CONVERGENCE OF LEMPEL-ZIV ENCODERS

Marcelo S. Pinho and Weiler A. Finamore*
CETUC / PUC-Rio
Rua Marquês de São Vicente, 225
22453-900, Rio de Janeiro - RJ
E-mail : mpinho@cetuc.puc-rio.br

Resumo : A optimalidade de duas variações do codificador proposto por Ziv e Lempel é demonstrada. Estas variações, denominadas LZW e mLZ respectivamente, tem desempenho, na prática, melhores do que o codificador LZ78. O LZ78 não codifica alguns símbolos, denominados símbolos de inovação, que, do ponto de vista prático não é uma boa estratégia. O LZW e o mLZ não usam os símbolos de inovação explicitamente o que justifica os resultados melhores, o fato de apresentarem melhor desempenho, quando usados para comprimir arquivos de tamanho finito, não constituem no entanto uma garantia de convergência.

Abstract: The optimality of two variations of the encoder proposed by Ziv and Lempel is proved. These variations, which are called LZW and mLZ, respectively, achieve better practical results than the LZ78. The LZ78 does not encode some symbols(called inovation symbols) which is not a good strategy for practical applications. The LZW and mLZ do not explicitly use the inovation symbols which may explain the better practical results. This is not however a guarantee of optimality.

Keywords: Information theory, source coding, universal algorithms for data compression.

1. INTRODUCTION

One of the most popular encoders in the literature is the Lempel-Ziv parsing scheme [1]; also known as LZ78 algorithm. Many studies about the optimality of the LZ78 have already been done [1, 4], and was proved that LZ78 is optimum in many senses. An interesting analisys, done by Ziv and Lempel, proved that no information lossless encoder of finite order (ILF) outperforms (assimptoticaly) the LZ78 in the compression of any individual sequence.

After the LZ78 was proposed, many variations of this algorithm have been done [2, 3, 5]. Simulations results [6] showed that the variations proposed in [2] (called LZW algorithm) and in [3] (called mLZ algorithm) achieve better compression rates. However, these better performances obtained by simulation are not a guarantee of the optimality (in any sense) of the proposed variations.

When asked about LZ78 in [7], Ziv commented that the convergence is related to the parsing of the sequence in the largest possible number of distinct strings. To better understand this point consider that $c(a)$ is the largest number of distinct strings whose concatenation forms the sequence a. We then will need $c(a)log_2c(a)$ bits to encode a. Ziv had previously shown [8] that the quantity $\frac{c(a)log_2c(a)}{n}$ converges to the complexity of the sequence a, being as such, a bound for any ILF encoder.

A practical problem intrinsic to the LZ78 is that some symbols, called inovation symbols, are not encoded by the algorithm. The LZW and mLZ are variations were proposed to handle this question. Both variations do not make explicit use of inovation symbols. They do however parse the sequence in a number of phrases $c_{LZW}(a)$ (or $c_{mLZ}(a)$) which is greater than $c(a)$. So, the number of bits needed to encode a, is $c_{LZW}(a)log_2c_{LZW}(a)$ (or $c_{mLZ}(a)log_2c_{mLZ}(a)$) which is greater than $c(a)log_2c(a)$. One can not therefore rely on simple simulation to state that these algorithms are optimum.

In this paper it is proved the optimality of the LZW and mLZ (optimality in the sense that there is no ILF encoder can perform better, assimptoticaly). We will be using the following notation in the paper:

1. $\lceil x \rceil$ denote the smallest integer $\geq x$.
2. $|A|$ denote the cardinality of a given set A.
3. I_A denote a map from a given set A to $\mathcal{N}$ (the set of non negative integers), such that if $A = \{\alpha_0, \alpha_1, \ldots, \alpha_{|A|-1}\}$, then $I_A(\alpha_i) = i$, $0 \leq i \leq |A| - 1$.
4. $a_i^j = a_i a_{i+1} \ldots a_j$ denote a finite sequence of symbols a_k, $i \leq k \leq j$, that take values in a given set A.
5. s denote a string, which is a finite sequence.
6. $|s|$ denote the length of the string s. For example $|a_i^j| = j - i + 1$.
7. λ denote the null length string, i.e. $|\lambda| = 0$.
8. $A^n = \{a_1^n : a_i \in A;\ 1 \leq i \leq n\}$.
9. $A^* = \{\lambda\} \cup A^1 \cup A^2 \cup \cdots \cup A^k \cup \cdots$
10. $\pi(a_i^j)$ denote the longest prefix of a_i^j, i.e. $\pi(a_i^j) = a_i^{j-1}$. If $i = j$, we consider $\pi(a_i) = \lambda$.
11. $\mathcal{L}(a_i^j)$ denote the last symbol of a_i^j, i.e. $\mathcal{L}(a_i^j) = a_j$.
12. $\mathcal{F}(a_i^j)$ denote the first symbol of a_i^j, i.e. $\mathcal{F}(a_i^j) = a_i$.
13. $c(a_i^j)$ denote the largest number of distinct strings whose concatenation forms a_i^j.
14. $\phi_k(i)$ denote the binary represantation of the integer i using a block of k bits.

*This work was supported by CNPq, Grants 133198/94-4 and 300361/85-RN.

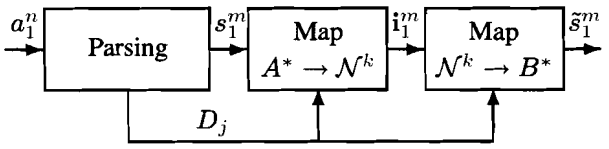

Figure 1: Basic Structure of Lempel-Ziv Encoders

The paper is organized as follows: in Section 2, we describe the LZ78 algorithm and its variations, i.e. LZW and mLZ; the analisys of LZ78 convergence is treated in Section 3; Our main results are showed in Section 4; Section 5 is devoted to the proofs; and in Section 6 the conclusions and some comments are presented.

2. LEMPEL-ZIV ENCODERS

The general denomination Lempel-Ziv encoder will be used to refer to the class formed by the LZ78 encoder and all its variations (we are not interested in the LZ77, which was proposed by Ziv and Lempel in [9], and its variations). The basic structure undelying of all encoders in this class is described next.

Let a be an infinite sequence. The Lempel-Ziv encoders break the sequence in (usually large) blocks of fixed length n ($a = a_1^n\, a_{n+1}^{2n} \ldots a_{pn+1}^{(p+1)n} \ldots$), and encode each of the fixed length block according to the same procedure, next to be discussed. Therefore we restrict the discussion to the first block a_1^n. This procedure is illustrated in figure 1.

1. **Parsing:** The block a_1^n is parsed in strings $s_j \in A^*$, such that $a_1^n = s_1^m$, $m \leq n$. At the same time a set $D_j = \{d_0, d_1, \ldots, d_{|D_j|-1}\}$, called dictionary, which is to be be used in the forthcoming steps, is constructed.
2. **Map $A^* \to \mathcal{N}^k$:** Maps the string s_j into a vector of integers $\mathbf{i}_j = (i_{1,j}; \ldots; i_{k,j})^T$, $1 \leq j \leq m$. In general the vector length k is variable and depends on s_j.
3. **Map $\mathcal{N}^k \to B^*$:** Maps the vector of integers into a string $\tilde{s}_j$, which take its value in the output set B^*. If $B = \{0, 1\}$, the encoder output is a sequence of bits.

In the framework of the structure just described the workings of the LZ78 algorithm and its variations will be shown.

2.1. LZ78

Parsing

1. Set $d_0 = \lambda$ and $D_0 = \{d_0\}$.
2. For $1 \leq j < m$, set s_j such that $\pi(s_j) \in D_{j-1}$ and $s_j \notin D_{j-1}$. Set $d_j = s_j$, and $D_j = D_{j-1} \cup \{d_j\}$.
3. For $j = m$, set s_j such that $\pi(s_j) \in D_{m-1}$.

Map $A^* \to \mathcal{N}$

1. For $1 \leq j \leq m$, set $i_j = I_{D_{j-1}}(\pi(s_j)).|A| + I_A(\mathcal{L}(s_j))$.

Map $\mathcal{N} \to B^*$

1. For $1 \leq j \leq m$, set $\tilde{s}_j = \phi_{\lceil \log_2(j|A|) \rceil}(i_j)$.

To ilustrate the algorithm, an example is given next. We want to encode the sequence $a = 0100011011\ldots$. After the parsing, the LZ78 finds

$$s_1 = 0,\ s_2 = 1,\ s_3 = 00,\ s_4 = 01,\ s_5 = 10,\ s_6 = 11,\ \ldots$$

and

$$D_6 = \{\lambda\} \cup \{0\} \cup \{1\} \cup \{00\} \cup \{01\} \cup \{10\} \cup \{11\}.$$

The map $A^* \to \mathcal{N}$ produce

$$i_1 = 0,\ i_2 = 1,\ i_3 = 2,\ i_4 = 3,\ i_5 = 4,\ i_6 = 5,\ \ldots,$$

and the LZ78 output, generated by the map $\mathcal{N} \to B^*$, is

$$\tilde{s}_1 = 0,\ \tilde{s}_2 = 01,\ \tilde{s}_3 = 010,\ \tilde{s}_4 = 011,$$
$$\tilde{s}_5 = 0100,\ \tilde{s}_6 = 0101,\ \ldots$$

2.2. LZW

Parsing

1. Set $D_0 = A$.
2. For $1 \leq j < m$, set s_j such that $s_j \in D_{j-1}$ and $s_j \mathcal{F}(s_{j+1}) \notin D_{j-1}$. Set $d_{j+|A|-1} = s_j \mathcal{F}(s_{j+1})$, and $D_j = D_{j-1} \cup \{d_{j+|A|-1}\}$.
3. For $j = m$, set s_j such that $s_j \in D_{m-1}$.

Map $A^* \to \mathcal{N}$

1. For $1 \leq j \leq m$, set $i_j = I_{D_{j-1}}(s_j)$.

Map $\mathcal{N} \to B^*$

1. For $1 \leq j \leq m$, set $\tilde{s}_j = \phi_{\lceil \log_2(j+|A|-1) \rceil}(i_j)$.

Considering the same sequence of the last example, i.e. $a = 0100011011\ldots$. The LZW parsing after seven steps then produces

$$s_1 = 0,\ s_2 = 1,\ s_3 = 0,\ s_4 = 00,$$
$$s_5 = 1,\ s_6 = 10,\ s_7 = 11,\ \ldots$$

and

$$D_7 = \{0, 1\} \cup \{01\} \cup \{10\} \cup \{00\} \cup$$
$$\cup \{001\} \cup \{11\} \cup \{101\}.$$

Therefore the map $A^* \to \mathcal{N}$ generates

$$i_1 = 0,\ i_2 = 1,\ i_3 = 0,\ i_4 = 4,$$
$$i_5 = 1,\ i_6 = 3,\ i_7 = 6,\ \ldots,$$

and the output of the LZW encoder is

$$\tilde{s}_1 = 0,\ \tilde{s}_2 = 01,\ \tilde{s}_3 = 00,\ \tilde{s}_4 = 100,$$
$$\tilde{s}_5 = 001,\ \tilde{s}_6 = 011,\ \tilde{s}_7 = 110,\ \ldots$$

2.3. mLZ

Parsing

1. Set $d_0 = \lambda$ and $D_0 = \{d_0\}$.
2. For $1 \leq j < m$, set s_j such that
   i. Either $s_j \notin D_{j-1}$ and $\pi(s_j) = \lambda$,
   ii. Or $s_j \in D_{j-1}$ and $s_j\mathcal{F}(s_{j+1}) \notin D_{j-1}$.

   If (i) is true, then set $d_j = s_j$;
   else, set $d_j = s_j\mathcal{F}(s_{j+1})$.
   Then set $D_j = D_{j-1} \cup \{d_j\}$.
3. For $j = m$, set s_j such that
   i. Either $s_j \notin D_{m-1}$ and $\pi(s_j) = \lambda$,
   ii. Or $s_j \in D_{m-1}$.

Map $A^* \rightarrow \mathcal{N}^k$

1. For $1 \leq j \leq m$, if $s_j \notin D_{j-1}$,
   then set $k = 2$, $i_{1,j} = 0$ and $i_{2,j} = I_A(s_j)$;
   else set $k = 1$ and $i_{1,j} = I_{D_{j-1}}(s_j)$.

Map $\mathcal{N}^k \rightarrow B^*$

1. For $1 \leq j \leq m$, if $(i_1)_j = 0$
   then set $\tilde{s}_j = \phi_{\lceil \log_2(j) \rceil}((i_1)_j).\ \phi_{\lceil \log_2 |A| \rceil}((i_2)_j)$, where dot means concatenated strings;
   else set $\tilde{s}_j = \phi_{\lceil \log_2(j) \rceil}((i_1)_j)$.

The output of each step when the sequence $a = 0100011011\ldots$ is encoded by mLZ is shown below

$$s_1 = 0,\ s_2 = 1,\ s_3 = 0,\ s_4 = 00,$$
$$s_5 = 1,\ s_6 = 1,\ s_7 = 0,\ s_8 = 11,\ \ldots,$$

$$D_7 = \{\lambda\} \cup \{0\} \cup \{1\} \cup \{00\} \cup$$
$$\cup\{001\} \cup \{11\} \cup \{10\} \cup \{01\},$$

$$\mathrm{i}_1 = (0,0)^T,\ \mathrm{i}_2 = (0,1)^T,\ i_3 = 1,\ i_4 = 3,$$
$$i_5 = 2,\ i_6 = 2,\ i_7 = 1,\ i_8 = 5,\ \ldots,$$

and

$$\tilde{s}_1 = 0,\ \tilde{s}_2 = 01,\ \tilde{s}_3 = 01,\ \tilde{s}_4 = 11, \tilde{s}_5 = 010$$
$$\tilde{s}_6 = 010,\ \tilde{s}_7 = 001,\ \tilde{s}_8 = 101,\ \ldots$$

3. CONVERGENCE OF LZ78

In this section we show some results for the LZ78 algorithm. Many proofs of optimality have been done in books and papers. The optimality is not regarded in the same sense in all works. The most commom proofs, like the proof in [4], show that for any ergodic source the rate of LZ78 almost surely converges to the source entropy. This is different from the original analisys [1], which proves that no one ILF encoder can be (assimptoticaly) better than the LZ78, for any individual sequence. In the original paper [1], Ziv and Lempel also showed that if a sequence is drawn from a ergodic source, then the rate of LZ78 almost surely converges to the source entropy. The proofs in this work follow the same lines of original paper analisys.

An outiline of the proof of convergence of LZ78 algorithm, in the same lines as [1] is presented in this section. For a detailed proof the reader can refer to [1]. The following definitions will be needed.

1. $\rho_E(a_1^n)$ is the compression rate for a_1^n achieved by an ILF encoder E.
2. $\rho_{E(s)}(a_1^n) = min_{E \in E(s)} \rho_E(a_1^n)$, where $E(s)$ is the class of all ILF with input alphabet $|A|$ and number of states $|S| \leq s$.
3. $\rho_{E(s)}(a) = limsup_{n \rightarrow \infty} \rho_{E(s)}(a_1^n)$.
4. $\rho(a) = lim_{s \rightarrow \infty} \rho_{E(s)}(a)$ is the compressibility of a.
5. $\hat{H}_l(a_1^n)$ is the normalized l-order 'entropy', which as obtained from the relative frequence taken from a_1^n.
6. $\hat{H}_l(a) = limsup_{n \rightarrow \infty} \hat{H}_l(a_1^n)$.
7. $\hat{H}(a) = lim_{l \rightarrow \infty} \hat{H}_l(a)$.
8. The compression rate for an infinite sequence a achieved by a Lempel-Ziv encoder LZ is defined by

$$\rho_{LZ}(a, n) = \limsup_{k \rightarrow \infty} \frac{1}{k} \sum_{i=1}^{k} \rho_{LZ}(a_{(i-1)n+1}^{in}),$$

where n is the length of the break in the infinite sequence.

In [1] Ziv and Lempel compare the compression rate achieved by the LZ78 encoder when compressing the sequence a to the compressibility of the sequence a. Even though the compressibility refers to an individual sequence, it can whatsoever be compared to the entropy of a source, as shown by the development that follows.

Lemma 1 *For any given ILF encoder E with $s = |S|$ states*

$$K \doteq \sum_{w \in A^l} 2^{-L(W)} \leq s^2 \left(1 + log_2 \frac{s^2 + |A|^l}{s^2}\right), \quad (1)$$

where

$$L(W) = \min_{z \in S} \{L(f(z, w))\} \quad (2)$$

and $L(f(z,w))$ is the length in bits of the string $f(z,w)$ output by E when in the initial state z is driven by the sequence w.

Theorem 1 *For every infinite sequence a*

$$\hat{H}(a) = \rho(a).$$

Theorem 2 *If a is drawn from an ergodic source with entropy H then*

$$Pr[\rho(a) = H] = 1.$$

Before stating the main LZ78 result is two theorems ought to be mentioned: the first, which is called converse-to-coding theorem, was introduced by Ziv and Lempel [1]; the other was also proved by Ziv and Lempel [8]. These theorems are the basis to the proof of the main theorem.

Theorem 3 *For every $a_1^n \in A^n$*

$$\rho_{E(s)}(a_1^n) \geq \frac{c(a_1^n) + s^2}{n log_2|A|} log_2 \frac{c(a_1^n) + s^2}{4s^2} + \frac{2s^2}{n log_2|A|}.$$

Theorem 4 *For every $a_1^n \in A^n$*

$$m - 1 \leq c(a_1^n) < \frac{n \log_2|A|}{(1 - \epsilon_n)\log_2(n)}, \tag{3}$$

where m is the number of strings produced by the LZ78 parsing and $\lim_{n\to\infty} \epsilon_n = 0$

The central result which establishes the convergence of the LZ78 encoder is given by the next theorem.

Theorem 5 *For every infinite sequence a*

$$\lim_{n\to\infty} \rho_{LZ78}(a, n) = \rho(a).$$

4. CONVERGENCE OF LZW AND mLZ

Our main results are the convergence, in the same sense considered in section 3., of the LZW an mLZ algorithms. The proof of convergence in the sense given in [4] also can be done. It is an easy task to extend the proof given in [4] to LZW and mLZ, and we therefore ommit it. Although our proof is more intricate, it is also more general. It guarantees that LZW and mLZ are assymptotically optimal not only for sequence for an ergodic source, but that they are also optimal for every infinite sequence. To prove our main results we need a lemma, which is given below.

Lemma 2 *Let a be an infinite sequence and E_1 be a map, which can be thought of as an ILF encoder. Consider $\tilde{a}_1^{\tilde{n}}$ to be the output of E_1, when driven by a_1^n, and that the encoder input and output alphabet are the same set A. If $\limsup_{n\to\infty} \frac{n}{\tilde{n}} = R$, then*

$$\rho(\tilde{a}) \leq R\rho(a). \tag{4}$$

The convergence of LZW and mLZ are given by the following theorems.

Theorem 6 *For every infinite sequence a*

$$\lim_{n\to\infty} \rho_{LZW}(a, n) = \rho(a).$$

Theorem 7 *For every infinite sequence a*

$$\lim_{n\to\infty} \rho_{mLZ}(a, n) = \rho(a).$$

5. PROOFS

Although the proofs of some theorems and lemmas mentioned in this paper can be found in the references, we show all proofs in this paper. These proofs are showed (in this section) to make the reading easy.

To make the discussion simpler, with no loss of generality we assume that the encoder output alphabet is binary.

PROOF (Lemma 1): Let k_j denote the number of strings $w \in A^l$ for which $L(w) = j$. Then $K = \sum_j k_j 2^{-j}$ and $|A|^l = \sum_j k_j$. By the ILF property of E, it is clear that $k_j \leq s^2 2^j$. It is also clear that to obtain an upper bound on K, we may overestimate k_j, $j = 0, 1, \ldots$, at the expense of $\sum_{i>j} k_i$, provided the sum of all k_j remains equal to $|A|^l$. We can thus write

$$K \leq \sum_{j=0}^{M} (s^2 2^j) 2^{-j} = s^2(M + 1), \tag{5}$$

where M is the integer satifying

$$\sum_{j=0}^{M-1} s^2 2^j < \alpha^l \leq \sum_{j=0}^{M} s^2 2^j.$$

Furthermore

$$2^M = 1 + \sum_{j=0}^{M-1} 2^j \leq \frac{|A|^l}{s^2} + 1,$$

which together with (5) yields (1).

Q.E.D.

PROOF (Theorem 1): From the definition of $L(w)$ in (2), it is clear that for any ILF encoder with s states

$$\begin{aligned}
\rho_{E(s)}(a_1^n) &= \frac{1}{n log_2|A|} \sum_{i=1}^{n} L(f(z_i, a_i)) \\
&= \frac{1}{l n log_2|A|} \sum_{i=1}^{n} l L(f(z_i, a_i)) \\
&\geq \frac{1}{l n log_2|A|} \sum_{i=1}^{n-l} L(f(z_i, a_{i+1}^{i+l})) \\
&\geq \frac{1}{l n log_2|A|} \sum_{i=1}^{n-l} L(a_{i+1}^{i+l}).
\end{aligned} \tag{6}$$

Considering the definition for the relative frequence of a string w with respect to a sequence a_1^n,

$$P(a_1^n, w) = \frac{1}{n - l + 1} \sum_{i=0}^{n-l} \delta(a_{i+1}^{i+l}, w).$$

We can rewrite (6) as

$$\rho_{E(s)}(a_1^n) \geq \frac{n - l + 1}{l n log_2|A|} \sum_{w \in A^l} P(a_1^n, w) L(W),$$

and obtain

$$\rho_{E(s)}(a) \geq \frac{1}{l n log_2|A|} \limsup_{n\to\infty} \sum_{w\in A^l} P(a_1^n, w) L(W).$$

By the definition of $\hat{H}_l(a_1^n)$ and $\hat{H}_l(a)$, we have

$$\hat{H}_l(a) - \rho_{E(s)}(a) \leq$$

$$\frac{1}{l log_2|A|} \limsup_{n\to\infty} \sum_{w\in A^l} P(a_1^n, w) \left(log_2 \frac{1}{P(a_1^n, w)} - L(W) \right)$$

$$= \frac{1}{l log_2|A|} \limsup_{n\to\infty} P(a_1^n, w) log_2 \frac{2^{-L(w)}}{P(a_1^n, w)},$$

which, by the convexity of the logarithm function and from Lemma 1, reduces to

$$\begin{aligned} \hat{H}_l(a) - \rho_{E(s)}(a) &\leq \frac{1}{l log_2|A|} \limsup_{n\to\infty} log_2 \sum_{w\in A^l} 2^{-L(w)} \\ &= \frac{s^2}{l log_2|A|} \left(1 + log_2 \frac{|A|^l}{s^2} \right). \end{aligned}$$

Taking the limit as l approaches infinity yields

$$\hat{H}(a) - \rho_{E(s)}(a) \leq 0, \tag{7}$$

and since(7) holds for every finite s, we have

$$\hat{H}(a) \leq \rho(a) \tag{8}$$

for every infinite sequence a.

Using Huffman's coding scheme for input blocks of length l, it is easy to show [4] that

$$\rho(a) \leq \hat{H}_l(a) + \frac{log_2|A|}{l},$$

which when l tends to infinity becomes

$$\rho(a) \leq \hat{H}(a) \tag{9}$$

for all a.

Combining(8) with (9) completes the proof.

Q.E.D.

PROOF (Theorem 2): Since a is drawn from an ergodic source, it follows that for every $w \in A^l$

$$Pr[P(a, w) = Pr(w)] = 1,$$

where $P(a, w) = \lim_{n\to\infty} P(a_1^n, w)$ and $Pr(w)$ is the probability measure of w.

If we now take

$$H_l = \frac{-1}{l} \sum_{w\in A^l} Pr(w) log_2 Pr(w)$$

we then get

$$Pr[\hat{H}_l(a) = H_l] = 1,$$

which, when l approaches infinity, becomes

$$Pr[\hat{H}(a) = H] = 1.$$

From this and Theorem 1, we obtain Theorem 2.

Q.E.D.

PROOF (Theorem 3): Given an encoder $E \in E(s)$ and an input string a_1^n, let

$$a_1^n = s_1 s_2 \ldots s_c$$

be the parsing of a_1^n into $c = c(x_1^n)$ distinct phrases, and let c_j denote the number of phrases s_i, for which $\tilde{s}_i$, the corresponding output phrases, is j-bits long. Since the input phrases are all distinct, it follows from the ILP property of E that $c_j \leq s^2 2^j$ for all j. It is also clear that to obtain a lower bound on the length $L(\tilde{s}_1^c)$ in bits of $\tilde{s}_1^c$, we may overestimate c_j, $j = 0, 1, \ldots$, at the expense of $\sum_{i>j} c_i$, provided the sum of all c_j remains equal to c. Thus if q and r are the nonnegative integers satisfying $c = qs^2 + r$, and if

$$q = \sum_{j=0}^{k} 2^j + \Delta_k, \qquad 0 \leq \Delta_k < 2^{k+1},$$

then we may assume that $c_j = s^2 2^j$ for $0 \leq j \leq k$, $c_{k+1} = s^2\Delta_k + r$, and $c_j = 0$ for $j > k+1$. Therefore

$$c = s^2 \sum_{j=0}^{k} 2^j + s^2\Delta_k + r = s^2(2^{k+1} + t), \tag{10}$$

where

$$t = \Delta_k - 1 + \frac{r}{s^2}, \tag{11}$$

and

$$\begin{aligned} L(\tilde{s}_1^c) &\geq \sum_{j=0}^{k} j2^j + (k+1)(s^2\Delta_k + r) \\ &= s^2[(k-1)2^{k+1} + 2] + (k+1)(s^2\Delta_k + r) \\ &= s^2(k-1)(2^{k+1} + t) + s^2(k+3+2t) \\ &= (k-1)(c+s^2) + 2s^2(t+2). \end{aligned} \tag{12}$$

From (10) we have

$$\begin{aligned} k-1 &= log_2 \frac{c - s^2 t}{s^2} - 2 \\ &= log_2 \frac{c+s^2}{4s^2} - log_2 \left[1 + \frac{(t+1)s^2}{c - s^2 t} \right], \end{aligned}$$

which together with (12) yields

$$L(\tilde{s}_1^c) \geq (c+s^2) \left(log_2 \frac{c+s^2}{4s^2} + \tau \right), \tag{13}$$

where

$$\tau = \frac{2s^2(t+2)}{c+s^2} - log_2 \left[1 + \frac{(t+1)s^2}{c - ts^2} \right].$$

Let $\phi = \frac{(t+1)s^2}{c - ts^2}$. Then

$$\tau = \frac{2s^2}{c+s^2} + \frac{2\phi}{1+\phi} - log_2(1+\phi),$$

and by (10) and (11) we have

$$\phi = \left(\Delta_k + \frac{r}{s^2}\right) 2^{-(k+1)}.$$

From the definitions of Δ_k and r it follows that $0 \leq \phi < 1$, and one can readily verify that over this interval, $2\phi \geq (1+\phi)log_2(1+\phi)$. Hence $\tau \geq \frac{2s^2}{c+s^2}$, which together with (13) yields

$$L(\tilde{s}_1^c) \geq (c+s^2)log_2\frac{c+s^2}{4s^2} + 2s^2.$$

Dividing both sides of this inequality by $nlog_2|A|$, we conclude the proof.

Q.E.D.

PROOF (Theorem 4): The first part of the theorem is easy to prove since the LZ78 algorithm parses a_1^n into $m-1$ distinct strings (except the last); and $c(a_1^n)$ is the largest number of distinct strings whose concatenation forms a_1^n. Therefore

$$m - 1 \leq c(a_1^n).$$

To proof the other part, let s_1^c be the distinct strings whose concatenation forms a_1^n. For a given n, a bound on $c(a_1^n)$ (dnoted by c in this proof, just to simplify) can be found considering that a_1^n can be parsed into all strings of length less than $l+1$, and some strings of length $l+1$. So we have

$$n \geq |A| + 2|A| + \ldots + l|A|^l + (l+1)\delta,$$

or

$$n \geq \frac{|A|}{|A|-1}\left[|A|^l\left(l - \frac{1}{|A|-1}\right) + \frac{1}{|A|-1}\right] + (l+1)\delta,$$

where δ is the number of strings of length $l+1$ in a_1^n. After some manipulations, we can obtain

$$\begin{aligned}\frac{|A|}{|A|-1}|A|^l &\leq \left[n - (l+1)\delta - \frac{|A|}{|A|-1}\right]\left[l - \frac{1}{|A|-1}\right]^{-1} \\ &\leq \frac{n-(l+1)\delta}{l-\frac{1}{|A|-1}} \\ &\leq \frac{n-(l+1)\delta}{l-1}. \end{aligned} \quad (14)$$

We can upperbound c by

$$\begin{aligned} c &\leq |A| + |A|^2 + \ldots |A|^l + \delta \\ &= \frac{|A|}{|A|-1}(|A|^l - 1) + \delta \\ &\leq \frac{|A|}{|A|-1}|A|^l + \delta. \end{aligned} \quad (15)$$

Combining (14) and (15), we have

$$\begin{aligned} c &\leq \frac{n-(l+1)\delta}{l-1} + \delta \\ &\leq \frac{n}{l-1}. \end{aligned} \quad (16)$$

It is easy to verify that

$$l \leq log_{|A|}n,$$

and

$$\begin{aligned} log_{|A|}n &\leq log_{|A|}(\sum_{i=1}^{l+1} i|A|^i) \leq log_{|A|}((l+1)^2|A|^{l+1}) \\ &= (l+1) + 2log_{|A|}(l+1). \end{aligned} \quad (17)$$

Thus

$$l - 1 \geq log_{|A|}n - 2log_{|A|}(log_{|A|}n + 1) - 2. \quad (18)$$

Defining ϵ by

$$\epsilon = 2\frac{log_{|A|}(log_{|A|}n+1)+1}{log_{|A|}n},$$

then $\epsilon \to 0$ when $n \to \infty$, and combining (16) and (18), we can obtain (3).

Q.E.D.

PROOF (Theorem 5): The total number of bits that go into the coding of an input block a_1^n that is parsed incrementally into m words is given by

$$L = \sum_{j=1}^{m} L_j = \sum_{j=1}^{m} \lceil log_2(j|A|) \rceil.$$

Hence

$$L \leq \sum_{j=1}^{m} log_2(2j|A|) \leq m(log_2 m + log(2|A|)).$$

Since $m - 1 \leq c(a_1^n)$ (or c), it follows that the compression rate attainable by the LZ78 for a_1^n satisfies

$$\rho_{LZ78}(a_1^n) \leq \frac{c+1}{nlog_2|A|}log_2\left[2|A|(c+1)\right]. \quad (19)$$

From (19) and Theorem 2, we obtain

$$\rho_{LZ78}(a_1^n) \leq \rho_{E(s)}(a_1^n) + \frac{\Delta(c)}{nlog_2|A|}, \quad (20)$$

where

$$\begin{aligned}\Delta(c) &= (c+1)log_2\frac{2|A|(c+1)}{c+s^2} - (s^2-1)log_2(c+s^2) \\ &\quad +(c+s^2)log_2(4s^2). \end{aligned} \quad (21)$$

From (21) one can see that $\Delta(c)$ increases with c, which by (3) implies

$$\delta_s(n) \doteq \frac{1}{nlog_2|A|}\Delta\left(\frac{nlog_2|A|}{(1-\epsilon_n)log_2 n}\right) \geq \frac{\Delta(c)}{nlog_2|A|}. \quad (22)$$

It is easy to verify that

$$\lim_{n\to\infty} \delta_s(n) = 0,$$

which together with (20) and (22) proves that

$$\rho_{LZ78}(a_1^n) \leq \rho_{E(s)}(a_1^n) + \delta_s(n). \quad (23)$$

The compression rate of LZ78 for an infinite sequence a can be computed by definition

$$\rho_{LZ78}(a,n) = \limsup_{k\to\infty} \frac{1}{knlog_2|A|} \sum_{p=0}^{k} L_p, \quad (24)$$

where L_p is the number of bits used to encode the $(p+1)th$ block $a_{pn+1}^{(p+1)n}$. Since (19) and (23) hold for any input block of length n, we can write

$$\frac{L_p}{nlog_2|A|} = \rho_{LZ78}(a_{pn+1}^{(p+1)n}) \leq \rho_{E(s)}(a_{pn+1}^{(p+1)n}) + \delta_s. \quad (25)$$

From (24) and (25) we obtain

$$\rho_{LZ78}(a,n) \leq \delta_s(n) + \limsup_{k\to\infty} \frac{1}{k} \sum_{p=0}^{k} \rho_{E(s)}(a_{pn+1}^{(p+1)n}),$$

which reduces to

$$\rho_{LZ78}(a,n) \leq \delta_s(n) + \rho_{E(s)}(a),$$

Since $\rho_{LZ78}(a,n) \geq \rho(a)$ for every n by definition, we can complete the proof taking the limit when $n \to \infty$.

Q.E.D.

PROOF (Lemma 2): We can affirm by the definition of $\rho_{E(s)}(a_1^n)$ that it exist an ILF encoder E_2 such that $\rho_{E_2}(a_1^n) = \rho_{E(s)}(a_1^n)$.

Let E_3 be an encoder defined by the composition of E_1^{-1} with E_2, i.e. $E_3 = E_2 \circ E_1^{-1}$. Then we can write

$$\rho_{E(s)}(\tilde{a}_1^{\tilde{n}}) = \frac{L}{\tilde{n}log_2|A|},$$

where L is the output length of E_2 when it is fed with a_1^n. Hence

$$\rho_{E_3}(\tilde{a}_1^{\tilde{n}}) = \frac{n}{\tilde{n}}\rho_{E_2}(a_1^n) = \frac{n}{\tilde{n}}\rho_{E(s)}(a_1^n).$$

Since E_3 is a composition of two ILF encoders, it is also an ILF encoder. Considering s_3 the number of states of E_3. Then $\forall s, \exists\, s' \geq s_3$, which is a function of s, such that

$$\rho_{E(s')}(\tilde{a}_1^{\tilde{n}}) \leq \frac{n}{\tilde{n}}\rho_{E(s)}(a_1^n).$$

taking the lim sup when $n \to \infty$, we obtain

$$\limsup_{n\to\infty} \rho_{E(s')}(\tilde{a}_1^{\tilde{n}}) \leq \limsup_{n\to\infty} \frac{n}{\tilde{n}}\rho_{E(s)}(a_1^n),$$

and by the lim sup property, we have

$$\rho_{E(s')}(\tilde{a}) \leq \rho_{E(s)}(a) \limsup_{n\to\infty} \frac{n}{\tilde{n}} = R\rho_{E(s)}(a). \quad (26)$$

Since the (26) holds for every s, the proof is completed.

Q.E.D.

PROOF (Theorem 6): To show this theorem we consider an encoder which maps a_1^n into $\tilde{a}_1^{\tilde{n}} = \alpha_0 \ldots \alpha_{|A|-1}\beta_1 \ldots \beta_{m-1}s_m$, where α_j's are the symbols of the input alphabet A and β_j's are the symbols from the dictionary (produced by the LZW parsing). It is easy to see that this encoder is an ILF encoder. It is also easy to see that

$$c_{\tilde{a}} = c(\tilde{a}_1^{\tilde{n}}) \geq m-1,$$

$$n = \tilde{n} - (m-1) - |A|,$$

and

$$n < \tilde{n} \leq n + c_{\tilde{a}} + |A| \leq 2n. \quad (27)$$

From (3) we can obtain

$$\lim_{\tilde{n}\to\infty} \frac{c_{\tilde{a}}}{\tilde{n}} = 0.$$

Furthermore

$$\lim_{n\to\infty} \frac{c_{\tilde{a}}}{n} = 0, \quad (28)$$

and

$$\lim_{n\to\infty} \frac{\tilde{n}}{n} = 1. \quad (29)$$

Let $LZ78(\tilde{a}_1^{\tilde{n}})$ be the output of the LZ78 encoder when the input is $\tilde{a}_1^{\tilde{n}}$. Let $LZW(a_1^n)$ be the output of the LZW encoer when driven by a_1^n. It is easy to see that $|LZ78(\tilde{a}_1^{\tilde{n}})| > |LZW(a_1^n)|$ (because of the first symbols). The LZW encoder assign a total number L of bits to the coding of an input block a_1^n, that is parsed into m words, which can be upperbounded by

$$\begin{aligned} L = \sum_{j=1}^{m} L_j &\leq \sum_{j=1}^{m} log_2[2(j|A|-1)] \\ &\leq mlog_2[2(|A|+m-1)]. \end{aligned}$$

Therefore

$$L \leq (c_{\tilde{a}}+1)log_2[2(|A|+c_{\tilde{a}})]. \quad (30)$$

From (30) and Theorem 3, we can obtain

$$\rho_{LZW}(a_1^n) \leq \rho_{E(s)}(\tilde{a}_1^{\tilde{n}}) + \frac{\Delta(c_{\tilde{a}},n,\tilde{n})}{\tilde{n}log_2|A|}, \quad (31)$$

where

$$\begin{aligned} \Delta(c_{\tilde{a}},n,\tilde{n}) &= \frac{\tilde{n}}{n}(c_{\tilde{a}}+1)log_2[2(|A|+c_{\tilde{a}})] \\ &\quad -(c_{\tilde{a}}+s^2)log_2(c_{\tilde{a}}+s^2) \\ &\quad +(c_{\tilde{a}}+s^2)log_2(4s^2)+2s^2. \end{aligned}$$

We can upperbound $\Delta(c_{\tilde{a}},n,\tilde{n})$ by

$$\Delta(c_{\tilde{a}},n,\tilde{n}) \leq \tilde{\Delta}(c_{\tilde{a}},n),$$

where

$$\begin{aligned}\tilde{\Delta}(c_{\tilde{a}},n) &= (c_{\tilde{a}}+1)log_2\frac{2(|A|+c_{\tilde{a}})}{c_{\tilde{a}}+s^2} \\ &-(s^2-1)log_2(c_{\tilde{a}}+s^2) \\ &+(c_{\tilde{a}}+s^2)log_2(4s^2)+2s^2 \\ &+\frac{c_{\tilde{a}}}{n}(c_{\tilde{a}}+1)log_2[2(|A|+c_{\tilde{a}})]. \quad (32)\end{aligned}$$

It is possible to see that $\tilde{\Delta}(c_{\tilde{a}},n)$ increase with $c_{\tilde{a}}$, and from (27) and Theorem 4 we can obtain

$$\begin{aligned}\delta_s(n) &= \frac{1}{nlog_2|A|}\tilde{\Delta}\left(\frac{2nlog_2|A|}{(1-\epsilon_n)log_2n},\right) \\ &\geq \frac{\tilde{\Delta}(c_{\tilde{a}},n)}{\tilde{n}log_2|A|}, \quad (33)\end{aligned}$$

and analisying (28), (32) and (33), we can conclude that

$$\lim_{n\to\infty}\delta_s(n)=0.$$

Since (31), (32) and (33) hold for every input block of length n, we can write

$$\rho_{LZW}(a_{pn+1}^{(p+1)n}) \leq \rho_{E(s)}(\tilde{a}_{\tilde{n}_p}^{\tilde{n}_{p+1}-1}),$$

where $\tilde{n}_0 = 1$ and $\tilde{n}_p$ is a function of the $(p+1)th$ block. Therefore

$$\rho_{LZW}(a,n) \leq \delta_s + \limsup_{k\to\infty}\frac{1}{k}\sum_{p=0}^{k}\rho_{E(s)}(\tilde{a}_{\tilde{n}_p}^{\tilde{n}_{p+1}-1}),$$

and

$$\rho_{LZW}(a,n) \leq \delta_s(n) + \rho_{E(s)}(\tilde{a}).$$

With steps analogous to those on the proof of Theorem 6, we can write

$$\rho_{LZW}(a,n) \leq \rho(\tilde{a}) + \delta_s(n).$$

From (29) and Lemma 2, we get

$$\rho(\tilde{a}) \leq \rho(a),$$

and then

$$\rho_{LZW}(a,n) \leq \rho(a) + \delta_s(n).$$

Since $\rho_{LZW}(a,n) \geq \rho(a)$ for every n, by definition, we can complete the proof taking the limit when $n \to \infty$.

Q.E.D.

The proof of the Theorem 7 is analogous that of the Theorem 6. The only difference is on how encoder is set up. We can choose an encoder which maps a_1^n into $\tilde{a}_1^{\tilde{n}} = \beta_1 \ldots \beta_{m-1}s_m$, and by arguing the same as in the proof of theorem 6, the convergence of mLZ can be shown.

6. CONCLUSION

In this work we proved the optimality of two versions of the LZ78 encoder [1], optimality in the sense that no ILF encoder can perform better (assimptoticaly) then these versions. These versions were proposed in [2] (the LZW encoder), and in [3] (the mLZ encoder). Using previous results (Theorem 1 and 2), we can conclude that the rates of these encoders converge almost surely to the source entropy, for every ergodic source.

The redundancy of variations of Lempel-Ziv encoders have recently been computed [10, 11, 12]. The redundancy shows us how the rate of an encoder converges to the source entropy. In [11], Savari proved that the LZ78 and LZW converge as $O(\frac{1}{log_2n})$ for a markovian source, which is better than the LZ77, which was proved in [10], to converge as $O(\frac{log_2log_2n}{log_2n})$. A natural question is how the mLZ converge. We conjecture that mLZ converge as fast as LZW, i.e. with $O(\frac{1}{log_2n})$, since its parsing is similar to the LZW parsing.

REFERENCES

[1] J. ZIV and A. LEMPEL, "Compression of individual sequences via variable-Rate Coding," IEEE Trans. Inform. Theory, vol. IT-24, pp. 530-536, Sep. 1978.

[2] T. A. WELCH, "A technique for high-performance data compression," Computer, vol. 17, pp. 8-19, Jun. 1984.

[3] W. A. FINAMORE and P. R. R. L. NUNES, "A modified Lempel-Ziv algorithm and its aplication to errorless image compaction," IEEE International Conference on Acoustic, Speech and Signal Processing, pp. 14-17, Toronto, ON, Canada, May 1991.

[4] T. M. Cover and J. A. Thomas, Elements of Information Theory. New York, Wiley, 1991.

[5] W. A. FINAMORE, P. R. R. L. NUNES and A. SILVA, "An alternative node labelling on Lempel-Ziv trees," Proceedings of the Abstracts, IEEE Inform. Theory Workshop, pp. 2-3, Salvador, Brazil, 1992.

[6] M. S. PINHO, Compressão de Dados via Codificadores Universais, de Estado Finito e sem Perda de Informação. Master thesis, DEE/PUC-Rio, Feb. 1996.

[7] A. WYNER, "A conversation with Jacob Ziv," IEEE Inform. Theory Soc. Newsletter, Vol. 46, Dec., 1996.

[8] A. LEMPEL and J. ZIV, "On the complexity of finite sequences," IEEE Trans. Inform. Theory, vol. IT.22, pp. 75-81, Jan. 1976.

[9] J. Ziv and A. Lempel, "A universal algorithm for sequencial data compression" IEEE Trans. Inform. Theory, vol. IT-23, pp. 337-343, May 1977.

[10] E. Plotnik, M. J. Weinberg and J. Ziv, "Upper Bounds on the probability of sequences emitted by finite-state sources and on the redundancy of the Lempel-Ziv algorithm," IEEE Trans. Inform. Theory, vol. 38, pp. 66-72, 1992.

[11] S. A. Savari, "Redundancy of Lempel-Ziv incremental parsing rule," IEEE Trans. Infrom. Theory, vol. 43, pp. 9-21, Jan. 1997.

[12] S. A. Savari, "Redundancy of the Lempel-Ziv string matching code," IEEE Trans. Infrom. Theory, vol. 44, Mar. 1998.

Marcelo Pinho received the B.S. and M.S. degree in electrical engineering from Pontificia Universidade Catolica do Rio de Janeiro, in 1994 and 1996, respectively. He is currently a D.Sc. candidate at the Pontificia Universidade Catolica do Rio de Janeiro and now he is at Rensselear Polytechnic Institute, Troy - NY, in the SWE program, supported by CNPq. His main research interests are in the area of source coding and image processing.

Weiler A. Finamore was born in Cisneiros, Minas Gerais, on October 16, 1944. He graduated in Electrical Engineering at Pontif'icia Universidade Cat'olica do Rio de Janeiro (PUC-RJ) in 1969, and got his MS and PhDdegrees in Electrical Engineering from the University of Wisconsin, USA, in 1974 and 1978, respectively. He was a faculty member in the Department of Electrical Engineering of the Universidade Federal do Pará from 1970 to 1973 and during the 1978-79 academic year. Since 1979 he has been a faculty member in the Department of Electrical Engineering of PUC-RJ, working at its Center for Telecommunications Studies (CETUC). His research interests are on coding and information theory, digital image processing, and digital transmission.