JEAN-YVES LE BOUDEC, PATRICK THIRAN,

RÜDIGER URBANKE

SCIENCES DE L'INFORMATION

NOVEMBER 28, 2014

Les fichiers échangés sur Internet et stockés sur les disques durs contiennent de l'information qui deviendra finalement du texte, des images ou des sons. Comment cette information est-elle mesurée et comprimée ? comment est-elle sécurisée pour éviter les copies illicites ? comment est-elle protégée contre les erreurs lors d'une recopie ? Ce sont les trois questions auxquelles ce livre répond.

Il s'adresse aux étudiants de première année de l'enseignement supérieur, et au-delà, à toute personne intéressée à une première approche scientifique des sciences de l'information. Les pré-requis sont une formation de base en mathématiques du niveau de la fin de l'enseignement secondaire. Il est issu des supports du cours "Sciences de l'Information" donné en première année d'informatique et de systèmes de communication à la faculté I&C de l'EPFL.

Ce livre a été écrit avec le souci constant de délivrer les idées profondes qui sous-tendent les théories, tout en restant aussi simple que possible, mais sans déformer la vérité scientifique. Nous avons mis en avant les concepts fondamentaux à chaque fois que cela simplifie l'exposé, même si en apparence la description peut paraître plus longue. Ainsi par exemple, nous présentons le théorème des restes chinois comme un isomorphisme, plutôt que comme une méthode astucieuse pour résoudre un certain type d'équations ; ou encore, nous présentons le théorème d'Euler comme un cas particulier du théorème plus simple qui dit que la période d'un élément d'un groupe fini divise le nombre d'éléments du groupe. Cette approche, qui dévoile les structures sous l'habillage des algorithmes, nous semble essentielle à la compréhension des sciences de l'information ; elle vise aussi à cultiver chez le lecteur l'habitude de reconnaître des abstractions communes dans des situations apparemment très différentes – ce qui est à la base de la pensée algorithmique moderne. Et peut-être permettra-t-elle à la lectrice ou au lecteur de percevoir une certaine beauté dans les sciences de l'information...

Dans la partie I nous découvrons comment définir la quantité d'information et sa mesure, en utilisant des axiomes simples et intuitifs. Les concepts essentiels qui apparaissent sont l'entropie et l'entropie conditionnelle. La définition même de ces concepts est basée sur une interprétation probabiliste du monde ; il n'est cependant pas nécessaire de connaître en détail la théorie des probabilités car les concepts nécessaires sont expliqués de manière élémentaire dans un chapitre de rappel. Les codes de source permettent de comprimer n'importe quel fichier numérique ; leurs propriétés fondamentales sont analysées et expliquées sur des exemples très simples. Cela permet d'aborder le premier théorème de la théorie de l'information, qui établit que la quantité d'information, définie plus haut à partir d'axiomes intuitifs, correspond exactement à la taille en bits que peut obtenir le meilleur code de source.

Dans la partie II nous découvrons comment sécuriser l'information par la cryptographie. Nous analysons tout d'abord ce qu'est

la confidentialité parfaite – cela dérive simplement des concepts de la partie I. La cryptographie est fille de la théorie des nombres ; nous en donnons une présentation aussi élémentaire et pratique que possible, sans en cacher les idées fondamentales. Pour cela, nous introduisons l'arithmétique modulaire et quelques éléments d'algèbre abstraite, ce qui forme aussi le sur lequel est bâti la partie suivante. La présentation et l'analyse de l'algorithme de cryptographie asymétrique RSA en découlent alors simplement.

La partie III est consacrée à la protection de l'information contre les erreurs qui apparaissent durant l'écriture, la lecture, la transmission ou le stockage. Les concepts fondamentaux sont la distance minimale et la borne de Singleton, qui permettent de quantifier la puissance correctrice d'un code correcteur. Pour la réalisation pratique de codes, nous avons choisi de présenter les codes linéaires : ce sont parmi les plus puissants et les plus commodes à analyser. Cela nous permet de permet de présenter et comprendre les codes de Reed Solomon, qui sont parmi les plus efficaces et les plus utilisés. Ici aussi, la présentation n'essaie pas d'escamoter la théorie sous-jacente, qui est celle des espaces vectoriels sur des corps finis – une théorie particulièrement puissante et dont la beauté est immédiatement accessible à toute personne ayant étudié un petit peu de géométrie vectorielle élémentaire.

Une lecture interactive de ce livre est facilitée et encouragée par les nombreuses questions en marge, qui ont été insérées dans le but de d'activer compréhension profonde.

Q. 1. Où se trouve la réponse à une question en marge ?

Pour la commodité du lecteur, l'index rassemble tous les *termes nouveaux* et permet de les retrouver rapidement dans leur contexte.

Les preuves des théorèmes sont classées en deux niveaux de difficultés. Les preuves signalées par ce signe ☺ sont des preuves faciles mais qui illustrent bien la logique et la façon de raisonner en sciences de l'information ; chacun(e) devrait les étudier. Les autres preuves sont abordables mais demandent plus de temps et peuvent être omises. Les chapitres, sections, questions marqués par cet autre signe ⋆ peuvent être omis sans risque pour la compréhension globale.

De nombreuses personnes ont contribué à ce livre de manière directe ou indirecte, en particulier : Gregory Dyke, Iryna Andriyanova, Shrinivas Kudekar, Ramtin Pedarsani, Bastani Parizi Mani, Nicolas Gast et Dan-Cristian Tomozei.

Que tous soient remerciés ici.

Jean-Yves Le Boudec, Patrick Thiran, Rüdiger Urbanke

Table des matières

I

Codage de Source

0
Préliminaires de Probabilités

Combien d'information un message contient-il ? Comment mesurer l'information ? Pourquoi est-il possible de comprimer des données sans en perdre ? Ce sont quelques unes des questions que nous allons aborder dans cette partie. Pour cela nous allons découvrir le concept central d'*entropie* d'une source d'information.

Mais avant de définir l'entropie, il nous faut *modéliser* les sources de données informatiques en utilisant quelques concepts simples de la théorie des probabilités, que nous rappelons maintenant.

0.1 Source et Probabilité

Nous appelons *source* la donnée d'un ensemble fini appelé *alphabet* $\mathcal{A}$ et d'une *densité de probabilité*, c'est à dire d'une application $p : \mathcal{A} \to [0,1]$ satisfaisant $\sum_{s\in\mathcal{A}} p(s) = 1$. Les éléments de $\mathcal{A}$ sont appelés les *symboles*.

Une *application* d'un ensemble E vers un ensemble F fait correspondre à tout élément x de E un et un seul élément de F (appelé l'*image* de x). Une *fonction* d'un ensemble E vers un ensemble F fait correspondre à *certains* élément de E un et un seul élément de F. Le domaine de définition d'une fonction est le sous-ensemble de E constitué des éléments qui ont une image. Ainsi $x \mapsto x^2$ définit une application de $\mathbb{R}$ vers $\mathbb{R}$, alors que $x \mapsto \sqrt{x}$ définit une fonction de $\mathbb{R}$ vers $\mathbb{R}$, dont le domaine de définition est $[0,+\infty)$; $x \mapsto \sqrt{x}$ définit une application de $[0,+\infty)$ vers $\mathbb{R}$.

Dans la notation d'intervalle telle que $[0,1)$, 0 est inclus et 1 est exclus, alors que dans la notation $[0,1]$, les deux bornes 0 et 1 sont incluses.

La notation $\sum_{s\in\mathcal{A}} p(s)$ veut dire la somme de tous les $p(s)$ quand s prend toutes les valeurs possibles dans l'ensemble fini $\mathcal{A}$.

Exemple 0.1 (Pièce Non Biaisée) $\mathcal{A} = \{\text{“}P\text{”},\text{“}F\text{”}\}$, $p(\text{“}P\text{”}) = p(\text{“}F\text{”}) = 0.5$.

Exemple 0.2 (Le Vélo d'Anne) *Anne prête son vélo à Bernard mais ne lui donne pas le numéro du cadenas, qui est un nombre de 4 chiffres. Bernard pense pouvoir le deviner et appelle Anne, qui n'accepte de répondre qu'à une seule question, et par oui ou non. Ici* $\mathcal{A} = \{\text{“}Oui\text{”},\text{“}Non\text{”}\}$ *et* $p(\text{“}Oui\text{”}) = 0.0001$, $p(\text{“}Non\text{”}) = 0.9999$.

En théorie des probabilités, une source est appelée “*variable aléatoire* discrète".

Exemple 0.3 (Dé Non Pipé) $\mathcal{A} = \{1,2,3,4,5,6\}$ *et* $p(i) = 1/6$ *pour tout* $i \in \mathcal{A}$.

On dit que les sources des exemples 0.1 et 0.2 sont *binaires* ($\mathcal{A}$ a 2 éléments).

Exemple 0.4 (Deux Tirages Successifs d'un Dé Non Pipé) $\mathcal{A} = \{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}$ *et* $p(i,j) = 1/36$ *pour tout* $(i,j) \in \mathcal{A}$.

Le signe $\times$ est le *produit cartésien*. Il signifie que $\mathcal{A}$ est l'ensemble des 36 couples (i,j) avec $i = 1,2...,6$ et $j = 1,2,...,6$. Un *couple* est une suite de deux éléments ; l'ordre compte et il peut y avoir des répétitions.

On dit que tous les symboles de la source sont *équiprobables*, ou encore que la densité de probabilité est *uniforme* quand $p(s) = 1/M$ pour tout $s \in \mathcal{A}$, où $M = \text{card}(\mathcal{A})$ est le nombre de symboles dans l'alphabet. Les symboles des sources des Exemples 0.1, 0.3 et 0.4 sont équiprobables. Ce n'est pas le cas pour l'Exemple 0.2.

Si A est un ensemble fini, $\text{card}(A)$ est le nombre d'éléments de A et se lit *cardinal* de A.

0.2 *Probabilité d'un Evénement, Indépendance, Probabilités Conditionnelles*

On appelle *événements* les sous-ensembles de $\mathcal{A}$; on dit que la *probabilité* d'un événement E est

$$P(E) \stackrel{\text{def}}{=} \sum_{s \in E} p(s)$$

Exemple 0.5 (Suite de l'Exemple 0.4) *$E = \{(i,j) \in \mathcal{A}$ tels que $i = j\}$. E modélise l'événement "Les deux dés ont produit le même résultat". On a*

$$P(E) = p(1,1) + p(2,2) + p(3,3) + p(4,4) + p(5,5) + p(6,6) = 1/6$$

On dit que deux sous-ensembles B et C sont *indépendants* si

$$P(B \cap C) = P(B)P(C)$$

$B \cap C$ se lit "B inter C". C'est le sous-ensemble obtenu en prenant tous les éléments de l'alphabet qui sont à la fois dans B et C. En langage d'événements, on dit aussi "B et C"

L'indépendance exprime que la réalisation d'un événement ne donne aucune information sur l'autre.

Exemple 0.6 (Suite de l'Exemple 0.4) *$B = \{(6,j)$ avec $j \in \{1,...,6\}\}$. B est l'événement "Le premier tirage vaut 6". On a $P(B) = 6 \cdot \frac{1}{36} = 1/6$, $B \cap E = \{(6,6)\}$ et $P(B \cap E) = 1/36 = P(B)P(E)$, donc B et E sont indépendants. Savoir que le premier tirage vaut 6 n'aide pas à savoir si les deux tirages sont égaux, et réciproquement. Soit $C = \{(i,j) \in \mathcal{A} : i + j \geq 10\}$ ("la somme des tirages vaut au moins 10"). On a :*

Quand $P(E) = 1$ on dit que l'événement E est "certain".

$$\begin{aligned} P(C) &= p(4,6) + p(5,5) + p(5,6) + p(6,4) + p(6,5) + p(6,6) = 1/6 \\ B \cap C &= \{(6,4),(6,5),(6,6)\} \\ P(B \cap C) &= 3/36 \neq P(B)P(C) \end{aligned}$$

donc B et C ne sont pas indépendants. Savoir que le premier tirage vaut 6 donne de l'information sur la somme, qui a plus de chance de valoir au moins 10.

Soient B et C deux sous-ensembles de $\mathcal{A}$ avec $P(C) > 0$. La *probabilité conditionnelle* de B sachant C est

$$P(B|C) \stackrel{\text{def}}{=} \frac{P(B \cap C)}{P(C)}$$

Elle exprime la probabilité que l'on assigne à B quand on suppose a priori que C est réalisé.

$P(C|C) = 1$, comme il faut s'y attendre, C est certain sachant C !

Soient B et C deux sous-ensembles tels que $P(B) > 0$ et $P(C) > 0$. B et C sont indépendants si et seulement si $P(B|C) = P(B)$, ou bien encore si et seulement si $P(C|B) = P(C)$.

Exemple 0.7 (Suite de l'Exemple 0.6) *Calculons la probabilité $P(B|C)$, c'est à dire la probabilité que le premier tirage soit 6 sachant que la somme des deux tirages vaut au moins 10 :*

$$P(B|C) = \frac{P(B \cap C)}{P(C)} = \frac{3/36}{1/6} = 0.5$$

Notons que $P(B|C) \neq P(B)$, c'est à dire que B et C ne sont pas indépendants. Par contre.

$$P(B|E) \quad = \quad \frac{P(B \cap E)}{P(E)} = \frac{1/36}{1/6} = 1/6 = P(B)$$

c'est à dire que B et E sont indépendants, comme on le sait déjà.

0.3 Sources Composées, Source Indépendantes

Dans l'Exemple 0.4 (deux tirages successifs d'un dé), l'alphabet de la source est un produit cartésien $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ (avec $\mathcal{A}_1 = \mathcal{A}_2 = \{1,2,3,4,5,6\}$), c'est à dire que la source S produit un symbole de la forme (i,j) avec $i \in \{1,2,3,4,5,6\}$ et $j \in \{1,2,3,4,5,6\}$. On dit qu'on a une *source composée* à deux composantes. On peut dériver de la source composée S deux *sources marginales* qui sont S_1, le résultat du premier tirage, et S_2, le résultat du deuxième tirage. L'alphabet de S_1 est $\mathcal{A}_1 = \{1,2,3,4,5,6\}$, idem pour S_2. La densité de probabilité p_{S_1} de S_1 est déduite de la densité de probabilité p de S par la formule :

Q. 2. Quelle est la densité de probabilité de S_2 dans l'Exemple 0.4 ?

$$p_{S_1}(i) = \sum_{j \in \mathcal{A}_2} p(i,j) = 1/6, \quad \forall i \in \{1,2,3,4,5,6\}$$

Plus généralement, on dit qu'une source S est une *source composée* si son alphabet est de la forme $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_n$. Cela modélise une suite de n observations. A partir d'une source composée on peut dériver n *sources marginales*, obtenues en considérant chaque composante individuellement, ce qu'on écrit

L'expression "$\forall i \in E$" se lit "*quel que soit* $i \in E$". L'expression "$\exists i \in E$" se lit "*il existe* $i \in E$".

$$S = (S_1, S_2, ..., S_n)$$

La source marginale S_k a pour alphabet $\mathcal{A}_k$ et pour densité de probabilité

$$p_{S_k}(s) \stackrel{\text{def}}{=} \sum_{s_1 \in \mathcal{A}_1 ..., s_{k-1} \in \mathcal{A}_{k-1}, s_{k+1} \in \mathcal{A}_{k+1}, ..., s_n \in \mathcal{A}_n} p(s_1, ..., s_{k-1}, s, s_{k+1}, ..., s_n) \tag{1}$$

Dans l'Equation (2) nous pourrions écrire $\sum_{s_k \in \mathcal{A}_k} p_{S_k}(s_k)$ ou $\sum_{\text{truc} \in \mathcal{A}_k} p_{S_k}(\text{truc})$ au lieu de $\sum_{s \in \mathcal{A}_k} p_{S_k}(s)$: la variable de sommation est *muette* et son identificateur n'a qu'un sens local à l'intérieur de l'expression sur laquelle porte le signe $\sum$.

Notons que la densité marginale de probabilité est une densité de probabilité (comme le nom l'indique), et donc en particulier

$$\sum_{s \in \mathcal{A}_k} p_{S_k}(s) = 1 \tag{2}$$

Exemple 0.8 (Somme de Deux Dés Codée Sur Deux Chiffres) *Lisa lance deux dés non pipés et annonce la somme des deux nombres obtenus, codée comme un entier décimal à 2 chiffres. Par exemple si les dés donnent 5 et 6, Lisa annonce 11 ; si les dés donnent 2 et 2, Lisa annonce 04. La source L ainsi obtenue a pour alphabet $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ où $\mathcal{A}_1 = \{0,1\}$ et $\mathcal{A}_2 = \{0,1,2,3,4,5,6,7,8,9\}$ c'est à dire qu'un symbole de la source est de la forme (i,j) avec $i \in \mathcal{A}_1$ et $j \in \mathcal{A}_2$.*

La densité de probabilité de la source est donnée dans la table 1. Expliquons par exemple comment est obtenue $p_L(1,1)$, c'est-à dire la probabilité que la somme des deux dés soit 11. Considérons la source S de

l'Exemple 0.4 qui donne les résultats des deux dés. La probabilité demandée est la probabilité de l'événement $(5,6),(6,5)$ *pour la source* S*, donc vaut*

$$p_L(1,1) = p_S(5,6) + p_S(6,5) = \frac{1}{36} + \frac{1}{36} = \frac{2}{36}$$

Soient L_1 *et* L_2 *les sources marginales de* L*, i.e.* L_1 *est le premier chiffre de la somme et* L_2 *le deuxième. Les densités de probabilités de* L_1 *et* L_2 *peuvent s'obtenir par l'Eq.(1) ; par exemple :*

$$\begin{aligned} &p_{L_1}(0) = p_L(00) + p_L(01) + p_L(02) + p_L(03) + p_L(04) \\ &+p_L(05) + p_L(06) + p_L(07) + p_L(08) + p_L(09) \\ &= 0 + 0 + \tfrac{1}{36} + \tfrac{2}{36} + \tfrac{3}{36} + \tfrac{4}{36} + \tfrac{5}{36} + \tfrac{6}{36} + \tfrac{5}{36} + \tfrac{4}{36} = \tfrac{30}{36} = \tfrac{5}{6} \end{aligned}$$

j \ i	$p_L(i,j)$	0	1	$p_{L_2}(j)$
0		0	3/36	3/36
1		0	2/36	2/36
2		1/36	1/36	2/36
3		2/36	0	2/36
4		3/36	0	3/36
5		4/36	0	4/36
6		5/36	0	5/36
7		6/36	0	6/36
8		5/36	0	5/36
9		4/36	0	4/36
	$p_{L_1}(i)$	5/6	1/6	

TABLE 1: Densité de probabilité $p_L(i,j)$ de la source L de l'Exemple 0.8 et des deux sources marginales L_1 et L_2. $p_L(i,j)$ est la probabilité que la somme des deux dés soit ij, $p_{L_1}(i)$ la probabilité que le premier chiffre de la somme soit i et $p_{L_2}(j)$ la probabilité que le premier chiffre de la somme soit j.

Définition 0.1 *On dit que les sources marginales* $S_1,...,S_n$ *sont des sources indépendantes si*

$$p(s_1,s_2,...,s_n) = p_{S_1}(s_1)p_{S_2}(s_2)...p_{S_n}(s_n), \quad \forall (s_1,s_2,...,s_n) \in \mathcal{A}$$

Exemple 0.9 (Suite de l'Exemple 0.4, deux tirages successifs d'un dé.) *La source* S_1 *donne le résultat du tirage du premier dé,* S_2 *du deuxième. On a* $p(i,j) = 1/36 = p_{S_1}(i)p_{S_2}(j)$ *pour tout* $i \in \{1,2,3,4,5,6\}$ *et* $j \in \{1,2,3,4,5,6\}$*, donc les sources sont indépendantes. Pour un observateur qui connaît la densité de probabilité de* S*, c'est à dire qui sait que le dé n'est pas pipé, observer le résultat du premier tirage ne donne aucune information sur le deuxième tirage.*

Q. 3. Les deux sources marginales de l'Exemple 0.8 sont-elles indépendantes ?

Définition 0.2 *Si* S *est une source composée à deux composantes, la densité conditionnelle de la source marginale* S_2 *sachant que* $S_1 = s_1$ *est définie pour tout* $s_1 \in \mathcal{A}_1$ *tel que* $p_{S_1}(s_1) > 0$ *par*

$$p_{S_2|S_1}(s_2|s_1) \stackrel{\text{def}}{=} \frac{p(s_1,s_2)}{p_{S_1}(s_1)}, \quad \forall (s_1,s_2) \in \mathcal{A} \tag{3}$$

Si $p_{S_1}(s_1) = 0$ la densité conditionnelle de la source marginale S_2 sachant que $S_1 = s_1$ n'est pas définie. La densité conditionnelle de la source marginale S_2 sachant que $S_1 = s_1$ exprime ce que nous savons de S_2 quand on nous révèle que $S_1 = s_1$.

Notons cependant qu'il n'y pas de concept de temps ni de causalité dans cette définition : on n'a pas besoin qu'une source soit observée avant l'autre pour définir la densité conditionnelle.

La définition s'étend facilement à plus de deux sources marginales, auquel cas on peut faire de nombreuses combinaisons ; nous ne les écrivons pas en toute généralité car la notation devient lourde, mais il est facile de la deviner ; par exemple, si $S = (S_1,S_2,...,S_n)$ on peut définir la densité marginale de (S_1,S_2) sachant que $(S_3,...,S_n) = (s_3,...,s_n)$ par

$$p_{S_1,S_2|S_3,...,S_n}(s_1,s_2|s_3,...,s_n) = \frac{p(s_1,...,s_n)}{p_{S_3,...,S_n}(s_3,...,s_n)}$$

avec $p_{S_3,...,S_n}(s_3,...,s_n) = \sum_{(s_1,s_2)\in\mathcal{A}_1\times\mathcal{A}_2} p(s_1,s_2,s_3,...,s_n)$.

Le théorème suivant est essentiel pour comprendre ce que signifie l'indépendance. En bref, il exprime que deux sources marginales sont indépendantes si et seulement si la densité conditionnelle de l'une sachant l'autre ne dépend pas de l'autre. La propriété d'indépendance exprime donc que l'observation d'une source marginale ne donne aucune information sur l'autre, pour un observateur qui connaît la densité de probabilité de la source S.

Théorème 0.1 *Soit $S = (S_1, S_2)$ une source composée. Les propriétés suivantes sont équivalentes*

1. *les deux sources marginales S_1 et S_2 sont indépendantes, c'est à dire $p(s_1, s_2) = p_{S_1}(s_1)p_{S_2}(s_2)$, $\forall (s_1, s_2) \in \mathcal{A}$;*
2. *la densité conditionnelle de S_2 sachant que $S_1 = s_1$ (qui est définie pour tout s_1 tel que $p_{S_1}(s_1) > 0$) vaut $p_{S_2|S_1}(s_2|s_1) = p_{S_2}(s_2)$, $\forall (s_1, s_2) \in \mathcal{A}$;*
3. *la densité conditionnelle de S_2 sachant que $S_1 = s_1$ (qui est définie pour tout s_1 tel que $p_{S_1}(s_1) > 0$) ne dépend pas de la valeur de s_1 ;*

L'ordre des deux sources dans ce théorème n'a pas d'importance et peut être inversé. Ainsi nous pouvons dire que S_1 et S_2 sont indépendantes si et seulement si la densité conditionnelle de S_1 sachant que $S_2 = s_2$ (qui est définie pour tout s_2 tel que $p_{S_2}(s_2) > 0$) ne dépend pas de la valeur de s_2.

Preuve :

(1) $\Rightarrow$ (2) Supposons que S_1 et S_2 sont indépendantes. Donc $p(s_1, s_2) = p_{S_1}(s_1)p_{S_2}(s_2)$ pour tout $(s_1, s_2) \in \mathcal{A}$. Soit s_2 tel que $p_{S_2}(s_2) > 0$. Alors $p_{S_2|S_1}(s_2|s_1) = \frac{p_{S_1}(s_1)p_{S_2}(s_2)}{p_{S_1}(s_1)} = p_{S_2}(s_2)$ ce qui montre (2)

(2) $\Rightarrow$ (1) Soit s_1 tel que $p_{S_1}(s_1) > 0$. En utilisant la définition de la densité conditionnelle :

$$p(s_1, s_2) = p_{S_1}(s_1)p_{S_2}(s_2). \tag{4}$$

Il reste à montrer que l'égalité précédente vaut aussi pour un éventuel s_1 tel que $p_{S_1}(s_1) = 0$. Soit donc un $s_1 \in \mathcal{A}_1$ fixé, tel que $p_{S_1}(s_1) = 0$. Nous avons :

$$p_{S_1}(s_1) = \sum_{s_2 \in \mathcal{A}_2} p(s_1, s_2) = 0$$

or si une somme de nombres ≥ 0 est nulle, c'est que chaque élément de la somme est nulle. Donc $p(s_1, s_2) = 0$ pour tout $s_2 \in \mathcal{A}_2$. Donc l'égalité (4) vaut aussi pour notre s_1 fixé. En résumé, nous avons montré que l'égalité (4) vaut pour tous $s_1 \in \mathcal{A}_1, s_2 \in \mathcal{A}_2$, ce qui montre que S_1 et S_2 sont indépendantes.

(2) $\Rightarrow$ (3) C'est évident : nous supposons que $p_{S_2|S_1}(s_2|s_1) = p_{S_2}(s_2)$ qui ne dépend pas de s_1 donc (3) est vrai.

(3) $\Rightarrow$ (2) Nous supposons que $p_{S_2|S_1}(s_2|s_1)$ ne dépend pas de s_1. Donc nous pouvons écrire $p_{S_2|S_1}(s_2|s_1) = \varphi(s_2)$ où φ est une certaine application définie sur $\mathcal{A}_2$. Donc, en utilisant la définition de la densité conditionnelle, pour s_1 tel que $p_{S_1}(s_1) > 0$:

$$p(s_1, s_2) = p_{S_1}(s_1)\varphi(s_2) \tag{5}$$

Cette égalité reste vraie si $p_{S_1}(s_1) = 0$ car nous avons vu plus haut qu'alors $p(s_1, s_2) = 0$. En sommant cette égalité sur toutes les valeurs de s_1 dans $\mathcal{A}_1$ nous obtenons à gauche la densité marginale de S_2, donc, pour tout $s_2 \in \mathcal{A}_2$:

$$p_{S_2}(s_2) = \left[\sum_{s_1 \in \mathcal{A}_1} p_{S_1}(s_1) \right] \varphi(s_2) = \varphi(s_2)$$

(le crochet vaut 1, car p_{S_1} est une densité de probabilité sur $\mathcal{A}_1$). Cela montre que (2) est vraie.

Q. 4. Supposons que la densité conditionnelle de S_1 sachant que $S_2 = s_2$ ne dépende pas de la valeur de s_2, pour tout s_2 tel que $p_{S_2}(s_2) > 0$. Peut on conclure que la densité conditionnelle de S_2 sachant que $S_1 = s_1$ ne dépend pas de la valeur de s_1, pour tout s_1 tel que $p_{S_1}(s_1) > 0$?

Nous avons montré (1) $\Leftrightarrow$ (2) et (2) $\Leftrightarrow$ (3). □

Exemple 0.10 (Suite de l'Exemple 0.8) *L_1 est le premier chiffre de la somme de deux dés, et L_2 le deuxième. La densité de probabilité conditionnelle du deuxième chiffre sachant le premier est donnée par la Table 2. Elle est obtenue par application de l'Eq.(3) ; par exemple*

$$p_{L_2|L_1}(2|0) = \frac{p_L(02)}{p_{L_1}(0)} = \frac{1/36}{5/6} = \frac{1}{30}$$

Les deux sources ne sont pas indépendantes, la densité conditionnelle de L_2 sachant que $L_1 = i$ dépend de i (i.e. les deux dernières colonnes de la Table 2 ne sont pas identiques).

	i	0	1
j	$p_{L_2\|L_1}(j\|i)$		
0		0	3/6
1		0	2/6
2		1/30	1/6
3		2/30	0
4		3/30	0
5		4/30	0
6		5/30	0
7		6/30	0
8		5/30	0
9		4/30	0

TABLE 2: Densité conditionnelle du deuxième chiffre L_2 de l'Exemple 0.8 sachant que le premier est $L_1 = i$.

Q. 5. Quelle est la densité conditionnelle $p_{S_1|S_2}(i,j)$ de S_1 sachant que $S_2 = j$ dans l'Exemple 0.9 ?

1
Information et Entropie

Quelle quantité d'information y a-t-il dans une source délivrant des messages ? C'est la question à laquelle ce chapitre répond, sous le nom d'"entropie". Avant d'introduire cette notion, il nous faut introduire le concept d'"information" d'un événement.

1.1 Comment Mesurer l'Information

Considérons une source discrète d'information $S = (\mathcal{A}, p)$ délivrant un message, ou symbole $s \in \mathcal{A}$ avec la probabilité $p(s)$. Lorsqu'on reçoit un tel message s, si $p(s) = 1$ (et donc $p(s') = 0$ pour tout $s' \neq s$), il n'y a aucune surprise à recevoir le symbole s, celui-ci n'apporte aucune information. Par contre, si $p(s) = 0.0001$, la "surprise" de recevoir s parmi les M symboles que la source peut délivrer est beaucoup plus grande, ainsi que la quantité d'information apportée. L'information réside dans l'effet de surprise qu'elle engendre et croît donc en sens inverse de la probabilité.

On cherche à définir l'information $I(E)$ d'un événement E ; pour cela, il est donc naturel de prendre une fonction décroissante de la probabilité $P(E)$ de cet événement, c'est à dire de poser $I(E) \stackrel{\text{def}}{=} \varphi(P(E))$, où $\varphi : [0,1] \to \mathbb{R}^+$ est une fonction décroissante qu'il nous faut maintenant déterminer. Il y a beaucoup de telles fonctions décroissantes ; pour en choisir une, nous sommes guidés par le désir d'obtenir la propriété suivante :

$$\text{si } B \text{ et } C \text{ sont indépendants, alors } I(B \cap C) = I(B) + I(C)$$

qui exprime que quand deux événements sont indépendants, observer l'un *et* l'autre donne la somme des informations qu'on obtient en observant l'un ou l'autre séparément.

Comme la probabilité de deux événements indépendants est le produit des probabilités, la fonction φ doit vérifier $\varphi(pq) = \varphi(p) + \varphi(q)$. Cette condition est satisfaite si on prend $\varphi(p) = -\log_b(p)$ où b est un nombre positif à déterminer, c'est à dire que l'on prend $I(E) = -\log_b(P(E))$. C'est même le seul choix possible si l'on impose que φ soit une fonction continue.

On pourrait choisir b comme on le souhaite, mais aujourd'hui tout le monde est d'accord pour prendre $b = 2$:

$\log(x)$ est le logarithme népérien, ou naturel, de x, défini pour $x > 0$ par $\log(x) \stackrel{\text{def}}{=} \int_1^x \frac{dt}{t}$, c'est à dire que la dérivée de $\log(x)$ est $1/x$ et $\log(1) = 0$. log est une fonction dérivable, donc continue, et est une application $(0, +\infty) \to \mathbb{R}$, qui satisfait $\log(xy) = \log(x) + \log(y)$.

Le nombre e est défini par $\log(e) = 1$. Pour $b > 0$, le logarithme à base b est $\log_b(x) \stackrel{\text{def}}{=} \frac{\log(x)}{\log(b)}$; il vérifie $\log_b(xy) = \log_b(x) + \log_b(y)$ et $\log_b(b) = 1$. On utilise fréquemment le logarithme décimal ($b = 10$) et le logarithme binaire ($b = 2$).

On peut montrer que les fonctions logarithmes à base b sont les seules fonctions $f : (0, +\infty) \to \mathbb{R}$ qui soient continues et satisfassent $f(xy) = f(x) + f(y)$.

Notons que $\log_b(1) = 0$ et $\log_b(1/x) = -\log_b(x)$.

x	$\log_2(x)$
0	$-\infty$
$2^{-10} \approx 10^{-3}$	-10
0.25	-2
0.5	-1
1	0
2	1
4	2
10	3.32219
256	8
1024	10
1 048 576	20

Table 1.1: Quelques valeurs de $\log_2(x)$.

Définition 1.1 *Soit $(\mathcal{A}, p)$ une source et E un sous-ensemble de $\mathcal{A}$. L'information de l'événement E est $I(E) \stackrel{\text{def}}{=} -\log_2(P(E))$*

L'unité d'information est le *bit*, parfois aussi appelé *shannon*.

Claude E. Shannon (1916 – 2001) a inventé en 1948 la théorie de l'information que nous étudions ici.

C.E. Shannon. The mathematical theory of communication. *Bell Syst. Tech. J*, 27:379–423, 1948

Exemple 1.1 *Anne et Bernard jouent aux échecs et tirent au sort le joueur qui aura les blancs. C'est Anne qui est choisie. L'information reçue peut être modélisée par une source semblable au Pile ou Face de l'Exemple 0.1, et l'information reçue est $-log_2(0.5) = 1$ bit.*

Exemple 1.2 (Le Vélo d'Anne, suite) *Bernard demande à Anne si le numéro de son cadenas est 6987, et Anne répond non. L'information reçue par Bernard est $log_2(0.9999) \approx 1.4 \cdot 10^{-4}$ bit, c'est à dire presque rien. Si Bernard avait vu juste, l'information reçue aurait été $-log_2\left(10^{-4}\right) \approx 13.3$ bits. Supposons qu'au lieu de deviner un numéro au hasard, Bernard pose la question "Le numéro est-il inférieur à 5000 ?". La réponse sera oui avec probabilité 0.5 et non avec la même probabilité. Quelle que soit la réponse, Bernard recevra 1 bit d'information.*

Le choix de $b = 2$ dans la définition de l'information vient du désir d'obtenir une quantité d'information égale à 1 bit dans l'exemple 1.1.

1.2 Entropie d'une Source

L'équipe d'Alan Turing qui travaillait à décrypter la machine allemande Enigma pendant la deuxième guerre mondiale utilisait le *déciban*, qui est le dixième de l'unité d'information obtenue en prenant $b = 10$. On utilise parfois aussi le *nat*, qui correspond à $b = e$.
Q. 6. Combien de bits vaut un déciban ?

Nous avons introduit la notion d'information d'un événement, mais nous allons voir maintenant que cela ne suffit pas et que la notion centrale est celle d'*entropie*. Pour voir pourquoi l'information ne suffit pas, considérons de nouveau le début de l'Exemple 1.2. Savoir que Bernard reçoit 13.3 bits d'information s'il a vu juste ne donne pas une bonne mesure, car la probabilité de cet événement est très faible (10^{-4}). En pratique, on est presque sûr de tomber dans l'autre cas, c'est à dire que Bernard n'a pas vu juste et reçoit $1.4 \cdot 10^{-4}$ bit d'information. C'est pourquoi on utilise l'information *moyenne*, appelée entropie :

Définition 1.2 *Soit $S = (\mathcal{A}, p)$ une source. L'entropie de S est*

$$H(S) \stackrel{\text{def}}{=} -\sum_{s \in \mathcal{A}} p(s) \log_2(p(s)).$$

Dans cette définition, si $p(s) = 0$ pour un certain s, on remplace par convention le terme $p(s)\log_2(p(s))$ de la somme par 0.

L'unité d'entropie est la même que l'unité d 'information, à savoir, le *bit* ou *shannon*.

Exemple 1.3 (Suite de l'Exemple 1.2) *Soit S la source qui modélise la réponse à la question de Bernard : "ton numéro est-il 6987 ?". L'entropie de S est $H(S) = 0.0001 \cdot 13.3 + 0.99991.4 \cdot 10^4 = 0.0015$ bit.*

Si au lieu de cela Bernard pose la question "ton numéro est-il inférieur à 5000 ?", l'entropie de la réponse est $H(S) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$ bit.

On peut généraliser l'exemple précédent : l'entropie d'une *source binaire*, c'est à dire qui émet deux symboles, disons 0 et 1, avec les probabilités q et $(1-q)$, vaut :

$$h(q) \stackrel{\text{def}}{=} -q\log_2(q) - (1-q)\log_2(1-q) \qquad (1.1)$$

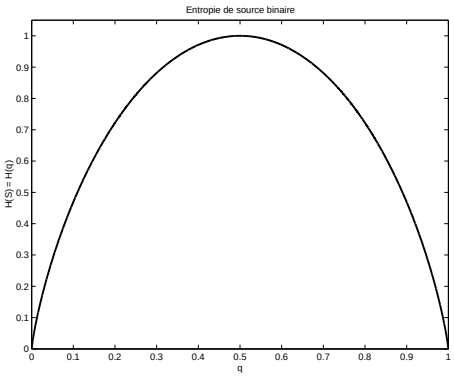


FIGURE 1.1: La fonction $q \mapsto h(q)$, donnant l'entropie d'une source binaire.

voir Figure 1.1.

Q. 7. Pour quelle valeur de q l'entropie $h(q)$ est-elle maximum ? minimum ?

Q. 8. Comparer $h(q)$ et $h(1-q)$

1.3 *Propriétés de l'Entropie*

La propriété suivante exprime que les seules sources d'entropie nulle (qui n'apportent aucune information) sont les sources certaines. Elle dérive immédiatement de la définition de l'entropie, ce qui ne l'empêche pas d'être importante en pratique.

Théorème 1.1 *1.* $H(S) \geq 0$

2. *Si pour un certain* $s \in \mathcal{A}$, $p(s) = 1$ *(i.e. la source émet le symbole* s *avec certitude, et donc* $p(s') = 0$ *pour* $s' \neq s$*), alors* $H(S) = 0$.
3. *Réciproquement, si* $H(S) = 0$, *alors il existe un* $s \in \mathcal{A}$ *tel que* $p(s) = 1$ *et* $p(s') = 0$ *pour tous les autres symboles* $s' \neq s$.

Preuve : (1) $H(S)$ est une somme de termes ≥ 0 donc est ≥ 0.
(2) Si $p(s) = 1$ alors il suffit d'appliquer la définition et on trouve $H(S) = 0$.
(3) Nous supposons donc que l'entropie de S vaut 0. L'entropie est une somme de S termes, qui sont tous ≥ 0. Donc si la somme est nulle, c'est que chaque terme est nul. Si $p(s) \in (0,1)$ alors $-p(s)\log_2(p(s)) > 0$, ce qui n'est pas possible. Donc $p(s) = 0$ ou 1 pour tout $s \in \mathcal{A}$. Comme la somme des probabilités vaut 1, il faut qu'exactement un des $p(s)$ soit égal à 1. □

Dans l'Exemple de la source binaire illustrée en Figure 1.1, nous voyons que l'entropie est maximum quand les symboles de source sont équiprobables. Cette propriété est importante et est vraie en général ; avant de voir pourquoi, nous avons besoin de la propriété suivante de la fonction log :

Théorème 1.2 (*Concavité de* log**, Inégalité de** *Jensen***)** *Le logarithme d'une moyenne est supérieur ou égal à la moyenne des logarithmes.*

Plus précisément, soient M *nombres* $x_i > 0$ *et* M *coefficients* $\alpha_i \geq 0$ *avec* $\alpha_1 + ... + \alpha_M = 1$ *; alors*

$$\log_2(\alpha_1 x_1 + ... + \alpha_M x_M) \geq \alpha_1 \log_2(x_1) + ... + \alpha_M \log_2(x_M) \qquad (1.2)$$

S'il y a égalité dans Eq.(1.2) et $\alpha_i > 0$ *pour tout* i, *alors tous les* x_i *sont égaux.*

Nous ne démontrons pas ce théorème, mais signalons simplement qu'il est dû au fait que la fonction $\log_2$ est concave, c'est à dire que le graphe de la fonction est au-dessus des cordes (Figure 1.2). Nous sommes maintenant en mesure de montrer l'inégalité principale de l'entropie :

Théorème 1.3 *Soit* S *une source avec un alphabet de* M *symboles.*

1. $H(S) \leq \log_2(M)$
2. *Si les* M *symboles de la source sont équiprobables, alors* $H(S) = \log_2(M)$
3. *Si* $H(S) = \log_2(M)$ *alors les* M *symboles de la source sont équiprobables.*

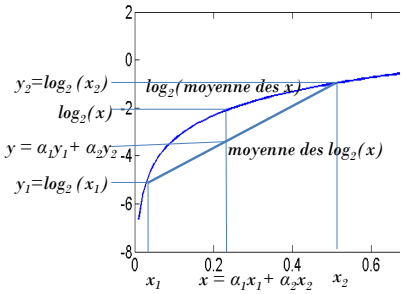


FIGURE 1.2: Tout x compris entre x_1 et x_2 peut se mettre sous la forme $x = \alpha_1 x_1 + \alpha_2 x_2$ avec $\alpha_1 \in [0,1]$ et $\alpha_2 = 1 - \alpha_1$. Soit $y_1 = \log_2(x_1)$ et $y_2 = \log_2(x_2)$; $y = \alpha_1 y_1 + \alpha_2 y_2$ est l'ordonnée du point de la corde d'abscisse x. L'inégalité (1.2) exprime que $\log_2(x) \geq y$, ce qui signifie que la corde est au-dessous du graphe.

Une fonction qui possède la propriété que son graphe est au-dessus de ses cordes est dite *concave*. Si une fonction définie sur un intervalle est deux fois dérivable et sa dérivée seconde est négative, elle est concave. C'est le cas de la fonction $\log_2$ puisque sa dérivée seconde est $\frac{-1}{\log_2 x^2}$.

L'inégalité de Jensen (1.2) est vraie si l'on remplace $\log_2$ par une fonction concave quelconque.

Une fonction qui possède la propriété que son graphe est au-dessous de ses cordes est dite *convexe*. Si une fonction définie sur un intervalle est deux fois dérivable et sa dérivée seconde est positive, elle est convexe. L'inégalité de Jensen (1.2) est vraie dans l'autre sens si l'on remplace $\log_2$ par une fonction convexe. La fonction $x \mapsto x^2$ est convexe, donc le carré d'une moyenne est inférieur ou égal à la moyennes des carrés.

☺*Preuve :* (1) Soit une source quelconque avec M symboles.
(1a) Supposons d'abord que $p(s) > 0$ pour tout $s \in \mathcal{A}$. Utilisons la notation $\mathcal{A} = \{s_1, ..., s_M\}$. L'entropie de la source est

$$\begin{aligned} H(S) &= -p(s_1)\log_2(p(s_1)) - ... - p(s_M)\log_2(p(s_M)) \\ &= p(s_1)\log_2\left(\frac{1}{p(s_1)}\right) + ... + p(s_M)\log_2\left(\frac{1}{p(s_M)}\right) \end{aligned}$$

Appliquons l'inégalité (1.2) à $\alpha_i = p(s_i)$ et $x_i = \frac{1}{p(s_i)}$, et obtenons

$$\begin{aligned} \log_2(1 + ... + 1) &\geq H(S) \\ \log_2(M) &\geq H(S) \end{aligned}$$

(1b) Il reste à montrer l'inégalité quand $p(s_i) = 0$ pour certains i. Soit S' la source obtenue en supprimant de S les symboles qui ont une densité de probabilité nulle. Nous avons $H(S') = H(S)$ d'après la définition de l'entropie H. La source S' possède $M' < M$ symboles et tous les symboles de S' ont une densité de probabilité non nulle. Nous pouvons appliquer (1a) à S' et

$$H(S) = H(S') \leq \log_2(M') < \log_2(M)$$

(2) Pour une source dont tous les M symboles sont équiprobables on a :

$$\begin{aligned} H(S) &= -\frac{1}{M}\log_2(1/M) - ... - \frac{1}{M}\log_2(1/M) \\ &= \frac{1}{M}\log_2(M) + ... + \frac{1}{M}\log_2(M) = M\frac{1}{M}\log_2(M) = \log_2(M) \end{aligned}$$

(3) Nous supposons que $H(S) = \log_2(M)$. Il n'est pas possible que $p(s_i) = 0$ pour certains i car nous avons vu en (1b) que dans un tel cas $H(S) < \log_2(M)$. Appliquons le Théorème 1.2 à $\alpha_i = p(s_i)$ et $x_i = \frac{1}{p(s_i)}$. Nous avons égalité dans l'inégalité de Jensen, et $\alpha_i > 0$ donc tous les x_i sont égaux, c'est à dire que les M symboles sont équiprobables. ☺□

Exemple 1.4 (Entropie d'un robot-page francophone) *Les fréquences d'apparition des caractères du Français ont été calculées*[1] *et on trouve les valeurs données dans la table 1.2. L'entropie calculée à partir de cette table est* 3.95 *bits : comparez à l'entropie maximale pour une source de 26 symboles, qui est* $\log_2(26) \approx 4.70$.

A quoi correspond cette entropie ? A celle d'une source qui tire au sort une *lettre selon la répartition de la langue française. C'est peut-être le cas d'un "robot-page", une machine qui ouvre un livre à une page au hasard et lit une lettre, tirée au sort dans la page. Les textes français ne sont pas (sauf exception) produits par des robots-pages, et nous verrons en Section 5.1 que l'entropie* par caractère *du Français est bien plus faible que* 3.95 *bits.*

1. G. Michaud-Brière, Y. Pearson, S. Perreault, and L.-O. Roof. La cryptographie. `http://nomis80.org/cryptographie/cryptographie.html`, 2002

lettre	fréquence
A	8,11
B	0,81
C	3,38
D	4,28
E	17,69
F	1,13
G	1,19
H	0,74
I	7,24
J	0,18
K	0,02
L	5,99
M	2,29
N	7,68
O	5,20
P	2,92
Q	0,83
R	6,43
S	8,87
T	7,44
U	5,23
V	1,28
W	0,06
X	0,53
Y	0,26
Z	0,12

TABLE 1.2: Fréquences des lettres du Français, exprimées en pourcentages.

1.4 Entropie d'une Source Composée

Soit S une source composée avec deux marginales $S = (S_1, S_2)$. Nous pouvons calculer l'entropie de la source et de ses marginales ; faisons-le pour deux exemples.

Exemple 1.5 (Somme de Deux Dés Codée Sur Deux Chiffres, suite) *La densité de probabilité de $L = (L_1, L_2)$ est donnée dans la Table 1 (page 11), d'où on trouve :*

$$H(L) = 3.27 \text{ bits}$$

La densité marginale du premier chiffre L_1 est donnée dans la dernière ligne de la Table 1 (page 11), d'où :

$$H(L_1) = -5/6 \log_2(5/6) - 1/6 \log_2(1/6) = 0.65 \text{ bit}$$

La densité marginale du deuxième chiffre L_2 est donnée dans la dernière colonne de la Table 1 (page 11), on obtient :

$$H(L_2) = 3.22 \text{ bits}$$

Notons que

$$H(L) < H(L_1) + H(L_2)$$

c'est à dire que l'information moyenne donnée par L est moindre que la somme des informations données par L_1 et L_2. Il y a une certaine redondance entre L_1 et L_2 : une partie de l'information contenue dans L_2 est déjà contenue dans L_1 ; par exemple, quand le premier chiffre L_1 vaut 0 on sait que le deuxième chiffre L_2 ne peut valoir que 0, 1 ou 2.

Exemple 1.6 (Deux Dés non Pipés, suite) *S_1 est le résultat du tirage d'un premier dé non pipé, S_2 du deuxième. On a $P_{S_1,S_2}(i,j) = 1/36$, c'est à dire que tous les symboles sont équiprobables donc*

$$H(S_1, S_2) = \log_2(36) = 2\log_2(6) = 5.170$$

On a aussi $H(S_1) = H(S_2) = \log_2(6) = 2.585$. Ici $H(S_1, S_2) = H(S_1) + H(S_2)$. Il n'y a pas de redondance entre S_1 et S_2. Cela est naturel puisque nous savons que S_1 et S_2 sont indépendantes (Exemple 0.9).

Dans les exemples précédents, nous avons vu que $H(S_1, S_2) \leq H(S_1) + H(S_2)$, avec égalité quand S_1 et S_2 sont indépendantes. C'est un fait général :

Théorème 1.4 *Soit $S = (S_1, ..., S_n)$ une source composée.*

1. $H(S_1, ..., S_n) \leq H(S_1) + ... + H(S_n)$
2. $H(S_1, ..., S_n) = H(S_1) + ... + H(S_n)$ *si et seulement si les n sources marginales $S_1, ..., S_n$ sont indépendantes.*

Preuve : Nous faisons la preuve seulement pour $n = 2$. Pour simplifier la notation, supposons que les alphabets de S_1 et S_2 sont $\mathcal{A}_1 = \{1, 2, ..., I\}$ et $\mathcal{A}_2 = \{1, 2, ..., J\}$. Supposons aussi pour simplifier que $p(i,j) > 0$ pour tous (i,j).

(1) Soit $j \in \mathcal{A}_2$ fixé ; appliquons l'inégalité de Jensen (1.2) avec $\alpha_i = \frac{p(i,j)}{p_{S_2}(j)}$ et $x_i = \frac{p_{S_1}(i)}{p(i,j)}$:

$$x \overset{\text{def}}{=} \sum_{i=1}^{I} \alpha_i x_i = \sum_{i=1}^{I} \frac{p_{S_1}(i)}{p_{S_2}(j)} = \frac{1}{p_{S_2}(j)} \overbrace{\sum_{i=1}^{I} p_{S_1}(i)}^{1} = \frac{1}{p_{S_2}(j)}$$

et on a bien $\sum_{i=1}^{I} \alpha_i = 1$ donc l'inégalité de Jensen donne :

$$\begin{aligned} \log_2(x) = \log_2(1/p_{S_2}(j)) &\geq \sum_{i=1}^{I} \frac{p(i,j)}{p_{S_2}(j)} \log_2\left(\frac{p_{S_1}(i)}{p(i,j)}\right) \\ &= \frac{1}{p_{S_2}(j)} \sum_{i=1}^{I} p(i,j) \left[\log_2\left(p_{S_1}(i)\right) - \log_2\left(p(i,j)\right)\right] \end{aligned} \tag{1.3}$$

donc, en multipliant par $p_{S_2}(j)$ et en sommant sur tous les j :

$$\begin{aligned} \overbrace{\sum_{j=1}^{J} p_{S_2}(j) \log_2(1/p_{S_2}(j))}^{H(S_2)} &\geq \sum_{j=1}^{J} \sum_{i=1}^{I} p(i,j) \left[\log_2\left(p_{S_1}(i)\right) - \log_2\left(p(i,j)\right)\right] \\ &= \sum_{j=1}^{J} \sum_{i=1}^{I} p(i,j) \log_2\left(p_{S_1}(i)\right) - \sum_{j=1}^{J} \sum_{i=1}^{I} p(i,j) \log_2\left(p(i,j)\right) \\ &= \sum_{i=1}^{I} \left[\sum_{j=1}^{J} p(i,j) \log_2\left(p_{S_1}(i)\right)\right] + H(S_1, S_2) \\ &= \sum_{i=1}^{I} \log_2\left(p_{S_1}(i)\right) \left[\sum_{j=1}^{J} p(i,j)\right] + H(S_1, S_2) \\ &= \sum_{i=1}^{I} \log_2\left(p_{S_1}(i)\right) \left[p_{S_1}(i)\right] + H(S_1, S_2) \\ &= -H(S_1) + H(S_1, S_2) \end{aligned}$$

Ce sont l'*associativité* et la *commutativité* de l'addition des nombres réels qui permettent de permuter l'ordre des indices dans et d'écrire

$$\sum_{j=1}^{J} \sum_{i=1}^{I} x(i,j) = \sum_{j=1}^{J} \left[\sum_{i=1}^{I} x(i,j)\right] = \sum_{i=1}^{I} \left[\sum_{j=1}^{J} x(i,j)\right]$$

ce qui montre l'inégalité demandée.

(2)(a) Indépendance ⇒ Egalité.

Si S_1 et S_2 sont indépendants, alors, pour j fixé tous les x_i sont égaux, donc on a égalité partout dans ce qui précède.

(2)(b) Egalité ⇒ indépendance.

Si on a égalité dans ce qui précède, comme l'inégalité est obtenue en sommant J inégalités (1.3), il faut qu'il y ait égalité (1.3) pour tout j. Par la stricte concavité de la fonction $\log_2$, cela implique que tous les x_i sont égaux, car $\alpha_i > 0$. Or $x_i = 1/p_{S_2|S_1}(j|i)$. Donc $p_{S_2|S_1}(j|i)$ est le même pour tous les i. D'après le Théorème 0.1, cela implique que S_1 et S_2 sont indépendants. □

Q. 9. On tire un dé n fois de suite et on envoie un message S contenant la suite des résultats obtenus. Quelle est l'entropie de S ?

2
Codage de Source

Après avoir mis en place les concepts théoriques d'entropie et de source, nous pouvons passer au thème principal de ce module, le *codage de source*. On appelle ainsi l'opération qui traduit les symboles d'une source en des symboles utilisables par une machine, à des fins de transmission ou de stockage. Pourquoi traduire les symboles de source ? Une première raison immédiate est d'adapter l'alphabet de la source à celui de la machine : un ordinateur ne comprend pas les lettres de l'alphabet, mais des suites de 0 et 1. Une autre raison est l'efficacité : nous voulons *comprimer* autant que possible la source et prendre le moins de place possible sur le disque ou en transmission, ceci sans aucune altération (compression sans perte). Nous reviendrons en Section 3 sur ce que nous entendons par efficacité. Pour l'instant, nous allons étudier ce qu'est un code de source.

Le Code ASCII est un code de source. Il traduit les caractères alphanumériques des langues occidentales en suites de 8 bits. Par exemple, le caractère 'A´ est représenté par la suite 01000001.

Notons qu'il y a différentes sortes de codage, en plus du codage de source qui vise à comprimer les messages : le *codage détecteur et correcteur d'erreurs*, au contraire du précédent, augmente la longueur et la redondance des messages pour permettre la détection et éventuellement la correction à la réception des erreurs provoquées par un canal perturbé par le bruit. Nous l'étudierons en détail dans un autre module. Il y a aussi le *codage d'émission* ou *de ligne* qui vise une adaptation technique des signaux à celles du canal (bande passante, distorsion linéaire, etc.).

2.1 *Définition d'un Code de Source*

Nous avons, comme précédemment, une source S d'alphabet $\mathcal{A} = \{s_1, ..., s_M\}$. Nous avons un deuxième alphabet, l'*alphabet du code* $\mathcal{D}$, qui est un ensemble de D symboles de code. Le plus souvent $D = 2$ et alors $\mathcal{D} = \{0, 1\}$. Les éléments de $\mathcal{D}$ sont les *symboles de code*. Un *dictionnaire* $\mathcal{C}$ (en Anglais *codebook*) est un sous-ensemble fini de suites finies d'éléments construites avec l'alphabet $\mathcal{D}$. Un élément de $\mathcal{C}$ est appelé *mot de code*.

Une suite de n éléments de $\mathcal{A}$ est notée le plus souvent sous la forme : (s_1, s_2, s_3, s_4) (par exemple pour $n = 4$). On utilise aussi la notation $s_1s_2s_3s_4$ (qui ne veut alors pas dire un produit), quand le contexte est clair. Ainsi les notations 01000001 et $(0, 1, 0, 0, 0, 0, 0, 1)$ sont synonymes.

Quand $D = 2$ on dit qu'on a un code *binaire*.

Pour le code ASCII, l'alphabet du code est $\{0, 1\}$, le dictionnaire est constitué des 256 suites de 8 chiffres binaires ; 01000001 est un mot de code.

Définition 2.1 *(Code de Source) Un code de source, ou encodage,* Γ*, est une application* bijective $\Gamma : \mathcal{A} \to \mathcal{C}$.

Le code Γ permet donc de traduire tout symbole de la source en un mot de code, de façon que pour chaque symbole il existe un mot de code, et inversement, à chaque mot de code correspond un symbole unique de la source (c'est la définition d'une application bijective). Puisqu'une application bijective ne peut exister qu'entre ensembles ayant mêmes nombres d'éléments, le dictionnaire $\mathcal{C}$ comporte exactement S mots de code, comme l'alphabet de la source.

La *longueur* d'un mot $c \in \mathcal{C}$ est $\ell(c) =$ le nombre de symboles de code de c (ainsi si $c = x_1x_2...x_k$ alors $\ell(c) = k$). On dit qu'un code est à *longueur constante* si tous les mots de code ont la même longueur, dans le cas contraire on dit que le code est à *longueur variable*.

Exemple 2.1 (Quatre Petits Codes) *La Table 2.1 donne quatre exemples de codes binaires. L'alphabet de chacun des codes est* $\{0,1\}$. *Le dictionnaire du code O est* $\{00,01,10,11\}$, *celui du code A est* $\{0,01,10,11\}$. *Le code O est à longueur fixe, les codes A, B et C sont à longueur variable.*

Puisqu'un code est une application bijective, elle peut être inversée ; l'application inverse, $\Gamma^{-1} : \mathcal{C} \to \mathcal{A}$ qui consiste à traduire *un* mot de code en *un* symbole de la source, est appelé *décodage*.

2.2 *Représentation d'un Code par son Arbre Complet*

Pour raisonner sur les codes, il est utile d'utiliser la représentation par *arbre complet*, plutôt que par une table comme plus haut. La Figure 2.1 donne les arbres complets des exemples que nous avons vu précédemment.

L'arbre complet d'un code est construit de la façon suivante. Un tel arbre débute par une racine, donnant naissance à D branches (où D est le nombre de symboles de code). Chaque branche se termine en un noeud, et est étiquetée avec un des D symboles de l'alphabet du code. Chaque noeud de cette première génération est à son tour ramifié en D branches et ainsi de suite. On construit un tel arbre jusqu'à une profondeur égale à $L_{\max}$, la longueur maximale des mots du code.On obtient ainsi un arbre dont chaque noeud représente une suite d'au plus $L_{\max}$ symboles de code. Certains des noeuds sont dans le dictionnaire du code, on les étiquette par le mot de code correspondant. L'arbre ainsi étiqueté est appelé *l'arbre complet du code*. L'arbre est dit *complet* parce qu'on y met tous les $D^{L_{\max}}$ noeuds possibles, qu'ils soient utiliés par le code ou non. Sur l'arbre complet d'un code, les mots de codes sont placés à une profondeur égale à leur longueur.

On dit que l'arbre obtenu est un arbre D-aire, c'est à dire qu'à chaque noeud de l'arbre il y a D branches. Sur la Figure 2.1, $D = 2$ et nous avons des arbres binaires.

Si f est une application d'un ensemble de départ E vers un ensemble d'arrivée F, tout élément x de l'ensemble de départ E a une image unique, notée $f(x)$. Pour y dans l'ensemble d'arrivée F on dit que $x \in E$ est un *antécédent* de y si $y = f(x)$. En général, un $y \in F$ peut avoir 0, 1 ou plusieurs antécédents.

Si tout $y \in F$ possède exactement un et un seul antécédent, on dit que l'application f est *bijective*, ou encore que f est une *bijection*. Cela équivaut à dire que pour tout $y \in F$ l'équation $y = f(x)$, où l'inconnue est $x \in E$, a une solution unique.

Si E et F sont des ensembles finis et s'il existe une bijection de E vers F alors $\text{card}(E) = \text{card}(F)$. Par exemple, le dictionnaire du code $\mathcal{C}$ et l'alphabet de la source $\mathcal{A}$ ont le même nombre d'éléments.

Si f est une application bijective, elle peut être inversée, et l'application inverse est notée f^{-1}. Par exemple,

$$\begin{array}{rcl} f : (0,+\infty) & \to & \mathbb{R} \\ x & \mapsto & \log_2(x) \end{array}$$

est bijective car l'équation en x : $y = \log_2(x)$ a une solution unique égale à $x = 2^y$; l'application inverse est définie par la formule $y \mapsto 2^y$, ce qu'on peut aussi écrire :

$$\begin{array}{rcl} f^{-1} : \mathbb{R} & \to & (0,+\infty) \\ x & \mapsto & 2^x \end{array}$$

puisque x comme y est une variable muette.

Q. 10. L'application $\mathbb{R} \to \mathbb{R}$, $x \mapsto x^2$ est-elle bijective ?

Q. 11. Montrer que si le code Γ est à longueur constante L, alors $M \leq D^L$.

code	O	A	B	C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

TABLE 2.1: Quatre codes binaires. L'alphabet de la source est $\{a,b,c,d\}$.

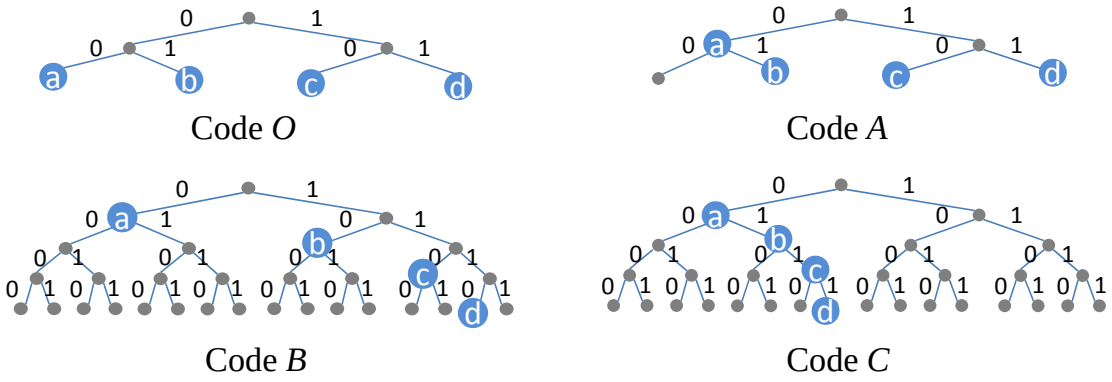


FIGURE 2.1: Les arbres complets des codes O, A, B,C.

2.3 *Décodage Unique*

Supposons que l'on nous donne *un* mot de code, reçu sans erreur (nous traiterons le cas des erreurs dans un autre module), par exemple 01 alors que le code est le code A de l'exemple précédent. Le décodage est toujours possible, par définition d'un code, et nous pouvons conclure que le symbole émis est b. Supposons maintenant que l'on vous donne une *suite* de mots de code, par exemple 0110, reçue sans erreur, et que vous souhaitiez décoder cette suite. Si l'on vous donne un moyen de trouver les frontières des mots de code, par exemple comme dans 01 10 cela ne pose pas de difficulté, il suffit d'appliquer l'opération de décodage Γ^{-1} à chacun des mots reçus pour obtenir que la suite de symboles de la source est bc. Mais en faisant une telle hypothèse, on a supposé qu'il existe un symbole de code spécial, le délimiteur (ici l'espace). En pratique, dans les ordinateurs, le délimiteur n'existe pas, il faut utiliser le code lui-même pour savoir où s'arrêtent les mots.

Nous considérons donc dans la suite que nous recevons une suite de mots de code, mis bout à bout. Si nous recevons 0110 et que le code est O, nous pouvons dire que le message de la source est bc. Par contre si le code est A, il y a une ambiguïté : les deux messages bc et ada sont possibles. Cela montre que notre définition de code n'est pas suffisante, il nous faut introduire le concept suivant.

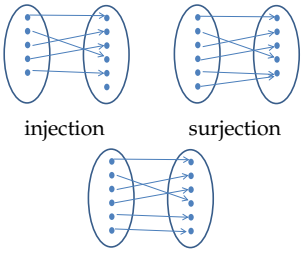
Définition 2.2 *Soit Γ un code de source. Γ est à décodage unique si pour toute suite de symboles de code, qui résulte de l'encodage d'une suite de symboles de la source, il existe un décodage unique.*

Une autre façon, plus formelle, de définir le décodage unique, est la suivante. Soient $\mathcal{A}^*$ [resp. $\mathcal{D}^*$] l'ensemble de toutes les suites finies d'au moins un élément de $\mathcal{A}$ [resp. $\mathcal{D}$]. $\mathcal{A}^*$ est l'ensemble des "mots" que l'on peut écrire avec l'alphabet de la source. Ainsi, avec le code A de la Table 2.1, la suite de symboles bc est un élément de $\mathcal{A}^*$ et la suite 0110 de $\mathcal{D}^*$. On peut étendre par concaténation l'application d'encodage Γ définie sur $\mathcal{A}$ en une application d'encodage Γ^* définie sur $\mathcal{A}^*$, c'est à dire que si $s_1...s_n \in \mathcal{A}^*$ est une suite de symboles de la source, $\Gamma^*(s_1...s_n)$ est la suite des symboles de code

Une application $f : E \to F$ est *injective* (on dit aussi que f est une *injection* ; Ang.*one to one*) si tout élément de l'ensemble d'arrivée F possède 0 ou 1 antécédent. Cela équivaut à dire que pour tout $y \in F$ l'équation $y = f(x)$, où l'inconnue est $x \in E$, a au plus une solution.

Une application $f : E \to F$ est *surjective* (on dit aussi que f est une *surjection* ; Ang.*onto*) si tout élément de l'ensemble d'arrivée F possède au moins 1 antécédent. Cela équivaut à dire que pour tout $y \in F$ l'équation $y = f(x)$, où l'inconnue est $x \in E$, a au moins une solution.

Une application est bijective si et seulement si elle est injective et surjective.



Q. 12. Les applications $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto x^2$; $g : \mathbb{R} \to [0, +\infty)$, $x \mapsto x^2$ et $h : [0, +\infty) \to [0, +\infty)$, $x \mapsto x^2$ sont-elles injectives, surjectives, bijectives ?

de $\Gamma(s_1), ..., \Gamma(s_n)$ mis bout à bout. Ainsi $\Gamma(b) = 01$, $\Gamma(c) = 10$ et $\Gamma^*(bc) = 0110$. On dit alors que le code Γ est à décodage unique si l'application Γ^* est *injective*.

Par exemple, avec $\Gamma = A$ on a $\Gamma^*(bc) = 0110$ et $\Gamma^*(ada) = 0110$; l'élément 0110 de $\mathcal{D}^*$ possède deux antécédents distincts bc et ada, donc Γ^* n'est pas injective et le code n'est pas à décodage unique.

Le code O est à décodage unique. En fait, tout code de longueur constante, disons L, est à décodage unique : par hypothèse Γ est bijectif donc toute suite de symboles de code *reçue sans erreur* peut être décodée de manière unique, en découpant les symboles de code par paquets de L symboles. Par exemple, si on reçoit la séquence ASCII 01001000011001010110110001101100011011111, on peut la décoder comme suit :

1. découper en quatre blocs de 8 bits : 01001000, 01100101, 01101100, 01101100 et 01101111
2. Chercher chaque mot de code dans la table du code et obtenir : *Hello*

Il est facile de comprendre intuitivement pourquoi les codes à longueur constante peuvent être inefficaces : Imaginons que nous devons encoder un texte français avec un code binaire de longueur constante ; même en ignorant les accents, les majuscules, les espaces et les signes de ponctuation (ce qui est très laid), il faut 26 caractères, donc un code de longueur constante doit avoir au moins 5 symboles binaires (aussi appelés bits). Or le caractère 'E' est très fréquent (plus d'un caractère sur six) alors que 'W' est très rare (moins d'un caractère sur 1600) ; il serait donc plus efficace de coder 'E' avec une séquence courte, peut-être 1 ou2 bits, et 'W' avec une séquence plus longue, peut-être 7 ou 8 bits.

On pourrait se demander pourquoi ne pas se limiter aux codes à longueur constante, puisque leur décodage est plus simple. Nous verrons plus loin que, en général, le code optimal (qui assure la compression maximale) est à longueur variable, ce qui est une bonne raison pour considérer des codes à longueur variable.

Le code B est à décodage unique car le symbole 0 marque la fin des mots de code, donc on peut analyser toute suite de symboles reçus en groupant les symboles jusqu'à trouver un 0. Par exemple, la suite 0110 est décodable en ac et il n'y a pas d'autre possibilité. De la même façon, le code C est à décodage unique car le symbole 0 marque le début des mots. La suite reçue 0110 est décodable en ca et il n'y a pas d'autre possibilité.

"la suite 0110 est décodable en ac" peut se dire aussi : "0110 a pour antécédent ca" par l'application Γ^* qui à une suite de symboles de la source associe la suite des mots de code.

Nous nous intéressons à la compression sans perte, aussi seuls les codes à décodage unique sont intéressants. En effet, pour pouvoir obtenir exactement les symboles de la source à partir d'une suite de symboles de code, il faut que le code soit à décodage unique.

2.4 *Code Instantané, Code Sans Préfixe*

Parmi tous les codes possibles, il en est de plus faciles à manipuler, ce sont les codes instantanés.

Définition 2.3 *Nous disons qu'un code est* instantané

1. *s'il est à décodage unique,*
2. *et si, à mesure que les séquences de symboles de l'alphabet du code sont reçus, les mots du code peuvent être déterminés sans s'inquiéter des symboles de code suivants.*

Q. 13. Un code à longueur constante est-il instantané ?

Par exemple, le code B défini ci-dessus est instantané, au contraire du code C. En effet, supposons que la séquence reçue soit 00110. Avec le code B, dès la réception du premier symbole 0, le récepteur

sait qu'il s'agit de a. Avec le code C par contre, aucune conclusion ne peut être tirée dès la réception du premier symbole. Il faut attendre le deuxième 0 pour décider que le premier représentait a.

Définition 2.4 *On dit qu'un mot de code $c = x_1x_2...x_k$ est préfixe d'un autre mot de code c' si on peut écrire $c' = x_1x_2x_kx_{k+1}...x_\ell$ pour un certain $\ell \geq k+1$. Sur l'arbre du code, cela veut dire que c' est un descendant de c. On dit que le code Γ est sans préfixe si aucun mot de code n'est préfixe d'un autre mot de code. Sur l'arbre du code, cela veut dire que aucun mot de code n'est descendant d'un autre mot de code.*

Q. 14. Un code de longueur constante est-il nécessairement sans préfixe ?

Nous voyons sur la Figure 2.1 que les codes O et B sont sans préfixe, mais que les codes A et C ne le sont pas. Par exemple, dans le code A, le mot de code 0 est préfixe du mot de code 01, donc A n'est pas sans préfixe.

Les codes sans préfixe sont toujours des "bons" codes :

Théorème 2.1 *Un code est sans préfixe si et seulement si il est instantané.*

Preuve : (1) ☺ instantané ⇒ sans préfixe. Nous montrons cette implication par contraposition : non (sans préfixe) ⇒ non instantané. Soit donc Γ un code qui n'est pas sans préfixe, c'est à dire qu'il est avec préfixe, ou encore, qu'il existe un mot de code $c = x_1x_2...x_k$ qui est préfixe d'un autre mot de code $c' = x_1x_2x_kx_{k+1}...x_\ell$. Supposons que nous ayons reçu les symboles de code $x_1x_2...x_k$. Nous ne pouvons pas décoder à cet instant, car il se pourrait très bien que le mot de code reçu soit c, ou le début de c'. Pour le savoir, il faut attendre d'avoir reçu les symboles de code $x_{k+1}...x_\ell$. Donc le code n'est pas instantané.☺
(2) sans préfixe ⇒ instantané. Nous avons deux choses à montrer.
(2a) sans préfixe ⇒ à décodage unique. Par contraposition : non (à décodage unique) ⇒ non (sans préfixe). Par hypothèse le code n'est pas à décodage unique. Nous allons d'abord montrer que le code vérifie la propriété (P) :

> (P) Il existe une suite de symboles de code que l'on peut décoder de deux façons différentes au moins, et dont les décodages diffèrent par le premier symbole de source.

Pour cela, remarquons que, par hypothèse, il existe une suite de symboles de code $X_1 = x_1x_2...x_n$, de longueur $n > 0$, que l'on peut décoder de deux façons au moins, soient $S = s_1s_2...s_{n_1}$ et $S' = s'_1s'_2...s'_{n_2}$. Si $s_1 \neq s'_1$ alors P est vraie et c'est fini. Sinon, soit $s_1...s_k$ le plus long préfixe commun à S et S' et soient $\tilde{S}$ et $\tilde{S}'$ les deux suites de symboles obtenues en supprimant de S et S' leurs préfixes communs. Nécessairement $\tilde{S}$ n'est pas vide, car sinon S est préfixe de S', et l'encodage de S' est plus long que celui de S, ce qui contredit l'hypothèse que S et S' ont le même encodage. Pour la même raison, $\tilde{S}'$ n'est pas vide. De plus, $\tilde{S}$ et $\tilde{S}'$ diffèrent par leur premier symbole.

Les implications "$A \Rightarrow B$" et "(non B) ⇒ (non A)" sont équivalentes. On dit que la deuxième est la *contraposée* de la première.

Par contre, "$A \Rightarrow B$" et "(non A) ⇒ (non B)" ne sont pas équivalentes.

L'implication "$B \Rightarrow A$" est appelée l'*implication réciproque* de "$A \Rightarrow B$". Donc "(non A) ⇒ (non B)" est équivalente à la réciproque de "$A \Rightarrow B$".

Il se peut qu'une implication soit vraie mais pas la réciproque, ou vice versa. Par contre, si une implication est vraie, la contraposée l'est aussi.

Si "$A \Rightarrow B$" et "$B \Rightarrow A$" sont vraies toutes les deux en même temps, on dit que A et B sont *équivalentes*. On écrit aussi A *si et seulement si* B, ou en abrégé A *ssi* B.

Q. 15. Soit n un entier et soit l'implication $(P1)$: (n est pair) ⇒ (n est divisible par 4). Quelles sont la contraposée, la réciproque, et la contraposée de la réciproque ? Lesquelles sont vraies pour tout entier n ?

Par exemple avec le code A la suite de symboles de code $X_1 = x_1x_2...x_n =$ 0110 peut être décodée de deux façons $s_1s_2...s_{n_1} = bc$ et $s'_1s'_2...s'_{n_2} = ada$. Les premiers symboles de source des décodages sont respectivement b et a, et sont différents.

Soit $x_1...x_\ell$ l'encodage du préfixe commun $s_1...s_k$. Supprimons de X_1 les ℓ premiers symboles de code, et soit X_2 la suite de symboles résultante. X_2 est l'encodage de $\tilde{S}$ et $\tilde{S}'$, qui diffèrent par leur premier symbole, donc nous avons montré (P).

Nous pouvons maintenant achever la preuve de (2a). Soit $x_1x_2...x_n$ une suite de symboles de code que l'on peut décoder de deux façons en $S = s_1s_2...s_{n_1}$ et $S' = s'_1s'_2...s'_{n_2}$ avec $s_1 \neq s'_1$. Soit $c = x_1...x_\ell$ le mot de code correspondant à s_1 et $c' = x_1...x_{\ell'}$ le mot de code correspondant à s'_1. Forcément, $c \neq c'$ car un code est bijectif et donc $\ell \neq \ell'$. Si $\ell < \ell'$ alors c est préfixe de c', sinon c' est préfixe de c. Donc le code n'est pas sans préfixe.
(2b) sans préfixe $\Rightarrow$ item 2 de la définition de code instantané. Supposons que nous ayons reçu une suite de symboles du code $x_1x_2...x_k$, qui est un mot de code c. Quand nous avons reçu le dernier symbole x_k, nous savons que le mot de code reçu est c, car il n'y a aucun autre mot de code commençant par $x_1x_2...x_k$, par hypothèse. Donc nous pouvons décider que le mot reçu est c, et le code est donc instantané. □

Exemple 2.2 (Quatre Petits Codes, suite) *Les codes O et B sont instantanés et sont donc à décodage unique. Le code C n'est pas instantané mais est à décodage unique. Le code A n'est pas à décodage unique et n'est pas instantané.*

Q. 16. Les implications suivantes sont-elles vraies ?

1. à décodage unique $\Rightarrow$ instantané
2. instantané $\Rightarrow$ à décodage unique
3. non (à décodage unique) $\Rightarrow$ non instantané
4. non instantané $\Rightarrow$ non (à décodage unique)

2.5 *Arbre de Décodage d'un Code Instantané*

Pour un code est instantané, il est très pratique d'utiliser l'*arbre de décodage*. Il est construit à partir de l'arbre complet du code en supprimant toutes les branches qui ne mènent pas à un mot de code. Comme le nom l'indique, l'arbre de décodage permet de décoder simplement, en utilisant par exemple l'algorithme décrit dans l'Algorithme de la Figure 2.3.

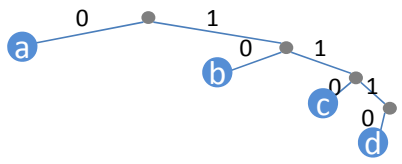


FIGURE 2.2: L'arbre de décodage du code instantané B.

Notons que l'arbre de décodage est défini seulement si le code est instantané. Pour un code à décodage unique non instantané tel que le code C de la Figure 2.1, il n'y en a pas (le décodage d'un tel code est plus complexe).

Q. 17. Quel est l'arbre de décodage du code O ?
Q. 18. Quels sont les codes dont l'arbre de décodage est égal à l'arbre complet du code ?

2.6 *Théorème de Kraft-McMillan*

Nous sommes intéressés à avoir des codes dont la longueur des mots de code soit aussi petite que possible. Cependant, on ne peut

```
1: aller à la racine de l'arbre de décodage
2: erreurDeDécodage ← false
3: while X ≠ ∅ and not erreurDeDécodage do
4:     x ← entête de X ; supprimer x de la tête de X
5:     descendre l'arbre en suivant la branche étiquetée par x
6:     if cela produit une erreur then
7:         erreurDeDécodage ←true
8:     else
9:         if le noeud courant possède une étiquette s then
10:            imprimer s ; aller à la racine de l'arbre
11:        end if
12:    end if
13: end while
```

FIGURE 2.3: Pseudo-code de l'algorithme de décodage d'un code instantané. Supposons que nous ayons reçu une suite X de symboles de code. L'algorithme imprime la suite des symboles de source reçus jusqu'à épuisement des symboles reçus, ou jusqu'à ce qu'une erreur ait lieu.

pas prendre trop de mots de code ayant une très faible longueur, comme l'exprime le théorème suivant.

Théorème 2.2 (Kraft-McMillan) *Soit Γ un code D-aire dont les longueurs des M mots de code sont $\ell_1, ..., \ell_M$. Si Γ est à décodage unique alors il satisfait l'inégalité de Kraft :*

$$D^{-\ell_1} + ... + D^{-\ell_M} \leq 1 \tag{2.1}$$

Réciproquement, si des nombres $\ell_1, ..., \ell_M$ satisfont l'inégalité de Kraft (2.1), il existe un code D-aire instantané (donc à décodage unique) dont le dictionnaire possède M mots de code et dont les longueurs des mots de code sont $\ell_1, ..., \ell_M$.

Preuve : (1) décodage unique $\Rightarrow$ inégalité de Kraft. Pour chaque entier $n \geq 1$ soit $\mathcal{C}_n$ l'ensemble des concaténations de n mots de code de Γ. Comme Γ est à décodage unique, tout élément c de $\mathcal{C}_n$ peut être décodé de manière unique, et on peut donc écrire $c = c_1...c_n$ de manière unique, où $c_1, ..c_n$ sont des mots de code de Γ. Nous allons calculer de deux façons la quantité

$$F_n \stackrel{\text{def}}{=} \sum_{c \in \mathcal{C}_n} D^{-\ell(c)}$$

où $\ell(c)$ est la longueur du mot de code c, comptée en symboles de code ; notons que si $c = c_1...c_n$ alors $\ell(c) = \ell(c_1) + ... + \ell(c_n)$. Remarquons que pour $n = 1$, F_1 est le membre de gauche de l'inégalité de Kraft.

Premièrement, puisque c s'écrit de manière unique $c_1...c_n$, nous pouvons écrire

$$\begin{aligned} F_n &= \sum_{c_1 \in \mathcal{C} ... c_n \in \mathcal{C}} D^{-\ell(c_1...c_n)} = \sum_{c_1 \in \mathcal{C} ... c_n \in \mathcal{C}} \left[D^{-\ell(c_1)} ... D^{-\ell(c_n))} \right] \\ &= \left[\sum_{c_1 \in \mathcal{C}} D^{-\ell(c_1)} \right] ... \left[\sum_{c_n \in \mathcal{C}} D^{-\ell(c_n))} \right] \end{aligned}$$

par la formule du produit du développement d'une somme. Chacun des crochets est le même car la variable de sommation est

Q. 19. Pourquoi pourrait-il y avoir une erreur dans l'Algorithme de la Figure 2.3 ?

Par exemple, si $n = 2$ et Γ est le code B, qui est à décodage unique :

$$\begin{aligned} \mathcal{C}_2 = \{&00, 010, 0110, 01110, \\ &100, 1010, 10110, 101110, \\ &1100, 11010, 110110, 1101110, \\ &11100, 111010, 1110110, 11101110\} \end{aligned}$$

Les 16 éléments de $\mathcal{C}_2$ peuvent s'écrire de manière unique comme concaténation de 2 éléments de $\mathcal{C}_1$:

$\mathcal{C}_2$	0	10	110	1110
0	00	010	0110	01110
10	100	1010	10110	101110
110	1100	11010	110110	1101110
1110	11100	11010	1110110	11101110

Par contre, si Γ est le code A, qui n'est pas à décodage unique, $\mathcal{C}_2$ comporte seulement 15 éléments et on ne peut pas décomposer un élément de $\mathcal{C}_2$ de manière unique :

$\mathcal{C}_2$	0	01	10	11
0	00	001	**010**	011
01	**010**	0101	0110	0111
10	100	1001	1010	1011
11	110	1101	1110	1111

Par exemple si Γ est le code B :

$$\begin{aligned} F_2 &= 2^{-\ell(00)} + 2^{-\ell(010)} + 2^{-\ell(0110)} + 2^{-\ell(01110)} \\ &+2^{-\ell(100)} + 2^{-\ell(1010)} + 2^{-\ell(10110)} + 2^{-\ell(101110)} \\ &+2^{-\ell(1100)} + 2^{-\ell(11010)} + 2^{-\ell(110110)} + 2^{-\ell(1101110)} \\ &+2^{-\ell(11100)} + 2^{-\ell(111010)} + 2^{-\ell(1110110)} + 2^{-\ell(11101110)} \end{aligned}$$

$$\begin{aligned} &= 2^{-\ell(0)}2^{-\ell(0)} + 2^{-\ell(0)}2^{-\ell(10)} \\ &+2^{-\ell(0)}2^{-\ell(110)} + 2^{-\ell(0)}2^{-\ell(1110)} \\ &+2^{-\ell(10)}2^{-\ell(0)} + 2^{-\ell(10)}2^{-\ell(10)} \\ &+2^{-\ell(10)}2^{-\ell(110)} + 2^{-\ell(10)}2^{-\ell(1110)} \\ &+2^{-\ell(110)}2^{-\ell(0)} + 2^{-\ell(110)}2^{-\ell(10)} \\ &+2^{-\ell(110)}2^{-\ell(110)} + 2^{-\ell(110)}2^{-\ell(1110)} \\ &+2^{-\ell(1110)}2^{-\ell(0)} + 2^{-\ell(1110)}2^{-\ell(10)} \\ &+2^{-\ell(1110)}2^{-\ell(110)} + 2^{-\ell(1110)}2^{-\ell(1110)} \end{aligned}$$

$$\begin{aligned} &= \left[2^{-\ell(0)} + 2^{-\ell(10)} + 2^{-\ell(110)} + +2^{-\ell(1110)} \right] \\ &\cdot \left[2^{-\ell(0)} + 2^{-\ell(10)} + 2^{-\ell(110)} + +2^{-\ell(1110)} \right] \\ &= F_1 \cdot F_1 = F_1^2 \end{aligned}$$

muette, et vaut F_1. Donc :

$$F_n = F_1 ... F_1 = (F_1)^n \tag{2.2}$$

Deuxièmement, comme l'addition est associative, on peut regrouper les termes de la somme comme on veut. Mettons ensemble les éléments de $\mathcal{C}_n$ qui ont la même longueur, c'est à dire soit $\mathcal{C}_n^k = \{c \in \mathcal{C}_n, \ell(c) = k\}$, pour $k = 1$ à $nL_{\max}$ ($L_{\max}$est la longueur maximale d'un mot de code de Γ) :

$$\begin{aligned} F_n &= \sum_{c\in\mathcal{C}_n^1} D^{-\ell(c)} + ... + \sum_{c\in\mathcal{C}_n^k} D^{-\ell(c)} + ... + \sum_{c\in\mathcal{C}_n^{nL_{\max}}} D^{-\ell(c)} \\ &= \sum_{c\in\mathcal{C}_n^1} D^{-1} + ... + \sum_{c\in\mathcal{C}_n^k} D^{-k} + ... + \sum_{c\in\mathcal{C}_n^{nL_{\max}}} D^{-nL_{\max}} \\ &= D^{-1}\text{card}\mathcal{C}_n^1 + ... + D^{-k}\text{card}\mathcal{C}_n^k + ... + D^{-nL_{\max}}\text{card}\mathcal{C}_n^{nL_{\max}} \end{aligned}$$

Par exemple si Γ est le code B :

$$\begin{aligned} \mathcal{C}_2^1 &= \varnothing \text{ (ensemble vide)} \\ \mathcal{C}_2^2 &= \{00\} \\ \mathcal{C}_2^3 &= \{010, 100\} \\ \mathcal{C}_2^4 &= \{0110, 1010, 1100\} \\ \mathcal{C}_2^5 &= \{01110, 10110, 11010, 11100\} \\ \mathcal{C}_2^6 &= \{101110, 110110, 111010\} \\ \mathcal{C}_2^7 &= \{1101110, 1110110\} \\ \mathcal{C}_2^8 &= \{11101110\} \end{aligned}$$

Notons que certains de ces ensembles $\mathcal{C}_n^k$ peuvent être vides, auquel cas la somme correspondante vaut 0. Maintenant, remarquons que chaque élément de $\mathcal{C}_n^k$ est une suite de k symboles de code ; il y a au maximum D^k telles suites, donc $\text{card}(\mathcal{C}_n^k) \leq D^k$. Donc

$$\begin{aligned} F_n &\leq D^{-1}D^1 + ... + D^{-k}D^k + ... + D^{-nL_{\max}}D^{nL_{\max}} \\ &= \overbrace{1 + ... + 1 + ... + 1}^{nL_{\max} \text{ fois}} = nL_{\max} \end{aligned}$$

En comparant avec Eq.(2.2), nous avons montré que :

$$\forall n \geq 1 : \quad (F_1)^n/n \leq L_{\max} \tag{2.3}$$

Pour $b > 1$, $\lim_{x\to+\infty} b^x = +\infty$ et la fonction $x \mapsto b^x$ croît "plus vite" que x.

Or, nous savons du cours d'analyse que pour tout nombre $b > 1$, on a

$$\lim_{x\to+\infty} \frac{b^x}{x} = +\infty$$

Appliquons cela à $b = F_1$; si on avait $F_1 > 1$, on aurait

$$\lim_{n\to+\infty} (F_1)^n/n = +\infty \tag{2.4}$$

C'est un *raisonnement par l'absurde* : pour montrer $A \Rightarrow B$, on suppose que l'hypothèse A est vraie et que la conclusion B est fausse, et on arrive à une contradiction. Ici la contradiction est entre Eq.(2.3) et Eq.(2.4).

ce qui est impossible compte tenu de Eq.(2.3). Donc $F_1 \leq 1$, c'est à dire que l'inégalité de Kraft est satisfaite.

(2) inégalité de Kraft $\Rightarrow$ il existe un code instantané D-aire de longueurs ℓ_i, $i = 1$ à M. Classons les longueurs de mots par ordre croissant : $\ell_1 \leq \ell_2 \leq ... \leq \ell_M \stackrel{\text{def}}{=} L_{\max}$. Nous allons construire un arbre de décodage, de la façon suivante (Figure 2.4).

Construisons d'abord un arbre D-aire complet de profondeur $L_{\max}$. Puis choisissons le premier noeud de profondeur ℓ_1, étiquetons-le c_1, et supprimons tous ses descendants. Ce faisant, nous avons supprimé $D^{L_{\max}-\ell_1}$ noeuds terminaux de profondeur $L_{\max}$. Puis recommençons : choisissons le premier noeud de profondeur ℓ_2 de l'arbre ainsi construit, étiquetons-le c_2, et supprimons tous ses descendants. Lesdits descendants sont distincts de ceux de c_1, par construction, et les descendants terminaux de profondeur $L_{\max}$ sont au nombre de $D^{L_{\max}-\ell_2}$. Ce faisant, nous avons supprimé $D^{L_{\max}-\ell_2}$

noeuds terminaux de profondeur $L_{\max}$, donc en tout nous en avons supprimé $D^{L_{\max}-\ell_1} + D^{L_{\max}-\ell_2}$.

Continuons cette procédure. A la me étape, nous avons supprimé $D^{L_{\max}-\ell_1} + ... + D^{L_{\max}-\ell_m}$ noeuds terminaux de profondeur $L_{\max}$. Mais l'inégalité de Kraft est vraie par hypothèse, donc

$$D^{L_{\max}-\ell_1} + ... + D^{L_{\max}-\ell_m} \leq D^{L_{\max}}$$

c'est à dire que ce nombre est inférieur au nombre total de noeuds de profondeur $L_{\max}$. Donc, après avoir placé les mots $c_1...c_m$ de cette façon, il reste au moins un noeud terminal de profondeur $L_{\max}$. Choisissons le premier de ces noeuds, et remontons l'arbre vers la source jusqu'à la profondeur ℓ_{m+1} ; choisissons ce noeud, étiquetons-le c_{m+1}, et supprimons tous ses descendants. Cette procédure peut continuer jusqu'à avoir placé c_M. A la fin, nous avons construit un arbre de décodage, donc un code sans préfixe, dont les longueurs de mots sont $\ell_1, ..., \ell_M$. □

Exemple 2.3 (Interprétation Graphique) *La preuve de la deuxième partie du Théorème de Kraft-McMillan nous donne aussi une interprétation simple et utile de l'inégalité de Kraft pour un code instantané. Nous l'illustrons sur le code B en Figure 2.4. Pour chaque mot de code placé sur l'arbre de décodage, comptons le nombre de noeuds terminaux de l'arbre complet qui descendent du noeud où est placé le mot de code. On trouve :* 2^3, 2^2, 2^1 *et* 2^0. *Il y a en tout* 2^4 *noeud terminaux donc* $2^3 + 2^2 + 2^1 + 2^0 \leq 2^4$.

En général, pour un mot de longueur ℓ_i, *il y a* $D^{L_{\max}-\ell_i}$ *noeuds terminaux. Il y a en tout* $D^{L_{\max}}$ *noeuds terminaux, donc*

$$D^{L_{\max}-\ell_1} + ... + D^{L_{\max}-\ell_M} \leq D^{L_{\max}}$$

ce qui, après multiplication par $D^{-L_{\max}}$ *est l'inégalité de Kraft.*

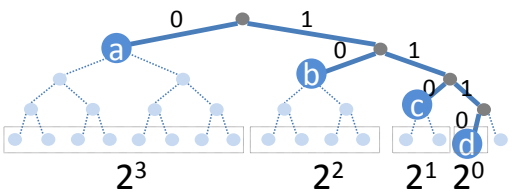


FIGURE 2.4: L'inégalité de Kraft expliquée sur l'exemple du code$B$.

Exemple 2.4 (Quatre Petits Codes, suite) *L'inégalité de Kraft donne :*

O : $2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 1 \leq 1$
A : $2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} = 1.25 > 1$
B *ou* C : $2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 0.9375 \leq 1$

Elle est vérifiée sauf pour le code A. *Les codes* O, B *et* C *sont à décodage unique donc satisfont l'inégalité de Kraft, comme on le vérifie.*

Par contraposition du théorème, nous pouvons conclure que A *n'est pas à décodage unique, ce qu nous savions déjà.*

Soit un code Γ dont les longueurs de mots sont $\ell_1, ..., \ell_M$. Si l'inégalité de Kraft *n'est pas* satisfaite, le Théorème de Kraft-McMillan permet de conclure, par contraposition, que le code n'est pas à décodage unique (comme dans l'exemple précédent).

Par contre, si l'inégalité de Kraft *est* satisfaite, nous ne pouvons pas conclure que Γ est à décodage unique. Le théorème dit seulement qu'*il existe* un code à décodage unique ayant les mêmes longueurs de mots, mais rien n'assure que ce code soit précisément Γ, comme l'illustre l'exemple suivant.

Exemple 2.5 (Non décodage unique malgré Kraft) *Soit le code A', ternaire, (c'est à dire sur l'alphabet de code $\{0,1,2\}$) donné par la même table que le code A. . Il n'est pas à décodage unique pour la même raison que A (bc et ada sont encodés de la même façon). Par contre il satisfait l'inégalité de Kraft : $3^{-1} + 3^{-2} + 3^{-2} + 3^{-2} = 2/3 \leq 1$.*

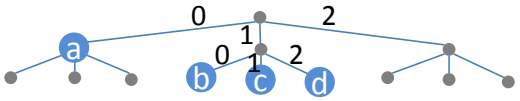
Notons que le code A' est ternaire par définition de l'alphabet, mais en fait il n'utilise pas le symbole 2. C'est un code d'intérêt purement académique, que nous utilisons seulement comme contre-exemple.

Q. 20. Que donne l'inégalité de Kraft pour les codes de longueur constante L ?

2.7 *Construire un Code Instantané dont les Longueurs de Mots sont Données.*

Bien sûr, ceci n'est possible que si les longueurs de mot satisfont l'inégalité de Kraft. La preuve de la deuxième partie du Théorème de Kraft-McMillan donne une telle construction : il suffit de construire un arbre D-aire complet de profondeur égale à la plus grande longueur des mots, de classer les longueurs de mots par ordre croissant, et de placer les mots sur l'arbre, en prenant soin de supprimer les branches descendant d'un mot placé.

Exemple 2.6 (Code équivalent à A', mais à décodage unique) *Le code A' de l'exemple précédent (qui n'est pas un bon code) peut-être remplacé par un code ternaire qui a les mêmes longueurs de mots mais est à décodage unique. Pour cela, on construit l'arbre de décodage ci-dessous :*



On obtient le code :

symbole de source	*a*	*b*	*c*	*d*
mot de code	0	10	11	12

qui a les mêmes longueur de mots que A', mais qui est instantané (par construction) donc à décodage unique.

Enfin, une conséquence spectaculaire du théorème de Kraft-McMillan est qu'on peut toujours remplacer un code à décodage unique par un code instantané :

Théorème 2.3 *Pour tout code à décodage unique, il existe un code* instantané *sur les mêmes alphabets de source et de code qui a les mêmes longueurs de mot.*

☺*Preuve :* Soit Γ un code à décodage unique. Les longueurs de mots satisfont l'inégalité de Kraft (première partie du Théorème de Kraft-McMillan). Donc il existe un code instantané avec ces longueurs de mots (deuxième partie du Théorème de Kraft-McMillan). ☺□

Par exemple, le code C est à décodage unique et a les mêmes longueurs de mot que le code B. On peut le remplacer par le code B et obtenir un code qui est équivalent du point de vue des longueurs de mots.

Q. 21. Les implications suivantes sont-elles vraies ?

1. Γ est instantané ⇒ Γ vérifie l'inégalité de Kraft
2. Γ est à décodage unique ⇒ Γ vérifie l'inégalité de Kraft
3. Γ vérifie l'inégalité de Kraft ⇒ Γ est instantané
4. Γ vérifie l'inégalité de Kraft ⇒ Γ est à décodage unique
5. Γ ne vérifie pas l'inégalité de Kraft ⇒ Γ n'est pas à décodage unique
6. Γ n'est pas à décodage unique ⇒ Γ ne vérifie pas l'inégalité de Kraft
7. Γ ne vérifie pas l'inégalité de Kraft ⇒ Γ n'est pas instantané
8. Γ n'est pas instantané ⇒ Γ ne vérifie pas l'inégalité de Kraft

3
Efficacité d'un Code de Source

MAINTENANT QUE NOUS AVONS FAIT CONNAISSANCE avec les codes de longueur variable, nous pouvons commencer à nous intéresser à obtenir des codes *efficaces*. A cette occasion nous allons retrouver notre vieille amie l'entropie.

3.1 Première Inégalité de l'Entropie

La quantité d'intérêt pour l'efficacité d'un code est sa longueur moyenne, définie comme le nombre moyen de symboles de code par symbole de source :

On utilise le même mot "bit" pour désigner deux concepts différents : un symbole binaire, et l'unité d'information.

Définition 3.1 *Soit une source S d'alphabet $\mathcal{A}$ et de densité de probabilité p, et soit Γ un code D-aire de la source S. La longueur moyenne du code Γ est*

$$L(\Gamma) \stackrel{\text{def}}{=} \sum_{s\in\mathcal{A}} p(s)\ell(\Gamma(s))$$

L'unité est le symbole de code par symbole de source (si $D = 2$ on dit bits par symbole source*).*

Exemple 3.1 (Codes O, B et B') *Supposons que les probabilités pour la source des codes O et B (voir Table 2.1 en page 21) soient comme dans la Table 3.1. Le code O est de longueur constante égale à 2 symboles binaires (bits) donc sa longueur moyenne est $L(O) = 2$ bits. Pour le code B les longueurs des mots sont respectivement $1, 2, 3$ et 4 donc*

$$L(B) = 0.05\cdot 1 + 0.05\cdot 2 + 0.1\cdot 3 + 0.8\cdot 4 = 3.65 \text{ bits par symbole}$$

Si on remplace le code B par le code B' dont les mots sont pris en ordre inverse de B (Table 3.1) on obtient

$$L(B') = 0.05\cdot 4 + 0.05\cdot 3 + 0.1\cdot 2 + 0.8\cdot 1 = 1.35 \text{ bits par symbole}$$

Le code B' est plus efficace que les codes B et O, car il donne des longueurs courtes aux symboles les plus fréquents.

symbole de source s	proba $p(s)$	code B'
a	0.05	1110
b	0.05	110
c	0.1	10
d	0.8	0

TABLE 3.1: Densité de probabilité pour la source des quatre petits codes de la Table 2.1, et code B'.

Q. 22. Montrez que le code B' est plus efficace (c'est à dire : a une longueur moyenne plus faible) que n'importe quel code binaire de longueur constante pour cette source.

Pour obtenir un code efficace (c'est à dire pour comprimer l'information), nous sommes intéressés à avoir des codes de longueur moyenne aussi petite que possible. L'entropie nous donne une limite inférieure à ce qu'il est possible d'atteindre :

Théorème 3.1 (Première Inégalité de l'Entropie) *Soit une source S d'entropie $H(S)$ et soit Γ un code D-aire de la source S. Si Γ est à décodage unique, sa longueur moyenne satisfait*

$$L(\Gamma) \geq \frac{H(S)}{\log_2(D)} \tag{3.1}$$

Pour un code Γ *binaire* on a donc simplement $L(\Gamma) \geq H(S)$.

☺*Preuve :* Nous utilisons l'inégalité de Kraft et l'inégalité de concavité du logarithme. Soient $M = \text{card}(\mathcal{A})$, ℓ_i la longueur du $i$ème mot de code et p_i la densité de probabilité du $i$ème symbole de source, pour $i = 1...M$. Par le Théorème de Kraft-McMillan :

$$D^{-\ell_1} + ... + D^{-\ell_M} \leq 1$$

donc

$$\log_2\left(D^{-\ell_1} + ... + D^{-\ell_M}\right) \leq 0 \tag{3.2}$$

Appliquons l'inégalité de concavité (1.2) à $\alpha_i = p_i$ et $x_i = \frac{D^{-\ell_i}}{p_i}$, il vient :

$$\begin{aligned}
&\log_2\left(D^{-\ell_1} + ... + D^{-\ell_M}\right) \geq p_1 \log_2\left(\tfrac{D^{-\ell_1}}{p_1}\right) + ... + p_M \log_2\left(\tfrac{D^{-\ell_M}}{p_M}\right) \\
&= p_1 \log_2\left(D^{-\ell_1}\right) - p_1 \log_2 p_1 + ... + p_M \log_2\left(D^{-\ell_M}\right) - p_M \log_2 p_M \\
&= -p_1\ell_1 \log_2(D) - ... - p_M\ell_M \log_2(D) - p_1 \log_2 p_1 - ... - p_M \log_2 p_M \\
&= -L(\Gamma)\log_2(D) + H(S)
\end{aligned}$$

En comparant avec Eq.(3.2), il vient

$$0 \geq -L(\Gamma)\log_2(D) + H(S)$$

ce qui donne l'inégalité à démontrer. ☺□

Exemple 3.2 (Codes B et B', suite) *Les trois codes sont à décodage unique (nous le savons déjà pour O et B, B' l'est aussi, pour la même raison que B). L'entropie de la source est*

$$H(S) = -2 \cdot 0.05 \log_2(0.05) - 0.1 \log_2(0.1) - 0.8 \log_2(0.8) = 1.022 \text{ bits}$$

et on a bien $L(B) = 3.65 \geq H(S)$ et $L(B') = 1.35 \geq H(S)$ Le code B' est sans doute assez efficace car sa longueur moyenne est proche de la borne inférieure de l'entropie.

3.2 Code de Shannon-Fano et Deuxième Inégalité de l'Entropie

Pour un nombre réel x, on note $\lceil x \rceil$ la partie entière par excès de x, définie comme le plus petit nombre entier $\geq x$. Par exemple $\lceil 3.14 \rceil = 4$, $\lceil 3 \rceil = 3$ et $\lceil -3.14 \rceil = -3$. On a :

$$\forall x \in \mathbb{R} : \quad x \leq \lceil x \rceil < x + 1$$

Pour obtenir un code efficace, il faut donner des longueurs petites aux symboles les plus fréquents. L'idée du code de Shannon-Fano est de choisir un code D-aire avec $\log_D(1/p(s))$ comme longueur du mot de code pour le symbole s ; plus exactement, comme ce nombre n'est pas forcément entier, on choisit l'arrondi entier par excès $\lceil \log_D(1/p(s)) \rceil$. Il reste à voir si de tels codes à décodage unique existent, ce qui est le cas :

Théorème 3.2 *Soit une source S avec M symboles dont les densités de probabilité sont $p_1, ..., p_M$. Il existe des codes D-aires instantanés (donc à décodage unique) dont les longueurs de mots sont $\ell_i = \lceil \log_D(1/p_i) \rceil$ pour $i = 1...M$. De tels codes sont appelés codes D-aires de Shannon-Fano.*

☺*Preuve :* Il suffit de montrer que l'inégalité de Kraft est vraie. Or

$$\ell_i \stackrel{\text{def}}{=} \lceil \log_D (1/p_i) \rceil \geq \log_D (1/p_i)$$

donc $D^{-\ell_i} \leq p_i$ et

$$D^{-\ell_1} + ... + D^{-\ell_M} \leq p_1 + ... + p_M = 1$$

donc l'inégalité de Kraft est vérifiée. ☺□

Pour construire un code de Shannon-Fano, il suffit d'appliquer la méthode de la Section 2.7, puisque les longueurs de mots de code sont connues.

Exemple 3.3 *(Code Γ_{SF}). Considérons la source des quatre petits codes et construisons un code binaire Γ_{SF} de Shannon-Fano. Les longueurs des mots sont données dans la Figure 3.1. L'arbre de décodage est donné sur la figure.*

La longueur moyenne du code Γ_{SF} est

$$L(\Gamma_{SF}) = 2 \cdot 0.05 \cdot 5 + 0.1 \cdot 4 + 0.8 \cdot 1 = 1.7 \text{ bits par symbole}$$

Le code Γ_{SF} est assez efficace, il n'est pas trop loin de la borne inférieure de l'entropie (1.022), et est plus efficace que le code B (longueur moyenne 3.65). Cependant, il n'est pas le plus efficace : le code B' a une longueur moyenne plus petite (longueur moyenne 1.35).

symbole de source	proba	longueur
a	0.05	5
b	0.05	5
c	0.1	4
d	0.8	1

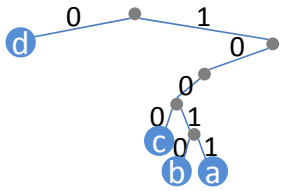


FIGURE 3.1: Code de Shannon-Fano pour la source de la Table 3.1.

Comme l'illustre l'exemple ci-dessus, les codes de Shannon-Fano sont assez efficaces, sans être en général les plus efficaces. Cependant, et c'est leur principal attrait, ils sont *garantis* : ils ne peuvent pas être à plus d'une unité de la borne inférieure de l'entropie.

Théorème 3.3 (Deuxième Inégalité de l'Entropie) *La longueur moyenne $L(\Gamma_{SF})$ d'un code D-aire de Shannon-Fano d'une source d'entropie $H(S)$ vérifie*

$$\frac{H(S)}{\log_2(D)} \leq L(\Gamma_{SF}) < \frac{H(S)}{\log_2(D)} + 1 \tag{3.3}$$

☺*Preuve :* Notons que la première inégalité de (3.3) est la première inégalité de l'entropie, qui est vraie car un code de Shannon-Fano est instantané, donc à décodage unique. Il nous reste à montrer la deuxième inégalité. Soient $\ell_1, ..., \ell_M$ les longueurs des mots du code de Shannon-Fano et $p_1, ..., p_M$ les probabilités. On a :

$$\ell_i \stackrel{\text{def}}{=} \lceil \log_D (1/p_i) \rceil < \log_D (1/p_i) + 1$$

donc

$$L(\Gamma_{SF}) \stackrel{\text{def}}{=} \ell_1 p_1 + ... + \ell_M p_M$$

$$
\begin{aligned}
&< p_1 \log_D (1/p_1) + \ldots + p_M \log_D (1/p_M) + \overbrace{p_1 + \ldots + p_M}^{1} \\
&= \frac{p_1 \log_2 (1/p_1) + \ldots + p_M \log_2 (1/p_M)}{\log_2(D)} + 1 \\
&= \frac{H(S)}{\log_2(D)} + 1
\end{aligned}
$$

☺□

Pour un code binaire de Shannon-Fano on a

$$H(S) \leq L(\Gamma_{SF}) < H(S) + 1$$

Exemple 3.4 (Code Γ_{SF}) *La longueur moyenne du code Γ_{SF} est 1.7, on a bien*

$$1.022 = H(S) \leq L(\Gamma_{SF}) = 1.7 < 2.022$$

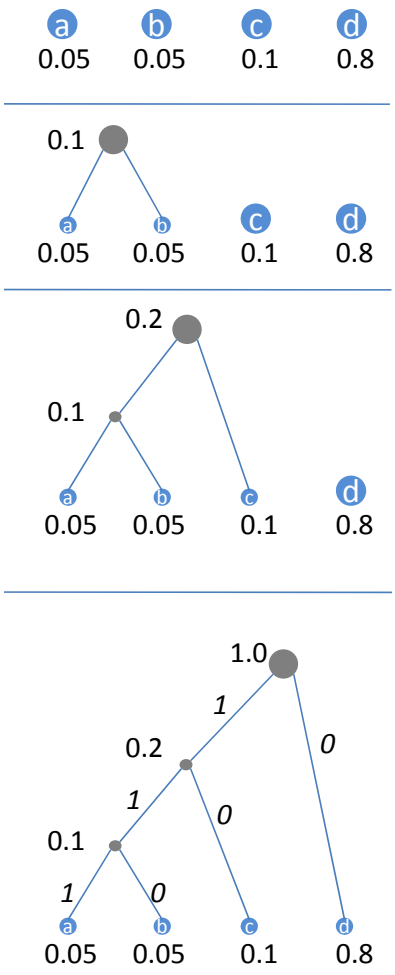
3.3 Code Optimal ou Code de Huffman

Nous avons obtenus jusqu'ici une borne inférieure sur la longueur de tout code, et nous avons vu que les codes de Shannon-Fano sont à au plus unité de cette borne inférieure. Mais est-ce que les meilleurs codes possibles atteignent la borne inférieure ? En général, la réponse est non. Par contre, nous allons voir dans cette section que l'on peut toujours créer un *code optimal*, c'est à dire de longueur minimale parmi tous les codes à décodage unique possibles : ce sont les *codes de Huffman*. Nous commençons par décrire la procédure pour créer un code de Huffman.

Les codes de Huffman existent pour toutes les valeurs de D, mais leur description est un peu compliquée quand $D \geq 3$, aussi nous nous limitons aux codes de Huffman *binaires*. Etant donné une source S, un code binaire de Huffman est un code instantané binaire dont l'arbre de décodage est construit de la manière suivante.

1. L'arbre est construit à l'envers, en partant des noeuds terminaux. Chaque noeud est étiqueté avec deux attributs : un nombre dans $[0, 1]$ (la probabilité du noeud), et une indication de statut "actif" ou "inactif".
2. Créer M noeuds terminaux, un par symbole de source, la probabilité d'un noeud est celle du symbole de source correspondant. Tous les noeuds ont le statut "actif".
3. Choisir 2 noeuds de probabilités les plus petites, changer leur statut à "inactif", et créer un nouveau noeud ancêtre de ces deux noeuds. Le nouveau noeud prend le status "actif" et sa probabilité est la somme des probabilités des noeuds qu'il remplace.
4. Continuer l'étape précédente jusqu'à obtention d'un noeud dont la probabilité est 1. On a alors obtenu un arbre binaire dont les noeuds terminaux sont associés aux symboles de source. Étiqueter les branches de l'arbre avec les symboles de code 0 et 1 selon un choix arbitraire. L'arbre obtenu est un arbre de décodage qui définit un code instantané.

Exemple 3.5 *La Figure 3.2 illustre la construction du code de Huffman pour la source des 4 petits codes (Table 3.1). Le code obtenu est le code Γ_H, qui diffère du code B' par le codage du symbole a.*



symbole de source s	proba $p(s)$	code Γ_H
a	0.05	111
b	0.05	110
c	0.1	10
d	0.8	0

FIGURE 3.2: Construction d'un code de Huffman. Les noeuds actifs à la fin de chaque étape sont plus gros que les noeuds inactifs. Le code obtenu, Γ_H, est optimal.

Le théorème suivant exprime que les codes de Huffman sont optimaux. La preuve est un peu longuette et nous l'omettons.

Voir par exemple `http://icwww.epfl.ch/~chappeli/it/courseFR/Glossary.php` pour une preuve du Théorème 3.4, ainsi qu'une description des codes de Huffman D-aires avec $D > 2$.

Théorème 3.4 (Code de Huffman) *La méthode décrite ci-dessus produit un code binaire instantané Γ_H optimal, c'est à dire que pour tout autre code binaire à décodage unique Γ pour la même source, on a*

$$L(\Gamma_H) \leq L(\Gamma)$$

Notons que, puisque le binaire code de Huffman Γ_H est optimal, il domine le code binaire de Shannon-Fano Γ_{SF}, donc $L(\Gamma_H) \leq L(\Gamma_{SF}) < H(S) + 1$. Donc finalement, pour des codes binaires

$$H(S) \leq L(\Gamma_H) \leq L(\Gamma_{SF}) < H(S) + 1$$

Exemple 3.6 (Source des 4 Petits Codes) *Le code de Huffman Γ_H obtenu sur la Figure 3.2 a pour longueur moyenne* 1.30 *bits. D'après le Théorème 3.4, on ne peut pas faire mieux que ce code et sa longueur est la longueur minimale d'un code pour cette source.*

Notons que la longueur du code optimal Γ_H est supérieure *à la borne inférieure de l'entropie, qui vaut $H(S) = 1.022$. Pour cette source, la borne inférieure de l'entropie ne peut pas être atteinte.*

Exemple 3.7 (Robot-page, suite) *Nous avons calculé un code binaire de Huffman Γ_H pour le robot-page de l'Exemple 1.4. Les longueurs, en bits, des mots de code assignés par le code à chaque lettre de l'alphabet sont indiquées dans la Table 3.2. La lettre la plus fréquente, E, est codée sur 2 bits, alors que la lettre la moins fréquente, K, nécessite 11 bits.*

La longueur moyenne du code optimal est $L(\Gamma_H) = 4.00$, ce qui est proche de la borne de l'entropie $H(S) = 3.95$.

lettre	fréquence	bits
A	8,11	4
B	0,81	7
C	3,38	5
D	4,28	5
E	17,69	2
F	1,13	7
G	1,19	6
H	0,74	7
I	7,24	4
J	0,18	9
K	0,02	11
L	5,99	4
M	2,29	6
N	7,68	4
O	5,20	4
P	2,92	5
Q	0,83	7
R	6,43	4
S	8,87	4
T	7,44	4
U	5,23	4
V	1,28	6
W	0,06	11
X	0,53	7
Y	0,26	8
Z	0,12	10

TABLE 3.2: Fréquences des lettres du Français, exprimées en pourcentages, et nombre de bits assignés par un code binaire de Huffman.

Q. 23. La Table 3.2 donne les longueur des mots du code de Huffman, mais pas les mots de code. Est-il possible de déduire les mots de code à partir des longueurs de mots seulement ?

Q. 24. La longueur moyenne d'un code de Huffman est-elle égale à la borne inférieure donnée dans Eq.(3.1) ?

4
Entropie Conditionnelle

La théorie de l'entropie et du codage que nous avons vue jusqu'ici concerne le codage d'un seul message d'une source. Dans le cas du robot-page de l'Exemple 3.7, le meilleur code de source utilise 4 bits par lettre, ce qui est très proche de l'entropie de la source (3.95 bits). En comparaison avec un code naïf qui consisterait à coder les 26 lettres sur 5 bits ($2^5 = 32$) le gain n'est pas nul, mais n'est pas spectaculaire non plus. Des algorithmes de compression utilisés sur des fichiers de texte peuvent faire beaucoup mieux, atteignant environ 1 bit par lettre. N'est ce pas contradictoire avec la borne inférieure de l'entropie ? La réponse est non, bien sûr, puisque les inégalités de l'entropie sont mathématiquement prouvées. La raison pour cette différence est que les textes français ne sont pas produits par des robots-pages, mais par des auteurs. Cela nous amène à considérer l'entropie conditionnelle, définie dans ce chapitre.

4.1 Entropie Conditionnelle

Définition 4.1 *Soit $S = (S_1, S_2)$ une source composée. L'entropie conditionnelle de S_2 sachant que $S_1 = s_1$ est l'entropie de la densité conditionnelle de S_2 sachant que $S_1 = s_1$:*

$$H(S_2|S_1 = s_1) \stackrel{\text{def}}{=} -\sum_{s_2 \in \mathcal{A}_2} p_{S_2|S_1}(s_2|s_1) \log_2(p_{S_2|S_1}(s_2|s_1))$$

L'entropie conditionnelle de S_2 sachant S_1 en est la moyenne :

$$H(S_2|S_1) \stackrel{\text{def}}{=} \sum_{s_1 \in \mathcal{A}_1} H(S_2|S_1 = s_1) p_{S_1}(s_1)$$

L'entropie conditionnelle mesure la quantité d'information moyenne que l'on reçoit quand on observe une source, après avoir observé l'autre, ce qu'on peut aussi appeler l'information "supplémentaire".

Notons que l'ordre des sources dans la définition n'a pas d'importance, on définit de la même façon l'entropie conditionnelle de S_1 sachant S_2 par

$$H(S_1|S_2 = s_2) \stackrel{\text{def}}{=} -\sum_{s_1 \in \mathcal{A}_1} p_{S_1|S_2}(s_1|s_2) \log_2(p_{S_1|S_2}(s_1|s_2))$$

et

$$H(S_1|S_2) \stackrel{\text{def}}{=} \sum_{s_2 \in \mathcal{A}_2} H(S_1|S_2 = s_2) p_{S_2}(s_2)$$

Exemple 4.1 (Somme de Deux Dés Codée Sur Deux Chiffres, suite) *L_1 est le premier chiffre de la somme de deux dés, L_2 le deuxième. La densité conditionnelle de L_2 sachant que $L_1 = i$ pour $i = 0, 1$ est donnée dans*

la Table 2 en page 13. On obtient donc

$$H(L_2|L_1 = 0) = 2.857426 \text{ bits}$$
$$H(L_2|L_1 = 1) = 1.459148 \text{ bits}$$

La densité marginale de L_1 est donnée dans la dernière ligne de la Table 1 (page 11), d'où :

$$H(L_2|L_1) = {}^5/_6 \times 2.857426 + {}^1/_6 \times 1.459148 = 2.624379 bits$$

Connaissant le premier chiffre, observer le deuxième donne 2.62 *bits d'information (alors qu'observer le deuxième chiffre sans connaître le premier donne $H(L_2) = 3.22$ bits d'information).*

4.2 Propriétés de l'Entropie Conditionnelle

Notons d'abord que, comme l'entropie, l'entropie conditionnelle est toujours ≥ 0, puisque c'est une somme de nombres ≥ 0.

Il est souvent plus facile de calculer l'entropie conditionnelle en utilisant le théorème suivant, qui exprime que l'information que nous délivre la source composée est la somme de l'information délivrée par une composante, plus l'information supplémentaire délivrée par l'autre composante :

Théorème 4.1 (Calcul de l'Entropie Conditionnelle) *Soit $S = (S_1, S_2)$ une source composée.*

$$\begin{aligned} H(S_1, S_2) &= H(S_1) + H(S_2|S_1) \\ &= H(S_2) + H(S_1|S_2) \end{aligned}$$

Comme l'entropie conditionnelle est ≥ 0, il s'en suit en particulier que

$$H(S_1) \leq H(S_1, S_2) \tag{4.1}$$

Q. 25. Nous avons trouvé que le robot-page, qui met tous les caractères en majuscule, a une entropie de 3.95. Comment cette entropie serait-elle modifiée par un nouveau robot page qui conserverait la casse des lettres (minuscule ou majuscule) ?

Preuve : Nous faisons la preuve seulement pour le cas où la densité de probabilité de S est positive (c'est à dire que $p_{S_1}(s_1) > 0$ pour tout symbole s_1). Appliquons la définition de l'entropie conditionnelle :

$$\begin{aligned} H(S_2|S_1) &= -\sum_{s_1 \in \mathcal{A}_1} p_{S_1}(s_1) \sum_{s_2 \in \mathcal{A}_2} p_{S_2|S_1}(s_2|s_1) \log_2 \left(p_{S_2|S_1}(s_2|s_1) \right) \\ &= -\sum_{(s_1,s_2) \in \mathcal{A}} p_{S_1}(s_1) p_{S_2|S_1}(s_2|s_1) \log_2 \left(p_{S_2|S_1}(s_2|s_1) \right) \end{aligned}$$

où nous avons appliqué l'associativité de la somme. Maintenant remarquons que

$$\log_2 \left(p_{S_2|S_1}(s_2|s_1) \right) = \log_2(p(s_1, s_2)) - \log_2(p_{S_1}(s_1))$$

où p est la densité de probabilité de $S = (S_1, S_2)$. Donc

$$\begin{aligned} H(S_2|S_1) = &\overbrace{-\sum_{(s_1,s_2) \in \mathcal{A}} p_{S_1}(s_1) p_{S_2|S_1}(s_2|s_1) \log_2(p(s_1, s_2))}^{(1)} \\ &+ \overbrace{\sum_{(s_1,s_2) \in \mathcal{A}} p_{S_1}(s_1) p_{S_2|S_1}(s_2|s_1) \log_2(p_{S_1}(s_1))}^{(2)} \end{aligned}$$

Or

$$(1) = -\sum_{(s_1,s_2)\in\mathcal{A}} p(s_1,s_2)\log_2(p(s_1,s_2)) = H(S_1,S_2)$$

et

$$(2) = \sum_{s_1\in\mathcal{A}_1} \overbrace{\left[\sum_{s_2\in\mathcal{A}_2} p_{S_2|S_1}(s_2|s_1)\right]}^{=1} p_{S_1}(s_1)\log_2(p_{S_1}(s_1)) = -H(S_1)$$

Donc $H(S_2|S_1) = H(S_1,S_2) - H(S_1)$. □

Exemple 4.2 (Somme de Deux Dés Codée Sur Deux Chiffres, suite) *Nous avons déjà calculé les entropies dans l'Exemple 1.5 :* $H(L) = H(L_1,L_2) = 3.274402$ *bits,* $H(L_1) = 0.650022$ *bit et* $H(L_2) = 3.218846$ *bit. Donc*

$$H(L_2|L_1) \quad = \quad H(L_1,L_2) - H(L_1) = 3.274402 - 0.650022 = 2.624379 \approx 2.62 \text{ bits}$$

ce qui est la même valeur que celle déjà calculée dans l'Exemple 4.1, mais cette méthode de calcul est plus simple.

Notons que

$$H(L_2|L_1) \quad = \quad 2.62 \text{ bits} \; < H(L_2) = 3.22 \text{ bits}$$

Q. 26. Quelle est l'entropie conditionnelle du premier chiffre L_1 sachant le deuxième L_2 ?

i.e. l'entropie conditionnelle (= information supplémentaire) est moindre que l'entropie.

Exemple 4.3 (Deux Dés non Pipés, suite) *S_1 est le résultat du tirage d'un premier dé non pipé, S_2 du deuxième. Dans l'Exemple 1.6 nous avons calculé que*

$$H(S_1,S_2) \quad = \quad 2\log_2(6), \quad H(S_1) = H(S_2) = \log_2(6)$$

donc

$$\begin{aligned} H(S_2|S_1) &= H(S_1,S_2) - H(S_1) = \log_2(6) \\ H(S_1|S_2) &= H(S_2,S_1) - H(S_2) = \log_2(6) \end{aligned}$$

Ici $H(S_2|S_1) = H(S_2)$, ce qui est naturel puisque S_1 et S_2 sont indépendantes. Connaître S_1 n'apporte aucune information sur S_2 quand on sait que le dé n'est pas pipé.

De même $H(S_1|S_2) = H(S_1)$.

Dans les exemples précédents, nous avons vu que $H(S_2|S_1) \leq H(S_2)$, avec égalité quand S_1 et S_2 sont indépendantes. C'est un fait tout à fait général, conséquence du Théorème 1.4 :

Théorème 4.2 (Conditionner Réduit l'Entropie) *Soit $S = (S_1,S_2)$ une source composée.*

1. $H(S_2|S_1) \leq H(S_2)$
2. $H(S_2|S_1) = H(S_2)$ *si et seulement si S_1 et S_2 sont indépendantes.*

On a bien sûr le même résultat en inversant l'ordre : $H(S_1|S_2) \leq H(S_1)$ et il y a égalité si et seulement si S_1 et S_2 sont indépendantes.

☺*Preuve :* (1) Par les Théorèmes (4.1) et (1.4) on a

$$H(S_2|S_1) \quad = \quad H(S_1,S_2) - H(S_1) \leq [H(S_1)+H(S_2)] - H(S_1) = H(S_2)$$

ce qui prouve l'item 1. (2) Supposons qu'il y ait égalité dans ce qui précède. Alors $H(S_1,S_2) = H(S_1)+H(S_2)$; d'après le Théorème 1.4, S_1 et S_2 sont indépendantes. ☺□

Une variante du Théorème 4.2, qui nous sera utile plus tard, est le théorème suivant, que nous ne démontrons pas. Il exprime que l'information supplémentaire apportée par S_3 quand on connaît S_1 et S_2 est moindre que celle obtenue quand on ne connaît que S_2 :

Théorème 4.3 (Conditionner Réduit l'Entropie, suite) *Soit $S = (S_1,S_2,S_3)$ une source composée.*

$$H(S_3|S_1,S_2) \leq H(S_3|S_2)$$

Dans une notation telle que $H(S_3|S_1,S_2)$, on considère qu'il y a une source composée $S = ((S_1,S_2),S_3)$ dont la première composante est elle-même une source composée.

Enfin nous terminons cette section avec la règle suivante, appelée parfois *règle d'enchaînement*, qui généralise le calcul de l'entropie conditionnelle quand on a n sources :

Théorème 4.4 (Calcul Incrémental de l'Entropie Conditionnelle) *Soit $S = (S_1,S_2,...,S_n)$ une source composée à n composantes.*

$$\begin{aligned} H(S_1,S_2...,S_n) \quad = \quad & H(S_n|S_1,S_2,...,S_{n-1}) + H(S_{n-1}|S_1,S_2,...,S_{n-2}) \\ & +... + H(S_3|S_1,S_2) + H(S_2|S_1) + H(S_1) \end{aligned}$$

Le théorème est facile à retenir si on interprète l'entropie conditionnelle comme information supplémentaire : l'information totale délivrée par la source S est l'information délivrée par S_1, plus l'information supplémentaire délivrée par S_2, plus etc.... plus l'information supplémentaire délivrée par S_n.

4.3 ⋆ *Traitement de l'Information*

Le traitement de l'information dans un ordinateur est en principe déterministe. Quand une source S_2 est obtenue à partir d'un algorithme appliqué à S_1, il n'y a aucune information supplémentaire quand on observe S_2, pour un observateur qui connaît l'algorithme. Cela nous amène au concept suivant :

Définition 4.2 *Soit $S = (S_1,S_2)$ une source composée. On dit que S_2 se déduit de manière déterministe de S_1, ou encore que S_2 est fonction de S_1 si pour tout $s_1 \in \mathcal{S}_1$ tel que $p_{S_1}(s_1) > 0$ il existe un unique $s_2 \in \mathcal{S}_2$ tel que $p_{S_2|S_1}(s_2|s_1) = 1$.*

Dire que S_2 se déduit de manière déterministe de S_1 est équivalent à dire qu'il existe une application $f : \mathcal{A}_1 \to \mathcal{A}_2$ telle que $p_{S_2|S_1}(f(s_1)|s_1) = 1$, c'est à dire que S_2 se déduit de S_1 par une application.

Théorème 4.5 *S_2 est fonction de S_1 si et seulement si $H(S_2|S_1) = 0$.*

Q. 27. Soit $S = (S_1,S_2)$ une source composée.

1. Si $H(S_2|S_1) = H(S_2)$ que peut-on conclure ?
2. Même question avec $H(S_2|S_1) = H(S_1)$.

Q. 28. Prouvez le Théorème 4.4.

Q. 29. Soit $S = (S_1,S_2)$ une source composée. Parmi les égalités ou inégalités suivantes, dire celles qui sont toujours vraies :

1. $H(S_1,S_2) = H(S_1)+H(S_2|S_1)$
2. $H(S_1,S_2) = H(S_2)+H(S_2|S_1)$
3. $H(S_1) \geq H(S_1,S_2)$
4. $H(S_1,S_2) = H(S_1)+H(S_2)$
5. $H(S_2|S_1) \geq 0$
6. $H(S_1,S_2) \geq H(S_1)+H(S_2)$
7. $H(S_1|S_2) \leq H(S_1)$
8. $H(S_1,S_2) = H(S_1)+H(S_1|S_2)$
9. $H(S_1) \leq H(S_1,S_2)$
10. $H(S_1|S_2) \leq H(S_2)$
11. $H(S_2) \leq H(S_1,S_2)$
12. $H(S_2|S_1) \leq H(S_2)$
13. $H(S_2) \geq H(S_1,S_2)$
14. $H(S_1,S_2) \leq H(S_1)+H(S_2)$
15. $H(S_1,S_2) = H(S_2)+H(S_1|S_2)$

Preuve : (1) (S_2 est fonction de S_1)$\Rightarrow$ ($H(S_2|S_1) = 0$) : $H(S_2|S_1 = s_1) = 0$ pour tout $s_1 \in \mathcal{A}_1$ (Théorème 1.1) donc $H(S_2|S_1) = 0$.

(2) ($H(S_2|S_1) = 0$) $\Rightarrow$ (S_2 est fonction de S_1) :

$$0 = H(S_2|S_1) = \sum_{s_1 \in \mathcal{A}_1} p_{S_1}(s_1) H(S_2|S_1 = s_1)$$

et chacun des termes de la somme est ≥ 0, donc chaque terme est nul. Soit s_1 tel que $p_{S_1}(s_1) > 0$; donc $H(S_2|S_1 = s_1) = 0$, et donc (Théorème 1.1) il existe un s_2 tel que $p_{S_2|S_1}(s_2|s_1) = 1$; cet s_2 est unique car s'il y en avait un deuxième la somme des probabilités conditionnelles serait au moins 2, ce qui est impossible. □

Comme corollaire du Théorème 4.5 nous avons :

Théorème 4.6 (*Traitement de l'Information***)** *Si S_2 est fonction de S_1 alors*

$$H(S_1, S_2) = H(S_1) \tag{4.2}$$
$$H(S_2) \leq H(S_1) \tag{4.3}$$

On a bien sûr le même résultat en inversant l'ordre : si S_1 est fonction de S_2, alors $H(S_1, S_2) = H(S_2)$ et $H(S_1) \leq H(S_2)$.

Intuitivement, l'inégalité du traitement de l'information explique ce qui se passe quand on applique un algorithme à un message S_1 pour produire un message S_2. S_2 n'apporte aucune information (si on connaît l'algorithme), donc l'entropie de $S = (S_1, S_2)$ est égale à celle de S_1, Eq.(4.2). D'autre part, S_2 ne peut contenir que de l'information déjà présente dans S_1 ; donc son entropie est au mieux égale à celle de S_1, Eq.(4.2).

Preuve : Par les Théorèmes 4.5 et 4.1, $H(S_1, S_2) = H(S_1) + H(S_2|S_1) = H(S_1)$, ce qui prouve Eq.(4.2). Eq.(4.3) est alors une conséquence de Eq.(4.1). □

Exemple 4.4 (Somme de Deux Dés Codée Sur Deux Chiffres, suite) *Soient $S = (S_1, S_2)$ les résultats des deux tirages de dés et $L =$ la somme des deux tirages. Nous savons que $H(S) = 5.17$ bits et $H(L) = 3.27$ bits.*

La somme L se déduit de manière déterministe de S. Appliquons le Théorème 4.6 à la source composée (S, L) dont la première composante est elle même une source composée. Le théorème dit qu'on doit avoir $H(L) \leq H(S)$, ce qui est vérifié.

On suppose ici qu'on peut distinguer les deux dés, par exemple l'un est rouge et l'autre est vert.

Une conséquence immédiate est que si est S_2 est fonction de S_1 et réciproquement, alors S_1 et S_2 contiennent la même information, donc ont même entropie :

Dire "S_2 est fonction de S_1 et réciproquement" équivaut à dire "S_2 se déduit de S_1 par une application *bijective*".

Théorème 4.7 *Soit $S = (S_1, S_2)$ une source composée telle que S_2 se déduit de manière déterministe de S_1, et vice versa, S_1 se déduit de manière déterministe de S_2. Alors $H(S_1) = H(S_2) = H(S_1, S_2)$.*

Q. 30. Prouvez le Théorème 4.7.

Exemple 4.5 (Le Vélo d'Anne, suite) *Bernard pose à Anne des questions dont la réponse ne peut être que "oui" ou "non". Le but de Bernard*

est de deviner le numéro du cadenas d'Anne, qui est un nombre de quatre chiffres décimaux. Bernard a le droit de poser n questions au maximum.

Nous pouvons modéliser les réponses d'Anne comme une source $S = (S_1, S_2, ..., S_n)$*, où* S_k *est la réponse d'Anne à la k-ième question. Il se peut que Bernard pose moins de n questions, auquel cas nous donnons la valeur "oui" pour les réponses aux questions non posées. Imaginons une source* S' *qui délivre* $(X, S_1, ..., S_n)$ *où X représente le numéro de cadenas ; bien sûr cette source ne peut pas être observée par Bernard, qui ne peut observer que la source marginale* $S = (S_1, ..., S_n)$*.*

Par le Théorème de Traitement de l'Information, S est une fonction déterministe de X (nous supposons qu'Anne ne triche pas ; les réponses $S_1, ..., S_n$ *sont entièrement déterminées si le numéro de cadenas X est connu). Donc*

$$H(X, S) = H(X)$$

D'autre part, par les Théorèmes 4.1, 1.3 et 1.4 :

$$\begin{aligned} H(X, S) &= H(X|S) + H(S) \\ H(S) &\leq H(S_1) + ... + H(S_n) \leq n \end{aligned}$$

donc

$$H(X|S) \geq H(X) - n$$

L'information supplémentaire $H(X|S)$ *est celle qui manque à Bernard pour trouver le numéro du cadenas. Si les questions sont bien conçues, cette information est petite, si elles sont mal conçues, cette information est grande. Les questions sont bien conçues si l'entropie* $H(S)$ *est aussi grande que possible, ce qui a lieu quand les réponses aux question posées sont équiprobables. Donc Bernard doit essayer de poser des questions dont les probabilités de réponse soient proches de* 0.5*.*

Q. 31. Existe-t-il un système de 14 questions qui permette à Bernard de trouver le numéro d'Anne à coup sûr ?

Supposons qu'Anne a choisi son numéro uniformément parmi les 10′000 *possibles, donc* $H(X) = \log_2(10'000) = 4\log_2(10) = 13.287$*. Nous avons alors*

$$H(X|S) \geq 13.287 - n$$

En particulier si $n \leq 13$ *l'entropie conditionnelle est* > 0 *donc Bernard ne peut pas être certain de trouver le numéro en 13 questions ou moins.*

Q. 32. Soit $S = (S_1, S_2)$ une source composée.

1. Si $H(S_1, S_2) = H(S_1)$, que peut-on conclure ? Même question avec :
2. $H(S_2|S_1) = 0$

5
Théorème du Codage de Source

NOUS ARRIVONS MAINTENANT AU BOUT de nos efforts et pouvons comprendre comment mesurer l'information d'une source réelle (plutôt que d'un robot-page). Nous allons considérer un modèle de source plus complexe, celui de "source étendue", qui modélise mieux la production d'un texte en français. Pour une source étendue, le concept-clé est celui d'"entropie par symbole", défini à partir de l'entropie conditionnelle.

5.1 Sources Etendues

Jusqu'ici nous avons considéré des source étendues avec un nombre fixé de composantes. Pour aller plus loin, en particulier pour modéliser des textes écrits par des auteurs plutôt que des robots-pages, il nous faut pouvoir considérer des sources produisant un nombre indéfini de symboles. Pour cela, nous introduisons le concept de "source étendue", que nous pouvons imaginer comme une machine à produire, sur demande, pour tout n, une suite (appelée *bloc*) de n symboles définis sur le même alphabet. Pour être cohérent, il faut que la densité de probabilité du bloc des n premières observations soit la même quel que soit le nombre total d'observations.

En bref, nous pouvons dire qu'une source étendue modélise un nombre illimité de symboles du même alphabet.

Définition 5.1 (*Source Etendue*) *Une source étendue $\mathcal{S}$ sur l'alphabet $\mathcal{A}$ est la donnée d'une famille de sources S^n définies pour tout $n = 1, 2, 3, ...$ telles que*

1. *S^n est une source à n composantes, sur l'alphabet $\mathcal{A} \times ... \times \mathcal{A}$; notons p_{S^n} sa densité de probabilité ;*
2. *la densité de probabilité de la source constituée des n premières sources marginales de S^{n+k} est égale à p_{S^n}, pour tous $k \geq 1$ et $n \geq 1$.*

On note $\mathcal{S} = (S_1, S_2, ..., S_k, ...)$ la source étendue, où S_k est la k-ième marginale, et $S^n = (S_1, ..., S_n)$ la source à n composantes qui en dérive. On dit aussi que S^n est un "bloc" de n symboles de la source étendue (rappelons que chaque symbole du bloc est élément du même alphabet $\mathcal{A}$).

Une définition apparemment plus simple serait de définir une source étendue comme la donnée d'un alphabet $\mathcal{A}$ et d'une densité de probabilité définie sur l'ensemble des suites *infinies* d'éléments de $\mathcal{A}$. Mais cela nous emmènerait un peu loin, car il faudrait faire une théorie des probabilités sur des ensembles infinis, ce qui est plus complexe et n'est pas nécessaire pour ce cours.

Exemple 5.1 (Pile ou Face) *$\mathcal{S}_{PF}$ modélise des tirages successifs d'une pièce non biaisée ; S_k représente le résultat du k-ième tirage. La densité de probabilité de p_{S^n} est définie par*

$$p_{S^n}(s_1,...,s_n) = \frac{1}{2^n}, \quad \forall (s_1,...,s_n) \in \{\text{“}P\text{”},\text{“}F\text{”}\}^n$$

c'est à dire que, pour n fixé, tous les blocs de n symboles sont équiprobables.

Il est intuitivement clair que $\mathcal{S}_{PF}$ satisfait à la définition de source étendue ; l'item 1 est clair ; pour l'item 2, il faut se demander si la densité de probabilité des n premières observations est la même quel que soit le nombre total d'observations. La réponse semble évidemment oui.

⋆ Voici une preuve formelle que $\mathcal{S}_{PF}$ satisfait à la définition de source étendue ; il faut montrer que l'item 2 est vrai. La densité des n premières marginales de S^{n+k} est :

$$\begin{aligned} & p_{S_1^{n+k},...,S_n^{n+k}}(s_1,...,s_n) \stackrel{\text{def}}{=} \\ & \sum_{s_{n+1},...,s_{n+k}} p_{S^{n+k}}(s_1,...,s_n,s_{n+1},...,s_{n+k}) \\ & = \overbrace{\sum_{s_{n+1},...,s_{n+k}} \frac{1}{2^{n+k}}}^{2^k \text{ termes}} = 2^k \frac{1}{2^{n+k}} = \frac{1}{2^n} \\ & = p_{S^n}(s_1,...,s_n) \end{aligned}$$

ce qui prouve que $\mathcal{S}_{PF}$ satisfait l'item 2 de la définition.

De la même façon, pour $\mathcal{S}_{bm}$, notons que, d'après Eq.(5.1) la densité de S^{n+k} satisfait :

$$\begin{aligned} & p_{S^{n+k}}(s_1,...,s_{n+k}) = \\ & p_{S^n}(s_1,...,s_n) p_{S_2^{k+1},...,S_{k+1}^{k+1}|S_1^{k+1}}(s_{n+1},...,s_{n+k}|s_n) \end{aligned}$$

Donc

$$\begin{aligned} & p_{S_1^{n+k},...,S_n^{n+k}}(s_1,...,s_n) \stackrel{\text{def}}{=} \\ & \sum_{s_{n+1},...,s_{n+k}} p_{S^{n+k}}(s_1,...,s_n,s_{n+1},...,s_{n+k}) \\ & = p_{S^n}(s_1,...,s_n) \times \\ & \overbrace{\sum_{s_{n+1},...,s_{n+k}} p_{S_2^{k+1},...,S_{k+1}^{k+1}|S_1^{k+1}}(s_{n+1},...,s_{n+k}|s_n)}^{1 \text{ car c'est une proba}} \\ & = p_{S^n}(s_1,...,s_n) \end{aligned}$$

ce qui prouve que $\mathcal{S}_{bm}$ satisfait l'item 2 de la définition.

Exemple 5.2 (Beau ou Mauvais) *Le temps qu'il fait jour après jour n'est pas indépendant d'un jour à l'autre, il a tendance à se répéter souvent (mais pas toujours). Supposons que le temps qu'il fait un jour k est, avec probabilité $q = 6/7$, le même que le jour précédent. Supposons que le temps au jour $k = 1$ est équiprobable. Nous modélisons cela par une source étendue $\mathcal{S}_{bm}$ sur l'alphabet $\mathcal{A} = \{b, m\}$ (“**b**eau”, “**m**auvais”). La densité de probabilité de S^4, par exemple, est telle que :*

$$p_{S^4}(bbbm) = 0.5 \times q \times q \times (1-q)$$

En général nous avons

$$p_{S^n}(s_1,...,s_n) = 0.5 q^{n-1-c(s_1,...,s_n)}(1-q)^{c(s_1,...,s_n)} \tag{5.1}$$

où $c(s_1,...,s_n) \stackrel{\text{def}}{=}$ le nombre de changements dans la suite $(s_1,...,s_n)$; ainsi $c(bbbm) = 1$, $c(bbbb) = 0$ et $c(bmbm) = 3$.

Notons aussi que, par construction de $\mathcal{S}$, la densité conditionnelle du temps d'aujourd'hui sachant les temps des jours passés ne dépend que du temps d'hier. En termes mathématiques :

$$p_{S_n^n|S_1^n,...,S_{n-1}^n}(s_n|s_1,...,s_{n-1}) = \begin{cases} q & si\ s_n = s_{n-1} \\ (1-q) & si\ s_n \neq s_{n-1} \end{cases}$$

Comme dans l'exemple précédent, il est intuitivement clair que $\mathcal{S}_{bm}$ satisfait, par construction, à la définition de source étendue. Quelle est la probabilité u_k qu'il fasse beau au jour k ? Montrons que $u_k = 0.5$ pour tout $k = 1, 2, 3....$

(Etape d'initialisation) Pour $k = 1$, $u_1 = 0.5$ par construction.

(Etape de récurrence) Supposons que $u_1 = u_2 = ... = u_k = 0.5$. Soit $v_k = 1 - u_k$ la probabilité qu'il fasse mauvais au jour k ; nous avons :

$$u_{k+1} = qu_k + (1-q)v_k = 0.5q + 0.5 - 0.5q = 0.5$$

donc la propriété est vraie pour $k + 1$.

Pour prouver un résultat du type

$$\forall n \in \{n_0, n_0+1, ...\}, P(n)$$

on peut utiliser un raisonnement par *récurrence*. Cela consiste à prouver :

1. (étape initiale) : $P(n_0)$ est vraie.
2. (étape de récurrence) : pour tout $n \in \{n_0, n_0+1, ...\}$, si $P(n)$ est vraie alors $P(n+1)$ est vraie.

Par exemple, soit $S(n) = 1 + 2 + ... + n$ défini pour $n \geq 1$, et soit $P(n)$ la phrase

$$S(n) = \frac{n(n+1)}{2}$$

Nous pouvons montrer par récurrence que $P(n)$ est vraie pour tout $n = 1, 2, 3, ...$:

1. (étape initiale) : $P(1)$ est vraie car $S(1) = 1 = \frac{1(1+1)}{2}$
2. (étape de récurrence) : Supposons $P(1), P(2), ..., P(n)$ donc en particulier $S(n) = \frac{n(n+1)}{2}$. Alors

$$\begin{aligned} S(n+1) &= 1 + ... + n + (n+1) \\ &= S(n) + (n+1) \\ &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

donc $P(n+1)$ est vraie.

Donc $P(n)$ est vraie pour tout $n \in \{1, 2, 3, ...\}$.

Exemple 5.3 (Vert ou Bleu) *Au pays des Schtroumpfs, il y a deux partis : les bleus (nationalistes) et les verts (écologistes). Quand un Schtroumpf atteint l'âge de voter, il tire à pile ou face et se prononce une fois pour toutes pour l'un des deux partis. Pour le reste de sa vie, il conservera ce choix.*

Nous pouvons modéliser les votes d'un Schtroumpf par une source étendue $\mathcal{S}_{VB}$ sur l'alphabet $\mathcal{A} = \{V, B\}$. La densité de probabilité de S^n est :

$$\begin{aligned} p_{S^n}(VVV...V) &= 0.5 \\ p_{S^n}(BBB...B) &= 0.5 \end{aligned}$$

et toutes les autres suites de symboles ont une probabilité nulle. Ici aussi, il est clair que $\mathcal{S}_{VB}$ satisfait, par construction, à la définition de source étendue.

La probabilité qu'un Schtroumpf vote B à la k-ième votation est 0.5*, c'est à dire que les symboles de la k-ième source marginale sont équiprobables.*

Pour les trois sources étendues des exemples précédents, la probabilité que le k-ième symbole prenne une des deux valeurs possibles vaut 0.5. Cependant, les trois sources sont très différentes, comme l'illustre la Figure 5.1 et nous verrons que nous pouvons les comprimer avec des rapports de compression différents.

F	m	V
F	b	V
P	b	V
F	m	V
F	m	V
P	m	V
P	m	V
F	m	V
F	b	V
F	m	V
P	m	V
F	b	V
F	m	V
P	m	V
F	m	V
P	m	V
P	m	V
F	b	V
F	b	V
F	m	V
F	m	V
P	m	V
F	m	V
F	b	V
F	b	V
F	b	V
F	b	V
P	b	V
F	b	V
P	b	V
F	b	V
P	b	V
P	b	V
P	b	V
P	b	V
F	b	V
F	b	V
P	b	V
F	m	V
P	m	V

FIGURE 5.1: Exemple de suites de 40 symboles produits par les trois sources étendues "Pile ou Face", "Beau ou Mauvais" et "Vert ou Bleu".

5.2 *Entropie par Symbole d'une Source Etendue Régulière*

Pour une source étendue, nous pouvons calculer l'entropie du k-ième symbole, et appliquer un code efficace pour chaque symbole, dont la longueur moyenne sera proche de cette entropie. Cependant, nous allons voir que cette méthode n'est pas la plus efficace : il est plus malin de considérer des blocs de n symboles. Le concept essentiel devient alors l'entropie *par symbole*, qui est définie comme la quantité d'information moyenne supplémentaire obtenue quand on reçoit un symbole. Pour éviter des complications inutiles, nous avons besoin de poser une hypothèse technique :

Définition 5.2 *La source étendue $\mathcal{S}$ est dite régulière si les deux limites*

1. $H(\mathcal{S}) \stackrel{\text{def}}{=} \lim_{n\to+\infty} H(S_n)$ *et*
2. $H^*(\mathcal{S}) \stackrel{\text{def}}{=} \lim_{n\to+\infty} H(S_n|S_1, S_2, ..., S_{n-1})$

existent et sont finies.

Pour une source étendue régulière $\mathcal{S}$, $H(\mathcal{S})$ est appelée l'entropie d'un symbole et $H^*(\mathcal{S})$ est appelée l'entropie par symbole.

Toutes les sources étendues utilisées en pratique pour modéliser les sources d'information sont régulières. Une raison simple est que toutes les sources stationnaires sont régulières, et que la plupart des modèles de source sont stationnaires.

Exemple 5.4 (Les Trois Sources) *Examinons si les trois sources étendues des exemples précédents sont régulières, et si oui, calculons leurs entropies d'un symbole et par symbole.*

Pile ou Face. *D'une part la densité de probabilité de S_n est la même pour tout n donc $H(S_n) = H(S_1)$, i.e. la suite $H(S_n)$ est une suite constante, donc elle a une limite. L'entropie d'un symbole existe donc et est $H(\mathcal{S}_{PF}) = H(S_1) = 1$ bit.*

On dit que la source étendue $\mathcal{S}$ est stationnaire si, pour tout n fixé, la densité de probabilité de $(S_{k+1}, ..., S_{k+n})$ est la même pour toutes les valeurs de $k \geq 0$.

En d'autres termes, la stationnarité signifie que les blocs

$$\begin{array}{cccccc} (S_1, & S_2, & ..., & S_n) & & \\ & (S_2, & S_3, & ..., & S_{n+1}) & \\ & & ... & & & \\ & & & (S_{k+1}, & S_{k+2}, & ..., & S_{n+k}) \end{array}$$

ont toutes la même densité de probabilité, quel que soit $k \geq 0$. La source ne change pas son comportement moyen quand le temps passe, elle ne vieillit ni ne rajeunit.

On peut montrer que les trois sources "Pile ou Face", "Beau ou Mauvais" et "Vert ou Bleu" sont stationnaires.

Pour déterminer si l'entropie par symbole existe, il nous faut l'entropie conditionnelle. Or

$$H(S_n|S_1,...,S_{n-1}) = H(S_n) = 1 \text{ bit}$$

car S_n est indépendante de $S_1,...,S_{n-1}$. Donc, évidemment, la limite existe et vaut aussi 1 bit. Donc l'entropie par symbole est $H^*(\mathcal{S}_{PF}) = 1$ bit. Cette source est régulière, et ses entropies d'un *symbole et* par *symbole sont égales.*

Q. 33. ⋆ Démontrez qu'une source stationnaire est régulière.

Beau ou Mauvais. *Nous avons montré que la probabilité qu'il fasse beau au jour n est 0.5, donc $H(S_n) = 1$ bit. La suite $H(S_n)$ est une suite constante, donc elle a une limite. L'entropie d'un symbole existe donc et est $H(\mathcal{S}_{bm}) = H(S_1) = 1$ bit.*

Sachant que $(S_1,...,S_{n-1} = s_1,...,s_{n-1})$, S_n vaut "beau" ou "mauvais" avec probabilités q et $1-q$, ou l'inverse. Dans tous les cas :

$$H(S_n|S_1 = s_1,...,S_{n-1} = s_{n-1}) = h(q)$$

où la fonction h est l'entropie d'une source binaire, Eq.(1.1). Cela ne dépend pas de $(s_1,...,s_{n-1})$, donc, en prenant la moyenne nous obtenons :

$$H(S_n|S_1,...,S_{n-1}) = h(q)$$

aussi. Donc l'entropie conditionnelle $H(S_n|S_1,...,S_{n-1})$ est la même pour tout n, donc elle converge et l'entropie par symbole existe. Elle vaut :

$$H^*(\mathcal{S}_{bm}) = h(q) = 0.592 \text{ bit}$$

Cette source est régulière, et son entropie d'un *symbole est plus grande que son entropie* par *symbole.*

Vert ou Bleu. *Ici $S_n = S_1$ pour tout n donc comme pour les deux autres sources, la suite $H(S_n)$ est une suite constante égal à 1 bit, donc l'entropie d'un symbole existe et vaut $H(\mathcal{S}_{VB}) = H(S_1) = 1$ bit.*

La densité de probabilité conditionnelle de S_n sachant que $(S_1,...,S_{n-1} = s_1,...,s_{n-1})$ est la densité de probabilité d'une source certaine, et donc

$$H(S_n|S_1 = s_1,...,S_{n-1} = s_{n-1}) = 0$$

Donc $H(S_n|S_1,...,S_{n-1}) = 0$ et la limite existe, avec $H^*(\mathcal{S}_{VB}) = 0$. Cette source est régulière, et son entropie d'un *symbole est plus grande que son entropie* par *symbole.*

	H	H^*
Pile ou Face	1 bit	1 bit
Beau ou Mauvais	1 bit	0.592 bit
Vert ou Bleu	1 bit	0 bit

TABLE 5.1: Entropie d'un symbole (H) et par symbole (H^*) de trois sources binaires.

Exemple 5.5 (Source Non Régulière) *Tirons à pile ou face une fois par jour, avec une pièce biaisée qui dépend du jour de la semaine. La pièce des dimanches retourne "P" avec probabilité 1, celle des lundis avec probabilité 1/2, des mardis avec probabilité 1/3, etc... celle des samedi avec probabilité 1/7.*

Soit $\mathcal{S} = (S_1, S_2, ...)$ une suite infinie de tirages commençant un dimanche. Alors

$$H(S_1) = 0, H(S_2) = 1, ..., H(S_7) = 0.592$$
$$H(S_8) = 0, H(S_9) = 1, ..., H(S_{14}) = 0.592$$
$$...$$
$$H(S_{7n+1}) = 0, H(S_{7n+2}) = 1, ..., H(S_{7n+7}) = 0.592$$

de sorte que la suite $H(S_n)$ ne converge pas. Cette source n'est pas régulière.

Dans les exemples précédents de sources régulières, l'entropie par symbole est majorée par l'entropie d'un symbole, et il n'y a égalité que dans le cas "Pile ou Face", où les sources marginales sont indépendantes. C'est une illustration du résultat suivant :

Théorème 5.1 *Pour une source étendue régulière :*

1. *l'entropie* par *symbole est* $\leq$ *l'entropie* d'un *symbole, i.e.* $H^*(\mathcal{S}) \leq H(\mathcal{S})$;
2. *si les sources marginales sont indépendantes alors il y a égalité.*

☺*Preuve :* (1) Soit $\mathcal{S} = (S_1, S_2, ...)$ la source étendue régulière. L'entropie d'un symbole est $h = \lim_{n\to\infty} H(S_n)$ et l'entropie par symbole est $H^*(\mathcal{S}) = \lim_{n\to\infty} H(S_n|S_1, ...S_{n-1})$. Or conditionner réduit l'entropie (Théorème 4.2) donc

$$H(S_n|S_1, ...S_{n-1}) \leq H(S_n)$$

donc par passage à la limite $H^*(\mathcal{S}) \leq H(\mathcal{S})$.

(2) indépendance $\Rightarrow$ égalité : Les sources sources marginales sont indépendantes par hypothèse, donc $H(S_n|S_1, ...S_{n-1}) = H(S_n)$ et par passage à la limite : $H(\mathcal{S}) = H^*(\mathcal{S})$. ☺□

Exemple 5.6 (Robot-Page contre Flaubert) *L'entropie du robot page est l'entropie d'un symbole de la langue française et vaut environ* 3.95 *bits. L'entropie par symbole peut être calculée en estimant directement* $u_n = H(S_n|S_1, ..., S_{n-1})$. *Pour cela, on commence par faire l'hypothèse que les textes écrits, par exemple un livre de Gustave Flaubert, peuvent être modélisés par une source régulière (car stationnaire). Cela est valide si on estime que la langue ne change pas au cours d'un livre, ce qui est vrai en général.*

Ensuite on calcule u_n *pour chaque* $n = 1, 2, ...$ *fixé en estimant*

1. *d'une part les densités de probabilités conditionnelles* $p_{S_n|S_1,...,S_{n-1}}(s_n|s_1, ..., s_{n-1})$ *pour chaque* $(s_1, ..., s_{n-1})$ *et chaque* s_n ; *cela permet de calculer* $H(S_n|S_1 = s_1, ..., S_{n-1} = s_{n-1})$ *pour chaque* $(s_1, ..., s_{n-1})$;
2. *d'autre part les densités de probabilités marginales* $p_{S_1,...,S_{n-1}}(s_1, ..., s_{n-1})$ *pour chaque* $s_1, ..., s_{n-1}$; *en combinant avec ce qui précède cela permet de calculer*

$$u_n = H(S_n|S_1, ..., S_{n-1}) = \sum_{(s_1,...,s_{n-1})} p_{S_1,...,S_{n-1}}(s_1, ..., s_{n-1}) H(S_n|S_1 = s_1, ..., S_{n-1} = s_{n-1})$$

On fait cela pour plusieurs n et on cherche la limite de u_n *quand n croît, ce qui donne l'entropie par symbole. Shannon l'a fait pour la langue anglaise* [1] *et obtenu l'entropie par caractère* $h^* \approx 1.5$. *Cela a été fait par d'autres pour la langue française* [2] *et on trouve l'entropie par caractère* $h^* \approx 1$. *La valeur dépend des textes et des auteurs choisis et des conventions utilisées pour l'alphabet (par exemple, avec ou sans le caractère espace, avec ou sans les minuscules, les caractères accentués, etc.).*

Q. 34. Si pour une source régulière l'entropie par symbole est strictement inférieure à l'entropie d'un symbole que peut-on conclure ?

Q. 35. ⋆ Démontrez que si pour une source stationnaire l'entropie par symbole et l'entropie d'un symbole sont égales, alors les marginales sont indépendantes

Q. 36. Si pour une source régulière l'entropie par symbole est nulle, que peut-on conclure ?

Voici un exemple pour comprendre comment sont estimées les densités de probabilités conditionnelles et marginales de l'Exemple 5.6. Supposons que nous ayons un texte de $N = 2173$ caractères, dans lequel nous avons observé 53 fois le préfixe QU. Nous avons trouvé les résultats suivants pour le successeur de QU :

caractère	occurrences
A	23
E	16
I	9
0	5
autres	0
total	53

Nous obtenons alors

$$\begin{aligned} p_{S_3|S_1,S_2}(A|QU) &= {}^{23}/_{53} \approx 0.434 \\ p_{S_3|S_1,S_2}(E|QU) &= {}^{16}/_{53} \approx 0.302 \\ p_{S_3|S_1,S_2}(I|QU) &= {}^{9}/_{53} \approx 0.170 \\ p_{S_3|S_1,S_2}(O|QU) &= {}^{5}/_{53} \approx 0.094 \\ p_{S_3|S_1,S_2}(s_n|QU) &= 0 \text{ sinon} \end{aligned}$$

d'où nous calculons que $H(S_3|S_1 = Q, S_2 = U) = 1.800$ bits. Notons qu'ici nous utilisons implicitement l'hypothèse que la source qui modélise le texte est régulière.

Il faut ensuite faire cela pour tous les préfixes (s_1, s_2) de $n = 2$ lettres. Puis, nous calculons les probabilités $p_{S_1,S_2}(s_1, s_2)$ pour tous les préfixes (s_1, s_2) (il y en a 2^{26}, nous ignorons les espaces dans cet exemple). Nous avons un texte de $N = 2173$ caractères, il y a donc $N - 2 = 2171$ suites possibles de 2 caractères qui peuvent être suivis d'un caractère : nous avons observé 53 fois la suite QU donc nous prenons $p_{S_1,S_2}(QU) = {}^{53}/_{2171} \approx 0.00244$. Il faut faire ces deux étapes pour tous les préfixes (s_1, s_2) de $n = 2$ lettres, ce qui permet de calculer u_2.

1. C.E. Shannon. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64, 1951
2. Alexis Fabre-Ringborg and Sébastien Saunier. Entropie du français. `http://cb.sogedis.fr/files/entropie/Entropie_Francais_FabreRingoborg_Saunier.pdf`, 2003

5.3 Théorème du Codage de Source

Le moment est venu d'introduire la botte secrète du codage de source : le codage par bloc. L'idée est simple : au lieu de coder un symbole de la source S_1, on code un bloc de n symboles, c'est à dire que nous codons la source $S^n = (S_1, ..., S_n)$. Le premier résultat remarquable concerne l'entropie de S^n, qui, pour n grand, est à peu près égale à n fois l'entropie par symbole :

Théorème 5.2 (Entropie d'un Bloc) *Soit $\mathcal{S}$ une source étendue régulière. Alors*

$$\lim_{n\to+\infty} \frac{H(S_1, ..., S_n)}{n} = H^*(\mathcal{S})$$

où $H^*(\mathcal{S})$ est l'entropie par symbole et $H(S_1, ..., S_n)$ l'entropie d'un bloc de n symboles.

☺*Preuve :* Nous appliquons le théorème de Cesàro (que nous ne démontrons pas) à la formule de calcul incrémental de l'entropie conditionnelle :

$$H(S^n) = H(S_1, ..., S_n) = H(S_n|S_1, ..., S_{n-1}) + H(S_{n-1}|S_1, ..., S_{n-2}) \\ +... + H(S_3|S_1, S_2) + H(S_2|S_1) + H(S_1)$$

Posons $u_n = H(S_n|S_1, ..., S_{n-1})$ pour $n \geq 2$ et $u_1 = H_1$. Nous avons donc

$$\frac{H(S^n)}{n} = \frac{u_1 + ... + u_n}{n}$$

Or $\lim_{n\to+\infty} u_n = H^*(\mathcal{S})$ donc, par le théorème de Cesàro,

$$\lim_{n\to+\infty} \frac{H(S^n)}{n} = H^*(\mathcal{S})$$

☺□

Théorème de Cesàro : si une suite u_n de nombres réels converge vers une limite ℓ (finie ou infinie) quand $n \to \infty$, alors la moyenne $v_n = \frac{u_1+...+u_n}{n}$ converge aussi vers ℓ.

Exemple 5.7 (Les Trois Sources, suite) *Pour chacune des trois sources étendues régulières des exemples précédents, nous pouvons calculer exactement l'entropie d'un bloc.*

Pile ou Face. *Puisque les marginales sont indépendantes, nous avons*

$$H(S_1, ..., S_n) = H(S_1) + ... + H(S_n) = n \text{ bits}$$

car $H(S_1) = ... = H(S_n) = 1$. Nous avons $\frac{H(S_1,...,S_n)}{n} = 1$ donc a fortiori $\lim_{n\to+\infty} \frac{H(S_1,...,S_n)}{n} = 1 = H^*(\mathcal{S}_{PF})$ et le Théorème 5.2 est vérifié.

Beau ou Mauvais. *Nous avons calculé que $H(S_n|S_1 = s_1, ..., S_{n-1} = s_{n-1}) = h(q)$ où la fonction h est l'entropie d'une source binaire, Eq.(1.1). Donc*

$$H(S_1, ..., S_n) = H(S_n|S_1, ..., S_{n-1}) + H(S_{n-1}|S_1, ..., S_{n-2}) \\ +... + H(S_3|S_1, S_2) + H(S_2|S_1) + H(S_1) \\ = \overbrace{h(q) + ... + h(q)}^{(n-1) \text{ fois}} + 1 = (n-1)h(q) + 1 = 0.592n + 0.408$$

Nous avons $\lim_{n\to+\infty} \frac{H(S_1,...,S_n)}{n} = 0.592 = H^*(\mathcal{S}_{bm})$ et le Théorème 5.2 est vérifié.

	H	H^*	H^n
Pile ou Face	1	1	n
Beau ou M.	1	0.592	$0.592n + 0.408$
Vert ou Bleu	1	0	1

TABLE 5.2: Entropie (exprimée en bits) d'un symbole (H), par symbole (H^*) et d'un bloc de n symboles (H^n) de trois sources binaires.

Vert ou Bleu. *La source $(S_1, ..., S_n)$ prend deux valeurs (VV...V et BB...B) avec probabilités 0.5 donc*

$$H(S_1, ..., S_n) = H(S_1) = 1 \text{ bit}$$

Nous avons $\frac{H(S_1,...,S_n)}{n} = \frac{1}{n}$ et $\lim_{n\to+\infty} \frac{H(S_1,...,S_n)}{n} = 0 = H^*(\mathcal{S}_{VB})$; le Théorème 5.2 est vérifié.

Considérons maintenant un bloc de n symboles d'une source étendue régulière. Nous pouvons l'encoder en utilisant un code binaire efficace, par exemple un code de Shannon-Fano ou un code de Huffman. Nous avons maintenant réuni tous les éléments pour prouver que, pour de tels codes et pour n grand, le nombre moyen de symboles de code par symbole de source approche l'entropie par symbole d'aussi près qu'on veut. Ce résultat est connu comme le "premier théorème de Shannon", ou le "théorème du codage de source".

Théorème 5.3 (Codage de Source) *Soit $\mathcal{S}$ une source étendue régulière et $H^*(\mathcal{S})$ son entropie par symbole. Soient L^n_{SF}, respectivement L^n_H, les longueurs moyennes des codes D-aires de Shannon-Fano, respectivement Huffman, pour un bloc de n symboles de la source. Alors*

$$\lim_{n\to+\infty} \frac{L^n_H}{n} = \lim_{n\to+\infty} \frac{L^n_{SF}}{n} = \frac{H^*(\mathcal{S})}{\log_2(D)}$$

☺*Preuve :* Par les deux inégalités de l'entropie :

$$\frac{H(S_1, ..., S_n)}{\log_2(D)} \leq L^n_H \leq L^n_{SF} < \frac{H(S_1, ..., S_n)}{\log_2(D)} + 1$$

Nous nous intéressons au nombre moyen de symboles de code par symbole de source, $\frac{L^n_H}{n}$ et $\frac{L^n_{SF}}{n}$; il s'en suit que :

$$\frac{H(S_1, ..., S_n)}{n\log_2(D)} \leq \frac{L^n_H}{n} \leq \frac{L^n_{SF}}{n} < \frac{H(S_1, ..., S_n)}{n\log_2(D)} + \frac{1}{n} \qquad (5.2)$$

Appliquons maintenant le Théorème 5.2, il vient :

$$\lim_{n\to+\infty} \frac{H(S_1, ..., S_n)}{n\log_2(D)} = \frac{H^*(\mathcal{S})}{\log_2(D)}$$
$$\lim_{n\to+\infty} \frac{H(S_1, ..., S_n)}{n\log_2(D)} + \frac{1}{n} = \frac{H^*(\mathcal{S})}{\log_2(D)}$$

Les deux termes extrêmes de Eq.(5.2) convergent vers la même limite quand $n \to +\infty$, donc, par le "critère des deux gendarmes" [3], les termes du milieu aussi. ☺□

Remarque. Pour tout autre méthode de codage à décodage unique, sa longueur moyenne L^n pour un texte de n symboles de la source satisfait $L_n \geq L^n_H$ (puisque un code de Huffman est optimal). Donc toute autre méthode de codage ne peut pas faire mieux que $\frac{H^*(\mathcal{S})}{\log_2(D)}$ symboles de code par symbole de source. Appliquons cela avec $D = 2$: $H^*(\mathcal{S})$ est donc le nombre minimum de bits par symbole de source que n'importe quelle méthode de codage à décodage unique peut atteindre.

3. Y. Biollay, A. Chaabouni, and J. Stubbe. *Savoir-faire en maths: bien commencer ses études scientifiques*. Presses polytechniques et universitaires romandes, 2008. ISBN 2880747791

Le *critère des deux gendarmes* est le suivant : si $u_n \leq v_n \leq w_n$ et $\lim_{n\to+\infty} u_n = \lim_{n\to+\infty} w_n = \ell$, alors $\lim_{n\to+\infty} v_n = \ell$ aussi.

Exemple 5.8 (Les Trois Sources, suite) *Calculons la longueur moyenne d'un code binaire de Huffman pour un bloc de n symboles pour chacune des trois sources étendues régulières.*

Pile ou Face. *Les symboles de la source S^n sont tous équiprobables, il est donc naturel d'essayer un code dont tous les mots de code ont la même longueur. Considérons par exemple le code Γ^n qui traduit "F" en 0 et "P" en 1. La longueur moyenne de ce code est*

$$L(\Gamma^n) = 2^n \times \frac{1}{2^n} \times n = n \text{ bits}$$

Or $H(S^n) = n$ bits donc le code Γ^n atteint la borne inférieure de l'entropie, et donc il est optimal et un code de Huffman a forcément la même longueur moyenne. Nous avons

$$\frac{L(\Gamma_H^n)}{n} = 1 = H^*(\mathcal{S}_{PF})$$

La limite prévue par le Théorème 5.3 est en atteinte pour tout $n \geq 1$.

Beau ou Mauvais. *La longueur moyenne d'un code binaire de Huffman appliqué à un bloc de n symboles satisfait*

$$0.592 + \frac{0.408}{n} \leq \frac{L(\Gamma_H^n)}{n} < 0.592 + \frac{1.408}{n}$$

Pour obtenir la valeur exacte de $L(\Gamma_H^n)$ il faut calculer le code de Huffman, ce qui est facile à faire si n est petit. Pour $n = 6$ nous trouvons le code illustré en Table 5.3 et

$$0.660 \leq \frac{L(\Gamma_H^n)}{n} = 0.6632 < 0.8267$$

Le code de Huffman est très proche de la borne inférieure, et plus n est grand plus il en est proche. D'autre part, plus n est grand plus la borne inférieure est proche de $H^*(\mathcal{S}_{bm}) = 0.592$. En codant par bloc, on gagne donc sur deux tableaux.

Vert ou Bleu. *La source S^n peut émettre deux suites de symboles : $BBBB...B$ et $VVVV...V$, et chaque suite a la même probabilité, égale à 0.5. Donc un code évident pour S^n est donné par $\Gamma^n(BBBB...B) = 0$, $\Gamma^n(VVVV...V) = 1$. Sa longueur moyenne est 1, et aucun code ne peut faire mieux, donc c'est un code optimal. Donc $L(\Gamma^n) = 1$ bit et*

$$\frac{L(\Gamma_H^n)}{n} = \frac{1}{n}$$

La limite prévue par le Théorème 5.3 est 0, ce qui est bien le cas ici.

bbbbbb	2	bbbbbm	5
mbbbbb	5	mbbbbm	7
bmbbbb	7	bmbbbm	9
mmbbbb	5	mmbbbm	7
bbmbbb	7	bbmbbm	10
mbmbbb	10	mbmbbm	12
bmmbbb	7	bmmbbm	10
mmmbbb	5	mmmbbm	7
bbbmbb	7	bbbmbm	10
mbbmbb	9	mbbmbm	12
bmbmbb	12	bmbmbm	13
mmbmbb	9	mmbmbm	12
bbmmbb	7	bbmmbm	10
mbmmbb	9	mbmmbm	11
bmmmbb	7	bmmmbm	9
mmmmbb	5	mmmmbm	7
bbbbmb	7	bbbbmm	5
mbbbmb	10	mbbbmm	7
bmbbmb	12	bmbbmm	10
mmbbmb	10	mmbbmm	7
bbmbmb	12	bbmbmm	10
mbmbmb	13	mbmbmm	12
bmmbmb	12	bmmbmm	10
mmmbmb	10	mmmbmm	7
bbbmmb	7	bbbmmm	5
mbbmmb	9	mbbmmm	7
bmbmmb	12	bmbmmm	9
mmbmmb	9	mmbmmm	7
bbmmmb	7	bbmmmm	5
mbmmmb	9	mbmmmm	7
bmmmmb	7	bmmmmm	5
mmmmmb	5	mmmmmm	2

TABLE 5.3: Source "Beau ou Mauvais" : nombre de bits alloué par le code de Huffman à chaque bloc de $n = 6$ symboles. La longueur moyenne du code est 3.97929 bits.

Q. 37. Considérons les suites des 60 symboles produites par les trois sources de la Figure 5.1. Supposons que nous les encodions avec le code binaire de Huffman pour la source correspondante par blocs de 6 symboles. Quelle est la longueur en bits de la suite encodée, pour chacun des trois cas ?

5.4 Compression et codage de source en pratique

Le code de Huffman est optimal, mais comme nous l'avons vu, il faut pouvoir coder par bloc si l'on veut comprimer efficacement. Si l'alphabet de la source est de grande taille, sa complexité peut devenir grande. la construction de l'arbre binaire peut devenir extrêmement complexe en nombre d'opérations. L'algorithme de

Huffman ne peut donc être utilisé qu'avec des alphabets de source de taille raisonnable (Par exemple, le codage des opérations sur le processeur Intel 432 fait appel à des techniques du type Huffman pour la compression des programmes). De tels alphabets sont obtenus après plusieurs étapes préliminaires de compression utilisant d'autres algorithmes, dont certains sont mentionnés ci-dessous et seront étudiés dans les cours de traitement du signal, des images, de l'audio, ainsi naturellement que dans ceux de théorie de l'information et du codage. Le théorème du codage de source permet alors de vérifier, a posteriori, si une méthode de codage est proche de l'optimum.

Ainsi pour des textes ASCII, on utilise souvent l'algorithme de Lempel-Ziv. Cet algorithme sera vu dans le cours de théorie de l'information en 4ème année. Les commandes Unix `compress` et `zip` (`winzip` sur Windows) utilisent cet algorithme. Vous pouvez facilement observer en pratique la performance de ce code, on trouve fréquemment un rapport de compression de l'ordre de 1 : 8. Or l'entropie par symbole d'un texte français est, en gros, de l'ordre de 1 bit par caractère ; un caractère encodé suivant le code ASCII coûte 8 bits. Le théorème de codage de source dit donc qu'un code par bloc efficace permet d'atteindre environ 1 bit par caractère, ce qui, pour un texte ASCII, donne un rapport de compression égal à 1 : 8. Cela montre que l'algorithme de Lempel-Ziv est quasi-optimal.

Les fichiers audio utilisée en téléphonie sont des suites de symboles de 8 bits ; l'entropie par symbole en dehors des périodes de silence est de l'ordre de 4 bits. Il est donc possible de comprimer les périodes d'activité avec un rapport de compression proche de 1 : 2, en utilisant un code de Huffman [4]. Comme les silences occupent environ la moitié du temps (pour une conversation équilibrée), il est possible de comprimer avec un rapport d'environ 1 : 4, sans perte. Les fichiers audio HiFi sont des suites de symboles de 16 bits, là aussi il est possible d'atteindre un rapport de compression (sans perte) proche de 1 : 2.

4. H. Gharavi and R. Steele. Conditional entropy encoding of LOG-PCM speech. *Electronics Letters*, 21(11): 475–476, 2007. ISSN 0013-5194

Exemple 5.9 (Codage par Longueur de Plage ("Run Length Encoding"))
Un algorithme de codage assez simple pouvant s'appliquer à des images graphiques à deux niveaux (noir (N), blanc (B)), dont l'exemple typique est le fax, est le codage par longueur de plage ("Run Length Encoding"). L'image est découpée en carrés élémentaires, les pixels. Prenons une ligne d'une telle image, qui est donc une succession de pixels blancs et noirs. Par exemple, prenons la colonne de gauche dans la Figure 5.2.

Au lieu d'encoder chaque pixel blanc par un bit 0 *et chaque pixel noir par un bit* 1*, ce qui ne procure aucune compression, on compte le nombre de pixels blancs et noirs successifs (qu'on appelle plages), et on code cette longueur par un entier, en notation décimale. On obtient la colonne du milieu dans la figure.*

Ensuite on code ces symboles entiers par un code binaire approprié, soit à longueur constante, soit plus efficacement par un code de Huffman

B		
B		
B		
B		
N		
N		
N		
B		
B		
B		
N		
N		
B		
B		
B		
B		
N		
N		
N		
B		1
B		0
B		1
B		1
B		1
B		0
B		1
B		0
B		0
B		0
B		1
B		1
B		1
B		0
B		1
B		1
B		1
B		0
B		0
B		1
B		0
B		1
B		0
B		0
N		0
N	*B*4	0
N	*N*3	1
N	*B*3	1
B	*N*2	1
B	*B*4	1
B	*N*3	0
B	*B*24	0
B	*N*4	0
B	*B*6	1
N	*N*1	0

FIGURE 5.2: Une suite de pixels noirs et blancs, son encodage par longueur de plage, et l'encodage final, obtenu en encodant les longueurs de plages par le code de Huffman dans la Table 5.4.

(les symboles de la source sont les entiers qui représentent les longueurs des plages de pixels blancs et noirs). Une des normes recommandées par le CCITT (Comité Consultatif International Téléphonique et Télégraphique, devenu ITU, International Telecommuncation Union) propose un algorithme de Huffman modifié pour le codage des plages. Par exemple, une plage de 100 pixels blancs est encodée par la suite 1101100010101, *soit* 13 *bits de code pour 100 bits de sources. Les probabilités d'apparition des différentes longueurs de plages sont estimées à partir d'images tests. Quelques valeurs du code de Huffman sont données dans la Figure 5.2, ce qui permet de déduire l'encodage final de l'exemple.*

longueur de plage	*blanche*	*noire*
1	000111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010
...	...	...
24	0101000	...

TABLE 5.4: Code de Huffman utilisé pour encoder les longueurs de plages blanches et noires.

Au-Delà de la Compression Sans Perte Tous les algorithmes de compression décrits ci-dessus sont sans pertes (lossless compression), ce qui veut dire qu'ils sont uniquement décodables. Pour des sources audio ou vidéo, il est souvent possible d'augmenter fortement le taux de compression en acceptant une modification des données (compression avec perte, lossy compression). Ainsi, les standards de compression audio/video JPEG, MPEG, MP3, etc se font avec pertes. Ils seront étudiés dans les cours de traitement des signaux, audio et images, lors des 3ème et 4ème années.

Q. 38. Pourrait-on supprimer les "B" et "N" dans le codage par longueur de plage illustré dans la Figure 5.2 (colonne du milieu) ?

II
Cryptographie

6
La Cryptographie

Le besoin de protection de l'information est aussi ancien que la civilisation elle-même. D'abord exclusivement utilisée par les militaires et les diplomates, elle a acquis avec l'apparition des réseaux informatiques quantité d'applications commerciales. On distingue habituellement les propriétés suivantes :

- l'*intégrité* : le message reçu est identique à celui qui a été envoyé ;
- la *confidentialité* : seul le destinataire autorisé est capable de lire le message ;
- la *authentification* : le destinataire peut être certain que le message a vraiment été écrit par la personne qui prétend en être l'auteur.

6.1 Eléments d'un Système Cryptographique

Lors de l'opération de chiffrement, le *texte clair* (*"plaintext"*) P est transformé par une fonction E paramétrée par une *clé* K, pour ainsi obtenir un *texte chiffré*, appelé aussi *cryptogramme* ("*ciphertext*") $C = E_K(P)$. Ce cryptogramme est alors transmis au récepteur, qui applique un algorithme de déchiffrement D_k muni d'une clé k, qui recouvre le texte clair original : $P = D_k(C) = D_k(E_K(P))$. La clé de déchiffrement k peut être identique à la clé de chiffrement K (auquel cas on parle de cryptographie *symétrique*), ou pas (on parle alors de cryptographie *asymétrique*). Si le système est symétrique il faut que la clé soit maintenue secrète. Nous étudierons au Chapitre 10 un algorithme asymétrique, qui n'utilise pas la même clé au chiffrement et au déchiffrement.

On suppose que "l'intrus" écoute et peut reproduire fidèlement le cryptogramme complet. Il ne connaît cependant pas les clés, et ne peut retrouver aisément le message en clair bien que l'on suppose qu'il connaisse l'algorithme utilisé. Cette hypothèse est connue sous le nom de *thèse de Kerckhoffs*. Parfois l'intrus ne se contente pas d'écouter le canal de communication (intrus passif), mais peut altérer les messages ou injecter ses propres messages dans le canal de communication (intrus actif). L'art de composer des cryptogrammes est la *cryptographie*, l'art de les briser est la *cryptanalyse*.

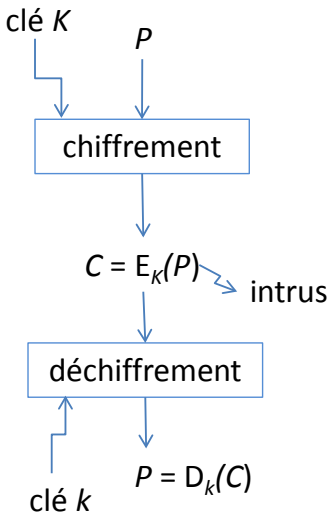


FIGURE 6.1: La cryptographie : un texte clair P est chiffré à l'aide d'une clé K, transmis, puis déchiffré à l'aide d'une clé k. Un intrus peut voir le texte transmis C, mais, sans la connaissance de la clé k, il ne peut le déchiffrer.

Pourquoi faire l'hypothèse de Kerckhoffs ? L'histoire a montré qu'un secret ne le reste pas longtemps une fois qu'il est partagé par plusieurs personnes, nous ne pouvons donc pas supposer qu'un algorithme de chiffrement restera inconnu. Alors que changer d'algorithme chaque fois que le système est compromis serait long et couteux, il n'y a pas de problème à changer fréquemment la clé. Le modèle de base de chiffrement comporte donc une méthode générale de chiffrement constante et connue, paramétrée par une clé secrète et facilement modifiable.

Le problème du cryptanalyste peut être divisé en plusieurs catégories, en fonction de l'information dont il dispose. S'il ne dispose que d'une certaine quantité de cryptogrammes, mais pas du texte clair correspondant, il essaie de compromettre le système cryptographique par une attaque *à texte chiffré seul* ("ciphertext-only attack"). Clairement, un chiffrement doit être protégé contre une telle attaque, puisque nous faisons l'hypothèse qu'il a accès au canal de communication. Si le cryptanalyste peut analyser des paires composées de texte clair et du cryptogramme correspondant, l'attaque devient *à texte clair connu* ("known plaintext attack"). Une telle attaque est possible lorsque l'intrus, à un moment donné, a eu accès à une base de données contenant de telles paires. Comme nous ne pouvons être sûrs que cela ne se produira jamais, notre système doit aussi pouvoir résister à ce genre d'attaque. Enfin, si les circonstances sont tellement favorables (du point de vue du cryptanalyste) que celui-ci peut obtenir le cryptogramme correspondant au texte clair de son choix (si, par exemple, il parvient à accéder au mécanisme d'encodage, mais ne peut pas voir la clé) on parle alors d'attaque *à texte clair choisi* ("chosen plaintext attack").

Exemple 6.1 (Chiffre de *César*) *Jules César utilisait un procédé de chiffrement symétrique qui consistait en une rotation de toutes les lettres de l'alphabet de trois positions. Ainsi, a devenait d, b devenait e, . . .et z devenait c. On peut généraliser le chiffrement de César pour permettre une rotation de K lettres, au lieu de prendre toujours 3 lettres. Dans ce cas la clé K est un nombre entre* 0 *et* 25*, interprété comme une rotation de l'alphabet.*

Evidemment, le chiffrement de rotation n'est pas très sécurisé. Puisque, par hypothèse (de Kerckhoffs), le cryptanalyste sait que nous utilisons un chiffrement de rotation, il n'y a que 26 clés possibles et il est trivial de toutes les essayer.

Q. 39. Dans le film *Odyssée de l'Espace 2001*, le nom de l'ordinateur est HAL, et c'est en fait un texte chiffré, utilisant un chiffre de César. Quel est le texte clair correspondant ? Et quelle est la clé K ?

Exemple 6.2 (Chiffrement par Substitution) *Un chiffrement symétrique plus sécurisée que le chiffre de César consiste à remplacer chaque lettre de l'alphabet du texte clair par une autre lettre, sans respecter une relation de rotation entre les deux alphabets. Ce système général est appelé substitution monoalphabétique, la clé étant le tableau de correspondance entre les alphabets du texte en clair et du texte crypté. Par exemple, on pourrait prendre la clé de la Figure 6.2.*

Pour un alphabet de D caractères, il y a D! clés possibles. Puisque $26! \sim 10^{26}$*, un très grand nombre, ce qui semble indiquer que cet algorithme soit relativement sécurisé. Malheureusement, ce n'est pas tout à fait le cas.*

En effet, le cryptanalyste peut utiliser la distribution des fréquences des lettres dans le cryptogramme. Par exemple, en Anglais, les lettres les plus fréquentes sont e et t, que le cryptanalyste peut alors essayer d'assigner aux deux lettres les plus fréquentes dans le cryptogramme. Il trouvera vraisemblalement alors beaucoup de triplets de la forme tXe, suggérant fortement que X corresponde à h. En procédant de la sorte, il peut retrouver, lettre par lettre, la clé de chiffrement assez rapidement (du

texte clair	texte chiffré	texte clair	texte chiffré
a	*Q*	*n*	*F*
b	*W*	*o*	*G*
c	*E*	*p*	*H*
d	*R*	*q*	*J*
e	*T*	*r*	*K*
f	*Z*	*s*	*L*
g	*U*	*t*	*Y*
h	*I*	*u*	*X*
i	*O*	*v*	*C*
j	*P*	*w*	*V*
k	*A*	*x*	*B*
l	*S*	*y*	*N*
m	*D*	*z*	*M*

FIGURE 6.2: Exemple de clé du chiffrement par substitution monoalphabétique.

moins avec un ordinateur).

De plus, il peut parfois deviner certains mots selon le contexte. Par exemple, s'il s'agit d'une transaction boursière, le message risque de comporter les mots "action" ou "cours"...

Pour compliquer la tâche du cryptanalyste, il faut donc "cacher" la distribution de fréquences des lettres, de telle sorte que des lettres comme e, a, t ne soient pas si faciles à repérer. Une manière de procéder est d'introduire plusieurs alphabets cycliques, pour obtenir un chiffrement de Vigenère, qui est un exemple de chiffrement utilisant la substitution polyalphabétique. La clé est d'habitude un mot court, facile à mémoriser, comme BONJOUR (au lieu du tableau complet des 26 lettres dans le cas monoalphabétique). La clé est répétée constamment sous le texte en clair, et indique quelle rangée de la table précédente doit être utilisée pour le chiffrement – voir Figure 6.3.

Un exemple célèbre de méthode de chiffrement par substitution polyalphabétique est la machine ENIGMA utilisée par les forces de l'axe (Allemagne, Japon et leurs alliés) pendant la seconde guerre mondiale. La substitution s'opérait grâce à trois rotors ayant chacun 26 positions. Les positions initiales des rotors étaient encodées sur l'en-tête du message, ce qui était une faiblesse exploitée par les cryptanalystes britanniques, français et polonais (`http://www.bletchleypark.org.uk/`).

alphabet d'origine	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
rangée *A*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
rangée *B*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*
rangée *C*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*
rangée *D*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*
rangée *E*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*
rangée *F*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*
rangée *G*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*
rangée *H*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
rangée *I*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
rangée *J*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
rangée *K*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
rangée *L*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*
rangée *M*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
rangée *N*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
rangée *O*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*
rangée *P*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
rangée *Q*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*
rangée *R*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*
rangée *S*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*
rangée *T*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*
rangée *U*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*
rangée *V*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
rangée *W*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*
rangée *X*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*
rangée *Y*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*
rangée *Z*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*

texte clair :	l	a	c	r	y	p	t	o	g	r	a	p	h	i	e
clé :		B	O	N	J	O	U	R	B	O	N	J	O	U	R	B
texte crypté :	M	O	P	A	M	J	K	P	U	E	J	D	B	Z	F

FIGURE 6.3: Chiffrement de Vigenère. La lettre *l* est chiffrée avec l'alphabet de la rangée *B* et devient *M*, la lettre *a* est chiffrée avec l'alphabet de la rangée *O* et devient *O*, etc.

Bien qu'il soit plus sûr qu'un chiffrement de substitution monoalphabétique, un chiffrement de substitution polyalphabétique peut encore être brisé par une attaque à texte crypté seul. L'astuce consiste à deviner la longueur de la clé.

6.2 ⋆ Confidentialité Parfaite

Il est possible en théorie de fabriquer un système qui assure une confidentialité parfaite (mais nous allons voir que cela pose quelques problèmes). Considérons un crypto-système symétrique, dont la clé $K = k$ est secrète et inconnue de l'intrus. L'intrus peut observer le texte chiffré C, mais ne connaît pas le texte clair P. Interprétons les messages P et C comme délivrés par des sources $\mathcal{P}$ et $\mathcal{C}$. Pour que le système soit parfait, il faut que l'information qu'il obtient en observant C ou en ne l'observant pas soit la même, ce qui correspond à dire que les sources sont indépendantes :

Définition 6.1 *On dit que le cryptosystème est à confidentialité parfaite si les sources qui délivrent les messages P et C sont indépendantes.*

Exemple 6.3 (Vernam) *Le masque à usage unique (Ang. one time pad) ou cryptosystème de Vernam est un système utilisé dans les ambassades. Il fonctionne de la façon suivante.*

Le texte clair P est encodé comme une suite de n bits (chiffres binaires). La longueur n est constante et connue de tous. On s'arrange pour que tous les textes aient pour longueur n (les messages plus long que n bits sont fragmentés, les messages plus courts que n bits sont allongés en ajoutant des caractères "espace"). La clé K est aussi une suite de n bits, tirée au sort, indépendamment de P, et utilisée une seule fois. Nous supposons que tous les choix de clés ont la même probabilité. Le texte chiffré est $C = P \oplus K$, où $\oplus$ est l'opération `xor` bit par bit. Le déchiffrement se fait par

$$P = C \oplus K$$

ce qui suppose que le récepteur connaît par avance la clé secrète.

Soit $\mathcal{P}$ la source qui émet le texte clair P, $\mathcal{K}$ celle qui émet la clé secrète K et $\mathcal{C}$ celle qui émet le texte chiffré C (un intrus ne peut observer que $\mathcal{C}$). La densité de probabilité de la source $(\mathcal{P}, \mathcal{K})$ a une forme produit car le message en clair P et la clé K sont indépendants : $p_{\mathcal{P},\mathcal{K}}(P, K) = p_{\mathcal{P}}(P)p_{\mathcal{K}}(K)$.

Notons que $K = C \oplus P$ donc la probabilité de l'événement E : "les sources $\mathcal{P}$ et $\mathcal{C}$ émettent P et C" est $p_{\mathcal{P},\mathcal{C}}(P, C) = p_{\mathcal{P},\mathcal{K}}(P, C \oplus P)$, puisque l'événement E est aussi égal à "$\mathcal{P}$ émet P et $\mathcal{K}$ émet $C \oplus P$". Donc

$$p_{\mathcal{P},\mathcal{C}}(P, C) = p_{\mathcal{P}}(P)p_{\mathcal{K}}(C \oplus P) = \frac{1}{2^n} p_{\mathcal{P}}(P) \tag{6.1}$$

où la dernière égalité est parce que toutes les clés ont même probabilité.

Nous pouvons en déduire la densité de probabilité conditionnelle du texte chiffré sachant que le texte clair est P :

$$p_{\mathcal{C}|\mathcal{P}}(C|P) \stackrel{\text{def}}{=} \frac{p_{\mathcal{P},\mathcal{C}}(P, C)}{p_{\mathcal{P}}(P)} = \frac{1}{2^n}$$

L'opération `xor` sur 1 bit (aussi appelé "ou exclusif")est définie par la table

xor	0	1
0	0	1
1	1	0

L'opération `xor` bit par bit, que nous notons $\oplus$, porte sur des suites de n bits, et est obtenue en appliquant la table bit par bit :

$$a_1a_2...a_n \oplus b_1b_2...b_n = c_1c_2...c_n$$

avec $c_i = a_i$ `xor` b_i pour $i = 1...n$. Par exemple :

$$\begin{array}{rr} & 0100 \\ \oplus & 1101 \\ \hline = & 1001 \end{array}$$

Notons que $a = b \oplus c$ est équivalent à $b = a \oplus c$.

Supposons que chaque lettre est codée sur 5 bits par sa position dans l'alphabet (A est codée par 1, B est codée par 2, etc. et l'espace par 0). Utilisons le cryptosytème de Vernam avec des suites de 8 lettres, ce qui donne un masque K de longueur $n = 40$ bits. Nous voulons transmettre le mot BONJOUR ; le texte clair P est obtenu en ajoutant un espace et vaut, en base 10 :

2 15 14 10 15 21 18 0

Converti en binaire, cela donne le texte clair :

$P =$
00010 01111 01110 01010
01111 10101 10010 00000

Le masque à usage unique est aléatoire ; supposons qu'on ait :

$K =$
01100 01000 11001 10111
00101 11010 01011 10001

Le texte chiffré est obtenu par l'opération `xor` : $C = P \oplus K$

$C =$
01110 00111 01111 11101
01010 01111 11001 10001

Le déchiffrement s'obtient de la même façon en calculant $C \oplus K$.

Donc cette densité conditionnelle est indépendante de P ; par le Théorème 0.1, les sources $\mathcal{P}$ et $\mathcal{C}$ sont indépendantes. Donc le cryptosystème de Vernam est à confidentialité parfaite. Dans un certain sens, il offre la meilleure confidentialité possible (à condition de pouvoir se mettre d'accord sur les clés par avance, et les conserver en lieu sûr).

Notons que l'hypothèse que les clés ont toute même probabilité est essentielle. En effet, si elle n'est pas vraie, le terme $p_{\mathcal{K}}(C \oplus P)$ dans l'Eq.(6.1) dépend à la fois de P et C et nous ne pouvons pas factoriser. Par contre, il n'y a aucune hypothèse sur la densité de probabilité du texte clair (qui, en général, n'est pas uniforme).

Dans l'exemple précédent, nous voyons que la clé est aussi longue que le texte clair. C'est un fait général, comme l'expriment les deux théorèmes suivants. Le premier s'applique à tout système cryptographique symétrique, qu'il soit à confidentialité parfaite ou pas. Il dit que l'entropie du texte chiffré est au moins aussi grande que celle du texte clair.

Théorème 6.1 (Cryptographie Symétrique) *Considérons un système de cryptographie symétrique (donc à clé secrète) ; supposons de plus (ce qui est raisonnable) que la clé soit choisie indépendamment du texte clair. Alors*

$$H(\mathcal{P}) \leq H(\mathcal{C}) \tag{6.2}$$

où $H(\mathcal{P})$ est l'entropie du texte clair, et $H(\mathcal{C})$ l'entropie du texte chiffré.

☺*Preuve :* Le texte chiffré dépend de manière déterministe de la combinaison du texte clair et de la clé, donc d'après le théorème de traitement de l'information (Eq.(4.2) en Section 4.3) $H(\mathcal{C}, \mathcal{P}, \mathcal{K}) = H(\mathcal{P}, \mathcal{K})$. De même, puisque le déchiffrage fonctionne, le texte clair dépend de manière déterministe de la combinaison du texte chiffré et de la clé, donc $H(\mathcal{C}, \mathcal{P}, \mathcal{K}) = H(\mathcal{C}, \mathcal{K})$. Donc, en combinant les deux :

$$H(\mathcal{P}, \mathcal{K}) = H(\mathcal{C}, \mathcal{K}) \tag{6.3}$$

Puisque $\mathcal{P}$ et $\mathcal{K}$ sont indépendantes, le membre de gauche de Eq.(6.3) est égal à $H(\mathcal{P}) + H(\mathcal{K})$ (Théorème 1.4). Par la règle d'enchaînement (Théorème 4.4), le membre de droite est $H(\mathcal{K}) + H(\mathcal{C}|\mathcal{K})$. Donc

$$H(\mathcal{P}) = H(\mathcal{C}|\mathcal{K})$$

Or conditionner réduit l'entropie (Théorème 4.2), donc $H(\mathcal{C}|\mathcal{K}) \leq H(\mathcal{C})$. ☺□

Le deuxième théorème s'applique aux systèmes à confidentialité parfaite et exprime que l'entropie de la clé secrète doit être au moins aussi grande que celle du texte clair (et aussi du texte chiffré) :

Théorème 6.2 (Confidentialité Parfaite) *Considérons un système de cryptographie symétrique à confidentialité parfaite. Supposons que la clé soit choisie indépendamment du texte clair. Alors*

$$H(\mathcal{P}) \leq H(\mathcal{C}) \leq H(\mathcal{K}) \tag{6.4}$$

Q. 40. Calculer l'entropie de la clé et du texte chiffré pour le cryptosystème de Vernam (Exemple 6.3) et vérifier les inégalités du Théorème 6.1 et du Théorème 6.2.

où $H(\mathcal{P})$ est l'entropie du texte clair, $H(\mathcal{K})$ l'entropie de la clé et $H(\mathcal{C})$ l'entropie du texte chiffré.

Preuve : La première inégalité du théorème est l'Eq.(6.2), donc il nous reste seulement à démontrer la seconde.

Montrons d'abord l'inégalité (6.5) ci-dessous, qui est vraie pour les systèmes cryptographiques symétriques, qu'ils soient à confidentialité parfaite ou pas. Le texte chiffré dépend de manière déterministe de la combinaison du texte clair et de la clé, donc (Théorème 4.5 en Section 4.3) $H(\mathcal{C}|\mathcal{P},\mathcal{K}) = 0$. Par la règle d'enchaînement (Théorème 4.4), $H(\mathcal{C},\mathcal{K}|\mathcal{P}) = H(\mathcal{K}|\mathcal{P}) + H(\mathcal{C}|\mathcal{P},\mathcal{K}) = H(\mathcal{K}|\mathcal{P})$. D'autre part, toujours par la règle d'enchaînement, $H(\mathcal{C}|\mathcal{P}) = H(\mathcal{C},\mathcal{K}|\mathcal{P}) - H(\mathcal{K}|\mathcal{C},\mathcal{P}) \leq H(\mathcal{C},\mathcal{K}|\mathcal{P})$. En combinant les deux, nous obtenons :

$$H(\mathcal{C}|\mathcal{P}) \leq H(\mathcal{K}|\mathcal{P}) \tag{6.5}$$

En termes simples, cette inégalité exprime que l'information supplémentaire ajoutée par la clé quand on connaît le texte clair est au moins aussi grande que celle ajoutée par le texte encrypté – ce qui est normal puisque si on connaît la clé et le texte clair, on peut trouver le texte encrypté.

Supposons maintenant que le système soit à confidentialité parfaite, c'est à dire que $\mathcal{P}$ et $\mathcal{C}$ sont indépendantes. Alors (Théorème 4.3) $H(\mathcal{C}|\mathcal{P}) = H(\mathcal{C})$, donc en utilisant l'Eq.(6.5) : $H(\mathcal{C}) \leq H(\mathcal{K}|\mathcal{P}) \leq H(\mathcal{K})$, où la dernière inégalité est parce que conditionner réduit l'entropie. □

7
Arithmétique

La cryptographie moderne ne se base pas sur des substitutions alphabétiques. Au lieu de cela, on considère un texte clair comme une suite de bits, et, en regroupant les bits par blocs, comme une suite de nombres entiers. On applique ensuite des opérations sur les nombres entiers. Pour aller plus avant, il nous faut donc étudier la théorie des nombres entiers, appelée *arithmétique*.

Q. 41. Quels sont le reste et le quotient dans la division de 23 par 5 ? de -23 par 5 ?
Q. 43. Combien vaut 24163584354 mod 10 ?

7.1 Les Entiers

On désigne par $\mathbb{Z}$ l'ensemble des entiers (positifs et négatifs). Si on ajoute, soustrait ou multiplie des entiers, on obtient des nombres entiers. Par contre, ce n'est pas vrai, en général, pour la division (par exemple $\frac{1}{2}$ n'est pas un nombre entier). On peut cependant définir une division qui produit des nombres entiers, à condition de conserver un reste.

Q. 42. Combien valent 13 mod 10, (-13) mod 10, 13 mod (-10), (-13) mod (-10) et $13 mod 0$?

Théorème 7.1 (Division Euclidienne) *Soient a et b des entiers avec $b \neq 0$. Il existe un couple unique d'entiers (q, r) tel que*

$$a = bq + r \quad et \quad 0 \leq r \leq |b| - 1.$$

⋆**Q. 44.** Prouvez le Théorème 7.1.

Définition 7.1 *L'entier q est appelé le quotient et r le reste de a dans la division par b. Le reste de a dans la division par b se note habituellement a mod b.*

Dans le langage Python, $a\%b$ est le reste de a dans la division par b pour $b > 0$, ainsi 23%5 retourne 3 et $(-23)\%5$ retourne 2.
Dans les langages C++ et Java, malheureusement, c'est un peu plus compliqué ; si $a \geq 0$ et $b > 0$, $a\%b$ vaut le reste de la division de a par b, mais si $a < 0$ le résultat vaut (reste $-$ b). Ainsi, dans ces langages, 23%5 retourne 3 mais $(-23)\%5$ retourne -3.

Définition 7.2 *Si $a, b \in \mathbb{Z}$ et $b \neq 0$, on dit que b divise a, ou que b est un diviseur de a, ou encore que a est un multiple de b, si $\frac{a}{b}$ est un entier. C'est équivalent à dire que le reste de a dans la division par b est 0.*

Par exemple 6 divise 12 et ne divise pas 13 ; 12 est un multiple de 6 et 13 n'est pas un multiple de 6.

Définition 7.3 *On dit que $a \in \mathbb{Z}, a > 1$ est un nombre premier s'il n'a pas d'autre diviseur positif que a et 1.*

Q. 45. Les nombres suivants sont-ils premiers : $27, 255, 256$?

Les nombres premiers jouent un rôle important en cryptographie. La suite des nombres premiers commence par $2, 3, 5, 7, 11, 13, 17, 19, 23....$

Un des théorèmes de base de l'arithmétique, que nous ne démontrons pas, exprime que tout nombre entier décompose de manière unique en produit de nombres premiers :

Théorème 7.2 (Factorisation) *Pour tout entier a strictement positif il existe une suite unique de nombres premiers $p_1 < p_2 ... < p_k$ et une suite unique d'exposants $\alpha_1 > 0, ..., \alpha_k > 0$ tels que*

$$a = p_1^{\alpha_1} ... p_k^{\alpha_k}$$

Les nombres $p_1, ... p_k$ sont appelés les facteurs premiers de a.

Q. 46. Quelles sont les factorisations de 12, de 100 et de 256 ?

Théorème 7.3 *Soient a et b deux entiers positifs ; a divise b si et seulement si tous les facteurs premiers de a apparaissent dans la décomposition en facteurs premiers de b, avec un exposant supérieur ou égal.*

Par exemple, la décomposition en facteurs premiers de de $a = 12$ est $12 = 2^2.3$, et celle de $b = 168$ est $168 = 2^3.3.7$. Les facteurs premiers de a sont 2 et 3 avec exposants 2 et 1. Ils apparaissent tous les deux dans la décomposition de b avec un exposant égal à 3 et 1 donc a divise b. Par contre a ne divise pas $c = 30$ car $c = 2.3.5$ et le facteur 2 est présent dans a avec un exposant trop grand.

Si p et q sont deux nombres donnés, il est très facile de calculer $a = pq$, même si p et q sont grands. Par contre, si un nombre a est donné, dont on soupçonne qu'il se factorise sous la forme $a = pq$ avec p et q premiers, il est (jusqu'à aujourd'hui) très difficile de calculer p et q, si a est grand (par exemple si a est un nombre de 1000 bits). De manière générale, on pense que la factorisation est difficile à calculer pour un nombre a très grand ; aucune méthode connue existe qui soit de complexité raisonnable. Un des postulats de la cryptographie moderne est que les intrus ne possèdent pas de méthode pour factoriser rapidement de très grands nombres.

On dit que la fonction qui à deux nombres premiers p, q associe leur produit pq est une fonction à *sens unique*, c'est à dire facile à calculer mais difficile à inverser. De la même façon, un bottin téléphonique est une fonction à sens unique : il est facile de trouver le numéro de quelqu'un en connaissant son nom, mais beaucoup moins évident de trouver le nom de la personne au numéro de téléphone donné. La cryptographie utilise beaucoup les fonctions à sens unique.

Q. 47. Montrez que si un nombre a n'est pas premier alors son plus petit facteur premier est $\leq \sqrt{a}$.

Q. 48. Le nombre 257 est-il premier ?

Soient a et b deux entiers non tous deux nuls. Les entiers positifs qui divisent à la fois a et b forment un ensemble fini (ces nombres sont $\leq a$ et $\leq b$), non vide puisque 1 divise tous les entiers. Par conséquent, cet ensemble possède un plus grand élément :

Définition 7.4 (PGCD) *Soient a et b deux entiers non tous deux nuls. On appelle plus grand commun diviseur (PGCD) de a et b, (noté $pgcd(a, b)$), le plus grand nombre entier positif qui divise à la fois a et b.*

Notons que 0 est divisible par tous les nombres donc les diviseurs communs à 0 et $b \neq 0$ sont les diviseurs de b, dont le plus grand est $|b|$; donc $\text{pgcd}(0, b) = |b|$ si $b \neq 0$.

Si $a = b = 0$ l'ensemble des diviseurs de a et b est infini donc $\text{pgcd}(0, 0)$ n'est en principe pas défini ; on fait cependant la convention que $\text{pgcd}(0, 0) = 0$; la raison en est l'identité de Bézout (Théorème 8.4).

Pour des nombres positifs pas trop grands, le PGCD peut être calculé simplement à partir des décompositions en facteurs premiers, en utilisant le théorème suivant (que nous ne démontrons pas) :

Théorème 7.4 *Soient a et b deux entiers positifs et soient $p_1 < p_2 < ... < p_n$ la suite des nombres premiers qui divisent a ou b. On peut donc écrire*

$$\begin{aligned} a &= p_1^{\alpha_1} ... p_k^{\alpha_k} \\ b &= p_1^{\beta_1} ... p_k^{\beta_k} \end{aligned}$$

avec $\alpha_i \geq 0$ *et* $\beta_i \geq 0$. *Alors*

$$pgcd(a,b) = p_1^{\gamma_1} ... p_k^{\gamma_k}$$

avec $\gamma_i = \min(\alpha_i, \beta_i)$

En d'autres termes, nous obtenons le PGCD en prenant les facteurs premiers en commun, avec le plus petit exposant. S'il n'y a aucun facteur en commun, le PGCD vaut 1.

Nous utiliserons souvent la propriété suivante :

Définition 7.5 *On dit que deux entiers a et b sont premiers entre eux ou étrangers, ou encore que a est premier avec b si* $pgcd(a,b) = 1$.

Pour des nombres pas trop grand, un critère facile (que nous ne démontrons pas) est le suivant.

Théorème 7.5 *Soient a et b deux entiers positifs. Ils sont premiers entre eux si et seulement si ils n'ont aucun facteur premier commun.*

Trois conséquences immédiates, et importantes, sont les suivantes.

Théorème 7.6
1. *Deux nombres premiers distincts sont premiers entre eux.*
2. *Soit p un nombre premier et a un entier tel que* $1 \leq a \leq p-1$. *Alors a et p sont premiers entre eux.*
3. *Soient a et b deux nombres premiers entre eux, et c un entier. Si a et b divisent c alors ab divise c.*

Notons que contrairement à la factorisation, savoir si deux nombres sont premiers entre eux est un problème facile, que l'on peut résoudre avec l'algorithme d'Euclide, que nous verrons plus loin.

7.2 Congruences

Définition 7.6 (Congruence modulo m) *Soient a et b deux entiers et m un entier non nul. On dit que a est congru à b modulo m, et on écrit*

$$a \equiv b \pmod{m} \tag{7.1}$$

si a et b ont le même reste dans la division par m. L'expression (7.1) s'appelle congruence et m est son module.

Théorème 7.7 *Soient a et b deux entiers et m un entier non nul.* $a \equiv b \pmod{m}$ *si et seulement si m divise* $b - a$.

☺*Preuve :* 1. $a \equiv b \pmod{m} \Rightarrow m$ divise $b-a$. Faisons la division euclidienne de a et b par m ; les restes sont les mêmes, donc nous pouvons écrire $a = mq + r$ et $b = mq' + r$. Donc $b - a = m(q' - q)$ est un multiple de m.

2. m divise $b - a \Rightarrow a \equiv b \pmod{m}$. Faisons la division euclidienne de a par m : nous pouvons écrire $a = mq + r$ avec

Q. 49. Quel est le PGCD de 12 et 100 ?

Q. 50. Les nombres 12 et 20 sont ils premiers entre eux ? 12 et 35 ? 234 et 257 ?

Q. 51. Prouvez le Théorème 7.6.

Q. 52. Combien y a-t-il de nombres positifs < 257 qui soient premiers avec 257 ?

Q. 53. Est-il vrai qu'un nombre est divisible par 12 si et seulement si il est divisible par 3 et par 4 ?

Q. 54. Est-il vrai qu'un nombre est divisible par 12 si et seulement si il est divisible par 2 et par 6 ?

Q. 55. Les congruences suivantes sont-elles vraies ?

$-2394860 \equiv 32474364 \pmod{2}$
$-1 \equiv 1 \pmod{2}$
$2394860 \equiv 0 \pmod{2}$
$23 \equiv 3 \pmod{5}$
$-23 \equiv 3 \pmod{5}$
$-23 \equiv -3 \pmod{5}$
$-23 \equiv 2 \pmod{5}$

$0 \leq r \leq |m| - 1$ De plus $b - a$ est un multiple de m, donc nous pouvons écrire $b - a = \lambda m$ avec λ entier. Donc

$$b = (b - a) + a = \lambda m + qm + r = (\lambda + q)m + r$$

comme $0 \leq r \leq |m| - 1$, par le Théorème 7.1, cette équation donne le quotient $(\lambda + m)$ et le reste (r) de la division euclidienne de b par m. En particulier, le reste de la division de b par m est r, le même que a. ☺□

Les congruences sont utilisées abondamment en cryptographie et codage correcteur. Leur intérêt principal est qu'on peut calculer avec elles, comme avec des égalités. Pour commencer, notons que la congruence modulo m possède les propriétés suivantes, qui en rendent la manipulation aisée, et dont la preuve est une conséquence immédiate du Théorème 7.7 :

Théorème 7.8 *Soient a, b, c des entiers et m un entier non nul.*
- *réflexivité : $a \equiv a \pmod m$,*
- *symétrie : si $a \equiv b \pmod m$ alors $b \equiv a \pmod m$,*
- *transitivité : si $a \equiv b \pmod m$ et $b \equiv c \pmod m$ alors $a \equiv c \pmod m$.*

L'ensemble de ces trois propriétés font que l'on dit que la congruence est une "relation d'équivalence".

Elles ont pour conséquence que l'on peut enchaîner des congruences et oublier l'ordre dans lequel on les écrit, comme par exemple dans $-2 \equiv 0 \equiv 2 \equiv 4 \pmod 2$. D'autre part, la congruence modulo m se combine naturellement avec les opérations de base, sauf la division :

Théorème 7.9 (Arithmétique Modulaire) *Soient a, a', b, b', m et n des entiers. Si*

$$\begin{aligned} a &\equiv a' \pmod m \\ b &\equiv b' \pmod m \end{aligned}$$

alors

$$\begin{aligned} a + b &\equiv a' + b' \pmod m \\ ab &\equiv a'b' \pmod m \\ a^n &\equiv a'^n \pmod m \end{aligned}$$

Ainsi $2 \equiv (-1) \pmod 3$ donc $2^n \equiv (-1)^n \pmod 3$, par exemple $2^{1000} \equiv 1 \pmod 3$ et donc $2^{1000} + 2 \equiv 2 + 1 \equiv 0 \pmod 3$ donc $2^{1000} + 2$ est divisible par 3.

☺*Preuve :* Nous prouvons la première égalité, les autres sont laissées au soin du lecteur alerte. Supposons que $a \equiv a' \pmod m$ et $b \equiv b' \pmod m$. Par le Théorème 7.7, il existe x, x' entiers tels que $a - a' = xm$ et $b - b' = x'm$. Donc $(a + b) - (a' + b') = (x - x')m$ et donc $a + b \equiv a' + b' \pmod m$. ☺□

Q. 56. Vrai ou faux :
1. $a \equiv 0 \pmod m \Leftrightarrow a$ divise m
2. $a \equiv 0 \pmod m \Leftrightarrow m$ divise a
3. $a \equiv 0 \pmod m \Leftrightarrow m$ et a sont premiers entre eux.

Intuitivement, une *relation* $\mathcal{R}$ définie sur un ensemble A fait correspondre à tout élément a de A 0, 1 ou plusieurs éléments de A. On écrit $a\mathcal{R}b$ si l'élément b correspond à a. Formellement, $\mathcal{R}$ est un sous-ensemble du produit cartésien $A \times A$.

Si pour chaque a il existe 0 ou 1 élément b tel que $a\mathcal{R}b$ alors $\mathcal{R}$ est une fonction. S'il existe pour chaque a exactement 1 élément b, alors $\mathcal{R}$ est une application.

On dit que la relation est
- *réflexive* si $\forall a \in A, a\mathcal{R}a$
- *symétrique* si $\forall a, b \in A, a\mathcal{R}b \Rightarrow b\mathcal{R}a$
- *transitive* si $\forall a, b, c \in A$, $(a\mathcal{R}b$ et $b\mathcal{R}c) \Rightarrow a\mathcal{R}c$

Une relation qui possède ces trois propriétés est dite *relation d'équivalence*. Une relation d'équivalence exprime une propriété du type "a et b ont le même *quelque chose*" (par exemple "a et b ont le même reste dans la division par m").

Q. 57. Les relations suivantes $\mathcal{R}_1$ et $\mathcal{R}_2$, définies sur l'ensemble $\mathbb{N}$ des entiers positifs ou nuls, sont-elles des relations d'équivalence ?
1. $a\mathcal{R}_1 b$ si et seulement si a et b s'écrivent avec le même nombre de chiffres en base 10.
2. $a\mathcal{R}_2 b$ si et seulement si $|a - b| \leq 1$.

Attention aux Divisions La congruence est compatible avec les opérations d'addition, soustraction et multiplication, mais pas la division. Cela implique que l'on ne peut pas toujours simplifier. Par exemple

$$2 \times 9 \equiv 2 \times 3 \pmod{12}$$

mais on ne peut pas simplifier par 2 (car on conclurait que $9 \equiv 3 \pmod{12}$, ce qui est faux). Nous verrons plus tard dans quels cas la simplification est possible.

Exemple 7.1 (Reste dans la division par 9) *D'après le théorème d'arithmétique modulaire*

$$10 \equiv 1 \pmod{9}$$

donc pour tout entier $k \geq 0$ *:*

$$10^k \equiv 1^k \equiv 1 \pmod{9}$$

Soit maintenant a un nombre entier positif ; son écriture en base 10 *est* $d_k d_{k-1} ... d_1 d_0$*, ce qui signifie que*

$$a = d_k \times 10^k + d_{k-1} \times 10^{k-1} + ... + d_1 \times 10^1 + d_0 \times 10^0$$

De nouveau d'après le théorème d'arithmétique modulaire :

$$\begin{aligned} a &\equiv d_k \times 1 + d_{k-1} \times 1 + ... + d_1 \times 1 + d_0 \times 1 \pmod{9} \\ a &\equiv d_k + d_{k-1} + ... + d_1 + d_0 \pmod{9} \end{aligned}$$

En d'autres termes, tout nombre entier est congru modulo 9 *à la somme de ses chiffres décimaux. On peut donc calculer le reste d'un nombre dans la division par* 9 *en le remplaçant par la somme de ses chiffres, puis en remplaçant le nombre obtenu par la somme de ses chiffres, etc, jusqu'à obtention d'un nombre à un seul chiffre. Par exemple*

$$298242 \equiv 2 + 9 + 8 + 2 + 4 + 2 \equiv 27 \equiv 2 + 7 \equiv 9 \equiv 0 \pmod{9}$$

donc 298242 *est divisible par* 9*.*

Q. 58. Trouvez une règle semblable pour le reste dans la division par 10, par 3, par 4 ? Appliquer ces règles pour trouver les restes dans la division par 10 [resp. 3 et 4] de $a = 12345678901234567890123456789 0$.

Exemple 7.2 (MOD 97-10 et IBAN) *Supposons que nous voulions transmettre un numéro de n chiffres décimaux (numéro de téléphone, ou de compte en banque, ou adresse IP) en l'écrivant sur un bout de papier. Pour détecter des erreurs simples, telles que l'omission d'un chiffre ou une interversion, nous pouvons y ajouter les deux chiffres de contrôle modulo 97, définis comme le reste dans la division par* 97 *du nombre original de n chiffres. Par exemple, les deux chiffres de contrôle pour le numéro* `021 235 1234` *sont* 95 *car* $212351234 \equiv 95 \pmod{97}$*. Nous écrivons sur le bout de papier :* `021 235 1234 - 95`*.*

Supposons que nous écrivions par erreur `021 253 1234 - 95` *(nous avons interverti deux chiffres). Le destinataire peut détecter l'erreur en recalculant les chiffres de contrôle ; en effet,* $212531234 \equiv 63 \pmod{97}$ *donc les deux chiffres de contrôle devraient être* 63*, et non pas* 95*. La procédure appelée MOD 97-10 est basée sur ce principe, avec les modifications suivantes.*

Dans MOD 97-10, le nombre 97 fait référence au module de la congruence, et 10 au fait que le numéro est interprété comme un nombre écrit en base 10.

1. *Ajouter* 00 *à la fin du numéro.*
2. *Calculer le reste* r *dans la division par* 97 *du numéro ainsi obtenu.*
3. *Les deux chiffres de contrôle MOD 97-10 sont les deux chiffres du complément à* 98 *de* r*. Remplacer le* 00 *final par ces deux chiffres.*
4. *Pour vérifier la validité d'un numéro, vérifier que le reste dans la division par* 97 *est égal à* 1.

Appliquons la procédure au numéro `021 235 1234` *:*

1. $x = 21235123400$
2. $21235123400 \equiv 91 \pmod{97}$
3. *Le complément à* 98 *de* 91 *est* $7 = 98 - 91$*. Les deux chiffres de contrôle sont donc* 07*. Le numéro avec chiffres de contrôle MOD 97-10 est* `021 235 1234 - 07`
4. *Vérification :* $21235123407 \equiv 1 \pmod{97}$ *donc le numéro est valable. Si nous recevons le numéro* `021 253 1234 - 07`*, nous obtenons* $21253123407 \equiv 2 \pmod{97}$*, donc ce numéro n'est pas valable.*

Pourquoi la vérification fonctionne-t-elle de la façon décrite dans l'étape 4 ? Le théorème d'arithmétique modulaire nous donne la réponse. En effet, soit x *le nombre obtenu en interprétant le numéro de départ en base* 10 *et en ajoutant deux zéros (dans l'exemple,* $x = 21235123400$*). Le reste* r *de l'étape 2 vérifie*

$$x \equiv r \pmod{97}$$

Soit x' *le nombre avec les chiffres de contrôle (c'est à dire, le nombre obtenu en remplaçant les deux zéros finaux par les chiffres de contrôle ; dans l'exemple,* $x' = 21235123407$*). Nous avons* $x' = x + (98 - r)$*, donc par l'arithmétique modulaire :*

$$\begin{aligned} x' &\equiv x + 98 - r \pmod{97} \\ &\equiv x + 1 - r \pmod{97} \quad \textit{car } 98 \equiv 1 \pmod{97} \\ &\equiv r + 1 - r \pmod{97} \quad \textit{car } x \equiv r \pmod{97} \\ &\equiv 1 \pmod{97} \end{aligned}$$

Le calcul du reste d'une division d'un très grand nombre entier peut poser des problèmes pratiques (overflow). Cela peut arriver par exemple si nous appliquons MOD 97-10 aux numéros IBAN, même avec des entiers longs. Le théorème d'arithmétique modulaire apporte une solution. Par exemple, supposons que $x = 1234567890123456789$ *et que nous voulions calculer le reste* r *de* x *dans la division par* 97*. Soit* $x' = 123456789012345678$ *le nombre obtenu en supprimant le dernier chiffre et* r' *son reste dans la division par* 97*. Alors* $x = 10x' + 9$ *et donc*

$$r \equiv 10r' + 9 \pmod{97}$$

et le calcul de r' *est un peu plus simple puisque* x' *comporte un chiffre de moins.*

Nous pouvons donc utiliser une procédure récursive, en traitant un numéro comme une chaîne de n *symboles décimaux plutôt que comme un entier – voir ci-contre.*

Exemple 7.3 (⋆ Preuve par 9) *Vous êtes naufragé(e) sur une île déserte, sans aucun instrument de calcul, et devez calculer votre position d'après*

L'*IBAN* (International Bank Account Number) est un format international pour les numéros de comptes bancaires. Le numéro commence par le code du pays, suivi des deux chiffres de contrôle MOD 97-10, puis du numéro de compte proprement dit. Les symboles sont des chiffres décimaux ou des lettres de l'alphabet latin majuscule. Tous les numéros d'un même pays ont le même nombre de symboles.

Ainsi tous les numéros suisses commencent par `CH`, suivis de deux chiffres de contrôle, et comportent 21 symboles en tout. Un numéro suisse pourrait être par exemple `CH54 0024 3000 1234 5678 9`.

Les deux chiffres de contrôle (ici `54`) sont obtenus comme suit :

(a) Déplacer les 2 premiers symboles (code de pays) à la fin et supprimer les deux chiffres de contrôle. Nous obtenons `0024 3000 1234 5678 9CH`.

(b) Remplacer les symboles non décimaux (c'est à dire les lettres) par deux chiffres, selon $A = 10, B = 11, ..., Z = 35$. Nous obtenons `0024 3000 1234 5678 91217`.

(c) Calculer les deux chiffres de contrôle par la procédure MOD 97-10 appliquée au numéro ainsi obtenu. Nous obtenons `54`.

Pour vérifier la validité d'un numéro IBAN, il suffit d'applique l'étape 4 de MOD 97-10, c'est à dire :

(a) Déplacer les 4 premiers symboles (code de pays et chiffres de contrôle) à la fin. Nous obtenons `0024 3000 1234 5678 9CH54`.

(b) Remplacer les symboles non décimaux (c'est à dire les lettres) par deux chiffres, selon $A = 10, B = 11, ..., Z = 35$. Nous obtenons `0024 3000 1234 5678 9121754`.

(c) Le reste dans la division par 97 du nombre ainsi obtenu doit être égal à 1.

```
1: function ResteGros(x, m, b)
2:         ▷ x : suite d'entiers entre 0 et b − 1
3:                                  ▷ m : entier ≥ 2
4:                                  ▷ b : entier ≥ 2
5:     n ← longueur de la suite x
6:     x_0 ← dernier chiffre de x
7:     if n = 1 then
8:         r ← x_0 mod m
9:     else
10:        x' ← enlever dernier élément de x
11:        r' ← ResteGros(x', m)
12:        r ← (br' + x_0) mod m
13:    end if
14:    return (r)
15: end function
```

Fonction qui calcule le reste de x dans la division par m, où x est traité comme une chaîne de chiffres en base b plutôt que comme un entier, pour éviter les problèmes d'overflow. Pour MOD 97-10, prendre $m = 97$ et $b = 10$.

le soleil et votre montre. Pour cela vous devez effectuer à la main (dans le sable) une monstrueuse multiplication de deux grands nombres, par exemple $c = ab$ avec $a = 23765$ et $b = 79087$. Pour la plupart des gens, les chances de se tromper dans une telle opération sont grandes. Vous avez fort heureusement appris la preuve par 9, qui vous permet de vérifier au moins partiellement vos calculs.

Pour cela vous calculez les restes dans la division par 9 de a et b, disons a' et b', puis le reste c' du produit $a'b'$. Par le théorème d'arithmétique modulaire, le reste de c dans la division par 9 doit aussi être égal à c', ce qui donne un moyen de vérifier si c est correct. Ainsi :

$$\begin{aligned} a = 23765 &\equiv 2+3+7+6+5 \equiv 5 \pmod 9 \\ b = 79087 &\equiv 7+9+0+8+7 \equiv 4 \pmod 9 \\ c' = 5 \times 4 &\equiv 2+0 \equiv 2 \pmod 9 \end{aligned}$$

donc on sait, avant de faire la multiplication, que l'on doit avoir $c \equiv 2 \pmod 9$. On dit que 2 est un "checksum" ou "chiffre de contrôle" de la multiplication.

Nous trouvons $c = 1879502555$, donc $c \equiv 2 \pmod 9$ et le chiffre de contrôle est correct.

Q. 59. La preuve par 9 permet-elle de détecter les erreurs simples, c'est à dire portant sur un seul chiffre ?

Exemple 7.4 (⋆ Checksum IP ou UDP) *Les paquets de données utilisés dans l'internet utilisent un code détecteur d'erreur, appelé checksum IP, qui protège certaines informations importantes (notamment l'adresse de destination). Il est calculé de la façon suivante. Supposons que nous ayons un champ C de $16n$ bits à protéger, c'est à dire une suite de $2n$ octets. Nous considérons cette suite de bits comme la représentation en base 2 d'un nombre x. Puis nous calculons le reste r de x dans la division par $m = 2^{16} - 1$. Notons que ce nombre m s'écrit en binaire $m = 1111\ 1111\ 1111\ 1111_b$ (l'indice $_b$ indique que le nombre est écrit en représentation binaire).*

Le *complément à 1* d'une suite de chiffres binaires est la suite obtenue en remplaçant les 0 par des 1 et inversement. Par exemple, le complément à 1 de 001101_b est 110010_b.

Soit une suite de 16 bits et soit x le nombre dont elle est la représentation en base 2. Soit x' le nombre dont la représentation en base 2 est son complément à 1. Alors $x + x' = 1111\ 1111\ 1111\ 1111_b$ donc $x' = 2^{16} - 1 - x$.

Le checksum est le "complément à 1" de r. Par exemple, si on nous donne le nombre suivant de 32 bits (écrit en binaire) :

$$x = 1000\ 0001\ 0000\ 0011\ 1000\ 0000\ 0001\ 0010_b$$

le reste dans la division par $2^{16} - 1$ est

$$r = 0000\ 0001\ 0001\ 0110_b$$

donc le checksum est

$$c = 1111\ 1110\ 1110\ 1001_b$$

Le checksum c est transmis en même temps que le champ à protéger C ; le destinataire refait les mêmes calculs sur le champ C reçu et vérifie s'il trouve un checksum égal à c. Si la réponse est négative, le paquet est certainement en erreur et est détruit.

Calcul du checksum par la somme en complément à 1. *Comment peut-on calculer efficacement ce checksum ? En fait, c'est la même idée que*

pour la preuve par 9. *Pour voir pourquoi, regroupons les bits par blocs de* 16 *bits et soient* $W_{n-1}...W_1, W_0$ *les nombres qu'ils représentent en base* 2. *Nous avons donc :*

$$x = 2^{16(n-1)} W_{n-1} + ... + 2^{16} W_1 + W_0 \qquad (7.2)$$

Avec l'exemple précédent, cela donne

$$\begin{aligned} x &= \overbrace{1000\ 0001\ 0000\ 0011}^{W_1}\ \overbrace{1000\ 0000\ 0001\ 0010}^{W_0} \\ &= 2^{16} W_1 + W_0 \end{aligned}$$

Maintenant $2^{16} \equiv 1 \bmod m$ *puisque* $2^{16} = m + 1$. *Donc*

$$x \equiv W_{n-1} + ... + W_1 + W_0 \bmod m$$

c'est à dire que l'on peut remplacer x *par la somme de ses blocs de* 16 *bits (interprétés comme des nombres écrits en base* 2*). Comme avec la preuve par* 9, *il suffit de répéter l'opération jusqu'à obtenir un résultat* r' *qui tienne sur un seul bloc de* 16 *bits. Ainsi pour notre exemple :*

$$\begin{aligned} W_1 &= 1000\ 0001\ 0000\ 0011 \\ W_2 &= 1000\ 0000\ 0001\ 0010 \\ W_1 + W_0 &= \overbrace{0000\ 0000\ 0000\ 0001}^{W'_1}\ \overbrace{0000\ 0001\ 0001\ 0101}^{W'_0} \\ W'_1 + W'_0 &= \overbrace{0000\ 0001\ 0001\ 0110}^{W''_0} = r' \end{aligned}$$

A chaque étape le nombre obtenu est congru au précédent modulo m, *donc finalement*

$$r' \equiv x \pmod m$$

Le nombre r' *est entre* 0 *et* m *et est appelé la somme en complément à* 1 *de* x *(voir ci-contre). Il est presque toujours égal à* r, *le reste de* x *dans la division par* m. *Plus précisément,* r *est obtenu par :*

$$\textit{si } r' = 1111\ 1111\ 1111\ 1111_b \textit{ alors } r = 0 \textit{ sinon } r = r'$$

puis le checksum c *est le complément à* 1 *de* r. *En résumé le calcul du checksum* c *se fait de la façon suivante :*

calculer r', *la somme en complément à* 1 *de* x
$c' = 2^{16} - r'$ *(i.e. prendre le complément à* 1
si $(c' = 0)$ $c = 1111\ 1111\ 1111\ 1111_b$ *sinon* $c = c'$

Notons que le checksum ainsi calculé ne peut jamais être égal à 0.

Pour rendre plus lisibles les calculs en base 2 on utilise souvent la représentation hexadécimale, c'est à dire en base 16. Les chiffres hexadécimaux sont $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f$ et représentent les nombres entiers de 0 à 15. Ainsi par exemple $7 + 8 = f$.

Un chiffre hexadécimal représente un bloc de 4 bits. Ainsi $2^{16} - 1$, qui est en binaire 1111 1111 1111 1111, se note en hexadécimal $ffff$. Avec l'exemple ci-contre :

$$x = 8103\ \ 8012$$

et le calcul du checksum IP peut alors s'écrire :

$W_1 = 8103$
$W_0 = 8012$
$W_1 + W_0 = 0001\ \ 0115$
$W'_1 + W'_0 = 0116 = r$
$c = fee9$

Notons que c est obtenu par l'opération $c = ffff - r$.

La *somme en complément à 1 sur 16 bits*, sc1-16(x), est définie formellement de la façon suivante. x est une suite de $16n$ bits, interprétée comme un entier positif ou nul écrit en base 2. Soient $W_{n-1}, ...W_0$ les n entiers positifs ou nuls correspondant aux blocs de 16 bits, comme en Eq.(7.2).

si $n = 1$
  alors sc1-16$(x) = W_0$
  sinon sc1-16$(x) =$
    sc1-16$(W_{n-1} + ... + W_0)$

où la dernière addition est l'addition usuelle des entiers positifs ou nuls.

Notons que $x \equiv$ sc1$(x) \pmod m$. De plus, sc1(x) est égal au reste de x dans la division par $m = 2^{16} - 1$ sauf si $x > 0$ et $x \equiv 0 \pmod m$ auquel cas sc1$(x) = m$.

8
Arithmétique Modulaire

L'étude systématiques des congruences nous amène dans des nouveaux ensembles de "nombres", les ensembles $\mathbb{Z}/m\mathbb{Z}$; ils sont utilisés en cryptographie et en codage d'erreurs, comme nous le verrons. L'étude des calculs dans ces ensembles s'appelle l'*arithmétique modulaire*.

8.1 Les Ensembles $\mathbb{Z}/m\mathbb{Z}$

Définition 8.1 (Classe de Congruence) *Soit $m \geq 2$ un entier fixé (le "module"). Pour tout entier a, on appelle classe de congruence de a modulo m, et on note $[a]_m$ l'ensemble des entiers a' tels que $a \equiv a' \pmod{m}$.*

Par exemple, avec $m = 2$, $[24]_2$ est l'ensemble des entiers pairs, alors que $[23]_2$ est l'ensemble des entiers impairs. Notons que $[24]_2 = [0]_2 = [-100]_2$ et $[-1]_2 = [1]_2 = [3]_2$.

Notons que

$$(a \equiv a' \pmod{m}) \Leftrightarrow ([a]_m = [a']_m) \tag{8.1}$$

Si c est une classe de congruence modulo m et $c = [a]_m$ on dit que a est un *représentant* de c. Ainsi, 24, 0 et -100 sont trois représentants de $[0]_2$.

Il y a exactement 2 classes de congruence modulo 2, l'ensemble des entiers pairs et l'ensemble des entiers impairs ; ces deux classes peuvent être notées de différentes façons, mais le plus simple est de les noter $[0]_2$ et $[1]_2$. Plus généralement :

Théorème 8.1 *Il y a exactement m classes de congruence modulo m, ce sont $[0]_m, [1]_m, ..., [m-1]_m$.*

☺*Preuve :* Nous avons à montrer que (i) les m classes dans le théorème sont toutes distinctes et (ii) toute classe dans $\mathbb{Z}/m\mathbb{Z}$ est égale à l'une de ces m classes.

(i) Nous raisonnons par l'absurde et supposons donc que i et j sont dans $\{0, 1, ...m-1\}$ avec $i \neq j$ et $[i]_m = [j]_m$.

Notons d'abord que le reste de i dans la division par m est i car $i = 0 \times i + i$ et $0 \leq i \leq m-1$, et la même chose vaut pour j. Par

Soit $\mathcal{R}$ une relation d'équivalence définie sur un ensemble A. La *classe d'équivalence* $c\ell(a)$ de $a \in A$, est l'ensemble des $a' \in A$ tels que $a\mathcal{R}a'$. Le fait que $\mathcal{R}$ soit une relation d'équivalence entraîne que

(i) les classes d'équivalence forment une partition de A, c'est à dire que tout $a \in A$ appartient à une classe d'équivalence et une seule.

(ii) $(a\mathcal{R}a') \Leftrightarrow (c\ell(a) = c\ell(a'))$

Q. 60. Soit $\mathcal{R}_1$ la relation d'équivalence définie en **Q.57**. Quelles sont les classes d'équivalence ?

Soit A un ensemble et $\mathcal{A}$ un ensemble de sous-ensembles de A. On dit que $\mathcal{A}$ est une *partition* de A si tout élément de A appartient à un sous-ensemble de $\mathcal{A}$ et un seul.

Par exemple, soit $A = \mathbb{Z}$, $A_0 = \{... -2, 0, 2, 4, ...\}$, $A_1 = \{... -3, -1, 1, 3, ...\}$ (ensembles des entiers resp. pairs et impairs) et $\mathcal{A} = \{A_0, A_1\}$. $\mathcal{A}$ est une partition de A.

Eq.(8.1), $i \equiv j \pmod m$ donc i et j ont même reste dans la division par m. Donc $i = j$, ce qui est une contradiction.

(ii) Soit $[a]_m$ une classe de congruence modulo m. Soit r le reste de a dans la division par m. Nous avons $[a]_m = [r]_m$ et $0 \leq r \leq m-1$. ☺□

Définition 8.2 *On note $\mathbb{Z}/m\mathbb{Z}$ l'ensemble des classes de congruence modulo m.*

Notons qu'un élément a de $\mathbb{Z}/m\mathbb{Z}$ peut s'écrire de différentes façons, par exemple $a = [18]_{12} = [6]_{12}$, mais le Théorème 8.1 exprime qu'il existe une seule façon d'écrire $a = [r]_m$ avec $0 \leq r \leq m-1$; on l'appelle *forme réduite*. Ainsi la forme réduite de $[18]_{12}$ est $[6]_{12}$. A partir de maintenant nous pouvons voir $\mathbb{Z}/m\mathbb{Z}$ tout simplement comme un ensemble à m éléments :

$$\mathbb{Z}/m\mathbb{Z} = \{[0]_m, [1]_m, ..., [m-1]_m\} \tag{8.2}$$

8.2 Opérations dans $\mathbb{Z}/m\mathbb{Z}$

Nous allons définir une addition et une multiplication dans $\mathbb{Z}/m\mathbb{Z}$. Notons que d'après le Théorème 7.9, si $[a]_m = [a']_m$ et $[b]_m = [b']_m$ alors $[a+b]_m = [a'+b']_m$ et $[ab]_m = [a'b']_m$. On peut donc définir la somme de deux classes de congruence $[a]_m$ et $[b]_m$ comme la classe de congruence de $[a+b]_m$, sans se soucier des représentants choisis, et la même chose vaut pour la multiplication :

Définition 8.3 *On définit la somme et le produit dans $\mathbb{Z}/m\mathbb{Z}$ par :*

$$[a]_m + [b]_m = [a+b]_m \tag{8.3}$$
$$[a]_m\,[b]_m = [ab]_m \tag{8.4}$$

Comme d'habitude, la multiplication est notée de différentes façons, à savoir $[a]_m\,[b]_m$, $[a]_m \cdot [b]_m$ ou $[a]_m \times [b]_m$, qui sont toutes synonymes.

L'addition a les propriétés suivantes :

Associativité : $[a]_m + ([b]_m + [c]_m) = ([a]_m + [b]_m) + [c]_m$
Elément neutre : $[0]_m$ est l'élément neutre : $[a]_m + [0]_m = [a]_m$
Elément symétrique : $[-a]_m$ est l'opposé de $[a]_m$: $[a]_m + [-a]_m = [0]_m$
Commutativité : $[a]_m + [b]_m = [b]_m + [a]_m$

La multiplication a les propriétés suivantes :

Associativité : $[a]_m([b]_m[c]_m) = ([a]_m[b]_m)[c]_m$
Elément neutre : $[1]_m$ est l'élément neutre : $[a]_m[1]_m = [a]_m$
Commutativité : $[a]_m[b]_m = [b]_m[a]_m$

Les deux opérations ont la propriété suivante :

Distributivité : $[a]_m([b]_m + [c]_m) = ([a]_m[b]_m) + ([a]_m[c]_m)$

$\mathbb{Z}/3\mathbb{Z}$	$+$	0	1	2
	0	0	1	2
	1	1	2	0
	2	2	0	1

$\mathbb{Z}/3\mathbb{Z}$	$\times$	0	1	2
	0	0	0	0
	1	0	1	2
	2	0	2	1

$\mathbb{Z}/4\mathbb{Z}$	$+$	0	1	2	3
	0	0	1	2	3
	1	1	2	3	0
	2	2	3	0	1
	3	3	0	1	2

$\mathbb{Z}/4\mathbb{Z}$	$\times$	0	1	2	3
	0	0	0	0	0
	1	0	1	2	3
	2	0	2	0	2
	3	0	3	2	1

TABLE 8.1: Addition et Multiplication dans $\mathbb{Z}/3\mathbb{Z}$ et $\mathbb{Z}/4\mathbb{Z}$. Par souci de légèreté, nous notons dans ces tables $0, 1, 2$ au lieu de $[0]_3, [1]_3, [2]_3$ et $0, 1, 2, 3$ au lieu de $[0]_4, [1]_4, [2]_4, [3]_4$.

Q. 61. Résoudre pour $m = 3$ puis pour $m = 4$ les équations suivantes :

1. $[x]_m^2 = [1]_m$
2. $[2]_m[x]_m = [0]_m$
3. $[x]_m + [x]_m = [0]_m$

Un ensemble A muni de deux opérations, $+$ et $\times$ qui satisfont les huit propriétés ci-contre (depuis *Associativité* jusqu'à *Distributivité*) est appelé *anneau commutatif*. $(\mathbb{Z}, +, \times)$ et $(\mathbb{Z}/m\mathbb{Z}, +, \times)$ sont des anneaux commutatifs.

Comme pour les nombres entiers, on adopte la convention que la multiplication a priorité sur l'addition, ce qui permet d'économiser des parenthèses :

$$([a]_m[b]_m) + ([a]_m[c]_m) = [a]_m[b]_m + [a]_m[c]_m$$

On note $-[a]_m$ l'opposé de $[a]_m$. Nous pouvons donc écrire :

$$-[a]_m = [-a]_m$$

De même, on peut définir la multiplication par un entier ordinaire, de la façon suivante. Pour k entier positif :

$$k[a]_m \stackrel{\text{def}}{=} \overbrace{[a]_m + ... + [a]_m}^{k \text{ fois}} \quad \text{et} \quad (-k)[a]_m \stackrel{\text{def}}{=} \overbrace{[-a]_m + ... + [-a]_m}^{k \text{ fois}}$$

et pour $k = 0$: $0[a]_m \stackrel{\text{def}}{=} [0]_m$. Ainsi $3[11]_{18} = [11]_{18} + [11]_{18} + [11]_{18}$ et $(-3)[11]_{18} = [-11]_{18} + [-11]_{18} + [-11]_{18}$.

Q. 62. Mettre sous forme réduite les expressions suivantes :

1. $([-5]_7)^2 + [-4]_7\,[-3]_7$
2. $2\,([3]_4 + [5]_4) - 5\,([3]_4)^2$
3. $([298242]_9)^{1000}$

Q. 63. Que représente l'expression $[3]_4 + [4]_3$?

Les propriétés plus haut entraînent alors que, pour $k \in \mathbb{Z}$ et $a \in \mathbb{Z}$:

$$k[a]_m = [ka]_m$$

Par exemple,

$$\begin{aligned} 3[11]_{18} &= [11]_{18} + [11]_{18} + [11]_{18} = [33]_{18} = [3 \cdot 11]_{18} = [15]_{18} \\ (-3)[11]_{18} &= [-11]_{18} + [-11]_{18} + [-11]_{18} = [-33]_{18} = [3 \cdot (-11)]_{18} = [3]_{18} \end{aligned}$$

En résumé, l'addition et la multiplication dans $\mathbb{Z}/m\mathbb{Z}$ ont les mêmes propriétés que l'addition et la multiplication ordinaires, à l'exception de tout ce qui concerne la division et, par conséquent, les simplifications. Ainsi $ab = 0$ n'implique pas, en général, $a = 0$ ou $b = 0$. Par exemple

$$[2]_6[3]_6 = [0]_6 \text{ alors que } [2]_6 \neq [0]_6 \text{ et } [3]_6 \neq [0]_6$$

Si $ab = 0$ alors que ni a ni b ne sont nuls on dit qu'ils sont des *diviseurs de zéro*. Ainsi $[2]_6$ et $[3]_6$ sont des diviseurs de zéro. De même, $ab = ac$ avec $a \neq 0$ n'implique pas forcément $b = c$. Par exemple

Q. 64. Y a-t-il des diviseurs de zéro dans $\mathbb{Z}/3\mathbb{Z}$? dans $\mathbb{Z}/4\mathbb{Z}$?

$$[2]_{12}[9]_{12} = [2]_{12}[3]_{12} \text{ alors que } [2]_{12} \neq [0]_{12} \text{ et } [9]_{12} \neq [3]_{12}$$

Pour $k \geq 0$ entier on définit la $k$ième puissance par

$$([a]_m)^k \stackrel{\text{def}}{=} \overbrace{[a]_m \cdot ... \cdot [a]_m}^{k \text{ fois}}$$

et $([a]_m)^0 = [1]_m$. Nous pouvons facilement vérifier que $([a]_m)^k = [a^k]_m$, i.e. les règles usuelles s'appliquent (tant qu'il n'y a pas de division, donc pour des exposants $k \geq 0$).

Exemple 8.1 (Calcul d'une Puissance) *Le calcul d'une puissance en arithmétique modulaire est particulièrement simple, il suffit de décomposer*

l'exposant. Calculons par exemple $([3]_7)^{12}$ *(nous laissons tomber l'indice* 7*, ce qui est toujours possible si nous nous rappelons que toutes les opérations sont dans* $\mathbb{Z}/7\mathbb{Z}$*) :*

$$[3]^{12} = ([3]^2)^6 = [9]^6 = [2]^6 = \left([2]^3\right)^2 = [8]^2 = [1]^2 = [1]$$

Par contre, l'opération inverse, c'est à dire trouver un entier x *s'il existe tel que* $[a]_m^x = [b]_m$ *est nettement plus difficile et aucun algorithme efficace n'est connu, si* m *est très grand (cette opération inverse s'appelle le logarithme discret). L'application* $x \mapsto [a]_m^x$ *est un exemple d'application à "sens unique".*

Q. 65. Trouver un entier $x \geq 0$ tel que $([2]_{13})^x = [3]_{13}$.

8.3 Eléments Inversibles de $\mathbb{Z}/m\mathbb{Z}$

Nous avons vu que la division pouvait poser problème dans $\mathbb{Z}/m\mathbb{Z}$, mais il est quand même possible de clarifier les choses.

Théorème et Définition 8.4 *Soit* $m \geq 2$*; on dit que* $a \in \mathbb{Z}/m\mathbb{Z}$ *est inversible s'il existe* $a' \in \mathbb{Z}/m\mathbb{Z}$ *tel que* $aa' = [1]_m$*. Un tel élément* a'*, s'il existe, est unique. Il est appelé l'inverse de* a *et est noté* a^{-1}*.*

Par exemple $2 \times 7 \equiv 1 \pmod{13}$ donc $[2]_{13}[7]_{13} = [1]_{13}$. Donc $[2]_{13}$ est inversible, et $[2]_{13}^{-1} = [7]_{13}$. Nous pouvons aussi conclure que $[7]_{13}$ est inversible, et $[7]_{13}^{-1} = [2]_{13}$.

Par contre $[2]_{12}$ n'est pas inversible. En effet, en calculant $[2]_{12} \cdot a'$ pour tous les $a' \in \mathbb{Z}/12\mathbb{Z}$ nous trouvons successivement : $[0]_{12}, [2]_{12}, [4]_{12}, [6]_{12}, [8]_{12}, [10]_{12}, [0]_{12}, [2]_{12}, [4]_{12}, [6]_{12}, [8]_{12}, [10]_{12}$. Nous n'obtenons jamais $[1]_{12}$.

Q. 66. Montrer que $[0]_m$ n'est jamais inversible pour tout $m \geq 2$.
Est-il vrai que $[1]_m$ est inversible pour tout $m \geq 2$?

☺*Preuve :* Soient a' et a'' tels que $aa' = [1]_m$ et $aa'' = [1]_m$. En multipliant la première égalité par a'', et comme $[1]_m$ est élément neutre et la multiplication est commutative et associative , nous obtenons

$$a'' = a''[1]_m = a''(aa') = (aa'')a' = [1]_m a' = a'$$

donc $a' = a''$. ☺□

Théorème 8.2 *Si* $a \in \mathbb{Z}/m\mathbb{Z}$ *est inversible alors* a^{-1} *l'est aussi et*

$$\left(a^{-1}\right)^{-1} = a$$

☺*Preuve :* Par définition, on a $\cdot a^{-1}a = [1]_m$ donc a^{-1} satisfait la définition d'inversibilité et son inverse est a. ☺□

L'existence d'un inverse, et éventuellement son calcul sont des problèmes numériquement "faciles", résolus par les théorèmes suivants. Nous commençons par une méthode pour calculer le PGCD sans devoir factoriser en nombres premiers :

Théorème 8.3 (Algorithme d'Euclide) *Soient* a *et* b *deux entiers avec* $b \neq 0$*, et soit* $a = bq + r$ *la division euclidienne de* a *par* b*. Alors*

$$pgcd(a, b) = pgcd(b, r) \tag{8.5}$$

En particulier, si $r = 0$, $pgcd(a, b) = b$.

Preuve : Nous allons montrer que, pour tout entier positif d :

$$(d \text{ divise } a \text{ et } b) \Leftrightarrow (d \text{ divise } b \text{ et } r) \tag{8.6}$$

(1) ($\Rightarrow$) Soit d qui divise a et b ; or $\frac{r}{d} = \frac{a}{d} - q\frac{b}{d}$ est un entier donc d divise r, donc il divise b et r.

(2) ($\Leftarrow$) se montre de la même façon que (1)

Donc Eq.(8.6) est vraie, donc les diviseurs communs à a et b sont les mêmes que les diviseurs communs à b et r donc les PGCD sont les mêmes. □

Le Théorème 8.3 donne un algorithme *récursif* (c'est à dire qui fait appel à lui même) pour calculer le PGCD. En effet, il suffit d'appeler b le plus petit des deux nombres ; Eq.(8.5) permet alors de remplacer (a, b) par (b, r) avec $0 \leq r < b$, puis de recommencer jusqu'à obtenir un reste nul. A chaque étape le reste diminue d'au moins 1, donc l'algorithme s'arrête forcément, en au plus b étapes.

Appliquons le Théorème 8.3 au calcul du PGCD de 120 et 22 :

a	b	r	pgcd$(122, 22)$
120	22	10	$=$ pgcd$(22, 10)$
22	10	2	$=$ pgcd$(10, 2)$
10	2	0	$= 2$

Au lieu d'utiliser le Théorème 8.3, nous pouvons employer la méthode de décomposition en facteurs premiers :

$$\begin{array}{rcccccc} 120 & = & 2^3 & \cdot & 3 & \cdot & 5 \\ 22 & = & 2 & \cdot & 11 & & \end{array}$$

ce qui donne le même résultat pgcd$(122, 22) = 2$.

Théorème 8.4 (Identité de *Bézout*) *Soient a et b deux entiers ; il existe deux nombres entiers u et v tels que*

$$au + bv = pgcd(a, b) \tag{8.7}$$

Si nous voulons que l'identité de Bézout soit vraie pour $a = b = 0$ il faut convenir que pgcd$(0, 0) = 0$.

Preuve : (1) Nous montrons par récurrence sur $n \geq 1$ que si $1 \leq a \leq n$ ou $1 \leq b \leq n$ alors l'identité de Bézout est vraie.

(étape d'initialisation :) Supposons $n = 1$, donc soit $a = 1$ soit $b = 1$; supposons que $a = 1$ (sinon c'est pareil), donc pgcd$(a, b) = 1$ et Eq.(8.7) est vraie avec $u = 1$, $v = 0$.

(étape de récurrence :) Supposons que l'hypothèse de récurrence soit vraie jusqu'à n et montrons qu'elle est vraie pour $n + 1$. Supposons que $a > b$ (sinon nous échangeons les rôles de a et b) donc nous pouvons supposer que $b \leq n + 1$. Effectuons la division euclidienne de a par b : $a = bq + r$. Si $r = 0$ alors pgcd$(a, b) = b$; il suffit de prendre $u = 0$, $v = 1$ et Eq.(8.7) est satisfaite. Si au contraire $r > 0$, notons que $r \leq b - 1 \leq n$ donc nous pouvons appliquons l'hypothèse de récurrence à (b, r) ; il existe des entiers u_1 et v_1 tels que pgcd$(b, r) = u_1 b + v_1 r$. Or pgcd$(b, r) =$ pgcd(a, b) d'après le Théorème 8.3. En combinant avec $a = bq + r$ nous obtenons

$$av_1 + b(u_1 - qv_1) = \text{pgcd}(a, b) \tag{8.8}$$

ce qui montre que Eq.(8.7) est satisfaite.

(2) Nous avons donc montré que Eq.(8.7) est satisfaite pour a et b positifs. Etudions maintenant le cas où a ou b est négatif. Supposons par exemple que $a < 0$ et $b > 0$, notons que pgcd$(a, b) =$ pgcd$(|a|, b)$ donc d'après (1) on peut trouver u_1 et v_1 tels que $u_1 |a| + vb =$ pgcd(a, b). Donc Eq.(8.7) est satisfaite avec

$u = -u_1$ et $v = v_1$. La même chose vaut si $a > 0$ et $b < 0$ ou si a et b sont négatifs.

Il reste le cas où l'un des deux nombres est nul ; supposons par exemple que $a = 0$ et $b \neq 0$; alors $\text{pgcd}(a, b) = b$ et Eq.(8.7) est satisfaite avec $u = 0, v = 1$. Si les deux entiers sont nuls, nous avons convenu que $\text{pgcd}(a, b) = 0$ et Eq.(8.7) est satisfaite. □

Q. 67. Soient a et m deux entiers ≥ 2. Montrer que $[a]_m$ est inversible si et seulement si $[m]_a$ est inversible.

Théorème 8.5 (Eléments Inversibles de $\mathbb{Z}/m\mathbb{Z}$) *Soient a et m deux entiers avec $m \geq 2$; $[a]_m$ est inversible si et seulement si a et m sont premiers entre eux.*

Les éléments inversibles se trouvent facilement sur la table de multiplication. Par exemple (Table 8.1), dans $\mathbb{Z}/3\mathbb{Z}$ les éléments $[1]_3$ et $[2]_3$ sont inversibles (1 et 2 sont premiers avec 3), par contre $[0]_3$ ne l'est pas (0 n'est pas premier avec 3). Dans $\mathbb{Z}/4\mathbb{Z}$, les éléments inversibles sont $[1]_4$ et $[3]_4$ (1 et 3 sont premiers avec 4) ; les éléments non inversibles sont $[0]_4$ et $[2]_4$ (0 et 2 ne sont pas premiers avec 4).

Preuve : ($[a]_m$ inversible) $\Rightarrow$ (a et m premiers entre eux) : Soit a' un représentant de $([a]_m)^{-1}$. Nous avons $aa' \equiv 1 \pmod{m}$, donc le reste de aa' dans la division par m est 1 et il existe un entier v (le quotient) tel que

$$aa' = mv + 1$$

Soit d un entier positif qui divise à la fois a et m, nous avons donc $a = \alpha d$ et $m = \mu d$ avec α et μ entiers. En combinant avec l'équation précédente :

$$(\alpha - \mu)d = 1$$

donc d est un diviseur de 1 donc $d = 1$. Donc tout diviseur positif de a et m est égal à 1, donc $\text{pgcd}(a, m) = 1$, c'est à dire que a et m sont premiers entre eux.

(a et m premiers entre eux) $\Rightarrow$ ($[a]_m$ inversible) : d'après l'identité de Bézout il existe des entiers u et v tels que $au + mv = 1$ donc

$$[a]_m[u]_m + [m]_m[v]_m = [1]_m$$

or $[m]_m = [0]_m$ donc $[a]_m[u]_m = [1]_m$, c'est-à-dire que $[a]_m$ est inversible et son inverse est $[u]_m$. □

Simplifications en Arithmétique Modulaire Supposons que

$$ax \equiv ay \pmod{m} \tag{8.9}$$

Nous avons envie de simplifier par a, mais nous avons vu en page 63 que ce n'est pas toujours légitime. Le Théorème 8.5 nous permet d'en dire plus, dans le cas où a est premier avec m. Notons tout d'abord que l' Eq.(8.9) est équivalente à

$$[a]_m[x]_m = [a]_m[y]_m$$

Si a est premier avec m, alors $[a]_m$ est inversible, donc nous pouvons "diviser" par $[a]_m$ (c'est à dire multiplier par l'inverse de $[a]_m$), ce qui donne

$$[x]_m = [y]_m$$

que nous pouvons ré-écrire sous la forme

$$x \equiv y \pmod{m}$$

En d'autres termes, nous pouvons simplifier a dans l'Eq.(8.9) pourvu que a soit premier avec m.

Une conséquence du Théorème 8.5, que nous allons souvent utiliser dans la suite, est :

Théorème 8.6 (Cas de $\mathbb{Z}/p\mathbb{Z}$ avec p premier) *Si p est un nombre premier, tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $[0]_p$ sont inversibles.*

☺*Preuve :* Un élément x non nul de $\mathbb{Z}/p\mathbb{Z}$ peut s'écrire $x = [a]_p$ avec $1 \leq a \leq p-1$. Par le Théorème 7.6, a et p sont premiers entre eux. Donc $[a]_p$ est inversible ☺□

Q. 68. Vrai/Faux (a et m sont des entiers avec $m \geq 2$) :

1. a est premier avec $m \Rightarrow [a]_m$ n'est pas un diviseur de 0
2. $[a]_m$ est un diviseur de $0 \Rightarrow a$ et m ont un diviseur commun ≥ 2

Exemple 8.2 (MOD 97-10) *Les chiffres de contrôle MOD 97-10 peuvent détecter toutes les interversions de deux chiffres contiguës, par exemple* `021 235 1234` *remplacé par* `021 253 1234`*. En effet, soit x le numéro original, y compris les chiffres de contrôle, et x' le numéro erroné, dans lequel les chiffres contigus c_k et c_{k+1} sont intervertis. Alors*

$$\begin{aligned} x &= c_{n-1}10^{n-1} + \ldots + c_{k+1}10^{k+1} + c_k 10^k + \ldots + c_0 \\ x' &= c_{n-1}10^{n-1} + \ldots + c_k 10^{k+1} + c_{k+1}10^k + \ldots + c_0 \\ x - x' &= 10^k (c_{k+1} - c_k)(10-1) = 9 \cdot 10^k (c_{k+1} - c_k) \end{aligned}$$

Donc (la notation $[x]_{97}$ est remplacée par $[x]$ car il n'y a pas d'ambiguïté sur le module) :

$$[x] - [x'] = [9] \cdot [10]^k ([c_{k+1}] - [c_k])$$

Q. 69. Montrer que toutes les erreurs consistant à modifier un seul chiffre sont détectées.

Supposons que l'erreur ne soit pas détectée ; alors par définition des chiffres de contrôle MOD 97-10, les restes de x et de x' dans la division par 97 chiffres valent 1, donc $[x] = [x']$ et

$$[0] = [9] \cdot [10]^k ([c_{k+1}] - [c_k])$$

Or 97 est un nombre premier et donc $[9]$ et $[10]$ sont inversibles dans $\mathbb{Z}/97\mathbb{Z}$. Cette dernière équation implique donc que

$$[c_{k+1}] = [c_k]$$

et donc $c_k = c_{k+1}$. Donc la seule façon pour que l'interversion ne soit pas détectée est que les chiffres intervertis soient égaux au départ. Dans un tel cas il n'y pas vraiment d'interversion. Donc l'interversion est détectée.

Définition 8.5 *Pour tout entier $m \geq 1$ on appelle* Indicatrice d'Euler *$\varphi(m)$ le nombre d'éléments inversibles de $(\mathbb{Z}/m\mathbb{Z}, \cdot)$; $\varphi(m)$ est donc égal au nombre de nombres entiers n positifs qui sont inférieurs à m et premiers avec m.*

D'après le Théorème 8.6, si p est un nombre premier, $\varphi(p) = p - 1$.

Par exemple

- $\varphi(8) = 4$ car les entiers n positifs inférieurs à 8 et premiers avec 8 sont $1, 3, 5$ et 7 ;
- $\varphi(5) = 4$ car 5 est un nombre premier.

Q. 70. Combien vaut $\varphi(257)$?

8.4 Calcul de l'Inverse

La preuve du Théorème 8.5 montre que, pour calculer l'inverse d'un élément inversible $[a]_m$ de $\mathbb{Z}/m\mathbb{Z}$, il suffit de connaître les coefficients de l'inégalité de Bézout appliquée à a et m. La preuve du Théorème 8.4 nous donne une méthode récursive pour calculer ces coefficients (et en même temps pour déterminer si $[a]_m$ est inversible). Nous avons donc obtenu un algorithme du calcul de l'inverse – voir la fonction BEZOUT dans l'Algorithme 1.

Algorithm 1 La fonction BEZOUT détermine si $[a]_m$ est inversible, et si oui, donne la forme réduite de l'inverse $([a]_m)^{-1}$.

```
1: function BEZOUT(a, m)                          ▷ a et m entiers avec m ≥ 2
2:     (u, v, d) ← EUCLIDE(a, m)
3:     if d ≠ 1 then
4:         [a]_m n'est pas inversible
5:     else
6:         calculer le reste r de u dans la division euclidienne par m
7:         ([a]_m)^{-1} ← [r]_m
8:     end if
9: end function
```

Algorithm 2 La fonction EUCLIDE calcule les coefficients de l'identité de Bézout. C'est une version récursive de l'algorithme connu sous le nom d'*Algorithme d'Euclide*.

```
1: function EUCLIDE(a, b)                         ▷ a et b entiers ≥ 0
2:             ▷ Retourne les coefficients u, v de l'identité de Bézout
3:                                          ▷ et le PGCD d de a et b.
4:     if a < b then                                  ▷ Echanger a et b
5:         (u_1, v_1, d_1) ← EUCLIDE(b, a)
6:         u ← v_1, v ← u_1, d ← d_1
7:     else
8:         if b = 0 then                              ▷ PGCD est a
9:             u ← 1, v ← 0, d ← a
10:        else
11:            Effectuer division euclidienne de a par b
12:            (q, r) ← quotient et reste
13:            (u_1, v_1, d_1) ← EUCLIDE(b, r)
14:            u ← v_1, v ← u_1 − q v_1, d ← d_1          ▷ Eq.(8.6)
15:        end if
16:    end if
17: end function
```

Appliquons la fonction BEZOUT au calcul de l'inverse de $[21]_{122}$, s'il existe.

$$
\begin{array}{cc}
\text{BEZOUT}(21,122) & [21]_{122}^{-1} = [93]_{122} \\
\downarrow & \uparrow \\
\text{EUCLIDE}(21,122) & u=-29, v=5 \\
\downarrow & \uparrow \\
\text{EUCLIDE}(122,21) & \\
q=5, r=17 & u=5, v=-29 \\
\downarrow & \uparrow \\
\text{EUCLIDE}(21,17) & \\
q=1, r=4 & u=-4, v=5 \\
\downarrow & \uparrow \\
\text{EUCLIDE}(17,4) & \\
q=4, r=1 & u=1, v=-4 \\
\downarrow & \uparrow \\
\text{EUCLIDE}(4,1) & \\
q=4, r=0 & u=0, v=1 \\
\downarrow & \uparrow \\
\text{EUCLIDE}(1,0) & \\
\multicolumn{2}{c}{d=1 \to u=1, v=0}
\end{array}
$$

La fonction BEZOUT calcule le PGCD et les coefficients u, v de 'identité de Bézout appliquée à $a = 21, b = 122$, en appelant EUCLIDE$(21, 122)$, ce qui déclenche une successions d'appels récursifs à la fonction EUCLIDE. Nous trouvons que le PGCD est $d = 1$, donc l'inverse existe, et l'identité de Bézout obtenue est

$$-29 \times 21 + 5 \times 122 = 1$$

donc $[21]_{122}^{-1} = [-29]_{122}$. Sous forme réduite, nous trouvons $[21]_{122}^{-1} = [-29]_{122} = [93]_{122}$.

Q. 71. $[93]_{122}$ est-il inversible, et si oui quel est son inverse ?

Q. 72. $[143]_{122}$ est-il inversible, et si oui quel est son inverse ?

Q. 73. Résoudre dans $\mathbb{Z}/122\mathbb{Z}$ l'équation $x \cdot [93]_{122} = [40]_{122}$.

9
Eléments d'Algèbre Abstraite

L'algèbre dite *abstraite* s'intéresse aux structures d'anneau, groupe etc, c'est à dire aux propriétés d'opérations qui restent vraies dans des contextes différents. Nous allons dans ce chapitre étudier des éléments de la théorie des groupes commutatifs finis, qui nous seront utiles pour comprendre la cryptographie et, en Partie III, les codes correcteurs d'erreur.

Je préfère la nommer ainsi [algèbre abstraite] plutôt qu'algèbre moderne, parce qu'elle vivra sans doute longtemps et finira donc par devenir l'algèbre ancienne.
Francesco Severi, Liège 1949, cité par Serge Lang dans son livre *Algebra*, Addison-Wesley, 1965.

9.1 Groupes Commutatifs

Définition 9.1 *Soit $(G, \star)$ un ensemble G muni d'une opération binaire $\star$, c'est à dire d'un mécanisme qui associe à deux éléments a et b de G, distincts ou non, un élément de G noté $a \star b$.*

$(G, \star)$ est appelé groupe commutatif, ou groupe abélien s'il possède les propriétés suivantes :

(Associativité) $a \star (b \star c) = (a \star b) \star c)$ pour tous $a, b, c \in G$;

(Neutre) Il existe un élément $e \in G$ tel que $a \star e = e \star a = a$ pour tout $a \in G$;

(Symétrique) Pour tout élément $a \in G$ il existe un élément $a' \in G$ tel que $a \star a' = e$; a' est appelé élément symétrique de a.

(Commutativité) $a \star b = b \star a$ pour tous $a, b \in G$;

Une opération binaire définie dans G est donc une application de $G \times G$ vers G.

Notons que l'élément neutre e est forcément unique ; de même, le symétrique a' de a est unique, pour chaque a.

Si $(G, \star)$ possède les trois premières propriétés, mais pas la commutativité, on dit que G est un *groupe non commutatif*. Dans ce chapitre nous n'étudions que les groupes commutatifs.

Si l'opération binaire est notée $+$, on note habituellement l'élément neutre 0 ; l'élément symétrique de a est appelé *opposé* de a et est noté $-a$.

Si l'opération binaire est notée $\cdot$, on note habituellement l'élément neutre 1 ; l'élément symétrique de a est appelé *inverse* de a et est noté a^{-1}.

Exemple 9.1 (Le groupe $(\mathbb{Z}/m\mathbb{Z}, +)$) *Nous avons vu en Section 8.2 que $\mathbb{Z}/m\mathbb{Z}$ muni de l'addition modulaire possède les 4 propriétés qui en font un groupe commutatif. L'élément neutre est $[0]_m$ et le symétrique de $[a]_m$ est $-[a]_m = [-a]_m$.*

Exemple 9.2 (Un non-groupe) *Par contre $(\mathbb{Z}/m\mathbb{Z}, \cdot)$ pour m entier ≥ 2 n'est pas un groupe commutatif ; les propriétés d'associativité, existence d'un élément neutre et commutativité sont vraies (avec $e = [1]_m$). Cependant, la propriété d'existence d'un élément symétrique (i.e. inverse) n'est pas vrai : il existe au moins un élément, $[0]_m$, qui ne possède pas d'inverse, car on ne peut jamais avoir $[0]_m a' = [1]_m$.*

Théorème 9.1 *Soit $\mathbb{Z}/m\mathbb{Z}^*$ l'ensemble des éléments inversibles de $\mathbb{Z}/m\mathbb{Z}$, pour $m \geq 2$. $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$ est un groupe commutatif.*

☺*Preuve :* Il faut d'abord démontrer que la multiplication est bien une opération binaire dans $\mathbb{Z}/m\mathbb{Z}^*$, c'est à dire que si $a \in \mathbb{Z}/m\mathbb{Z}^*$ et $b \in \mathbb{Z}/m\mathbb{Z}^*$ alors $ab \in \mathbb{Z}/m\mathbb{Z}^*$. Si a et b sont inversibles, alors $(ab)(a^{-1}b^{-1}) = [1]_m$ donc ab est inversible (et son inverse est $a^{-1}b^{-1}$).

D'autre part, $[1]_m$ est inversible donc $[1]_m \in \mathbb{Z}/m\mathbb{Z}^*$ et donc la multiplication dans $\mathbb{Z}/m\mathbb{Z}^*$ possède un élément neutre. L'existence d'un élément symétrique est évidente par définition $\mathbb{Z}/m\mathbb{Z}^*$. Les autres propriétés (associativité, commutativité) ont déjà été démontrées en Section 8.2. ☺□

Théorème et Définition 9.2 *Soient $(G, \star)$ et $(H, \star)$ deux ensembles munis chacun d'une opération binaire. L'opération produit est l'opération binaire $\star$ définie sur l'ensemble $G \times H$ par*

$$(a, b) \star (a', b') = (a \star a', b \star b')$$

Si e est élément neutre dans G et f dans H alors (e, f) est élément neutre pour l'opération produit.

Si a admet a' comme élément symétrique dans G et si b admet b' comme élément symétrique dans H alors (a, b) admet (a', b') comme élément symétrique dans $G \times H$.

Si $(G, \star)$ et $(H, \star)$ sont des groupes alors $(G \times H, \star)$ aussi. On l'appelle le groupe produit.

Exemple 9.3 *Le groupe produit de $(\mathbb{Z}/m_1\mathbb{Z}, +)$ et $(\mathbb{Z}/m_2\mathbb{Z}, +)$ possède m_1m_2 éléments. L'élément neutre est $([0]_{m_1}, [0]_{m_2})$. L'élément symétrique de $([a]_{m_1}, [b]_{m_2})$ est $([-a]_{m_1}, [-b]_{m_2})$.*

L'opération produit est appelée l' addition *dans $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$, c'est à dire que nous écrivons :*

$$\begin{aligned} ([j]_{m_1}, [k]_{m_2}) + ([j']_{m_1}, [k']_{m_2}) &\stackrel{\text{def}}{=} ([j]_{m_1} + [j']_{m_1}, [k]_{m_2} + [k']_{m_2}) \\ &= ([j + j']_{m_1}, [k + k']_{m_2}) \end{aligned} \quad (9.1)$$

Exemple 9.4 *L'opération produit de $(\mathbb{Z}/m_1\mathbb{Z}, \cdot)$ et $(\mathbb{Z}/m_2\mathbb{Z}, \cdot)$ possède un élément neutre $([1]_{m_1}, [1]_{m_2})$ mais cela n'en fait pas un groupe car certains éléments n'ont pas d'élément symétrique (ici appelé inverse). Par exemple $([0]_{m_1}, [b]_{m_2})$ n'a pas d'inverse, quel que soit b.*

L'élément $([a]_{m_1}, [b]_{m_2})$ est inversible si et seulement si $[a]_{m_1}$ et $[b]_{m_2}$ sont inversibles, c'est à dire si et seulement si a est premier avec m_1 et b est premier avec m_2. L'ensemble des éléments inversibles de $(\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}, \cdot)$ est donc $\mathbb{Z}/m_1\mathbb{Z}^* \times \mathbb{Z}/m_2\mathbb{Z}^*$, il constitue un groupe, qui est le groupe produit de $(\mathbb{Z}/m_1\mathbb{Z}^*, \cdot)$ et $(\mathbb{Z}/m_2\mathbb{Z}^*, \cdot)$.

L'opération produit est appelée la multiplication *dans $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$ et est aussi notée $\cdot$, c'est à dire que nous écrivons :*

$$\begin{aligned} ([j]_{m_1}, [k]_{m_2}) \cdot ([j']_{m_1}, [k']_{m_2}) &\stackrel{\text{def}}{=} ([j]_{m_1} \cdot [j']_{m_1}, [k]_{m_2} \cdot [k']_{m_2}) \\ &= ([j \cdot j']_{m_1}, [k \cdot k']_{m_2}) \end{aligned} \quad (9.2)$$

Q. 74. Voici deux groupes commutatifs avec leurs tables (nous omettons les crochets) :

$(\mathbb{Z}/2\mathbb{Z}, +)$

$+$	0	1
0	0	1
1	1	0

$(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$

$\cdot$	1	3
1	1	3
3	3	1

Dans chaque cas, quel est l'élément neutre ?

$G \times H$ est le *produit cartésien* de G et H, c'est à dire l'ensemble des couples (x, y) avec $x \in G$ et $y \in H$.

Le signe $\times$, utilisé entre deux ensembles, signifie le *produit cartésien* ; utilisé entre deux nombres, il signifie la multiplication.

Il y a ici trois opérations binaires (une dans G, une dans H et une dans le produit cartésien $G \times H$) ; elles sont toutes les trois notées de la même façon ($\star$), mais elles sont différentes. C'est un abus de notation qui permet d'éviter une certaine lourdeur.

$+$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

TABLE 9.1: Table du groupe produit $((\mathbb{Z}/2\mathbb{Z})^2, +)$ qui possède 4 éléments. L'élément neutre est 00. Nous reconnaissons l'opération `xor` bit par bit sur les suites de 2 bits.

Q. 75. Quel est l'élément symétrique de chaque élément du groupe produit $(\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}, +)$?

Q. 76. Combien y a-t-il d'éléments dans les groupes produits $(\mathbb{Z}/5\mathbb{Z} \times \mathbb{Z}/7\mathbb{Z}, +)$ et $(\mathbb{Z}/5\mathbb{Z}^* \times \mathbb{Z}/7\mathbb{Z}^*, \cdot)$?

9.2 *Isomorphisme*

Des groupes apparemment différents peuvent être les mêmes, si nous acceptons de changer les noms des éléments ou de l'opération. Par exemple, les groupes $(\mathbb{Z}/2\mathbb{Z}, +)$ et $(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$ ont des tables identiques, si nous faisons la correspondance $0 \mapsto 1$ et $1 \mapsto 3$. Bien que dans le premier cas l'opération binaire soit appelée addition et dans le deuxième multiplication, ce sont en fait les mêmes, après renommage. Cette idée, appelée isomorphisme, est utilisée en cryptographie et dans bien d'autres domaines.

Définition 9.3 (Isomorphisme) *Soient $(G, \star)$ et $(H, \otimes)$ deux ensembles munis chacun d'une opération binaire. Un isomorphisme de $(G, \star)$ vers $(H, \otimes)$ est une application $\psi : G \to H$ telle que*
- *ψ est bijective ;*
- *$\psi(a \star b) = \psi(a) \otimes \psi(b)$ pour tous $a, b \in G$.*

On dit que $(G, \star)$ et $(H, \otimes)$ sont isomorphes s'il existe un isomorphisme de $(G, \star)$ vers $(H, \otimes)$.

Ainsi, l'application

$$\begin{array}{ccc} \mathbb{Z}/2\mathbb{Z} & \to & \mathbb{Z}/4\mathbb{Z}^* \\ 0 & \mapsto & 1 \\ 1 & \mapsto & 3 \end{array}$$

est un isomorphisme de $(\mathbb{Z}/2\mathbb{Z}, +)$ vers $(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$.

Supposons que ψ est un isomorphisme de $(G, \star)$ vers $(H, \otimes)$ alors les opérations $\star$ et $\otimes$ sont les mêmes pour quiconque connaît la correspondance ψ : donc les propriétés de ces deux opérations doivent se correspondre une par une. En particulier :

1. Si G et H sont finis, ils ont le même cardinal ;
2. Si $(G, \star)$ est un groupe alors $(H, \otimes)$ aussi, et réciproquement ; dans un tel cas, ψ transforme l'élément neutre de $(G, \star)$ en l'élément neutre de $(H, \otimes)$, et l'élément symétrique de a dans $(G, \star)$ en l'élément symétrique de $\psi(a)$ dans $(H, \otimes)$;
3. L'application réciproque $\psi^{-1} : (H, \otimes) \to (G, \star)$ est aussi un isomorphisme.

Soient G, H, K des sous-ensembles de $\mathbb{Z}$, et $\star, \oplus, \otimes$ trois opérations binaires définies sur G, H et K.

(a) $(G, \star)$ est isomorphe à $(G, \star)$ puisque l'identité est évidemment un isomorphisme.

(b) Si $(G, \star)$ et $(H, \otimes)$ sont isomorphes alors $(H, \otimes)$ et $(G, \star)$.

(c) Si $(G, \star)$ et $(H, \otimes)$ sont isomorphes, et si $(H, \otimes)$ et $(K, \oplus)$ sont isomorphes, alors $(G, \star)$ et $(K, \oplus)$ aussi.

La relation "être isomorphe" est donc une relation d'équivalence sur l'ensemble des sous-ensembles de $\mathbb{Z}$ munis d'une opération binaire.

Exemple 9.5 *Les groupes $(\mathbb{Z}/4\mathbb{Z}, +)$ et $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$ sont-ils isomorphes ?*

Voyons d'abord s'ils ont le même cardinal. Les éléments inversibles de $\mathbb{Z}/5\mathbb{Z}$ sont $[1]_5, [2]_5, [3]_5$ et $[4]_5$ donc $\mathbb{Z}/5\mathbb{Z}^*$ a 4 éléments, comme $\mathbb{Z}/4\mathbb{Z}$. Comparons maintenant les tables de ces deux groupes : (en omettant les crochets) :

$(\mathbb{Z}/4\mathbb{Z}, +)$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Ré-écrivons la table de $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$ en changeant l'ordre de 3 et 4 :

$(\mathbb{Z}/4\mathbb{Z}, +)$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$	1	2	4	3
1	1	2	4	3
2	2	4	3	1
4	4	3	1	2
3	3	1	2	4

puis ré-écrivons encore une fois la table de $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$ en changeant les symboles : $1 \mapsto O$, $2 \mapsto I$, $4 \mapsto II$ et $3 \mapsto III$.

$(\mathbb{Z}/4\mathbb{Z}, +)$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$	O	I	II	III
O	O	I	II	III
I	I	II	III	O
II	II	III	O	I
III	III	O	I	II

Les deux sont maintenant les mêmes, l'une en chiffres arabes, l'autre en chiffres romains. Donc les deux groupes sont isomorphes, avec comme isomorphisme celui qui fait correspondre $0 \mapsto O$, $1 \mapsto I$, $2 \mapsto II$ et $3 \mapsto III$. Rétablissons les symboles d'origine de $\mathbb{Z}/5\mathbb{Z}^*$, nous avons donc obtenu comme isomorphisme :

$$\begin{aligned}&\psi : (\mathbb{Z}/4\mathbb{Z}, +) \to (\mathbb{Z}/5\mathbb{Z}^*.\cdot)\\ &[0]_4 \mapsto [1]_5\,, [1]_4 \mapsto [2]_5\,, [2]_4 \mapsto [4]_5\,, [3]_4 \mapsto [3]_5\end{aligned}$$

ψ est bien bijectif et $\psi(a+b) = \psi(a) \cdot \psi(b)$ L'élément neutre de $(\mathbb{Z}/4\mathbb{Z}, +)$ correspond à l'élément neutre de $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$:

$$\psi([0]_4) = [1]_5$$

et ψ transforme l'opposé en inverse, par exemple $\psi(-[3]_4) = \psi([3]_4)^{-1}$ comme nous pouvons le vérifier :

$$\begin{aligned}&\psi(-[3]_4) = \psi([1]_4) = [2]_5\\ &\psi([3]_4)^{-1} = [3]_5^{-1} = [2]_5\end{aligned}$$

Exemple 9.6 *Les éléments inversibles de $\mathbb{Z}/8\mathbb{Z}$ sont $[1]_8, [3]_8, [5]_8$ et $[7]_8$ donc $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ est un groupe à 4 éléments. Est-il isomorphe à $(\mathbb{Z}/4\mathbb{Z}, +)$?*

La table de multiplication de $\mathbb{Z}/8\mathbb{Z}^*$ est donnée ci-contre. Nous voyons que chaque élément b de $\mathbb{Z}/8\mathbb{Z}^*$ vérifie $b \cdot b = [1]_8$; donc s'il existait un isomorphisme entre $(\mathbb{Z}/4\mathbb{Z}, +)$ et $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ on devrait avoir la propriété correspondante dans $(\mathbb{Z}/4\mathbb{Z}, +)$, c'est à dire qu'on devrait avoir $a + a = [0]_4$ pour tout $a \in \mathbb{Z}/4\mathbb{Z}$, ce qui n'est pas vrai (prendre par exemple $a = [1]_4$. Donc les deux groupes ne sont pas isomorphes, bien qu'ils aient le même nombre d'éléments.

$(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$	1	3	5	7
1	1	3	5	7
3	3	1	7	5
5	5	7	1	3
7	7	5	3	1

TABLE 9.2: Table du groupe des éléments inversibles de $\mathbb{Z}/8\mathbb{Z}$.

Exemple 9.7 *Le groupe produit de $(\mathbb{Z}/2\mathbb{Z}, +)$ et $(\mathbb{Z}/2\mathbb{Z}, +)$ est-il isomorphe à $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$? En comparant les tables 9.1 et 9.2, nous voyons que l'application*

$$\begin{aligned}&\psi : \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \to \mathbb{Z}/8\mathbb{Z}^*\\ &00 \mapsto 1, 01 \mapsto 3, 10 \mapsto 5, 11 \mapsto 7\end{aligned}$$

est un isomorphisme.

9.3 Période d'un Elément

Théorème et Définition 9.4 (Période) *Soit $(G, \star)$ un groupe commutatif fini dont l'élément neutre est noté e.*

Q. 77. Soit f l'application

$$\begin{aligned}&f : (\mathbb{Z}/5\mathbb{Z}^*.\cdot) \to (\mathbb{Z}/4\mathbb{Z}, +)\\ &[1]_5 \mapsto [0]_4\,, [2]_5 \mapsto [1]_4\,,\\ &[3]_5 \mapsto [3]_4\,, [4]_5 \mapsto [2]_4\end{aligned}$$

Montrer que $f(a \cdot b) = f(a) + f(b)$.

On dit parfois qu'une application telle que ψ est une "exponentielle discrète" à cause de la relation $\psi(a+b) = \psi(a) \cdot \psi(b)$.

Q. 78. Parmi les groupes suivants, dire lesquels sont isomorphes :

1. $(\mathbb{Z}/4\mathbb{Z}, +)$
2. $(\mathbb{Z}/5\mathbb{Z}, +)$
3. $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$
4. $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$
5. le groupe produit de $(\mathbb{Z}/2\mathbb{Z}, +)$ et $(\mathbb{Z}/2\mathbb{Z}, +)$

1. *Pour tout élément $a \in G$ il existe un entier $k \geq 1$ tel que $\overbrace{a \star a \star ... \star a}^{k \text{ fois}} = e$. Le plus petit de ces entiers est appelé la période de a.*
2. *Pour tout entier positif ℓ, $\overbrace{a \star a \star ... \star a}^{\ell \text{ fois}} = e$ si et seulement si la période de a divise ℓ.*

Notons que la période de l'élément neutre est 1.

Preuve :

1. Soit a^ℓ la suite définie pour $\ell = 1, 2, ...$ par $a^\ell = \overbrace{a \star a \star ... \star a}^{\ell \text{ fois}}$. Comme G est un ensemble fini, cette suite ne peut prendre qu'un nombre fini de valeurs distinctes, c'est à dire qu'il existe $\ell_1 \geq 1$ et $\ell_2 > \ell_1$ tels que $a^{\ell_2} = a^{\ell_1}$, donc

$$\overbrace{a \star a \star ... \star a}^{\ell_2 - \ell_1 \text{ fois}} \star \overbrace{a \star a \star ... \star a}^{\ell_1 \text{ fois}} = \overbrace{a \star a \star ... \star a}^{\ell_1 \text{ fois}}$$

   Soit a' l'élément symmétrique de a ; nous avons donc :

$$\overbrace{a \star a \star ... \star a}^{\ell_2 - \ell_1 \text{ fois}} \star \overbrace{a \star a \star ... \star a}^{\ell_1 \text{ fois}} \star \overbrace{a' \star a' \star ... \star a'}^{\ell_1 \text{ fois}} = \overbrace{a \star a \star ... \star a}^{\ell_1 \text{ fois}} \star \overbrace{a' \star a' \star ... \star a'}^{\ell_1 \text{ fois}}$$

   donc $\overbrace{a \star a \star ... \star a}^{\ell_2 - \ell_1 \text{ fois}} = e$ donc il existe un entier $k = \ell_2 - \ell_1 \geq 1$ tel que $a^k = e$. Comme tout ensemble d'entiers positifs possède un plus petit élément, la période est bien définie.
2. ($\Leftarrow$) Notons tout d'abord que si ℓ et ℓ' sont des entiers positifs

$$\overbrace{a \star a \star ... \star a}^{\ell \text{ fois}} \star \overbrace{a \star a \star ... \star a}^{\ell' \text{ fois}} = \overbrace{a \star a \star ... \star a}^{\ell + \ell' \text{ fois}}$$

   c'est à dire que nous pouvons écrire comme d'habitude $a^{\ell + \ell'} = a^\ell \star a^{\ell'}$. De la même façon,

$$a^{u\ell} = \overbrace{a^u \star a^u \star ... \star a^u}^{\ell \text{ fois}} = (a^u)^\ell$$

   Soit maintenant ℓ tel que $a^\ell = e$ et soit k la période de a. Nous faisons un raisonnement par l'absurde. Supposons que k ne divise pas ℓ. Soit $\ell = kq + r$ la division euclidienne de ℓ par k ; nous avons $1 \leq r \leq k - 1$. Donc

$$e = a^\ell = (a^k)^q \star a^r = e^q \star a^r = a^r$$

   Donc r satisfait $a^r = e$, ce qui est impossible car $r < k$.
   ($\Rightarrow$) Soit $\ell = uk$ avec u entier. Donc $a^\ell = \left(a^k\right)^u = e^u = e$.

□

Exemple 9.8 ($(\mathbb{Z}/12\mathbb{Z}, +)$) *Dans le groupe $(\mathbb{Z}/m\mathbb{Z}, +)$ la période de $[a]_m$ est le plus petit entier $k \geq 1$ tel que $k[a]_m = [0]_m$, c'est à dire que c'est le plus petit entier $k \geq 1$ tel que*

$$ka \equiv 0 \pmod{m}$$

a	période
0	1
1	12
2	6
3	4
4	3
5	12
6	2
7	12
8	3
9	4
10	6
11	12

TABLE 9.3: Périodes des éléments de $(\mathbb{Z}/12\mathbb{Z}, +)$ (les crochets sont omis, ainsi $a = 11$ signifie $a = [11]_{12}$).

Prenons $m = 12$ et $a = 2$ ou 5 :

k	2	3	4	5	6	7	8	9	10	11	12
$k[2]_{12}$	4	6	8	10	0	...					
$k[5]_{12}$	10	3	8	1	6	11	4	9	2	7	0

Donc la période de $[2]_{12}$ est 6, et la période de $[5]_{12}$ est 12. En répétant ces calculs pour les 12 éléments de $\mathbb{Z}/12\mathbb{Z}$ nous obtenons la Table 9.4

Les entiers $\ell \geq 1$ tels que $\ell[2]_{12} = [0]_{12}$, c'est à dire tels que $2\ell \equiv 0 \pmod{12}$ sont les multiples de la période $k = 6$.

Exemple 9.9 ($(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$) *Dans le groupe $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$ la période de $[a]_m$ est le plus petit entier $k \geq 1$ tel que $([a]_m)^k = [1]_m$, c'est à dire que c'est le plus petit entier $k \geq 1$ tel que*

$$a^k \equiv 1 \pmod{m}$$

Il est défini si $[a]_m \in \mathbb{Z}/m\mathbb{Z}^*$, c'est à dire si a et m sont premiers entre eux.

Prenons $m = 8$, le groupe $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ comporte 4 éléments, $[1]_8, [3]_8, [5]_8$ et $[7]_8$. La période de $[1]_8$ est 1 (c'est l'élément neutre). Calculons les périodes des autres éléments : nous avons $[3]_8^2 = [5]_8^2 = [7]_8^2 = [1]_8$ donc tous les éléments sauf $[1]_8$ ont pour période 2.

a	période
1	1
3	2
5	2
7	2

TABLE 9.4: Périodes des éléments de $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ (les crochets sont omis).

Q. 79. Quelle est la période de $[1]_m$ dans $(\mathbb{Z}/m\mathbb{Z}, +)$? dans $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$?

Si deux groupes sont isomorphes, ils ont les mêmes tables après renommage, donc les périodes des éléments sont les mêmes. Nous avons donc le théorème suivant, que nous citons sans démonstration :

Théorème 9.2 *Soient $(G, \star)$ et $(H, \otimes)$ deux groupes isomorphes et ψ un isomorphisme $G \to H$. Pour tout $x \in G$, la période de x est égale à la période de $\psi(x)$.*

Exemple 9.10 (Suite de l'Exemple 9.6) *Les périodes des 4 éléments de $(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ sont $1, 2, 2$ et 2 alors que les périodes des 4 éléments de $(\mathbb{Z}/4\mathbb{Z}, +)$ sont $1, 2, 4$ et 4 donc ces deux groupes ne peuvent pas être isomorphes.*

Dans chacun des exemples précédents, la période d'un élément est un diviseur du nombre d'éléments du groupe. Ceci est vrai en général, comme l'exprime le théorème le théorème suivant, qui est une version particulière du théorème de Lagrange :

Théorème 9.3 (Lagrange) *Soit $(G, \star)$ un groupe commutatif fini, de cardinal n. La période de tout élément de G divise n.*

En particulier, notons e l'élément neutre de G. Pour tout $a \in G$:

$$\overbrace{a \star a \star \ldots \star a}^{n \text{ fois}} = e$$

Preuve : Soit $a \in G$ fixé, de période k.
(1) Introduisons la relation $\mathcal{R}$ sur G définie par

$$x\mathcal{R}y \Leftrightarrow \exists \ell \in \{0,1,2,...,k-1\} : y = a^\ell \star x$$

avec la convention $a^\ell = \overbrace{a \star a \star ... \star a}^{\ell \text{ fois}}$ et $a^0 = e$. Montrons que $\mathcal{R}$ est une relation d'équivalence.

- (Réflexive) Pour tout $x \in G$: $x\mathcal{R}x$ (prendre $\ell = 0$) ;
- (Symmétrique) Supposons que $x\mathcal{R}y$; alors il existe $\ell \in \{0,1,2,..k-1\}$ tel que $y = a^\ell \star x$. Si $\ell = 0$ alors $x = y$ donc $y\mathcal{R}x$. Sinon, $k \in \{1,2,..k-1\}$. Nous avons

  $$a^{k-\ell} \star y = a^{k-\ell} \star a^\ell \star x = a^k \star x = e \star x = x$$

  avec $(k-\ell) \in \{1,2,...,k-1\}$ donc $y\mathcal{R}x$;
- (Transitive) Supposons que $x\mathcal{R}y$ et $y\mathcal{R}z$, et soient $\ell, \ell' \in \{0,1,2,...,k-1\}$ tel que $y = a^\ell \star x$ et $z = a^{\ell'} \star y$. Donc $z = a^{\ell+\ell'} \star x$. Soit $\ell + \ell' = qk + r$ la division euclidienne de $\ell + \ell'$ par k. Donc $z = a^{qk+r} = \left(a^k\right)^q \star a^r = a^r$ avec $r \in \{1,2,..k-1\}$, et donc $x\mathcal{R}z$.

(2) Soit $x \in G$; la classe d'équivalence de x est (par définition de $\mathcal{R}$) :

$$\left\{x, a \star x, a^2 \star x, ..., a^{k-1} \star x\right\}$$

Tous ces éléments sont distincts ; en effet, sinon on aurait $a^{\ell_1} \star x = a^{\ell_2} \star x$ avec $0 \leq \ell_1 < \ell_2$ donc $a^{\ell_1} = a^{\ell_2}$ et donc, comme dans la preuve du Théorème 9.4 : $a^{\ell_2-\ell_1} = e$ ce qui est impossible car $1 \leq \ell_2 - \ell_1 < k$. Donc la classe d'équivalence de x comporte k éléments.
(3) Toutes les classes d'équivalence comportent k éléments chacune. Or, l'ensemble des classes d'équivalences constitue une partition de G (c'est la propriété des relations d'équivalence). Soit u le nombre de classes d'équivalence. Nous avons donc $n = uk$, c'est à dire que k divise n. □

Si $G = (\mathbb{Z}/m\mathbb{Z}, +)$, le Théorème 9.3 ne nous apprend rien de nouveau. En effet, le cardinal du groupe est m et le théorème dit que pour tout $a \in \mathbb{Z}$, nous avons $m[a]_m = [0]_m$, ce qui est évident car nous savons que $m[a]_m = [m]_m[a]_m = [0]_m[a]_m = [0]_m$.

Par contre, pour $G = (\mathbb{Z}/m\mathbb{Z}^*, \cdot)$, le résultat est non évident (et forme un élément important de la méthode de chiffrage asymétrique du Chapitre 10). Le cardinal de $\mathbb{Z}/m\mathbb{Z}^*$ est $\varphi(m)$ (indicatrice d'Euler) ; nous avons donc le résultat suivant :

Corollaire 9.4 (Théorème d'Euler) *Pour tout entier positif m et tout entier a premier avec m :*

$$([a]_m)^{\varphi(m)} = [1]_m$$

ou encore

$$a^{\varphi(m)} \equiv 1 \pmod{m}$$

Q. 80. Vérifiez le théorème d'Euler avec $a = 7$ et $m = 10$.

Dans le cas particulier où le module est un nombre premier, nous obtenons le résultat suivant :

Corollaire 9.5 (Théorème de Fermat) *Si p est un nombre premier, pour tout entier a :*

$$([a]_p)^p = [a]_p$$

ou encore

$$a^p \equiv a \pmod{p}$$

Q. 81. Vérifiez le théorème de Fermat avec $a = 10$ et $p = 7$.

☺*Preuve :* Nous avons vu que $\varphi(p) = p - 1$ et tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $[0]_p$ sont inversibles.

(1) Si $x \in \mathbb{Z}/p\mathbb{Z}$ et $x \neq [0]_p$, d'après le Théorème d'Euler : $x^{p-1} = [1]_p$. En multipliant les deux côtés par x :

$$x^p = x \tag{9.3}$$

(2) Si $x = [0]_p$, l'égalité 9.3 est aussi vraie, donc elle est vraie pour tout $a \in \mathbb{Z}/p\mathbb{Z}$. ☺□

10
Cryptographie Asymétrique

Nous sommes maintenant presque arrivés au bout des théories qui nous permettent de comprendre comment fonctionne la cryptographie asymétrique. Il nous reste à découvrir le théorème des restes chinois, que nous utiliserons pour développer un système cryptographique.

10.1 Le Théorème des Restes Chinois

Commençons par un petit jeu. Nous voulons remplir une boîte $m_1 \times m_2$, comme suit. Nous commençons à la position $(0,0)$, et nous y inscrivons 0. Puis nous continuons en suivant la diagonale et inscrivons 1 dans la case $(1,1)$. Si nous sortons de la boîte, nous y revenons par le bord opposé et continuons parallèlement à la diagonale (voir Figure 10.1). La question du jeu est : pourrons nous remplir la boîte ?

Le théorème des restes chinois donne la solution : si m_1 et m_2 sont premiers entre eux, alors la réponse est oui, et nous pouvons remplir la boîte en exactement $m_1 m_2$ étapes, c'est à dire en utilisant les nombres de 0 à $(m_1 m_2 - 1)$. Par contre, si m_1 et m_2 ne sont pas premiers entre eux, il restera des cases vides, quel que soit le nombre d'étapes.

	0	1	2
0	0	4	8
1	9	1	5
2	6	10	2
3	3	7	11

$m_1 = 3, m_2 = 4$

	0	1
0	0, 4	
1		1, 5
2	2, 6	
3		3, 7

$m_1 = 2, m_2 = 4$

FIGURE 10.1: Le jeu des restes chinois.

Pour comprendre le lien avec l'arithmétique modulaire, imaginons que les coordonnées des cases sont numérotées en utilisant $\mathbb{Z}/m_1\mathbb{Z}$ et $\mathbb{Z}/m_2\mathbb{Z}$, comme dans la Figure 10.1. Le jeu consiste en fait à placer le nombre entier k dans la case dont les coordonnées sont $[k]_{m_1}$ et $[k]_{m_2}$. La question du jeu est donc : l'application

$$\begin{cases} \mathbb{N} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \\ k \mapsto ([k]_{m_1}, [k]_{m_2}) \end{cases} \tag{10.1}$$

est-elle surjective ?

Observons que si $k' = k + nm_1m_2$ avec n entier, alors $([k']_{m_1}, [k']_{m_2}) = ([k]_{m_1}, [k]_{m_2})$. En d'autres termes, deux entiers k et k' qui sont congrus modulo m_1m_2 sont placés dans la même case de la boîte. Nous pouvons donc arrêter le jeu quand $k = m_1m_2 - 1$. D'autre part, au lieu de l'application définie en (10.1), nous pouvons tout

aussi bien considérer que le jeu est une réalisation de l'application :

$$\psi : \begin{cases} \mathbb{Z}/m_1m_2\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \\ [k]_{m_1m_2} \mapsto ([k]_{m_1}, [k]_{m_2}) \end{cases} \tag{10.2}$$

puisque la case dans laquelle est placé k ne dépend que de $[k]_{m_1m_2}$. La question du jeu devient donc : l'application ψ est-elle surjective ?

Théorème 10.1 (Restes Chinois) *Soient m_1 et m_2 deux entiers ≥ 2.*

(1) Si m_1 et m_2 sont premiers entre eux, l'application ψ définie par l'Eq.(10.2) est bijective.

(2) De plus c'est un isomorphisme à la fois pour l'addition et la multiplication.

(3) Si m_1 et m_2 ne sont pas premiers entre eux, l'application ψ n'est ni surjective ni injective.

☺*Preuve :* (1) Supposons que m_1 et m_2 sont premiers entre eux et montrons que ψ est *injective*. Soient k et k' des entiers tels que $\psi([k]_{m_1m_2}) = \psi([k']_{m_1m_2})$. Alors

$$\psi([k-k']_{m_1m_2}) = ([k-k']_{m_1}, [k-k']_{m_2}) = ([0]_{m_1}, [0]_{m_2})$$

donc $(k-k') \equiv 0 \pmod{m_1}$ et $(k-k') \equiv 0 \pmod{m_2}$.

Donc m_1 et m_2 divisent $(k-k')$ et sont premiers entre eux. D'après le Théorème 7.6, item 3, m_1m_2 divise $(k-k')$ donc $[k-k']_{m_1m_2} = [0]_{m_1m_2}$, donc $[k]_{m_1m_2} = [k']_{m_1m_2}$. Par contraposition, nous avons montré que si $[k]_{m_1m_2} \neq [k']_{m_1m_2}$ alors $\psi([k]_{m_1m_2}) \neq \psi([k']_{m_1m_2})$, c'est à dire que ψ est injective.

Or les ensembles $\mathbb{Z}/m_1m_2\mathbb{Z}$ et $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$ sont des ensembles finis de même cardinal (m_1m_2). Comme ψ est injective, elle est aussi bijective.

(2) Montrons que ψ est un isomorphisme pour l'addition ; il faut montrer

$$\psi([k]_{m_1m_2} + [k']_{m_1m_2}) = \psi([k]_{m_1m_2}) + \psi([k']_{m_1m_2}) \tag{10.3}$$

pour tous entiers k, k'. Le membre de gauche est $([k+k']_{m_1}, [k+k']_{m_2})$. Par définition de la loi produit (Eq.(9.1)), le membre de droite est $([k]_{m_1} + [k']_{m_1}, [k]_{m_2} + [k']_{m_2})$; les deux sont donc égaux.

La même chose est vraie pour la multiplication, c'est à dire :

$$\psi([k]_{m_1m_2} \cdot [k']_{m_1m_2}) = \psi([k]_{m_1m_2}) \cdot \psi([k']_{m_1m_2}) \tag{10.4}$$

(3) Supposons que m_1 et m_2 ne sont pas premiers entre eux. Soit $d \geq 2$ un diviseur commun et $m = \frac{m_1m_2}{d}$; m est un multiple de m_1 car $\frac{m_2}{d}$ est entier, et la même chose vaut pour m_2 ; donc $\psi([m]_{m_1m_2}) = ([0]_{m_1}, [0]_{m_2})$. Or $1 < m < m_1m_2$, donc $[m]_{m_1m_2} \neq [0]_{m_1m_2}$. Donc ψ n'est pas injective. Comme $\mathbb{Z}/m_1m_2\mathbb{Z}$ et $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$ sont des ensembles finis de même cardinal, ψ n'est pas surjective non plus. ☺□

Une conséquence immédiate concerne la fonction indicatrice d'Euler.

	$[0]_3$	$[1]_3$	$[2]_3$
$[0]_4$	$[0]_{12}$	$[4]_{12}$	$[8]_{12}$
$[1]_4$	$[9]_{12}$	$[1]_{12}$	$[5]_{12}$
$[2]_4$	$[6]_{12}$	$[10]_{12}$	$[2]_{12}$
$[3]_4$	$[3]_{12}$	$[7]_{12}$	$[11]_{12}$

$m_1 = 3, m_2 = 4$

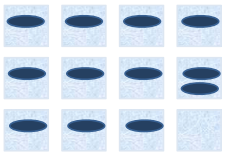
	$[0]_2$	$[1]_2$
$[0]_4$	$[0]_8, [4]_8$	
$[1]_4$		$[1]_8, [5]_8$
$[2]_4$	$[2]_8, [6]_8$	
$[3]_4$		$[3]_8, [7]_8$

$m_1 = 2, m_2 = 4$

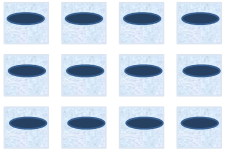
FIGURE 10.2: Le jeu des restes chinois, vu comme une application de $\mathbb{Z}/m_1m_2\mathbb{Z}$ vers $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$: $[k]_{m_1m_2}$ est placé dans la case $([k]_{m_1}, [k]_{m_2})$.

Le *principe des tiroirs*, ou *principe des boîtiers*, (Ang. *pigeon holes*), dans une de ses nombreuses variantes, établit des liens entre nombres d'éléments des ensembles et les propriétés d'injection / surjection. Soient E et F des ensembles finis *de même cardinal* et f une application $E \to F$. Ce principe dit que :

1. Si f est injective alors f est surjective (donc bijective).
2. Si f est surjective alors f est injective (donc bijective).



Ni injective ni surjective



Bijective

Par exemple, supposons que nous ayons des CDs à ranger dans des boîtiers (ou tiroirs). E est l'ensemble des CDs et F l'ensemble des boîtiers. Un rangement des CDs dans les boîtiers définit une application $f : E \to F$ car il associe à chaque CD un boîtier. Un rangement est injectif s'il n'y a jamais deux CDs dans le même boîtier. Un rangement est surjectif si tous les boîtiers sont occupés.

Le principe des boîtiers dit que si le nombres de CDs est égal au nombre de boîtiers, et s'il n'y a jamais deux CDs dans le même boîtier, alors tous les boîtiers sont occupés (et vice-versa).

Corollaire 10.2 *Si m_1 et m_2 sont premiers entre eux, alors $\varphi(m_1 m_2) = \varphi(m_1)\varphi(m_2)$.*

Preuve : $(\mathbb{Z}/m_1m_2\mathbb{Z}, \cdot)$ est isomorphe (par ψ) à $(\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}, \cdot)$. Il y a donc autant d'éléments inversibles dans $\mathbb{Z}/m_1m_2\mathbb{Z}$ que dans $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$. Or un élément de ce dernier est inversible ssi chacune de ses deux composantes l'est, donc il y a $\varphi(m_1)\varphi(m_2)$ éléments inversibles. □

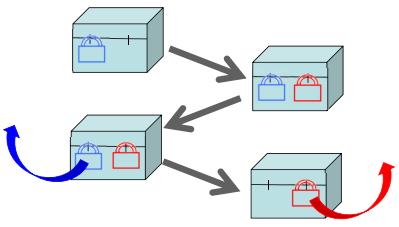
Q. 82. Soit $f : E \to E$ où E est un ensemble infini. Est-il vrai que si f est injective alors f est surjective ? Et réciproquement ?

Q. 83. Combien vaut $\varphi(35)$?

10.2 *Cryptographie à Clé Publique*

Reprenons le schéma général de la Figure 6.1 ; rappelons qu'un algorithme de chiffrement E, paramétré par une clé K, transforme le texte clair P en cryptogramme $C = E_K(P)$. L'algorithme de déchiffrement D, paramétré par une clé k, effectue la transformation inverse $P = D_k(C)$. Dans ce chapitre nous étudions un système de chiffrement où la clé de chiffrement K est publique, alors que la clé de déchiffrement k est secrète, connue seulement par le destinataire. Avec un tel système, un usager qui désire envoyer et recevoir des cryptogrammes se munit donc deux clés, l'une (k) qu'il garde secrète et l'autre (K) qu'il publie dans un annuaire que tous les autres usagers peuvent consulter. La distribution des clés est donc extrêmement facile.

Comment envoyer du chocolat à un ami dans un pays où les postiers aiment beaucoup le chocolat et ont tendance à le voler ? Si mon ami possède une clé de mon cadenas, c'est facile. Sinon, nous pouvons utiliser un double échange. Chacun garde alors les clés de son cadenas par devers soi.



C'est un problème semblable que résout la cryptographie à clé publique, mais avec un seul échange au lieu de deux.

Pour qu'un tel système fonctionne, il faut que certaines conditions soient remplies :

(1) (Exactitude) L'algorithme de déchiffrement doit rétablir le texte clair : $D_k(E_K(P)) = P$.
(2) Le chiffrement $E_K(P)$ d'un message clair P est une opération aisée et rapide.
(3) Le déchiffrement $D_k(C)$ d'un cryptogramme C est une opération aisée pour quiconque connaît la clé k.
(4) Par contre, le déchiffrement sans la connaissance de k est extrêmement difficile, et impossible à effectuer en pratique, en un temps raisonnable.
(5) Enfin, il doit être extrêmement difficile de deviner la clé privée k.

Dans la section suivante, nous allons étudier le système RSA et montrer qu'il satisfait à ces conditions.

10.3 *L'Algorithme de Rivest-Shamir-Adleman (RSA)*

Le cryptosystème RSA encode tous les messages (clairs ou chiffrés) comme des entiers modulo K (c'est à dire des éléments de $\mathbb{Z}/K\mathbb{Z}$). Les longs messages sont subdivisés en blocs, si bien que tous les messages sont compris entre 0 et $K-1$.

1. Le module K est toujours de la forme $K = pq$ où p et q sont des nombres premiers. Le module K est la clé publique.

La clé privée k est le plus petit multiple commun à $p-1$ et $q-1$.

Les facteurs p et q sont secrets et utilisés seulement pour construire la clé privée k. Ils peuvent être détruits une fois k calculée.

On peut aussi prendre comme clé privée n'importe quel multiple commun à $(p-1)$ et $(q-1)$, par exemple $k = \varphi(pq) = (p-1)(q-1)$.

2. Le chiffrement est défini par $C \stackrel{\text{def}}{=} E_K(P)$ tel que $[C]_K = ([P]_K)^e$, où l'exposant e est connu et public.
3. La clé publique K doit être telle que e est premier avec k.

   En pratique, on prend souvent $e = 65537$, qui est un nombre premier ; dans un tel cas, la condition que e soit premier avec k est équivalente à : e ne divise ni $(p-1)$ ni $(q-1)$.
4. Le déchiffrement est obtenu à l'aide d'un nombre f tel que $[f]_k = [e]_k^{-1}$. Notons que cet inverse existe précisément grâce à la condition précédente. Il peut être calculé simplement en utilisant l'identité de Bézout, si on connaît la clé privée k.

   Le déchiffrement est alors défini par $P' = D_k(C)$ avec $[P']_K = ([C]_K)^f$.

Notons que les congruences utilisées pour le chiffrement et le déchiffrement sont modulo K, alors que la congruence qui définit l'exposant de déchiffrement f est modulo k.

Exemple 10.1 *Chaque lettre du message clair est codée par sa position dans l'alphabet, et nous prenons comme clé publique $K = 33$. Les facteurs premiers sont $p = 3$ et $q = 11$. La clé privée k est le plus petit multiple commun à 2 et 10, d'où $k = 10$.*

Prenons comme exposant $e = 7$, qui est premier avec k. L'exposant de déchiffrement est $f = 3$ car $7 \cdot 3 \equiv 1 \pmod{10}$. Le chiffrement et le déchiffrement du mot "BONJOUR" sont montrés dans la Figure 10.3

	Texte clair codé P	Texte chiffré $C = ([P]_{33})^7$	Texte déchiffré $([C]_{33})^3$	
B	02	29	02	B
O	15	27	15	O
N	14	20	14	N
J	10	10	10	J
O	15	27	15	O
U	21	21	21	U
R	18	06	18	R

FIGURE 10.3: L'algorithme RSA avec le module $K = 33$ et clé exposant $e = 7$. La clé publique est K et la clé privée est $(3, 11)$.

Vérifions maintenant que cet algorithme satisfait aux conditions posées en 10.2. Commençons par une conséquence immédiate du théorème des restes chinois :

Théorème 10.3 (Exactitude de RSA) *Soient p et q deux nombres premiers distincts. Soit m un multiple commun à $p-1$ et $q-1$. Pour tout entier n :*

$$\left([n]_{pq}\right)^{1+m} = [n]_{pq} \tag{10.5}$$

☺*Preuve :* Puisque $(\mathbb{Z}/pq\mathbb{Z}, \cdot)$ est isomorphe (par l'application ψ des restes chinois) à $(\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}, \cdot)$, l'Eq.(10.5) est équivalente à

$$\left([n]_p\right)^{1+m} = [n]_p \tag{10.6}$$

$$\text{et } \left([n]_q\right)^{1+m} = [n]_q \tag{10.7}$$

Montrons l'Eq.(10.6). Distinguons deux cas :

1. Si $[n]_p = 0$ alors l'Eq.(10.6) est trivialement vraie.
2. Sinon, $[n]_p$ est inversible car p est premier (Théorème 8.6) et donc, par le Théorème d'Euler, $\left([n]_p\right)^{p-1} = [1]_p$. Par ailleurs m est multiple de $p-1$, donc il existe un entier $\ell \geq 0$ tel que $m = \ell(p-1)$. Donc

$$\left([n]_p\right)^m = \left([n]_p\right)^{(p-1)\ell} = \left([1]_p\right)^\ell = [1]_p$$

En multipliant par $[n]_p$ nous obtenons l'Eq.(10.6).

L'Eq.(10.7) se montre de la même façon. ☺□

Nous pouvons maintenant vérifier les conditions de la section 10.2.

(1) (Exactitude) Le message déchiffré est P' tel que $[P']_K = ([C]_K)^f = ([P]_K)^{ef}$. Or il existe $\ell \in \mathbb{Z}$ tel que $ef = 1 + \ell k$, d'après la condition 4 de l'algorithme RSA. Comme k (et donc aussi ℓk) est un multiple commun à $(p-1)$ et $(q-1)$, par le Théorème 10.3, $[P']_K = [P]_K$. Donc $P' = P$ puisque les messages sont des entiers compris entre 0 et $K-1$.

(2) Le chiffrement est un calcul de puissance, qui est facile comme nous l'avons vu sur l'Exemple 8.1. L'exposant $e = 65537$ est particulièrement intéressant, voir ci-contre.

Pour $e = 65537 = 2^{16} + 1$, le calcul de $y = x^e$ peut se faire en élevant 16 fois au carré puis en multipliant par x :

1: $y \leftarrow x$
2: **for** $i = 1$ to 16 **do**
3: $\quad y \leftarrow y^2$
4: **end for**
5: $y \leftarrow y \cdot x$

Pour le chiffrement, il suffit d'appliquer cet algorithme avec $x = [P]_K$, nous obtenons alors $y = [C]_K$.

(3) Si on connaît la clé privée k, le calcul de l'exposant de déchiffrement f est simple comme nous l'avons déjà vu. Le déchiffrement est également un calcul de puissance, qui est donc aussi un problème facile.

(4) Le déchiffrement sans connaître la clé privée k revient à résoudre l'équation $x^e = [C]_K$ où tout est connu sauf x, c'est à dire calculer la racine e-ième de $[C]_K$. La méthode exhaustive consiste à essayer toutes les valeurs de x (et il faut le faire pour tous les blocs du message chiffré). Il faut donc que soit un nombre très grand pour que cette méthode soit chère en temps de calcul. Une autre méthode utilise le logarithme discret (Exemple 8.1), dont le calcul est aussi un problème difficile pour K grand. On pense donc aujourd'hui qu'il n'y a pas de moyen simple pour calculer la racine e-ième dans $\mathbb{Z}/K\mathbb{Z}$.

L'élévation à la puissance e est donc une fonction à sens unique (pour quiconque ne connaît pas la clé privée).

(5) Pour que la dernière condition soit satisfaite, il doit être extrêmement compliqué de déchiffrer sans connaître (p, q), ce qui revient à ce que la la factorisation de K en nombres premiers soit difficile à calculer. Une méthode de factorisation consisterait par exemple à essayer tous les entiers jusqu'à $\sqrt{K}$ pour retrouver p et q. D'autres algorithmes sont plus rapides, mais restent totalement inefficaces si K est très grand. Rivest, Shamir et Adleman ont calculé qu'il faudrait 4 millions d'années pour factoriser un nombre m de 200 chiffres avec un processeur de 1Ghz. On pense aujourd'hui que la factorisation de nombres entiers très grands est un problème difficile.

C'est donc sur les hypothèses de quasi-impossibilité de calcul de la racine dans $\mathbb{Z}/K\mathbb{Z}$ et de la factorisation en nombres premiers que repose la sécurité de RSA.

10.4 *Choix des Paramètres du Cryptosystème RSA : Nombres Premiers Sûrs*

Bien que le système RSA satisfasse à toutes les conditions de la Section 10.3, il est nécessaire de bien choisir les nombres entiers p et q, sans quoi certains problèmes peuvent survenir, en particulier celui des *messages non cachés*. Lors du chiffrement de l'Exemple 10.3, les nombres 10 et 21 du texte clair sont cryptés respectivement en

10 et 21 ! En effet, $10 \equiv 10^7 \pmod{33}$ et $21 \equiv 21^7 \pmod{33}$, et donc le cryptogramme et le texte clair sont identiques pour ces valeurs, ce qui est évidemment tout bénéfice pour le cryptanalyste. Le texte clair $[P]_K = [0]_K$ n'est jamais caché. En général, les textes clairs P qui ne sont pas cachés sont ceux qui satisfont

$$([P]_K)^e = [P]_K \tag{10.8}$$

Il peut y avoir beaucoup de solutions à cette équation en P, sauf si on choisit pour p et q des nombres premiers "sûrs".

Définition 10.1 *Un nombre premier p est dit sûr s'il est de la forme $p = 2p' + 1$ où p' est aussi un nombre premier.*

Il existe beaucoup de nombres premiers sûrs, on pense que leur nombre est infini.
Q. 84. Les nombres suivants sont-ils des nombres premiers sûrs : 17, 83, 107 ?

Le théorème suivant dit que dans un tel cas, et avec un exposant de chiffrement aussi bien choisi, le nombre de messages P qui ne sont pas cachés est 9 (cela inclus le message $P = 0$). Comme $K = pq$ est très grand, leur nombre est infime et il doit donc être possible d'éviter de tels message avec grand probabilité.

Théorème 10.4 *Soient p et q des nombres premiers sûrs distincts supérieurs à 5 et supposons que l'exposant de chiffrement e est tel que $e - 1$ est une puissance de 2. Le nombre de solutions $[P]_K \in \mathbb{Z}/K\mathbb{Z}$ de Eq.(10.8) est égal à 9.*

Par exemple $e = 65537 = 2^{16} + 1$ satisfait cette propriété.
Q. 85. Montrez que si $e - 1$ est impair et p, q sont des entiers premiers sûrs $> 2e + 1$ alors e satisfait les conditions de RSA, i.e. e est premier avec $\text{ppcm}(p - 1, q - 1)$.

Preuve : Soient $p = 2p' + 1$ et $q = 2q' + 1$ avec p et q premiers distincts (donc $p' > 2$ et $q' > 2$). Nous avons à résoudre l'équation $z^e = z$ où l'inconnue est $z \in \mathbb{Z}/pq\mathbb{Z}$. Faisons le changement de variable $\psi(z) = (x, y)$ où $x \in \mathbb{Z}/p\mathbb{Z}, y \in \mathbb{Z}/q\mathbb{Z}$. Par le théorème des restes chinois :

$$z^e = z,\ z \in \mathbb{Z}/pq\mathbb{Z} \Leftrightarrow \begin{cases} x^e = x,\ x \in \mathbb{Z}/p\mathbb{Z} \\ \text{et} \\ y^e = y,\ y \in \mathbb{Z}/q\mathbb{Z} \end{cases}$$

Étudions l'équation en x. Elle équivaut à

$$(x^{e-1} - [1]_p)x = [0]_p,\ x \in \mathbb{Z}/p\mathbb{Z}$$

De deux choses l'une : soit $x = [0]_p$, soit $x \neq [0]_p$. Dans le premier cas $x = [0]_p$ est solution ; dans le deuxième cas, x est inversible (car p est premier) donc nous pouvons simplifier par x (en multipliant les deux membres par $[x]_p^{-1}$) et donc $x^{e-1} = [1]_p$.

Dans ce dernier cas, la période d de x dans $\mathbb{Z}/p\mathbb{Z}^*$ divise $e - 1$, qui est une puissance de 2, donc d est une puissance de 2 ; elle divise aussi le cardinal de $\mathbb{Z}/p\mathbb{Z}^*$, qui est $p - 1 = 2p'$; comme p' est premier et $\neq 2$, d divise 2, donc $d = 1$ ou 2. Donc $x = [1]_p$ ou sinon $x^2 = [1]_p$. Dans ce dernier cas, $(x - [1]_p)(x + [1]_p) = [0]_p$ et donc comme $x - [1]_p \neq [0]_p$, nous pouvons simplifier par $(x - [1]_p)$ et $x = [-1]_p$. Notons que $[-1]_p \neq [1]_p$ car $p > 2$.

En résumé, l'équation $x^e = x,\ x \in \mathbb{Z}/p\mathbb{Z}$ a trois solutions : $x \in \{[0]_p, [1]_p, [-1]_p\}$. De même $y^e = y,\ y \in \mathbb{Z}/q\mathbb{Z}$ a trois solutions : $y \in \{[0]_q, [1]_q, [-1]_q\}$. A chaque solution en x et en y correspond

une solution $z = \psi^{-1}(x, y)$ et il y a $3 \times 3 = 9$ façons de choisir le couple de solutions (x, y). Donc l'équation $z^e = z, z \in \mathbb{Z}/pq\mathbb{Z}$ a 9 solutions. □

III

Codes Correcteurs

11
Les Codes Correcteurs ou Détecteurs

Maintenant que nous savons comment comprimer l'information et la sécuriser contre des attaques, il nous reste à comprendre comment la *protéger* contre les erreurs. De telles erreurs arrivent quotidiennement : sur un CD à cause des rayures ou de la poussière ; dans un ordinateur lorsque les données sont lues et perturbées par le bruit thermique introduit par un système électrique ; ou encore dans la mémoire d'un lecteur MP3 quand les circuits ont vieilli et que certains transistors sont morts.

Pour protéger l'information, l'idée est toujours d'ajouter des bits (appelés bits de *redondance*), en utilisant ce qu'on appelle un *code correcteur ou détecteur* (aussi appelé *code* tout court). Ces bits de redondance sont utilisés lors du décodage pour reconstruire l'information initiale même s'il y a des bits perdus ou erronés. Les chiffres de contrôle MOD 97-10 de l'IBAN que nous avons rencontrés au Chapitre 7 sont un tel code. Il est peu efficace, et nous allons voir dans cette partie comment fabriquer des codes bien meilleurs. En particulier, nous allons construire les codes de Reed-Solomon, qui sont utilisés dans un très grand nombre de systèmes, par exemple pour lire un code-barre ou pour obtenir des données depuis un CD ou un disque dur.

Dans la Partie I, nous avons utilisé des codes de sources. Les codes dont nous parlons ici sont différents. Les codes de source compriment l'information ; au contraire, les code correcteurs ou détecteurs augmentent la taille de l'information. Typiquement, les données sont d'abord comprimées à l'aide d'un code de source, puis on utilise un code correcteur ou détecteur avant de les stocker ou de les transmettre.

11.1 Codes Correcteurs ou Détecteurs

Commençons par un exemple simple.

Exemple 11.1 *Nous voulons stocker un fichier sur un disque.*

Nous supposons que le fichier comporte $k = 3$ bits ; la valeur de k pour n'importe quel vrai fichier sera bien sûr beaucoup plus grande que 3, mais en considérant une si petite valeur nous pouvons comprendre plus

$k = 3$		$n = 7$
000	$\mapsto$	0000000
001	$\mapsto$	0011100
010	$\mapsto$	0111011
100	$\mapsto$	1110100
011	$\mapsto$	0100111
101	$\mapsto$	1101000
110	$\mapsto$	1001111
111	$\mapsto$	1010011

Figure 11.1: Un code de longueur $n = 7$ et sa table d'encodage.

facilement le concept.

Nous ne pouvons pas savoir par avance la valeur des 3 *bits. Il nous faut donc prévoir tous les cas. Il y a* $2^3 = 8$ *fichiers possibles* (000, 001, 010, 011, 100, 101, 110 *et* 111*). A chaque cas possible nous faisons correspondre une suite de* $n = 7$ *bits ; toutes ces suites, appelées mots de code, doivent être toutes distinctes. Nous espérons pouvoir utiliser l'information redondante qui se trouve dans le mot de code ensuite, quand nous voudrons reconstruire le fichier original d'une observation peut-être altérée. La table d'encodage est fixée une fois pour toutes, et est connue aussi bien par la personne qui écrit les données sur le disque (lors de l'encodage) que la personne qui lit les données (lors du décodage).*

Le code *$\mathcal{C}$ est l'ensemble de tous les mots de code. Nous disons que c'est un code en bloc, de longueur* $n = 7$.

Le code utilise 7 *bits par mot de code, alors qu'il suffit de* $k = 3$ *bits pour décrire les* 8 *mots originaux. Nous pouvons donc dire que le code est de* rendement $r = \frac{k}{n} = \frac{3}{7}$.

$\mathcal{A}^n$ est le produit cartésien $\mathcal{A} \times ... \times \mathcal{A}$. Un élément de $\mathcal{A}^n$ (appelé "mot" dans le contexte des codes) est une suite $(a_1, a_2, ..., a_n)$ de n éléments de $\mathcal{A}$. L'ordre compte et il peut y avoir des répétitions.

Définition 11.1 (*Code en Bloc***)** *Un code en bloc de longueur n, défini sur un alphabet $\mathcal{A}$, est un sous-ensemble $\mathcal{C}$ de $\mathcal{A}^n$, c'est à dire un ensemble de suites de n éléments de $\mathcal{A}$. Les éléments du code sont appelés les mots de code.*

Le rendement du code est défini par $r = \frac{1}{n} \log_{card(\mathcal{A})} card(\mathcal{C})$.

L'expression "en bloc" signifie que tous les mots de code ont la même longueur n, ce qui est le seul cas auquel nous nous intéressons dans cette partie.

Notons que contrairement aux codes de sources, ici la table de correspondance ne joue pas de grand rôle, seul l'ensemble des mots de code est important.

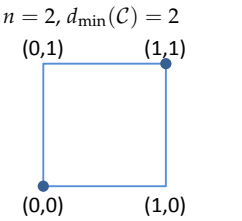
Le rendement r est aussi appelé *débit* (Ang. *rate*) du code.

Le rendement du code mesure son coût en nombre de symboles. Notons que pour l'exemple précédent nous avions $r = \frac{1}{7} \log_2(8) = \frac{3}{7}$, ce qui est compatible avec la définition.

Q. 86. Considérons le code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 (Exemple 7.2, page 63). Quel est le rendement de ce code ?

11.2 Distance de Hamming

Jusqu'ici nous avons vu qu'un code en bloc a une longueur n et un rendement r. Mais comment pouvons nous choisir un code ? Qu'est-ce que c'est un bon code ? Pour cela il nous faut évaluer son efficacité, ce qui se fait à l'aide des concepts de *distance de Hamming* et *distance minimale*.



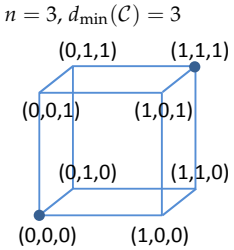


FIGURE 11.2: Un code en bloc avec 2 mots de code. La longueur du code est $n = 2$ (en haut) ou $n = 3$ (en bas).

Définition 11.2 (Distance de *Hamming***)** *Soit $\mathcal{A}$ un ensemble fini (l'alphabet) et $n \geq 1$ un entier. Soient $x = (x_1, ..., x_n) \in \mathcal{A}^n$ et $y = (y_1, ..., y_n) \in \mathcal{A}^n$ deux suites de n éléments de $\mathcal{A}$. La distance de Hamming $d(x, y)$ est le nombre de positions où x et y diffèrent :*

$$d(x, y) \stackrel{\text{def}}{=} card\,\{i \in \{1, ..., n\} \text{ tels que } x_i \neq y_i\}$$

Par exemple,

$$x = (1, 0, 1, 1, 1, 0) \text{ et}$$
$$y = (1, 0, 0, 1, 1, 1)$$

ne diffèrent qu'en leur troisième et dernière positions, donc leur distance de Hamming est $d(x, y) = 2$. Notons que le cardinal de l'ensemble vide est 0 donc $d(x, x) = 0$.

Théorème 11.1 (Distance de Hamming) *La distance de Hamming possède les trois propriétés suivantes. Pour tous $x, y, z \in \mathcal{A}^n$:*

Un ensemble E sur lequel est défini une distance vérifiant les trois propriétés du Théorème 11.1 est appelé *espace métrique*. L'ensemble $\mathcal{A}^n$ des mots de n symboles, muni de la distance de Hamming, est donc un espace métrique.

1. $d(x,y) \geq 0$ *et* $d(x,y) = 0$ *si et seulement si* $x = y$;
2. *(symétrie)* $d(x,y) = d(y,x)$;
3. *(inégalité triangulaire)* $d(x,z) \leq d(x,y) + d(y,z)$.

☺*Preuve :* Nous montrons seulement l'inégalité triangulaire, les deux premiers items sont laissés aux bons soins du lecteur ou de la lectrice. Soit A l'ensemble des positions où x et y diffèrent, B l'ensemble des positions où y et z diffèrent et C l'ensemble des positions où x et z diffèrent. Nous avons donc $d(x,y) = \text{card}(A)$, $d(y,z) = \text{card}(B)$ et $d(x,z) = \text{card}(C)$. Or :

$$(x_i = y_i \text{ et } y_i = z_i) \Rightarrow (x_i = z_i)$$

donc par contraposition

$$(x_i \neq z_i) \Rightarrow (x_i \neq y_i \text{ ou } y_i \neq z_i)$$

c'est à dire que

$$(i \in C) \Rightarrow (i \in A) \text{ ou } (i \in B)$$

ou encore

$$C \subset (A \cup B)$$

donc $\text{card}(C) \leq \text{card}(A \cup B) \leq \text{card}(A) + \text{card}(B)$. ☺□

La négation de la phrase"A et B" est "non A ou non B".

Si les deux ensembles E et F sont finis, et si E est inclus dans F (c'est à dire $E \subset F$) alors le nombre d'éléments de E est $\leq$ celui de F (c'est à dire $\text{card}(E) \leq \text{card}(F)$).

Le nombre d'éléments de la réunion de E et F est la somme des cardinaux de chacun, moins le nombre de doublons :

$$\text{card}(E \cup F) = \text{card}(E) + \text{card}(F) - \text{card}(E \cap F)$$

En particulier :

$$\text{card}(E \cup F) \leq \text{card}(E) + \text{card}(F)$$

La distance de Hamming possède les propriétés habituelles d'une distance, de la même façon que la distance euclidienne $d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ définie sur $\mathbb{R}^2$: c'est pourquoi nous pouvons ré-utiliser une grande partie de notre intuition géométrique.

Définition 11.3 (*Distance minimale***)** *La distance minimale d'un code en bloc $\mathcal{C}$, notée $d_{\min}(\mathcal{C})$, est*

$$d_{\min}(\mathcal{C}) \stackrel{\text{def}}{=} \min_{x,y \in \mathcal{C}; x \neq y} d(x,y)$$

Autrement dit, la distance minimale d'un code est la plus petite distance de Hamming entre deux mots de code distincts. Comme nous allons voir par la suite, la distance minimale reflète bien la capacité d'un code à détecter ou corriger des erreurs, et un bon code est un code qui a une grande distance minimale.

Exemple 11.2 (Distance Minimale du code de la Figure 11.1) *Pour déterminer la distance minimale d'un code donné avec M mots de code, nous devons prendre le minimum sur toutes les paires distinctes de mots de code. Nous devons donc vérifier le nombre de combinaisons de 2 pris parmi M, à savoir, $\binom{M}{2} = \frac{M(M-1)}{2}$ paires. Pour notre exemple, $M = 8$, et donc nous devons vérifier 28 paires. Ceci est encore faisable, et nous obtenons $d_{\min}(\mathcal{C}) = 3$.*

Pour les codes utilisés en pratique, la vérification directe peut ne plus être possible. Par exemple, nous allons bientôt étudier des codes de Reed-Solomon où le nombre de mots de code est de l'ordre $M = 256^{128}$. Il nous faudra trouver d'autres méthodes pour déterminer la distance minimale.

Q. 87. Quelle est la distance minimale du code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 (Exemple 7.2, page 63) ?

Une *paire* est un ensemble à 2 éléments, et est noté $\{i,j\}$. L'ordre n'a pas d'importance ($\{i,j\} = \{j,i\}$), et il faut que $i \neq j$ (pas de répétition). Si E est un ensemble à M éléments, il y a $\frac{M(M-1)}{2}$ paires d'éléments de E.

Ne confondez pas avec le couple, noté (i,j) ou encore ij, qui est une suite de 2 éléments ; l'ordre compte et il est possible que $i = j$ (les répétitions sont possibles). Il y a M^2 couples d'éléments de E.

11.3 Modèles de canal

La nature des phénomènes qui provoquent des erreurs dans les fichiers peut être arbitrairement compliquée, et nous voulons éviter de considérer chaque cas séparément. C'est pourquoi nous allons introduire des *modèles de canal*, qui sont des abstractions mathématiques qui ne retiennent que certaines propriétés importantes du canal physique. Nous allons utiliser deux modèles : le canal *à effacements* et le canal *à erreurs*.

Définition 11.4 (Canal à Effacements) *Pour le canal à effacements, nous supposons que chaque composante du mot de code est soit connue parfaitement, soit* effacée. *L'effacement signifie que le symbole transmis à une position effacée a été remplacé par un symbole spécial, disons le symbole " ?". Le destinataire sait quelles positions ont été effacées, mais ne sait pas quelles symboles étaient présents avant l'effacement.*

Le poids d'un effacement est le nombre de positions qui sont modifiées.

Par exemple, avec le code de la Figure 11.1, supposons que le mot transmis soit $x = (0100111)$. Une sortie d'un canal à effacement est par exemple $y = (0?001?1)$: les bits effacés du canal sont les composantes 2 et 6. Le poids de l'effacement est 2.

Définition 11.5 (Canal à Erreurs) *Pour le canal à erreurs, chaque composante du mot de code est soit reçue parfaitement, soit échangée pour un autre symbole de l'alphabet. Le destinataire ne sait pas quelles positions sont victimes d'erreurs.*

Le poids d'une erreur est le nombre de positions qui sont modifiées.

Toujours avec le code de la Figure 11.1, supposons que le mot de code transmis soit $x = (0100111)$. Une sortie possible d'un canal à erreurs est $y = (0000101)$. Le canal a introduit deux erreurs, sur les positions 2 et 6. Le poids de cette erreur est 2.

11.4 Les Théorèmes de la Distance Minimale

Dans cette section nous allons voir pourquoi la distance minimale permet de quantifier la puissance d'un code.

Supposons que l'opérateur d'une station nucléaire veuille envoyer un signal d'alarme au réacteur nucléaire pour réduire sa puissance ; ou, moins dramatiquement, que vous vouliez sauvegarder les données de votre compte bancaire sur un disque. Dans les deux cas, nous voulons assurer que le signal émis (l'information sauvegardée) est reçue (est lisible) correctement. Autrement dit, nous voulons assurer que les erreurs de transmission sont détectées.

Soit $\mathcal{C}$ le code utilisé. Supposons que nous choisissions un mot de code $x \in \mathcal{C}$, qui est envoyé sur un canal à erreurs. Le destinataire observe la sortie du canal, que nous notons y. Bien sûr, le destinataire *ne connaît pas* le mot de code émis x. Il connaît seulement le mot reçu y. Si le mot reçu y n'est pas un mot de code, alors le destinataire sait que le canal a introduit des erreurs. Dans ce cas,

le destinataire peut donner l'alerte pour prévenir l'utilisateur que la transmission a été erronée. Nous disons que le destinataire peut *détecter l'erreur*.

Q. 88. Y-a-t'il un résultat analogue au Théorème 11.2 pour la détection d'effacements ?

Théorème 11.2 (Détection d'Erreurs) *(1) Un code $\mathcal{C}$ est capable de détecter toutes les erreurs de poids $p < d_{\min}(\mathcal{C})$.*

(2) Inversement, si un code $\mathcal{C}$ peut détecter toutes les erreurs de poids $\leq p$, alors $p < d_{\min}(\mathcal{C})$.

☺*Preuve :* (1) Soit x un mot de code transmis et y le mot reçu, avec erreur de poids p. Il nous faut montrer que

$$p < d_{\min}(\mathcal{C}) \Rightarrow \text{ l'erreur est détectable}$$

Or l'erreur est détectable si et seulement si $y \notin \mathcal{C}$. Il nous faut donc montrer :

$$p < d_{\min}(\mathcal{C}) \Rightarrow y \notin \mathcal{C}$$

Nous allons montrer la contraposée :

$$y \in \mathcal{C} \Rightarrow p \geq d_{\min}(\mathcal{C})$$

Nous supposons donc maintenant que $y \in \mathcal{C}$. Comme il y a une erreur $d(x,y) \neq 0$; de plus $x \in \mathcal{C}$. Donc, par définition de la distance minimale, $p = d(x,y) \geq d_{\min}(\mathcal{C})$.

(2) Montrons la contraposée :

Si $p \geq d_{\min}(\mathcal{C})$ alors il existe des erreurs de poids $\leq p$ non détectables.

Par définition de la distance minimale, il existe au moins deux mots de code x, y tels que $d(x,y) = d_{\min}(\mathcal{C})$. Transmettons le mot x de sorte que le mot y soit reçu. L'erreur est non détectable car y est un mot de code. Le poids de l'erreur est $d_{\min}(\mathcal{C}) \leq p$. Donc il existe au moins une erreur de poids $\leq p$ non détectable. ☺□

Par exemple, avec le code de la Figure 11.1, supposons que le mot de code transmis soit $x = (0100111)$ et le mot reçu $y = (0000101)$. Le poids de l'erreur est 2, et nous savons que la distance minimale du code est 3, donc cette erreur peut être détectée. Effectivement, en inspectant la liste de tous les mots du code, nous voyons que y n'y figure pas.

Q. 89. Quels effacements peut-on corriger avec le code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 (Exemple 7.2, page 63) ?

Détecter c'est bien, corriger c'est mieux. Dans le cas d'un canal à effacements, la correction consiste à trouver un mot de code x qui soit compatible avec le mot reçu y. Si le nombre d'effacements n'est pas trop grand, cela est possible :

Théorème 11.3 (Correction d'Effacements) *(1) Un code $\mathcal{C}$ est capable de corriger tous les effacements de poids $p < d_{\min}(\mathcal{C})$.*

(2) Inversement, si un code $\mathcal{C}$ peut corriger tous les effacements de poids $\leq p$, alors $p < d_{\min}(\mathcal{C})$.

☺*Preuve :* (1) Soit x le mot transmis et y le mot reçu à travers un canal à effacement. Le mot y n'est pas un mot du code $\mathcal{C}$ mais est

un mot de n symboles construit sur l'alphabet étendu $\mathcal{A}' = \mathcal{A} \cup \{?\}$. Soit p le poids de l'erreur, donc $d(x, y) = p$.

Nous connaissons y mais pas x ; nous ne connaissons pas p non plus, mais nous savons que $p < d_{\min}(\mathcal{C})$. Pour corriger l'effacement, nous cherchons un mot x qui ne diffère de y que sur les positions où il y a un effacement. Il en existe au moins un, par hypothèse. Nous allons montrer par l'absurde qu'il n'en existe pas d'autre.

En effet, soit x' un deuxième mot possible. Les mots x et x' ne diffèrent que dans les positions où il y a eu effacement. Donc, $d(x, x') \leq p < d_{\min}(\mathcal{C})$. Comme x et x' sont deux mots de code distincts, $d_{\min}(\mathcal{C}) \leq d(x, x')$, ce qui est une contradiction avec l'inégalité précédente.

(2) Montrons la contraposée :

Si $p \geq d_{\min}(\mathcal{C})$ alors il existe des effacements incorrigibles de poids $\leq p$.

Par définition de la distance minimale, il existe au moins deux mots de code distincts x, x' tels que $d(x, x') = d_{\min}(\mathcal{C})$. Il y a donc $d_{\min}(\mathcal{C})$ positions où les symboles de x et x' sont identiques. Transmettons x, respectivement x', dans un canal à effacement et effaçons précisément ces positions. Dans les deux cas le mot reçu y est le même. En recevant y, il est impossible de savoir si c'est x ou x' qui a été transmis. Donc ces deux effacements sont incorrigibles. Le poids de l'effacement est $d_{\min}(\mathcal{C}) \leq p$. Donc il existe au moins un effacement de poids $\leq p$ non corrigible. ☺□

Par exemple, toujours avec le code de la Figure 11.1, supposons que le mot de code transmis soit $x = (0100111)$ et le mot reçu $y = (0?001?1)$. Le poids de l'effacement est 2, et nous savons que la distance minimale du code est 3, donc cet effacement peut être corrigé. Effectivement, en inspectant la liste de tous les mots du code, nous voyons que $x = (0100111)$ est le seul mot de code compatible avec y.

Dans le cas d'un canal à erreurs, la correction est plus difficile, car on ne sait pas à quelles positions il y a des erreurs. Une méthode simple consiste à chercher le mot x le plus proche du mot y reçu. Si le nombre d'erreurs n'est pas trop grand, cela marche :

Q. 90. Quelles erreurs peut-on détecter ou corriger avec le code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 (Exemple 7.2, page 63) ?

Théorème 11.4 (Correction d'erreurs) *(1) Un code $\mathcal{C}$ est capable de corriger toutes les erreurs de poids $p < \frac{d_{\min}(\mathcal{C})}{2}$.*

Plus précisément, si l'erreur est de poids $p < \frac{d_{\min}(\mathcal{C})}{2}$, le mot transmis x est le mot le plus proche (pour la distance de Hamming) du mot reçu y.

(2) Inversement, si un code $\mathcal{C}$ peut corriger toutes les erreurs de poids $\leq p$ alors $p < \frac{d_{\min}(\mathcal{C})}{2}$.

Preuve : (1) Soit x le mot transmis et y le mot reçu à travers un canal à erreurs. Soit p le poids de l'erreur, donc $d(x, y) = p$ et, par hypothèse, $p < \frac{d_{\min}(\mathcal{C})}{2}$. Nous connaissons y mais pas x ; nous ne connaissons pas p non plus, mais nous savons que $p < \frac{d_{\min}(\mathcal{C})}{2}$. Pour corriger l'erreur, nous cherchons un mot qui soit à distance de y inférieure à $\frac{d_{\min}(\mathcal{C})}{2}$. Il en existe au moins un, par hypothèse, le mot

x transmis. Nous allons montrer par l'absurde qu'il n'en existe pas d'autre.

En effet, soit x' un deuxième mot possible. Nous avons, par construction :

$$d(x,y) < \frac{d_{\min}(\mathcal{C})}{2}$$
$$d(y,x') < \frac{d_{\min}(\mathcal{C})}{2}$$

donc par l'inégalité triangulaire :

$$d(x,x') \leq d(x,y) + d(y,x') < d_{\min}(\mathcal{C})$$

Or x et x' sont deux mots de code distincts, donc $d_{\min}(\mathcal{C}) \leq d(x,x')$, ce qui est une contradiction.

Donc nous avons montré que pour tout autre mot de code x', $d(x',y) \geq \frac{d_{\min}(\mathcal{C})}{2}$. Donc $d(x,y) < d(x',y)$ et x est le mot de code le plus proche de y.

(2) Nous raisonnons par l'absurde. Nous supposons donc que l'hypothèse est vraie et que la conclusion est fausse. Posons $\delta = d_{\min}(\mathcal{C})$. Nous supposons donc que les entiers positifs p et δ sont tels que le code $\mathcal{C}$ peut corriger toutes les erreurs de poids $\leq p$ et $\frac{\delta}{2} \leq p$.

Montrons tout d'abord des relations intéressantes. Posons $p_1 = \lfloor \delta/2 \rfloor$, $p_2 = \lfloor (\delta+1)/2 \rfloor$ et montrons que

$$p_1 + p_2 = \delta \tag{11.1}$$
$$p_1 \leq p \text{ et } p_2 \leq p \tag{11.2}$$

Par exemple, pour $\delta = 5$, $p_1 = 2$ et $p_2 = 3$ et nous avons bien $p_1 + p_2 = \delta$.
Si l'entier p vérifie $\frac{\delta}{2} \leq p$ alors $2.5 \leq p$ et donc $3 \leq p$. Nous avons bien $p_1 \leq p$ et $p_2 \leq p$.

Pour cela considérons séparément les cas pair et impair :

- Si δ est pair alors $\delta = 2\lambda$ avec λ entier positif. Nous avons $p_1 = p_2 = \lambda$ ce qui prouve l'Eq.(11.1). De plus si l'entier p vérifie $\frac{\delta}{2} \leq p$ alors $\lambda \leq p$, ce qui prouve l'Eq.(11.2).
- Si δ est impair alors $\delta = 2\lambda + 1$ avec λ entier positif ou nul. $p_1 = \lambda$ et $p_2 = \lambda + 1$, ce qui prouve l'Eq.(11.1). De plus si l'entier p vérifie $\frac{\delta}{2} \leq p$ alors $\lambda + 0.5 \leq p$, donc (car λ et p sont entiers) $\lambda + 1 \leq p$ ce qui prouve l'Eq.(11.2).

Revenons maintenant à notre preuve par l'absurde. Par définition de la distance minimale, il existe au moins deux mots de code x, x' tels que $d(x,x') = \delta$. Soit $\mathcal{I}$ l'ensemble des positions où les mots x et x' diffèrent ; le cardinal de $\mathcal{I}$ est δ. Soit $\mathcal{I}_1$ le sous-ensemble des p_1 premiers éléments de $\mathcal{I}$ et $\mathcal{I}_2$ le sous-ensemble des p_2 éléments suivants de $\mathcal{I}$. Rappelons que $p_1 + p_2 = \delta$ donc nous avons réalisé une partition de $\mathcal{I}$. Définissons le mot y de la façon suivante :

Par exemple, avec $\delta = 5$ nous pourrions avoir

$$x = 10100000$$
$$x' = 10111111$$

Ici $\mathcal{I}$ est l'ensemble des positions 3 à 8, i.e. $\mathcal{I} = \{4,5,6,7,8\}$. Nous avons alors $\mathcal{I}_1 = \{4,5\}$, $\mathcal{I}_2 = \{6,7,8\}$ et

$$y = 10100111$$

Le mot y résulte de la transmission de x avec $p_1 = 2$ erreurs ou de la transmission de x' avec $p_2 = 3$ erreurs.

$$y_i = x_i \text{ si } i \in \mathcal{I}_1$$
$$y_i = x'_i \text{ si } i \in \mathcal{I}_2$$
$$y_i = x_i = x'_i \text{ si } i \notin \mathcal{I}$$

Le mot y résulte de l'injection de p_1 erreurs dans x, ou de l'injection de p_2 erreurs dans x'. Considérons un canal à erreur dans lequel nous transmettons le mot x, et qui délivre y. Comme $p_1 \leq p$, il est possible de corriger ces erreurs et décider que c'est x qui a été transmis. Supposons maintenant que nous transmettions le mot x'

et que le canal délivre aussi y. Comme $p_2 \leq p$, nous pouvons aussi décoder et décider que c'est x' qui a été transmis. Dans les deux cas nous avons reçu le même mot y donc il est impossible de savoir si c'est x ou x' qui a été transmis ; il y a une contradiction et notre preuve par l'absurde est achevée. □

Q. 91. Quelles erreurs peut corriger un code de distance minimale égale à 4 ?

Par exemple, continuons avec le code de la Figure 11.1, et supposons que le mot de code transmis soit $x = (0100111)$. Supposons que nous ayons reçu $y = (0000111)$. En inspectant la liste des mots de code, nous voyons que le seul élément de $\mathcal{C}$ à la distance $\frac{d_{\min}-1}{2} = 1$ de y est le mot $x = (0100111)$. Donc, nous déclarons que le mot transmis était $x = (0100111)$, ce qui est correct. Ce code a une distance minimale de 3, donc il est capable de corriger toutes les erreurs portant sur 1 seul bit.

Enfin nous terminons par une inégalité qui montre que la distance minimale ne peut pas être arbitrairement grande.

Théorème 11.5 (Borne de *Singleton*) [1]*Pour un code en bloc $\mathcal{C}$ de longueur n et de rendement r la distance minimale satisfait :*

$$d_{\min}(\mathcal{C}) \leq n(1-r)+1$$

1. R. Singleton. Maximum distance q-nary codes. *Information Theory, IEEE Transactions on*, 10(2):116–118, 1964

☺*Preuve :* Posons $\delta = d_{\min}(\mathcal{C})$. Soit f l'application qui, à un mot de code x, associe le mot obtenu en supprimant les $\delta - 1$ derniers symboles. C'est donc une application $\mathcal{C} \to \mathcal{A}^{n-\delta+1}$.

Montrons que f est injective. Soient deux mots de code $x \neq x'$; montrons par l'absurde que $f(x) \neq f(x')$. Supposons que $f(x) = f(x')$, alors x et x' ne peuvent différer que dans leur $\delta - 1$ derniers symboles, donc $d(x, x') \leq \delta - 1$, ce qui contredit la définition de la distance minimale. Donc, par le principe des tiroirs,

$$\text{card}(\mathcal{C}) \leq \text{card}(\mathcal{A}^{n-\delta+1}) = [\text{card}(\mathcal{A})]^{n-\delta+1}$$

En prenant le logarithme à base $\text{card}(\mathcal{A})$ nous obtenons

$$\log_{\text{card}(\mathcal{A})} \text{card}(\mathcal{C}) \leq n - \delta + 1$$

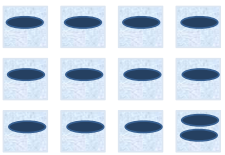
or le terme de gauche vaut rn (par définition du rendement r), d'où le résultat voulu. ☺□

Nous utilisons la variante suivante du *principe des tiroirs* ou *principe des boîtiers*, (Ang. *pigeon holes*). Soient E et F des ensembles finis et f une application $E \to F$. Cette variante du principe dit que :

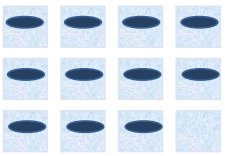
1. Si f est injective alors $\text{card}(E) \leq \text{card}(F)$.
2. Si f est surjective alors $\text{card}(E) \geq \text{card}(F)$.
3. Si f est bijective alors $\text{card}(E) = \text{card}(F)$.

Par contraposition, les items 1 et 2 donnent :

1 bis. Si $\text{card}(E) > \text{card}(F)$, f n'est pas injective.
2 bis. Si $\text{card}(E) < \text{card}(F)$, f n'est pas surjective.



$\text{card}(E) > \text{card}(F)$, f ne peut pas être injective



$\text{card}(E) < \text{card}(F)$, f ne peut pas être surjective

Pour le code de la Figure 11.1, la borne de Singleton donne $d_{\min}(\mathcal{C}) \leq 5$, alors que nous savons que $d_{\min}(\mathcal{C}) = 3$, c'est à dire que la borne n'est pas atteinte. Nous verrons dans la suite de ce module des codes qui atteignent la borne.

Exemple 11.3 (Le Robot-Code) *Le but de cet exemple est d'illustrer ce que signifie la borne de Singleton. Nous avons un code $\mathcal{C}$ de longueur n sur un alphabet $\mathcal{A}$, et le nombre de mots de code est $[card(\mathcal{A})]^k$ où k est un entier $< n$. Le code est utilisé pour encoder des messages de longueur k, et les mots de code ont une longueur n.*

Un message de k symboles est choisi et encodé. Un robot-code envoie à Anne un des symboles du mot de code pris au hasard parmi n, puis un deuxième, pris au hasard parmi les $n-1$ restant, etc, jusqu'à ce qu'Anne

Reprenons l'exemple des CDs à ranger dans des boîtiers (page 84), et rappelons qu'un rangement est injectif s'il n'y a jamais deux CDs dans le même boîtier et surjectif si tous les boîtiers sont occupés. S'il y a moins de boîtiers que de CDs, il existe un boîtier avec au moins 2 CDs (item 1bis). S'il y a plus de boîtiers que de CDs, il y a au moins un boîtier vide (item 2bis).

dise "STOP". Anne connaît la position dans le mot de code du symbole reçu, et la valeur du symbole reçu. Le but du jeu est de décoder le message en recevant un nombre minimum de symboles. Quand Anne est sûre qu'il n'existe qu'un mot de code correspondant, elle dit STOP. Dans le pire des cas, de combien de symboles Anne a-t-elle besoin ?

Par exemple, supposons que le code soit celui de la Figure 11.1 ; card$(\mathcal{A}) = 2$*, les symboles sont des bits. Supposons que les positions des symboles choisis par le robot soient 1,2,6,7,3,4 et 5. Supposons que le message soit* 001*, donc le mot de code est* 0011100*. La Figure 11.3 montre la suite envoyée par le robot-code. Après avoir reçu 4 symboles, Anne ne sait pas si le mot de code est* 0000000 *ou* 0011100*. Par contre après avoir reçu le 5ième symbole, Anne sait que le seul mot de code possible est* 0011100 *; elle peut donc dire STOP et décider que le message est* 001*.*

En général, puisque la distance minimale de ce code est 3*, il est possible de corriger 2 effacements, donc Anne peut être sûre de pouvoir décoder après avoir reçu* 5 *symboles. En d'autres termes, pour pouvoir décoder le message original (qui comporte 3 symboles), il faut recevoir jusqu'à 5 symboles.*

Supposons maintenant que nous utilisions un code qui atteigne la borne de Singleton, et pour fixer les idées, supposons toujours que $k = 3$ *et* $n = 7$*. Le rendement du code est* $r = k/n$ *et la distance minimale du code est donc* $d_{\min}(\mathcal{C}) = n - k + 1 = 5$*. Le code est donc capable de corriger* 4 *effacements, et il suffit de recevoir 3 symboles pour décoder. Anne pourra toujours dira STOP après avoir reçu 3 symboles, quels qu'ils soient.*

En d'autre termes, pour un code $\mathcal{C}$ *sur un alphabet* $\mathcal{A}$ *qui atteint la borne de Singleton, et si* $k = \log_{card(\mathcal{A})} card(\mathcal{C})$ *est entier, il suffit de recevoir k symboles quelconques pour reconstruire le message original (qui est de longueur k symboles).*

position	symbole
1	0
2	0
6	0
7	0
3	1
4	1
5	1

FIGURE 11.3: La suite des symboles reçus par Anne quand le robot-code envoie les symboles du mot de code dans un ordre aléatoire. Le message est 001 et le code utilisé est celui de la Figure 11.1

Q. 92. Que donne la borne de Singleton pour le code code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 ?

12
Corps Finis et Espaces Vectoriels

Nous avons vu dans le chapitre précédent qu'il est important pour un code d'avoir une distance minimale aussi grande que possible, mais qu'en même temps, il peut être difficile de concevoir de tels codes, et même plus simplement de calculer la distance minimale d'un code. Pour résoudre ce problème, nous allons utiliser des codes linéaires sur des corps finis. Mais avant cela, il nous faut apprendre ce que ces termes recouvrent.

12.1 Corps Finis

Un corps commutatif est un ensemble dans lequel les 4 opérations d'addition, multiplication, soustraction et division fonctionnent comme nous en avons l'habitude quand nous utilisons les nombres réels ($\mathbb{R}$) ou complexes ($\mathbb{C}$). Plus précisément :

Définition 12.1 *Soit $(\mathcal{K}, +, \cdot)$ un ensemble muni de deux opérations binaires notées $+$ et $\cdot$. Nous disons que c'est un corps commutatif (Ang. field) si*

1. *L'addition fait de $\mathcal{K}$ un groupe commutatif. Son élément neutre est noté 0.*
2. *La multiplication fait de l'ensemble $\mathcal{K}$ privé de 0 un groupe commutatif. En particulier, tous les éléments sauf 0 sont inversibles. L'élément neutre de la multiplication est noté 1.*
3. *La multiplication est distributive par rapport à l'addition : $a \cdot (x + y) = a \cdot x + a \cdot y$ pour tous $a, x, y \in \mathcal{K}$.*

$(\mathcal{K}, +, \cdot)$ est un corps *non commutatif* s'il vérifie toutes les propriétés de corps commutatif sauf une : la multiplication n'est pas commutative. Tous les corps finis sont commutatifs, donc il n'est pas nécessaire de préciser "commutatif" quand nous parlons d'un corps fini.

Par contre il existe des corps infinis non commutatifs, par exemple le corps des quaternions utilisé en infographie.

Les corps finis sont aussi appelés corps de Galois, en l'honneur d'Evariste Galois, qui a lancé les bases de la théorie des corps finis (et bien plus) durant les deux semaines avant le duel qui a mis fin à sa vie à l'âge de 21 ans (elle s'appelait Stéphanie).

Les ensembles $\mathbb{R}$ et $\mathbb{C}$ sont des corps commutatifs infinis. Nous nous intéressons aux corps commutatifs finis, qui peuvent être utilisés comme alphabet par un code correcteur ou détecteur.

Exemple 12.1 ($\mathbb{Z}/p\mathbb{Z}$) *Nous savons (Théorème 8.6) que si p est un nombre premier, tous les éléments de $\mathbb{Z}/p\mathbb{Z}$ sauf $[0]_p$ sont inversibles. Il est facile de voir que cela entraîne que $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$ est un corps commutatif. C'est notre premier exemple de corps fini.*

Exemple 12.2 (Des Non-Corps) *$(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ n'est pas un corps si m n'est pas un nombre premier. En effet, il existe alors des éléments non nuls qui n'ont pas d'inverse (les diviseurs de m). Par exemple dans $\mathbb{Z}/6\mathbb{Z}$, $[3]_6$*

n'a pas d'inverse alors que $[3]_6 \neq [0]_6$. *$(\mathbb{Z}, +, \cdot)$ n'est pas un corps car les entiers non nuls autres que* 1 *et* -1 *n'ont pas d'inverse dans* $\mathbb{Z}$.

Q. 93. $(\mathbb{Q}, +, \cdot)$ est-il un corps commutatif ? ($\mathbb{Q}$, ensemble des nombres rationels, est l'ensemble des nombres réels qui peuvent s'écrire comme fraction de deux nombres entiers, positifs ou négatifs.

Tout ce que vous avez étudié en algèbre linéaire sur la résolution de systèmes d'équations linéaire reste valable si on utilise un corps fini au lieu de $\mathbb{R}$ ou $\mathbb{C}$. Illustrons ceci sur un exemple :

Exemple 12.3 (Système d'équations dans $\mathbb{Z}/7\mathbb{Z}$) *Considérons le système d'équations*

$$\begin{cases} [5]_7 x_1 + [4]_7 x_2 = [2]_7 \\ x_1 + [2]_7 x_2 = [0]_7 \end{cases}$$

où les inconnues x_1 et x_2 sont dans $\mathbb{Z}/7\mathbb{Z}$. Il est fastidieux de traîner une notation telle que $[3]_7$, aussi nous supprimons les crochets (il faudra simplement se rappeler dans quel corps nous sommes en train de faire des calculs). Nous écrivons donc le système ainsi :

$$\begin{cases} 5x_1 + 4x_2 = 2 \\ x_1 + 2x_2 = 0 \end{cases}$$

Pour le résoudre, nous procédons comme d'habitude. Par exemple, nous pouvons éliminer x_2 par combinaisons :

$$\begin{array}{rcrcll} 5x_1 & + & 4x_2 & = & 2 & \mid \cdot 1 \\ x_1 & + & 2x_2 & = & 0 & \mid \cdot (-2) \\ \hline 3x_1 & & & = & 2 & \end{array}$$

D'où

$$x_1 = 2 \cdot 3^{-1} = 2 \cdot 5 = 10 = 3$$

où nous avons utilisé le fait que $3^{-1} = 5$ (c'est à dire que l'inverse de $[3]_7$ est $[5]_7$). Nous obtenons x_2 à partir de la deuxième équation :

$$\begin{aligned} 2x_2 &= -x_1 = -3 = 4 \\ x_2 &= 4 \cdot 2^{-1} = 4 \cdot 4 = 16 = 2 \end{aligned}$$

Vérification :

$$\begin{cases} 5x_1 + 4x_2 = 15 + 8 = 1 + 1 = 2 \text{ - } OK \\ x_1 + 2x_2 = 3 + 4 = 7 = 0 \text{ - } OK \end{cases}$$

Nous avons vu que les calculs algébriques dans un corps fini sont semblables aux calculs usuels. Il y a quand même une différence, c'est le fait que, par exemple, $7 = 1+1+1+1+1+1+1 = 0$ dans $\mathbb{Z}/7\mathbb{Z}$. En général, cela est lié au concept de caractéristique :

Théorème et Définition 12.2 *Dans un corps fini, il existe un plus petit entier $p > 0$ tel que $p \cdot 1 = 0$, c'est à dire tel que*

$$\overbrace{1 + 1 + \ldots + 1}^{p \text{ fois}} = 0$$

Ce nombre est un nombre premier. Il est appelé la caractéristique du corps.

☺*Preuve :* Ce nombre existe car il est la période de 1 dans le groupe $(\mathcal{K}, +)$ (Théorème 9.4). Montrons par l'absurde qu'il est premier. En effet, sinon, nous pouvons factoriser $p = p_1 p_2$ avec p_1, p_2 entiers et $1 < p_1 < p$ et $1 < p_2 < p$. Soit $x_1 = p_1 \cdot 1 = \overbrace{1+1+...+1}^{p_1 \text{ fois}}$ et idem pour x_2. Alors $x_1 \neq 0$ car p est la période et $p_1 < p$; donc l'inverse x_1^{-1} de x_1 existe ; de même $x_2 \neq 0$. Or $x_1 x_2 = p1 = 0$, donc en multipliant x_1^{-1} nous obtenons : $x_2 = 0$, ce qui est une contradiction. ☺□

Enfin, citons sans démonstration le théorème suivant :

Théorème 12.1 *(1) Le cardinal d'un corps fini est une puissance de sa caractéristique.*

(2) Tous les corps finis de même cardinal sont isomorphes.

(3) Pour tout nombre premier p et tout entier $m \geq 1$ il existe un corps fini de cardinal p^m.

Ce théorème implique que le cardinal d'un corps fini est de la forme p^m où p est un nombre premier et m un entier. Par ailleurs, pour p premier et m entier donnés, il n'y a essentiellement qu'un seul corps fini, tous les autres s'en déduisent par re-nommage des éléments. Nous notons $\mathbb{F}_{p^m}$ le corps fini à p^m éléments. En particulier, pour $m = 1$, $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$.

Q. 94. Tous les groupes finis commutatifs de même cardinal sont-ils isomorphes ?

Q. 95. Le corps $(\mathbb{F}_4, +, \cdot)$ est-il isomorphe à $(\mathbb{Z}/4\mathbb{Z}, +, \cdot)$?

Q. 96. Existe-t-il un corps fini à 15 éléments ?

Exemple 12.4 ($\mathbb{F}_2$) *Le corps binaire, $\mathbb{F}_2$, est égal à $\mathbb{Z}/2\mathbb{Z}$, qui contient contient seulement deux éléments, 0 et 1. L'addition et la multiplication sont les opérations modulo 2.*

Il joue un rôle très important car nous pouvons interpréter les deux éléments 0 et 1 comme des valeurs logiques. L'addition correspond à l'opération `xor`, tandis que la multiplication correspond à `and`.

$+$ ou `xor`	0	1
0	0	1
1	1	0

$\cdot$ ou `and`	0	1
0	0	0
1	0	1

Le corps $\mathbb{F}_2$ a aussi une propriété intéressante. Si $a \in \mathbb{F}_2$, alors $a + a = 0$ et $a = -a$. L'addition et la soustraction sont les mêmes et donc les signes ne jouent pas d'importance. Dans $\mathbb{F}_2$, les erreurs de signe n'existent pas !

Exemple 12.5 ($\mathbb{F}_4$) *Le corps $\mathbb{F}_4$ existe car $4 = 2^2$ a un seul facteur premier ; il est constitué de 4 éléments, dont l'élément neutre pour l'addition (noté 0) et l'élément neutre pour la multiplication (noté 1). Soient a et b les deux autres éléments.*

Nous savons que ce corps existe, et nous pouvons en déduire ses tables d'addition et de multiplication. La caractéristique de $\mathbb{F}_4$ est 2 donc $1 + 1 = 0$ donc plus généralement $2x = 0$ pour tout $x \in \mathbb{F}_4$ (en effet $x + x = 1 \cdot x + 1 \cdot x = (1 + 1) \cdot x = 0x = 0$). Donc la table d'addition

d'addition de $\mathbb{F}_4$ est de la forme

$+$	0	1	a	b
0	0	1	a	b
1	1	0		
a	a		0	
b	b			0

Q. 97. Prouvez que dans la table d'un groupe commutatif $(G, \star)$ chaque élément doit se trouver une fois et une seule dans chaque ligne et chaque colonne.

Pour boucher les trous, observons que chaque élément doit se trouver une fois et une seule sur chaque ligne et chaque colonne de la table. En considérant la colonne de a, il vient que $a+1$ vaut 1 ou b ; mais 1 n'est pas possible à cause de la ligne de 1. Donc $a+1=b$. En continuant de la sorte on obtient la table d'addition de $\mathbb{F}_4$. La table de multiplication s'obient avec le même raisonnement (appliqué au groupe $\mathbb{F}_4^* = \{1, a, b\}$ muni de la multiplication) et en remarquant que $0x = 0$ pour tout x. On obtient ainsi les tables de $\mathbb{F}_4$:

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

C'est l'unique corps à 4 éléments, à isomorphisme près. Une représentation alternative s'obtient par la correspondance $0 \mapsto 00$,$1 \mapsto 11$, $a \mapsto 01$, $b \mapsto 10$, ce qui donne les tables suivantes :

$+$	00	11	01	10
00	00	11	01	10
11	11	00	10	01
01	01	10	00	11
10	10	01	11	00

$\cdot$	00	11	01	10
00	00	00	00	00
11	00	11	01	10
01	00	01	10	11
10	00	10	11	01

ou encore par la correspondance $0 \mapsto 00$,$1 \mapsto 01$, $a \mapsto 10$, $b \mapsto 11$, ce qui donne les tables suivantes :

$+$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$\cdot$	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Nous avons ainsi obtenu trois représentations différentes du corps $\mathbb{F}_4$; bien sûr, elles sont toutes trois isomorphes.

La dernière représentation est celle qui est le plus ouvent utilisée car elle correspond à la construction de $\mathbb{F}_4$ avec des polynômes, comme en Section 14.3.

Sous les deux dernières formes, on voit que l'addition dans $\mathbb{F}_4$ est identique à l'opération `xor` sur 2 bits. Par contre, la multiplication est entièrement nouvelle.

12.2 Espaces Vectoriels

Définition 12.3 *Soit $\mathcal{K}$ un corps commutatif et $(\mathcal{V}, +)$ un groupe commutatif, muni d'une opération binaire notée $+$. Supposons qu'une opération externe est définie sur $\mathcal{K}$ et $\mathcal{V}$, c'est à dire une application qui à*

$\lambda \in \mathcal{K}$ et $\vec{x} \in \mathcal{V}$ associe un élément, noté $\lambda\vec{x}$ de $\mathcal{V}$. Les éléments du corps commutatif sont appelés scalaires et les éléments de $\mathcal{V}$ sont appelés vecteurs. L'opération externe est appelée multiplication scalaire ou encore produit d'un vecteur par un scalaire. Nous disons que $\mathcal{V}$ muni de ces deux opérations est est un espace vectoriel sur le corps $\mathcal{K}$ si les propriétés suivantes sont vraies : pour tous scalaires λ, μ et vecteurs $\vec{u}, \vec{v}$,

- *associativité pour la multiplication scalaire : $\lambda(\mu\vec{v}) = (\lambda\mu)\vec{v}$*
- *identité : $1 \cdot \vec{v} = \vec{v}$*
- *distributivité : $\lambda(\vec{u} + \vec{v}) = \lambda\vec{u} + \lambda\vec{v}$ et $(a + b)\vec{u} = a\vec{u} + b\vec{u}$*

L'exemple le plus connu d'espace vectoriel est l'ensemble $\mathbb{R}^n$ des suites de n réels. De manière générale, si $\mathcal{K}$ est un corps commutatif, l'ensemble $\mathcal{K}^n$ des suites de n éléments de $\mathcal{K}$ (aussi appelés mots de longueur n) est un espace vectoriel pour les opérations

$$\begin{aligned}(x_1, ..., x_n) + (y_1, ..., y_n) &= (x_1 + y_2, ..., x_n + y_n)\\ \lambda(x_1, ..., x_n) &= (\lambda x_1, ..., \lambda x_n)\end{aligned}$$

Dans ce module, nous allons utiliser les espaces vectoriels $\mathcal{V} = \mathcal{K}^n$ où $\mathcal{K}$ est un corps fini.

Un sous-ensemble $\mathcal{S}$ de $\mathcal{V}$ est un sous-espace vectoriel, s'il est aussi un espace vectoriel sur $\mathcal{K}$. Il est facile de voir que cela est équivalent à dire que $\mathcal{S}$ est stable pour les deux opérations, c'est à dire que

$$\lambda\vec{u} \in \mathcal{S} \text{ et } \vec{u} + \vec{v} \in \mathcal{S} \text{ pour tous } \lambda \in \mathcal{K} \text{ et } \vec{u}, \vec{v} \in \mathcal{S}$$

Dans le contexte d'espace vectoriel, nous mettons en général une flèche sur les vecteurs et utilisons la notation en ligne $\vec{x} = (x_1, ..., x_n)$ (il existe aussi la notation en colonne, que nous n'utilisons pas ici).

L'élément neutre de l'addition est noté $\vec{0}$; ainsi dans $\mathcal{K}^n$, $\vec{0} = (0, ..., 0)$.

Les règles habituelles de manipulation de signes jouent dans tout espace vectoriel. En particulier :

$$\begin{aligned}0\vec{x} &= \vec{0}\\ (-\lambda)\vec{x} &= -(\lambda\vec{x})\end{aligned}$$

et enfin il n'y a pas de "diviseur de zéro", c'est à dire que

$$\lambda\vec{x} = \vec{0} \Rightarrow (\lambda = 0 \text{ ou } \vec{x} = \vec{0})$$

Exemple 12.6 *Dans l'espace vectoriel $\mathbb{F}_7^3$, considérons l'ensemble $\mathcal{S}$ des vecteurs de la forme $(u, 3u, 6u)$ avec $u \in \mathbb{F}_7$. $\mathcal{S}$ est un sous-espace vectoriel car, pour tous $\lambda, u, v \in \mathbb{F}_7$:*

$$\begin{aligned}\lambda(u, 3u, 6u) = (\lambda u, 3(\lambda u), 6(\lambda u)) &\in \mathcal{S}\\ (u, 3u, 6u) + (v, 3v, 6v) = (u + v, 3(u + v), 6(u + v)) &\in \mathcal{S}\end{aligned}$$

Considérons aussi l'ensemble $\mathcal{S}'$ des vecteurs $\vec{x} = (x_1, x_2, x_3)$ qui satisfont la condition

$$x_1 + 4x_2 + 3x_3 = 0 \tag{12.1}$$

$\mathcal{S}'$ est aussi un sous-espace vectoriel car, si $\vec{x} = (x_1, x_2, x_3), \vec{y} = (y_1, y_2, y_3)$ sont dans $\mathcal{S}'$ alors

$$(x_1 + y_1) + 4(x_2 + y_2) + 3(x_3 + y_3) = 0$$

donc $\vec{x} + \vec{y} = (x_1 + y_1, x_2 + y_2, x_3 + y_3) \in \mathcal{S}'$. De même, pour tout $\lambda \in \mathbb{F}_7$:

$$(\lambda x_1) + 4(\lambda x_2) + 3(\lambda x_3) = 0$$

donc $\lambda\vec{x} = (\lambda x_1, \lambda x_2, \lambda x_3) \in \mathcal{S}'$.

Une combinaison linéaire de vecteurs $\vec{v}_i \in \mathcal{V}$, $i = 1...m$ est une somme de la forme

$$\vec{u} = \sum_{i=1}^{m} \lambda_i \vec{v}_i \tag{12.2}$$

où les coefficients λ_i sont des scalaires. On peut montrer que l'ensemble des vecteurs $\vec{u}$ engendrés par toutes les combinaisons linéaires de m vecteurs $\boldsymbol{v_i}$ forme un sous-espace vectoriel. On appelle ce sous-espace vectoriel l'espace *engendré* par les $\vec{v}_i$.

Q. 98. La suite formée d'un seul vecteur $\vec{a}$ est-elle linéairement indépendante ?

Les vecteurs $\vec{v}_i$ sont *linéairement indépendants* si et seulement si $\sum_{i=1}^{m} \lambda_i \vec{v}_i = \vec{0}$ entraîne que tous les coefficients λ_i sont nuls. Dans ce cas, une représentation telle que (12.2) est unique : il n'y a qu'une seule suite de coefficients (λ_i) qui permette d'écrire le vecteur $\vec{u}$ sous cette forme.

La suite de vecteurs $\vec{v}_i$, $i = 1, ..., m$ est une *base* de l'espace vectoriel $\mathcal{V}$ si les vecteurs sont linéairement indépendants et engendrent l'espace vectoriel $\mathcal{V}$. Cela est équivalent à dire que tout vecteur de $\mathcal{V}$ s'écrit de manière unique comme combinaison linéaire des $\vec{v}_i$. Les coefficients d'une telle combinaison s'appellent les *coordonnées* du vecteur relativement à cette base.

12.3 *Propriétés de la Dimension*

Si $\mathcal{V} = \mathcal{K}^n$ ou si $\mathcal{V}$ est un sous-espace vectoriel de $\mathcal{K}^n$, il possède des bases finies, et toutes les bases ont le même cardinal, appelé la *dimension* de l'espace vectoriel, notée $\dim(\mathcal{V})$.

La dimension de $\mathcal{K}^n$ est n.
Si $\mathcal{V}$ est de dimension n, tout sous-espace vectoriel $\mathcal{S}$ a une dimension $k \leq n$, et si $k = n$, alors $\mathcal{S} = \mathcal{V}$. Le sous-espace vectoriel $\mathcal{S} = \left\{\vec{0}\right\}$ est de dimension $k = 0$.

Le concept de dimension possède quelques propriétés intéressantes qui permettent de jongler entre les propriétés d'indépendance linéaire et de génération. Dans un espace vectoriel $\mathcal{V}$ de dimension n :

1. Si une suite de $n = \dim(\mathcal{V})$ vecteurs est linéairement indépendante, elle engendre $\mathcal{V}$ (donc c'est une base de $\mathcal{V}$) ;
2. si une suite de $n = \dim(\mathcal{V})$ vecteurs engendre $\mathcal{V}$, elle est linéairement indépendante (donc c'est une base de $\mathcal{V}$).
3. Une suite de $m > n$ vecteurs de $\mathcal{V}$ est nécessairement linéairement dépendante, et une suite de $m < n$ vecteurs ne peut pas engendrer $\mathcal{V}$.
4. Si $\mathcal{S}$ et $\mathcal{S}'$ sont des sous-espaces vectoriels de $\mathcal{V}$, et si $\mathcal{S} \subset \mathcal{S}'$ alors $\dim(\mathcal{S}) \leq \dim(\mathcal{S}')$.
5. Si $\mathcal{S}$ et $\mathcal{S}'$ sont des sous-espaces vectoriels de $\mathcal{V}$, si $\mathcal{S} \subset \mathcal{S}'$ et $\dim(\mathcal{S}) = \dim(\mathcal{S}')$ alors $\mathcal{S} = \mathcal{S}'$.

Exemple 12.7 *Dans $\mathcal{V} = \mathbb{F}_7^3$, la suite de vecteurs $((1,0,0),\ (0,1,0),\ (0,0,1))$ est linéairement indépendante, car toute combinaison linéaire est de la forme*

$$(\lambda_1 + \lambda_2 \cdot 0 + \lambda_3 \cdot 0,\quad \lambda_1 \cdot 0 + \lambda_2 + \lambda_3 \cdot 0,\quad \lambda_1 \cdot 0 + \lambda_2 \cdot 0 + \lambda_3) = (\lambda_1, \lambda_2, \lambda_3)$$

et les composantes ne peuvent être nulles que si $\lambda_1 = \lambda_2 = \lambda_3 = 0$.

Nous pouvons voir que ces trois vecteurs engendrent $\mathbb{F}_7^3$ car n'importe quel vecteur $\vec{u} = (u_1, u_2, u_3)$ peut être écrit comme leur combinaison linéaire en choisissant $\lambda_1 = u_1$, $\lambda_2 = u_2$ et $\lambda_3 = u_3$. Cette suite de vecteurs est donc une base de $\mathbb{F}_7^3$, ce qui signifie que $\mathbb{R}^3$ est de dimension 3.

Nous pouvons facilement voir que le rajout d'un quatrième vecteur à cette suite la rendrait linéairement dépendante, car ce quatrième vecteur peut toujours être écrit comme une combinaison linéaire des trois premiers. Par exemple,

$$(3,2,0) = 3 \cdot (1,0,0) + 2 \cdot (0,1,0).$$

Le sous-espace $\mathcal{S}$ de l'Exemple 12.6 est engendré par $\vec{x} = (1,3,6)$ car tout vecteur de $\mathcal{S}$ peut s'écrire $(u,3u,6u) = u\vec{v}$ avec $u \in \mathbb{F}_3$. la suite constituée du vecteur $\vec{v}$ tout seul est une suite linéairement indépendante car $\vec{v} \neq \vec{0}$. Donc $\dim(\mathcal{S}) = 1$ (on dit que c'est une droite vectorielle).

Q. 99. Quel est le cardinal de l'espace vectoriel $\mathcal{S}$ de l'Exemple 12.6 ?

Théorème 12.2 *Si $\mathcal{V}$ est un espace vectoriel de dimension n sur un corps fini $\mathcal{K}$, alors $\mathcal{V}$ est fini et $card(\mathcal{V}) = [card(\mathcal{K})]^n$.*

☺*Preuve :* Soit $\vec{v}_1, ..., \vec{v}_n$ une base de $\mathcal{V}$. Tout élément $\vec{x}$ de $\mathcal{V}$ s'écrit de manière unique $\vec{x} = \lambda_1\vec{v}_1 + ... + \lambda_n\vec{v}_n$ donc l'application

$$\begin{aligned} &\mathcal{K}^n \to \mathcal{V} \\ &(\lambda_1, ..., \lambda_n) \mapsto \lambda_1\vec{v}_1 + ... + \lambda_n\vec{v}_n \end{aligned}$$

est une bijection. Par le principe des boîtiers : $\text{card}(\mathcal{V}) = \text{card}(\mathcal{K}^n) = [\text{card}(\mathcal{K})]^n$. ☺□

12.4 *Equations Linéaire et Rang d'une Matrice*

Dans $\mathcal{V} = \mathcal{K}^n$, nous appelons *équation linéaire* une équation de la forme

$$a_1x_1 + ... + a_nx_n = 0 \tag{12.3}$$

où l'inconnue est $\vec{x} = (x_1, ..., x_n)$. Les scalaires $a_1, ..., a_n$ sont les *coefficients* de l'équation, et le vecteur $\vec{a} = (a_1, ..., a_n)$ est le *vecteur de coefficients* de l'équation. Nous donnons sans démonstration le résultat suivant :

Théorème 12.3 (Equations d'un Sous-Espace Vectoriel) *L'ensemble $\mathcal{S}$ des solutions dans $\mathcal{V} = \mathcal{K}^n$ de m équations linéaires est un sous-espace vectoriel. Soit r la dimension de l'espace vectoriel engendré par les vecteurs de coefficients. Alors la dimension de $\mathcal{S}$ est $n - r$.*

En particulier, si les vecteurs de coefficients sont linéairement indépendants, la dimension de $\mathcal{S}$ est $n - m$.

Réciproquement, soit $\mathcal{S}$ un sous-espace vectoriel de $\mathcal{V} = \mathcal{K}^n$, avec $\dim(\mathcal{S}) = k$. Il existe une suite de $(n - k)$ équations linéaires dont l'ensemble des solutions est $\mathcal{S}$, et dont les vecteurs de coefficients sont linéairement indépendants.

Le Théorème 12.3 est bien connu en géométrie classique : une droite ($k = 1$) du plan $\mathbb{R}^2$ est défini par $2 - 1 = 1$ équation. Une droite de l'espace $\mathbb{R}^3$ est défini par $3 - 1 = 2$ équations. Un plan ($k = 1$) de l'espace est défini par $3 - 2 = 1$ équation.

Le *rang* d'une matrice rectangulaire A à coefficients dans un corps commutatif est par définition la dimension du sous-espace vectoriel engendré par les lignes de A. Le nombre r du théorème précédent est donc le rang de la matrice obtenue en écrivant les coefficients des équations. Nous rappelons les résultats suivants d'algèbre linéaire. Pour une matrice rectangulaire A à coefficients dans un corps commutatif :

1. Le rang de A est égal à la dimension du sous-espace vectoriel engendré par les lignes de A ; il est aussi égal à la dimension du sous-espace vectoriel engendré par les colonnes de A.
2. Le rang de A est le maximum des rangs des matrices carrées extraites de A.
3. Le rang d'une matrice carrée triangulaire $n \times n$ dont tous les termes diagonaux sont non nuls est n.
4. Le rang r d'une matrice $m \times n$ est tel que $r \leq m$ et $r \leq n$. Si $r = m$ ou $r = n$ la matrice est dite de *rang maximal*.

Une *matrice extraite* de A est obtenue en supprimant certaines lignes et certaines colonnes. Par exemple dans $\mathbb{F}_7$ avec

$$A = \begin{pmatrix} 4 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

la matrice

$$A' = \begin{pmatrix} 4 & 0 \\ 1 & 1 \end{pmatrix}$$

est une matrice extraite, obtenue en supprimant la deuxième colonne. B est de rang 2 (item 3), donc A est de rang ≥ 2 (item 2). Or A est de rang ≤ 2 (item 4). Donc le rang de A est 2. A est de rang maximal.

Le rang peut être calculé par exemple en appliquant la méthode du pivot de Gauss.

Exemple 12.8 *Le sous-espace $\mathcal{S}$ de l'Exemple 12.6 est de dimension 1 dans $\mathcal{V} = \mathbb{F}_7^3$, donc nous pouvons trouver un système de $3 - 1 = 2$ équations linéaires dont l'ensemble des solutions est $\mathcal{S}$. Trouvons de telles équations. Par définition de $\mathcal{S}$, un vecteur (x_1, x_2, x_3) est élément de $\mathcal{S}$ si et seulement si il existe $u \in \mathbb{F}_7$ tel que*

$$\begin{cases} x_1 = u \\ x_2 = 3u \\ x_3 = 6u \end{cases} \tag{12.4}$$

Le système d'équations (12.4) est appelé système d'*équations paramétriques* de $\mathcal{S}$.

Eliminons le paramètre u : d'une part, si Eq.(12.4) est satisfaite alors :

$$\begin{cases} x_2 = 3x_1 \\ x_3 = 6x_1 \end{cases} \tag{12.5}$$

Réciproquement, si Eq.(12.5) est satisfaite, posons $u = x_1$ et Eq.(12.4) est satisfaite. Donc Eq.(12.5) est satisfaite si et seulement si $(x_1, x_2, x_3) \in \mathcal{S}$, en d'autres termes, $\mathcal{S}$ est l'ensemble des solutions du système d'équations (12.5), que nous pouvons écrire aussi (car les calculs sont dans $\mathbb{F}_7$) :

$$\begin{cases} 4x_1 + x_2 = 0 \\ x_1 + x_3 = 0 \end{cases} \tag{12.6}$$

La matrice des coefficients du système est

$$A = \begin{pmatrix} 4 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Puisque la dimension de $\mathcal{S}$ est 2, le rang de A doit être 2, ce que nous pouvons facilement vérifier en observant que la sous-matrice $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ est de rang 2 (car triangulaire de termes diagonaux non nuls).

Exemple 12.9 *Le sous-espace $\mathcal{S}'$ de l'Exemple 12.6 est défini par une équation (Eq.(12.1)), dont le vecteur de coefficients est $\vec{a} = (1, 4, 3)$. Ce vecteur est non nul donc (Question 98) la suite constituée de $\vec{a}$ est linéairement indépendante, donc la dimension de $\mathcal{S}'$ est $k = 2$.*

Nous allons maintenant trouver une base de $\mathcal{S}'$. Nous savons que $\mathcal{S}'$ est de dimension 2, donc il suffit de trouver 2 vecteurs qui l'engendrent.

Q. 100. Combien y a-t'il de solutions à l'équation $x_1 + [4]_7 x_2 + [3]_7 x_3 = [0]_7$, où l'inconnue est la suite (x_1, x_2, x_3) d'éléments de $\mathbb{Z}/7\mathbb{Z}$?

Nous en obtenons 2 *en fixant d'abord* $x_2 = 1, x_3 = 0$ *puis* $x_2 = 0, x_3 = 1$. *Nous obtenons ainsi par exemple :*

$$\vec{a} = (3, 1, 0)$$
$$\vec{b} = (4, 0, 1)$$

Montrons qu'ils engendrent $\mathcal{S}'$. *La matrice dont ces vecteurs sont les lignes est*

$$B = \begin{pmatrix} 3 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}$$

Elle est de rang 2 *car la sous-matrice* $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ *est de rang* 2 *(car triangulaire de termes diagonaux non nuls). Donc la dimension de l'espace vectoriel engendré par les lignes de* B, *c'est-à-dire par* $\vec{a}$ *et* $\vec{b}$, *est* 2. *Cet espace vectoriel est inclus dans* $\mathcal{S}'$ *(car* $\vec{a}$ *et* $\vec{b}$ *sont dans* $\mathcal{S}'$*), donc il est égal* $\mathcal{S}'$ *(car il est de même dimension que* $\mathcal{S}'$. *Donc* $\vec{a}$ *et* $\vec{b}$ *forment une base de* $\mathcal{S}'$.

13
Codes Linéaires

Nous pouvons maintenant introduire les codes correcteurs ou détecteurs linéaires, qui sont les codes utilisés dans les systèmes informatiques.

13.1 Code Linéaire

Définition 13.1 *Soit $\mathcal{C}$ un code en bloc de longueur n. Nous disons que $\mathcal{C}$ est un code linéaire si*

1. *L'alphabet du code est un corps fini $\mathcal{K}$.*
2. *Le code est un sous-espace vectoriel de $\mathcal{K}^n$.*

Puisqu'un code linéaire est un sous-espace vectoriel, il a une dimension, que nous noterons souvent k. Le nombre de mots de code est $\text{card}(\mathcal{K})^k$ (Théorème 12.2), donc le rendement d'un code linéaire de dimension k est $\frac{k}{n}$.

Pour un code linéaire, la borne de Singleton prend donc la forme $d_{\min}(\mathcal{C}) \leq n - k + 1$.

Exemple 13.1 (Petit Code de la Figure 11.1 en page 91) *L'alphabet est $\mathcal{A} = \mathcal{K} = \mathbb{F}_2$ qui est un corps. Le code comporte huit mots de code, donnés dans la colonne de droite de la Figure 11.1. Soient $\vec{v}_0 = \vec{0}, \vec{v}_1 = (0,0,1,1,1,0,0), ..., \vec{v}_7 = (1,0,1,0,0,1,1)$ les huit mots de code. Nous avons*

$$\begin{aligned} \vec{v}_4 &= \vec{v}_1 + \vec{v}_2 \\ \vec{v}_5 &= \vec{v}_1 + \vec{v}_3 \\ \vec{v}_6 &= \vec{v}_2 + \vec{v}_3 \\ \vec{v}_7 &= \vec{v}_1 + \vec{v}_2 + \vec{v}_3 \end{aligned}$$

Selon le contexte, nous notons un élément de $\mathbb{F}_2^n$ soit sous la forme 0011100, soit sous la forme $(0,0,1,1,1,0,0)$. Les deux notations sont synonymes.

Il s'en suit que le code $\mathcal{C}$ est l'ensemble de toutes les combinaisons linéaires de et $\vec{v}_1, \vec{v}_2$ et $\vec{v}_3$ (en effet, une telle combinaison est de la forme $\lambda_1\vec{v}_1 + \lambda_2\vec{v}_2 + \lambda_3\vec{v}_3$ avec $\lambda_i = 0$ ou 1). Donc $\mathcal{C}$ est un sous-espace vectoriel, engendré par $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$. Donc $\mathcal{C}$ est un code linéaire, de dimension $k = 3$.

Q. 101. Quelle est la dimension de $\mathcal{C}$?

La première simplification importante qu'apporte un code linéaire est que sa distance minimale peut être calculée d'une façon plus efficace.

Q. 102. Le code obtenu en rajoutant à une suite de k chiffres décimaux les deux chiffres de contrôle de la procédure MOD 97-10 (Exemple 7.2, page 63) est-il liéaire ?

Théorème et Définition 13.2 *Si $\mathcal{K}$ est un corps fini, le poids de Hamming de $\vec{x} \in \mathcal{K}^n$ est le nombre de composantes non nulles, c'est à dire aussi $w(\vec{x}) \stackrel{\text{def}}{=} d(\vec{0}, \vec{x})$.*

La distance minimale d'un code linéaire $\mathcal{C}$ est égale à

$$d_{\min}(\mathcal{C}) = \min_{\vec{x}\in\mathcal{C};\vec{x}\neq\vec{0}} w(\vec{x}) \tag{13.1}$$

En d'autres termes, la distance minimale est le plus petit poids de Hamming d'un mot de code non nul.

Preuve : Soit $d^* = \min_{\vec{x}\in\mathcal{C};\vec{x}\neq\vec{0}} w(\vec{x})$ la valeur donnée dans le théorème ; d^* est la distance de $\vec{0}$ à un certain mot de code (un de ceux qui atteignent le minimum dans la formule (13.1)). Comme $\vec{0}$ est un mot de code, $d_{\min}(\mathcal{C}) \leq d^*$.

Réciproquement, soient $\vec{x}^*$ et $\vec{y}^*$ deux mots de code qui atteignent le minimum dans la définition de $d_{\min}(\mathcal{C})$, c'est à dire $d_{\min}(\mathcal{C}) = d(\vec{x}^*, \vec{y}^*)$. Comme le code est linéaire, $-\vec{y}^* = (-1)\cdot\vec{y}^*$ est aussi un mot de code, et donc $\vec{x}^* - \vec{y}^*$ est aussi un mot de code. Or $d(\vec{x}^*, \vec{y}^*) = w(\vec{y}^* - \vec{x}^*)$, donc $d_{\min}(\mathcal{C}) \geq d^*$. Donc finalement $d^* = d_{\min}(\mathcal{C})$. □

Exemple 13.2 (Petit Code de la Figure 11.1) *En inspectant les 7 mots de code non nuls de la colonne de droite de la Figure 11.1, nous voyons que les poids de Hamming sont $3, 5, 4, 4, 3, 5$ et 4, donc la distance minimale du code est 3.*

Voici deux autres exemples de codes parfois utilisés en pratique.

Exemple 13.3 ***Code de parité*** $(n, n-1)$**.** *L'alphabet est $\mathbb{F}_2$, les messages sont des suites de $n-1$ bits. Les mots de code sont obtenus en ajoutant aux messages un seul bit de contrôle, qui est tel que le nombre de 1 dans le mot de code soit pair. Par exemple, $\vec{x} = (1,0,1,1,0,0,1)$ devient $\vec{y} = (1,0,1,1,0,0,1,0)$. Les mots de codes sont donc des suites de bits $(y_1, ..., y_n)$ tels que*

$$y_1 + ... + y_n = 0 \tag{13.2}$$

(où l'addition est dans $\mathbb{F}_2$, c'est à dire c'est l'opération `xor`). Le code est donc un sous-espace vectoriel de $\mathbb{F}_2^n$, c'est le sous-espace défini par l'équation (13.2). Il est donc de dimension $k = n-1$.

Sa distance minimale vaut 2, donc il peut détecter (mais pas corriger) les erreurs portant sur un bit. (En fait, ce code détecte tout nombre impair d'erreurs mais ne détecte aucune erreur portant sur un nombre pair de bits.)

Nous appelons ici "bit" un élément de $\mathbb{F}_2$.

Q. 103. Que donne la borne de Singleton pour ce code ?

Exemple 13.4 ***Code à répétition*** $(n, 1)$**.** *Ici, par contre, il n'y a qu'un seul bit d'information et $n-1$ bits de contrôle qui sont obtenus par répétition du bit d'information. Les mots de codes sont donc des suites de bits $(y_1, ..., y_n)$ tels que*

$$\begin{cases} y_2 = y_1 \\ ... \\ y_n = y_1 \end{cases} \tag{13.3}$$

C'est donc un code linéaire, de dimension $k = 1$.

Il n'y a en fait que deux mots de code, $\vec{0}$ et $(1,1,...,1)$. La Figure 11.2 montre les codes à répétition pour $n = 2$ et $n = 3$.

La distance de ce code vaut donc $d_{\min} = n$, ce qui permet de détecter toute erreur portant sur $n-1$ bits ou moins, et de corriger erreur portant sur moins de $(n-1)/2$ bits. La capacité correctrice/détectrice de ce code est très élevée, mais son rendement, $\frac{1}{n}$, est très faible.

Q. 104. Que donne la borne de Singleton pour ce code ?

13.2 Matrice Génératrice d'un Code Linéaire

Comme un code linéaire est un sous-espace vectoriel, un mot de code $\vec{x} \in \mathcal{C}$ peut être écrit d'une manière unique comme combinaison linéaire des vecteurs d'une base de $\mathcal{C}$. Nous allons écrire cela en termes matriciels.

13.2.1 Matrice Génératrice et Encodage

Définition 13.3 *Soit $\mathcal{C}$ un code linéaire sur le corps $\mathcal{K}$, de longueur n et dimension k. Soit $(\vec{v}_1,...,\vec{v}_k)$ une base de $\mathcal{C}$. La matrice obtenue en écrivant à la $i-$ième ligne le vecteur $\vec{v}_i$ est appelée une* matrice génératrice *du code.*

Exemple 13.5 (Petit Code de la Figure 11.1) *Nous avons vu dans l'Exemple 13.1 que les trois premiers vecteurs non nuls $\vec{v}_1 = (0,0,1,1,1,0,0)$, $\vec{v}_2 = (0,1,1,1,0,1,1)$ et $\vec{v}_3 = (1,1,1,0,1,0,0)$ constituent une base du code. Donc une matrice génératrice du code est*

$$G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \end{matrix} \qquad (13.4)$$

Les noms des vecteurs $\vec{v}_1$ etc écrits à droite de la matrice sont placés là dans un souci d'illustration ; ils ne font pas partie de la matrice.

Notons que nous pourrions aussi bien prendre comme base $(\vec{v}_3, \vec{v}_2, \vec{v}_1)$, ce qui donne une autre matrice génératrice :

$$G' = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \vec{v}_3 \\ \vec{v}_2 \\ \vec{v}_1 \end{matrix} \qquad (13.5)$$

qui diffère de G par une permutation des lignes. Il est facile de voir que $(\vec{v}_3 + \vec{v}_2, \vec{v}_2 + \vec{v}_1, \vec{v}_1)$ constitue aussi une base de $\mathcal{C}$. Une troisième matrice génératrice du code $\mathcal{C}$ est donc

Q. 105. Prouvez que $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$, avec $\vec{e}_1 = \vec{v}_3 + \vec{v}_2$, $\vec{e}_2 = \vec{v}_2 + \vec{v}_1$ et $\vec{e}_3 = \vec{v}_1$, constitue une base de $\mathcal{C}$.

$$G'' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \vec{v}_3 + \vec{v}_2 \\ \vec{v}_2 + \vec{v}_1 \\ \vec{v}_1 \end{matrix} \qquad (13.6)$$

Ici le corps est $\mathbb{F}_2$ et les additions de vecteurs se font donc modulo 2.

Puisqu'un code linéaire de dimension k possède $\text{card}(\mathcal{K})^k$ mots de code, il peut être utilisé pour encoder des suites de k symboles de $\mathcal{K}$. Chaque choix d'une matrice génératrice correspond à une méthode d'encodage, définie comme suit. Considérons la suite $\vec{e}^i = (0,..,0,1,0,...,0)$ (où le vecteur est de longueur k et le symbole 1 est dans la $i-$ième position). Cette suite est souvent appelée la

base canonique de $\mathcal{K}^k$. La méthode d'encodage encode $\vec{e}^i$ par la $i$ième ligne de la matrice génératrice. Ensuite, si $\vec{u} = (u_1, ..., u_k)$ est un mot quelconque à encoder, le mot de code correspondant $\vec{x}$ est obtenu par la formule

$$\vec{x} = \vec{u}\, G \tag{13.7}$$

Exemple 13.6 (Encodage, Petit Code de la Figure 11.1) *Supposons que nous utilisions la matrice G de l'Eq.(13.4) et calculons les encodages de quelques message. Pour le message* 100, *nous prenons* $\vec{u} = (1,0,0)$ *et obtenons le mot de code*

$$\vec{x} = (1,0,0) \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} = (0,0,1,1,1,0,0)$$

correspondant au message 0011100. *Nous avons obtenu la première ligne de la matrice, ce qui est normal puisque c'est ainsi que nous avons construit notre méthode d'encodage. Si le message à encoder est* 101, *nous mettons* $\vec{u} = (1,0,1)$ *et*

$$\vec{x} = (1,0,1) \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} = (1,1,0,1,0,0,0)$$

donc le mot de code est 1101000. *En faisant cela pour les* 6 *autres messages possibles, nous obtenons la même table d'encodage que dans la Figure 11.1.*

Par contre, si nous utilisons la matrice G' *de l'Eq.(13.5), nous obtenons une table d'encodage différente. Par exemple, le mot* 100 *est encodé par* 1110100 *au lieu de* 0011100 *précédemment.*

Q. 106. Existe-t-il un mot à encoder qui soit toujours encodé de la même façon, quelle que soit la matrice génératrice choisie ?

La *correction d'effacement* peut se faire à l'aide de la matrice génératrice, en résolvant un système d'équations linéaires, comme illustré sur l'exemple suivant.

Exemple 13.7 (Correction d'Effacement, Petit Code de la Figure 11.1) *Supposons que nous ayons reçu le mot* $\vec{y} = (0,?,1,1,?,0,0)$. *La distance minimale du code est* 3, *donc nous savons que nous pouvons corriger cet effacement. Pour cela, nous écrivons l'équation* $\vec{u}G = \vec{y}$ *en supprimant les lignes correspondant aux effacements. Nous obtenons le système*

$$\begin{aligned} u_3 &= 0 \\ u_1 + u_2 + u_3 &= 1 \\ u_1 + u_2 &= 1 \\ u_2 &= 0 \\ u_2 &= 0 \end{aligned}$$

C'est un système à 5 *équations et* 3 *inconnues ; il peut ne pas admettre de solution (cela arrive quand il y a eu à la fois des effacements et des erreurs). Ici il y a une solution unique :* $(u_1, u_2, u_3) = (1,0,0)$. *Le décodeur corrige l'effacement en déclarant que le mot transmis est* 100.

13.2.2 ⋆ Forme Systématique

Le sous-espace vectoriel engendré par les lignes de G reste inchangé si l'on permute ces lignes ou si l'on ajoute une ligne à une autre. Aussi, nous pouvons appliquer les opérations élémentaires sur les lignes de G dans le but de mettre G sous une forme plus simple à utiliser, par exemple comme la matrice G'' de l'Eq.(13.6).

Définition 13.4 *Une matrice G à k lignes et $n > k$ colonnes est dite sous forme systématique si*

$$G = [I_k \quad P] \tag{13.8}$$

où I_k est la matrice identité d'ordre k (une matrice dont les éléments de la diagonale sont égaux à 1 et tous les autres à 0) et P est une matrice de dimensions $k \times (n-k)$.

La matrice G'' de l'Eq.(13.6) est sous forme systématique. Cette forme présente des avantages pratiques. En effet, le mot de code $\vec{x}$ correspondant au message $\vec{u}$ est alors de la forme

$$\vec{x} = (u_1, u_2, ..., u_k, r_1, ..., r_{n-k})$$

c'est à dire que les symboles porteurs d'information $u_1, ..., u_k$ sont inchangés. L'encodage consiste donc à ajouter les symboles restant $r_1, ..., r_{n-k}$ (appelés symboles de contrôle), calculés à partir de la matrice P.

Exemple 13.8 (Forme Systématique, Petit Code de la Figure 11.1) *Supposons que nous utilisions la matrice génératrice G'' de l'Eq.(13.6). L'encodage consiste à ajouter au message $\vec{u} = (u_1, u_2, u_3)$ quatre bits de contrôle contrôle donnés par*

Ici le corps est $\mathbb{F}_2$ et les additions sont donc modulo 2.

$$\begin{aligned} r_1 &= u_1 + u_3 \\ r_2 &= u_1 + u_2 + u_3 \\ r_3 &= u_1 + u_2 \\ r_4 &= u_1 + u_2 \end{aligned}$$

En général, la théorie de l'élimination de Gauss montre qu'il est possible de mettre toute matrice G de rank k sous forme systématique par des opérations élémentaires consistant soit à permuter des lignes, soit des colonnes, soit à ajouter à une ligne un multiple d'une autre ligne, soit à multiplier une ligne par un élément non nul du corps.

La matrice génératrice d'un code linéaire de dimension k est forcément de rang k, puisque le rang est la dimension de l'espace vectoriel engendré par les colonnes.

Notons que toutes ces opérations sauf les permutations de colonnes ne changent pas le sous-espace vectoriel engendré par les lignes de la matrice, donc ne changent pas le code. Par contre, les permutations de colonnes changent le code (le changement correspond à permuter les composantes du mot de code). Mais le code reste essentiellement le même, seul l'ordre des symboles est éventuellement modifié ; en particulier le rendement et la distance minimale restent les mêmes. En résumé, quitte à modifier l'ordre des symboles du code, tout code linéaire peut être mis sous forme systématique.

Un code linéaire systématique présente l'avantage d'être facile à réaliser à l'aide de circuits logiques et de registres à décalages, car il ne faut stocker que la matrice P, qui est dans certains cas beaucoup plus petite que G.

13.3 Matrice de Contrôle et Syndrome

13.3.1 Matrice de Contrôle

Puisque un code linéaire est un sous-espace vectoriel de $\mathcal{K}^n$ de dimension k, il est possible, d'après le Théorème 12.3 de définir les mots de codes par $n-k$ équations linéaires :

Définition 13.5 *Soit $\mathcal{C}$ un code linéaire sur le corps $\mathcal{K}$, de longueur n et dimension k. Une matrice de contrôle du code est une matrice à $n-k$ lignes et n colonnes, dont les lignes sont les vecteurs de coefficients de $n-k$ équations linéaires qui définissent le sous-espace vectoriel $\mathcal{C}$ dans $\mathcal{K}^n$. Ces $n-k$ lignes sont nécessairement indépendantes.*

En écriture matricielle, H est une matrice de contrôle du code si l'équation

$$\vec{x}\, H^T = \vec{0} \tag{13.9}$$

définit les mots de code $\vec{x}$.

La notation H^T signifie la *transposée* de H. C'est la matrice obtenue en permutant lignes et colonnes.

Pour trouver une matrice de contrôle, il faut mettre le sous-espace vectoriel en équations.

Exemple 13.9 (Matrice de Contrôle, Petit Code de la Figure 11.1) *Supposons que nous utilisions la matrice G de l'Eq.(13.4). Il nous fait trouver 4 équations du sous-espace vectoriel $\mathcal{C}$. Soit $a_1x_1 + a_2x_2 + ... + a_7x_7 = 0$ une telle équation. Nous voulons trouver les coefficients $a_1, ..., a_7$. Pour cela nous exprimons que l'équation doit être vérifiée par les trois lignes de G, ce qui donne les conditions :*

$$\begin{aligned} a_3 + a_4 + a_5 &= 0 \\ a_2 + a_3 + a_4 + a_6 + a_7 &= 0 \\ a_1 + a_2 + a_3 + a_5 &= 0 \end{aligned}$$

Rappelons nous que nous sommes dans $\mathbb{F}_2$ donc

$$\begin{aligned} a_3 &= a_4 + a_5 \\ a_2 &= a_3 + a_4 + a_6 + a_7 \\ a_1 &= a_2 + a_3 + a_5 \end{aligned}$$

et nous avons un système sous forme triangulaire, qui permet de calculer a_1, a_2 et a_3 à partir de a_4, a_5, a_6, a_7. Nous faisons d'abord $(a_4, a_5, a_6, a_7) = (1,0,0,0)$ et obtenons un premier vecteur de coefficients $\vec{a} = (1,0,1,1,0,0,0)$. Ensuite nous faisons : $(a_4, a_5, a_6, a_7) = (0,1,0,0)$, puis $(a_4, a_5, a_6, a_7) = (0,0,1,0)$, et finalement $(a_4, a_5, a_6, a_7) = (0,0,0,1,)$. Nous obtenons ainsi quatre vecteurs de coefficients, qui sont nécessairement linéairement indépendants. Finalement, nous obtenons la matrice

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{13.10}$$

En d'autres termes, $\vec{x}$ est un mot de code si et seulement si

$$\begin{aligned} x_1 + x_3 + x_4 &= 0 \\ x_1 + x_2 + x_3 + x_5 &= 0 \\ x_1 + x_2 + x_6 &= 0 \\ x_1 + x_2 + x_7 &= 0 \end{aligned}$$

Ces équations sont appelées des équations de contrôle de parité.

13.3.2 Syndrome

La matrice de contrôle permet de détecter simplement des erreurs. Soit $\vec{x}$ le mot de code transmis et $\vec{y}$ le mot de code reçu. Le *syndrome* est par définition

$$\vec{s} = \vec{y}\, H^T$$

S'il n'y a pas d'erreur, le syndrome est égal à $\vec{0}$. Une méthode simple de *détection* d'erreur consiste donc à déclarer une erreur si le syndrome est non nul. Quelles sont les erreurs non détectées par cette méthode ?

Soit $\vec{e} = \vec{y} - \vec{x}$ l'erreur. Notons que $\vec{x}$ est un mot de code donc $\vec{e}\ H^T = \vec{y}\ H^T$. L'erreur $\vec{e}$ est non détectée par cette méthode si $\vec{e}\, H^T = \vec{0}$, c'est à dire si $\vec{e} \in \mathcal{C}$. Les erreurs non détectées sont donc celles qui correspondent à l'addition d'un mot de code lors de la transmission.

Comme tous les mots de code non nuls ont un poids de Hamming au moins égal à $d_{\min}(\mathcal{C})$, toutes les erreurs de poids inférieur à $d_{\min}(\mathcal{C})$ sont détectées, comme nous nous y attendons.

Q. 107. Est-il possible que cette méthode de détection d'erreur déclare une erreur alors qu'il n'y en a pas ?

Exemple 13.10 (Petit Code de la Figure 11.1) *Supposons que nous recevions le mot $\vec{y} = (1,0,1,0,1,1,1)$. En appliquant par exemple la matrice de contrôle H de l'Eq.(13.10), nous obtenons le syndrome $\vec{s} = (1,1,0,0)$, qui est non nul donc il y a une erreur.*

La *correction d'erreur* est plus compliquée, sauf exception. Une méthode générale, basée sur le Théorème 11.4, consiste à chercher une erreur $\vec{e}$ de poids inférieur à $\frac{d_{\min}(\mathcal{C})}{2}$, telle que $\vec{y} - \vec{e}$ soit un mot de code, c'est à dire telle que $\vec{e}\ H^T = \vec{y}\ H^T$. Le Théorème 11.4 garantit que si un tel vecteur d'erreur existe, il est unique. Quand $d_{\min}(\mathcal{C})$ est très petit, une recherche exhaustive est possible.

Exemple 13.11 (Correction d'erreur pour le code a répétition $(n,1)$) *Rappelons que ce code répète le même bit n fois et que sa distance minimale est $d_{\min}(\mathcal{C}) = n$. Supposons par exemple que $n = 6$ et que le mot de code reçu est $\vec{y} = (1,0,1,1,0,1)$; un vecteur d'erreur possible est obtenu en supposant que les symboles 0, qui sont en minorité sont erronés. Cela donne $\vec{e} = (0,1,0,0,1,0)$, qui est de poids $2 < \frac{d_{\min}(\mathcal{C})}{2} = 2.5$. Donc c'est l'unique vecteur d'erreur possible de poids inférieur à 2.5 et c'est celui que calcule la méthode de correction d'erreur proposée plus haut. Le mot de code original obtenu est $\vec{x} = (1,1,1,1,1,1)$ et le bit encodé est donc $u = 1$.*

De manière générale : pour corriger les erreurs dans un code à répétition il suffit de déterminer le bit qui se trouve en majorité. Cette méthode de décodage corrige toutes les erreurs de poids $< \frac{d_{\min}(\mathcal{C})}{2}$. Elle ne permet pas de corriger les erreurs de poids supérieur. Par exemple si le mot reçu est $\vec{y} = (1,0,0,0,0,0)$ et que le vecteur d'erreur est $(0,1,1,1,1,1)$, ce décodeur déclare que le bit encodé est $u = 0$, ce qui est faux, mais il est impossible de le savoir.

Il peut même arriver que ce décodeur ne fournisse pas de résultat, quand il n'y a pas de majorité, ce qui arrive quand n est pair et qu'il y a exactement $\frac{n}{2}$ erreurs, par exemple si le mot de code reçu est $(1,0,1,0,1,0)$.

13.3.3 ⋆ Matrice de Contrôle et Forme Systématique

Le calcul d'une matrice de contrôle est très facile si nous disposons d'une matrice génératrice sous forme systématique.

Théorème 13.1 *Si la matrice génératrice G est sous la forme systématique $G = [I_k \quad P]$ alors une matrice de contrôle est*

$$H = [-P^T \quad I_{n-k}]. \tag{13.11}$$

☺*Preuve :* Il nous fait montrer que l'ensemble $\mathcal{S}$ des solutions de $\vec{x}\ H^T = \vec{0}$, où $\vec{x} \in \mathcal{K}^n$, est le code $\mathcal{C}$. Pour cela nous montrons que (1) $\mathcal{S}$ contient $\mathcal{C}$ et (2) $\mathcal{S}$ et $\mathcal{C}$ ont même dimension. D'après la note de marge de la page 105, cela prouvera en effet que $\mathcal{S} = \mathcal{C}$.

(1) Soit $\vec{x}$ un mot de code quelconque ; il peut se mettre sous la forme $\vec{x} = \vec{u}G$ pour un certain $\vec{u} \in \mathcal{K}^k$. Notons que

$$GH^T = [I_k \quad P] \begin{bmatrix} -P \\ I_{n-k} \end{bmatrix} = -P + P = 0.$$

Donc $\vec{x}H^T = \vec{u}GH^T = \vec{u}0 = \vec{0}$. Donc $\vec{x} \in \mathcal{S}$.

(2) La dimension de $\mathcal{C}$ est k (par définition de k). Les lignes de la matrice H sont linéairement indépendantes à cause du bloc I_{n-k}. D'après le Théorème 12.3, la dimension de $\mathcal{S}$ est donc $n - (n-k) = k$. ☺□

Exemple 13.12 (Petit Code de la Figure 11.1) *Supposons que nous utilisions la matrice G'' de l'Eq.(13.6). Elle est sous forme systématique avec*

$$P = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

La matrice de contrôle correspondante est

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Notons que c'est la même matrice que celle que nous avons trouvé en Eq.(13.10) ; ceci est par hasard. En effet, en général, une matrice de contrôle n'est pas unique et différentes méthodes de calcul donnent des matrices de contrôle différentes. Par contre, bien sûr, si on obtient un syndrome nul avec une matrice de contrôle, il sera nul avec toute autre matrice de contrôle.

14
Codes de Reed-Solomon

A l'exception du code de répétition dont le rendement est très faible, les meilleurs codes que nous ayons vu dans ce module ont une distance minimale de 3. C'est assez bon pour montrer l'idée principale du codage mais n'est pas suffisant dans beaucoup d'applications réelles. Sur un disque, par exemple, nous pouvons faire un trou de 1mm de diamètre et le code utilisé est encore assez puissant pour retrouver l'information. Comme un bit occupe a peu près 10^{-6} mètre sur le disque, un tel trou couvre des milliers de bits sur chaque piste. Comment pouvons nous construire des codes ayant une grande distance minimale ?

Le problème est difficile pour les codes binaires, c'est-à-dire, pour les codes sur $\mathbb{F}_2$. Mais si nous considérons un alphabet plus grand, une solution optimale a été proposée dans les années 1950 par Irvine Reed et Gus Solomon. Ces codes sont appelés les codes de Reed-Solomon en leur honneur et jusqu'à aujourd'hui ce sont les codes les plus utilisés. Chaque seconde, il y a des centaines de millions des codes Reed-Solomon en fonctionnement, et ils assurent que la plupart de nos communications soient essentiellement sans erreurs.

Même si les codes Reed-Solomon sont puissants, leur définition est simple ; elle est basée sur des évaluations de polynômes.

14.1 Définition

Rappelons d'abord ce qu'est un polynôme, puis nous allons introduire une notation très utile pour la suite.

Définition 14.1 (Polynômes) *Soit $\mathcal{K}$ un corps fini. Un polynôme P à coefficients dans $\mathcal{K}$ est une application $\mathcal{K} \to \mathcal{K}$ de la forme*

$$X \mapsto P(X) = a_1 + a_2 X + ... + a_{m+1} X^m$$

où $a_1, ..., a_{m+1}$ sont des éléments de $\mathcal{K}$.

Le degré du polynôme est la plus grande puissance affectée d'un coefficient non nul.

Le polynôme 0 est de degré indéfini.

Pour toute suite $\vec{u} = (u_1, ..., u_k) \in \mathcal{K}^k$ de k éléments de $\mathcal{K}$ nous appelons $P_{\vec{u}}$ le polynôme dont les coefficients sont $u_1, ..., u_k$, par ordre de puissances croissantes. En d'autre termes

Par exemple, avec $\mathcal{K} = \mathbb{F}_5$, $P_{243}(X) = 2 + 4X + 3X^2$ et $P_{0000432}(X) = 4X^4 + 3X^5 + 2X^6$.

$$P_{\vec{u}}(X) \stackrel{\text{def}}{=} u_1 + u_2 X + u_3 X^2 + ... + u_k X^{k-1} \tag{14.1}$$

et donc $P_{\vec{u}}$ est un polynôme de degré $k-1$. Nous pouvons maintenant définir les codes de Reed-Solomon.

Définition 14.2 (Reed-Solomon) *Soient n et k des entiers avec $1 \leq k \leq n$. Un code de Reed Solomon de paramètres (n,k) est défini comme suit :*

1. *L'alphabet est un corps fini $\mathcal{K}$ de cardinal $\geq n$.*
2. *Choisissons n éléments distincts de $\mathcal{K}$, $a_1, a_2 ..., a_n$. Une suite de k symboles $\vec{u} = (u_1, ..., u_k) \in \mathcal{K}^k$ est encodée en la suite de n symboles $\vec{x} = (x_1, ..., x_n) \in \mathcal{K}^n$ définie par*

$$x_i = P_{\vec{u}}(a_i) \quad \text{pour } i = 1...n \tag{14.2}$$

Le code de Reed-Solomon $\mathcal{C}$ est l'ensemble de tous les encodages $\vec{x}$ possibles, pour tous les $\vec{u} \in \mathcal{K}^k$. C'est donc un code en bloc de longueur n.

Exemple 14.1 (Code de Reed-Solomon sur $\mathbb{F}_5$) *Construisons un code de Reed-Solomon sur $\mathbb{F}_5$. Nous pouvons choisir n'importe quelle longueur n entre 1 et $5 = card(\mathbb{F}_5)$. Choisissons le maximum possible, $n = 5$. Il nous faut aussi choisir 5 éléments de $\mathbb{F}_5$, mais ici nous n'avons pas le choix, il faut les prendre tous : $a_1 = 0, a_2 = 1, a_3 = 2, a_4 = 3$ et $a_5 = 4$.*

Calculons les encodages. Par exemple, si le message est $\vec{u} = (0,0)$, le polynôme à évaluer est $P_{\vec{0}} = 0$ donc $P_{\vec{0}}(a_1) = ... = P_{\vec{0}}(a_5) = 0$, et le mot de code correspondant est $\vec{x} = (0,0,0,0,0)$.

Si $\vec{u} = (3,2)$, alors $P_{\vec{u}}(X) = P_{32}(X) = 3 + 2X$ et donc

$$\begin{aligned} P_{\vec{u}}(a_1) &= P_{\vec{u}}(0) = 3 \\ P_{\vec{u}}(a_2) &= P_{\vec{u}}(1) = 0 \\ P_{\vec{u}}(a_3) &= P_{\vec{u}}(2) = 2 \\ P_{\vec{u}}(a_4) &= P_{\vec{u}}(3) = 4 \\ P_{\vec{u}}(a_5) &= P_{\vec{u}}(4) = 1 \end{aligned}$$

donc $\vec{x} = (3,0,2,4,1)$. En faisant cela pour les 25 suites de $k = 2$ éléments de $\mathbb{F}_5$ nous obtenons la table d'encodage de la Figure 14.1.

$\vec{u}$	$\mathbb{P}_{\vec{u}}(X)$	$\vec{x}$
00	0	00000
01	X	01234
02	$2X$	02413
03	$3X$	03142
04	$4X$	04321
10	1	11111
11	$1+X$	12340
12	$1+2X$	13024
13	$1+3X$	14203
14	$1+4X$	10432
20	2	22222
21	$2+X$	23401
22	$2+2X$	24130
23	$2+3X$	20314
24	$2+4X$	21043
30	3	33333
31	$3+X$	34012
32	$3+2X$	30241
33	$3+3X$	31420
34	$3+4X$	32104
40	4	44444
41	$4+X$	40123
42	$4+2X$	41302
43	$4+3X$	42031
44	$4+4X$	43210

FIGURE 14.1: Table d'encodage $\vec{u} \mapsto \vec{x}$ d'un code de Reed-Solomon de longueur $n = 5$ et de dimension $k = 2$.

14.2 Propriétés

Nous allons maintenant voir que les codes de Reed-Solomon jouissent de propriétés remarquables. Pour cela nous avons besoin d'un résultat fondamental sur les polynômes, qui dit qu'un polynôme a un nombre de racines inférieur ou égal à son degré.

Théorème 14.1 (Racines d'un Polynôme) *Soit $\vec{u} \in \mathcal{K}^k$ où $\mathcal{K}$ est un corps commutatif. Le polynôme $P_{\vec{u}}$ est de degré $k-1$. S'il existe k éléments tous distincts $a_1, ..., a_k$ de $\mathcal{K}$ tels que $P_{\vec{u}}(a_i) = 0$, alors $\vec{u} = \vec{0}$.*

☺*Preuve :* Nous faisons la preuve quand $\mathcal{K}$ est un corps fini, ce qui suffit à notre propos (mais le théorème est vrai même si le corps commutatif $\mathcal{K}$ est infini). Soit ψ l'application $\mathcal{K}^k \to \mathcal{K}^k$ qui à

ψ est en fait l'encodage d'un code de Reed-Solomon de paramètres (k,k).

$\vec{v} \in \mathcal{K}^k$ quelconque associe $(P_{\vec{v}}(a_1), ..., P_{\vec{v}}(a_k))$. L'hypothèse est que $\psi(\vec{u}) = \vec{0}$ et nous voulons montrer que $\vec{u} = \vec{0}$.

Pour cela nous allons montrer que ψ est *injective*. Par le principe des tiroirs, comme les ensembles d'arrivée et de départ de ψ sont finis et ont même cardinal, il suffit de montrer que ψ est *surjective*, ce que nous faisons maintenant.

Soit $(x_1, ..., x_k) \in \mathcal{K}^k$ quelconque. Cherchons un polynôme P de degré $\leq k-1$ tel que $(P(a_1) = x_1, ..., P(a_k) = x_k)$. Un tel problème est connu sous le nom d'"interpolation" : les valeurs x_i et les points d'évaluation a_i sont connus et il faut trouver un polynôme qui donne ces valeurs. Sa solution est connue, il suffit d'utiliser le polynôme d'interpolation de *Lagrange*. Soit Q_i, $i = 1, ...k$, le polynôme défini par

$$Q_i(X) = \frac{(X-a_1)...(X-a_{i-1})(X-a_{i+1})...(X-a_k)}{(a_i-a_1)...(a_i-a_{i-1})(a_i-a_{i+1})...(a_i-a_k)} \tag{14.3}$$

C'est un polynôme de degré $\leq k-1$ (produit de $k-1$ termes de degré 1) et il a la propriété que $Q_i(a_i) = 1$ et $Q_i(a_j) = 0$ pour $i \neq j$. Soit maintenant P le polynôme défini par

Le degré ici est par rapport à la variable X

$$P(X) = x_1 Q_1(X) + ... + x_k Q_k(X)$$

de sorte que P est un polynôme de degré $\leq k-1$ (comme somme de k polynômes de degré ≤ 1) et $P(a_i) = x_i$ pour $i = 1...k$. C'est donc une solution à notre problème d'interpolation.

Soit alors $v_1 =$ le coefficient de degré 0 de P, etc..., $v_k =$ le coefficient de degré $k-1$ de P, de sorte que $P = P_{\vec{v}}$ et donc $\psi(\vec{v}) = \vec{x}$. Nous avons donc montré que ψ est surjective.

Donc ψ est injective. Or $\psi(\vec{0}) = \vec{0}$. Donc $\vec{u}$ et $\vec{0}$ ont la même image par ψ, donc $\vec{u} = \vec{0}$. ☺□

14.2.1 *Linéarité*

Les codes de Reed-Solomon sont en fait des codes linéaires :

Théorème 14.2 *Un code de Reed Solomon de paramètres (n, k) est un code en bloc linéaire de taille n et de dimension k.*

Preuve : (1) Montrons d'abord la linéarité. Soient $\vec{x}$ et $\vec{x}'$ deux mots de codes. Donc il existe $\vec{u}, \vec{u}'$ tels que

$$P_{\vec{u}}(a_j) = x_j$$
$$P_{\vec{u}'}(a_j) = x'_j$$

pour $j = 1, ..., n$. Il est immédiat de voir que $P_{\vec{u}+\vec{u}'}(X) = P_{\vec{u}}(X) + P_{\vec{u}'}(X)$ donc $P_{\vec{u}+\vec{u}'}(a_j) = P_{\vec{u}}(a_j) + P_{\vec{u}'}(a_j)$, en d'autres termes

$$x_j + x'_j = P_{\vec{u}+\vec{u}'}(a_j)$$

donc le mot $\vec{x} + \vec{x}'$, dont le terme générique est $x_j + x'_j$, est un mot de code (correspondant à $\vec{u} + \vec{u}'$). De la même façon, pour tout

$\lambda \in \mathcal{K}$, le mot $\lambda\vec{x}$ est un mot de code (correspondant à $\lambda\vec{u}$). Donc $\mathcal{C}$ est un sous-espace vectoriel de $\mathcal{K}^n$.
(2) Il nous reste à montrer que $\dim(\mathcal{C}) = k$. Pour cela nous allons trouver une base de $\mathcal{C}$ de cardinal k. Soit $(\vec{e}^i)$, $i = 1...k$ la base canonique de $\mathcal{K}^k$ et $\vec{x}^i$ le mot de code correspondant à $\vec{e}^i$. Montrons que $(\vec{x}^i)$, $i = 1...k$ est une base de $\mathcal{C}$. Tout message $\vec{u}$ peut s'écrire $\vec{u} = u_1\vec{e}^1 + ... + u_k\vec{e}^k$ et le mot de code correspondant est $\vec{x} = u_1\vec{x}^1 + ... + u_k\vec{x}^k$, donc $(\vec{x}^i)$, $i = 1...k$ engendre de $\mathcal{C}$.

Rappelons que $\vec{e}^i = (0, .., 0, 1, 0, ..., 0)$ où le vecteur est de longueur k et le symbole 1 est dans la $i$ième position. Le polynôme correspondant à $\vec{e}^i$ est X^{i-1} donc $\vec{x}^i_j = a_j^{i-1}$.

Montrons qu'elle est linéairement indépendante. Supposons qu'une combinaison linéaire soit nulle, c'est-à-dire $u_1\vec{x}^1 + ... + u_k\vec{x}^k = \vec{0}$. Donc le polynôme $P_{\vec{u}}$ vérifie $P_{\vec{u}}(a_j) = 0$ pour tous j, c'est à dire qu'il possède n racines distinctes. Comme $n > k$, $P_{\vec{u}}$ possède k racines distinctes. D'après le Théorème 14.1, $\vec{u} = \vec{0}$, ce qui montre que $(\vec{x}^i)$, $i = 1...k$ est linéairement indépendante. □

Exemple 14.2 (Code de la Figure 14.1) *Une matrice génératrice de ce code de Reed-Solomon sur $\mathbb{F}_5$ est obtenue en considérant les encodages de la base canonique de $\mathbb{F}_5^2$. La première ligne est obtenue est obtenue en prenant $\vec{u} = (1, 0)$, ce qui correspond au polynôme $P_{10} = 1$; la deuxième ligne est obtenue est obtenue en prenant $\vec{u} = (0, 1)$, ce qui correspond au polynôme $P_{01} = X$. Donc*

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ a_0 & a_1 & a_2 & a_3 & a_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \qquad (14.4)$$

Le mot de code correspondant au message $\vec{u} = (u_1, u_2)$ est

$$\vec{x} = (u_1, u_2)G$$

UNE MATRICE DE CONTRÔLE peut facilement être obtenue à l'aide des polynômes d'interpolation de Lagrange. En effet, soit $\vec{x}$ un mot de code. Le message $\vec{u}$ peut être obtenu comme la suite des coefficients du polynôme d'interpolation sur les k premières valeurs :

$$P_{\vec{u}}(X) = x_1Q_1(X) + ... + x_kQ_k(X)$$

où Q_i est défini en Eq.(14.3). Donc nécessairement, x_j pour $j > k$ est la valeur de ce polynôme en a_j, c'est à dire que

$$x_1Q_1(a_j) + ... + x_kQ_k(a_j) = x_j \text{ pour } j = k+1, ..., n \qquad (14.5)$$

ce qui donne un système de $n - k$ équations linéaires en $\vec{x}$. Réciproquement, si un vecteur $\vec{x}$ satisfait ces $n - k$ équations, il est nécessairement le mot de code correspondant au message $\vec{u}$. En d'autres termes, ces équations définissent le code $\mathcal{C}$ et nous pouvons les prendre pour obtenir une matrice de contrôle.

Exemple 14.3 (Code de la Figure 14.1) *Nous allons obtenir une matrice de contrôle en écrivant les équations (14.5). Les polynômes Q_1 et Q_2 sont*

$$Q_1(X) = \frac{X-1}{-1} = 4(X-1)$$
$$Q_2(X) = \frac{X}{1} = X$$

et les équations (14.5) sont

$$\begin{aligned} x_1Q_1(2) + x_2Q_2(2) &= x_3 \\ x_1Q_1(3) + x_2Q_2(3) &= x_4 \\ x_1Q_1(4) + x_2Q_2(4) &= x_5 \end{aligned}$$

donc une matrice de contrôle est

$$\begin{aligned} H &= \begin{pmatrix} -Q_1(2) & -Q_2(2) & 1 & 0 & 0 \\ -Q_1(3) & -Q_2(3) & 0 & 1 & 0 \\ -Q_1(4) & -Q_2(4) & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 3 & 1 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Q. 108. Comment pouvez vous vérifier que H est bien une matrice de contrôle compatible avec la matrice génératrice de l'Eq.(14.4) ?

14.2.2 *Optimalité*

Théorème 14.3 *Un code de Reed Solomon de paramètres (n, k) a pour distance minimale $d_{\min} = n - k + 1$. En d'autres termes, il atteint la borne de Singleton, et sa distance minimale est la plus grande possible pour un code en bloc de longueur n et de dimension k.*

☺*Preuve :* (1) Montrons par l'absurde que tout mot de code non nul a un poids de Hamming $> n - k$. Soit donc un mot de code $\vec{x}$ tel que $w(\vec{x}) \leq n - k$. Cela veut dire qu'au moins k des composantes de $\vec{x}$ sont nulles. Soit $\vec{u}$ le message correspondant au mot de code $\vec{x}$. Si la $j$ième composante de $\vec{x}$ est nulle, cela signifie que $P_{\vec{u}}(a_j) = 0$, donc le polynôme $P_{\vec{u}}$ possède au moins k racines. Ceci contredit le Théorème 14.1.
(2) Donc, d'après le Théorème 13.2, $d_{\min}(\mathcal{C}) > n - k$ donc $d_{\min}(\mathcal{C}) \geq n - k + 1$. Par la borne de Singleton, $d_{\min}(\mathcal{C}) \leq n - k + 1$, donc il y a égalité. ☺□

Q. 109. Dans l'Exemple 11.3, nous supposons avoir à disposition un code de longueur 7 et de dimension 3 et qui atteigne la borne de Singleton. Pouvez vous proposer un tel code ?

Exemple 14.4 (Correction d'Effacements) *Avec le code de la Figure 14.1, supposons que nous recevions le mot $\vec{y} = (??2?1)$. Il y a trois effacements. Comme $3 \leq d_{\min}(\mathcal{C}) - 1 = n - k = 3$, nous savons que nous pouvons reconstruire le mot transmis de façon unique. Soit $\vec{x}$ le mot transmis et $\vec{u}$ le message. Nous savons que $\vec{x} = \vec{u}G$. De plus, $\vec{x}$ et $\vec{y}$ coincident dans les positions 3 et 5. Nous avons donc le système d'équations*

$$\begin{cases} u_1 + 2u_2 = 2 \\ u_1 + 4u_2 = 1 \end{cases}$$

que nous pouvons facilement résoudre dans $\mathbb{F}_5$ pour obtenir $u_1 = 3$, $u_2 = 2$. Le message est donc 32. Nous pouvons aussi calculer le mot de code complet. Nous obtenons $\vec{x} = \vec{u}G = 30241$.

Le choix des éléments a_j du corps $\mathcal{K}$ est en théorie sans importance, car les propriétés que nous avons vues du code de Reed Solomon en sont indépendantes.

En pratique, cependant, il existe des choix qui rendent les calculs plus rapides. Le choix standard consiste à trouver tout d'abord un

élément spécial g du corps $\mathcal{K}$ tel que tout élément non nul de $\mathcal{K}$ soit une puissance de g (la période de g dans le groupe $(\mathcal{K}^*, \cdot)$ est donc égale à $\text{card}(\mathcal{K}) - 1$). Un tel élément existe toujours dans un corps fini, et est appelé un *générateur* de $\mathcal{K}$. Soit $m = \text{card}(\mathcal{K}) - 1$. L'application $(\mathbb{Z}/m\mathbb{Z}, +) \to (\mathcal{K}^*, \cdot)$, $[\ell]_m \mapsto g^\ell$ est alors un isomorphisme de groupes, dont l'application inverse est appelé un logarithme discret. Un logarithme discret transforme les multiplications et divisions dans le corps fini $\mathcal{K}$ en additions et soustractions modulo m.

On choisit alors $a_j = g^{j-1}$, pour $j = 1...n$. Le terme générique de la matrice génératrice est alors $G_{i,j} = g^{(i-1)(j-1)}$. Il existe des algorithmes très rapides, en particulier utilisant le logarithme discret, pour le calcul de l'encodage et de la correction d'effacement avec de telles matrices.

Par exemple, dans $\mathcal{K} = \mathbb{F}_5$, $g = 2$ est un générateur car ses puissances sont $\{2, 4, 3, 1\}$; la période de g est 4 donc l'ensemble des puissances de g est l'ensemble des 4 éléments non nuls de $\mathbb{F}_5$.

Q. 110. Le corps $\mathbb{F}_5$ possède-t-il un autre générateur ?

Q. 111. Prouver que l'application $[k]_4 \mapsto g^k$ avec $g = [2]_5$ est un isomorphisme des groupes $(\mathbb{F}_5^*, \cdot)$ et $(\mathbb{Z}/4\mathbb{Z}, +)$.

14.3 ⋆ *Le Corps* $\mathbb{F}_{256}$

Les codes de Reed-Solomon utilisent comme alphabet un corps fini. Il serait très pratique de pouvoir prendre comme alphabet $\mathcal{K} = \mathbb{Z}/256\mathbb{Z}$, car alors il y a un symbole par octet, ce qui est la quantité d'information fondamentale des systèmes numériques. Mais nous savons que $\mathbb{Z}/256\mathbb{Z}$ n'est pas un corps car 256 n'est pas un nombre premier. Heureusement, nous savons aussi qu'il existe un corps $\mathbb{F}_{256}$ à 256 éléments, car 256 est une puissance du nombre premier 2. Dans cette section nous allons décrire comment est construit ce corps.

Pour cela il nous faut définir la division des polynômes à coefficients dans un corps fini, qui est très semblable à la division euclidienne des entiers. Soient $a(X)$ et $b(X)$ deux polynômes à coefficients dans un corps commutatif, et supposons que les coefficients de $b(X)$ ne sont pas tous nuls. La *division selon les puissances décroissantes*, aussi appelée *division longue*, est définie de manière semblable à celle des entiers. Nous pouvons trouver des polynômes $q(X)$ et $r(X)$, uniques, appelés quotient et reste, tels que

$$a(X) = q(X)b(X) + r(X) \text{ avec } \deg r(X) < \deg b(X)$$

$\deg r(X)$ est le degré du polynôme $r(X)$.

Nous ne prouvons pas ce fait, mais remarquons seulement que la condition importante est que les coefficients des polynômes appartiennent à un corps commutatif, par exemple $\mathbb{F}_2$. Illustrons ceci sur un exemple.

Exemple 14.5 (Division) *Effectuons la division de* $a(X) = X^{11} + X^9 + X^8 + X^7 + X^6 + X^4$ *par* $b(X) = X^8 + X^4 + X^3 + X^2 + 1$. *Ce sont des polynômes à coefficients dans* $\mathbb{F}_2$ *(donc* $- = +$ *!).*

$X^{11}+X^9+X^8+X^7+X^6+X^4$	$X^8+X^4+X^3+X^2+1$
$X^{11}+X^7+X^6+X^5+X^3$	X^3
$X^9+X^8+X^5+X^4+X^3$	
$X^9+X^8+X^5+X^4+X^3$	$+X$
$X^9+X^5+X^4+X^3+X$	
X^8+X	
X^8+X	$+1$
$X^8+X^4+X^3+X^2+1$	
$+X^4+X^3+X^2+X+1$	
$X^4+X^3+X^2+X+1$	

Le quotient est $q(X) = X^3+X+1$ *et le reste est* $r(X) = X^4+X^3+X^2+X+1$.

L'analogie avec les nombres entiers va plus loin. Nous disons que $b(X)$ divise $a(X)$, ou encore que $a(X)$ est multiple de $b(X)$ si le reste de $a(X)$ dans la division par $b(X)$ est nul ; cela équivaut à dire qu'il existe un polynôme $q(X)$ tel que $a(X) = b(X)q(X)$. Un polynôme est dit *irréductible* si les seuls polynômes qui le divisent sont de degré 0 (c'est à dire les constantes) Nous pouvons maintenant définir $\mathbb{F}_{256}$:

Le concept de polynôme irréductible est l'équivalent du concept de nombre premier.

Définition 14.3 (Construction de $\mathbb{F}_{256}$) *Le corps* $\mathbb{F}_{256}$ *est constitué des* 256 *polynômes binaires distincts de degré au plus* 7 *à coefficients dans* $\mathbb{F}_2$, *c'est-à-dire tous les polynômes* $\{0, 1, X, X+1, X^2, X^2+1, X^2+X, X^2+X+1, \cdots, X^7+X^6+X^5+X^4+X^3+X^2+X+1\}$.

L'addition est l'addition des polynômes usuelle, où les composantes binaires sont additionnées dans le corps $\mathbb{F}_2$.

Pour la multiplication, considérons d'abord un polynôme fixé $f(X) = X^8+X^4+X^3+X^2+1$, *qui est irréductible. Le produit de deux polynômes* $a(X)$ *et* $b(X)$ *est défini comme le reste dans la division par* $f(X)$ *du produit usuel de* $a(X)$ *et* $b(X)$.

Cela signifie que pour multiplier deux polynômes $a(X), b(X) \in \mathbb{F}_{256}$, nous faisons d'abord la multiplication usuelle des deux polynômes . Ceci donne, disons, le polynôme $\tilde{c}(X)$ qui peut avoir un degré au plus égal à 14. Calculons le reste dans la division par $f(X)$, c'est-à-dire, écrivons $\tilde{c}(X)$ comme $\tilde{c}(X) = f(X)\alpha(X) + c(X)$, où $c(X)$ est un polynôme de degré 7 au maximum. Notons qu'il y a une façon unique d'écrire $\tilde{c}(X)$. Le produit dans $\mathbb{F}_{256}$ de $a(X)$ et de $b(X)$ est alors, par définition, égal à $c(X)$.

Exemple 14.6 *Soit* $a(X) = X^4+X^2+1$, $b(X) = X^7+X^4$. *Alors,*

$$a(X)+b(X) = X^7+X^2+1$$

Notons que chaque fois que nous additionnons deux polynômes de degré 7 *au maximum, nous obtenons un polynôme de degré au plus* 7, *qui est aussi un élément de* $\mathbb{F}_{256}$.

La multiplication est un peu plus complexe. Notons $a(X) \cdot b(X)$ *cette multiplication, où les deux polynômes sont considérés comme les éléments*

du corps $\mathbb{F}_{256}$. Nous calculons d'abord le produit usuel des polynômes, que nous notons dans ce contexte $a(X) * b(X)$:

$$a(X) * b(X) = (X^4 + X^2 + 1) * (X^7 + X^4) = X^{11} + X^9 + X^8 + X^7 + X^6 + X^4$$

La partie droite n'est pas un élément du $\mathbb{F}_{256}$ parce que son degré est 11. Nous avons besoin de le "réduire", c'est à dire calculer son reste dans la division par $f(X)$:

$$X^{11} + X^9 + X^8 + X^7 + X^6 + X^4 =$$
$$(X^8 + X^4 + X^3 + X^2 + 1)(X^3 + X + 1) + (X^4 + X^3 + X^2 + X + 1).$$

donc

$$X^{11} + X^9 + X^8 + X^7 + X^6 + X^4 \equiv X^4 + X^3 + X^2 + X + 1 \pmod{f(X)}$$

donc finalement le produit dans $\mathbb{F}_{256}$ de $a(X)$ et $b(X)$ est $a(X) \cdot b(X) = X^4 + X^3 + X^2 + X + 1$.

Pour montrer que notre définition donne bien un corps, le seul point non évident est que tout polynôme possède un inverse. Cela est vrai parce que le polynôme $f(X)$ (le module) est irréductible. Nous ne montrons pas ce fait, mais signalons que c'est l'analogue du fait que $\mathbb{Z}/p\mathbb{Z}$ est un corps si p est un nombre premier.

Mentionnons aussi que l'algorithme d'Euclide de la Section 8.4 s'étend de manière immédiate au cas des polynômes et peut être utilisé pour calculer l'inverse dans $\mathbb{F}_{256}$.

POUR CALCULER EFFICACEMENT dans $\mathbb{F}_{256}$, nous utilisons un élément générateur, c'est à dire un élément $g(X) \in \mathbb{F}_{256}$ tel que tous les éléments non nuls de $\mathbb{F}_{256}$ soient une puissance de $g(X)$. Nous savons qu'un tel élément existe dans tout corps fini. De plus, il est possible de s'arranger (par le choix du module $f(X)$) pour que $g(X) = X$ soit un générateur. Cela permet de simplifier considérablement les calculs, ce qui revient à utiliser le le logarithme discret. Nous expliquons ceci sur un exemple.

Exemple 14.7 *Par simplicité, nous choisisson un corps plus petit $\mathbb{F}_{16}$ (notons que $16 = 2^4$). Ce corps contient tous les polynômes binaires de degré ≤ 3 $\{0, 1, X, X^2, X^3, X+1, X^2+1, ...X^3+X^2+X+1\}$, où l'addition est l'addition usuelle des polynômes et la multiplication est définie modulo le polynôme irréductible $f(X) = X^4 + X^3 + 1$. Ce polynôme est irréductible et est tel que X est un générateur.*

En effet, comme nous travaillons modulo $X^4 + X^3 + 1$, nous avons

$$\begin{aligned}
X^4 &= -X^3 - 1 = X^3 + 1 \\
X^5 &= X.(X^3 + 1) = X^4 + X = X^3 + X + 1 \\
X^6 &= X^4 + X^2 + X = X^3 + X^2 + X + 1 \\
X^7 &= X^4 + X^3 + X^2 + X = 2X^3 + X^2 + X + 1 = X^2 + X + 1 \\
&\text{etc.} \\
X^{15} &= 1
\end{aligned}$$

donc la période de X pour la multiplication est 15, qui est le cardinal du groupe multiplicatif, et donc $g(X) = X$ est un générateur.

Nous pouvons représenter un élément de $\mathbb{F}_{16}$ par la suite des coefficients (usuellement représentée par ordre de puissances décroissantes),

comme dans la colonne (c) de la Figure 14.2. L'addition est alors très facile à faire car nous additionnons les polynômes comme d'habitude, en ajoutant les coefficients modulo 2, et c'est donc l'opération `xor` *bit à bit.*

Par contre, pour la multiplication, il est plus simple d'utiliser la représentation par une puissance de X, comme dans la colonne (a) de la Figure 14.2. Par exemple,

$$\begin{aligned}(1110).(0101) &= X^8.X^9 = X^{17} = X^2 = (0100)\\ (0111)^{-1} &= X^{-7} = X^{-7}.X^{15} = X^8 = (1110)\end{aligned}$$

Exemple 14.8 (Le corps $\mathbb{F}_4$ revisité) *Dans l'Exemple 12.5 nous avons obtenu des représentations de $\mathbb{F}_4$ par déduction directe. Regardons maintenant, à titre de comparaison, ce que donne la construction à partir de polynômes. Avec cette construction, les éléments du corps $\mathbb{F}_4$ sont les polynômes binaires de degré ≤ 1, $\{0, 1, X, X+1\}$. L'addition est l'addition usuelle des polynômes et la multiplication est modulo le polynôme $f(X) = X^2 + X + 1$, dont on peut vérifier qu'il est irréductible.*

Construisons la table de multiplication. On a bien sûr $0 \cdot P(X) = 0$ et $1 \cdot P(X) = P(X)$. De plus

$$\begin{aligned}X \cdot X &= X^2 = 1 + X + f(X) \equiv X + 1 \pmod{f(X)}\\ (X+1)\cdot(X+1) &= X^2 + 1 = X + f(X) \equiv X \pmod{f(X)}\\ X \cdot (X+1) &= X^2 + X = 1 + f(X) \equiv 1 \pmod{f(X)}\end{aligned}$$

Nous obtenons donc les tables suivantes :

$+$	0	1	X	$X+1$
0	0	1	X	$X+1$
1	1	0	$X+1$	X
X	X	$X+1$	0	1
$X+1$	$X+1$	X	1	0

$\cdot$	0	1	X	$X+1$
0	0	0	0	0
1	0	1	X	$X+1$
X	0	X	$X+1$	1
$X+1$	0	$X+1$	1	X

Si nous représentons les polynômes par leurs deux coefficients (par puissance décroissante) nous obtenons les tables suivantes :

$+$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$\cdot$	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

qui sont les mêmes qu'à la fin de l'Exemple 12.5.

14.4 Codes Correcteurs et Détecteurs en Pratique

Nous avons vu que les codes linéaires permettent de résoudre efficacement le problème de la correction ou détection d'erreur. Les codes linéaires sont facile à stocker (par leur matrice génératrice ou de contrôle). De plus, l'encodage, la détection d'erreur et la correction d'effacement sont faciles. Les codes de Reed-Solomon sont

(a)	(b)	(c)
0	0	0000
1	1	0001
X	X	0010
X^2	X^2	0100
X^3	X^3	1000
X^4	X^3+1	1001
X^5	X^3+X+1	1011
X^6	X^3+X^2+X+1	1111
X^7	X^2+X+1	0111
X^8	X^3+X^2+X	1110
X^9	X^2+1	0101
X^{10}	X^3+X	1010
X^{11}	X^3+X^2+1	1101
X^{12}	$X+1$	0011
X^{13}	X^2+X	0110
X^{14}	X^3+X^2	1100
X^{15}	1	0001

FIGURE 14.2: Le corps $\mathbb{F}_{16}$. Les éléments de $\mathbb{F}_{16}$ peuvent être vus comme des polynômes de degré ≤ 3 à coefficients dans $\mathbb{F}_2$ (colonne (b)). Ils peuvent aussi être représentés comme nombres binaire (colonne (c)), ou comme puissance du générateur $g(X) = X$ (colonne (a)).

Q. 112. Démontrez que le polynôme binaire $f(X) = X^2 + X + 1$ est irréductible.

optimaux en terme de distance minimale, et sont utilisés sur un très grand nombre de systèmes. Par exemple, les lecteurs de CDs utilise plusieurs codes de Reed-Solomon imbriqués. La correction d'erreur est une tâche plus difficile. Ici, la linéarité aide beaucoup et il faut aussi des structures supplémentaires. Mais ceci est une autre histoire...

Bibliographie

Y. Biollay, A. Chaabouni, and J. Stubbe. *Savoir-faire en maths : bien commencer ses études scientifiques*. Presses polytechniques et universitaires romandes, 2008. ISBN 2880747791.

Alexis Fabre-Ringborg and Sébastien Saunier. Entropie du français. `http://cb.sogedis.fr/files/entropie/Entropie_Francais_FabreRingoborg_Saunier.pdf`, 2003.

H. Gharavi and R. Steele. Conditional entropy encoding of LOG-PCM speech. *Electronics Letters*, 21(11) :475–476, 2007. ISSN 0013-5194.

G. Michaud-Brière, Y. Pearson, S. Perreault, and L.-O. Roof. La cryptographie. `http://nomis80.org/cryptographie/cryptographie.html`, 2002.

C.E. Shannon. The mathematical theory of communication. *Bell Syst. Tech. J*, 27 :379–423, 1948.

C.E. Shannon. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1) :50–64, 1951.

R. Singleton. Maximum distance q-nary codes. *Information Theory, IEEE Transactions on*, 10(2) :116–118, 1964.

Livres

T. Cover and J. Thomas, *Elements of Information Theory*. Wiley & Sons, New York, 1991.

R. W. Hamming, *Coding and information theory*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

R. B. Ash, *Information Theory*. Dover Publications Inc, New York, 1990.

Gérard Battail, *Théorie de l'Information*. Ed Masson, 1997.

David MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003

S. Lin and D. J. Costello, Jr., *Error Control Coding : Fundamentals and Applications.* Prentice Hall, 1983.

R. E. Blahut, *Theory and Practice of Error-Control Codes.* Addison-Wesley, 1983.

W. C. Huffman and V. Pless, *Fundamentals of Error Correcting Codes.* Cambridge University Press, 2003

R. J. McEliece, *The Theory of Information and Coding.* Cambridge University Press

Réponses aux Questions en Marge

Q.1. *(p. 3).* Bravo, vous avez trouvé, c'est bien ici.

Q.2. *(p. 10).* La même que pour S_1, $p_{S_2}(j) = 1/6$ pour $j = 1, ..., 6$.

Q.3. *(p. 11).* Non, car par exemple on a $p_L(0,0) = 0 \neq p_{L_1}(0)p_{L_2}(0) = \frac{5}{6}\frac{3}{36}$.

Q.4. *(p. 12).* Oui, d'après la remarque précédente. L'hypothèse entraîne que S_1 et S_2 sont indépendantes, donc la densité conditionnelle de S_2 sachant que $S_1 = s_1$ ne dépend pas de la valeur de s_1, pour tout s_1 tel que $p_{S_1}(s_1) > 0$.

Q.5. *(p. 13).* $p_{S_1|S_2}(i|j) = 1/6 = p_{S_1}(i)$. La densité conditionnelle sachant que $S_2 = j$ est la même pour tous les j. S_1 et S_2 sont indépendantes.

Q.6. *(p. 15).* Si une information, correspondant à une probabilité p, vaut 1 ban, c'est qu'on a $\log_{10}(1/p) = 1$. Si elle vaut 1 déciban, c'est qu'on a $\log_{10}(1/p) = 0.1$. L'information mesurée en bits est $B = \log_2(1/p)$. On a $\log_2(1/p) = \frac{\log_{10}(1/p)}{\log_{10}(2)}$ donc $B = \frac{0.1}{\log_{10}(2)} = 0.332$. Un déciban vaut environ un tiers de bit.

Q.7. *(p. 16).* Le maximum vaut 1 bit et est obtenu pour $q = 0.5$ (les symboles sont équiprobables). Le minimum vaut 0 et est obtenu pour $q = 0$ ou $q = 1$ (la source est certaine).

Q.8. *(p. 16).* Il y égalité, comme on le voit par la définition.

Q.9. *(p. 19).* S est une source composée dont les n composantes $S_1, S_2, ...$ sont indépendantes, donc $H(S) = H(S_1) + H(S_2) + ... = n\log_2(6)$.

Q.10. *(p. 21).* Non, car -2 et 2 ont la même image, donc on ne peut pas dire qu'à tout élément de l'ensemble d'arrivée correspond un seul élément de l'ensemble de départ.

Q.11. *(p. 21).* Le dictionnaire $\mathcal{C}$ est un sous-ensemble de $\mathcal{D} \times ... \times \mathcal{D} = \mathcal{D}^L$, dont le cardinal est D^L. Le cardinal de $\mathcal{C}$ est M donc $M \leq D^L$.

Q.12. *(p. 22).* f n'est pas injective car par exemple $f(-1) = f(1)$. Elle n'est pas surjective car -1 n'a pas d'antécédent. g n'est pas injective pour la même raison mais est surjective car pour tout $y \in [0, +\infty)$ l'équation en $x : y = x^2$ a au moins une solution. h est bijective car pour tout $y \in [0, +\infty)$ l'équation en $x : y = x^2$ a une solution unique dans $[0, +\infty)$.

Q.13. *(p. 23).* Oui, car d'abord il est à décodage unique, et ensuite, quand on connaît la longueur commune L de tous les mots de code, il suffit de compter les symboles de code reçus. Quand on a reçu une suite de L symboles, on peut la décoder sans attendre.

Q.14. *(p. 24).* Oui, car si tous les mots de code ont même longueur, aucun mot de code ne peut être préfixe d'un autre.

Q.15. *(p. 24).* Contraposée : $(P2)$: (n n'est pas divisible par 4) $\Rightarrow$ (n est impair).

Réciproque : $(P3)$: (n est divisible par 4) $\Rightarrow$ (n est pair).

Contraposée de la réciproque : $(P4)$: (n est impair) $\Rightarrow$ (n n'est pas divisible par 4).

$(P3)$ et $(P4)$ sont vraies pour tout n, mais $(P1)$ et $(P2)$ ne le sont pas.

Q.16. *(p. 25).*

1. faux ; par exemple C est à décodage unique mais n'est pas instantané
2. vrai, c'est le Théorème 2.1
3. vrai, c'est la contraposée de l'item précédent
4. faux, c'est la contraposée de l'item 1

Q.17. *(p. 25).* Notons d'abord que O est de longueur constante, donc instantané, donc son arbre de décodage est bien défini. Son arbre de décodage est égal à son arbre complet (voir Figure 2.1).

Q.18. *(p. 25).* Ce sont les codes D-aires de longueur constante $L_{\max}$ et dont le nombre de mots de code est $D^{L_{\max}}$, tel le code O.

Q.19. *(p. 26).* Si la suite de symboles X est reçue sans erreur il n'y aura pas d'erreur de décodage. Par contre si certains symboles de code reçus sont faux ou manquant, l 'algorithme pourrait être conduit à descendre l'arbre en suivant des branches absentes. Par exemple, avec le code B, si on reçoit $X = 1111$, l'algorithme retourne une erreur.

Q.20. *(p. 29).* Elle dit que $MD^{-L} \leq 1$, c'est à dire $M \leq D^L$. Le théorème de Kraft-McMillan est évident dans ce cas : un code de longueur constante est à décodage unique donc doit satisfaire cette

inégalité, ce que l'on sait déjà puisque tous les mots de code sont distincts et on peut former au maximum D^L mots de longueur L ; réciproquement, si $M \leq D^L$ on peut trouver un code de longueur constante égale à L, il suffit de considérer toutes les suites de L éléments de $\mathcal{X}$ (il y en a $D^L \geq M$), et de n'en conserver que M.

Q.21. *(p. 30).*

1. vrai. Γ est à décodage unique (Théorème 2.1) donc satisfait l'inégalité de Kraft.
2. vrai d'après le théorème de Kraft-McMillan.
3. faux. Le code C vérifie l'inégalité de Kraft mais n'est pas instantané.
4. faux : voir Exemple 2.5.
5. = 2. (La proposition 5. est équivalente à la proposition 2., donc elle est vraie.)
6. = 4., faux
7. = 1., vrai
8. = 3., faux.

Q.22. *(p. 31).* Un code binaire de longueur constante L comporte au plus 2^L mots de code, comme le code doit comporter 4 mots de code, il faut que $L \geq 2$. Donc la longueur moyenne d'un code constant est $L \geq 2 > 1.35$, qui est la longueur de B' et B' est donc plus efficace.

Q.23. *(p. 35).* Oui, en utilisant la méthode de la Section 2.7.

Q.24. *(p. 35).* En général non. Pour les exemples que nous avons vu, Γ_H est un code de Huffman et $L(\Gamma_H) = 1.30$ qui est supérieur à la borne inférieure 1.022.

Q.25. *(p. 37).* Soit S_1 la source qui donne une lettre (sans préciser majuscule ou minuscule), et S_2 une deuxième source qui précise si le caractère est minuscule ou majuscule. L'entropie du nouveau robot-page est $H(S_1, S_2) \geq H(S_1)$ d'après Eq.(4.1). Nous pouvons même dire plus, car S_2 est une source binaire donc $H(S_2|S_1 = s_1) \leq 1$ bit pour tout s_1 et donc aussi $H(S_2|S_1) \leq 1$ bit. Or

$$H(S_1, S_2) = H(S_1) + H(S_2|S_1) \leq H(S_1) + 1$$

donc finalement l'entropie du nouveau robot-page est supérieure à celle de l'ancien, et la dépasse d'au maximum 1 bit.

Q.26. *(p. 38).*

$$H(L_1|L_2) = H(L) - H(L_2) = 0.0055556 \approx 0.05 \text{ bit}$$

Q.27. *(p. 39).*

1. Si $H(S_2|S_1) = H(S_2)$ alors S_1 et S_2 sont indépendantes.
2. $H(S_2|S_1) = H(S_1)$: on ne peut rien conclure de particulier.

Q.28. *(p. 39).* Par récurrence.

(Etape d'Initialisation) La propriété est vraie pour $n = 2$ (première égalité du Théorème 4.1).

(Etape de Récurrence) Supposons que la propriété soit vraie jusqu'à n et appliquons le Théorème 4.4 à la source $((S_1, ..., S_n), S_{n+1})$:

$$H(S_1, ..., S_n, S_{n+1}) = H(S_{n+1}|S_1, ..., S_n) + H(S_1, ..., S_n)$$
$$= H(S_{n+1}|S_1, ..., S_n) + H(S_n|S_1, S_2, ..., S_{n-1}) + H(S_{n-1}|S_1, S_2, ..., S_{n-2})$$
$$+ ... + H(S_3|S_1, S_2) + H(S_2|S_1) + H(S_1)$$

donc la formule est vraie aussi pour $n + 1$.

Q.29. *(p. 39).*

1. $H(S_1, S_2) = H(S_1) + H(S_2|S_1)$ VRAI
2. $H(S_1, S_2) = H(S_2) + H(S_2|S_1)$ FAUX en général
3. $H(S_1) \geq H(S_1, S_2)$ FAUX en général
4. $H(S_1, S_2) = H(S_1) + H(S_2)$ FAUX en général (vrai si S_1 et S_2 sont indépendantes)
5. $H(S_2|S_1) \geq 0$ VRAI
6. $H(S_1, S_2) \geq H(S_1) + H(S_2)$ FAUX en général (mais vrai avec égalité si S_1 et S_2 sont indépendantes)
7. $H(S_1|S_2) \leq H(S_1)$ VRAI
8. $H(S_1, S_2) = H(S_1) + H(S_1|S_2)$ FAUX en général
9. $H(S_1) \leq H(S_1, S_2)$ VRAI
10. $H(S_1|S_2) \leq H(S_2)$ FAUX en général
11. $H(S_2) \leq H(S_1, S_2)$ VRAI
12. $H(S_2|S_1) \leq H(S_2)$ VRAI
13. $H(S_2) \geq H(S_1, S_2)$ FAUX en général
14. $H(S_1, S_2) \leq H(S_1) + H(S_2)$ VRAI
15. $H(S_1, S_2) = H(S_2) + H(S_1|S_2)$ VRAI

Q.30. *(p. 40).* D'après le Théorème 4.6, $H(S_2) \leq H(S_1)$ et $H(S_1) \leq H(S_2)$. Donc $H(S_1) = H(S_2)$.

Q.31. *(p. 41).* Si on veut y parvenir, il faut maximiser l'information reçue à chaque question, donc essayer d'obtenir une entropie conditionnelle aussi proche que possible de 1bit, donc essayer de poser des questions dont les réponses sont équiprobables.

Cela amène à la méthode de *dichotomie* : à chaque étape, Bernard divise par 2 l'intervalle des réponses possibles. Une façon simple de décrire cela est d'utiliser la représentation binaire du numéro inconnu (c'est à dire de l'écrire en base 2). Tout nombre entier peut s'écrire en base 2, c'est à dire en utilisant seulement les chiffres 0 et 1. Par exemple la représentation binaire du nombre décimal 23 est

10111 car

$$23 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Le plus grand nombre entier qu'on peut écrire avec 14 chiffres binaires est $x = 11\ 1111\ 1111\ 1111$, si on lui ajoute 1 on obtient 100 0000 0000 0000, donc

$$x = 2^{14} - 1 = 16383$$

Donc on peut écrire tous les numéros de cadenas sur 14 chiffres binaires.

Imaginons que Bernard demande pour chaque chiffre binaire du cadenas s'il vaut 0. Cela fait 14 questions, et Bernard obtiendra le numéro tant convoité à coup sûr. Donc la réponse est oui.

Q.32. *(p. 41).*

1. Si $H(S_1, S_2) = H(S_1)$ alors S_2 est fonction de S_1.
2. Si $H(S_2|S_1) = 0$ alors S_2 est fonction de S_1.

Q.33. *(p. 45).* Soit $\mathcal{S} = (S_1, ...S_n, ...)$ une source stationnaire.

(1) $H(S_n)$ est indépendant de n puisque la densité de probabilité de S_n est indépendante de n. Donc l'entropie d'un symbole existe.

(2) Soit pour $n \geq 2$:

$$u_n \stackrel{\text{def}}{=} H(S_n|S_1, S_2, ..., S_{n-1}) \tag{14.6}$$

Par le Théorème 4.3 ("conditionner réduit l'entropie") :

$$u_n = H(S_n|S_1, S_2, ..., S_{n-1}) \leq H(S_n|S_2, ..., S_{n-1})$$

D'autre part, puisque la source est stationnaire, la densité de probabilité de $(S_2, ..., S_n)$ est la même que celle de $(S_1, ..., S_{n-1})$. Il s'en suit que

$$H(S_n|S_2, ..., S_{n-1}) = H(S_{n-1}|S_1, ..., S_{n-2}) = u_{n-1}$$

puisque ces quantités sont entièrement calculées à partir de la densité de probabilité. Donc

$$u_n \leq u_{n-1}$$

ce qui exprime que la suite u_n est décroissante au sens large. Or $u_n \geq 0$ puisque c'est une entropie conditionnelle. Un théorème classique de l'analyse des nombres réels dit qu'une suite ≥ 0 décroissante au sens large ne peut pas faire autrement que converger vers une limite finie, donc l'entropie par symbole existe.

Les deux conditions de la Définition 5.2 sont satisfaites donc la source étendue $\mathcal{S}$ est régulière.

Q.34. *(p. 46).* Que les sources marginales $S_1, S_2, ...$ ne sont pas indépendantes.

Q.35. *(p. 46).* Posons $u_1 = H(S_1) = H(\mathcal{S})$ et $u_n = H(S_n|S_1, ..., S_{n-1})$. Nous savons que $\mathcal{S}$ est régulière et que $\lim_{n\to\infty} u_n = H^*(\mathcal{S})$. Nous savons aussi (Question 33) que

$$H(\mathcal{S}) = u_1 \geq u_2 \geq ... \geq u_n ... \geq H^*(\mathcal{S})$$

Or $H(\mathcal{S}) = H^*(\mathcal{S})$ par hypothèse donc il y a égalité partout :

$$H(\mathcal{S}) = u_1 = u_2 = ... = u_n ... = H^*(\mathcal{S})$$

en particulier $u_n = u_1$ c'est à dire $H(S_n|S_1, ..., S_{n-1}) = H(S_1) = H(S_n)$; par le Théorème 1.4, S_n et $S_1, ..., S_{n-1}$ sont indépendantes, et cela est vrai pour tout n. Cela exprime que toutes les sources marginales sont indépendantes.

Q.36. *(p. 46).* Si l'entropie par symbole est nulle on ne peut rien conclure de très précis, mais on peut dire qu'une très longue suite de symboles a tendance à ne pas apporter beaucoup plus d'information qu'une suite moins longue ; on peut qualifier une telle source de "bavarde", comme un locuteur qui a tendance à se répéter.

Q.37. *(p. 49).* **Pile ou Face.** Le code de Huffman est le code 0/1, nous utilisons 1 bit de code par symbole de source, donc la longueur est 60 bits, ce qui fait un bit de code par symbole de source.

Beau ou Mauvais. Ici c'est plus compliqué, il nous faut encoder la suite de la Figure 5.1 en utilisant le code de la Table 5.3. Nous obtenons une suite de 10 mots de code dont les longueurs sont $2, 5, 5, 2, 5, 2, 2, 2, 5, 5$. La longueur totale est 35, ce qui fait 0.583 bit de code par symbole de source.

Vert ou Bleu. Il faut 1 bit pour coder toute la suite de 60 symboles, ce qui fait 0.0167 bit de code par symbole de source.

k			
0	H	A	L
25	I	B	M
24	J	C	N
23	K	D	O
22	L	E	P
21	M	F	Q
20	N	G	R
19	O	H	S
18	P	I	T
17	Q	J	U
16	R	K	V
15	S	L	W
14	T	M	X
13	U	N	Y
12	V	O	Z
11	W	P	A
10	X	Q	B
9	Y	R	C
8	Z	S	D
7	A	T	E
6	B	U	F
5	C	V	G
4	D	W	H
3	E	X	I
2	F	Y	J
1	G	Z	K

Q.38. *(p. 51).* Oui, car la suite des "B" et "N" dans la colonne du milieu est alternée, il suffit donc de connaître le premier. On peut même éviter totalement d'indiquer "B" ou "N en convenant par exemple que la première plage est blanche (si le premier pixel est noir, on considère alors que la longueur de la première plage est 0). Cela permet de supprimer 1 bit de la suite des symboles à encoder, et donc de réduire la longueur moyenne du code de Huffman.

Q.39. *(p. 54).* Essayons une recherche exhaustive, qui n'est pas trop compliquée puisqu'il n'y a que 26 clés. Le texte clair est sans doute "IBM" et la clé est $K = 25$.

Q.40. *(p. 57).* La clé est choisie uniformément parmi les 2^n clés possibles, donc $H(\mathcal{K}) = \log_2(2^n) = n$ bits.

La densité de probabilité du texte chiffré est aussi uniforme. En effet, d'après l'Eq.(6.1) et le Théorème 0.1, la densité de probabilité de la clé est $p_{\mathcal{C}}(C) = \frac{1}{2^n}$. Donc $H(\mathcal{C}) = n$ bits aussi.

Le texte clair comporte au plus n bits, donc $H(\mathcal{P}) \leq \log_2(2^n) = n$. Les deux inégalités des théorèmes sont bien vérifiées.

Q.41. *(p. 59).* $23 = 4 \times 5 + 3$ donc $q = 4$ et $r = 3$.

$-23 = -5 \times 5 + 2$ donc $q = -5$ et $r = 2$. Attention, le reste dans la division de -23 n'est pas -3, car, par définition de la division euclidienne, un reste est toujours ≥ 0.

Q.42. *(p. 59).*

$$13 \bmod 10 = 3$$
$$(-13) \bmod 10 = 7$$
$$13 \bmod (-10) = 3$$
$$(-13) \bmod (-10) = 7$$

13 mod 0 n'est pas défini.

Q.43. *(p. 59).* On a

$$24163584354 = 10 \times 2416358435 + 4$$

donc le reste de 24163584354 dans la division par 10 est 4. D'une manière générale, le reste d'un nombre entier positif dans la division par 10 est son dernier chiffre.

Q.44. *(p. 59).* Nous faisons la preuve seulement pour $b \geq 2$.

(Existence :) Pour un nombre réel quelconque x soit $\lfloor x \rfloor$ la partie entière par défaut, c'est à dire que $\lfloor x \rfloor \in \mathbb{Z}$ et $x - 1 < \lfloor x \rfloor \leq x$. Soit $q = \lfloor \frac{a}{b} \rfloor$ et $r = a - bq$, de sorte que $a = bq + r$. Nous avons $0 \leq r < b$ donc, comme r et b sont entiers, $0 \leq r \leq b - 1$. Donc il existe un couple d'entiers (q, r) qui satisfait les conditions demandées.

(Unicité :) Supposons que $a = bq + r$ avec q, r entiers satisfaisant les conditions demandées. Nous avons alors $q = \frac{a}{b} - \frac{r}{b}$ donc $\frac{a}{b} - \frac{b-1}{b} \leq q \leq \frac{a}{b}$ donc $\frac{a}{b} < q \leq \frac{a}{b}$. Comme q est entier, $q = \lfloor \frac{a}{b} \rfloor$, d'où $r = a - b\lfloor \frac{a}{b} \rfloor$ donc q et r sont uniquement définis.

Q.45. *(p. 59).* 27 et 255 sont divisibles par 3, 256 est divisible par 2 donc aucun de ces nombres n'est premier.

Q.46. *(p. 60).* $12 = 2^2 \times 3$, $100 = 2^2 \times 5^2$ et $256 = 2^8$.

Q.47. *(p. 60).* Soit p_1 le plus petit facteur premier. On peut écrire $a = p_1 b$ avec $b = p_1^{\alpha_1 - 1} ... p_k^{\alpha_k}$. Comme $p_2 > p_1$ et $b \geq p_2$ il s'en suit que $b > p_1$. Donc

$$a > p_1^2$$

donc $p_1 < \sqrt{a}$.

Q.48. *(p. 60).* Pour le savoir, nous testons si les éléments de la suite des nombres premiers divisent 257 ; il suffit même de s'arrêter à $\sqrt{257} \approx 16$ car si a n'est pas premier son plus petit facteur

premier est < 17 (Question 47). Nous testons donc $2, 3, 5, 7, 11, 13$: aucun de ces nombres ne divise 257 donc c'est un nombre premier.

Q.49. *(p. 61).* Nous avons les décompositions $12 = 2^2 \times 3$ et $100 = 2^2 \times 5^2$ donc $\text{pgcd}(12, 100) = 2^2 = 4$.

Q.50. *(p. 61).* $12 = 2^2 \times 3$ et $20 = 2^2 \times 5$, le facteur premier 2 est commun, donc ils ne sont pas premiers entre eux.

$12 = 2^2 \times 3$ et $35 = 5 \times 7$ donc 12 et 35 n'ont aucun facteur premier en commun, donc sont premiers entre eux.

257 est un nombre premier et $234 < 257$ donc 234 et 257 sont premiers entre eux.

Q.51. *(p. 61).*

1. C'est une conséquence de l'item 2 (il suffit d'appeler a le plus petit des deux nombres premiers et p l'autre).
2. La décomposition en facteurs premiers de a ne peut pas comporter p car sinon on aurait $a \geq p$, et la décomposition en facteurs premiers de p est p ; donc les deux décompositions n'ont aucun facteur commun. Donc (Théorème 50) a et p sont premiers entre eux.
3. Soit $a = p_1^{\alpha_1} ... p_k^{\alpha_k}$ et $b = q_1^{\beta_1} ... q_\ell^{\beta_\ell}$ les décompositions en facteurs premiers. Comme a et b sont premiers entre eux, $p_i \neq q_j$ pour tous i et j. Comme a divise c, la décomposition en facteurs premiers de c comporte p_i avec un exposant $\geq \alpha_i$, pour tout i, et aussi q_j avec un exposant $\geq \beta_j$. Donc

$$c = p_1^{\alpha'_1} ... p_k^{\alpha'_k} q_1^{\beta'_1} ... q_\ell^{\beta'_\ell} r_1^{\gamma_1} ... r_m^{\gamma_m}$$

avec $\alpha'_i \geq \alpha_i$ et $\beta'_j \geq \beta_j$, et où les r_m sont les autres facteurs premiers de c, s'il y en a. Donc

$$c = ab \left[p_1^{(\alpha'_1 - \alpha_1)} ... p_k^{(\alpha'_k - \alpha_k)} q_1^{(\beta'_1 - \beta_1)} ... q_\ell^{(\beta'_\ell - \beta_\ell)} r_1^{\gamma_1} ... r_m^{\gamma_m} \right]$$

et c est divisible par ab.

Q.52. *(p. 61).* 257 est un nombre premier donc tous les nombres de 1 à 256 sont premiers avec 257. Il y en a 256.

Q.53. *(p. 61).* Oui :

($\Rightarrow$) Supposons a divisible par 12, donc $\frac{a}{12}$ est entier, donc aussi $\frac{a}{3} = 4\frac{a}{12}$ donc a divisible par 3, de même par 4. ($\Leftarrow$) Supposons que a est divisible par 3 et par 4. Par l'item 3 du Théorème 7.6, a est divisible par 12 car 3 et 4 sont premiers entre eux.

Q.54. *(p. 61).* Non, par exemple 6 est divisible par 2 et 6 mais pas par 12. On ne peut pas appliquer le Théorème 7.6 car 2 et 6 ne sont pas premiers entre eux.

Q.55. *(p. 61).* Elles sont toutes vraies sauf $-23 \equiv 3 \pmod 5$.

Q.56. *(p. 62).*

1. FAUX ; par exemple $4 \equiv 0 \pmod 2$ mais 4 ne divise pas 2.
2. VRAI ; ($\Rightarrow$) : si $a \equiv 0 \pmod m \Leftrightarrow m$ alors $a = mq$ où q est le quotient dans la division par m donc $\frac{a}{m} = q$ et donc m divise a. ($\Leftarrow$) : supposons que m divise a ; donc $a = mx$ pour un certain $x \in \mathbb{Z}$. Donc $a = mx + 0$ et donc par le Théorème 7.1, le reste est $r = 0$, donc $a \equiv 0 \pmod m$.
3. FAUX : par exemple $4 \equiv 0 \pmod 2$ mais 4 et 2 ne sont pas premiers entre eux (leur PGCD est 2).

Q.57. *(p. 62).*

1. Oui.
2. Non (la transitivité n'est pas vraie : $0\mathcal{R}_2 1$ et $1\mathcal{R}_2 2$ mais on n'a pas $0\mathcal{R}_2 2$).

Q.58. *(p. 63).* **Division par** 10 **:** on a

$$\begin{aligned} &10^0 \equiv 1 \pmod{10} \\ &10 \equiv 0 \pmod{10} \\ &10^k \equiv 0 \pmod{10}, k \geq 1 \end{aligned}$$

donc

$$\begin{aligned} &a = d_k \times 10^k + d_{k-1} \times 10^{k-1} + ... + d_1 \times 10^1 + d_0 \times 10^0 \\ &\equiv d_k \times 0 + d_{k-1} \times 0 + ... + d_1 \times 0 + d_0 \times 1 \pmod{10} \\ &\equiv d_0 \pmod{10} \end{aligned}$$

donc tout nombre entier est congru modulo 10 à son dernier chiffre décimal.

Division par 3 **:** on a

$$10 \equiv 1 \pmod 3$$

comme pour $m = 9$ donc tout nombre entier est congru modulo 3 à la somme de ses chiffres décimaux.

Division par 4 **:** on a

$$\begin{aligned} &10^0 \equiv 1 \pmod 4 \\ &10^1 \equiv 1 \pmod 4 \\ &10^2 \equiv 0 \pmod 4 \\ &10^k \equiv 0 \pmod 4, k \geq 2 \end{aligned}$$

donc

$$\begin{aligned} &a = d_k \times 10^k + d_{k-1} \times 10^{k-1} + ... + d_1 \times 10^1 + d_0 \times 10^0 \\ &\equiv d_k \times 0 + d_{k-1} \times 0 + ... + d_2 \times 0 + d_1 \times 10 + d_0 \times 1 \pmod{10} \\ &\equiv d_1 \times 10 + d_0 \times 1 \pmod{10} \end{aligned}$$

donc tout nombre entier est congru modulo 4 au nombre constitué par ses deux derniers chiffres décimaux.

Les restes de a sont :

- division par 10 : $a \equiv 0 \pmod{10}$ donc le reste est 0 ; (a est divisible par 10)
- division par 3 : $a \equiv (1+2+3+4+5+6+7+8+9) \times 3 \equiv 135 \equiv 9 \equiv 0$ donc le reste est 0 (a est divisible par 3) ;
- division par 4 : $a \equiv 10 \equiv 2$ donc le reste est 2 ;

Q.59. *(p. 65).* Toutes les erreurs simples sauf l'erreur qui consiste à remplacer un chiffre 0 par 9 ou vice versa sont détectées.

Les erreurs de permutation ne sont pas détectées.

Q.60. *(p. 67).* Les classes sont $c_0 = \{0, 1, ..., 9\}$, $c_1 = \{10, 11, ..., 99\}$, $c_2 = \{100, 101, ..., 999\}$..., $c_k = \left\{10^k, ..., 10^{k+1} - 1\right\}$...

Q.61. *(p. 68).* Il suffit de consulter les tables d'addition et de multiplication dans $\mathbb{Z}/3\mathbb{Z}$ puis $\mathbb{Z}/4\mathbb{Z}$. Pour $m = 3$:

1. $[x]_3^2 = [1]_3$ a deux solutions : $x = [1]_3$ et $x = [2]_3$.
2. $[2]_3[x]_3 = [0]_3$ a une solution $x = [0]_3$.
3. $[x]_3 + [x]_3 = [0]_3$ est la même équation que en 2.

Pour $m = 4$:

1. $[x]_3^2 = [1]_4$ a deux solutions : $x = [1]_4$ et $x = [3]_4$.
2. $[2]_4[x]_4 = [0]_4$ a deux solution $x = [0]_4$ et $x = [2]_4$..
3. $[x]_4 + [x]_4 = [0]_4$ est la même équation que en 2.

Q.62. *(p. 69).*

1. $([-5]_7)^2 + [-4]_7\,[-3]_7 = ([2]_7)^2 + [12]_7 = [4]_7^2 + [5]_7 = [9]_7 = [2]_7$
2. $2\,([3]_4 + [5]_4) - 5\,([3]_4)^2 = 2[8]_4 - 5[9]_4 = 2[0]_4 - 5[1]_4 = [-5]_4 = [3]_4$
3. $([298242]_9)^{1000} = ([0]_9)^{1000} = [0]_9$ car $298242 \equiv 0 \pmod{9}$ (voir Exemple 7.1).

Q.63. *(p. 69).* Une erreur. Dans notre contexte, elle n'a pas de sens, car la somme (comme le produit) est définie uniquement pour des classes de congruence modulo le même m. Nous ne pouvons donc pas additionner $[3]_4$ et$[4]_3$.

Q.64. *(p. 69).* Il suffit de regarder les tables de multiplication (Table 8.1). Il n'y a pas de diviseur de zéro dans $\mathbb{Z}/3\mathbb{Z}$ car il n'y a pas de 0 dans la table de multiplication sauf dans les lignes ou colonnes de 0.

Par contre, il y a des diviseurs de 0 dans $\mathbb{Z}/4\mathbb{Z}$: $[2]_4$ est un diviseur de zéro.

Q.65. *(p. 70).* Il n'y a guère d'autre moyen que la force brute, c'est à dire un recherche exhaustive (nous laissons tomber les crochets, les calculs sont dans $\mathbb{Z}/13\mathbb{Z}$:) :

$$\begin{aligned} 2^0 &= 1 \\ 2^1 &= 2 \\ 2^2 &= 4 \\ 2^3 &= 8 \\ 2^4 &= 3 \end{aligned}$$

Donc $x = 4$ est une solution.

Q.66. *(p. 70).* $[0]_m a' = [0]_m$ pour tout $m \geq 2$ donc il est impossible de trouver a' tel que $[0]_m a' = [1]_m$.

C'est vrai, car $[1]_m[1]_m = [1]_m$ pour tout $m \geq 2$.

Q.67. *(p. 72).* Appliquer le Théorème 8.5 et remarquer que la relation "être premiers entre eux" est symétrique.

Q.68. *(p. 73).*

- VRAI : $[a]_m$ est inversible donc si $[a]_m[b]_m = [0]_m$, en multipliant par l'inverse de $[a]_m$ on obtient $[b]_m = 0$ donc $[a]_m$ ne peut pas être un diviseur de 0.
- VRAI : c'est la contraposition de 1.

Q.69. *(p. 73).* Utilisons le même raisonnement que dans l'Exemple 8.2. Soit x le numéro original, x' le numéro erroné, et k la position où il y a une erreur. Alors

$$x - x' = (c_k - c'_k)10^k$$

Si x' était un numéro valable, nous aurions $[x]_{97} - [x']_{97} = [0]_{97}$ donc $([10]_{97})^k \left([c_k]_{97} - [c'_k]_{97}\right) = [0]_{97}$ or $[10]_{97}$ est inversible donc $[c_k]_{97} = [c'_k]_{97}$ donc $c_k = c'_k$ ce qui est exclus.

Q.70. *(p. 73).* 257 est un nombre premier (voir **Q.48**) donc $\varphi(257) = 256$; il y a 256 éléments inversibles dans $\mathbb{Z}/257\mathbb{Z}$ (tous les éléments sauf $[0]_{257}$).

Q.71. *(p. 74).* Oui et l'inverse est $[21]_{122}$.

Q.72. *(p. 74).* La forme réduite de $[143]_{122}$ est $[143]_{122} = [21]_{122}$ donc $[143]_{122}$ est inversible et $([143]_{122})^{-1} = ([21]_{122})^{-1} = [93]_{122}$.

Q.73. *(p. 74).* Si x est solution, en multipliant par l'inverse de $[93]_{122}$ il vient :

$$x = [21]_{122} \cdot [40]_{122} = [21 \cdot 40]_{122} = [108]_{122}$$

Réciproquement, les mêmes calculs montrent que $[108]_{122}$ est solution.

Donc il y a une solution et une seule : $x = [108]_{122}$.

Q.74. *(p. 76).* Dans le premier cas 0, dans le deuxième 1.

Q.75. *(p. 76).* Chaque élément est son propre symétrique.

Q.76. *(p. 76).* Dans $(\mathbb{Z}/5\mathbb{Z} \times \mathbb{Z}/7\mathbb{Z}, +)$: 35 éléments ; dans $(\mathbb{Z}/5\mathbb{Z}^* \times \mathbb{Z}/7\mathbb{Z}^*, \cdot)$: $\varphi(5)\varphi(7) = 4 * 6 = 24$.

Q.77. *(p. 78).* f est l'application réciproque de ψ de l'Exemple 9.5 donc f est un isomorphisme donc $f(a \cdot b) = f(a) + f(b)$.

Q.78. *(p. 78).* $(\mathbb{Z}/4\mathbb{Z}, +)$ et $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$ sont isomorphes.
$(\mathbb{Z}/5\mathbb{Z}, +)$ a 5 éléments et tous les autres 4, donc $(\mathbb{Z}/5\mathbb{Z}, +)$ n'est isomorphe à aucun des autres.
$(\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ et $(\mathbb{Z}/2\mathbb{Z}^2, +)$ sont isomorphes

Q.79. *(p. 80).* Dans $(\mathbb{Z}/m\mathbb{Z}, +)$ c'est le plus petit entier $k \geq 1$ tel que $[k]_m = [0]_m$, la période est donc $k = m$.
Dans $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$ c'est le plus petit entier $k \geq 1$ tel que $[1]_m^k = [1]_m$, la période est donc $k = 1$.

Q.80. *(p. 81).* Les nombres de 1 à 9 premiers avec 10 sont $1, 3, 7$ et 9 donc $\varphi(10) = 4$. Nous pouvons appliquer le théorème d'Euler car 7 est premier avec 10. Calculons $([7]_{10})^4$:

$$([7]_{10})^2 = [49]_{10} = [9]_{10}$$
$$([7]_{10})^4 = ([9]_{10})^2 = [81]_{10} = [1]_{10}$$

Q.81. *(p. 82).* Le nombre $p = 7$ est premier donc nous pouvons appliquer le thèorème de Fermat. Calculons $([10]_7)^7$ (nous laissons tomber les crochets ; ainsi $10 = 3$ signifie $[10]_7 = [3]_7$) :

$$10 = 3$$
$$3^2 = 2$$
$$3^4 = 2^2 = 4$$
$$3^3 = 3^2 \cdot 3 = 2 \cdot 3 = 6$$
$$10^7 = 3^7 = 4 \cdot 6 = 24 = 3 = 10$$

donc nous avons bien $([10]_7)^7 = [10]_7$.

Q.82. *(p. 85).* Non. Il suffit de trouver un contre-exemple. Soit $f : \mathbb{N} \to \mathbb{N}$, $x \mapsto 2x$. L'application f est injective mais pas surjective (seuls les entiers pairs ont un antécédent).
La réciproque est aussi fausse. Considérons par exemple $g : \mathbb{N} \to \mathbb{N}$, $x \mapsto \lfloor x/2 \rfloor$. L'application g est surjective, mais pas injective.

Q.83. *(p. 85).* $\varphi(35) = \varphi(5)\varphi(7) = 4 \cdot 6 = 24$.

Q.84. *(p. 88).* Ce sont bien trois nombres premiers ; 17 n'est pas sûr car $17 = 2 \times 8 + 1$ et 8 n'est pas premier. Par contre 83 et 107 sont sûrs car $p = 41 \times 2 + 1$, $q = 2 \times 53 + 1$ et 41 et 53 sont des nombres premiers.

Q.85. *(p. 88).* Soient $p = 1 + 2p', q = 1 + 2q'$ avec p', q' premiers ; donc $\text{ppcm}(p-1, q-1) = 2p'q'$. Les facteurs premiers de e ne sont pas 2 (car e est impair) ni p' (car $p' > e$) ni q' (car $q' > e$) ; e et $2p'q'$ n'ont aucun facteur premier en commun et sont donc premiers entre eux.

Q.86. *(p. 92).* C'est un code en bloc si nous considérons k comme fixé. Les mots de code sont constitués de k chiffres arbitraires et de 2 chiffres de contrôle, donc $n = k + 2$. Il y a 10^k mots de code possibles. Le rendement est donc

$$r = \frac{1}{k+2} \log_{10}(10^k) = \frac{k}{k+2}$$

Q.87. *(p. 93).* Le code peut détecter toutes les erreurs simples (Question 69), donc deux mots de code distincts ne peuvent pas différer d'un seul bit et la distance minimale est ≥ 2.

Par ailleurs, il existe des mots de code dont la distance de Hamming est 2. Il suffit de prendre pour x un mot de code commençant par 00 et pour y le mot obtenu en remplaçant les deux premiers chiffres 00 par 97. Alors $[x]_{97} = [y]_{97}$, donc puisque x est un mot de code, y l'est aussi.

Donc la distance minimale de ce code est 2.

Q.88. *(p. 95).* Par définition du canal à effacement, tous les effacements sont détectables, quel que soit leur poids.

Q.89. *(p. 95).* La distance minimale est 2 donc on peut corriger tous les effacements simples.

Les effacements doubles ne peuvent pas tous être corrigés, par exemple si l'effacement a agi sur deux positions contiguës et si les chiffres originaux étaient 00, on ne peut pas savoir si les chiffres originaux étaient 00 ou 97.

Q.90. *(p. 96).* La distance minimale est 2 donc on peut détecter toutes les erreurs simples.

Par contre, on ne peut pas être assuré de pouvoir corriger les erreurs, même simples.

Cependant, nous avons vu que l'on peut détecter certaines erreurs doubles, par exemple celles qui consistent à intervertir deux chiffre contigus.

Q.91. *(p. 98).* Toutes les erreurs simples, c'est à dire de poids 1.

Q.92. *(p. 99).* $n - rn = 2$ donc la borne donne $d_{\min}(\mathcal{C}) \leq 3$; nous savons que $d_{\min}(\mathcal{C}) = 2$, donc la borne n'est pas atteinte.

Q.93. *(p. 101).* Oui, en particulier tout nombre rationnel sauf 0 a un inverse qui est aussi un nombre rationnel.

Q.94. *(p. 102).* Non, nous avons vu par exemple que $(\mathbb{Z}/4\mathbb{Z}, +)$ et $(\mathbb{Z}/2\mathbb{Z}^2, +)$ ont tous deux 4 éléments et ne sont pas isomorphes.

Q.95. *(p. 102).* Non, bien qu'ils aient le même nombre d'éléments, car $\mathbb{Z}/4\mathbb{Z}$ n'est pas un corps. En effet $[2]_4$ n'est pas inversible alors qu'il est non nul.

Q.96. *(p. 102).* Non, car 15 n'est pas une puissance d'un nombre premier (il a deux facteurs premiers, 3 et 5).

Q.97. *(p. 103).* Soit une colonne de la table du groupe, correspondant à l'élément a. L'application $G \to G$, $x \mapsto a \star x$ est une bijection car l'équation en x : $a \star x = y$ possède toujours une solution unique $x = y \star x'$ où x' est le symétrique de x. Les éléments de la colonne de a sont les images de cette application, donc chaque élément de G s'y trouve une fois et une seule. Idem pour les lignes.

Q.98. *(p. 105).* Une combinaison linéaire se réduit ici à $\lambda\vec{a} = \vec{0}$. Si $\vec{a} \neq \vec{0}$ alors il faut que $\lambda = 0$; donc si $\vec{a} \neq \vec{0}$ la suite formée d'un seul vecteur $\vec{a}$ est linéairement indépendante.

Par contre si $\vec{a} = \vec{0}$, la combinaison linéaire $1\vec{a}$ est nulle alors que le coefficient est non nul, donc la suite formée du seul vecteur $\vec{0}$ est linéairement dépendante.

Q.99. *(p. 106).* C'est un espace vectoriel de dimension 1 sur un corps de cardinal 7, donc le cardinal de $\mathcal{S}$ est $7^1 = 7$.

Q.100. *(p. 107).* Notons que $\mathbb{Z}/7\mathbb{Z} = \mathbb{F}_7$ et que cette équation est l'équation (12.1) qui définit $\mathcal{S}'$. Donc le nombre de solutions est le nombre d'éléments de $\mathcal{S}'$. Comme c'est un espace vectoriel de dimension 2, son cardinal est $7^2 = 49$. Il y a 49 solutions.

Q.101. *(p. 109).* $\mathcal{C}$ comporte 8 éléments donc (Théorème 12.2) sa dimension est 3. Ou bien : les vecteurs $((\vec{v}_1, \vec{v}_2, \vec{v}_3)$ sont linéairement indépendants donc ils constituent une base de $\mathcal{C}$, donc $\dim(\mathcal{C}) = 3$.

Q.102. *(p. 109).* L'alphabet est l'ensemble des 10 chiffres décimaux, qui ne peut pas être un corps car il n'existe pas de corps à 10 éléments.

Q.103. *(p. 110).* $d_{\min}(\mathcal{C}) \leq 2$ donc la borne de Singleton est atteinte.

Q.104. *(p. 111).* $d_{\min}(\mathcal{C}) \leq n$ donc la borne de Singleton est atteinte.

Q.105. *(p. 111).* $\mathcal{C}$ est de dimension 3 puisqu'il possède une base de 3 vecteurs, $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$. Notons que $\vec{e}_1, \vec{e}_2$ et $\vec{e}_3$ sont dans $\mathcal{C}$ car ils sont combinaisons linéaires d'éléments de $\mathcal{C}$.

Comme $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$ est aussi constituée de 3 vecteurs, il suffit de montrer, au choix, soit que ces vecteurs sont linéairement indépendants, soit qu'ils engendrent $\mathcal{C}$. Montrons qu'ils engendrent $\mathcal{C}$.

Comme $\mathcal{C}$ est engendré par $(\vec{v}_1, \vec{v}2, \vec{v}_3)$, il suffit de montrer que $\vec{v}_i$, pour $i = 1, 2, 3$, est combinaison linéaire de $\vec{e}_1, \vec{e}_2$ et $\vec{e}_3$. Or, par un peu de calcul nous obtenons facilement :

$$\begin{aligned} \vec{v}_1 &= \vec{e}_3 \\ \vec{v}_2 &= \vec{e}_2 + \vec{e}_3 \\ \vec{v}_3 &= \vec{e}_1 + \vec{v}_2 = \vec{e}_1 + \vec{e}_2 + \vec{e}_3 \end{aligned}$$

Q.106. *(p. 112).* Le mot 000 est toujours encodé par 0000000.

Q.107. *(p. 115).* Non, car s'il n'y a pas d'erreur le syndrome est nul.

Q.108. *(p. 121).* Il faut vérifier que les lignes de H sont orthogonales aux lignes de G.

Q.109. *(p. 121).* Par exemple : un code de Reed-Solomon sur $\mathbb{F}_7$ de longueur $n = 7$ et de dimension $k = 3$.

Q.110. *(p. 122).* Oui, $g = 3$ est aussi un générateur car ses puissances sont $\{3, 4, 2, 1\}$.

Q.111. *(p. 122).* Vérifions tout d'abord que la définition a un sens, c'est à dire que g^k ne dépend que de $[k]_4$. Cela vient du fait que g est de période 4, donc si $[k']_4 = [k]_4$ alors $k = k' + 4\lambda$ avec $\lambda \in \mathbb{Z}$ donc

$$g^k = g^{k'+4\lambda} = \left[g^4\right]^\lambda g^{k'} = 1 \cdot g^{k'} = g^{k'}$$

Ensuite, cette application est surjective car les puissances de g donnent tous les éléments non nuls de $\mathbb{F}_5$, donc elle est bijective.

Enfin, $g^{k+k'} = g^k g^{k'}$ donc c'est un isormorphisme.

Q.112. *(p. 125).* Supposons que $f(X) = X^2 + X + 1$ ne soit pas irréductible. Alors on peut le factoriser en deux polynômes de degré ≥ 1 ; comme la somme des degrés des facteurs vaut 2 (le degré de $f(X)$), ces deux facteurs sont de degré 1, donc sont soit X soit $1 + X$; dans tous les cas, $f(X)$ aurait une racine. Or

$f(0) = f(1) = 1$ donc f n'a pas de racine : contradiction. Donc $f(X)$ est irréductible.

Index