, 12765, ♣♡ are words on the alphabet $\{0, \ldots, 8, ♣, ♢, ♡, ♠\}$.

Empty word: represented by $e$, $\varepsilon$, or $\lambda$.

Length of a word $w$ : $|w|$

$w = aaabbaaaabb$
$w(1) = a$, $w(2) = a$,. . . , $w(11) = b$

# Representing problems

## Encoding a problem

Let us consider a binary problem whose instances are encoded by words defined over an alphabet $\Sigma$. The set of all words defined on $\Sigma$ can be partitioned in 3 subsets:

- *positive* instances: the answer is *yes* ;
- *negative* instances: the answer is *no*;
- words that do not represent an instance of the problem.

Alternatively:

- the words encoding instances of the problem for which the answer is *yes*, *the positive instances* ;

- the words that do not encode and instance of the problem, or that encode an instance for which the answer is *no* , *the negative instances*.

# Languages

Language: set of words defined over the same alphabet.

**Examples**

- $\{aab, aaaa, \varepsilon, a, b, abababababbbbbbbbbbb\}$, $\{\varepsilon, aaaaaaa, a, bbbbbb\}$ and $\emptyset$ (the empty set) are languages over the alphabet $\{a, b\}$.
- for the alphabet $\{0, 1\}$,

  $\{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100,$
  $101, 110, 111, \ldots\}$ is the language containing all words.
- language $\emptyset \neq$ language $\{\varepsilon\}$.
- the set of words encoding C programs that always stop.

# 1.4 Describing languages

## Operations on languages

Let $L_1$ and $L_2$ be languages.

- $L_1 \cup L_2 = \{w | w \in L_1 \text{ or } w \in L_2\}$ ;
- $L_1 \cdot L_2 = \{w | w = xy, x \in L_1 \text{ and } y \in L_2\}$ ;
- $L_1^* = \{w | \exists k \geq 0 \text{ and } w_1, \ldots, w_k \in L_1 \text{ such that } w = w_1 w_2 \ldots w_k\}$ ;
- $\overline{L_1} = \{w | w \notin L_1\}$.

# Regular Languages

The set $\mathcal{R}$ of regular languages over an alphabet $\Sigma$ is the smallest set of languages such that:

1. $\emptyset \in \mathcal{R}$and $\{\varepsilon\} \in \mathcal{R}$,

2. $\{a\} \in \mathcal{R}$ for all $a \in \Sigma$, and

3. if $A, B \in \mathcal{R}$, then $A \cup B$, $A \cdot B$ and $A^* \in \mathcal{R}$.

# Regular expressions

A notation for representing regular languages.

1. $\emptyset$, $\varepsilon$ and the elements of $\Sigma$ are regular expressions;

2. If $\alpha$ and $\beta$ are regular expressions, then $(\alpha\beta)$, $(\alpha \cup \beta)$, $(\alpha)*$ are regular expressions.

The set of regular expressions is a language over the alphabet $\Sigma' = \Sigma \cup \{), (, \emptyset, \cup, *, \varepsilon\}$.

# The language represented by a regular expression

1. $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$,

2. $L(a) = \{a\}$ for each $a \in \Sigma$,

3. $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$,

4. $L((\alpha\beta)) = L(\alpha) \cdot L(\beta)$,

5. $L((\alpha)*) = L(\alpha)^*$.

## Theorem

A language is regular

if and only if

it can be represented by a regular expression.

## Regulars languages : examples

- The set of all words over $\Sigma = \{a_1, \ldots, a_n\}$ is represented by $(a_1 \cup \ldots \cup a_n)^*$ (or $\Sigma^*$).

- The set of all nonempty words over $\Sigma = \{a_1, \ldots, a_n\}$ is represented by $(a_1 \cup \ldots \cup a_n)(a_1 \cup \ldots \cup a_n)^*$ (or $\Sigma\Sigma^*$, or $\Sigma^+$).

- the expression $(a \cup b)^* a (a \cup b)^*$ represents the language containing all words over the alphabet $\{a, b\}$ that contain at least one "a".

# Regulars languages : more examples

$$(a^*b)^* \cup (b^*a)^* = (a \cup b)^*$$

**Proof**

- $(a^*b)^* \cup (b^*a)^* \subset (a \cup b)^*$ since $(a \cup b)^*$ represents the set of all words built from the characters "a" and "b".

- Let us consider an arbitrary word

$$w = w_1 w_2 \dots w_n \in (a \cup b)^*.$$

One can distinguish 4 cases ...

1. $w = a^n$ and thus $w \subset (\varepsilon a)^* \subset (b^*a)^*$ ;
2. $w = b^n$ and thus $w \subset (\varepsilon b)^* \subset (a^*b)^*$ ;
3. $w$ contains both $a$'s and $b$'s and ends with a $b$

$$w = \underbrace{a \ldots ab}_{a^*b} \underbrace{\ldots b}_{(a^*b)^*} \underbrace{a \ldots ab}_{a^*b} \underbrace{\ldots b}_{(a^*b)^*}$$

$\Rightarrow w \in (a^*b)^* \cup (b^*a)^*$ ;

4. $w$ contains both $a$'s and $b$'s and ends with an $a \Rightarrow$ similar decomposition.

# 1.5 Languages that are not regular

**Fact**

There are not enough regular expressions to represent all languages!

**Definition**

Cardinality of a set. . .

**Example**

The sets $\{0, 1, 2, 3\}$, $\{a, b, c, d\}$, $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$ all have the same size. There exists a one-one correspondence (bijection) between them, for example $\{(0, \clubsuit), (1, \diamondsuit), (2, \heartsuit), (3, \spadesuit)\}$.

# Denumerable (countably infinite) sets

**Definition**

An infinite set is *denumerable* if there exists a bijection between this set and the set natural numbers.

**Remark**

Finite sets are all countable in the usual sense, but in mathematics countable is sometimes used to mean precisely countably infinite.

# Denumerable sets: examples

1. The set of even numbers is denumerable:

$$\{(0,0),(2,1),(4,2),(6,3),\ldots\}.$$

2. The set of words over the alphabet $\{a,b\}$ is denumerable :
   $\{(\varepsilon,0),(a,1),(b,2),(aa,3),(ab,4),(ba,5),$
   $(bb,6),(aaa,7)\ldots\}$.

3. The set of rational numbers is denumerable:
   $\{(0/1,0),(1/1,1),(1/2,2),(2/1,3),(1/3,4),$
   $(3/1,5),\ldots\}$.

4. The set of regular expressions is denumerable.

# The diagonal argument

**Theorem**

The set of subsets of a denumerable set is not denumerable.

**Proof**

| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $\cdots$ |
|---|---|---|---|---|---|---|
| $s_0$ | $\times$ | $\times$ | | $\times$ | | |
| $s_1$ | $\times$ | $\square$ | | $\times$ | | |
| $s_2$ | | $\times$ | $\times$ | | $\times$ | |
| $s_3$ | $\times$ | | $\times$ | $\square$ | | |
| $s_4$ | | $\times$ | | $\times$ | $\square$ | |
| $\vdots$ | | | | | | |

$$D = \{a_i \mid a_i \notin s_i\}$$

# Conclusion

- The set of languages is not denumerable.
- The set of regular languages is denumerable.
- Thus there are (many) more languages than regular languages

## 1.6 To follow . . .

- The notion of effective procedure (automata).
- Problems that cannot be solved by algorithms.
- Problems that cannot be solved efficiently.