Chapitre 2 - Fonctions et codages

Benjamin WACK (cours) - Mica MURPHY (note) - Antoine SAGET (note)

Lundi 05 Novembre 2018

1) Fonctions

a) Définitions et notations

Une **fonction** d'un ensemble X vers un ensemble Y associe à **chaque** élément x de X **un et un seul** élément de Y noté $f(x)$.

> *Remarque.* Lorsqu'on parle de la fonction racine carrée, on choisit de prendre la racine positive, afin d'avoir une seule solution.

Notation. "à la UML" : $X \ \underline{0..* \quad f > \quad 1}\ Y$ "À tout élément de X correspond un élément de Y" "À tout élément de Y peut correspondre 0, 1 ou plusieurs éléments de X"

Vocabulaire. - En maths, on appelle ça une fonction totale. - Une fonction $f : X \to \{0, 1\}$ est appelée un **prédicat**.

Notions d'image. (important !) Soient $x \in X$ et $y \in Y$. On appelle : - **l'image de** x **par** f l'élément $f(x)$ - **l'image de** f est $\{f(x) \mid x \in X\}$ (partie de Y) - **l'image réciproque de** y **par** f notée $f^{-1}(y) = \{x \in X \mid f(x) = y\}$

Exemples. - $X = \mathbb{Z}$, $Y = \{0, 1, 2, 3, 4\}$, $f(x) =$ reste de la division de x par 5 - $f(12) = 2$ - $f(X) = Y$ - $f^{-1}(0) = \{\dots, -10, -5, 0, 5, 10, \dots\}$: les multiples de 5

- $X = \{a, b\}^*$, $Y = \mathbb{Z}$, $f = lg$
  - $f(X) = \mathbb{N}$
  - $f^{-1}(3) = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ (antécédents de 3)
  - $y \in Y$ n'a pas d'antécédent ssi $f^{-1}(y) = \emptyset$ ssi $y \notin Im(f)$

b) Notion de bijection

Une **bijection** de X dans Y est une fonction telle que **chaque** élément de Y a **exactement un** antécédent.

Notation "à la UML" $X \ \underline{1 \quad f > \quad 1}\ Y$

Autrement dit : - pour tout $y \in Y$, il existe un unique $x \in X$ tq $f(x) = y$ - pour tout $y \in Y$, $f^{-1}(y)$ est un singleton (l'ensemble des antécédents) - il existe une fonction $g : Y \to X$ telle que
$$\begin{cases} \forall y \in Y, f(g(y)) = y \text{ (existence)} \\ \text{et} \\ \forall x \in X, g(f(x)) = x \text{ (unicité)} \end{cases}$$

g est alors appelée **la fonction réciproque de f** et dans ce cas on peut la noter f^{-1}

g est alors une bijection aussi dont la réciproque est f

Méthode pour démontrer qu'une fonction f est bijective

- Proposer une fonction $g : Y \to X$
- Démontrer que pour tout $y \in Y$ et $x \in X$, $\boxed{g(y) = x \Leftrightarrow y = f(x)}$

  Remarque. En pratique, on part de $y = f(x)$ et on procède par équivalences successives pour exprimer x en fonction de y

À propos des cardinaux

Si il existe une bijection de X vers Y (noté $X \cong Y$) alors $\boxed{card\ X = card\ Y}$.

Contraposée : si X et Y n'ont pas le même cardinal, il ne **peut pas** y avoir de bijection entre eux.

On peut également compter les bijections : si $card\ X = card\ Y = n$ alors il y a $\boxed{n!\text{ bijections}}$ de X vers Y

Exemples. - Soient $X = \{a, b, c\}$ et $Y = \{1, 2, 3\}$ alors il y a des bijections $X \to Y$

par exemple $f : \begin{cases} a & \mapsto & 1 \\ b & \mapsto & 5 \\ c & \mapsto & 2 \end{cases}$ est une bijection

- $\sin : [-\frac{\pi}{2}, \frac{\pi}{2}] \to [-1, 1]$ est une bijection
- $\lg : A^* \to \mathbb{N}$ n'est pas une bijection (sauf si A ne contient qu'un seul symbole)

c) Injectivité, Surjectivité

Définition. Une **injection** de X vers Y est une fonction f telle que chaque élément de X ait **au maximum un** antécédent.

$$\forall y \in Y,\ card(f^{-1}(y)) = 0 \text{ ou } 1$$

Diagramme UML : X $\underline{0..1 \quad\ f > \quad 1}$ Y

Deux caractérisation équivalentes : - $\forall x_1, x_2 \in X$, si $f(x_1) = f(x_2)$ alors $x_1 = x_2$ - $\forall x_1, x_2$, si $x_1 \neq x_2$ alors $f(x_1) \neq f(x_2)$

Propriété. Si il existe une injection de X dans Y, alors $card\ X \leq card\ Y$

Contraposée. Principe des pigeons (ou des tiroirs) : si $card\ X > card\ Y$ alors il existe deux éléments x_1 et $x_2 \in X$ tels que $x_1 \neq x_2$ et $f(x_1) = f(x_2)$

Remarque. Ne marche qu'avec des ensembles **finis** !

Application à l'informatique. Si f est un **code**, on souhaite pouvoir décoder de façon unique, autrement dit on veut que f soit injective.

Si $f(x_1) = f(x_2)$ avec $x_1 \neq x_2$, on parle de **collision** (voir tables de hashage)

Définition. Une **surjection** (peu utilisé en informatique) est une fonction $X \to Y$ telle que **chaque élément** de Y ait **au moins** un antécédent.

Diagramme UML : X $\underline{1..* \quad\ f > \quad 1}$ Y

Autrement dit $Im(f) = Y$

Propriété. Si **il existe une surjection** de X vers Y, alors $\boxed{card\ X \geq card\ Y}$

Propriété. Une fonction est une **bijection** ssi elle est à la dois une injection et une surjection.

> *Remarque.* Cette propriété est principalement utilisée pour montrer qu'une fonction n'est pas une bijection

Exemples. - sin : $\mathbb{R} \to [-1; 1]$ n'est pas une bijection car elle n'est pas injective : $sin(\pi) = sin(0)$ - sin : $[-\frac{\pi}{2}; \frac{\pi}{2}] \to \mathbb{R}$ n'est pas une bijection car elle n'est pas surjective : 2 n'a pas d'antécédent - sin : $[-\frac{\pi}{2}; \frac{\pi}{2}] \to [-1; 1]$ est une bijection

2) Codes

a) Définitions

Soient A et B deux alphabets.

Un **code** ou **codage** de A vers B est une fonction $C : A^* \to B^*$ (ou éventuellement $C : X \to Y$ avec $X \subseteq A^*$ et $Y \subseteq B^*$)

Exemple. Numéro de sécurité sociale : 2, 53, 07, 75, 073, 004, 83 : 13 chiffres signifiants + clé sur 2 chiffres

$K : \{0, \dots, 9\}^{13} \to \{0, 9\}^2$ $K(x)$ est calculé tel que : $x + K(x)$ soit divisible par 97 **et** $K(x) < 97$ (pour que K soit une fonction)

> *Remarque.* 97 est le plus grand nombre premier sur 2 digits

Si on cherche à **retrouver** l'information x à partir de son code $C(x)$, il **faut** que C soit injective (sinon il y a ambiguïté au moment du décodage).

Exemple. Somme de contrôle ($MD5$, CRC, ...) pour de gros fichiers. Ce n'est pas un code injectif.

b) Communications

```
                    |------canal-------|
message  > encodage >      >>>>>       > décodage > message
                    |------------------|
     émetteur       |  message encodé  |      récepteur
```

Plusieurs propriétés peuvent être demandées à l'encodage : 1. Peu gourmand en longueur de message : code **compresseur** 2. Confidentialité : code **secret** 3. code **détecteur ou correcteur d'erreurs** pour compenser les imperfections du canal

En général, on applique ces 3 étapes : - dans cet ordre à la compression - dans le sens inverse à la réception

c) Codes alphabétiques

Pour définir un code **alphabétique**, on part d'une fonction $C : A \to B^*$ et on prolonge C sur A^* de sorte que $C(a_1 a_2 \dots a_n) = C(a_1)C(a_2)\dots C(a_n)$

Autrement dit C est un **homomorphisme de monoïdes** de $(A^*, .)$ vers $(B^*, .)$

Exemple. Code morse : $\{A, B, \dots, Z, 0, 1, \dots, 9\} \to \{\cdot, -\}$ avec par exemple $M(S) = ---$ et $M(O) = \cdots$ d'où $M(SOS) = ---\cdots---$, mais $M(E) = \cdot$, d'où ambiguïté $M(O) = M(EEE)$. En fait on ajoute une pause entre les caractères qui doit être considéré comme un troisième symbole (on peut le noter $|$).

Exemple. Code ASCII $\{A, \ldots, Z, a, \ldots, z, 0, \ldots, 9, +, -, /, *, :, ;, ,, ., \ldots\} \to \{0,1\}^7$

Il est clairement injectif car chaque groupe de 7 bits code un caractère

3) Codes compresseurs

L'idée est de construire un code alphabétique efficace en donnant un code plus court aux symboles les plus fréquents.

Pour faciliter le décodage, on construira un code **préfixe** : un code tel que $\forall x, y \in A, C(x) \not\sqsubseteq C(y)$

Exemple. - $f : \{a, b, c\} \to 0, 1^*$, $f(a) = 0$, $f(b) = 10$, $f(c) = 110$ est un code préfixe - $g : \{a, b, c\} \to 0, 1^*$, $g(a) = 01$, $g(b) = 010$, $g(c) = 110$ n'est pas un code préfixe car $g(a) \sqsubseteq g(b)$

Propriété. $\boxed{\text{Tout code préfixe est injectif}}$

Algorithme de Huffman

Soit un alphabet A où chaque symbole x est associé à un poids $p(x)$ qui est sa fréquence (ou son nombre d'occurences dans le texte à code)

L'agorithme de Huffman va produire le code préfixe alphabétique **le plus court** pour cet alphabet.

On va contruire un **arbre de codage** : - Initialement, on construit une feuille pour **chaque** symbole x affectée du poids $p(x)$ - Tant qu'il reste plus d'un nœud sans père : - Soient x et y les deux nœuds de poids minimal - Construire un nouveau nœud n dont : - les fils sont x et y - le poids $p(n) = p(x) + p(y)$

Le code d'un symbole x est le chemin de la racine à la feuille qui porte x avec comme valeur 0 lorsqu'on passe à gauche et 1 lorsqu'on passe à droite. - Si $p(x) \geq p(y)$ alors le code de x est plus court que celui de y - C'est un code préfixe car tous les symboles sont dans des feuilles

Codage et décodage

> **Attention.** Cet algorithme peut produire des codes très variés pour le même alphabet.

En général on transmet le message compressé + la table des codes.

Codage : évident : il suffit de concaténer les codes des symboles du message initial : $C(bbad) = 0000001001$

Décodage : on utilise l'arbre (comme un automate) - à chaque bit lu, on suit une branche de l'arbre - à chaque fois qu'on arrive sur une feuille, on écrit le symbole correspondant et on repart de la racine pour la suite

4) Codes détecteurs et correcteurs d'erreur

Principe :

```
mot u -codage-> mot v codé --transmission--> mot w codé avec erreur --> v --> u
        émetteur           |---------------|              récepteur
```

Idée : ajouter dans le message de la **redondance** pour pouvoir reconstituer u malgré certaines erreurs de transmission

Exemple. Numéros avec "clé" numéro de Sécurité Sociale, RIB, code-barres

Exemple. Codes à répétition : on répéte chaque bit k fois. - Si $k = 2$: on peut détecter (mais pas corriger) une erreur sur 2 bits consécutifs - Si $k = 3$: on peut corriger 1 erreur sur 3 bits par une règle de majorité

Exemple subtil. Bit de parité : pour chaque groupe de k bits on rajoute 1 bit égal à leur "ou exclusif" Il y a alors un nombre pair de 1. Une erreur parmi k bits est détectée car elle donne un nombre impair de 1

Application ASCII sur 7 bits + 1 bit de parité

Hypothèse Aucun bit n'est perdu ou ajouté, la seule erreur possible est l'échange d'un 1 en 0 ou d'un 0 et 1. De plus en général on suppose un certain taux d'erreurs maximal (1 erreur pour k bits transmis).

Notion de distance minimale d'un code.

Soit $C : A^* \to B^*$. La distance minimale de C, notée d_C est la plus petite distance entre deux mots $C(a_1)$ et $C(a_2)$ pour a_1 et a_2 dans A^*. Elle vaut toujours au moins 1 (sinon C est ambigü).

De plus, - on peut détecter une erreur ssi $d_C \geq 2$ et on peut corriger une erreur ssi $d_C \geq 3$. - on peut corriger une erreur ssi $d_C \geq 3$ - plus généralement, on peut détecter $d_C - 1$ erreurs et corriger $\left\lfloor \frac{d_C - 1}{2} \right\rfloor$

Code de Hamming (Minitel)

Principe

On code chaque mot de 4 bits $x_1x_2x_3x_4$ sur 7 bits avec $x_5 = x_1 + x_2 + x_4$, $x_6 = x_1 + x_3 + x_4$ et $x_7 = x_2 + x_3 + x_4$

Il permet de corriger 1 erreur sur 7 bits.

Une matrice $n \times p$ à coefficients dans un ensemble A est un "tableau" de n lignes et p colonnes éléments pris dans A. Pour faire du calcul sur ces matrices, on demande que l'ensemble A soit un **anneau** donc qu'il possède deux opérations : - une opération + : associative, commutative, avec un élément neutre, chaque élément possède un opposé - une opération $\times$: associative, avec un élément neutre et distributive sur la somme: $x \times (y + z) = x \times y + x \times z$

Exemple. $(\mathbb{R}, +, \times)$ OK, $(\mathbb{Z}, +, \times)$ OK, $(\mathbb{N}, +, \times)$ NOK, $(\{0, 1\}, \text{or}, \text{and})$, $\boxed{(\{0, 1\}, \text{xor}, \text{and})}$

Opérations sur les matrices : - Somme : $A_{np} + B_{np} = (a_{ij} + b_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$ - Produit : $(A_{np} \times B_{pq})_{ij} = a_{i1} \times b1j + \cdots + a_{ip} \times b_{pj}$

Construction du code de Hamming [4, 7]

Pour représenter le calcul des 3 bits de redondance, on utilise la matrice génératrice G :

$$G_{4,7} = \begin{pmatrix} 1 & 0 & 0 & 0 & & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & & 1 & 1 & 1 \end{pmatrix}$$

Pour coder un message $U_{1,4} = (x_1, x_2, x_3, x_4)$ On calcul $U \times G = V_{1,7}$

Correction d'erreur

Le récepteur utilise la matrice de contrôle $H_{3,7} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$

Il calcule $W \times {}^tH = S_{1,3}$ le syndrome : - si $S = (000)$ alors il n'y a pas d'erreur - sinon, S est une des colonnes de H qui donne la position de l'erreur

Ensuite il suffit de ne garder que les 4 premiers bits du message corrigé

Preuve. - $G \times {}^tH = 0$ en effet $G = (I_4|G_0)$ et $H = ({}^tG_0|I_3)$ d'où $G \times {}^tH = (I_4|\ G_0) \times \left(\frac{G_0}{I_3}\right) = I_4 \times G_0 + G_0 \times I_3 = 2G_0 = 0$ (car $1+1=0$) - Donc si v est transmis sans ereur, $s = w \times {}^tH = v \times^t H = u \times G \times {}^tH = u \times 0 = 0$ - Et s'il y a une erreur sur le bit numéro i : $w = v + (0 \cdots 0\underset{i}{1}0 \cdots 0)$ et $w \times {}^tH = (v + (0 \cdots 010 \cdots 0)) \times {}^tH = v \times {}^tH + (0 \cdots 010 \cdots 0) \times {}^tH$

5) Codes secrets

Contexte et objectif

```
      émetteur --> message codé c --> récepteur
       écrit m          ^             reçoit c
    (clé k) v        espion            v (clé k)
message codé c                        message (décodé) m
```

k = clé, secret partagé entre l'émetteur et le récepteur

- L'émetteur **chiffre** m à l'aide de k
- Le récepteur **déchiffre** c à l'aide de k
- L'espion essaye de **décrypter** c sans l'aide de k

Exemple. Code de César : remplacer chaque lettre par celle 3 rangs plus loin pour chiffrer et reculer de 3 lettres pour déchiffrer.

Inconvénients : - facile à décrypter - en particulier car une lettre est toujours codée de la même façon - une seule lettre suffit à connaître le décalage

Exemple. Code de Vernam : une code binaire. - message m sur n bits - clé k de même longueur - message codé $c = m \oplus k$ bit par bit

C'est un code parfait impossible à décrypter.

Inconvénients : - taille de la clé >= taille du message à coder - elle ne peut être utilisée qu'une seule fois = jetable

Entiers modulo (très utilisés en cryptographie)

Définition. Soit un entier $N \geq 2$. Si x et $y \in \mathbb{Z}$ ont le même reste dans la division par N, on considère que x et y sont équivalentS.

Visualisation pour N = 6

On peut imaginer enrouler la droite des entiers relatifs sur un cercle de circonférence 6.

Les éléments aux mêmes endroits sur le cercle forment une **classe** (= tous les entiers qui ont le même reste). On note $\overline{x}$ la classe de x.

$$\overline{x} = \{x + kN \quad \text{pour } k \in \mathbb{Z}\}$$

En général on privilégie un représentant : celui entre 0 et $N-1$ (reste de la division euclidienne).

Deux entiers ont la même classe ssi ils ont le même reste dans la division par N.

On note $\mathbb{Z}/_{N\mathbb{Z}}$ l'ensemble des classes modulo N.

$$\mathbb{Z}/_{N\mathbb{Z}} = \{\overline{0}, \overline{1}, \overline{2}, \ldots, \overline{N-1}\}$$
$$\overline{x} \in \mathbb{Z}/_{N\mathbb{Z}} \text{ et } \overline{x} \subseteq \mathbb{Z}$$

Propriété. Si $\overline{x} = \overline{x'}$ et $\overline{y} = \overline{y'}$ alors - $\overline{x+y} = \overline{x'+y'}$ - $\overline{x-y} = \overline{x'-y'}$ - $\overline{x*y} = \overline{x'*y'}$

On définit donc des opérations sur $\mathbb{Z}/_{N\mathbb{Z}}$:

$$\overline{x} + \overline{y} = \overline{x+y}$$

$$\overline{x} \times \overline{y} = \overline{x \times y}$$

$$\overline{x} - \overline{y} = \overline{x-y}$$

Rapport avec les codes secrets

- César manipule des entiers modulo 26
  - chiffrer = ajouter 3
  - déchiffrer = soustraire 3
- Vernam travaille dans $\mathbb{Z}/_{2\mathbb{Z}}$ car le $\oplus$ est $+$ dans $\mathbb{Z}/_{2\mathbb{Z}}$

6) Théorème des bergers

Théorème. Soit une fonction $f = X \to Y$. Alors les ensembles $f^{-1}(y)$ (= antécédents de y) pour $y \in Im(f)$ forment une partition de X.

$$\boxed{X = \sum_{y \in Im(f)} f^{-1}(y)}$$

(X sont les moutons, Y sont leurs bergers : chaque mouton appartient à un et un seul berger)

On classe les éléments de X selon leur image par f.

Corrollaire. Si X est fini,

$$\boxed{card\ X = \sum_{y \in Im(f)} card(f^{-1}(y))}$$

En particulier, si chaque $y \in Im(f)$ a le même nombre k d'antécédents, alors

$$\boxed{cardX = k \times card(Im(f))}$$

Exemple. - Classer les mots binaires de longueur 3 par leur premier bit :

$$f : \begin{cases} \{0,1\}^3 & \to & 0,1 \\ xyz & \mapsto & x \end{cases}$$

$$\{0,1\}^3 = \{000, 001, 010, 011\} + \{100, 101, 110, 111\}$$

- Classer les mots binaires de longueur 3 par le nombre de bits distincts qu'ils contiennent :

$$\begin{cases} f(000) = f(111) = 1 \\ f(001) = f(010) = \cdots = f(110) = 2 \end{cases}$$

$$\{0,1\}^3 = \{000, 111\} + \{001, 010, 011, 100, 101, 110\}$$

- Classer les entiers selon leur *parité* : $\mathbb{Z} \to \{0,1\}$ et $\mathbb{Z} = Pairs + Impairs$ - Classer les mots de A^* selon leur *longueur* : $A^* \to \mathbb{N}$

$$A^* = \{\epsilon\} + \{a, b, c, \dots\} + \{aa, ab, ba, ac, \dots\} + \dots$$

- Classer les élèves d'INFO3 selon leur année de naissance
- Un **prédicat** $p : X \to \{vrai, faux\}$ permet de classer les éléments de X selon la réponse à une **question fermée**.

Ensembles de fonctions

On note Y^X l'ensemble des fonctions de X dans Y (par exemple $\{vrai, faux\}^X$ est l'ensemble des prédicats sur X).

- Si X et Y sont finis, $\boxed{card(Y^X) = (cardY)^{(cardX)}}$

  Remarque. Soit P une partie de X. Il lui correspond **un unique** prédicat $p : X \to \{vrai, faux\}$ tq $p(x) = vrai$ ssi $x \in P$.

  On vient d'établir une bijection entre $\mathcal{P}(X)$ et $\{vrai, faux\}^X$.

  On retrouve donc $\boxed{card\ \mathcal{P}(X) = 2^{card\ X}}$

- Si X et Y sont finis, il y a $(cardX)!$ bijections de X dans X et $A_p^n = \frac{n!}{p!(n-p)!} = n \times (n-1) \times \cdots \times (n-(p-1))$ injections de X vers Y, où $card\ X = p \leq card\ Y = n$.