# Codage des nombres

## 1 Codage des entiers naturels

### 1.1 Codage en base *b*

Soit $b \in \mathbb{N}\backslash\{0,1\}$. On considère l'alphabet $\Sigma = [0..b-1]$.
· Les éléments de $\Sigma$ seront appelés les chiffres.
· Les mots sur $\Sigma$, c'est-à-dire les suites finies d'éléments de $\Sigma$, sont appelés les nombres. L'ensemble de ces nombres se note $\Sigma^*$, et est défini par :

$$\Sigma^* = \bigcup_{i\in\mathbb{N}} \Sigma^i$$

où pour tout $i \in \mathbb{N}$, $\Sigma^i$ est l'ensemble des suites de longueur $i$ dans $\Sigma$.

Le seul mot de longueur nulle est le mot vide, noté $\varepsilon$ : autrement dit, $\Sigma^0 = \{\varepsilon\}$.

#### 1.1.1 Définition

**Définition (*écriture en base* $b$) :**

Pour $a = a_{l-1}a_{l-2}...a_1a_0 \in \Sigma^l$ un mot sur $\Sigma$ de longueur $l$.

On dit que $a$ est une écriture en base $b$ à $l$ chiffres de l'entier $n = \sum_{i=0}^{l-1} a_i b^i$.

On peut alors définir la fonction suivante, qui à un mot de $\Sigma^*$ associe l'entier qu'il représente :

$$\mathrm{val}_b = \begin{pmatrix} \Sigma^* & \to & \mathbb{N} \\ (a_{l-1}a_{l-2}...a_0) & \mapsto & \sum_{i=0}^{l-1} a_i b^i \end{pmatrix}$$

**Remarque :** On a en particulier $\mathrm{val}_b(\varepsilon) = 0$ pour n'importe quelle base $b \in \mathbb{N}\backslash\{0,1\}$.

**Remarque :** On trouvera parfois aussi la notation $\overline{a_{l-1}...a_0}^b$ pour désigner $\mathrm{val}_b(a_{l-1}...a_0)$.

**Exemples :** $\mathrm{val}_{10}(123) = 100 + 20 + 3 = 123$ et $\mathrm{val}_2(100) = \mathrm{val}_2(000100) = 2^2 = 4$.

#### 1.1.2 Taille du codage

**Lemme :** Soit $l \in \mathbb{N}$. $\forall\, a \in \Sigma^l$, $\mathrm{val}_b(a) \in [0..b^l[$.
Autrement dit, le plus grand entier que l'on peut écrire en base $b$ avec $l$ chiffres est $b^l - 1$.

**Preuve :**
Soit $(a_{l-1}, a_{l-2},..., a_1, a_0) \in \Sigma^l$. Puisque $\forall\, i \in [0..l-1]$, $a_i \in [0..b-1]$, on a :

$$0 \leq \mathrm{val}_b(a_{l-1}...a_0) = \sum_{i=0}^{l-1} a_i b^i \leq \sum_{i=0}^{l-1} (b-1)b^i = \sum_{i=1}^{l} b^i - \sum_{i=0}^{l-1} b^i = b^l - 1$$

**Propriété :** Soit $n \in \mathbb{N}$. Il faut $\lceil \log_b(n+1) \rceil$ chiffres (au moins) pour écrire $n$ en base $b$.

**Preuve :**
Soit $n \in \mathbb{N}$, on note $l = \lceil \log_b(n+1) \rceil$.

Supposons par l'absurde qu'il existe $a_{l'-1}...a_0 \in \Sigma^{l'}$ représentant $n$ en base $b$ avec $l' < l$ chiffres. Par définition de la partie entière supérieure d'un réel comme étant son plus petit majorant entier, $l'$ n'est pas un majorant de $\log_b(n+1)$, *i.e.* $l' < \log_b(n+1)$.

Alors, d'après le lemme, on a :

$$n = \mathrm{val}_b(a_{l'-1}...a_0) \leq b^{l'} - 1 < b^{\log_b(n+1)} - 1 = (n+1) - 1$$

c'est-à-dire $n < n$, d'où une contradiction.

**Remarque :** $\forall\,(n,k) \in \mathbb{N}^2$, $n \in [b^{k-1}..b^k[ \iff n+1 \in ]b^{k-1}..b^k]$
$\iff \log_b(n+1) \in ]k-1..k]$
$\iff \lceil \log_b(n+1) \rceil$.

### 1.1.3 Existence

**Propriété (*existence de l'écriture en base* $b$) :**

> Pour tout $n \in \mathbb{N}$, il existe un nombre $a_{l-1}a_{l-2}...a_0 \in \Sigma^l$ tel que $\mathrm{val}_b(a_{l-1}...a_0) = n$.
> Plus précisément, tout entier $n \in \mathbb{N}$ admet une écriture en base $b$ à $\lceil \log_b(n+1) \rceil$ chiffres.

**Preuve :**
Montrons par récurrence sur $l \in \mathbb{N}$ la propriété

$$\mathcal{H}_l : \text{“}\forall\, n \in [0..b^l - 1], n \text{ admet une écriture en base } b \text{ à } l \text{ chiffres”}$$

· On a $[0..b^0 - 1] = [0..0] = \{0\}$ et 0 admet bien une écriture en base $b$ à $\lceil \log_2(0+1) \rceil = 0$ chiffres : il s'agit du mot vide, $\varepsilon$. Ainsi, $\mathcal{H}_0$ est vraie.

· Supposons $\mathcal{H}_l$ pour un $l \in \mathbb{N}$ quelconque et montrons $\mathcal{H}_{l+1}$. Soit donc $n \in [0..b^{l+1} - 1]$.
Par définition de la division euclidienne, il existe $(q,r) \in \mathbb{N} \times [0..b^l - 1]$ tel que $n = b^l q + r$.
D'après $\mathcal{H}_l$, $r$ admet une écriture en base $b$ à $l$ chiffres ; notons-la $(a_i)_{i \in [0..l-1]} \in \Sigma^l$.
Alors, $r = \sum_{i=0}^{l-1} a_i b^i$ et donc $n = qb^l + \sum_{i=0}^{l-1} a_i b^i$.
Puisque $n < b^{l+1}$, nécessairement $q < b$ (car sinon on aurait $n = b^l q + r \geq b^l q \geq b^{l+1}$).

En posant ainsi $a_l = q$, on a $(a_i)_{i \in [0..l]} \in \Sigma^{l+1}$ et $n = \sum_{i=0}^{l} a_i b^i$ donc $a_l a_{l-1}...a_0$ est bien une écriture de $n$ en base $b$ à $l+1$ chiffres, d'où $\mathcal{H}_{l+1}$.

### 1.1.4 Quasi-unicité

**Propriété (*quasi-unicité de l'écriture en base* $b$) :**

> Soit $n \in \mathbb{N}$. Si $a = a_{l-1}a_{l-2}...a_0$ est une écriture de $n$ en base $b$, alors, *i.e.* $\mathrm{val}_b(a) = n$, alors pour tout $k \in [0..l-1]$, $a_k$ est le reste modulo $b$ du quotient de $n$ par $b^k$.

**Preuve :**
Soit $n \in \mathbb{N}$, $a_{l-1}...a_0$ une écriture de $n$ en base $b$ et $k \in [0..l-1]$. On a :

$$n = \mathrm{val}_b(a_{l-1}...a_0) = \sum_{i=0}^{l-1} a_i b^i = \sum_{i=0}^{k-1} a_i b^i + \sum_{i=k}^{l-1} a_i b^i = \underbrace{\sum_{i=0}^{k-1} a_i b^i}_{:=r_k} + b^k \underbrace{\sum_{i=k}^{l-1} a_i b^{i-k}}_{:=q_k} = q_k b^k + r_k \ (*)$$

On a clairement $r_k \in \mathbb{N}$ et comme $\forall\, i \in [0..l-1]$, $a_i \in [0..b-1]$, on a aussi

$$r_k = \sum_{i=0}^{l-1} a_i b^i \leq \sum_{i=0}^{l-1} (b-1) b^i = \sum_{i=0}^{k-1} b^{i+1} - \sum_{i=0}^{k-1} b^i = b^k - 1 < b^k$$

Ensuite, $\forall\, i \in [k..l-1]$, on a $i - k \geq 0$ donc $a_i b^{i-k} \in \mathbb{N}$ : ainsi, $q_k \in \mathbb{N}$ par somme.

Alors, d'après l'égalité $(*)$ : • $q_k$ est le quotient dans la division euclidienne de $n$ par $b$.
• $r_k$ est le reste de $n$ modulo $b$.

Montrons à présent que $a_k$ est le reste de $q_k$ modulo $b$. On a :

$$q_k = \sum_{i=k}^{l-1} a_i b^{k-i} = a_k + b^{k-k} + \sum_{i=k+1}^{l-1} a_i b^{k-1} = a_k + b \sum_{i=k+1}^{l-1} a_i b^{i-k-1}$$

D'une part, on sait que $0 \leq a_k < b$ car $a_k \in \Sigma$ et d'autre part, $\forall\, i \in [k+1..l-1]$, $i-1-k \geq 0$ donc $a_i b^{i-1-k} \in \mathbb{N}$. On déduit donc de l'égalité précédente que $a_k$ est bien le reste de $q_k$ modulo $b$.

**Exemple :** En base 10, $n = 1023$ a peut s'écrire $a_3 a_2 a_1 a_0$ avec :

$$a_0 = n \,\%\, 10 = 1023 \,\%\, 10 = 3$$
$$a_1 = (\lfloor n/10 \rfloor) \,\%\, 10 = 102 \,\%\, 10 = 2$$
$$a_2 = \left(\left\lfloor (\lfloor n/10 \rfloor)/10 \right\rfloor\right) \%\, 10 = 10 \,\%\, 10 = 0$$
$$a_3 = \left(\left\lfloor \left(\left\lfloor (\lfloor n/10 \rfloor)/10 \right\rfloor\right)/10 \right\rfloor\right) \%\, 10 = 1 \,\%\, 10 = 1.$$

**Remarque :** L'exemple précédent fournit un algorithme simple pour déterminer une écriture en base $b$ de n'importe quel entier naturel $n$ à partir de son écriture en base 10. On peut montrer que sa complexité est en $\Theta(\log(n))$ (*cf.* chapitre 3 – "complexité pire cas").

**Corollaire (*conséquences de la quasi-unicité*) :**

Soit $n \in \mathbb{N}$ et $l, l' \in \mathbb{N}$ tels que $l \leq l'$.
Si $a = a_{l-1}...a_0$ et $a' = a'_{l-1}...a'_0$ sont deux écritures de $n$ en base $b$, alors :

• $\forall\, k \in [0..l-1], a_k = a'_k$
• $\forall\, k \in [l..l'-1], a_k = 0$

En particulier, il y a donc unicité de l'écriture en base $b$ à longueur fixée.

### 1.1.5 Conclusion

La définition suivante nous permet à présent d'expliciter une écriture en base $b$ de n'importe quel entier naturel.

**Définition (*écriture en base $b$ à $l$ chiffres*) :**

Pour $l \in \mathbb{N}$, on définit l'écriture en base $b$ à $l$ chiffres :

$$\mathrm{ecr}_b^l = \begin{pmatrix} [0..b^l - 1] & \to & \Sigma^l \\ n & \mapsto & a_{l-1}...a_1 a_0 \end{pmatrix}$$

où $\forall\, k \in [0..l-1]$, $a_k$ est le reste modulo $b$ du quotient de $n$ par $b^k$.

On peut donc extraire de toute écriture en base $b$ l'entier représenté, et réciproquement, tout entier naturel admet une écriture en base $b$ que l'on sait expliciter.

## 1.2 Addition en base 2

*cf.* annexe "figures pour le chapitre 2".

## 1.3 Application de l'encodage en base $b$

*Dans cette section, on exploite l'encodage en base b pour démontrer que tous les problèmes ne peuvent pas être résolus par un algorithme (cf. notion de calculabilité, qui sort du cadre de ce cours)*

Pour $u \in \mathbb{N}^{\mathbb{N}}$, on considère le problème : $\boldsymbol{\mathcal{P}_u}$ $\left\| \begin{array}{l} \textbf{entrée :} \ n \in \mathbb{N} \\ \textbf{sortie :} \ u_n \end{array} \right.$

Un algorithme $\mathscr{A}$ pouvant être identifié à une suite finie de caractères, donc par extension à une suite finie d'entiers entre 1 et 255, on peut le représenter par un entier écrit en base 256.

On note donc $\varphi$ la fonction qui à un algorithme $\mathscr{A}$ associe la valeur de l'entier écrit en base 256 obtenu en remplaçant les caractères par l'entier entre 1 et 255 correspondant (étant donnée une telle correspondance, la table ASCII éténdue par exemple). Autrement dit :

$$\varphi : \begin{array}{c} \text{l'ensemble } \mathcal{A} \text{ des textes des} \\ \text{algorithmes qui prennent} \\ \text{en entrée un entier} \end{array} \to \mathbb{N}$$

$\varphi$ est facilement injective (deux algorithmes différents auront des écritures associées différentes en base 256, donc deux images différentes par quasi-unicité de l'écriture en base 256) mais elle n'est pas surjective (*cf.* chaînes de caractères n'ayant aucun sens).

En considérant $\varphi|_{\mathcal{A}}^{\mathrm{Im}(\varphi)}$, on obtient une application surjective, donc bijective de $\mathcal{A}$ sur $\mathrm{Im}(\varphi)$.

On note alors, pour $\mathscr{A} \in \mathcal{A}$ le texte d'un algorithme qui prend en entrée un entier et rend en sortie un entier, $\mathrm{eval}(\mathscr{A}, n)$ la valeur obtenue en lançant cet algorithme sur l'entrée $n$.

**Remarque :** Si $\mathscr{A}$ résout $\mathcal{P}_u$ pour $u \in \mathbb{N}^{\mathbb{N}}$, alors $\forall\, n \in \mathbb{N}$, $\mathrm{eval}(\mathscr{A}, n) = u_n$

On définit maintenant $u \in \mathbb{N}^{\mathbb{N}}$ par :

$$\forall\, n \in \mathbb{N},\ u_n = \begin{cases} \mathrm{eval}(\varphi^{-1}(n), n) + 1 \text{ si } n \in \mathrm{im}(\varphi) \\ 0 \text{ sinon} \end{cases}$$

On suppose que $\mathscr{A}_u$ est un algorithme qui résout $\mathcal{P}_u$. Alors,

$$\begin{aligned} \mathrm{eval}(\mathscr{A}_u, \varphi(\mathscr{A}_u)) &= u_{\varphi(\mathscr{A}_u)} \text{ car } \mathscr{A}_u \text{ résout } \mathcal{P}_u \\ &= \mathrm{eval}(\varphi^{-1}(\varphi(\mathscr{A}_u)), \varphi(\mathscr{A}_u)) + 1 \text{ puisque } \varphi(\mathscr{A}_u) \in \mathrm{im}(\varphi) \\ &= \mathrm{eval}(\mathscr{A}_u, \varphi(\mathscr{A}_u) + 1 \end{aligned}$$

ce qui est absurde et prouve donc qu'il n'existe pas d'algorithme résolvant $\mathcal{P}_u$.

**Remarque :** Cette preuve utilise ce que l'on appelle un argument diagonal.

# 2 Encodage des entiers relatifs

*On s'intéresse dans cette partie à l'encodage des entiers relatifs tels qu'il est fait sur les ordinateurs. On s'appuiera ainsi sur le codage des entiers naturels en binaire, c'est-à-dire en base 2 : on fixe donc* $\Sigma = \{0, 1\}$.

## 2.1 Encodage

Afin d'introduire la notion de signe dans l'encodage, le signe de l'entier encodé par un nombre sera donné par l'un de ses chiffres : il vaudra 0 pour les positifs et 1 pour les négatifs. Ce chiffre a donc une signification particulière, différente de celle des autres 0 ou 1.

De plus, pour des raisons pratiques qui apparaîtront plus bas, ce chiffre de signe sera celui le plus à gauche du nombre, soit à l'opposé du chiffre des unités. Il nous faut ainsi travailler à longueur fixée pour pouvoir identifier ce chiffre au statut particulier, ce qui limite nécessairement l'ensemble des entiers que l'on pourra encoder.

Soit $l \in \mathbb{N}^*$. On note $I^l = [-2^{l-1}..2^{l-1}]$, on remarque alors que $\mathrm{Card}(I^l) = 2^l$.

**Proposition (*encodage et décodage à longueur fixée*) :**

On définit les deux fonctions suivantes :

$$\varphi^l = \begin{pmatrix} I^l \to \{0,1\}^l \\ z \mapsto \begin{cases} 0\ \mathrm{ecr}_2^{l-1}(z) \text{ si } z \geq 0 \\ 1\ \mathrm{ecr}_2^{l-1}(z + 2^{l-1}) \text{ sinon} \end{cases} \end{pmatrix}$$

$$\psi^l = \begin{pmatrix} \{0,1\}^l \to I^l \\ a_{l-1}...a_1a_0 \mapsto -2^{l-1}a_{l-1} + \mathrm{val}_2(a_{l-2}...a_0) \end{pmatrix}$$

Alors, $\varphi_l$ et $\psi_l$ sont bien définies et sont réciproques (*c-à-d* $\varphi^l \circ \psi^l = \mathrm{Id}_{\Sigma^l}$ et $\psi^l \circ \varphi^l = \mathrm{Id}_{I^l}$).

**Preuve :**
$\varphi^l$ est bien définie : en effet, $\mathrm{ecr}_2^{l-1}$ l'est et est elle-même à valeurs dans $\{0,1\}^{l-1}$, ainsi ajouter un 0 ou un 1 à gauche donne bien un mot de $\{0,1\}^l$.

$\psi^l$ est également bien définie : en effet, $\mathrm{val}_2$ l'est et $\forall\, a \in \{0,1\}^{l-1}$, $\mathrm{val}_2(a) \in [0..2^{l-1}[$, donc ajouter 0 ou $-2^{l-1}$ au résultat donne un bien un entier de $[-2^{l-1}..2^{l-1}[ = I^l$.

Soit ensuite $z \in I^l$.

· Si $z \geq 0$, on a $\psi^l(\varphi^l(z)) = \psi^l(0\ \mathrm{ecr}_2^{l-1}(z)) = -2^{l-1} \times 0 + \mathrm{val}_2(\mathrm{ecr}_2^{l-1}(z))$.
Or par définition de l'écriture en base 2, $\mathrm{val}_2(\mathrm{ecr}_2^{l-1}(z)) = z$ (en fait $\mathrm{val}_2\big|_{\Sigma^l}^{[0..b^l-1]}$ et $\mathrm{ecr}_b^l$ sont toujours bijectives et réciproques l'une de l'autre). D'où $(\psi^l \circ \varphi^l)(z) = z$.

· Si $z < 0$, on a $\psi^l(\varphi^l(z)) = \psi^l(1\ \mathrm{ecr}_2^{l-1}(z + 2^{l-1})) = -2^{l-1} \times 1 + \mathrm{val}_2(\mathrm{ecr}_2^{l-1}(z + 2^{l-1}))$.
Par la même remarque qu'au-dessus, on a donc $(\psi^l \circ \varphi^l)(z) = -2^{l-1} + (z + 2^{l-1}) = z$.

Soit enfin $a = a_{l-1}...a_1a_0 \in \{0,1\}^l$. On note $\widetilde{a}$ son suffixe $a_{l-2}...a_1a_0$, alors $\widetilde{a} \in \{0,1\}^{l-1}$.

· Si $a_{l-1} = 0$, alors $\psi^l(a) = \mathrm{val}_2(\widetilde{a}) \in [0..2^{l-1}[$ donc en particulier $\psi^l(a) \geq 0$.
Ainsi, comme $\mathrm{ecr}_2^{l-1}(\mathrm{val}_2(\widetilde{a})) = \widetilde{a}$, on a $(\varphi^l \circ \psi^l)(a) = 0\ \mathrm{ecr}_2^{l-1}(\mathrm{val}_2(\widetilde{a})) = a_{l-1}\,\widetilde{a} = a$.

· Si $a_{l-1} = 1$, alors $\psi^l(a) = -2^{l-1} + \mathrm{val}_2(\widetilde{a}) \in [-2^{l-1}..0[$ donc $\psi^l(a) < 0$.
Par conséquent, on a de même $(\varphi^l \circ \psi^l)(a) = 1\ \mathrm{ecr}_2^{l-1}(\mathrm{val}_2(\widetilde{a})) = a_{l-1}\,\widetilde{a} = a$.

## 2.2 Opérations

### 2.2.1 Passage à l'opposé

**Définition (*complément à 2*) :**

On appelle complément à 2 d'un nombre écrit sur $\{0,1\}$ (donc en binaire) le nombre obtenu en remplaçant les 0 par des 1 et vice-versa, *i.e.* son image par la fonction :

$$\mathrm{comp}_2 = \begin{pmatrix} \Sigma^* & \to & \Sigma^* \\ a_{k-1}...a_1a_0 & \mapsto & \overline{a}_{k-1}...\overline{a}_1\overline{a}_0 \end{pmatrix}$$

où $\forall\, i \in [0..l-1]$, $\overline{a}_i = 1 - a_i$.

**Propriété (*lien entre complément à 2 et opposé*) :**

Pour tout entier relatif, le complément à 2 de son écriture encode son opposé moins 1 :

$$\forall\, z \in I^l,\ \psi^l(\mathrm{comp}_2(\varphi^l(z))) = -z - 1$$

**Preuve :**

Soit $z \in I^l$. On note :
- $a_{l-1}...a_1a_0 = \varphi^l(z)$
- $\widetilde{a} = a_{l-2}...a_1a_0$
- $\overline{a}_{l-1}...\overline{a}_1\overline{a}_0 = \mathrm{comp}_2(\varphi^l(z))$ (ainsi, $\forall\, i \in [0..l-1]$, $\overline{a}_i = 1 - a_i$)
- $\widehat{a} = \overline{a}_{l-2}...\overline{a}_1\overline{a}_0$

Alors, on peut écrire :
- $z = \psi^l(\varphi^l(z)) = -2^{l-1}a_{l-1} + \mathrm{val}_2(\widetilde{a})$
- $\psi^l(\mathrm{comp}_2(\varphi^l(z))) = \psi^l(\overline{a}_{l-1}\overline{a}_{l-2}...\overline{a}_1\overline{a}_0) = -2^{l-1}\overline{a}_{l-1} + \mathrm{val}_2(\widehat{a})$

Or, $\mathrm{val}_2(\widehat{a}) = \mathrm{val}_2(\overline{a}_{l-2}...\overline{a}_0) = \sum_{i=0}^{l-2} \overline{a}_i 2^i = \sum_{i=0}^{l-2}(1-a_i)2^i = \sum_{i=0}^{l-2} 2^i - \sum_{i=0}^{l-2} a_i 2^i = 2^{l-1} - 1 - \mathrm{val}_2(\widetilde{a})$.

Ainsi :
$$\begin{aligned} \psi^l(\mathrm{comp}_2(\varphi^l(z))) &= -2^{l-1}(1-a_{l-1}) + 2^{l-1} - 1 - \mathrm{val}_2(\widetilde{a}) \\ &= 2^{l-1}(1-1+a_{l-1}) - 1 - \mathrm{val}_2(\widetilde{a}) \\ &= -(2^{l-1}a_{l-1} + \mathrm{val}_2(\widetilde{a})) - 1 \\ &= -z - 1 \end{aligned}$$

**Exemple :** $\varphi^4(7) = 0111$ donc $\mathrm{comp}_2(\varphi^4(7)) = 1000$ qui encode bien $-8 = -7 - 1$.

### 2.2.2 Addition

**Définition (*addition en binaire*) :**

On définit l'addition en base 2 à longueur fixée à l'aide de la fonction suivante :

$$\mathrm{add}_2^l = \begin{pmatrix} \Sigma^l \times \Sigma^l & \to & \Sigma^{l+1} \\ (a,b) & \mapsto & c_l c_{l-1}...c_0 \end{pmatrix}$$

où on a noté successivement :
- $a = a_{l-1}a_{l-2}...a_0$ et $b = b_{l-1}b_{l-2}...b_0$
- $r_0 = 0$ et $\forall\, i \in [1..l]$, $r_i = \lfloor (a_{i-1} + b_{i-1} + r_{i-1})/2 \rfloor = \begin{cases} 1 \text{ si } a_{i-1} + b_{i-1} + r_{i-1} \geq 2 \\ 0 \text{ sinon} \end{cases}$
- $\forall\, i \in [0..l-1]$, $c_i = (a_i + b_i + r_i)\,\%\,2$ et $c_l = r_l$

**Remarque :** Il s'agit seulement d'une écriture formelle de l'algorithme d'addition réalisé par la machine présentée en cours, algorithme qui est lui-même l'équivalent en base 2 de

l'algorithme d'addition des nombres en base 10 que l'on connaît. On voit en particulier que $r_i$ correspond à la retenue à prendre en compte à l'étape $i$.

**Propriété (*addition des entiers naturels*) :**

$\text{add}_2^l$ réalise l'addition sur les écritures binaires des entiers naturels, c'est-à-dire :

$$\forall\,(a,b)\in(\Sigma^l)^2,\ \text{val}_2(\text{add}_2^l(a,b))=\text{val}_2(a)+\text{val}_2(b)$$

**Preuve :**
Montrons par récurrence sur $l\in\mathbb{N}$ la propriété

$$\mathcal{P}_l : \text{“}\forall\,(a,b)\in(\Sigma^l)^2,\ \text{val}_2(\text{add}_2^l(a,b))=\text{val}_2(a)+\text{val}_2(b)\text{”}$$

· On a $\Sigma^0=\{\varepsilon\}$ et $\text{val}_2(\text{add}_2^0(\varepsilon,\varepsilon))=c_0=r_0=0=\text{val}_2(\varepsilon)+\text{val}_2(\varepsilon)$, ce qui montre $\mathcal{P}_0$.

· Soit $l\in\mathbb{N}$, on suppose $\mathcal{P}_l$. Considérons alors $(a,b)\in(\Sigma^{l+1})^2$.
On note $a=a_l...a_0$ et $b=b_l...b_0$, puis $\widetilde{a}=a_{l-1}...a_0$ et $\widetilde{b}=b_{l-1}...b_0$.
On note enfin $c_{l+1}...c_0=\text{add}_2^{l+1}(a,b)$.

On remarque en utilisant la définition de $\text{add}_2^l$ et $\text{add}_2^{l+1}$ que les chiffres les plus à droite de $\text{add}_2^l(\widetilde{a},\widetilde{b})$ sont exactement les mêmes que ceux de $\text{add}_2^{l+1}(a,b)$ : plus précisément,

$$\text{add}_2^l(\widetilde{a},\widetilde{b})=\widetilde{c}_l c_{l-1}...c_0 \text{ où } \widetilde{c}_l=r_l$$

En utilisant alors que $\text{val}_2(\widetilde{a})+\text{val}_2(\widetilde{b})=\text{val}_2(\text{add}_2^l(\widetilde{a},\widetilde{b}))$ d'après $\mathcal{P}_l$, puisque $(\widetilde{a},\widetilde{b})\in(\Sigma^l)^2$,

$$\begin{aligned}\text{val}_2(a)+\text{val}_2(b)&=\text{val}_2(a_l\,\widetilde{a})+\text{val}_2(b_l\,\widetilde{b})\\&=2^l a_l+\text{val}_2(\widetilde{a})+2^l b_l+\text{val}_2(\widetilde{b})\\&=2^l(a_l+b_l)+\text{val}_2(\text{add}_2^l(\widetilde{a},\widetilde{b}))\\&=2^l(a_l+b_l)+\text{val}_2(\widetilde{c}_l c_{l-1}...c_0)\\&=2^l(a_l+b_l+\widetilde{c}_l)+\text{val}_2(c_{l-1}...c_0)\end{aligned}$$

D'autre part, $\text{val}_2(c_{l+1}c_l...c_0)=2^{l+1}c_{l+1}+2^l c_l+\text{val}_2(c_{l-1}...c_0)$.

Il reste donc à montrer que $2^l(a_l+b_l+\widetilde{c}_l=2^{l+1}c_{l+1}+2^l c_l)$, soit $a_l+b_l+\widetilde{c}_l=2c_{l+1}+c_l$. Or,
- $c_{l+1}=r_{l+1}=\lfloor(a_l+b_l+r_l)/2\rfloor$
- $c_l=(a_l+b_l+r_l)\,\%\,2$
- $\widetilde{c}_l=r_l$

En reportant la dernière égalité dans les deux premières, on obtient exactement la relation voulue par définition de la division euclidienne (par 2 ici). $\mathcal{P}_{l+1}$ est donc vraie.

**Propriété (*addition des entiers relatifs*) :**

Soit $(y,z)\in(I^l)^2$. Si $y+z\in I^l$, alors en notant $c_l c_{l-1}...c_0=\text{add}_2^l(\varphi^l(y),\varphi^l(z))$, on a :

$$\psi^l(c_{l-1}c_{l-2}...c_0)=y+z$$

Autrement dit, $\text{add}_2^l$ réalise aussi l'addition sur les écritures binaires des entiers relatifs pourvu que la somme soit dans l'intervalle $I^l$.

**Preuve :**
Notons $a=a_{l-1}...a_0=\varphi^l(y)$ et $b=b_{l-1}...b_0=\varphi^l(z)$.

**Remarque :** Dans la propriété précédente, on a tronqué $c_l c_{l-1}...c_1 c_0$ puisque, la “somme” étant dans $I^l$, il ne reste aucune retenue à la fin, autrement dit $c_l=0$.

# 3 Codage des entiers en C

## 3.1 Tailles des types en mémoire

# 4 Encodage des réels

Étant donné que tout intervalle non trivial (*i.e.* non réduit à un singleton) contient une infinité de réels (en fait, de tels intervalles ne sont pas dénombrable), il est impossible d'encoder "tous" ses réels. Il faut donc limiter d'une certaine façon l'ensemble des réels que l'on souhaite représenter, quitte à seulement approximer tout réel qui n'appartiendrait pas à ce nouvel ensemble réduit.

## 4.1 Écriture à précision arbitrairement élevée

*L'objet de cette section est d'étudier les bases de la théorie sur laquelle repose l'encodage des réels sur les machines.*

### 4.1.1 Se ramener dans $[0, 1]$

**Propriété :** Soit $x \in \mathbb{R}^+$. Il existe $k \in \mathbb{N}$ et $y \in [0, 1]$ tels que $x = y \times 2^k$.

**Preuve :**
Il suffit de prendre $y = k = 0$ si $x = 0$, ou bien $k = \lceil \log_2(x) \rceil$ et $y = x/2^k$ sinon.
En effet, dans le deuxième cas on a $k \geq \log_2(x)$ soit $2^k \geq 2^{\log_2(x)} = x$, donc $y \leq 1$. En particulier, quand $0 < x \leq 1$, on a $y = x$ et $k = 0$.

**Propriété (*approximation dyadique*) :**

Soit $x \in [0, 1]$ et $\varepsilon \in \mathbb{R}^{+*}$. Il existe $l \in \mathbb{N}$ et $(a_i)_{i \in [1..l]} \in \{0, 1\}^l$ tels que :

$$\sum_{i=1}^{l} \frac{a_i}{2^i} \in [x - \varepsilon, x + \varepsilon], \text{ c'est-à-dire } \left| \sum_{i=1}^{l} \frac{a_i}{2^i} - x \right| \leq \varepsilon$$

Une telle somme est appelée une fraction dyadique (ou encore un rationnel dyadique).

**Remarque :** La propriété précédente affirme entre autres la convergence du principe de dichotomie lorsque l'on l'applique à $x \in [0, 1]$.

## 4.2 Les flottants en C

Le C, comme la plupart des langages de programmation, dispose d'un type ayant spécifiquement pour fonction de représenter les réels : il s'agit des flottants, ou des `float`, qui s'appuient sur les deux propriétés abordées dans la section précédente[(1)] et dont le codage se décompose en trois parties :

- un signe $s$, stocké sur 1 bit
- un exposant $e_{k-1}...e_0$, sur un nombre de bits $k$ fixé
- une mantisse $m_1...m_l$, sur un nombre de bits $l$ fixé.

**Remarque :** Bien-sûr, les valeurs de $l$ et de $k$ déterminent l'ensemble des réels que l'on peut effectivement représenter avec une précision parfaite.

Considérons le nombre $N = s\, e_{k-1}e_{k-2}...e_0\, m_1m_2...m_l \in \{0, 1\}^{k+l+1}$.

Si $\exists\,(i,i') \in [0..k-1]^2$, $e_i = 0$ et $e_{i'} = 1$, il représente ce qu'on appelle sa valeur normalisée :

$$(-1)^s\, 2^{-(2^{k-1}-1)+\mathrm{val}_2(e_{k-1}\ldots e_0)}\left(1+\sum_{i=1}^{l}\frac{m_i}{2^i}\right)$$

Si $\forall\, i \in [0..k-1]$, $e_i = 0$, le réel représenté par $N$ porte le nom de valeur dénormalisée et vaut :

$$(-1)^s\, 2^{-2^{k-1}+2}\sum_{i=1}^{l}\frac{m_i}{2^i}$$

Si enfin $\forall\, i \in [0..k-1]$, $e_i = 1$, alors :

- soit $\forall\, i \in [1..l]$, $m_i = 0$ auquel cas $N$ encode $(-1)^s\,\infty$
- soit $\exists\, i \in [1..l]$, $m_i \neq 0$, auquel cas $N$ n'encode aucun élément de $\overline{\mathbb{R}}$, sa valeur en C est alors `NaN` (pour "not a number").

**Remarque :** L'exposant encode donc "l'ordre de grandeur" du réel représenté, tandis que la mantisse en donne les "chiffres significatifs". Conformément à ce qui a été dit précédemment[(1)], on a ainsi une écriture de la forme $x = (-1)^s 2^k y$ avec $(k,y) \in \mathbb{N}\times[0,1]$, de manière analogue à l'écriture scientifique en physique.