Cours/TD 2 Codage "proche de l'entropie"

1.1 Codage Shannon-Fano

Definition (Rappel de **l'entropie** de la source)**.** *Soit une source discrete et sans mémoire produisant (indépendament) des symboles (lettres/mots) d'un alphabet* $a_0, a_1, \cdot, a_{N-1}$ *selon la distribution de probabilité* $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$.
L'entropie d'un mot $I(w)$ *est la quantité d'information contenue dans un mot* w *produit par la source.* $I(w) = log_2(\frac{1}{p(w)}) = -log_2 p(w)$.
L'entropie d'une source est **l'information moyenne** *par symbole venant de la source (en bits par symbole).* $H(\mathbf{p}) \equiv p_0 I_0 + p_1 I_1 + \ldots + p_{N-1} I_{N-1} = -p_0 \log_2 p_0 - p_1 \log_2 p_1 - \ldots - p_{N-1} \log_2 p_{N-1}$

On a vu que l'information fonctionne comme un multiplicateur pour un codage binaire virtuel : Chaque lettre a_j produite (statistiquement correct) par la source "vaut" $I(a_j)$ bits, $0 \leq j \leq N-1$.

Longueur du codage quand les probabilités sont des puissances de 2

Soit l_j la longueur (i.e. no. de bits) du mot code associé au symbol a_j , $0 \leq j \leq N-1$. Considérons la longueur moyenne $\bar{l}$ des mots code : $\bar{l} = p_0 l_0 + \cdot + p_{N-1} l_{N-1}$. Si les probabilités des symboles sont des puissances (négatives) de 2, les valeurs de l'information sont des entiers et donc peuvent être exactement des longueurs de bits $l_j = I(a_j]$. Donc $\bar{l} = H(\mathbf{p})$, $\bar{l}$ est un multiplicateur qui transforme 1000 symboles produits par la source (selon les statistiques établies) en $1000 \times \bar{l}$ bits.

Idée naturelle (Shannon et Fano) : associer aux symboles de notre alphabet des mots code binaires dont la longueur est égale au contenu d'information des symboles à coder : $l_j = \lceil I(a_j) \rceil = \lceil -\log_2 p_j \rceil$***, où*** $\lceil \ \rceil$ ***signifie l'opération d'arrondi vers l'entier supérieur.***

Pour un symbole avec une probabilité d'apparition $p_j = \frac{1}{2^{n_j}}$ qui est une puissance binaire, ce contenu d'information $-\log_2 p_j = n_j$ est simplement le nombre de bits necessairés au

développement binaire (ou "dyadique") de cette probabilité.

L'algorithme de Fano est basé sur l'interprétation géométrique du développement binaire d'une puissance negative de 2 dans $[0,1[$: diviser $[0,1[$ dans deux sous-segments (ou un sous intervalles) jusqu'au moment où on retrouve cette puissance $p_j = \frac{1}{2^{n_j}}$ en tant que borne inférieure d'un sous intervalle. Son développement binaire a autant de bits que le "contenu d'information de l'intervalle" $-\log_2 p_j$ le dicte. Et comme cela on retrouve :

Algorithme de Fano

1. ordonner les symboles de l'alphabet sur une colonne par probabilité décroissante.
2. diviser l'ensemble des symboles en deux sous-ensembles de probabilités cumulées presque égales $\geq \frac{1}{2}$, $\leq \frac{1}{2}$
3. coder avec "0" (resp. "1") les éléments du premier (resp. deuxième) sous-ensemble
4. continuer de manière récursive jusqu'au moment où tous les sous-ensembles ne comportent plus qu'un élément

Fano construit un arbre binaire de manière descendante. Diviser de manière recursive l'ensemble des symboles en deux sous-ensembles de probabilités cumulées peut être vu comme une dichotomie recursive pour obtenir des subdivisions de l'intervalle $[0,1[$.

On notera que

- dans 1. l'ordonnancement des probabilités dans un ordre décroissant est équivalent à celui des valeurs de l'information dans un ordre croissant parce que rappelez-vous $I(\cdot)$ est une fonction décroissante
- dans 2. quand les probabilités ne sont pas des puissances de 2, à une étape donnée dans le calcul récursif on peut avoir deux choix possibles de regroupement d'éléments. Les mots du code binaire ainsi obtenus n'ont plus la longueur égale à leur contenu d'information.

Exemple. *Considérons une source discrète sans mémoire qui a produit* $aabbbbcddefffffgh$ *sur l'alphabet* $\{a,b,c,d,e,f,g,h\}$.

Après une analysé des fréquences nous avons $\mathbf{p} = (\ p_0, p_1, \cdots, p_7\ )$ avec $p_a = p_d = \frac{2}{16} = \frac{1}{8}$, $p_b = p_f = \frac{4}{16} = \frac{1}{4}$, $p_c = p_e = p_g = p_h = \frac{1}{16}$. On a donc : $I(a) = I(d) = 3$ $I(b) = I(f) = 2$ $I(c) = I(g) = 4$. Notre codage :

b	4	0	0			
f	4	0	1			
a	2	1	0	0		
d	2	1	0	1		
c	1	1	1	1	0	0
e	1	1	1	1	0	1
g	1	1	1	1	1	0
h	1	1	1	1	1	1

Le déséquilibre de la loi de probabilité avec une loi uniforme indique qu'une compression de l'information est possible. Avec un codage *sans statistiques* supposant le cas équiprobable, chacune des huit lettres serait codée en $\log_2 N = log_2 8 = 3$ bits. Avec le codage Fano on obtient $\overline{l} = H(\text{ı}) = 2(3/8 + 2/4 + 8/16) = (3 + 4 + 4)/4 = 11/4 = 2.75$. Les mots du code binaires Fano ont la longueur égale à leur contenu d'information.

Exemple. *Considérons une source discrète sans mémoire qui produit des symboles avec la loi de probabilité* $\mathbf{p} = (\ p_0, p_1, \cdots, p_7\ ) = \{0.4, 0.2, 0.15, 0.1, 0.05, 0.05, 0.05\}$ *sur l'alphabet* $\{a, b, c, d, e, f, g\}$.

Deux choix sont possibles :

a	00	0
b	01	100
c	100	101
d	101	110
e	1100	1110
f	1101	11100
g	111	11111

Le codage idéal donné par l'entropie sera $\overline{l} = H(\text{ı}) = 2.38$, tandis que avec un codage sans statistiques supposant le cas équiprobable, chacune des sept lettres serait codée en $\log_2 N = log_2 7 = 2.8$ bits. On obtient comme longueur moyenne du codage 2.5 et respectivement 2.45. Le dernier code a des mots les plus longs mais il exploite mieux le déséquilibre de la loi de probabilité.

Un algorithme similaire découvert de manière indépendante par Shannon permet de trouver un code binaire préfixe unique. Le codage Shannon, et celui de Fano ont été remplacées par celui de Huffman qui propose des codes plus proches de l'entropie. Mais, le codage du Shannon est très intéressant d'un point de vue théorique parce que il anticipe le **codage arithmétique** qui est une méthode de compactage utilisé dans certains types de JPEG – voir l'annexe 1.5.

1.2 Propriétés des codes. Code préfixe

Le minimum qu'on doit exiger d'un code est son caractère uniquement déchiffrable. Une condition suffisante pour qu'un code soit uniquement déchiffrable est qu'il soit préfixe, c'est-à-dire qu'il ne comporte aucun mot qui soit le début d'un autre mot.

Exemple. Code ambigu *Soit le codage* $\{a \longmapsto 0, b \longmapsto 01, c \longmapsto 10\}$. *Le décodage de la séquence* 010 *est ambigu, i.e. il y a plusieurs possibilités : ac , ba – voir figure 1.1.*

Les codes Fano sont des codes préfixe, donc uniquement déchiffrables – voir figure 1.2. Cependant, il y a des codes qui ne sont pas préfixe mais ils sont uniquement déchiffrables

Exemple. Code uniquement déchiffrable mais pas préfixe *Soit le codage* $\{a \longmapsto 0, b \longmapsto 01\}$. *Le décodage de la séquence* 010 *s'effectue sans ambiguïté, mais à condition de lire les symboles deux par deux. On dit alors que le code n'est pas instantané – voir figure 1.3.*

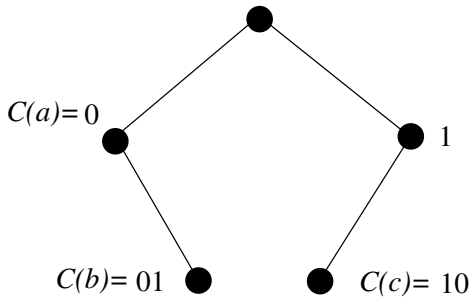

FIG. 1.1 – Code ambigu

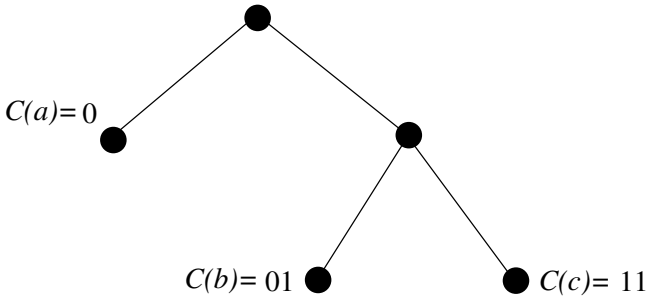

FIG. 1.2 – Code préfixe

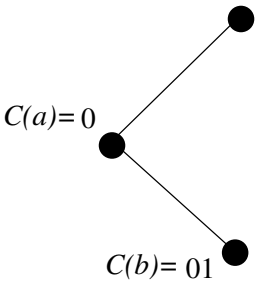

FIG. 1.3 – Code uniquement déchiffrable mais pas instantané ni préfixe

On peut, sans perte de généralité, se restreindre à la classe des codes préfixes parce que tout code uniquement déchiffrable possède un code préfixe équivalent. Ce résultat est basé sur le travail de Mc-Millan et de Kraft.

1.3 Arbre d'un code. Inégalité de Kraft

Sous quelles conditions existe-t-il un code binaire préfixe qui réalise les longueurs $(l_0, l_1, \ldots, l_{N-1})$ ***?***

En s'appuyant sur le fait que un codage binaire peut être représenté de manière graphique par un arbre binaire nous prouvons le théorème de Kraft qui nous donne la réponse. Dans un arbre binaire, chaque nœud différent de la racine a un père et peut avoir deux nœuds fils, le lien entre deux nœuds s'appelle une branche et un nœud qui n'a pas de fils, i.e de descendance, s'appelle une feuille.

Règles de correspondance
- chaque mot est un entier dans l'écriture binaire
- l'écriture de chaque entier binaire se fait en partant de la racine
- un déplacement à gauche correspond à un "0"
- un déplacement à droite correspond à un "1".
- chaque déplacement crée un nœud de l'arbre.

Dans cette correspondance *un code préfixe est un code dont les symboles codés sont des feuilles. Les mots code associes aux symboles sont les chemins de la racine vers les feuilles.*

Théorème. (Théorème de Kraft) *Une condition nécessaire et suffisante pour qu'un code baire puisse être transformé (en effectuant des permutations sur les b lettres formant les mots code) en un code préfixe équivalent (possédant la même distribution de longueur des mots) est que l'inégalité de Kraft soit satisfaite.*

$$\sum_{k=0}^{N-1} b^{-l_k} \leq 1 \tag{1.3.1}$$

Exemple. *Soit une source discrete et sans mémoire produisant des symboles d'un alphabet* $a_0, a_1, \cdot, a_{N-1}$ *selon la distribution de probabilité* $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$. *Les longueurs proposés par Shannon vérifient l'inégalité de Kraft.*

Nous avons $l_j = \lceil I(a_j) \rceil = \lceil -\log_2 p_j \rceil \qquad 0 \leq j \leq N-1$ où $\lceil\ \rceil$ signifie l'opération d'arrondi vers l'entier supérieur. En réécrivant :
$l_j - 1 < -\log_2 p_j \leq l_j, 0 \leq j \leq N-1 \Leftrightarrow 2^{-l_j} \leq p_j < 2 \cdot 2^{-l_j}, 0 \leq j \leq N-1$. En sommant sur tous les termes on obtient :

$$\sum_{j=0}^{N-1} 2^{-l_j} = \frac{1}{2^{l_0}} + \frac{1}{2^{l_1}} + \ldots + \frac{1}{2^{l_{N-1}}} \leq 1 \tag{1.3.2}$$

Exemple. *Soit le code* $C = 10, 11, 000, 101, 111, 1100, 1101$. *Trouver le code préfixe equivalent.*

Ce code binaire vérifie l'inégalité binaire de Kraft. La longueur maximale des mots est 4. Donc l'arbre devra comporter 4 niveaux. Le code initial se compose de :

- 2 mots de longueur 2, il faut donc réserver 2 nœuds au 2ième niveau
- 3 mots de longueur 3, il faut donc réserver 3 nœuds au 3ième niveau
- 2 mots de longueur 4, il faut donc réserver 2 nœuds au 4ième niveau

Un des codes ainsi obtenu est : $01, 10, 000, 001, 111, 1100, 1101$

Théorème. (Théorème de Mac-Millan) *Un code uniquement déchiffrable vérifie l'inégalité de Kraft.*

En associant les deux théorèmes précédents, on conclut que tout code uniquement déchiffrable possède un code préfixe équivalent.

Exercice. *Soit un ensemble à coder à* $N = 12$ *symboles. Votre code binaire préfixe a déjà 4 mots de longueur 3, et 6 mots de longueur 4. Combien de mots de longueur 5 pouvez-vous ajouter ?*

La conséquence la plus importante de la caractérisation des codes préfixes à l'aide de l'inégalité de Kraft est qu'on peut voir *l'entropie d'une source comme la valeur limite idéale de toute compression sans perte d'information.*

Théorème. (Théorème du codage Shannon-Fano) *Considérons une source sans mémoire produisant les* N *symboles selon la distribution de probabilité* $\mathbf{p} = (\ p_0, p_1, \ldots, p_{N-1}\ )$. *Soit* C *un code binaire préfixe quelconque associé, et soit* $\bar{l} = \sum_{j=0}^{N-1}\ p_j l_j$ *la longueur moyenne des mots code (en bits par symbole).*

Alors : $H(\mathbf{p}) \leq \bar{l}$.

En plus, pour les codes binaires préfixes de Shannon on a : $\bar{l} < H(\mathbf{p}) + 1$.

Preuve. $H(\mathbf{p}) - \bar{l} \ = \ - \ \sum_{j=0}^{N-1}\ p_j \log_2 p_j \ - \ \sum_{j=0}^{N-1}\ p_j l_j \ = \ \frac{1}{\ln 2} \cdot \sum_{j=0}^{N-1}\ p_j \ln(\frac{2^{-l_j}}{p_j})$.

Mais $\ln\ x \ \leq \ x - 1 \qquad$ *pour* $x \ > \ 0$, $\qquad$ *donc*

$H(\mathbf{p}) - \bar{l} \leq \frac{1}{\ln 2} \cdot \sum_{j=0}^{N-1} p_j(\frac{2^{-l_j}}{p_j} - 1) \ = \ \frac{1}{\ln 2} \cdot \sum_{j=0}^{N-1}\ (2^{-l_j} - p_j)$.

En utilisant l'inégalité de Kraft, on a : $\qquad \sum_{j=0}^{N-1}\ (2^{-l_j} - p_j) \ \leq \ 0$. *q.e.d.*

Pour prouver la deuxième inégalité on utilise la longueur Shannon du mot code, i.e. son entropie $l_j - 1 < -\log_2 p_j \leq l_j$, *pour* $0 \leq j \leq N-1$. *Donc :* $\qquad \sum_{j=0}^{N-1}(p_j l_j - p_j) \ < \ -\sum_{j=0}^{N-1} p_j \log_2 p_j$ *i.e.* $\bar{l} \ < \ (H\mathbf{p}) \ + \ 1$.

1.4 Annexe : Théorème de Kraft

Théorème. (Théorème de Kraft) *Une condition nécessaire et suffisante pour qu'un code b-aire puisse être transformé (en effectuant des permutations sur les b lettres formant les mots code) en un code préfixe équivalent (possédant la même distribution de longueur des mots) est que l'inégalité de Kraft soit satisfaite.*

$$\sum_{k=0}^{N-1} b^{-l_k} \leq 1 \tag{1.4.3}$$

Ébauche. "nécessaire"

Soit un alphabet à coder qui contient N symboles auxquels sont attribués des codes binaires de longueur variable $\{c_0, \ldots, c_{N-1}\}$. L'arbre associé au code est un sous-arbre de l'arbre B de tous les mots binaires. Dans B on peut prover par induction qu'il y a 2^N codes possibles (nombre de feuilles dans l'arbre du code). Supposons que le code c_k du codage binaire a la longueur l_k. Il se trouve donc à la hauteur l_k dans l'arbre associe au code, rangé selon sa valeur numérique. Les successeurs d'un nœud dans l'arbre binaire sont précisément les successeurs syntaxiques du mot code correspondant au node (i.e. les mots dont ce mot est un préfixe). Pour que le codage $\{c_0, \ldots, c_{N-1}\}$ soit préfixe il faut que ce nœud devienne une feuille de l'arbre du code. Cela interdit au codage les nœuds du sous-arbre ayant c_k comme racine. Le nombre de nœuds de B interdits dans l'arbre du code sera donc de 2^{N-l_k}. Ils seront élagués pour obtenir l'arbre associe au code. Le nombre total de feuilles interdites sera donc $\sum_{k=0}^{N-1} 2^{N-l_k}$. Le nombre de feuilles interdites est inférieur ou égal au nombre de feuilles de l'arbre binaire complet. Donc

$$\sum_{k=0}^{N-1} 2^{N-l_k} \leq 2^N \tag{1.4.4}$$

En divisant chaque membre de la relation par 2^N, nous obtenons l'inégalité binaire de Kraft.

"suffisante"

Pour construire le code préfixe équivalent ayant la même distribution des longueurs, on peut utiliser un arbre dont la profondeur correspond à la longueur maximale des mots. Supposons que $l_k \leq l_{k+1}, 0 \leq k < N-1$. Soit l'arbre de tous les mots binaires B. Soit n_k les mots code de longueur l_k. Pour chacun on construit dans B un chemin (succession d'arêtes) de longueur correspondante l_k, débutant à la racine. On réduit B en enlevant toute la descendence des nœuds fin de chemins, qui deviennent ensuite des feuilles. L'inégalité de Kraft étant satisfaite, le choix au niveau k est possible, après avoir bloqué les $n_1 \cdot 2^{l-1}$ successeurs du choix au niveau 1, les $n_2 \cdot 2^{l-2}$ successeurs du choix au niveau 2, ... , les $n_{k-1} \cdot 2$ successeurs du choix au niveau $k-1$.

Le raisonnement peut être généralisé pour un codage avec b symboles et nous obtenons

l'inégalité de Kraft :

$$\sum_{k=0}^{N-1} b^{-l_k} \leq 1 \tag{1.4.5}$$

Exercice. *Considérons un alphabet de 4 lettres : N, E, S, W. Existe-t-il un code préfixe sur cet alphabet qui consiste en 2 mots de longueur 1, 4 mots de longueur 2, 10 mots de longueur 3 et 16 mots de longueur 4 ?*

Exercice. *Une source sans mémoire produit les huit lettres a, b, c, d, f, g,h selon la distribution de probabilité* $\mathbf{p} = \{p(a) = \frac{27}{64}, p(b) = p(cC) = \frac{3}{16}, p(d) = \frac{1}{16}, p(e) = p(f) = \frac{3}{64}, p(g) = \frac{1}{32}, p(h) = \frac{1}{64}\}$

- *Calculer la quantité d'information dans chaque lettre, puis l'entropie* $H(\mathbf{p})$ *de la source.*
- *En suivant l'idée de Shannon, trouver avec l'algorithme de Fano un code binaire préfixe associé à* .
- *Calculer la longueur moyenne* $\bar{l}$ *des mots code, et comparer à* $H(\mathbf{p})$.

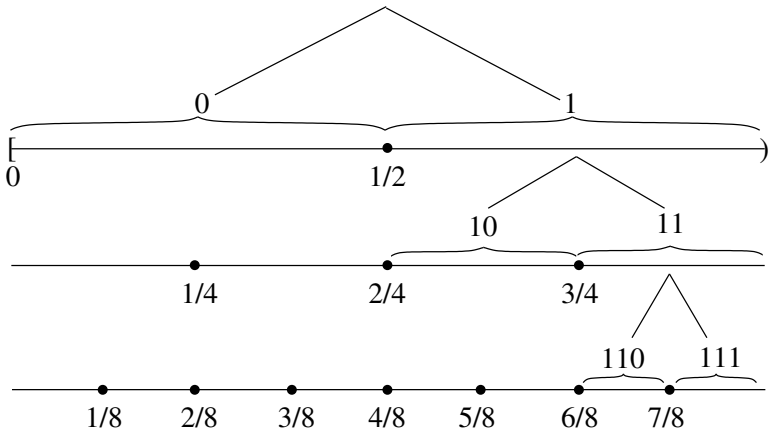

FIG. 1.4 – Arbre partiel de dichotomie binaire réitéré de $[0, 1)$ pour $p = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right)$

1.5 Annexe : De l'algorithme de Fano et Shannon au codage arithméthique

Que se passe-t-il si les probabilités ne sont pas des puissances de 2 ?

Soit une source discrete et sans mémoire produisant (indépendament) des symboles (lettres/mots) d'un alphabet $a_0, a_1, \cdot, a_{N-1}$ selon la distribution de probabilité $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$. Supposons que les symboles de l'alphabet sont déjà ordonnés par probabilité décroissante. $p_0 \geq p_1 \geq \cdots \geq p_{N-1}$.

Idée naturelle (Shannon et Fano) était d'associer aux symboles de notre alphabet des mots code binaires dont la longueur est égale au contenu d'information des symboles à coder.

Pour un symbol a_j avec une probabilités d'apparition p_j qui n'est pas une puissance binaire ce contenu d'information est $I(a_j) = -\log_2 p_j$ et Shannon demande que la longueur de bits necessairés soit $l_j = \lceil I(a_j) \rceil = \lceil -\log_2 p_j \rceil$.

L'interprétation géométrique de l'algorithme de Fano :

Par la suite d'une dichotomie binaire réittérée sur [0, 1 [l'algorithme de Fano trouvait une partition d'intervalles. La dichotomie d'un interval s'arrêtait quand l'intervalle $[A_j, A_{j+1}[$ avait la longueur égale à la probabilité 2^{-l_j} de (la production de) du symbol a_j (i.e. un seul element par intervalle). Le mot code était le développement binaire fini de la borne inférieure A_j de cet intervalle. La précision de ce mot était exactement $l_j = -\log_2 p_j$ – voir figure 1.4.

L'algorithme simplifié de Fano peut être lui aussi generalisé mais le code produit n'est pas unique, ni optimal.

1. ordonner les symboles de l'alphabet sur une colonne par probabilité décroissante.
2. diviser l'ensemble des symboles en deux sous-ensembles de probabilités cumulées presque égales.
3. coder avec "0" (resp. "1") les éléments du premier (resp. deuxième) sous-ensemble
4. continuer de manière récursive jusqu'au moment où tous les sous-ensembles ne comportent plus qu'un élément

D'une manière plus generale l'algorithme de Shannon associe aussi d'une manière biunivoque :

– aux symboles de l'alphabet une partition d'intervalles dans $[0, 1[$
– à un symbol un mot code qui est un nombre réel dans l'intervalle correspondant.

L'interprétation géométrique de l'algorithme de Shannon

est de trouver une partition d'intervalles avec la propriété que la longueur de chaque intervalle $[A_j, A_{j+1}[$ est égale à la probabilité de (la production de) du symbol a_j : $p_j = A_{j+1} - A_j$, $0 \leq j \leq N-1$.

$A_0 = 0$
$A_1 = p_0$
$A_2 = p_0 + p_1$
$A_3 = p_0 + p_1 + p_2$
$\vdots$
$A_N = p_0 + p_1 + \ldots + p_{N-1} = 1$

Un mot code c_j est associe à l'intervalle $[A_j, A_{j+1}[\subset [0, 1[$ de la manière suivante :

$$c(A_j, A_{j+1}) = \alpha_1\alpha_2\ldots\alpha_l \iff A = 0.\alpha_1\alpha_2\ldots\alpha_l* \text{ (début du développement binaire du nombre réel } A_j \text{), avec } l_j = \lceil -\log_2(p_j)\rceil.$$

On prend autant de bits de ce développement que le "contenu d'information de l'intervalle" l_j le dicte.
Maintenant pour assurer la correspondance biunivoque il faut prouver que le mot code associe a un intervalle $[A_j, A_{j+1}[$ est unique. Plus on peut montrer que :

Lemme. *Le code Shannon est préfixe, i.e. le mot code* $c_j = c(A_j, A_{j+1})$ *ne peut pas être un préfixe du mot* $c_{j+1} = c(A_{j+1}, A_{j+2})$, *pour* $0 \leq j \leq N-2$.

Preuve. *Supposons que le codage binaire de* A_j *a la longueur* $l_j = \lceil -\log_2 p_j \rceil$.
Réécrivons les conditions de Shannon :

$l_j - 1 < -\log_2 p_j \leq l_j, 0 \leq j \leq N-1 \Leftrightarrow 2^{-l_j} \leq p_j < 2 \cdot 2^{-l_j}, 0 \leq j \leq N-1.$

Par construction $A_k \geq A_{j+1} = A_j + p_j \geq A_j + \frac{1}{2^{l_j}}$. *Donc* $A_k - A_j \geq \frac{1}{2^{l_j}}$, *i.e. leurs développements binaires sont différents sur les premiers* l_j *positions. Par consequence le mot code* c_j *n'est pas un préfixe du mot code* c_k , $j < k$.

Remarque. *On note que on peut dire aussi que le codage Shannon-Fano associe au symbol a_j le bits du plus petit (i.e. celui avec moins de bits) nombre binaire rationnel de l'interval $[A_j, A_{j+1}[$.*

Algorithme de Shannon

1. ordonner les symboles de l'alphabet sur une colonne par probabilité décroissante.
2. calculer $A_0 = 0$ et $A_{j+1} = A_j + p_j$
3. pour chaque $0 \leq j \leq N-1$ écrire le d´eveloppement dyadique $A = 2 \cdot A_j$ jusqu'au $l_j = \lceil -\log_2 p_j \rceil$ "décalage de la virgule "
   - si $A \geq 1$, alors $\alpha_1 = 1$ et $A = 2A - 1$
   - si $A \leq 1$, alors $\alpha_1 = 0$ et $A = 2A$
4. afficher $A_j = 0.\alpha_1\alpha_2\alpha_3\alpha_4 \ldots \alpha_{l_j}$

Exemple. *Développement binaire de* $A = \frac{1}{11}$.

$$A = 0.\alpha_1\alpha_2\alpha_3\alpha_4 \ldots$$

$$\begin{aligned}
2A &= \tfrac{2}{11} < 1 \implies \alpha_1 = 0 \\
4A &= \tfrac{4}{11} < 1 \implies \alpha_2 = 0 \\
8A &= \tfrac{8}{11} < 1 \implies \alpha_3 = 0 \\
16A &= \tfrac{16}{11} = 1 + \tfrac{5}{11} \implies \alpha_4 = 1 \\
A^{(4)} &= \tfrac{5}{11} = 0.\alpha_5\alpha_6\alpha_7 \ldots \\
2A^{(4)} &= \tfrac{10}{11} < 1 \implies \alpha_5 = 0 \\
4A &= \tfrac{20}{11} = 1 + \tfrac{9}{11} \implies \alpha_6 = 1 \\
A^{(6)} &= \tfrac{9}{11} = 0.\alpha_7\alpha_8\alpha_9 \ldots \\
2A^{(6)} &= \tfrac{18}{11} = 1 + \tfrac{7}{11} \implies \alpha_7 = 1 \\
A^{(7)} &= \tfrac{7}{11} = 0.\alpha_8\alpha_9\alpha_{10} \ldots \\
2A^{(7)} &= \tfrac{14}{11} = 1 + \tfrac{3}{11} \implies \alpha_8 = 1 \\
A^{(8)} &= \tfrac{3}{11} = 0.\alpha_9\alpha_{10}\alpha_{11} \ldots \\
2A^{(8)} &= \tfrac{6}{11} < 1 \implies \alpha_9 = 0 \\
4A^{(8)} &= \tfrac{12}{11} = 1 + \tfrac{1}{11} \implies \alpha_{10} = 1 \\
A^{(10)} &= \tfrac{1}{11} = 0.\alpha_{11}\alpha_{12}\alpha_{13} \ldots = A\ .
\end{aligned}$$

Le développement binaire (10-périodique) de $A = \frac{1}{11} = 0.\overline{0001011101}$