CHAPTER 1

# Induction - Recursion

## 1.1. Introduction

Définition. A *sequence* S is a list of objects, enumerated in some order:

$$S(1), S(2), S(3), \dots, S(k), \dots$$

A sequence can be defined by giving the value of $S(k)$, for all $k$.

Exemple. $S(k) := 2^k$ defines the sequence: $2, 4, 8, 16, 32, \dots$

Imagine instead that I give you the following recipe:

- (a) $S(1) := 1$.
- (b) If $k > 1$, then $S(k) := S(k-1) + k$.

Can you compute $S(2)$? $S(3)$ ? $S(4)$ ?
Could you compute any $S(k)$, where $k > 0$.

Proposition. *The sequence $S(k)$ is fully and uniquely defined by (a) and (b).*

Définition. We say that S is defined *recursively* by (a) and (b).

- (a) is the *base case*
- (b) is the *induction step*

Exemple. The stair and the baby.

This is the idea of recursion (or induction), which is very useful in maths and computer science.
It allows to reduce an infinite process to a finite (small) number of steps.

Exemple. Define $S(k)$ by $S(1) := 4$.
Problem: how to compute S(2) ?

Exemple. Define S(k) by $S(k) := 2S(k-1)$
Problem: both the following sequences satisfies this definition!

- $1, 2, 4, 8, 16, \dots$
- $3, 6, 12, 24, 48, \dots$

Exemple. Proof that any integer is even.

Pitfall: Both base case and induction step are necessary !
You need to know how to start, and you need to know how to continue.

## 1.2. Proofs by induction

PROBLEM 1.2.1. Let S be the sequence defined by:

- $S(1) = 1$
- $S(k) := S(k-1) + 2^{k-1}$

Goal: compute $S(60)$

$$\begin{array}{rclcl} S(1) & = & 1 & = & \\ S(2) & = & 1+2 & = & \\ S(3) & = & & = & \\ S(4) & = & & = & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S(k) & = & 1+\cdots+2^k & = & \end{array}$$

CONJECTURE. $S(60) =$

The formula $S(k) = 2^k - 1$ is called a *closed form formula* for $S(k)$.
It allows to compute $S(k)$ easily, without computing all the previous values $S(1), S(2), \dots S(k-1)$.
So, how to prove this conjecture ?

***Introduce some notation:***
Let $P(k)$ be the predicate: $S(k) = 2^k - 1$.
For any positive integer $k$, $P(k)$ is either true or false.
***Look at examples:***

EXERCICE 1. Try the following:

- Prove $P(1), P(2), P(3)$
- Assume that $P(27)$ is true. Can you prove that $P(28)$ is true ?

Now, can you prove $P(60)$?

### 1.2.1. First principle of mathematical induction:

THÉORÈME. *First principle of mathematical induction*
*Let $P(k)$ be a predicate. If:*
*(a) $P(1)$ is true*
*(b) For any $k > 1$, $P(k-1)$ true implies $P(k)$ true*
*Then: $P(k)$ is true for all $k \geq 1$.*

DÉFINITION. P is the *inductive hypothesis.*
(a) is the *base case.*
(b) is the *induction step.*

EXEMPLE. Consider the sequence $S$ defined as above by:
(a) $S(1) := 1$.
(b) $S(k) := S(k) + 2^{k-1}$.
Let's prove that for all $k$, $S(k) = 2^k - 1$.

PROOF. Let $P(k)$ be the predicate $S(k) = 2^k - 1$.

(1) Base case:
$S(1) = 1 = 2^1 - 1$. So, $P(1)$ is true.
(2) Induction step:
Let $k > 1$ be an integer, and assume $P(k-1)$ is true: $S(k-1) = 2^{k-1} - 1$.
Then, $S(k) = S(k-1) + 2^{k-1} = 2^{k-1} - 1 + 2^{k-1} = 2^k - 1$.
So, P(k) is also true.
Conclusion: by the first principle of mathematical induction,
$P(k)$ is true for all $k \geq 1$, i.e. $S(k) = 2^{k-1}$.

$\square$

EXERCICE 2. Let $S(k) := 1 + 3 + 5 + \cdots + (2k-1)$. Find a closed form formula for $S(k)$.

EXERCICE 3. Prove that $k! > 2^k$ for any $k \geq 4$.

EXERCICE 4. Prove that for any integer $k$, $2^{2k} - 1$ is divisible by 3.

EXERCICE 5. Prove that $a + ar + ar^2 + \cdots + ar^n = \frac{a - ar^n}{1-r}$.

EXERCICE 6. Find a closed form formula for $S(k) := 1^4 + 2^4 + \cdots k^4$.
Hint: the difficulty is to find a suggestion. What tool could you use for this?

EXERCICE 7. The chess board problem.

### 1.2.2. Other forms of induction:

EXEMPLE. Define the sequence $F(n)$ by:
$F(1) := 1$
$F(2) := 1$
$F(k) = F(k-1) + F(k-2)$, for all $k > 2$.

EXERCICE 8. Can you compute $F(3), F(4), F(5)$?

This sequence is the famous and useful *Fibonacci* sequence.

PROBLEM 1.2.2. To compute $F(k)$, you not only need $F(k-1)$, but also $F(k-2)$. So that does not fit into the previous induction scheme.

That's fine, because to compute $F(k)$, you only need to know the values of $F(r)$ for $r < k$.

THÉORÈME. *Second principle of Mathematical induction:*
*Let $P(k)$ be a predicate. If*
*(a) $P(1)$ is true,*
*(b) For any $k > 1$, [ $P(r)$ true for any $r$, $1 \leq r < k$ ] implies $P(k)$ true,*
*then: $P(k)$ is true for all $k \geq 1$.*

EXEMPLE. Any integer is a product of prime integers

EXEMPLE. The coin problem

We did not prove that the first (or the second) principle of induction were valid.
Let's just give an idea of the proof:

THÉORÈME. *The three following principles are in fact equivalent:*

(1) *the first principle of induction*
(2) *(b) the second principle of induction*
(3) *(c) the principle of well ordering:*
*every non empty collection of positive integers has a smallest member.*

PROBLEM 1.2.3. Is the principle of well ordering valid for $\mathbb{Z}$ ? $\mathbb{R}$ ? $\mathbb{C}$ ?

### 1.3. Other uses of induction

Induction is a much more general problem solving principle.
If you have a family of problems such that:

- There is some measure of the size of a problem
- You can solve the smallest problems
- You can breakup each bigger problems in one (or several) strictly smaller cases, and build from it a solution for the big problem.

Then, you can solve by induction any problem in your family.

EXEMPLE. Problem: computation of $F(k)$.
Measure of the problem: $k$.
Value of $F(1)$ is known.
$F(k)$ can be computed from the values of $F(k-1)$ and $F(k-2)$

EXEMPLE. Inductive definition of well formed formulas from formal logic
<basic proposition>: $A \mid B \mid C \mid \dots$
<binary connective>: $\wedge \mid \vee \mid \rightarrow \mid \leftarrow \mid \leftrightarrow$
<wff>: <basic proposition> | <wff>′ | (<wff>) | <wff> <binary connective> <wff>

This kind of inductive description is called *BNF* (Backus Naur form).

EXEMPLE. Recursive proof of a property of wff

THÉORÈME. *If a wff contains $n$ propositions (counted with repetition: in $A \vee A$, there are two propositions), then it contains $n-1$ binary connectives.*

EXEMPLE. Recursive algorithm for the computation of the value of a wff:

```
bool function Evaluation(P, C)
// Precondition: P is a wff, C is the context, i.e.
// the truth values of all the basic propositions.
// Postcondition: returns the truth value of P in the // context C
Begin
  If P is a proposition (say A) Then
    Return the truth value of A;
  ElseIf P is of the form (Q) Then
    Return Evaluation(Q, C);
  ElseIf P is of the form Q' Then
    Return not Evaluation(Q,C);
  ElseIf P is of the form Q1 ∧ Q2 Then
    Return Evaluation(Q1, C)and Evaluation(Q2, C);
```

```
  ... // Other binary operators
  Else
    Error P is not a wff;
End;
```

Remarque. There are programming languages that are particularly well adapted to write this kind of algorithms. Actually, the real program would almost look like the pseudo-code. See for example CAML `ftp://ftp.inria.fr/lang/caml-light/`.

Exemple. Bijection between words of parenthesis and forests

## 1.4. Conclusion

Induction is a fundamental technique for solving problems, in particular in computer science.

It's the mathematical formalization of the divide and conquer approach.