The context-tree weighting method: extensions

The Context-Tree Weighting Method: Extensions

Frans M. J. Willems, *Member, IEEE*

***Abstract*— First we modify the basic (binary) context-tree weighting method such that the past symbols $x_{1-D}, x_{2-D}, \cdots, x_0$ are not needed by the encoder and the decoder. Then we describe how to make the context-tree depth D infinite, which results in optimal redundancy behavior for all tree sources, while the number of records in the context tree is not larger than $2T-1$. Here T is the length of the source sequence. For this extended context-tree weighting algorithm we show that with probability one the compression ratio is not larger than the source entropy for source sequence length $T \to \infty$ for stationary and ergodic sources.**

***Index Terms*— Binary stationary and ergodic sources, cumulative redundancy bounds, modeling procedure, sequential data compression, tree sources, universal source coding.**

I. Introduction

The context-tree weighting method, first presented in [7], appears to be an efficient implementation for weighting (mixing) the coding distributions (universal over the parameters) corresponding to all tree models in class $\mathcal{C}_D$, i.e., the set of all tree models $\mathcal{S}$ whose maximum depth does not exceed D. A tree model is determined by a proper and complete set $\mathcal{S}$ of suffixes. Together these suffixes form a tree that is grown in negative t (i.e., time) direction. Each semi-infinite sequence $\cdots x_{t-3}x_{t-2}x_{t-1}$ has a unique suffix in $\mathcal{S}$, i.e., it passes through a unique leaf in the corresponding model tree. We restrict ourselves here to *binary* sources. Then this suffix (leaf) determines the probability that x_t, i.e., the next symbol generated by the (binary) source, is a 1.

The analysis of the context-tree weighting method turns out to be very straightforward (see [8]). It shows that the performance is as good as we can possibly hope, not only asymptotically but also for finite sequence lengths. Here we will propose two extensions to the basic context-tree weighting method and derive an interesting consequence of these extensions. We will use the notation of [8]. Codewords are assumed to be *binary*, logarithms have base 2, and information quantities are expressed in bits.

II. Coding Without Knowledge of Past Symbols x_{1-D}, $x_{2-D}, \cdots$, and x_0

A. A Simple Adaptation of the Basic Context-Tree Weighting Method

In its basic form (described in [8]), where we assumed that the actual tree model $\mathcal{S} \in \mathcal{C}_D$, the context-tree weighting method needs, for processing the symbol x_t, for $t = 1, 2, \cdots, T$, the context $x_{t-D}^{t-1} = x_{t-D}x_{t-D+1}\cdots x_{t-1}$ for this symbol. Here D is the depth of the context tree. This implies that for processing x_1 the encoder and decoder must have access to $x_{1-D}x_{2-D}\cdots x_0$ and for x_2 they need $x_{2-D}x_{3-D}\cdots x_1$, etc. This is an unpleasant fact. The past symbols $x_{1-D}, x_{2-D}, \cdots$, and x_0 may not be available at all to the encoder and the decoder. A straightforward way to circumvent this problem is to start processing only after a full context is available to both, i.e., to start processing with x_{D+1}. The encoder then sends the first context $x_1x_2\cdots x_D$ to the decoder in an uncoded way. This requires D binary-code digits.

To study coding methods that do not assume availability of the past symbols $x_{1-D}, x_{2-D}, \cdots, x_0$ we first consider the case where the encoder and the decoder do know the tree model $\mathcal{S}$ of the source.

B. Known Model

In the situation where the encoder and the decoder already know the tree model $\mathcal{S}$, the source symbol x_t can be transmitted in an uncoded way, if its *available* context $x_1x_2\cdots x_{t-1}$ does not have a suffix[1] in $\mathcal{S}$. If, on the other hand, the available context of a symbol does have a suffix in $\mathcal{S}$ we use the Krichevsky–Trofimov estimator [2] that corresponds to this suffix. This results in the coding distribution

$$P_c'(x_1^t|\mathcal{S}) \triangleq 2^{-\Delta_{\mathcal{S}}(x_1^t)} \prod_{s\in\mathcal{S}} P_e(a_s'(x_1^t), b_s'(x_1^t)), \quad \text{for all } x_1^t \in \{0,1\}^t, t = 0, 1, \cdots, T. \tag{1}$$

Here $a_s'(x_1^t)$ denotes the number of zeros that occurred in x_1^t at instants τ with $\tau = 1, t$ such that s is a suffix of $x_1^{\tau-1}$, and $b_s'(x_1^t)$ denotes the number of ones in x_1^t at instants τ, $\tau = 1, t$, for which s is a suffix of $x_1^{\tau-1}$, for $s \in \mathcal{T}_D$. Furthermore, the estimator $P_e(\cdot,\cdot)$ is defined as

$$P_e(a,b) \triangleq \frac{\frac{1}{2}\cdot\frac{3}{2}\cdot\cdots\cdot(a-\frac{1}{2})\cdot\frac{1}{2}\cdot\frac{3}{2}\cdot\cdots\cdot(b-\frac{1}{2})}{1\cdot 2\cdot\cdots\cdot(a+b)}, \quad \text{for } a>0 \text{ and } b>0, \text{ etc.} \tag{2}$$

For the number of uncoded symbols $\Delta_{\mathcal{S}}(x_1^t)$, i.e., the number of symbols x_τ in x_1^t for which the available context $x_1^{\tau-1}$ does not have a suffix in $\mathcal{S}$, we can write

$$\Delta_{\mathcal{S}}(x_1^t) = t - \sum_{s\in\mathcal{S}} (a_s'(x_1^t) + b_s'(x_1^t)). \tag{3}$$

Note that (1) is a *sequentially available* coding distribution. Observe that the number of uncoded symbols $\Delta_{\mathcal{S}}(x_1^T)$ in the source sequence x_1^T depends on both the suffix set $\mathcal{S}$ and the source sequence x_1^T. We always have that

$$\Delta_{\mathcal{S}}(x_1^T) \le \Delta_{\mathcal{S}}^{\max} \triangleq \max_{s\in\mathcal{S}} l(s)$$

where $l(s)$ is the length of s. The number of uncoded symbols is also called the number of missing contexts.

Let

$$\gamma(z) \triangleq \begin{cases} z, & \text{for } 0 \le z < 1 \\ \frac{1}{2}\log z + 1, & \text{for } z \ge 1 \end{cases} \tag{4}$$

thus $\gamma(\cdot)$ is the "smallest" convex-$\cap$ continuation of $\frac{1}{2}\log z + 1$ for $0 \le z < 1$ satisfying $\gamma(0) = 0$. For the individual (cumulative) redundancy resulting from this coding distribution for the source sequence x_1^T with respect to the tree source with model $\mathcal{S}$ and parameter vector $\Theta_{\mathcal{S}}$, we now obtain for all past source sequences

Manuscript received August 7, 1995; revised September 1, 1997. The material in this correspondence was presented at the IEEE International Symposium on Information Theory, Trondheim, Norway, June 27–July 1, 1994.

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

Publisher Item Identifier S 0018-9448(98)00654-3.

[1] Also if $x_1x_2\cdots x_{t-1} \in \mathcal{S}$ we say that $x_1x_2\cdots x_{t-1}$ has a suffix in $\mathcal{S}$.

x_{1-D}^{0}

$$\begin{aligned}\log \frac{P_a(x_1^T|x_{1-D}^0,\mathcal{S},\Theta_{\mathcal{S}})}{P_c'(x_1^T|\mathcal{S})} &= \log \frac{\prod_{s\in\mathcal{S}}(1-\theta_s)^{a_s}\theta_s^{b_s}}{2^{-\Delta_{\mathcal{S}}(x_1^T)}\prod_{s\in\mathcal{S}}P_e(a_s',b_s')} \\ &\le \log \frac{\prod_{s\in\mathcal{S}}(1-\theta_s)^{a_s'}\theta_s^{b_s'}}{2^{-\Delta_{\mathcal{S}}(x_1^T)}\prod_{s\in\mathcal{S}}P_e(a_s',b_s')} \\ &= \sum_{s\in\mathcal{S}}\log\frac{(1-\theta_s)^{a_s'}\theta_s^{b_s'}}{P_e(a_s',b_s')} + \Delta_{\mathcal{S}}(x_1^T) \\ &\le |\mathcal{S}|\gamma\left(\frac{T-\Delta_{\mathcal{S}}(x_1^T)}{|\mathcal{S}|}\right) + \Delta_{\mathcal{S}}(x_1^T). \end{aligned} \tag{5}$$

The first inequality follows from the fact that $a_s \ge a_s' = a_s'(x_1^T)$ and $b_s \ge b_s' = b_s'(x_1^T)$, where $a_s = a_s(x_1^T|x_{1-D}^0)$, respectively $b_s = b_s(x_1^T|x_{1-D}^0)$, is the number of times that $x_\tau = 0$, respectively $x_\tau = 1$, in x_1^T for $1 \le \tau \le T$ such that $x_{\tau-l(s)}^{\tau-1} = s$ for $s \in \mathcal{T}_D$. The second inequality is obtained as in [8, the Proof of Theorem 2, eq. (23)], where the parameter redundancy is bounded. Observe, however, that here

$$\sum_{s\in\mathcal{S}}(a_s' + b_s') = T - \Delta_{\mathcal{S}}(x_1^T).$$

In the case of a known tree model, if the past symbols $x_{1-D}, x_{2-D}, \cdots$, and x_0 are available to the encoder and the decoder, the (parameter) redundancy can be upper-bounded (see [8, eq. (23)]) by $|\mathcal{S}|\gamma(T/|\mathcal{S}|)$. Therefore, regarding these upper bounds, we may conclude that not knowing the past symbols costs at most $\Delta_{\mathcal{S}}(x_1^T)$ bits, if the encoder and the decoder do know the model $\mathcal{S}$. We say that the *starting redundancy* is upper-bounded by $\Delta_{\mathcal{S}}(x_1^T)$. Note that $\Delta_{\mathcal{S}}(x_1^T) \le \Delta_{\mathcal{S}}^{\max} \le D$ since we assume that $\mathcal{S} \in \mathcal{C}_D$. Therefore, this method never performs worse than the (straightforward) method that was described in the previous subsection.

C. Unknown Model

Again suppose that the model $\mathcal{S} \in \mathcal{C}_D$. In this subsection we will show that, also in the case where the encoder and the decoder do not know the model $\mathcal{S}$, we loose not more than $\Delta_{\mathcal{S}}(x_1^T)$ bits, if they do not have access to the past symbols $x_{1-D}, x_{2-D}, \cdots$, and x_0. In other words, the starting redundancy is again not more than $\Delta_{\mathcal{S}}(x_1^T)$. We demonstrate this by modifying the basic context-tree weighting method.

The starting point of this modification is that the encoder and the decoder assign to all the unknown past symbols $x_{D-1}, x_{D-2}, \cdots$, and x_0, the value ε. The value ε is the *indeterminate symbol* value. Because of the alphabet extension we must change the binary context tree $\mathcal{T}_D$ into a *ternary* context tree $\tilde{\mathcal{T}}_D$. A node in this tree corresponds to a string of symbols from the alphabet $\{0, \varepsilon, 1\}$. To each node s in the ternary context tree, there corresponds a count $\tilde{a}_s(x_1^t|\varepsilon^D)$ that denotes the number of zeros that occur in x_1^t at instants $\tau, 1 \le \tau \le t$, such that s is a suffix of $\varepsilon^D x_1^{\tau-1}$, and a count $\tilde{b}_s(x_1^t|\varepsilon^D)$ that stands for the number of ones in x_1^t at instants $\tau, 1 \le \tau \le t$, where s is a suffix of $\varepsilon^D x_1^{\tau-1}$, for all $s \in \tilde{\mathcal{T}}_D$. The definition of the weighted distribution is now slightly different from the basic one given in [8].

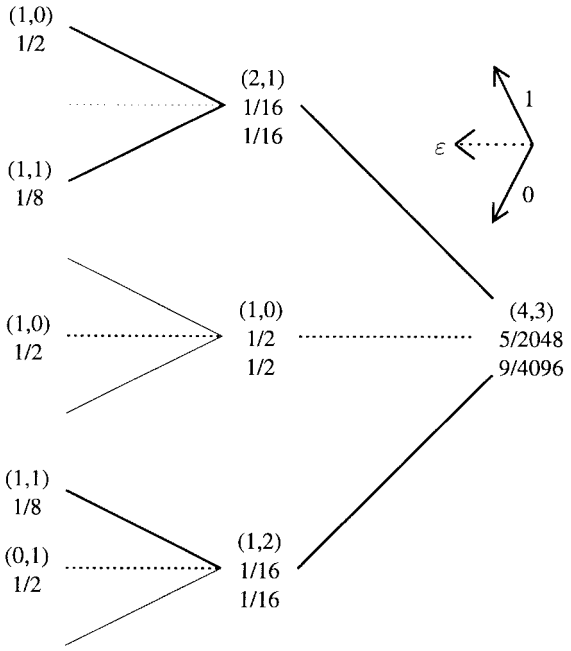

Fig. 1. Context tree $\tilde{\mathcal{T}}_2$ for $x_1^T = 0110100$.

Definition 1: To each node $s \in \tilde{\mathcal{T}}_D$ we assign a weighted probability which is defined as

$$\tilde{P}_w^s \triangleq \begin{cases} \frac{1}{2}P_e(\tilde{a}_s,\tilde{b}_s) + \frac{1}{2}\tilde{P}_w^{0s}\tilde{P}_w^{\varepsilon s}\tilde{P}_w^{1s}, & \text{for } 0 \le l(s) < D \\ P_e(\tilde{a}_s,\tilde{b}_s), & \text{for } l(s) = D. \end{cases} \tag{6}$$

The corresponding coding distribution is defined as

$$\tilde{P}_c(x_1^t) \triangleq \tilde{P}_w^{\lambda}(x_1^t|\varepsilon^D), \qquad \text{for all } x_1^t \in \{0,1\}^t,\ t = 0,1,\cdots,T. \tag{7}$$

Just like before, we can prove that this coding distribution satisfies [8, eq. (6)], i.e., that it is a probability distribution. Moreover it is sequentially updatable. And again the computational and storage complexity needed to update this distribution is not larger than linear in T. Before we continue with deriving an upper bound on the redundancy of this modified context-tree weighting method, we give an example.

Example 1: Suppose that a binary source generated the sequence $x_1^T = 0110100$. For $D = 2$ we have plotted the context tree $\tilde{\mathcal{T}}_D$ in Fig. 1. Node s contains the counts $(\tilde{a}_s(x_1^T|\varepsilon^D), \tilde{b}_s(x_1^T|\varepsilon^D))$, the Krichevsky–Trofimov estimate $P_e(\tilde{a}_s, \tilde{b}_s)$, and the weighted probability $\tilde{P}_w^s(x_1^T|\varepsilon^D)$. Nodes at depth $D = 2$ are listed only with their weighted probability $\tilde{P}_w^s(x_1^T|\varepsilon^D)$ which is equal to the Krichevsky–Trofimov estimate $P_e(\tilde{a}_s, \tilde{b}_s)$ there. The coding probability $\tilde{P}_c(x_1^T) = \tilde{P}_w^{\lambda}(x_1^T|\varepsilon^D)$ for the sequence 0110100 turns out to be $9/4096$.

Note that for a context path to have nonzero counts it is necessary to contain no ε's at all, or to have the structure $\varepsilon^d x_1 x_2 \cdots x_{D-d}$ for $d = 1, D$. In the latter case, the sum of the counts cannot exceed 1. There are only D such paths. In the figure, we see a path $\varepsilon\varepsilon$ (leaving the root node), and a path $\varepsilon 0$.

We are now ready to state a theorem that upper-bounds the redundancy of this new weighting method. First we repeat [8, Definition 2]. The cost $\Gamma_D(\mathcal{S})$ of a model $\mathcal{S}$ with respect to class $\mathcal{C}_D$ is defined as

$$\Gamma_D(\mathcal{S}) \triangleq |\mathcal{S}| - 1 + |\{s\colon s \in \mathcal{S}, l(s) \ne D\}| \tag{8}$$

where it is assumed that $\mathcal{S} \in \mathcal{C}_D$.

Theorem 1: For any tree source with unknown model $\mathcal{S} \in \mathcal{C}_D$ and unknown parameter vector $\Theta_{\mathcal{S}}$, the individual redundancies with respect to $(\mathcal{S}, \Theta_{\mathcal{S}})$ are upper-bounded by

$$\begin{aligned}\rho(x_1^T|x_{1-D}^0, \mathcal{S}, \theta_{\mathcal{S}}) &\triangleq L(x_1^T) - \log \frac{1}{P_a(x_1^T|x_{1-D}^0, \mathcal{S}, \Theta_{\mathcal{S}})} \\ &< \Gamma_D(\mathcal{S}) + |\mathcal{S}|\gamma\left(\frac{T - \Delta_{\mathcal{S}}(x_1^T)}{|\mathcal{S}|}\right) + \Delta_{\mathcal{S}}(x_1^T) + 2 \qquad (9)\end{aligned}$$

for all $x_1^T \in \{0,1\}^T$, for all sequences of past symbols x_{1-D}^0, if we use the coding distribution specified in (7).

Proof: Consider a sequence $x_1^T \in \{0,1\}^T$. As in [8, Proof of Theorem 2], we split the individual redundancy into three terms

$$\begin{aligned}\rho(x_1^T|x_{1-D}^0, \mathcal{S}, \Theta_{\mathcal{S}}) &= L(x_1^T) - \log \frac{1}{P_a(x_1^T|x_{1-D}^0, \mathcal{S}, \Theta_{\mathcal{S}})} \\ &= \log \frac{P_c'(x_1^T|\mathcal{S})}{\tilde{P}_c(x_1^T)} + \log \frac{P_a(x_1^T|x_{1-D}^0, \mathcal{S}, \Theta_{\mathcal{S}})}{P_c'(x_1^T|\mathcal{S})} \\ &\quad + \left(L(x_1^T) - \log \frac{1}{\tilde{P}_c(x_1^T)}\right). \qquad (10)\end{aligned}$$

The last term in (10), the coding redundancy term, is upper-bounded by 2. For the middle term, the parameter plus starting redundancy term, we use the bound given by (5)

$$\log \frac{P_a(x_1^T|x_{1-D}^0, \mathcal{S}, \Theta_{\mathcal{S}})}{P_c'(x_1^T|\mathcal{S})} \le |\mathcal{S}|\gamma\left(\frac{T - \Delta_{\mathcal{S}}(x_1^T)}{|\mathcal{S}|}\right) + \Delta_{\mathcal{S}}(x_1^T). \qquad (11)$$

What remains to be investigated is the first term, the model redundancy term. We can lower-bound $\tilde{P}_c(x_1^T)$ in terms similar to the terms that form $P_c'(x_1^T|\mathcal{S})$ in (1), as we shall soon see.

Consider the tree model $\mathcal{S}$ which we know to be contained in the context tree $\mathcal{T}_D$. *Leafs* of $\mathcal{S}$ are nodes s in the context tree $\tilde{\mathcal{T}}_D$ for which $s \in \mathcal{S}$ and *internal* nodes s' of $\mathcal{S}$ are nodes in $\tilde{\mathcal{T}}_D$ that are a suffix of some string $s \in \mathcal{S}, s \neq s'$.

First observe that for nodes $s \in \mathcal{S}$ we have that

$$\tilde{a}_s(x_1^T|\varepsilon^D) = a_s'(x_1^T),$$

and

$$\tilde{b}_s(x_1^T|\varepsilon^D) = b_s'(x_1^T). \qquad (12)$$

Next note that for $s \in \tilde{\mathcal{T}}_D$

$$\tilde{P}_w^s(x_1^T|\varepsilon^D) \ge \begin{cases} \frac{1}{2}P_e(a_s'(x_1^T), b_s'(x_1^T)), & \text{if } s \text{ is a leaf of } \mathcal{S} \\ & \text{with } l(s) \neq D \\ P_e(a_s'(x_1^T), b_s'(x_1^T)), & \text{if } s \text{ is a leaf of } \mathcal{S} \\ & \text{with } l(s) = D \\ \frac{1}{2}\tilde{P}_w^{0s}(x_1^T)\tilde{P}_w^{1s}(x_1^T), & \text{if } s \text{ is an internal node} \\ & \text{of } \mathcal{S}, \text{ and } x_1^{l(s)} \neq s \\ \frac{1}{4}\tilde{P}_w^{0s}(x_1^T)\tilde{P}_w^{1s}(x_1^T), & \text{if } s \text{ is an internal node} \\ & \text{of } \mathcal{S}, \text{ and } x_1^{l(s)} = s. \end{cases} \qquad (13)$$

Combining these inequalities, starting in the leaves of $\mathcal{S}$ and working towards the root of the context tree we see that we loose 1 bit in each internal node s of $\mathcal{S}$ and leaf $s \in \mathcal{S}$ not at depth D, which adds up to $\Gamma_D(\mathcal{S})$ in total. In addition to this we get an increase of 1 bit in internal nodes s of $\mathcal{S}$ that occur as a prefix of the source sequence x_1^T. Such a one-bit increase corresponds to a *missing context* or, in other words, to an uncoded symbol. These additional costs add up to $\Delta_{\mathcal{S}}(x_1^T)$.

We get as lower bound for the coding probability

$$\begin{aligned}\tilde{P}_c(x_1^T) &= \tilde{P}_w^{\lambda}(x_1^T|\varepsilon^D) \ge 2^{-\Gamma_D(\mathcal{S}) - \Delta_{\mathcal{S}}(x_1^T)} \\ &\quad \cdot \prod_{s \in \mathcal{S}} P_e(a_s'(x_1^T), b_s'(x_1^T)). \qquad (14)\end{aligned}$$

Combining this with (1), as in [8, eq. (25)], this results in the following upper bound for the model redundancy:

$$\log \frac{P_c'(x_1^T|\mathcal{S})}{\tilde{P}_c(x_1^T)} \le \Gamma_D(\mathcal{S}). \qquad (15)$$

Substitution of (11), (15), and the 2 bits for the coding redundancy in (10) finally leads to the theorem. □

Theorem 1 states that also in the case where the past symbols are not available to the encoder and the decoder, the loss of not knowing the model $\mathcal{S}$ is bounded by $\Gamma_D(\mathcal{S})$ bits. This is completely identical to the basic context-tree weighting result. In both the known and the unknown model situation, the loss of not having access to the past symbols $x_{1-D}, x_{2-D}, \cdots$, and x_0, i.e., the starting redundancy, is never more than $\Delta_{\mathcal{S}}(x_1^T)$.

Although the context tree $\tilde{\mathcal{T}}_D$ is, in principle, ternary, it is possible to show that $\tilde{P}_c(x_1^T)$ is a weighting over coding distributions $P_c'(x_1^T|\mathcal{S})$ for all binary-tree models $\mathcal{S} \in \mathcal{C}_D$. This is shown in the Appendix.

Although the coding distribution (6) suggests the use of a ternary context tree, this is not necessary at all. Since only binary contexts and contexts of the form $\varepsilon^d x_1^{D-d}$ for $d = 1, D$ can actually occur, it suffices to implement a binary context tree in which each node contains a Boolean variable that indicates whether or not this node has a *tail*. A node $s \in \mathcal{T}_D$ has a tail, if $x_1^{l(s)} = s$, and $l(s) < D$. The tail corresponds to the missing part of the context. If a node s has a tail, the product of the weighted probabilities of the children 0's and 1's of this node should be multiplied by $1/2$.

III. Infinite-Depth Context-Tree Weighting

In the previous section, we have dealt with a first unpleasant property of the basic context-tree weighting method, the fact that the past symbols were needed by the encoder and the decoder. A second shortcoming of the basic method is that the depth D of the context tree $\mathcal{T}_D$ is assumed to be finite. Only for models that fit into this finite-depth context tree, the weighting method achieves desirable redundancy bounds. The second result in this correspondence concerns a generalization of the basic context-tree weighting method to the situation where the context-tree depth is not bounded. In this case, we can still achieve a storage complexity (number of stored records) which does not increase faster than linear in the sequence length T.

The first observation that leads to this result is that after having processed the source sequence x_1^t, we have seen t semi-infinite contexts $\cdots\varepsilon\varepsilon x_1 x_2 \cdots x_{\tau-1}$, one for each $\tau = 1, t$ if we assume an infinite-depth context tree. It is important to note that all these contexts differ from each other.

In Fig. 2, we have depicted all these contexts up to the last ε-edge for $x_1^t = 110100$. We can observe the contexts $\cdots\varepsilon$ (for x_1), $\cdots\varepsilon 1, \cdots\varepsilon 11, \cdots\varepsilon 110, \cdots\varepsilon 1101$, and $\cdots\varepsilon 11010$ (for x_6). Note that as in the previous section we assume that $\cdots x_{-1}x_0 = \cdots\varepsilon\varepsilon$. Instead of labeling the edges that connect the nodes with values from the alphabet $\{0, \varepsilon, 1\}$ we label them with the time-index of the symbol in the last context that went through this edge. For example, the last context $\cdots\varepsilon 11010$ was formed by symbols $\cdots x_0 x_1 x_2 x_3 x_4 x_5$ and therefore the indices $\cdots 012345$ are found along the path representing this last context. The context corresponding to symbol x_4 was $\cdots\varepsilon 110$. Therefore, after having processed symbol x_4, the edges corresponding to context $\cdots\varepsilon 110$ were labeled with $\cdots 0123$. While processing the symbol x_6 with context $\varepsilon 11010$, the last two labels on this path were updated again and changed from 23 into 45. Note that a straightforward

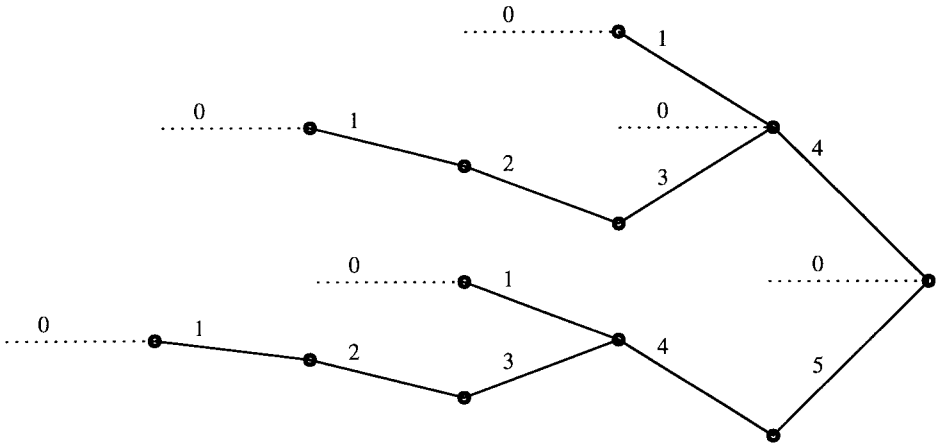

Fig. 2. Infinite complete context tree $\bar{\mathcal{T}}_\infty$ for $x_1^t = 110100$.

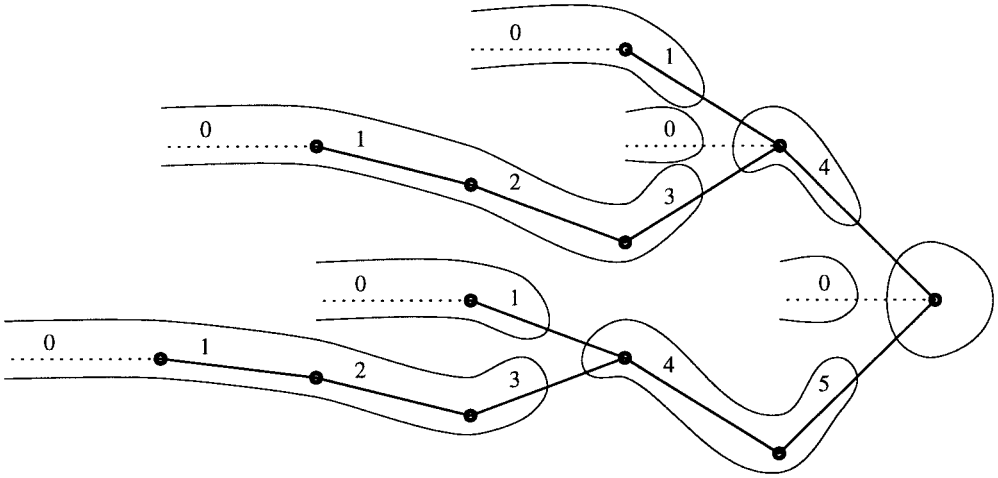

Fig. 3. Infinite context tree $\bar{\mathcal{T}}_\infty$ for $x_1^t = 110100$ with sets of equivalent nodes.

implementation of the structure that we have just described would yield a number of nodes that grows quadratically in T.

However, a second observation is that all *unique* nodes that correspond to the same context are *equivalent* and can be replaced by a single record (see Fig. 3). A node s is said to be unique if s occurs only *once* as a context in $\cdots \varepsilon\varepsilon x_1 x_2 \cdots x_t$. For example, for source sequence 110100 the nodes $\cdots, \varepsilon 11010, 11010, 1010, 010,$ are all unique and correspond to the same context $(\cdots \varepsilon 11010)$. They all can be replaced by a single record (we call this a leaf-record). We can label this record with the node closest to the root among the equivalent nodes. Furthermore, this record should contain a pointer to the position in the source sequence where the segment corresponding to the equivalent nodes occurs. This segment is formed by the unique edges in the context. The pointer contains the index of the most recent (closest to the root) edge in the segment. This pointer is (may be) needed for later updates. The implication of all this is that also the source sequence $x_1 x_2 \cdots x_t$ must be stored. For example, 010 was the unique node closest to the root resulting from context $\cdots \varepsilon 11010$. The corresponding segment with the unique edges is $\cdots \varepsilon 110 = \cdots x_0 x_1 x_2 x_3$. The first edge in this segment is x_3. The pointer, therefore, should have value 3. See Fig. 3.

The third observation is that also nonunique nodes that share the same set of contexts are equivalent and can be replaced by a single record (called internal, see Fig. 3). For example, the nodes 0 and 10 both share the context $\cdots \varepsilon 110$ (for x_4) and the context $\cdots \varepsilon 11010$ (for x_6) and are, therefore, equivalent and can be replaced by one record. Again we label this record with the node closest to the root (in Fig. 3 this is node 0) and again this record should contain a pointer to the most recent edge in the corresponding source segment. This segment is formed by the edges through which only all contexts in the mentioned set pass. In the figure, the segment corresponding to contexts $\cdots \varepsilon 110$ and $\cdots \varepsilon 11010$ is formed by the edges 1 and 0 labeled with x_4 and x_5. The most recent edge in the segment is x_5. The pointer is therefore 5. Also the length of the corresponding segment should be stored now (two edges, x_4 and x_5 in the figure, the length is therefore 2).

In addition to the pointer to the most recent edge (symbol) of the corresponding source segment, the length of that segment, and the $\tilde{a}$ and $\tilde{b}$ counts, an internal record contains pointers to its 0-, ε-, and 1-child records. Two of these pointers are non-`nil` so an internal node has two or three children. Updating the tree with a new context always creates a new leaf record. Therefore, after having processed the entire sequence x_1^T, the total number of produced leaf records will be T while the number of internal records is at most $T-1$. This results in a storage complexity ($<2T-1$ records) which grows not faster than *linear* in the source sequence length T. Note that also the sequence itself should be stored but this is also linear in T. It should be mentioned here that the described implementation of the context tree strongly relates to the DAWG concept proposed by Blumer *et al.* [1].

Apart from maintaining the context-tree structure, the counts $\tilde{a}_s$ and $\tilde{b}_s$ (note that now $s \in \tilde{\mathcal{T}}_\infty$), the estimated probabilies $P_e(\tilde{a}_s, \tilde{b}_s)$, and the weighted probabilities $\tilde{P}_{w,\infty}$ should be updated. In accordance to (6), taking $D = \infty$, we can now define the weighted distribution and the resulting coding distribution.

Definition 2: To each node $s \in \tilde{\mathcal{T}}_\infty$, we assign a weighted probability which is defined as

$$\tilde{P}^s_{w,\infty} \triangleq \begin{cases} \frac{1}{2}P_e(\tilde{a}_s, \tilde{b}_s) \\ \quad + \frac{1}{2}\tilde{P}^{0s}_{w,\infty}\tilde{P}^{\varepsilon s}_{w,\infty}\tilde{P}^{1s}_{w,\infty}, & \text{for nonunique } s \\ \frac{1}{2}, & \text{for unique } s. \end{cases} \tag{16}$$

The corresponding coding distribution is defined as

$$\tilde{P}_c(x_1^t) \triangleq \tilde{P}^\lambda_{w,\infty}(x_1^t|\varepsilon^\infty), \qquad \text{for all } s_1^t \in \{0,1\}^t,\ t = 0, 1, \cdots, T. \tag{17}$$

First note that equivalent nodes all have the same counts and therefore the same estimated probability. Note also that since the estimated probabilities of the equivalent nodes in a leaf record would all be equal to $1/2$, the weighted probability of all these nodes is also $1/2$. In an internal record the situation is slightly more complicated. Although the estimated probabilities are equal for all equivalent nodes, this does not hold for the weighted probabilities of these nodes. They are, however, easy to calculate and only the weighted probability corresponding to the node closest to the root is actually needed (by its parent). For example, for the equivalent nodes 10 and 0 in Fig. 3, the weighted probabilities are

$$\tilde{P}^{10}_{w,\infty} = \tfrac{1}{2}P_e + \tfrac{1}{2}\tilde{P}^{010}_{w,\infty}\tilde{P}^{110}_{w,\infty}$$

and

$$\tilde{P}^{0}_{w,\infty} = \tfrac{1}{2}P_e + \tfrac{1}{2}\tilde{P}^{10}_{w,\infty} = \tfrac{3}{4}P_e + \tfrac{1}{4}\tilde{P}^{010}_{w,\infty}\tilde{P}^{110}_{w,\infty}$$

respectively. Here

$$P_e = P_e(\tilde{a}_{10}, \tilde{b}_{10}) = P_e(\tilde{a}_0, \tilde{b}_0).$$

Only $\tilde{P}^0_{w,\infty}$ is really needed (by the root record labeled λ).

The individual redundancy of this infinite-depth context-tree weighting method is as expected. The only thing that changes is the cost of a model which is $2|\mathcal{S}| - 1$ bits for all $\mathcal{S}$ now.

Theorem 2: For any tree source with unknown model $\mathcal{S}$ and unknown parameter vector $\Theta_\mathcal{S}$, the individual redundancies with respect to $(\mathcal{S}, \Theta_\mathcal{S})$ are upper-bounded by

$$\begin{aligned} \rho(x_1^T|x^0_{-\infty}, \mathcal{S}, \Theta_\mathcal{S}) &\triangleq L(x_1^T) - \log \frac{1}{P_a(x_1^T|x^0_{-\infty}, \mathcal{S}, \Theta_\mathcal{S})} \\ &< 2|\mathcal{S}| - 1 + |\mathcal{S}|\gamma\left(\frac{T - \Delta_\mathcal{S}(x_1^T)}{|\mathcal{S}|}\right) \\ &\quad + \Delta_\mathcal{S}(x_1^T) + 2 \end{aligned} \tag{18}$$

for all $x_1^T \in \{0,1\}^T$, for all sequences $x^0_{-\infty} = \cdots x_{-1}x_0$ of past symbols, if we use coding distribution (17).

IV. Achieving Entropy for Arbitrary Stationary Ergodic Sources

Now that we can use the context-tree weighting algorithm for arbitrary-depth tree sources and without having access to the past symbols $\cdots x_{-1}x_0$, we can show that this method achieves entropy for arbitrary binary stationary and ergodic sources.

Theorem 3: For any binary stationary and ergodic source

$$\limsup_{t\to\infty} \frac{L(x_1^T)}{T} \le H_\infty(X) \text{ with probability one} \tag{19}$$

if we use the coding method and distribution (17) presented in the previous section.

Proof: We start this proof with the statement

$$\begin{aligned} \tilde{P}_{c,\infty}(x_1^T) = \tilde{P}^\lambda_{w,\infty}(x_1^T|\varepsilon^\infty) &\ge 2^{1-2|\mathcal{S}|-\Delta_\mathcal{S}(x_1^T)} \\ &\quad \cdot \prod_{s\in\mathcal{S}} P_e(a'_s(x_1^T), b'_s(x_1^T)) \end{aligned} \tag{20}$$

which holds for all tree models $\mathcal{S}$. This inequality is identical to the statement (14), however, for the infinite-depth context-tree weighting method the cost of a model $\mathcal{S}$ is $2|\mathcal{S}| - 1$ instead of $\Gamma_D(\mathcal{S})$. If we combine this with the arithmetic coding result (see [8, Theorem 1]) and use the fact that

$$a'_s(x_1^T) \le a_s(x_1^T|x^0_{-\infty})$$

and

$$b'_s(x_1^T) \le b_s(x_1^T|x^0_{-\infty})$$

we obtain lower bounds on the codeword length, one for each tree model $\mathcal{S}$.

$$\begin{aligned} L(x_1^T) &\le \log \frac{1}{\tilde{P}_{c,\infty}(x_1^T)} + 2 \\ &\le \log \frac{1}{\prod_{s\in\mathcal{S}} P_e(a'_s(x_1^T), b'_s(x_1^T))} + 2|\mathcal{S}| - 1 + \Delta_\mathcal{S}(x_1^T) + 2 \\ &\le \log \frac{1}{\prod_{s\in\mathcal{S}} P_e(a_s(x_1^T|x^0_{-\infty}), b_s(x_1^T|x^0_{-\infty}))} \\ &\quad + 2|\mathcal{S}| - 1 + \Delta_\mathcal{S}(x_1^T) + 2. \end{aligned} \tag{21}$$

For the terms $P_e(\cdot,\cdot)$ we now apply the lower bound [8, eq. (10)] which states (for $a + b \ge 1$) that

$$P_e(a,b) \ge \frac{1}{2}\cdot\frac{1}{\sqrt{a+b}}\left(\frac{a}{a+b}\right)^a\left(\frac{b}{a+b}\right)^b. \tag{22}$$

For $a = b = 0$ we have that $P_e(a,b) = 1$. Denoting $a_s(x_1^T|x^0_{-\infty})$ by a_s and $b_s(x_1^T|x^0_{-\infty})$ by b_s, this leads to the upper bound

$$\begin{aligned} \log \frac{1}{\prod_{s\in\mathcal{S}} P_e(a_s, b_s)} &\le \sum_{s\in\mathcal{S}:a_s+b_s>0}\Bigg(a_s\log\frac{a_s+b_s}{a_s} + b_s\log\frac{a_s+b_s}{b_s} \\ &\qquad + \frac{1}{2}\log(a_s+b_s) + 1\Bigg) \\ &= \sum_{s\in\mathcal{S}}\left(a_s\log\frac{a_s+b_s}{a_s} + b_s\log\frac{a_s+b_s}{b_s}\right) \\ &\quad + \sum_{s\in\mathcal{S}:a_s+b_s>0}\left(\frac{1}{2}\log(a_s+b_s)+1\right) \\ &\le \sum_{s\in\mathcal{S}}\left(a_s\log\frac{a_s+b_s}{a_s} + b_s\log\frac{a_s+b_s}{b_s}\right) + |\mathcal{S}|\gamma\left(\frac{T}{|\mathcal{S}|}\right). \end{aligned} \tag{23}$$

Note that by convention $0\log(0) = 0$. The last inequality follows from arguments as in [8, eq. (23)]. If we now combine (21) and (23) we obtain

$$\begin{aligned} L(x_1^T) &\le \sum_{s\in\mathcal{S}}\left(a_s\log\frac{a_s+b_s}{a_s} + b_s\log\frac{a_s+b_s}{b_s}\right) \\ &\quad + |\mathcal{S}|\gamma\left(\frac{T}{|\mathcal{S}|}\right) + 2|\mathcal{S}| - 1 + \Delta_\mathcal{S}(x_1^T) + 2. \end{aligned} \tag{24}$$

Now assume that $d = 0, 1, 2, \cdots$ and let $\mathcal{S}$ be determined by d. For $d = 0$ let $\mathcal{S} = \{\lambda\}$, i.e., a (memoryless) tree with only a root node. For $d = 1, 2, \cdots$, take $\mathcal{S} = \{0,1\}^d$, i.e., a full (dth-order Markov) tree with depth d.

Fix some d: Then for $T \geq |\mathcal{S}| = 2^d$, and noting that $\Delta_{\mathcal{S}}(x_1^T) \leq d$, we have that

$$L(x_1^T) \leq \sum_{s \in \mathcal{S}} \left(a_s \log \frac{a_s + b_s}{a_s} + b_s \log \frac{a_s + b_s}{b_s} \right) + 2^{d-1} \log \frac{T}{2^d} + 3 \cdot 2^d + d + 1. \tag{25}$$

Note that a_s and b_s are now the number of zeros respectively ones in x_1^T that follow context $s \in \{0,1\}^d$.

Next let

$$c(T,d) \triangleq 2^{d-1} \log \frac{T}{2^d} + 3 \cdot 2^d + d + 1. \tag{26}$$

Then, from this definition and (25), we obtain that

$$\frac{L(x_1^T)}{T} \leq \sum_{s \in \mathcal{S}} \left(\left(-\frac{a_s}{T} \log \frac{a_s}{T} - \frac{b_s}{T} \log \frac{b_s}{T} \right) - \left(-\frac{a_s + b_s}{T} \log \frac{a_s + b_s}{T} \right) \right) + \frac{c(T,d)}{T}. \tag{27}$$

From the ergodic theorem (see, e.g., Shields [4]), since the actual source is stationary and ergodic, we know for $s \in \{0,1\}^d$ that

$$\lim_{T \to \infty} \frac{a_s}{T} = P_a(X_1^d = s, X_{d+1} = 0)$$

and

$$\lim_{T \to \infty} \frac{b_s}{T} = P_a(X_1^d = s, X_{d+1} = 1) \tag{28}$$

with probability one. Moreover, for all $d = 0, 1, 2, \cdots$

$$\lim_{T \to \infty} \frac{c(T,d)}{T} = 0. \tag{29}$$

Therefore, with probability one

$$\begin{aligned} \limsup_{T \to \infty} \frac{L(x_1^T)}{T} &\leq H(X_1, X_2, \cdots, X_d, X_{d+1}) \\ &\quad - H(X_1, X_2, \cdots, X_d) \\ &= H(X_{d+1} | X_1, X_2, \cdots, X_d). \end{aligned} \tag{30}$$

This bound holds for all $d = 0, 1, 2, \cdots$. Since

$$\lim_{d \to \infty} H(X_{d+1} | X_1, X_2, \cdots, X_d) = H_\infty(X)$$

we may conclude that for all binary stationary and ergodic sources the codeword length $L(x_1^T)$ divided by the sequence length T is, with probability one, not larger than the entropy $H_\infty(X)$ of the source.

V. Some Remarks

Coding schemes for the class of stationary sources were probably first studied by Shtarkov and Babkin [5]. They showed, using a combinatorial approach, that over this class the average codeword length converges to the source entropy.

We have shown here that the extended version of the context-tree weighting algorithm achieves entropy for all binary stationary and ergodic sources in the sense that with probability one the compression ratio (i.e., the number of code symbols $L(x_1^T)$ divided by the sequence length T) is not larger than the source entropy $H_\infty(X)$ for $T \to \infty$. A similar result was proved for the Ziv–Lempel incremental parsing procedure (tree algorithm) presented in [10].

Moreover, Ziv and Lempel showed that for any finite-state code the achievable compression ratio for an individual infinite-length sequence is lower-bounded by the limit of the empirical normalized block entropy $H(x_1, x_2, \cdots, x_d)/d$ of this sequence for $d \to \infty$ (see [10, Theorem 3]). The extended context-tree weighting algorithm achieves a compression ratio which is upper-bounded by the limit of the empirical conditional entropy $H(x_{d+1} | x_1, x_2, \cdots, x_d)$ for $d \to \infty$ of the individual infinite length source sequence. Both the empirical normalized block entropy and the empirical conditional entropy have the same limit so the method presented here is optimal in this sense. This also holds for the Ziv–Lempel incremental parsing method (see [10, Theorem 2]).

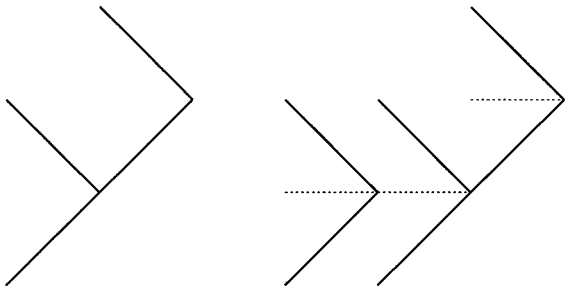

Fig. 4. Binary model $\{00, 10, 1\}$ and ternary model $\{00, 0\varepsilon 0, \varepsilon\varepsilon 0, 1\varepsilon 0, 10, \varepsilon, 1\}$.

The storage complexity of the method that we have described in this manuscript turned out to be not larger than linear in the source sequence length T. It should be mentioned that the storage complexity of the Ziv–Lempel incremental parsing algorithm is smaller, and behaves roughly like $T/\log(T)$. For the computational complexity the comparison is similar. The Ziv–Lempel method visits a new node in the dictionary tree for each processed source symbol, while for the context-tree weighting method it is necessary to go from the root of the context tree to a leaf for each processed symbol. Wyner and Ziv [9] showed that $\log(t)$ divided by the number of nodes that are visited in the context tree for processing x_t converges in probability to $H_\infty(X)$. Stronger results appear in Ornstein and Weiss [3] and Szpankowski [6].

Although it is possible to extend the context-tree weighting method to the nonbinary case, we emphasize that here only coding for binary sources is considered.

Appendix
Weighting over the Binary-Tree Models

We will show here that, although the context tree $\tilde{\mathcal{T}}_D$ is in principle ternary, $\tilde{P}_c(x_1^T)$ is a weighting over coding distributions $P_c'(x_1^T | \mathcal{S})$ for all binary-tree models $\mathcal{S} \in \mathcal{C}_D$. To see this, first consider, e.g., a binary-tree model $\mathcal{S} = \{00, 10, 1\}$ (see Fig. 4). A ternary-tree model that coexists with this binary model $\mathcal{S}$ is, e.g., $\tilde{\mathcal{S}} = \{00, 0\varepsilon 0, \varepsilon\varepsilon 0, 1\varepsilon 0, 10, \varepsilon, 1\}$. A ternary model $\tilde{\mathcal{S}}$ coexists with binary model $\mathcal{S}$ if $\mathcal{S} \subset \tilde{\mathcal{S}}$. The cost $\tilde{\Gamma}_D(\tilde{\mathcal{S}})$ of the ternary model $\tilde{\mathcal{S}}$ with respect to class $\tilde{\mathcal{C}}_D$ of ternary models is defined as

$$\tilde{\Gamma}_D(\tilde{\mathcal{S}}) \triangleq \frac{|\tilde{\mathcal{S}}| - 1}{2} + |\{s \colon s \in \tilde{\mathcal{S}}, l(s) \neq D\}| \tag{31}$$

where it is assumed that $\tilde{\mathcal{S}} \in \tilde{\mathcal{C}}_D$. Analogous to the binary case (see [8, Lemma 2]) we can show that $\tilde{P}_c(x_1^T)$ is a weighting over all ternary models $\tilde{\mathcal{S}}$. The weight of a ternary model $\tilde{\mathcal{S}}$ is $2^{-\tilde{\Gamma}_D(\tilde{\mathcal{S}})}$. We can, therefore, write

$$\begin{aligned} \tilde{P}_c(x_1^T) &= \sum_{\tilde{\mathcal{S}} \in \tilde{\mathcal{C}}_D} 2^{-\tilde{\Gamma}_D(\tilde{\mathcal{S}})} \prod_{s \in \tilde{\mathcal{S}}} P_e(\tilde{a}_s(x_1^T | \varepsilon^D), \tilde{b}_s(x_1^T | \varepsilon^D)) \\ &= \sum_{\mathcal{S} \in \mathcal{C}_D} \sum_{\tilde{\mathcal{S}} \in \tilde{\mathcal{C}}_{D \to \mathcal{S}}} 2^{-\tilde{\Gamma}_D(\tilde{\mathcal{S}})} \\ &\quad \cdot 2^{-\Delta_{\mathcal{S}}(x_1^T)} \prod_{s \in \mathcal{S}} P_e(a_s'(x_1^T), b_s'(x_1^T)) \\ &= \sum_{\mathcal{S} \in \mathcal{C}_D} 2^{-\Gamma_D(\mathcal{S})} P_c'(x_1^T | \mathcal{S}). \end{aligned} \tag{32}$$

Note that there is only one (underlying) binary model $\mathcal{S}$ that ternary model $\tilde{\mathcal{S}}$ can coexist with. The weights of all ternary models $\tilde{\mathcal{S}} \in \tilde{\mathcal{C}}_D$ that have underlying model $\mathcal{S}$ (i.e., models $\tilde{\mathcal{S}} \to \mathcal{S}$) sum up to $2^{-\Gamma_D(\mathcal{S})}$. Furthermore, observe that

$$\prod_{s \in \tilde{\mathcal{S}} \wedge\ s \notin \mathcal{S}} P_e(\tilde{a}_s, \tilde{b}_s) = 2^{-\Delta_{\mathcal{S}}(x_1^T)}$$

since the nodes s in this product must accommodate the $\Delta_{\mathcal{S}}(x_1^T)$ missing contexts. Finally, (1) is used to obtain the last equality.

Equation (32) can be used to give an alternative proof of (15). To see this note that

$$\tilde{P}_c(x_1^T) \geq 2^{-\Gamma_D(\mathcal{S})} P_c'(x_1^T|\mathcal{S}).$$

ACKNOWLEDGMENT

The author wishes to thank Associate Editor M. Feder and a reviewer for the comments which led to improvements of the present correspondence.

REFERENCES

[1] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, and R. McConell, "Linear size finite automata for the set of all subwords of a text: An outline of results," *Bull. Eur. Assoc. Theoret. Comput. Sci.*, vol. 21, pp. 12–20, 1983.

[2] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 199–207, Mar. 1981.

[3] D. S. Ornstein and B. Weiss, "Entropy and data compression schemes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 78–83, Jan. 1993.

[4] P. C. Shields, "The ergodic and entropy theorems revisited," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 263–266, Mar. 1987.

[5] Y. M. Shtarkov and V. F. Babkin, "Combinatorial method of universal coding for discrete stationary sources," in *Proc. 2nd Int. Symp. Information Theory* (Tsahkadsor, Armenian S.S.R., 1971). Budapest, Hungary: Publishing House of the Hungarian Acad. Sci., 1973, pp. 249–256.

[6] W. Szpankowski, "Asymptotic properties of data compression and suffix trees," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1647–1659, Sept. 1993.

[7] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "Context tree weighting: A sequential universal source coding procedure for FSMX sources," in *Proc. IEEE Int. Symp. Information Theory* (San Antonio, TX, Jan. 17–22, 1993), p. 59.

[8] ——, "Context tree weighting: Basic properties," *IEEE Trans. Inform. Theory*, vol. 41, pp. 653–664, May 1995.

[9] A. D. Wyner and J. Ziv, "Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1250–1258, Nov. 1989.

[10] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, Sept. 1978.

The Reed–Muller Code $R(r,m)$ Is Not $\mathbb{Z}_4$-Linear for $3 \leq r \leq m-2$

Xiang-Dong Hou, Jyrki T. Lahtonen, *Member, IEEE*, and Sami Koponen

***Abstract*— We show that the Reed–Muller code $R(r,m)$ is not $\mathbb{Z}_4$-linear for $3 \leq r \leq m-2$, proving a conjecture by Hammons, Kumar, Calderbank, Sloane, and Solé.**

***Index Terms*—General affine group, $\mathbb{Z}_4$-linear code, Reed–Muller code.**

I. INTRODUCTION

In the pioneering work [1], Hammons, Kumar, Calderbank, Sloane and Solé introduced the concept of $\mathbb{Z}_4$-linearity of binary codes. A binary code of even length is called $\mathbb{Z}_4$-linear if it is, up to a permutation of the coordinates, the image of a linear code over $\mathbb{Z}_4$ under the Gray map. The significant discovery of [1] is that several well-known families of nonlinear binary codes are actually $\mathbb{Z}_4$-linear. A question raised there was about the $\mathbb{Z}_4$-linearity of the Reed–Muller code $R(r,m)$. The authors of [1] proved that $R(r,m)$ is $\mathbb{Z}_4$-linear for $r = 0, 1, 2, m-1$, or m, but not $\mathbb{Z}_4$-linear for $r = m-2$ $(m \geq 5)$. Their conjecture was that $R(r,m)$ is not $\mathbb{Z}_4$-linear for $3 \leq r \leq m-2$. In this correspondence, we will prove the conjecture.

The multiplication by -1 in a linear code over $\mathbb{Z}_4$ induces a fixed-point-free involutory automorphism of its binary image. Such an automophism interchanges the two halves of the codewords in the binary code. A key step in proving $\mathbb{Z}_4$-nonlinearity is to classify the fixed-point-free involutory automorphisms of the binary code. Indeed, the proof of the $\mathbb{Z}_4$-nonlinearity of the Golay code in [1] was based on the classification of the conjugacy classes of the Mathieu group M_{24}.

II. BACKGROUND AND NOTATION

The Gray map ϕ: $\mathbb{Z}_4 \to \mathrm{GF}(2)^2$ is defined by

$$\begin{cases} 0 \mapsto (0,0) \\ 1 \mapsto (0,1) \\ 2 \mapsto (1,1) \\ 3 \mapsto (1,0) \end{cases} \tag{2.1}$$

and the Gray map ϕ: $\mathbb{Z}_4^n \to \mathrm{GF}(2)^{2n}$ is defined coordinate-wise using (2.1). A binary code C of even length $2n$ is called $\mathbb{Z}_4$-linear if, up to a permutation of coordinates, $C = \phi(\mathcal{C})$ for some subgroup $\mathcal{C}$ of $\mathbb{Z}_4^n$. The following criterion was established in [1].

Theorem 1: A binary linear code C of even length is $\mathbb{Z}_4$-linear if and only if there is a fixed-point-free involution $\sigma \in \mathrm{Aut}(C)$ such that

$$(u + \sigma(u)) * (v + \sigma(v)) \in C, \qquad \text{for all } u, v \in C \tag{2.2}$$

where $*$ is the coordinate-wise product (bitwise AND) of vectors.

The ambient space of the Reed–Muller codes is the algebra $\mathcal{P}_m$ of all Boolean functions on $\mathrm{GF}(2)^m$. Every $f \in \mathcal{P}_m$ is uniquely expressed as a polynomial function in $X_1, \cdots, X_m$ with coefficients

Manuscript received May 17, 1996; revised March 28, 1997.

X.-D. Hou is with the Department of Mathematics and Statistics, Wright State University, Dayton, OH 45435 USA.

J. T. Lahtonen and S. Koponen are with the Department of Mathematics, University of Turku, FIN-20014 Turku, Finland.

Publisher Item Identifier S 0018-9448(98)00844-X.