Universal Data Compression and Repetition Times

FRANS M. J. WILLEMS, MEMBER, IEEE

Abstract—**A new universal data compression algorithm is described. This algorithm encodes L source symbols at a time. For the class of binary stationary sources, its rate does not exceed $(H(U_0, U_1, \cdots, U_{L-1}) + \lceil \log_2(L+1) \rceil)/L$ bits per source symbol. In our analysis, a property of repetition times turns out to be of crucial importance.**

I. Introduction

In a data compression situation an encoder observes the output stream of an information source and transforms it into a code stream. We assume that the information source is stationary. The code stream is sent to a decoder whose task is to reconstruct the source stream from the code stream. The rate of such a system is defined as the expected number of code symbols per source symbol.

When the source statistics are known, we can design encoder–decoder pairs with rates arbitrarily close to the entropy of the source. Rates smaller than the entropy of the source cannot be achieved.

A data compression algorithm is called universal if the corresponding encoder and decoder are designed without knowing the source statistics. Such a universal compression algorithm is optimal if encoder–decoder pairs that achieve rates as close to source entropy as desired can be constructed via this algorithm no matter what the statistics of the actual source are.

We present an optimal universal data compression method for binary sources and a binary code alphabet. Also, we describe a modification to this method that decreases the complexity. Both methods can easily be generalized to arbitrary source and code alphabets.

II. Statement of Result

Consider a binary source producing $\{u_t\}_{t=-\infty}^{\infty}$, a sequence of source outputs with values in the alphabet $\{0,1\}$. The integer t can be identified with time. Throughout this paper we assume that the source is stationary. (For definitions we refer to Gallager [1, sect. 3.5].)

The encoder chops the source output stream into source words $\boldsymbol{u}(k) := (u_{(k-1)L}, u_{(k-1)L+1}, \cdots, u_{kL-1})$ of length L (k integer). At time $t = kL$ the source word $\boldsymbol{u}(k)$ is transformed into a variable-length codeword $\boldsymbol{c}(k)$. The length of this codeword is $Lg(\boldsymbol{c}(k))$ code symbols. Just like the source symbols, the code symbols take values in the alphabet $\{0,1\}$. When forming the codeword $\boldsymbol{c}(k)$, the encoder uses

$$\Phi_e(k) := (u_{(k-1)L-B}, u_{(k-1)L-B+1}, \cdots, u_{(k-1)L-1}),$$

i.e., the B most recent source outputs. These outputs are assumed stored in a buffer. The encoder can now be described as follows:

$$\boldsymbol{c}(k) := F_e(\boldsymbol{u}(k), \Phi_e(k)). \tag{1}$$

We require that the set of codewords generated by the encoder satisfies the prefix condition (see Gallager [1, par. 3.2]).

Just like the encoder, the decoder uses a buffer of size B. At time kL the contents $\Phi_d(k)$ of this buffer are assumed to be equal to $\Phi_e(k)$. When the decoder receives the codeword $\boldsymbol{c}(k)$, it uses $\Phi_d(k)$ to form the replica $\boldsymbol{v}(k)$ of $\boldsymbol{u}(k)$. Hence

$$\boldsymbol{v}(k) := F_d(\boldsymbol{c}(k), \Phi_d(k)). \tag{2}$$

The code is a prefix code. Therefore, it is uniquely decodable, i.e., for all source words $\boldsymbol{u}$ and all buffer contents Φ,

$$\boldsymbol{u} = F_d(F_e(\boldsymbol{u}, \Phi), \Phi). \tag{3}$$

Consequently, $\boldsymbol{v}(k) = \boldsymbol{u}(k)$, and with this $\boldsymbol{v}(k)$ and $\Phi_d(k)$ the decoder forms $\Phi_d(k+1)$, i.e., it updates its buffer. Note that again $\Phi_d(k+1) = \Phi_e(k+1)$.

The rate R of the above coding system is defined as

$$R := \boldsymbol{E}[Lg(\boldsymbol{c}(k))]/L \tag{4}$$

where k is arbitrary. The expectation in (4) is evaluated using the statistics of the source being compressed.

In Sections IV and V of this paper we describe and analyze an encoder–decoder pair for which

$$B = 2^L - 1 \tag{5a}$$

$$R \le (H(U_0, U_1, \cdots, U_{L-1}) + \lceil \log(L+1) \rceil)/L. \tag{5b}$$

(All algorithms in this paper are assumed to have base 2, and $\lceil a \rceil$ is equal to the smallest integer $\ge a$.)

Because of (5b) and the fact that for any coding system $R \ge H_\infty(U)$, it is clear that (see again Gallager [1, sect. 3.5])

$$\lim_{L \to \infty} R = H_\infty(U) \tag{6}$$

Manuscript received April 28, 1986; revised August 3, 1987. This paper was presented at the Oberwolfach Information Theory Meeting, Oberwolfach, Germany, May 11–17, 1986.

The author is with the Electrical Engineering Department, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.

IEEE Log Number 8825701.

where $H_\infty(U)$ is the entropy of the (stationary) source. From (6) we conclude that our coding strategy is optimal.

Our analysis hinges on a crucial result concerning repetition times. The next section is devoted to this subject.

III. Repetition Times

A discrete stationary source generates the sequence $\{x_t\}_{t=-\infty}^{\infty}$. We say that the (value of the) repetition time M_t of the source output x_t is equal to m if $X_{t-n} \neq x_t$ for $1 \leq n < m$ and $X_{t-m} = x_t$. For x such that $P(X_0 = x) > 0$, the average repetition time $T(x)$ of x is defined as

$$T(x) := \sum_{m=1,\infty} mQ_m(x) \tag{7a}$$

where

$$Q_m(x) := P(M_t = m | X_t = x)$$
$$= P(X_{-m} = x, X_{1-m} \neq x, \cdots, X_{-1} \neq x | X_0 = x) \tag{7b}$$

for $m = 1,2,3,\cdots$ and arbitrary t (stationarity). Two important properties of repetition times are stated in the following lemma.

Lemma: For a discrete stationary source and for any x for which $P(X_0 = x) > 0$,

1) $\sum_{m=1,\infty} Q_m(x) = 1$, and
2) $P(X_0 = x) \cdot T(x) = 1 - \lim_{N \to \infty} P(X_0 \neq x, X_1 \neq x, \cdots, X_N \neq x)$.

Proof: See the Appendix.

Note that when the source $\{u_t\}_{t=-\infty}^{\infty}$ is stationary, the source $\{w_t\}_{t=-\infty}^{\infty}$ with $w_t := (u_{t-L}, u_{t-L+1}, \cdots, u_{t-1})$ is also stationary. Therefore, the lemma applies to the source $\{w_t\}_{t=-\infty}^{\infty}$ as well.

IV. The Algorithm

Rough Outline

Using its buffer the encoder determines the repetition time of the source word to be transmitted. This repetition time is transformed into a codeword that is sent to the decoder. Knowing this codeword, the decoder reconstructs the repetition time. With this repetition time the decoder can recover the source word from its buffer. Since the buffers have finite length, it is possible that the encoder is unable to determine the repetition time of a source word. In this case the source word is sent uncoded to the decoder.

Formal Description

In keeping with (5a), we set $B := 2^L - 1$. Now assume that $t = kL$, and hence $w_t = \boldsymbol{u}(k) = (u_{t-L}, u_{t-L+1}, \cdots, u_{t-1})$ is being encoded. Knowing this $\boldsymbol{u}(k)$ and the buffer content $\Phi_e(k) = (u_{t-L-B}, u_{t-L-B+1}, \cdots, u_{t-L-1})$, the encoder has access to all w_{t-m} with $m = 1,2,\cdots,B$. With these B vectors of length L, the encoder determines whether or not the value m_t of the repetition time of w_t exceeds B and, if this is not the case, the value of m_t. For the sets $\mathcal{M}_p$ defined as:

$$\mathcal{M}_p := \begin{cases} \{m: 2^p \leq m \leq 2^{p+1} - 1\}, & \text{for } p = 0,1,\cdots,L-1, \\ \{m: m \geq 2^L\}, & \text{for } p = L, \end{cases} \tag{8}$$

the encoder can assign to each m_t a set index p such that $m_t \in \mathcal{M}_p$. This set index p is transmitted to the decoder by means of a fixed-length prefix of $\lceil \log(L+1) \rceil$ binary digits. This prefix is the first part of the codeword $\boldsymbol{c}(k)$. The construction of the second part of $\boldsymbol{c}(k)$, the suffix, depends on the value of the set index p.

First assume that $p < L$. The objective of the encoder is to send the repetition time m_t to the decoder. To this end, it determines the member index q of m_t. More precisely, $q := m_t - 2^p$. Since $q \in \{0,1,\cdots,2^p - 1\}$, a suffix of p binary digits suffices to send this member index q to the decoder.

When the set index $p = L$, the source word w_t does not occur in the buffers $\Phi_e(k)$ and $\Phi_d(k)$, and instead of a member index, the encoder sends the entire source word $\boldsymbol{u}(k)$ to the decoder. This requires a suffix of L binary digits.

One easily verifies that the decoder, after having received the codeword $\boldsymbol{c}(k)$, can reconstruct $\boldsymbol{u}(k)$. Note that the codewords emitted by the encoder satisfy the prefix condition and consequently are uniquely decodable.

Example (see Table I): Let $L = 3$ and $t = 0$. Assume that the buffer contains

$$\Phi_e(0) = (u_{-10}, u_{-9}, \cdots, u_{-4}) = (0,1,0,0,1,0,0)$$

and that

$$(u_{-3}, u_{-2}, \cdots) = 100|000|011|111|011|101|001|\cdots$$

is the future string. We find the following repetition times: $3|1| \geq 8|1|6|4| \geq 8|\cdots$. Note that a repetition time m_t equal to one occurs if $(u_{t-L}, u_{t-L+1}, \cdots, u_{t-1}) = (u_{t-L-1}, u_{t-L}, \cdots, u_{t-2})$. These repetition times give rise to the following codewords:

$$|01,1|00|11,011|00|10,10|10,00|11,001|\cdots$$

(prefix and suffix are separated by a comma).

TABLE I
Encoding Table for $L = 3$

m	p	q	Prefix	Suffix	Length
1	0	0	0 0	—	2
2	1	0	0 1	0	3
3	1	1	0 1	1	3
4	2	0	1 0	0 0	4
5	2	1	1 0	0 1	4
6	2	2	1 0	1 0	4
7	2	3	1 0	1 1	4
≥ 8	3	—	1 1	$\boldsymbol{u}(k)$	5

V. Analysis

Again let $t = kL$ (hence $\boldsymbol{w}_t = \boldsymbol{u}(k)$ is being encoded). From the description of the algorithm we know that

$$Lg(\boldsymbol{c}(k)) = \begin{cases} \lceil \log(L+1) \rceil + \lfloor \log(m_t) \rfloor, & \text{for } m_t \le 2^L - 1 \\ \lceil \log(L+1) \rceil + L, & \text{for } m_t \ge 2^L \end{cases} \tag{9}$$

where $\lfloor a \rfloor$ denotes the largest integer $\le a$. Suppose that $\boldsymbol{w}$ is such that $P(\boldsymbol{W}_t = \boldsymbol{w}) > 0$. For the average length $L(\boldsymbol{w})$ of the codeword $\boldsymbol{c}(k)$ when $\boldsymbol{W}_t = \boldsymbol{w}$, it follows that

$$\begin{aligned} L(\boldsymbol{w}) &= \sum_{m=1,2^L-1} Q_m(\boldsymbol{w})(\lceil \log(L+1) \rceil + \lfloor \log(m) \rfloor) \\ &\quad + \sum_{m=2^L,\infty} Q_m(\boldsymbol{w})(\lceil \log(L+1) \rceil + L) \\ &\le \sum_{m=1,\infty} Q_m(\boldsymbol{w})(\lceil \log(L+1) \rceil + \log(m)) \\ &\overset{a}{=} \lceil \log(L+1) \rceil + \sum_{m=1,\infty} Q_m(\boldsymbol{w}) \log(m) \\ &\overset{b}{\le} \lceil \log(L+1) \rceil + \log\Big(\sum_{m=1,\infty} m Q_m(\boldsymbol{w}) \Big) \\ &= \lceil \log(L+1) \rceil + \log(T(\boldsymbol{w})) \\ &\overset{c}{\le} \lceil \log(L+1) \rceil - \log(P(\boldsymbol{W}_t = \boldsymbol{w})). \end{aligned} \tag{10}$$

Here equality a follows from part 1) of the lemma in Section III, b from the convexity of the $\log(\cdot)$ function and part 1) of the lemma, and c from part 2) of the lemma.

Using (10) we can now upper-bound the rate of our code in the following way:

$$\begin{aligned} RL &= \sum_{\boldsymbol{w}:\ P(\boldsymbol{W}_t = \boldsymbol{w}) > 0} P(\boldsymbol{W}_t = \boldsymbol{w}) L(\boldsymbol{w}) \\ &\le \sum_{\boldsymbol{w}:\ P(\boldsymbol{W}_t = \boldsymbol{w}) > 0} P(\boldsymbol{W}_t = \boldsymbol{w})(\lceil \log(L+1) \rceil \\ &\quad - \log(P(\boldsymbol{W}_t = \boldsymbol{w}))) \\ &= \lceil \log(L+1) \rceil + H(\boldsymbol{W}_t) \\ &= H(U_0, U_1, \cdots, U_{L-1}) + \lceil \log(L+1) \rceil \end{aligned} \tag{11}$$

where we have used the convention $0 \cdot \log(0) = 0$. This concludes the result mentioned in Section II.

VI. The Modified Algorithm

In this section we slightly modify our algorithm. We show that for a given buffer size $B = 2^\lambda - 1$ (λ is a positive integer), the length of the transmitted source words can be increased from $L = \lambda$ to $L = \lambda + \lceil \log(\lambda) \rceil$ without affecting the factor $\lceil \log(\lambda+1) \rceil / \lambda$ much. Again we make use of subsets $\mathcal{M}_p$ that are now defined as follows:

$$\mathcal{M}_p = \begin{cases} \{ m: 2^p \le m \le 2^{p+1} - 1 \}, & \\ \qquad\qquad \text{for } p = 0, 1, \cdots, \lambda - 1. \\ \{ m: m \ge 2^\lambda \}, & \text{for } p = \lambda \end{cases} \tag{12}$$

After having determined the set index p of m_t, this set index is transmitted to the decoder. To do this, instead of a fixed-length prefix as in Section IV, we apply a variable-length prefix (see Table II). To distinguish between the sets $\mathcal{M}_0, \mathcal{M}_1, \cdots, \mathcal{M}_{\lambda-1}$, the encoder sends a zero followed by $\lceil \log(\lambda) \rceil$ binary digits and, as before, the suffix is the member index. For the remaining set $\mathcal{M}_\lambda$, the prefix is one, while the suffix is the source word $\boldsymbol{u}(k)$, which now has length $L = \lambda + \lceil \log(\lambda) \rceil$.

TABLE II
Encoding Table for $\lambda = 4$ ($L = 6$)

m	p	q	Prefix	Suffix	Length
1	0	0	0 0 0	—	3
2	1	0	0 0 1	0	4
3	1	1	0 0 1	1	4
4	2	0	0 1 0	0 0	5
5	2	1	0 1 0	0 1	5
6	2	2	0 1 0	1 0	5
7	2	3	0 1 0	1 1	5
8	3	0	0 1 1	0 0 0	6
9	3	1	0 1 1	0 0 1	6
...	...	...	...	...	...
15	3	7	0 1 1	1 1 1	6
≥ 16	4	—	1	$\boldsymbol{u}(k)$	7

If we analyze this modified algorithm for a given value of λ we find that

$$L := \lambda + \lceil \log(\lambda) \rceil \tag{13a}$$

$$B := 2^\lambda - 1 \tag{13b}$$

$$R \le \big(H(U_0, U_1, \cdots, U_{L-1}) + \lceil \log(\lambda) \rceil + 1 \big) / L. \tag{13c}$$

Note that for practical values of $\lambda = \log(B+1)$, say $\lambda \le 24$, the term $\lceil \log(\lambda+1) \rceil / \lambda$ in expression (5b) is bigger than the corresponding term $(\lceil \log(\lambda) \rceil + 1)/(\lambda + \lceil \log(\lambda) \rceil)$ in (13c). In addition, the modified algorithm has the advantage that the rate R is always less than or equal to $(L+1)/L$. Nevertheless, for large values of λ, the term $\lceil \log(\lambda+1) \rceil / \lambda$ appears to be smaller than $(\lceil \log(\lambda) \rceil + 1)/(\lambda + \lceil \log(\lambda) \rceil)$.

VII. Implementation and Complexity

A simple implementation for the main algorithm is achieved by using shift registers (SR's) as buffers. In this case the storage and search complexities for both the encoder and the decoder are

$$C_{\text{enc}}^{\text{storage}} = C_{\text{dec}}^{\text{storage}} = 2^L - 1 \text{ binary locations}, \tag{14a}$$

$$C_{\text{enc}}^{\text{search}} = C_{\text{dec}}^{\text{search}} = 2^L - 1 \text{ shifts/source word}. \tag{14b}$$

An obvious disadvantage of this SR-implementation is the high search complexity. A low search complexity can be achieved if we use two random accessible memories (RAM's) at the encoder and one RAM at the decoder instead of SR's. These three RAM's all have L address lines. Both the encoder and the decoder use a (word-) RAM which at time t contains all the source words $\boldsymbol{w}_\tau$ for $t - B \le \tau \le t - 1$. This is accomplished by storing $\boldsymbol{w}_t$ at

address $t \bmod (2^L - 1)$ at time t. The encoder uses an additional (time-) RAM, which contains at address w the value $\tau \bmod (2^L - 1)$ of the most recent time τ for which $w_\tau = w$. Note that the two RAM's in the encoder are updated at the end of every time unit and not just at multiples of L.

Now suppose that at time $t = kL$ the word $w_t = u(k) = w$ is the output of the source. Then the encoder uses w to find in the time-RAM the (reduced) time $\tau_r = \tau \bmod (2^L - 1)$ such that τ is the most recent time for which $w_\tau = w$. With this τ_r it checks, using the word-RAM, whether or not the word w is stored at address τ_r. If so, it concludes that $m_t - 1 = t - \tau_r - 1 \bmod (2^L - 1)$. If not, $m_t \geq 2^L$. Now the two RAM's at the encoder are updated and the codeword $c(k)$ corresponding to m_t is sent to the decoder. If $m_t \leq 2^L - 1$, the decoder can reproduce m_t. With this m_t it determines τ_r and finally w_t by using its time-RAM. Note that for $1 \leq m_t \leq L - 1$, this last operation is slightly more complex than for $L \leq m_t \leq 2^L - 1$. If $m_t \geq 2^L$ the source word w_t is contained in the codeword. With w_t the decoder updates its word-RAM (L times). For the storage and search complexities we now find that

$$C_{\text{enc}}^{\text{storage}} = 2 \cdot C_{\text{dec}}^{\text{storage}} = 2 \cdot 2^L \text{ source words} \tag{15a}$$

$$C_{\text{enc}}^{\text{search}} = 2 \cdot C_{\text{dec}}^{\text{search}} = 2 \text{ mem. references/source word} \tag{15b}$$

We see that the search complexity is decreased enormously at the expense of an increase in storage complexity. The updating complexities are not considered here. It is clear that the modfied algorithm can be implemented in a similar way.

VIII. Perspectives

The basic idea behind the algorithms described here is "repetition time coding." The author discovered and analyzed this principle in January 1986. It was first presented in May 1986 at the Oberwolfach meeting on Information Theory. In a recent paper Elias [2] investigates a universal source coding algorithm that employs what he calls "interval coding." It turns out that interval coding and repetition time coding are the same idea. However, this common basic principle is applied in a different way in the present paper than in Elias's paper. The main difference between the two algorithms is that Elias's can only encode intervals. Therefore, all source words must occur somewhere in the "past." This is accomplished by proper initialization. In our algorithm repetition times are only encoded if they are small enough ($\leq 2^L - 1$). If not, the source word is sent to the decoder in an uncoded form. This has the advantage that the time at which a source word occurred can be stored modulo $(2^L - 1)$ and no reinitialization is required (note that for the modified algorithm, occurrence times of source words of length $L = \lambda + \lfloor \log(\lambda) \rfloor$ can even be stored modulo $2^\lambda - 1$). Another advantage of our bounded repetition time scheme is that the average codeword length of source word w can be upper-bounded by $-\log(P(W_t = w))$ plus some constant, instead of some other function $G(\cdot)$ of $\log(P(W_t = w))$. This results in a bound on the rate which appears very natural (see (5b)).

Section III of our paper deals with repetition times. The lemma given was proved in a slightly weaker form by Kac [3]. Since this lemma may be of importance to information theorists and is rather unknown, a proof of it is given in the Appendix.

In this paper a new fixed-to-variable-length universal source coding algorithm is proposed. Many authors in the past have treated universal source coding in ways that are in some respects different from our approach and in other ways similar. Early contributions in this field (e.g., the "enumerative" Lynch–Davisson–Schalkwijk algorithm) were put into the proper perspective by Davisson [4]. A few years later Gallager [5] and other researchers investigated the so-call "dynamic" Huffman code, which is an on-line adaptive version of the Huffman code. More sophisticated are the Ziv–Lempel [6], [7] algorithms in which the encoder references substrings that have occurred in the past. Referencing items in the past is also the basic idea in this paper and in Elias's paper [2]. A related idea, "recency rank coding," is investigated in a paper by Bentley *et al.* [8], and also by Elias [2]. Finally, we mention Rissanen's paper [9] which demonstrates the close connection between universal source coding, information, prediction, and estimation. Despite this close connection, the present paper shows that it is not necessary for a universal source encoder to contain an explicit modeler or estimator. It may be possible, however, that the performance (complexity, speed of approaching the entropy) of a universal algorithm can be improved by using a modeler and/or an estimator.

Acknowledgment

The author would like to acknowledge discussions with R. Ashlswede, J. Massey, L. Ozarow, T. Tjalkens, and A. Wyner concerning the lemma in Section 3 and the algorithm and also thank the reviewers for their valuable comments and suggestions.

Appendix
The Proof of the Lemma in Section III

Let

$$P_m(x) := P(X_0 = x, X_1 \neq x, \cdots, X_{m-1} \neq x, X_m = x). \tag{A1}$$

From

$$\begin{aligned} P(X_0 = x) &= P(X_0 = x, X_1 = x) + P(X_0 = x, X_1 \neq x) \\ &= P(X_0 = x, X_1 = x) + P(X_0 = x, X_1 \neq x, X_2 = x) \\ &\quad + P(X_0 = x, X_1 \neq x, X_2 \neq x) \\ &= \cdots = \sum_{m=1,N} P_m(x) + P(X_0 = x, X_1 \neq x, \cdots, X_N \neq x), \end{aligned} \tag{A2}$$

we find that

$$\begin{aligned} P(X_0=x) &- \sum_{m=1,\infty} P(X_{-m}=x, X_{1-m}\neq x,\cdots,X_{-1}\neq x, X_0=x) \\ &\overset{*}{=} P(X_0=x) - \sum_{m=1,\infty} P_m(x) \\ &= \lim_{N\to\infty} P(X_0=x, X_1\neq x,\cdots, X_N\neq x) \\ &= \lim_{N\to\infty} P(X_1\neq x,\cdots, X_N\neq x) \\ &- \lim_{N\to\infty} P(X_0\neq x,\cdots, X_N\neq x) \\ &\overset{*}{=} \lim_{N\to\infty} P(X_0\neq x,\cdots, X_{N-1}\neq x) \\ &- \lim_{N\to\infty} P(X_0\neq x,\cdots, X_N\neq x) = 0. \end{aligned} \tag{A3}$$

The equalities marked with an asterisk follow from the stationarity of the source. Dividing (A3) by $P(X_0=x)$ proves part 1) of the lemma. We proceed with

$$\begin{aligned} &P(X_0=x)\cdot T(x) \\ &\quad = \sum_{m=1,\infty} mP(X_{-m}=x, X_{1-m}\neq x,\cdots, X_{-1}\neq x, X_0=x) \\ &\quad \overset{*}{=} \sum_{m=1,\infty} mP_m(x). \end{aligned} \tag{A4}$$

Noting that

$$\begin{aligned} \sum_{m=1,\infty} mP_m(x) &= \sum_{m=1,N} mP_m(x) + \sum_{m=N+1,\infty} mP_m(x) \\ &\geq \sum_{m=1,N} mP_m(x) + (N+1)\cdot \sum_{m=N+1,\infty} P_m(x) \\ &\overset{1}{=} \sum_{m=1,N} mP_m(x) + (N+1) \\ &\quad \cdot\left(P(X_0=x) - \sum_{m=1,N} P_m(x)\right) \\ &\geq \sum_{m=1,N} mP_m(x). \end{aligned} \tag{A5}$$

(equality 1 follows from (A3)), we may conclude that

$$\begin{aligned} &P(X_0=x)\cdot T(x) \\ &\quad = \lim_{N\to\infty}\Bigg(\sum_{m=1,N} mP_m(x) + (N+1) \\ &\qquad \cdot P(X_0=x, X_1\neq x,\cdots, X_N\neq x)\Bigg). \end{aligned} \tag{A6}$$

Finally with (A6) and

$$\begin{aligned} &P(X_0=x, X_1=x) + 2P(X_0=x, X_1\neq x, X_2=x) \\ &\quad + 3P(X_0=x, X_1\neq x, X_2\neq x, X_3=x) + \cdots \\ &\quad + NP(X_0=x, X_1\neq x,\cdots, X_{N-1}\neq x, X_N=x) \\ &\quad + (N+1)P(X_0=x, X_1\neq x,\cdots, X_N\neq x) \\ &\overset{2}{=} P(X_0=x) + P(X_0=x, X_1\neq x) \\ &\quad + P(X_0=x, X_1\neq x, X_2\neq x) + \cdots \\ &\quad + P(X_0=x, X_1\neq x,\cdots, X_{N-1}\neq x) \\ &\quad + P(X_0=x, X_1\neq x,\cdots, X_N\neq x) \\ &= P(X_0=x) + P(X_0=x, X_1\neq x) \\ &\quad + P(X_0=x, X_1\neq x, X_2\neq x) + \cdots \\ &\quad + P(X_0=x, X_1\neq x,\cdots, X_{N-1}\neq x) \\ &\quad + P(X_1\neq x,\cdots, X_N\neq x) - P(X_0\neq x,\cdots, X_N\neq x) \\ &\overset{*}{=} P(X_0=x) + P(X_0=x, X_1\neq x) \\ &\quad + P(X_0=x, X_1\neq x, X_2\neq x) + \cdots \\ &\quad + P(X_0=x, X_1\neq x,\cdots, X_{N-1}\neq x) \\ &\quad + P(X_0\neq x,\cdots, X_{N-1}\neq x) - P(X_0\neq x,\cdots, X_N\neq x) \\ &= P(X_0=x) + P(X_0=x, X_1\neq x) \\ &\quad + P(X_0=x, X_1\neq x, X_2\neq x) + \cdots \\ &\quad + P(X_1\neq x,\cdots, X_{N-1}\neq x) - P(X_0\neq x,\cdots, X_N\neq x) \\ &= \cdots = 1 - P(X_0\neq x,\cdots, X_N\neq x), \end{aligned} \tag{A7}$$

we obtain part 2) of the lemma. Equality 2 follows from Massey's [10] "leaf-node theorem" and the asterisk marks equality from stationarity of the source.

References

[1] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[2] P. Elias, "Interval and recency rank source coding: Two on-line adaptive variable-length schemes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 3–10, Jan. 1987.

[3] M. Kac, "On the notion of recurrence in discrete stochastic processes," *Bull. Amer. Math. Soc.*, vol. 53, pp. 1002–1010, Oct. 1947.

[4] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 783–795, Nov. 1973.

[5] R. G. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 668–674, Nov. 1978.

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

[7] ——, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, Sept. 1978.

[8] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Commun. Ass. Comput. Mach.*, vol. 29, pp. 320–330, April 1986.

[9] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 629–636, July 1984.

[10] J. L. Massey, "The entropy of a rooted tree with probabilities," presented at the IEEE Int. Symp. Inform. Theory, St. Jovite, Canada, Sept. 26–30, 1983.