Codage entropique à base de règles de ré-écriture

Hervé JÉGOU[1], Christine GUILLEMOT[2]

[1]IRISA/Université de Rennes 1

[2]IRISA/INRIA
Campus Universitaire de Beaulieu, 35042 RENNES CEDEX
herve.jegou@irisa.fr, christine.guillemot@irisa.fr

Résumé – Des codes de compression sans perte basés sur l'utilisation d'un ensemble de règles de ré-écriture sont introduits et leur performance théorique en compression est analysée. Le formalisme des règles de production permet d'obtenir un degré de liberté important lors de la création de codes. En particulier, il est possible de trouver des codes qui compressent sous le bit ou d'obtenir des codes dont la probabilité marginale bit est uniforme.

Abstract – This article introduces a new set of lossless source codes based on re-writing rules. Theoretical compression efficiency is analytically derived. These production rules allows to obtain an higher degree of freedom in the design of variable length codes. Hence, it is possible to design some codes generating less than 1 bit per encoded symbol or codes such that the marginal bit distribution is uniform.

1 Introduction

Cet article présente des codes de compression définis à partir de règles de production de la forme $a\overline{l} \rightarrow \overline{b}$, où $a \in \mathcal{A}$ représente un symbole de l'alphabet $\mathcal{A}$ et où $\overline{l}, \overline{b}$ sont des petites suites de bits. Contrairement aux algorithmes basés sur un dictionnaire (par ex. [1]), ou aux codes proposés dans [2], les règles de production sont ici pré-définies en accord avec les probabilités de source, ce qui rapproche ces codes des codes à longueur variable (CLV), tels que les codes de Huffman [3]. Ces derniers sont naturellement englobés par le formalisme proposé. Le degré de liberté résultant de l'extension des CLV permet d'obtenir des codes ayant une longueur de description plus courte que celle des codes de Huffman, pour la même complexité d'encodage et de décodage. En effet, l'encodage est effectué en utilisant des automates d'encodage et de décodage. Dans cet article, nous présentons ces codes dans la section 2 ainsi que la méthode de construction des automates d'encodage et de décodage. L'analyse de leur efficacité en compression est fournie dans la section 3. Nous décrivons ensuite, dans la section 4, une méthode de construction qui permet d'obtenir des codes tels que la probabilité marginale des bits émis est de $\frac{1}{2}$, et ceci même si la loi de probabilité n'est pas connue à l'encodage. Ce résultat est prouvé analytiquement. Ceci est intéressant dans la mesure où cette hypothèse est largement utilisée lors du codage de canal et permet d'espérer de meilleures performances en décodage conjoint source/canal.

2 Systèmes de ré-écriture entropiques

Dans la suite les variables aléatoires sont notées en majuscules et les réalisations correspondantes en minuscules. Les ensembles sont écrits en caractères calligraphiques. Le cardinal d'un ensemble $\mathcal{X}$ est noté $|\mathcal{X}|$. Nous définissons $\mathcal{X}^+ = \bigcup_{i=1}^{\infty} \mathcal{X}^i$ et $\mathcal{X}^* = \{\varepsilon\} \cup \mathcal{X}^+$, où ε représente la séquence vide, *i.e.* l'élément neutre pour la concaténation. Ainsi $\mathcal{X}^*$ représente l'ensemble des séquences formées d'éléments de $\mathcal{X}$. Soit $\mathbf{S} \in \mathcal{A}^+$ une séquence de symboles à valeur dans l'alphabet $\mathcal{A} = \{a_1, ...a_i, ...\}$. La longueur d'une telle séquence sera notée $L(\mathbf{S})$. Soit $\mathcal{B} = \{0, 1\}$ l'alphabet binaire. Dans la suite, le train binaire émis est noté $\mathbf{E} = E_1...E_{L(\mathbf{E})} \in \mathcal{B}^*$ et la réalisation correspondante $\mathbf{e}$.

Définition 1: un Système de Ré-écriture Entropique (SRE) est un ensemble $\mathcal{R}$ de règles de production de la forme $r_{i,j}$: $a_i \overline{l}_{i,j} \rightarrow \overline{b}_{i,j}$ où $\overline{l}_{i,j} \in \mathcal{B}^*, \overline{b}_{i,j} \in \mathcal{B}^+$. La règle $r_{i,j}$ représente ici la $j^{ème}$ règle de production associée au symbole a_i et $\mathcal{R}_i$ représente l'ensemble des règles associées à un symbole donné a_i. Le SRE doit en outre vérifier les conditions suivantes :

1. $\forall i,\ |\mathcal{R}_i| \geq 1$,
2. l'ensemble des mots de code $\bigcup_{i=1}^{|\mathcal{A}|} \bigcup_{j=1}^{|\mathcal{R}_i|} \{\overline{b}_{i,j}\}$ est tel qu'aucun mot de code n'est le préfixe d'un autre mot de code, *i.e.* cet ensemble correspond à un code préfixe [4].
3. $\forall i,\ \bigcup_{i=1}^{|\mathcal{R}_i|} \{\overline{l}_{i,j}\}$ est l'ensemble $\{\varepsilon\}$ ou est un code préfixe complet. (*i.e.* tel que l'égalité de Kraft est vérifiée).
4. $\forall i\, \forall i' \neq i, \forall j, j',\ \overline{b}_{i,j} = \overline{l}_{i',j'}$ ou $\overline{b}_{i,j}$ n'est *pas* préfixe de $\overline{l}_{i',j'}$.

La définition 1 ne garantit pas qu'un tel système définisse un système de compression valide. Par exemple, une règle de production $r_{i,j}$ telle que $\overline{b}_{i,j}$ est préfixe de $\overline{l}_{i,j}$ n'est pas valide. Nous supposons dans la suite que le système considéré est valide. Ces règles de production permettent de transformer une suite de symboles $\mathbf{s}$ en une suite de bits $\mathbf{e}$ par application successive de règles de production. Ces règles sont supposées réversibles, *i.e.* inverser les directions des flèches du système de ré-écriture permet de retrouver la séquence $\mathbf{s}$ à partir de $\mathbf{e}$. Un CLV peut être vu comme un SRE. C'est le cas de $\mathcal{C}_1$ dans l'exemple ci-dessous.

Exemple :

$$\mathcal{C}_1 = \begin{cases} r_{1,1} : & a_1 & \rightarrow & 0 \\ r_{2,1} : & a_2 & \rightarrow & 10 \\ r_{3,1} : & a_3 & \rightarrow & 11 \end{cases}$$

$$\mathcal{C}_2 = \begin{cases} r_{1,1} : & a_1 0 & \rightarrow & 10 \\ r_{1,2} : & a_1 1 & \rightarrow & 01 \\ r_{2,1} : & a_2 & \rightarrow & 00 \\ r_{3,1} : & a_3 & \rightarrow & 11 \end{cases} \quad (1)$$

$$\mathcal{C}_3 = \begin{cases} r_{1,1} : & a_1 1 & \rightarrow & 0 \\ r_{1,2} : & a_1 0 & \rightarrow & 10 \\ r_{2,1} : & a_2 & \rightarrow & 110 \\ r_{3,1} : & a_3 & \rightarrow & 111 \end{cases} \quad (2)$$

Le code $\mathcal{C}_2$ est un code à contrainte de suffixe, c'est-à-dire un code tel que $\forall i, j$ $\overline{l}_{i,j}$ est suffixe de $\overline{b}_{i,j}$. Ces codes ont été considérés dans [5] dans un contexte de résistance aux erreurs. Les SRE sont un ensemble de codes qui inclut et généralise celui proposé dans [5]. Ainsi, le code $\mathcal{C}_3$ n'est pas un code à contrainte de suffixe. Les SRE sont également représentés sous forme d'arbres, comme présentés sur la figure 1. La structure d'arbre correspond à celle du code préfixe défini par

$$\bigcup_{i=1}^{|\mathcal{A}|} \bigcup_{j=1}^{|\mathcal{R}_i|} \{\overline{b}_{i,j}\}. \quad (3)$$

Les feuilles sont en relation à la fois avec le symbole a_i et avec la suite de bits $\overline{l}_{i,j}$.

Rappelons qu'à l'encodage, les règles de production transforment $\mathbf{s}$ pour obtenir la séquence binaire $\mathbf{e}$. Un segment quelconque de la séquence courante (composée de symbole(s) *et* de bit(s)) peut être ré-écrit s'il existe une règle de production qui a ce segment en entrée. De par la définition des règles proposées, cette entrée est composée d'un symbole et d'un nombre variable de bits. Un SRE n'est valide que s'il n'y a pas d'ambiguïté sur la règle à appliquer. Par conséquent, les règles de productions s'arrêtent lorsqu'il n'y a plus de symbole et que la séquence courante n'est constituée que de bits.

Dans le cas général, comme c'est le cas pour les codes $\mathcal{C}_2$ et $\mathcal{C}_3$, les règles de production qui définissent le SRE ne permettent pas d'encoder la suite $\mathbf{S}$ en passe avant. Ainsi l'encodage est effectué en passe arrière. Pour initier le processus d'encodage, des règles spécifiques doivent être utilisées pour la fin de la séquence, car le dernier symbole peut ne pas être suffisant pour déclencher l'application d'une règle de production. Dans la plupart des cas, les règles à utiliser peuvent être arbitrairement définies en supposant que les bits manquant pour déclencher une règle sont égaux à zéro. Ceci est un choix valide si ces zéros ne permettent pas de déclencher une règle par eux-mêmes. En effet au décodeur un symbole de trop serait alors décodé. Ainsi, le choix du bit de terminaison 0 est valide pour les codes $\mathcal{C}_1$ et $\mathcal{C}_2$ mais ne peut pas être utilisé pour le code $\mathcal{C}_3$, parce que 0 suffit à déclencher la règle $r_{1,1}$. Si un code à contrainte de suffixe est utilisé, les bits de terminaison ne sont, par définition, pas modifiés par une règle subséquente. Il n'est donc pas requis de les transmettre, puisqu'ils sont connus au décodeur par convention. Dans le cas général, ces bits de terminaison doivent être transmis.

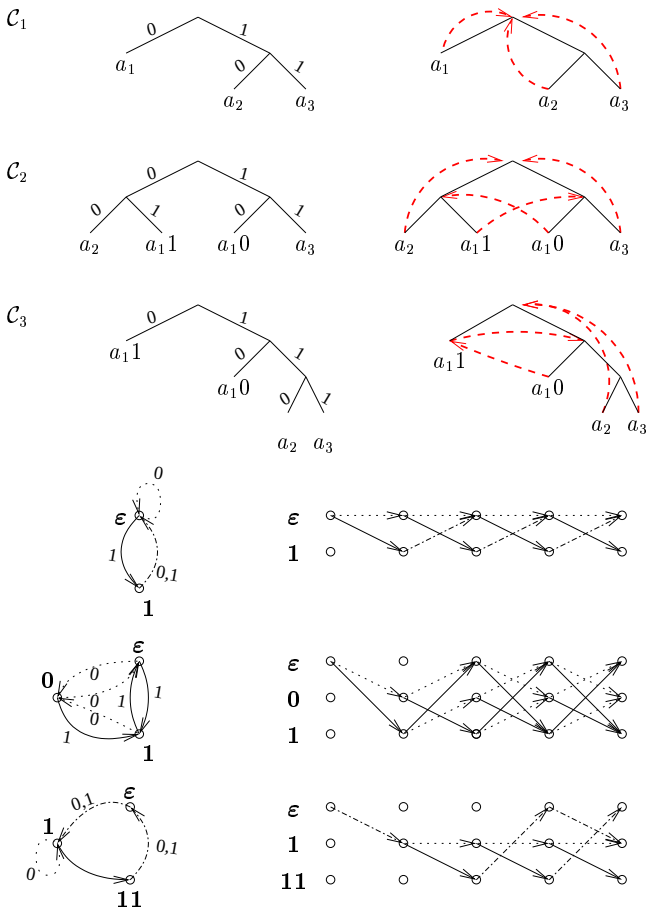

FIG. 1: *Exemples de SRE. De haut en bas puis de gauche à droite : représentation en arbre des codes, transitions générées par les règles, automates de décodage associés, treillis de décodage. Les transitions qui correspondent au bit 0 et au bit 1 sont représentées respectivement en pointillé et en train continu.*

Exemple : la séquence $\mathbf{s}_1 = a_1 a_1 a_1 a_1 a_1$ *est encodée avec le code* $\mathcal{C}_3$ *de la manière suivante (le bit de terminaison est* 1*) :*

$$\begin{aligned} r_{1,1} : \quad \mathbf{s}_1 1 &= a_1 a_1 a_1 a_1 \underline{a_1 1} \\ r_{1,2} : \quad & a_1 a_1 a_1 \underline{a_1 0} \\ r_{1,1} : \quad & a_1 a_1 \underline{a_1 10} \\ r_{1,2} : \quad & a_1 \underline{a_1 00} \\ r_{1,1} : \quad & \underline{a_1 100} \\ & \mathbf{e}_1 = 000 \end{aligned}$$

Remarquez que l'encodage de la séquence $\mathbf{s}_1$ produit un train binaire de longueur 3, ce qui revient à transmettre le symbole a_1 avec 0.6 bit seulement. Cet exemple illustre donc la capacité de ces codes à encoder des séquences avec moins d'un bit. Ceci est impossible dans le cas des codes de Huffman. Pour ces derniers, seule une vectorisation de la source permet d'encoder avec moins d'un bit. On parle alors de codes de Huffman généralisés. L'utilisation de codes généralisés (Huffman ou Tunstall [6]) requiert des tables dont la dimension croît géométriquement avec la longueur de la séquence, ce qui rend leur utilisation impossible.

Au décodage, les règles inverses sont utilisées. L'encodage et le décodage peuvent être facilement implantés sous forme d'automates. Ces automates sont utilisés afin de conserver la mémoire des processus d'encodage et de décodage. Aux états

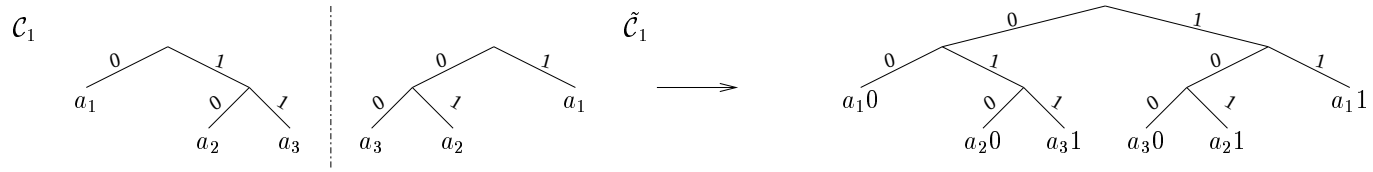

FIG. 2: *Le code primitif $\mathcal{C}_1$, son inverse $\tilde{\mathcal{C}}_1$ et le SRE résultant.*

de ces automates correspondent ainsi les segments de bits utiles pour appliquer les règles de production suivantes. Les automates sont générés à partir de l'ensemble des règles de réécriture. Les transitions sur l'automate d'encodage représentent le processus d'encodage et sont déclenchées par des symboles de $\mathbf{s}$. Les états internes de cet automate sont définis par les segments de longueur variable $\{\overline{l}_{i,j}\}$. Ainsi, l'ensemble des états de l'automate d'encodage est respectivement donné, pour les codes $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$ par $\{\varepsilon\}$, $\{0,1\}$ et $\{0,1\}$. Les états de l'automate de décodage correspondent aux segments de bit qui ont déjà été décodés, mais qui ne suffisent pas à identifier un symbole. Pour les CLV tels que les codes de Huffman, ces noeuds internes correspondent aux noeuds internes de l'arbre du code. Ainsi, pour les codes $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$, l'ensemble des noeuds internes est donné respectivement par $\{\varepsilon, 1\}$, $\{\varepsilon, 0, 1\}$ et $\{\varepsilon, 1, 11\}$.

La représentation graphique des automates de décodage ainsi que les treillis correspondants peuvent être facilement déduits de la représentation en arbre des codes, comme indiqué sur la figure 1. Bien qu'ils n'apparaissent pas sur la figure pour des raisons de clarté, notez que l'ensemble des symboles générés est également associé à chaque bit de transition. Pour les codes $\mathcal{C}_1$ et $\mathcal{C}_2$, il y a au maximum 1 symbole associé à chaque bit de transition. Ce n'est pas le cas pour le code $\mathcal{C}_3$, où la transition partant du l'état $\mathbf{1}$ de l'automate de décodage génère deux symboles a_1, permettant ainsi d'obtenir un coût d'encodage inférieur au bit pour ce SRE.

3 Efficacité en compression

La source $\mathbf{S}$ est supposée être sans mémoire et caractérisée par sa densité de probabilité sur $\mathcal{A}$: $\boldsymbol{\mu} = \{\mathbb{P}(a_1), \dots \mathbb{P}(a_i), \dots\}$. Soit

$$\delta(r_{i,j}) = L(\overline{b}_{i,j}) - L(\overline{l}_{i,j}) \tag{4}$$

le nombre de bits générés par une règle de production donnée $r_{i,j}$. Pour le cas particulier où $\forall i,\ \forall j, j'\ \delta(r_{i,j}) = \delta(r_{i,j'})$, l'espérance E_l de la longueur de description est donnée par

$$E_l = \sum_{a_i \in \mathcal{A}} \mathbb{P}(a_i)\delta_{i,1}. \tag{5}$$

Ainsi, si $\boldsymbol{\mu}_1 = \{0.7, 0.2, 0.1\}$, nous avons $E_l = 1.3$ pour $\mathcal{C}_1$ comme pour $\mathcal{C}_2$. Remarquez que l'entropie de la source est de 1.157.

Soit $R_t : S_t\overline{L}_t \to \overline{B}_t$ la variable aléatoire qui correspond à la règle utilisée pour encoder le symbole S_t. Le processus obtenu à partir de R_t en inversant l'horloge symbole forme une chaîne de Markov invariante $(Z_{t'})$. Les probabilités de cette chaîne de Markov sont obtenues à partir de $\boldsymbol{\mu}$ de la manière suivante :

$$\begin{aligned}\mathbb{P}(Z_{t'}|Z_{t'-1}) &= \mathbb{P}(R_t = r_{i,j}|R_{t+1} = r_{i',j'}) \\ &= \begin{cases} \mathbb{P}(a_i) & \text{si } \overline{l}_{i,j} \text{ est préfixe de } \overline{b}_{i',j'}, \\ 0 & \text{sinon.} \end{cases}\end{aligned} \tag{6}$$

Sous l'hypothèse que $(Z_{t'})$ est irréductible et apériodique, hypothèse dont il est possible de vérifier la validité par une analyse spectrale du noyau de transition de ce processus, la probabilité marginale du processus est obtenue comme l'unique vecteur de probabilité invariant pour le noyau de transition de l'équation 6. C'est-à-dire c'est le seul vecteur propre positif normalisé associé à la valeur propre 1. On en déduit la valeur de $\mathbb{E}(\delta(R_{t'}))$ de l'espérance du nombre de bits produits. Avec le théorème de Cesaro, cette valeur correspond également à la valeur asymptotique de E_l lorsque la taille de la séquence $\mathbf{S}$ tend vers l'infini.

Les performances obtenues en pratique sont très proches de cette performance asymptotique. Il est également possible de calculer de manière exacte de l'espérance du nombre de bits émis pour les séquences finies.

Exemple : pour le code $\mathcal{C}_3$, le calcul de la loi stationnaire mène à $\mathbb{P}(R_t = r_{i,j}) = \{0.412, 0.288, 0.2, 0.1\}$ et donc $E_l = 0.412 \times 0 + 0.288 \times 1 + 0.2 \times 3 + 0.1 \times 3 = 1.188$, contre $E_l = 1.3$ pour le code de Huffman.

4 Construction de codes avec une probabilité bit de $1/2$

Pour construire le code, on part d'un CLV initial

$$\mathcal{H} = \{\overline{b}_{1,1}, \dots \overline{b}_{|\mathcal{A}|,1}\}, \tag{7}$$

par exemple un code de Huffman. On considère alors le CLV $\tilde{\mathcal{H}}$ défini tel que chaque bit de transition de l'arbre de $\tilde{\mathcal{H}}$ est l'inverse du bit de transition correspondant de $\mathcal{H}$, comme décrit sur la figure 2. Le SRE recherché est obtenu en rassemblant ces deux codes avec un nouveau bit de transition. Remarquez que ce SRE est un code à contrainte de suffixe. La démonstration de la propriété annoncée repose sur le calcul de la quantité $f_t = \mathbb{P}(B_t^1 = 0)$:

$$f_t = \sum_{i \in [1..|\mathcal{A}|], j \in [1..2]} \mathbb{P}(R_t = r_{i,j}, B_t^1 = 0) \tag{8}$$

$$= \sum_{i \in [1..|\mathcal{A}|], j=1} \mathbb{P}(S_t = a_i, B_{t+1}^1 = l_{i,1}^{L(l_{i,1})}) \tag{9}$$

$$= \sum_{i \in [1..|\mathcal{A}|], j=1} \mathbb{P}(S_t = a_i, L_t^{L(L_t)} = 0)\, f_{t+1}$$

$$+ \sum_{i \in [1..|\mathcal{A}|], j=1} \mathbb{P}(S_t = a_i, L_t^{L(L_t)} = 1)\,(1 - f_{t+1}). \tag{10}$$

Ainsi cette probabilité vérifie la relation de récurrence

$$f_t = \alpha\, f_{t+1} + (1-\alpha)(1-f_{t+1}), \tag{11}$$

où

$$\alpha = \sum_{a_i \in \mathcal{A}} \mathbb{P}(a_i, l_{i,1}^{L(\bar{l}_{i,1})} = 0) \tag{12}$$

vérifie

$$0 < \alpha < 1, \tag{13}$$

ce qui permet d'appliquer le théorème du point fixe car la fonction $g(x) = \alpha\, x + (1-\alpha)(1-x)$ est contractante. En effet $g'(x) = 2\,\alpha - 1$ et avec l'inégalité stricte de l'équation 13 on déduit $|g'(x)| < 1$. Ce point fixe est 0.5. On en déduit alors la propriété annoncée en remarquant que deux mots de code opposés ont la même probabilité d'occurence, ce qui garantit l'équiprobabilité du 0 et du 1.

Références

[1] J. Ziv and A. Lempel, "A universal algorithm for data compression," *IEEE Trans. Inform. Theory*, vol. 23, pp. 337–343, 1977.

[2] H. Kieffer and E.-H. Yang, "Grammar-based codes: a new class of universal lossless source codes," *IEEE Trans. Inform. Theory*, vol. 46, pp. 737–754, 2000.

[3] D. Huffman, "A method for the construction of minimum redundancy codes," in *Proc. of the IRE*, vol. 40, 1952, pp. 1098–1101.

[4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991, ch. 5, pp. 78–124.

[5] H. Jégou and C. Guillemot, "Suffix-constrained codes for progressive and robust data compression: self-multiplexed codes," in *Proc. EUSIPCO'2004*, Sept. 2004, vienna.

[6] B. Tunstall, "Synthesis of noiseless compression codes," *Ph.D Dissertation, Georgia Institute of Technology, Atlanta*, 1967.