Théorie de l'Information
Notes de Cours (part 3)
2006-2007

SIC-SICOM

Maria-João Rendas

October 26, 2006

Contents

Chapter 3

Codes de Symbole

3.1 Définitions et Notation

Définition 1 *$\mathcal{A}^N$, $x^{(n)}$, $\mathcal{A}^+$, $\ell(x)$*
Soit $\mathcal{A}$ un ensemble discret et N un entier positif. Nous représentons par $\mathcal{A}^N$ l'ensemble des *N-tuples ordonnés* d'éléments de $\mathcal{A}$:

$$\mathcal{A}^N = \{(x_1 \cdots x_N), x_i \in \mathcal{A}\}.$$

Nous désignerons souvent les éléments de $\mathcal{A}^n$ par $x^{(n)}$:

$$x^{(n)} = x_1 \cdots x_n, \qquad x_i \in \mathcal{A}.$$

Nous représentons par $\mathcal{A}^+$ l'ensemble de toutes les *séquences de taille finie* avec des éléments dans $\mathcal{A}$:

$$\mathcal{A}^+ = \bigcup_n \mathcal{A}^n.$$

Soit $x \in \mathcal{A}^+$ une séquence de symboles appartenant à l'alphabet $\mathcal{A}$. Nous notterons par $\ell(x) > 0$ la *longueur* de la séquence x, de telle façon que

$$x = x_1 \cdots x_n \Rightarrow \ell(x) = n \Leftrightarrow x \in \mathcal{A}^{\ell(x)}.$$

Définition 2 *Codes de symbole (binaires), C*
Soient $X_i \in \mathcal{X}, i = 1, 2, \ldots$ les symboles émis par une source. Un code de symbole binaire C pour la source X est une application de $\mathcal{X}$ en $\{0,1\}^+$, qui fait correspondre à chaque élément de $\mathcal{X}$ une séquence binaire de taille finie:

$$\begin{array}{rccc} C: & \mathcal{X} & \rightarrow & \{0,1\}^+ \\ & x & \rightarrow & c(x) \end{array}$$

Nous notterons $C(\mathcal{X})$ l'ensemble de tous les mots de code associés aux possibles valeurs de la source:

$$C(\mathcal{X}) = \left\{ c \in \{0,1\}^+ \ : \ \exists x \in \mathcal{X}, c = c(x) \right\}.$$

△

Définition 3 *Code (binaire) étendu, C^+*
Soient $X_i \in \mathcal{X}, i = 1, 2, \ldots$ les symboles émis par une source et C un code de symbole binaire C pour la source X. Le code étendu C^+ est une application de $\mathcal{X}^+$ en $\{0,1\}^+$, qui fait correspondre à chaque séquence de symboles dans $\mathcal{X}$ une séquence binaire (nécessairement de taille finie) obtenue en *concatenant* le code des éléments de la séquence source :

$$\begin{array}{cccc} C^+ : & \mathcal{X}^+ & \longrightarrow & \{0,1\}^+ \\ & x = x_1 x_2 \cdots x_n & \to & c^+(x) = c(x_1)c(x_2)\cdots c(x_n). \end{array}$$

△

Définition 4 *Codes uniquement décodables*
Soit C un code (de symbole) pour la source $X \in \mathcal{X}$. C est uniquement décodable si C^+ est une application 1-to-1:

$$\forall x^{(n)}, y^{(m)} \in \mathcal{X}^+, \qquad x^{(n)} \neq y^{(m)} \Rightarrow c^+(x^{(n)}) \neq c^+(y^{(m)}).$$

△

Définition 5 *Codes de préfixe*
Soit C un code pour la source $X \in \mathcal{X}$. C est un code de préfixe si aucun mot de code n'est le début d'un autre mot de code:

$$\text{Si } c_2 = c_1\, c, \qquad \text{où} \qquad c_1 \in C(\mathcal{X}), c \in \{0,1\}^+ \Rightarrow c_2 \notin C(\mathcal{X}).$$

△

La propriété de préfixe garanti qu'un code est uniquement décodable:

$$C \text{ est un code de préfixe } \Rightarrow C \text{ est uniquement décodable .}$$

(Par construction, la terminaison de chaque mot de code $c(x_i)$ est bien définie.)

Remarque 1 *À un code (binaire) de préfixe C nous pouvons toujours associer un arbre binaire, où tous les mots de code se situent sur les* **feuilles** *de l'arbre.*

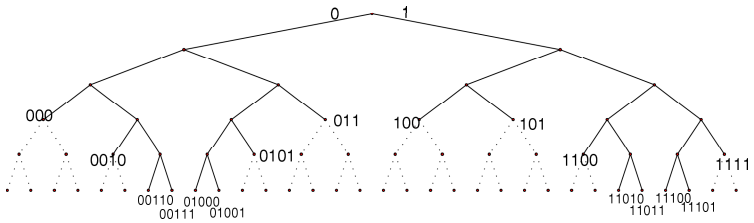

Figure 3.1: Association d'un code binaire à un arbre binaire.

La Figure 3.1 illustre un exemple de cette association entre codes (binaires) de préfixe et les arbres binaires. Dans cet exemple, l'ensemble des mots de code ($C(\mathcal{X})$) est

$$\begin{aligned} C(\mathcal{X}) = \{ \quad & 000, 011, 100, 101, \\ & 0010, 0101, 1100, 1111, \\ & 00110, 00111, 01000, 01001, 11010, 11011, 11100, 11101\} . \end{aligned}$$

La profondeur k de l'arbre binaire est égale à la longueur maximale des mots de code:

$$k = \max_{x \in \mathcal{X}} \ell(c(x)).$$

Les mots de code de longueur $n \leq k$ correspondent à des noeuds de l'arbre binaire au niveau n. À chaque niveau n, le nombre maximal de mots de code qui peuvent exister est

$$N_n = 2^n.$$

Lemme 1 Le nombre de séquences binaires distinctes de longueur inférieure ou égale à n est

$$M_n = \sum_{m=0}^{n} N_m = \sum_{m=0}^{n} 2^m = 2^{n+1} - 1. \tag{3.1}$$

△

Lemme 2 Dans un arbre binaire, le nombre de déscendants au niveau n d'un noeud du niveau $m \leq n$ est 2^{n-m}. △

Lemme 3 Dans un arbre binaire correspondant à un code C, si c_1 n'est pas un préfixe de c_2, alors les descendants de c_1 et de c_2 sont disjoints. △

Si le choix des branches descendantes est fait d'une façon consistante pendant le parcours de l'arbre (par exemple, la branche à gauche correspondant à un symbole "0" et la branche droite à un symbole "1", comme dans la Figure Figure 3.1), les séquences binaires correspondantes aux noeuds d'un même niveau n sont ordonnées (de gauche à droite) par l'ordre *lexicographique* héritée de la relation d'ordre dans l'alphabet binaire $(0 < 1)$ [1] :

$$c_1, c_2 \in \{0,1\}^n, \qquad c_1 > c_2 \Leftrightarrow \exists i \in [1,n] \ : \ c_1(j) = c_2(j), \forall j < i \wedge c_1(i) > c_2(i).$$

Nous pouvons généraliser cette relation d'ordre (totale) dans $\{0,1\}^n$ à l'ensemble $\{0,1\}^+$ de toutes les séquences binaires de taille finie de la façon suivante:
Soient c_1 et c_2 deux séquences consécutives, de même taille, et $c_2 > c_1$. Alors, toutes les continuations $c' = c_1 c$ de c_1 sont plus grandes que c_1 et inférieures à c_2 :

$$c' = c_1 c, c \in \{0,1\}^+, \ c_1 < c_2 \ \Rightarrow c' > c_1, \qquad c' < c_2.$$

La condition de préfixe implique qu'aucun autre mot de code peut être trouvé dans les sous-arbres qui ont comme racine un mot de code (les branches en pointillé dans la Figure 3.1).

Exemple 1 *Ordre lexicographique de séquences binaires*
Considérons dans un premier temps des séquences de même taille :

$$c_1 = 001101 \qquad \text{et} \qquad c_2 = 001011$$

Nous pouvons constater que $c_1 > c_2$. Prenons $i = 4$. Alors

$$c_1(j) = c_2(j), i = 1,2,3 \text{ (sous-séquence 001)} \quad \text{et :} \qquad c_1(4) = 1 > c_2(4) = 0.$$

Pour la généralisation à des séquences de tailles différentes. Nous venons de voir que $c_1 > c_2$. Alors, toutes les séquences $c' = c_2 c$ sont plus grandes que c_2, par exemple

$$c' = 00101100001 > c_2 = 001011 \, ,$$

et plus petites que c_1 :

$$c' = 00101100001 < c_1 = 001101 \, .$$

△

3.2 L'inégalité de Kraft-McMillan

L'inégalité de Kraft-McMillan impose une contrainte sur la **longueur des mots des codes uniquement décodables** :

[1] dans cette équation $c(i)$ représente le i-ème symbole de la s équence c

Théorème 1 *Inégalité de Kraft*
Soit $X \in \mathcal{X}$ une source et C un code de symbole (binaire) uniquement décodable pour X. Alors

$$\sum_{x \in \mathcal{X}} 2^{-\ell(c(x))} \leq 1. \tag{3.2}$$

△

Démonstration
Soit

$$\ell \geq \max_{x \in \mathcal{X}} \ell(c(x)).$$

Il existent 2^n séquences (binaires) distinctes de longueur ℓ. Chaque mot de code $c(x), x \in \mathcal{X}$ a un nombre de descendants égal à $2^{\ell - \ell(c(x))}$ (Lemme 2). Comme les ensembles de descendants sont disjoints (Lemme 3), nous devons avoir

$$2^\ell \geq \sum_{x \in \mathcal{X}} 2^{\ell - \ell(c(x))} \Rightarrow \sum_{x \in \mathcal{X}} 2^{-\ell(c(x))} \leq 1.$$

△

Théorème 2 *Inégalité de Kraft-McMillan et codes de préfixe*
Soit $\{n(x) > 0, x \in \mathcal{X}\}$, un ensemble de nombres entiers positifs qui vérifient l'inégalité de Kraft-McMillan, eq. (3.2):

$$\sum_{x \in \mathcal{X}} 2^{-n(x)} \leq 1.$$

Alors, il existe un *code de préfixe* C avec longueurs $\ell(x) = n(x)$. (Qui est donc, nécessairement, uniquement décodable.) △

Démonstration
Admettons que les entiers $n(x)$ ont été ordonnés par ordre croissante:

$$n(x_1) \leq n(x_2) \leq \cdots \leq n(x_m).$$

Nous allons choisir itérativement les mots de code $c(x), x \in \mathcal{X}$, par ordre lexicographique. *Par construction*, le code obtenu sera un code de préfixe. Nous commençons par fixer le plus petit mot de code comme la séquence de $n(x_1)$ zéros :

$$c(x_1) = 0 \cdots 0 \ \in \{0,1\}^{n(x_1)},$$

et $c(x_2)$ le plus petit successeur de $c(x_1)$ de taille $n(x_2)$:

$$c(x_2) = 00...10..0 \ \in \{0,1\}^{n(x_2)} \qquad \text{(le } n(x_1)\text{-ème bit est égale à 1)}.$$

Les mots de code restants $c(x_i)$, $i = 3, \ldots, |\mathcal{X}|$, sont choisis de façon itérative, de la manière suivante.
Le mot de code $c(x_i)$ est la plus petite séquence de taille $\ell = n(x_i)$ qui n'a pas comme préfixe les mots de code déjà attribués $\{c(x_1), \ldots c(x_{i-1})\}$. Le nombre de mots de code de longeur ℓ qui ne sont pas des descendants des mots de code déjà fixés est

$$2^\ell - \sum_{j<i} 2^{\ell - n(x_j)} = 2^\ell - 2^\ell \sum_{j<i} 2^{n(x_j)} > 2^\ell - 2^\ell \sum_{j=1}^{n} 2^{-n(x_j)} > 2^\ell - 2^\ell = 0,$$

car $n(x_i), i = 1, \ldots, m$ satisfont l'inégalité de Kraft, ce qui garant l'existance d'au moins un mot de code avec la longueur $\ell = n(x_i)$ souhaitée. Le code $c(x_i)$ est choisi comme la séquence la plus petite (dans l'ordre lexicographique) qui est libre. △

Remarque 2 Le code construit dans cette démonstration est ordonnée par ordre lexicographique.

Le théorème précédant permet de limiter notre attention aux codes de préfixe, car pour tout code C uniquement décodable (qui, par le Théorème 1, vérifie l'inégalité de Kraft), nous pouvons trouver un code de préfixe avec les mêmes longeurs de mots (par le Théorème 2).

Définition 6 *Code complet*
Un code C est complet, si la longueur de ses mots de code satisfait l'inégalité de Kraft-McMillan avec égalité. △

Ceci veut dire que tous les noeuds de l'arbre binaire correspondante soit ils ont *les deux descendants* soit ils sont des *feuilles* de l'arbre.

Exemple 2 Le code de la Figure **??** est un code complet. Cependant, le code suivant ne l'est pas :

$$C(\mathcal{X}) = \{01, 100, 101, 110\}.$$

△

Lemme 4 Soit X une source, $x \in \mathcal{X}$, $X \sim p$. Alors, il existe un code de préfixe avec longueurs

$$\ell(x) = \lceil \log \frac{1}{p(x)} \rceil \qquad \forall x \in \mathcal{X}. \tag{3.3}$$

△

Démonstration
Il suffit de vérifier que l'inégalité de Kraft est vérifiée:

$$\sum_{x\in\mathcal{X}} 2^{-\lceil \log 1/p(x)\rceil} \leq \sum_{x\in\mathcal{X}} 2^{-(\log 1/p(x))} = \sum_{x\in\mathcal{X}} 2^{\log p(x)} = 1.$$

L'inégalité découle du fait que

$$\lceil x \rceil \geq x.$$

△

3.3 Le Théorème codage source (sans pertes)

Définition 7 *Longueur moyenne, L_C*
Soit C un code de symbole pour la source $X \in \mathcal{X}$. La longueur moyenne de C est

$$L_C(X) = \sum_{x\in\mathcal{X}} p(x)\ell(c(x)). \tag{3.4}$$

△

Théorème 3 **Codage source** *(codes de symbole)*
Étant donnée une source $X \in \mathcal{X}$, $X \sim p$, avec entropie $H(X)$, il existe un code de préfixe C (binaire, de symbole) avec longueur moyenne à moins d'un bit de entropie :

$$H(X) \leq L_C(X) < H(X) + 1. \tag{3.5}$$

△

Démonstration (borne supérieure) $L_C(X) < H(X) + 1$
Il suffit de calculer la longueur moyenne du code du Lemme 4:

$$L_C(X) = \sum_{x\in\mathcal{X}} p(x)\lceil \log \frac{1}{p(x)} \rceil < \sum_{x\in\mathcal{X}} p(x) \left(\log \frac{1}{p(x)} + 1 \right) = H(X) + 1.$$

où nous avons utilisé

$$\lceil x \rceil < x + 1.$$

Le code optimal doit avoir une longueur inférieure ou ègale à celui ci. △

Propriété 1 *Redondance*
La redondance d'un code C (de préfixe) par rapport à une source $X \sim p$, $X \in \mathcal{X}$, est bornée par l'entropie relative :

$$\mathcal{R} = L_C(X) - H(X) \geq D(p||q_C) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q_C(x)}, \tag{3.6}$$

où la loi q_C est

$$q_C(x) = \frac{2^{-\ell(c(x))}}{\sum_{x \in \mathcal{X}} 2^{-\ell(c(x))}} \qquad \forall x \in \mathcal{X}.$$

△

Démonstration
Soit

$$s = \sum_{x \in \mathcal{X}} 2^{-\ell(c(x))} \leq 1,$$

par l'inégalité de Kraft.
De la définition de $q_C(x)$:

$$2^{-\ell(c(x))} = q_C(x)s \Rightarrow \ell(c(x)) = -\log q_C(x) - \log s. \tag{3.7}$$

La redondance $\mathcal{R}$ est donc, en utilisant la définition d'entropie,

$$\begin{aligned} \mathcal{R} &= L_C(X) - H(X) = \sum_{x \in \mathcal{X}} p(x)\ell(c(x)) - \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} \\ &= -\sum_{x \in \mathcal{X}} p(x) \log q_C(x) - \log s + \sum_{x \in \mathcal{X}} p(x) \log p(x), \end{aligned}$$

ce qui est équivalent à

$$\mathcal{R} = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q_C(x)} - \log s = D(p||q_C) - \log s. \tag{3.8}$$

où nous avons reconnu l'entropie relative entre p et q_C. Comme $s \leq 1 \Rightarrow \log s \leq 0$, nous pouvons concluir que

$$\mathcal{R} \geq D(p||q_C).$$

△

La Proposition 1 permet de démontrer facilement la borne inférieure dans le Théorème 3. Comme l'entropie relative est toujours non n égative

$$\mathcal{R} \geq 0 \Leftrightarrow L_C(X) - H(X) \geq 0 \Leftrightarrow L_C(X) \geq H(X).$$

Propriété 2 Les conditions pour qu'un code C atteigne la taille de code minimale

$$L_C(X) = H(X),$$

sont donc (voir équation (3.8))

$$\log s = 0 \Leftrightarrow s = 1,$$

c'est à dire, le code doit être *complet* **et**

$$D(p||q_C) = 0 \Leftrightarrow q_C(x) = p(x),$$

c'est à dire (de l'équation (3.7)) la longueur des mots de code doit être

$$\ell(c(x)) = \log \frac{1}{p(x)}.$$

Remarque 3 *Lien entre codes et lois de probabilité*
Les équations de cette Section établissent un lien intime entre *lois de probabilité*, d'un coté, et *longueurs de codes* optimaux, de l'autre coté. Pour une loi p fixée, un ensemble de longueurs optimales peut lui être associé :

$$\ell(x) = \log \frac{1}{p(x)}, \forall x \in \mathcal{X},$$

et, si nous avons un code C dont les mots ont longueurs $\ell(x)$, nous pouvons déterminer une loi de probabilité p pour laquelle le code est optimal :

$$p(x) = 2^{-\ell(c(x))} \qquad \forall x \in \mathcal{X}.$$

Ce lien sera exploité, à travers la notion de *Code Universel*, par le *Principe de Longueur de Description Minimale* pour résoudre le problème de la sélection de modèles de compléxité différente, dans une Section postérieure du cours.

△

Théorème 4 *Codage source (codes de bloc)*
Soit X une source de symboles i.i.d, $X_i \in \mathcal{X}$, $X_i \sim p$. Alors $\forall \epsilon > 0$ il existe un code de bloc C_n de longueur moyenne (par symbole source) qui vérifie

$$L_{C_n}(X) \leq H(X) + \epsilon. \tag{3.9}$$

△

Démonstration
La démonstration est une simple application du Théorème 3 à des blocs de taille n de symboles de la source X. L'entropie d'un ensemble $X^{(n)}$ de n variables i.i.d. est

$$H(X^{(n)}) = nH(X).$$

Le Théorème 3 nous dit que

$$L_C(X^{(n)}) \leq H(X^{(n)}) + 1 = nH(X) + 1.$$

La longueur *par symbole* de la source est

$$L_{C_n}(X) = \frac{L_C(X^{(n)})}{n} \leq H(X) + \frac{1}{n}.$$

Alors si nous prennons n de façon que

$$\frac{1}{n} \leq \epsilon \Leftrightarrow n \geq \frac{1}{\epsilon},$$

le code de bloc C_n satisfait (3.9). △

3.4 Le code de Huffman

3.4.1 Algorithme

L'algorithme de Huffman conduit à un code dont la longueur est minimale, près de l'entropie de la source. Comme nous le verrons, il construit l'arbre binaire correspondante au code recherché récursivement à partir de ses feuilles. Il peut encore être interprété comme transformant récursivement l'alphabet initial dans un alphabet dont la cardinalité diminue de 1 à chaque itération.

Soit $\mathcal{X} = \mathcal{X}_0$ l'alphabet de la source, $|\mathcal{X}| = m$, et $p_0 = p$ sa loi de probabilité.

Nous énnonçons d'abord une série de Lemmes nécessaires à la démonstration de l'optimalité des codes de Huffman. À partir de maintenant, et pour tout le reste de cette Section, nous allons considérer que les symboles de la source $X \sim p$, $X \in \mathcal{X} = \{a_1, a_2, \ldots, a_m\}$ sont ordonnés par *ordre décroissante de probabilité*:

$$p(a_1) \geq p(a_2) \geq \cdots \geq p(a_m).$$

Pour simplifier la présentation, nous utiliserons souvent les notations simplifiées suivantes:

$$p_i \equiv p(a_i) \qquad c_i = c(a_i) \qquad \ell_i = \ell(c_i), i = 1, \ldots, m.$$

Lemme 5 Soit C un code optimal (de longueur moyenne minimale). Alors

$$\text{si } p_i > p_j \Rightarrow \ell(c_i) \leq \ell(c_j).$$

C'est à dire, les mots les plus probables correspondent à des mots de code plus courts. △

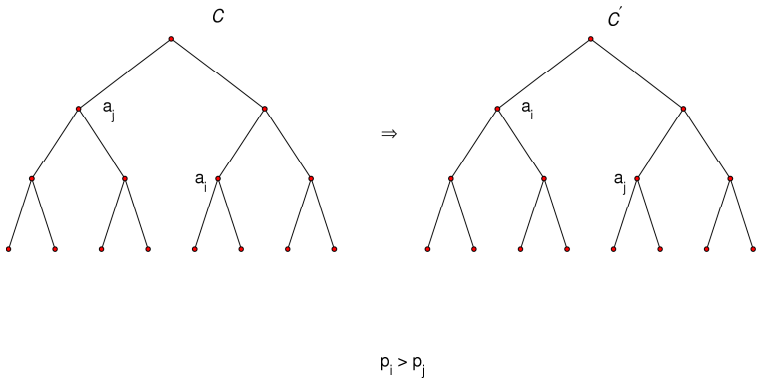

Figure 3.2: Illustration du Lemme 5.

Démonstration
(Par absurde.) Admettez qu'il existent des symboles a_i et a_j tels que

$$p_i - p_j > 0 \qquad \text{et} \qquad \ell(c_i) - \ell(c_j) > 0. \tag{3.10}$$

Nous allons voir que ce code ne peut pas être optimal.
Considérons le code C' obtenu en échangeant les codes associés aux symboles a_i et a_j, voir Figure 3.2,

$$c'(a_n) = c(a_n), \forall n \neq i, n \neq j, \qquad c'(a_i) = c(a_j), c'(a_j) = c(a_i).$$

La longueur des mots du code C' est

$$\ell(c'_n) = \ell(c_n), n \neq i, n \neq j, \qquad \ell(c'_i) = \ell(c_j), \ell(c'_j) = \ell(c_i), \tag{3.11}$$

et la longueur moyenne est donnée par

$$L_{C'}(X) = \sum_{n=1}^{m} p_n \ell(c'_n).$$

La différence entre les longueurs moyennes des deux codes est

$$L_C(X) - L_{C'}(X) = \sum_{n=1}^{m} p_n \left[\ell(c_n) - \ell(c'_n)\right].$$

De (3.11),

$$L_C(X) - L_{C'}(X) \quad = \quad \sum_{n=i,j} p_n \left[\ell(c_n) - \ell(c'_n)\right]$$

$$\begin{aligned} &= p_i\left(\ell(c_i) - \ell(c_j)\right) + p_j\left(\ell(c_j) - \ell(c_i)\right) \\ &= \left(\ell(c_i) - \ell(c_j)\right)\left(p_i - p_j\right). \end{aligned}$$

Comme les deux termes de ce produit sont positifs, nous pouvons déduire

$$L_C(X) - L_{C'}(X) > 0,$$

ce qui contrarie l'optimalité de C. △

Lemme 6 Il existe un code C de longueur moyenne minimale tel que

$$\ell(c_1) \leq \ell(c_2) \leq \cdots \leq \ell(c_m).$$

Démonstration
Nous avons vu que si C est optimal

$$p_i < p_j \Rightarrow \ell(c_i) \geq \ell(c_j).$$

La seule possibilité qui reste est

$$p_i = p_j \qquad \text{et} \qquad \ell(c_i) < \ell(c_j).$$

Si nous échangeons les codes de ces deux symboles, la longueur moyenne du code n'est pas modifiée, et nous obtenons un code qui satisfait le Lemme. △

Lemme 7 Il existe un code optimal C où le mot le plus long est $c(a_m)$. △

Ce Lemme est une conséquence des deux derniers Lemmes.

Lemme 8 L'arbre correspondant à un code optimal C ne possède pas des noeuds internes incomplets. △

Démonstration
(Par absurde.) Soit $c \in \{0,1\}^n$ la séquence binaire correspondante à un *noeud interne* de l'arbre. Le Lemme affirme que si C est un code optimal, alors les deux noeuds descendants de c sont racines de sous-arbres qui contiennent des mots du code C. Nous allons voir que si ceci n'est pas vrai, nous pouvons construir à partir de C un autre code de taille plus petite.

Soit c un noeud interne de l'arbre qui ne possède qu'une seule branche descendante (comme pour l'arbre à gauche dans la Figure 3.3), et $\mathcal{C}_n^d = \{c_i\}$ l'ensemble des mots de code qui ont c comme préfixe (qui sont des *descendants* de c, c'est à dire, qui sont

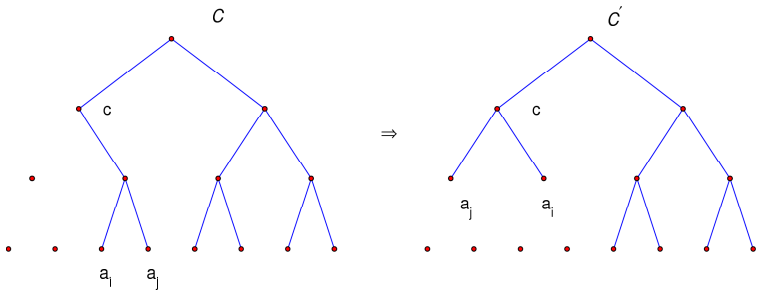

Figure 3.3: Illustration du Lemme 8.

sur la sous-arbre qui a c comme racine). Par hypothèse, il existe au moins un mot de code dans $\mathcal{C}_n^d$. Les mots de code dans cet ensemble doivent tous être de la forme

$$c_i = c\, b\, c_i', \qquad b \in \{0,1\}, c_i' \in \{0,1\}^+,$$

où la valeur de b dépend de la branche descendente du noeud c qui est retenue dans l'arbre du code C (voir Figure 3.3). Les longueurs de ces mots peut être écrite en fonction de la longueur de c :

$$\ell(c_i) = \ell(c) + 1 + \ell(c_i').$$

Le code C' qui maintient tous les mots de code de C, sauf pour les éléments $c_i \in \mathcal{C}_n^d$, auxquels il fait correspondre les séquences

$$c_i' = c\, c_i', c_i' \in \{0,1\}^+, \qquad c_i \in \mathcal{C}_n^d,$$

est encore un code de préfixe, et a une longueur moyenne strictement inférieure à celle de C. △

Lemme 9 Il existe un code (binaire) optimal où les mots c_m et c_{m-1} ne difèrent que dans le dernier bit (ils forment un pair de "jumeaux"). △

Démonstration
Si c_m n'a pas de "jumeau", alors, par le Lemme 8 le code n'est pas optimal (son prédécesseur serait alors in noeud interne incomplet). Si C est optimal il doit donc exister un autre mot de code c_i qui difère de c_m uniquement dans le dernier bit. Du Lemme 6, $\ell(i) = \ell(m) \Leftrightarrow p_i \leq p_m$, et, par hypothèse, $i \neq m-1 \Rightarrow p_i \geq p_{m-1}$, ce

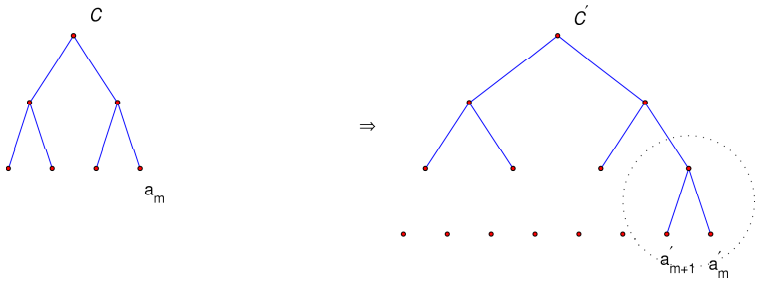

Figure 3.4: Illustration du Lemme 10.

qui implique $\ell(c_{m-1}) = \ell(c_i) = \ell(c_m)$. Si nous échangeons les mots de code de a_i et a_{m-1} nous obtenons donc un code de la même longueur qui satisfait le Lemme. △

Lemme 10 Soit C un code optimal pour une source $X \in \mathcal{X}$, $|\mathcal{X}| = m$, $X \sim p = [p_1 \cdots p_m]$ (nous admettons toujours que les symboles sont ordonnés par ordre croissante de probabilité : $i > j \Rightarrow p_i \leq p_j$). Le code C' obtenu à partir de C de la façon suivante

$$c'(a_i) = c(a_i), i = 1, \ldots, m-1; \qquad c'(a'_m) = c\,0, \qquad c'(a'_{m+1}) = c\,1,$$

est un code optimal pour une source

$$X' \in \mathcal{X}' = \{a_1, \ldots, a_{m-1}, a'_m, a'_{m+1}\}; \qquad X' \sim p' = [p_1 \ldots, p_{m-1}, p'_m, p'_{m+1}],$$

où

$$p'_m + p'_{m+1} = p_m.$$

△

La Figure 3.4 illustre ce Lemme.
Démonstration
(Par absurde.) La diférence des longueurs moyennes des codes C et C' est

$$L_{C'}(X') - L_C(X) = p'_{m+1} + p'_m.$$

Admettons qu'il existe un autre code optimal $Q' \neq C'$ pour la source X' :

$$L_{Q'} < L_{C'}. \tag{3.12}$$

Nous pouvons admettre que q'_m et q'_{m+1} ne difèrent que dans le dernier bit (par le Lemme 9). Le code Q pour la source $X \in \mathcal{X}$, obtenu en associant à a_m la racine commune de q'_m et q'_{m+1}, a une longueur moyenne

$$L_Q(X) = L_{Q'}(X') - \left(p'_m + p'_{m+1}\right).$$

De (3.12), nous pouvons conclure que

$$L_Q(X) < L_{C'}(X') - \left(p'_m + p'_{m+1}\right) = L_C(X).$$

Nous constatons donc que si Q est optimal, alors le code de départ, C, ne peut pas être optimal, ce qui contredit l'hypothèse du Théorème. △

Le Lemme 10 est la clef de l'algorithme de Huffman pour la construction de codes de symbole optimaux, que nous présentons par la suite.

Nous commençons par définir une opération de *réduction* sur une source probabiliste. Soit $X \in \mathcal{X}$, $|\mathcal{X}| = m$, $X \sim p$, et soient $x = a_m,\ y = a_{m-1} \in \mathcal{X}$ les deux éléments *les moins probables* de $\mathcal{X}_i$. Nous définissons $X_{i+1} \in \mathcal{X}_{i+1}$, $X \sim p^{i+1}$ de la façon suivante :

$$\begin{aligned} \mathcal{X}_{i+1} &= \mathcal{X}_i \setminus \{x, y\} \cup \{x'\}, \qquad x' \leftrightarrow (x, y), \qquad \mathcal{X}_o = \mathcal{X}, i = 1, \ldots, m-1, \\ p^{i+1} &= \left[p_1^i, \ldots, p_{m-1}^i, (p_{m-1}^i + p_m^i)\right], \qquad p^0 = p, i = 1, \ldots, m-1 \end{aligned} \tag{3.13}$$

où x' est un nouveau symbole qui représente le pair (x, y). Notez que l'on a nécessairement

$$|\mathcal{X}_{i+1}| = |\mathcal{X}_i| - 1 = m - i.$$

Considérons maintenant l'application itérée de la *réduction* (3.13):

$$\left\{\mathcal{X}_0, p^0\right\} \xrightarrow{\mathcal{R}_1} \left\{\mathcal{X}_1, p^1\right\} \xrightarrow{\mathcal{R}_2} \left\{\mathcal{X}_2, p^2\right\} \xrightarrow{\mathcal{R}_3} \cdots \xrightarrow{\mathcal{R}_{m-1}} \left\{\mathcal{X}_{m-1}, p^{m-1}\right\},$$

où, par construction

$$|\mathcal{X}_{m-1}| = 1 \qquad p^{m-1} = 1.$$

À chaque opération de *réduction* $\mathcal{R}_i$, définie par l'équation (3.13), correspond l'ajout d'un nouveau noeud x' dans un arbre binaire. Initialement, cet arbre ne contient que deux feuilles terminales correspondantes aux deux symboles les moins probables de la source X. Le noeud qui est ajouté est relié par ses deux branches d éscendantes aux deux symboles les moins probables $\mathcal{X}_{i-1}$. Le code est finalement obtenu en parcourant l'arbre depuis sa racine.

Exemple 3 Considérons la construction d'un code de Huffman pour la source suivante

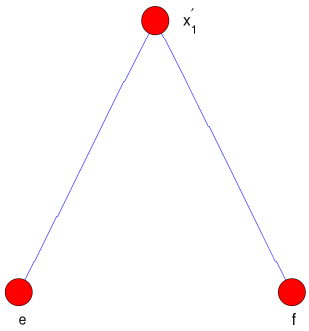

Figure 3.5: $\mathcal{R}_1$.

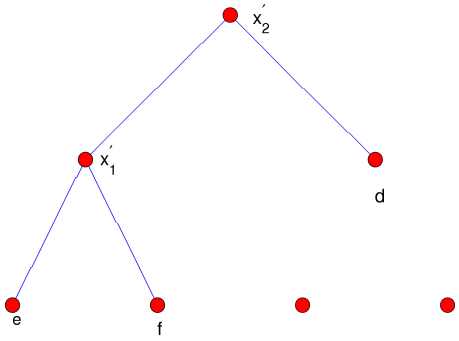

Figure 3.6: $\mathcal{R}_2$.

x	$p(x)$
a	0.3
b	0.25
c	0.2
d	0.1
e	0.08
f	0.07

La première opération de réduction nous même à

$$\mathcal{X}_1 = \{a, b, c, d, x_1'\}, \qquad x_1' \leftrightarrow (e, f), \qquad p^1 = [0.3, 0.25, 0.2, 0.15, 0.1]\,,$$

et le premier arbre illustré dans la Figure 3.5. La deuxième réduction agi sur $\mathcal{X}_1$ pour engendrer

$$\mathcal{X}_2 = \{a, b, c, x_2'\}, \qquad x_2' \leftrightarrow (x_1', d), \qquad p^2 = [0.3, 0.25, 0.2, 0.25]\,,$$

et donne origine à l'arbre de la Figure 3.6.

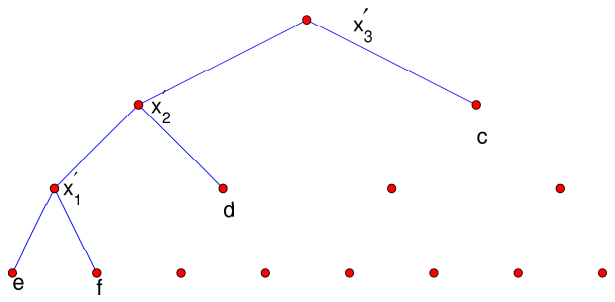

Figure 3.7: $\mathcal{R}_3$.

À la troisième réduction nous obtenons

$$\mathcal{X}_3 = \{a, b, x_3'\}, \qquad x_3' \leftrightarrow (x_2', c), \qquad p^3 = [0.3, 0.25, 0.45]\,,$$

et l'arbre de la Figure 3.7.

La réduction $\mathcal{R}_4$ conduit à

$$\mathcal{X}_4 = \{x_3', x_4'\}, \qquad x_5' \leftrightarrow (x_4', x_3'), \qquad p^4 = [0.45, 0.55]\,,$$

et à l'arbre représenté dans la Figure 3.8.

La dernière réduction nous conduit finalement à un seul symbole, avec toute la probabilité (1), et à l'arbre complet qui spécifie le code optimal, représenté dans la Figure 3.9.

Le code obtenu est

x	$p(x)$	$c(x)$	$\ell(c(x))$
a	0.3	10	2
b	0.25	11	2
c	0.2	01	2
d	0.1	001	3
e	0.08	0001	4
f	0.07	0000	4

Nous pouvons constater que ce code vérifie tous les Lemmes énnoncés dans cette Section. Sa longueur moyenne est

$$L_C(X) = \sum_{x\in\{a,\cdots,f\}} p(x)\ell(c(x)) = 2(0.3{+}0.25{+}0.2){+}3{\cdot}0.1{+}4(0.08{+}0.07) = 2.38 \text{ bits.}$$

L'entropie de la source est

$$H(X) = \sum_{x\in\{a,\cdots,f\}} p(x)\log\frac{1}{p(x)} = 2.377 \text{ bits.}$$

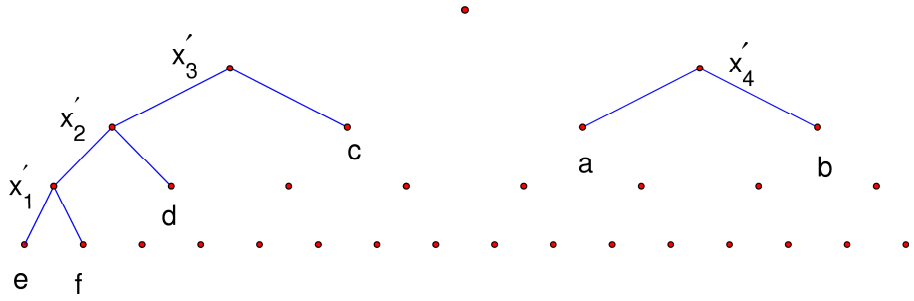

Figure 3.8: $\mathcal{R}_4$.

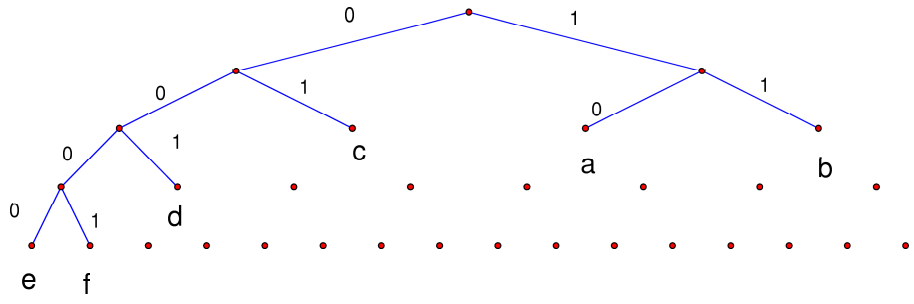

Figure 3.9: $\mathcal{R}_5$.

Nous pouvons donc constater que le code vérifie le Théorème 3:

$$H(X) < L_C(X) \leq H(X) + 1.$$

△

3.4.2 Inconvénients des codes de Huffman

Ajusté à une source spécifique
Le code de Huffman est un bon exemple d'une solution *adaptée*: le codeur obtenu est en fait optimal *pour une source donnée*, avec une certaine *loi de probabilité* p qui est supposée connue et fixe. Nous avons vu que si la source suit une loi de probabilité différente, $q \neq p$ le code exibira une rédondance qui est égale à l'entropie relative $D(q||p)$.
L'algorithme de Huffman a une complexité de $O(n \log n)$, associée surtout au besoin d'ordonner les symboles par ordre décroissante de probabilit à chaque itération. Cependant, pour un codeur générique, qui doit accepter des sources avec distributions arbitraires, la charge computationnelle associée au calcul du codeur optimale n'est pas importante. Ces codeurs (codeur universels) doivent s'adapter dynamiquement aux caractéristiques de la source, et demandent très rarement la détermination du code complet.

Pénalité d'un bit
Le code de Huffman satisfait le Théorème 3, et donc a une pénali'e qui peut aller jusqu'à 1 bit, par rapport à l'entropie de la source, ce que peut être non n égligeable pour certaines sources. La solution de coder des blocs de symboles source, comme nous l'avons vu dans le Théorème 4, permet de "distribuer" ce bit de pénalité par n symboles, mais uniquement au prix de perdre la possibilité de décoder d'une façon instantanée les symboles de la source (il faut attendre la fin d'un bloc).

Code de symbole
En associant d'une façon isolée (et fixe) un code à chaque symbole source, ce type de codes perd la possibilité d'ajuster le codage de chaque symbole à son *contexte*. Par exemple la séquence *ordinate* est suivie, avec grande probabilité, par les lettres *ur*. Pour aller au delà des codes de symbole, il faudra exploiter la dépendance statistique entre les symboles succéssivement émis par la source.

Le prochain Chapitre concerne la conception d'un autre type de codes, qui ne présentent pas ces inconvénients.