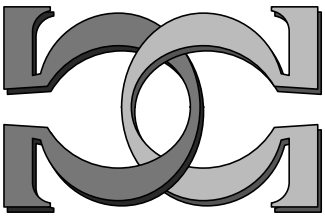
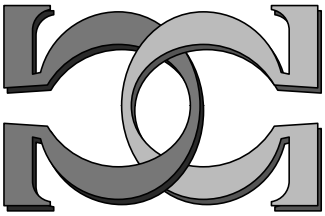
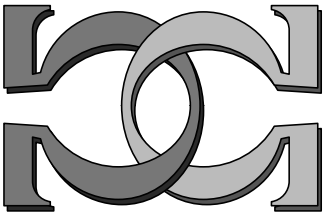


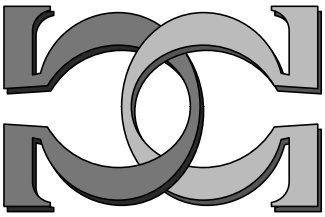
CDMTCS
Research Report Series



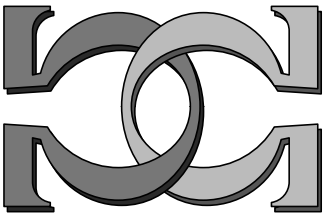
Deterministic Complexity and Entropy



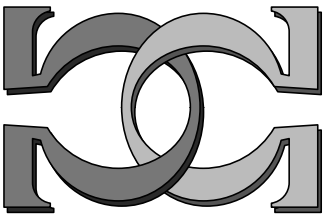
Mark R. Titchener
Department of Computer Science
University of Auckland
Auckland, New Zealand



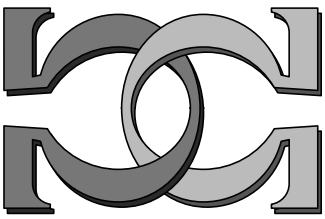
Aaron Gulliver
Department of Electrical & Computer Engineering
Univeristy of Victoria
Victoria, B.C. Canada



Radu Nicolescu
Ulrich Speidel
Department of Computer Science
University of Auckland
Auckland, New Zealand



Ludwig Staiger
Institut fir Informatik
Martin-Luther-Universität
Halle-Wittenburg, Germany



CDMTCS-255
December 2004

Centre for Discrete Mathematics and
Theoretical Computer Science

Deterministic Complexity and Entropy

Mark R. Titchener
Dept. of Computer Science
The University of Auckland, New Zealand

Aaron Gulliver
Dept. of Electrical & Computer Engineering
University of Victoria, Canada

Radu Nicolescu
Dept. of Computer Science
The University of Auckland, New Zealand

Ulrich Speidel
Dept. of Computer Science
The University of Auckland, New Zealand

Ludwig Staiger
Institut fűr Informatik
Martin-Luther-Universität Halle-Wittenberg, Germany

Abstract. Lempel and Ziv (1976) proposed a computable string production-complexity. In this paper, our emphasis is on providing the rigorous development, where possible, for the theoretical aspects of a more recent and contrasting measure of string complexity. We derive expressions for complexity bounds subject to certain constraints. We derive an analytic approximation to the upper bound to linearize the complexity measure. The linearized measure enables us to propose an entropy measure, observed elsewhere to correspond closely with the Kolmogorov-Sinai entropy in simple dynamical systems.

Keywords: Formal languages, generative systems, prefix codes, complexity measures, T-codes, entropy, information.

1. Introduction

In 1976, Lempel and Ziv proposed a computable measure, $C_{LZ} : A^+ \rightarrow \mathbb{N}$ for finite strings [6]. Motivated by the ideas of Solomonoff [13], Kolmogorov [5] and Chaitin [2], their production complexity measures the minimum number of steps required for the construction of a string. The steps are extracted

Address for correspondence: M. R. Titchener, Dept. of Computer Science, The University of Auckland, New Zealand, mark@tcode.auckland.ac.nz

from the string by means of a self-learning automaton. The measure, $\mathrm{C_T}$, discussed in this paper may be viewed as belonging to the same class of computable complexities as the Lempel-Ziv production complexity. Like the Lempel-Ziv production complexity, the measure discussed here [14, 15, 16, 17, 18, 19] uses a self-learning automaton to extract a production history from the string, and retains the notion of complexity based on the number of steps in the production process (or size of the vocabulary). However, the automaton used differs from that of Lempel and Ziv in that the vocabulary it builds represents new patterns as a recursive catenation of previously extracted patterns rather than as an existing vocabulary element extended by an "innovation". This means that, for a given string the T-complexity measure tends to be smaller than the LZ measure. For a given production step, the complexity increment in $\mathrm{C_T}$ is further weighted according to the number of repetitions of the corresponding vocabulary element, such that $\mathrm{C_T} : A^+ \rightarrow \mathbb{R}$.

As with the LZ complexity measure, the upper bound provides a reference against which string complexities may be measured, giving an estimate of the corresponding entropies. We derive the *lower* complexity bound, $\log_2 n$, where n is the string length, and bounds from *above* and *below* the *upper* complexity bound, for the special case where the copy factor at each production step is held to 1. It remains an open and mathematically challenging problem, to derive the upper bounds for the general case, when the copy factors are not restricted. However experimental results support the conjecture that the present bounds do in fact hold.

From the bounds above and below the upper T-complexity bound, we derive an analytical expression that lies approximately midway between, based on the logarithmic integral [1]. The expression, $\mathrm{li}(\ln r^n)$, where r is the alphabet size, though strictly a bound only in the limit turns out to be appropriate to characterise the maximum T-complexities for finite strings, and is thus effective for linearising the complexity measure. We propose alternative measures of information and entropy for finite strings.

We assume the reader to be familiar with the basic concepts and notations of formal languages and theory of codes such as alphabets, strings, languages, generative systems, code sets, codewords and prefix codes.

The rest of the paper is organized as follows. Section 2 introduces without proofs our basic concepts: T-augmentations, T-codes and the T-complexity measure for T-codes. Section 3 describes our string decomposition algorithm. Section 4 introduces our T-complexity measure for strings.

The new results on complexity bounds are then presented: a lower bound in Section 5, codeword distributions in Section 6, codeword distributions for exhaustive T-prescriptions in Section 7, the asymptotic behavior for the case $k_i = 1$ in Section 9. Alternative measures of information and entropy are proposed in Section 10, the open problem for $k_i > 1$ in Section 11, and we conclude in Section 12. Complete proofs of our results are published in Titchener et al. [20].

2. T-Augmentations, T-Codes and T-Complexity

In the following, we define a production step for generating prefix codes, called T-augmentation. Given an alphabet A, having r symbols, $r \geq 2$, a prefix code, $C \subset A^+$, a codeword, $p \in C$, and $k \in \mathbb{N}^+$, the *T-augmentation* of C with p and k is defined as:

$$C_{(p)}^{(k)} = \bigcup_{i=0}^{k} p^i C \setminus \bigcup_{i=0}^{k} p^i. \tag{1}$$

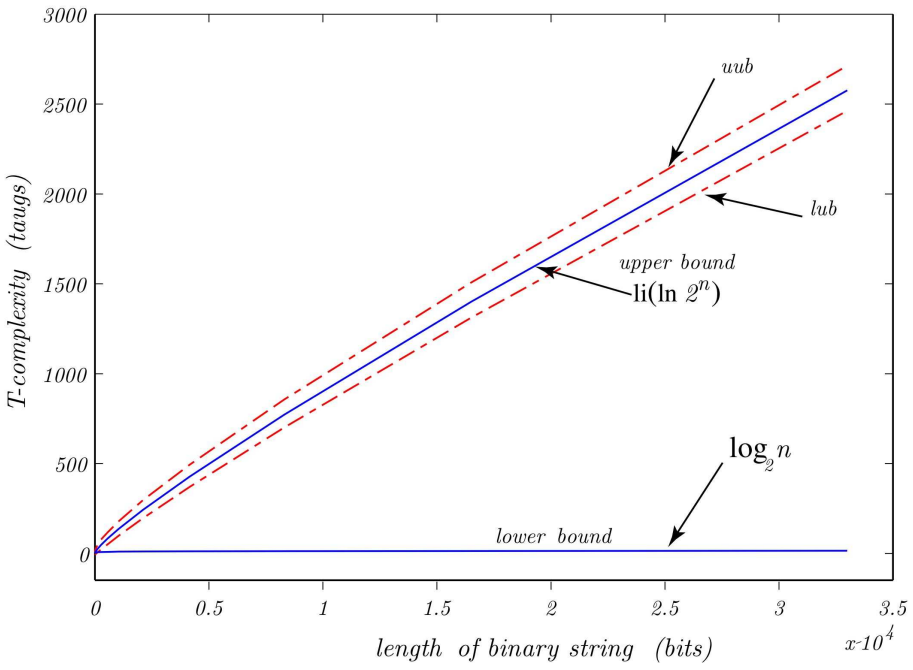


Figure 1. *Lower* and *upper* T-complexity bounds. The lower bound is $\log_2 n$ where n is the string length. *uub*, *lub* are, respectively, the *upper* and *lower* bounds on the *upper* bound, which itself lies near the logarithmic integral function, $\mathrm{li}(\ln r^n)$.

p is referred to as a *copy pattern* and k as a *copy factor* [12]. It follows that $C_{(p)}^{(k)}$ is also a prefix code [4, 12]. Eqn (1) may be applied recursively, starting with the alphabet A, to derive a series of prefix codes of the form:

$$A_{(p_1,\ldots,p_q)}^{(k_1,\ldots,k_q)} = \left(A_{(p_1,\ldots,p_{q-1})}^{(k_1,\ldots,k_{q-1})}\right)_{(p_q)}^{(k_q)},$$

where $q \geq 1$, $p_i \in A_{(p_1,\ldots,p_{i-1})}^{(k_1,\ldots,k_{i-1})}$ for $i = 1,\ldots,q$, and by convention $A_{()}^{()} = A$. Sets of the form $A_{(p_1,\ldots,p_{i-1})}^{(k_1,\ldots,k_{i-1})}$ are called *T-codes*, and, unrolling their definition, we may write:

$$A_{(p_1,\ldots,p_q)}^{(k_1,\ldots,k_q)} = \bigcup_{(i_1,\ldots,i_q)=(0,\ldots,0)}^{(k_1,\ldots,k_q)} p_q^{i_q} p_{q-1}^{i_{q-1}} \ldots p_1^{i_1} A \quad \setminus \bigcup_{(i_1,\ldots,i_q)=(0,\ldots,0)}^{(k_1,\ldots,k_q)} p_q^{i_q} p_{q-1}^{i_{q-1}} \ldots p_1^{i_1}. \tag{2}$$

T-augmentations may be visualised in graphical terms. Indeed, any prefix code, C, may be represented as a tree, denoted Γ_C, with the leaf nodes corresponding to the codewords of C. The T-augmentation process described by Eqn (1) entails making k copies of Γ_C, and appending the first copy to a selected leaf node, $p \in C$, and then appending each further copy in turn to the corresponding leaf node p in the most recently attached copy. Eqn (2) describes the elements of $A_{(p_1,\ldots,p_q)}^{(k_1,\ldots,k_q)}$, the leaf nodes of the tree, as the set of all nodes *minus* the set of internal nodes.

Example 2.1. Figure (2) illustrates the recursive construction of the T-code set, $A^{(2,1,1,1)}_{(1,10,10111,100)}$, where $A = \{0, 1\}$. (i) depicts the result of the first T-augmentation step, where $k_1 = 2$ implies two copies of the alphabet tree are to be made. The first of these is appended to the leaf node $p_1 = 1$, and the second to the corresponding position in the first copy. In the remaining steps, $k_i = 1$ for $i = 2, 3, 4$, so only one copy is appended to the leaf nodes, p_i.

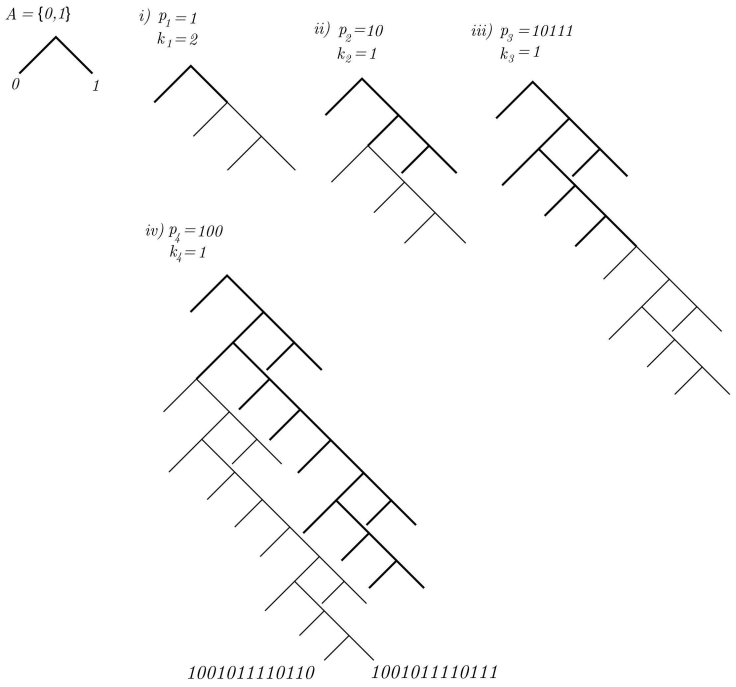


Figure 2. A recursive T-augmentation of a binary alphabet.

More formally, Nicolescu and Titchener [11, 12] refer to the system $P = \langle A, \boldsymbol{p}, \boldsymbol{k} \rangle$, formed by the alphabet, A, together with the vectors $\boldsymbol{p} = (p_1, \ldots, p_q)$ and $\boldsymbol{k} = (k_1, \ldots, k_q)$, as a *T-prescription* of the set $S = A^{\boldsymbol{k}}_{\boldsymbol{p}}$, written, $L(P) = S$. The number of elements in arrays $\boldsymbol{p}$ and $\boldsymbol{k}$ is referred to as the *arity* of the T-prescription, written $\mathrm{arity}(P) = q$. The length of the maximal-length codewords of P is denoted by $\mathrm{maxlen}(P)$.

- P is *simple*, if
  - all its copy factors are 1, i.e., $k_i = 1$, for $1 \leq i \leq q$.
- P is *systematic*, if
  1. it is simple, and
  2. it systematically uses shortest available codewords as copy patterns, i.e., $|p_{i+1}| = \min\{|p| \mid p \in L(\langle A, (p_1, \ldots, p_i), (k_1, \ldots, k_i) \rangle)\}$, for $0 \leq i \leq q - 1$.
- P is *exhaustive*, if

1. it is systematic, and
2. it has just exhausted all available copy patterns up to a certain length, but none longer, i.e., $p \in L(P) \Rightarrow |p_i| < |p| \; \forall i, 1 \leq i \leq q$.

For each of these T-prescription classes, we consider the sets of the lengths of their longest codewords: $\mathcal{N}_S = \{\text{maxlen}(P) \mid P \text{ simple T-prescription}\}, \mathcal{N}_Y = \{\text{maxlen}(P) \mid P \text{ systematic T-prescription}\}$, $\mathcal{N}_X = \{\text{maxlen}(P) \mid P \text{ exhaustive T-prescription}\}$. Clearly, $\mathcal{N}_X \subset \mathcal{N}_Y \subset \mathcal{N}_S = \mathbb{N}^+$. Specifically, the elements in $\mathcal{N}_X$ are referred to as *exhaustion lengths*. One may observe that, in the case of the binary alphabet, the exhaustion lengths are roughly powers of 2:

$$\mathcal{N}_X = \{1, 3, 9, 15, 39, 69, 135, 261, 549, 1053, 2103, 4149, 8301, 16491, \ldots\}$$

The T-complexity of P is defined by:

$$\boxed{\mathrm{C_T}(P) = \log_2(\prod_{i=1}^{q}(k_i + 1)) = \sum_{i=1}^{q} \log_2(k_i + 1).} \tag{3}$$

We extend the definition of the T-complexity by further defining two T-complexity measures for positive natural numbers, $\mathrm{C_T}, \mathrm{C_{T,S}} : \mathbb{N}^+ \to \mathbb{R}$:

$$\mathrm{C_T}(n) = \max\{\mathrm{C_T}(P) \mid \text{P is a T-prescription, } \text{maxlen}(P) = n\}, \tag{4}$$

$$\mathrm{C_{T,S}}(n) = \max\{\mathrm{C_T}(P) \mid \text{P is a simple T-prescription, } \text{maxlen}(P) = n\}. \tag{5}$$

Fact 2.1. Clearly,

$$\mathrm{C_{T,S}}(n) \leq \mathrm{C_T}(n), \forall n \in \mathbb{N}^+. \tag{6}$$

Fact 2.2. To evaluate $\mathrm{C_{T,S}}(n)$ for exhaustion lengths we need only consider exhaustive T-prescriptions:

$$\mathrm{C_{T,S}}(n) = \max\{\mathrm{C_T}(P) \mid \text{P is an exhaustive T-prescription, } \text{maxlen}(P) = n\}, \forall n \in \mathcal{N}_X. \tag{7}$$

Fact 2.3. Two exhaustive T-prescriptions with the same length longest codewords have the same T-complexity. Thus, given $n \in \mathcal{N}_X$, $\mathrm{C_{T,S}}(n) = \mathrm{C_T}(P)$, for any exhaustive T-prescription, P, having $\text{maxlen}(P) = n$.

The number of internal nodes in any T-code tree is given by $\prod_{i=1}^{q}(k_i + 1)$ and, fairly naturally, the T-complexity gives the number of bits required to label uniquely all the internal nodes. In a sense, we are effectively computing the effort to construct the T-code set from the alphabet, i.e., the number of T-augmentation steps. We measure this in *taugs* rather than *bits*, implying *T-augmentation steps*. Since all $k_i \geq 1$, we see that $\mathrm{C_T}(P) \geq q$. In the case where only a single copy is made at each of the T-augmentation steps, $\mathrm{C_T}(P) = q$.

Consider two T-prescriptions, P and P'. P and P' are *equivalent* if they generate the same language: $L(P) = L(P')$. P is in *canonical* form if it has minimal arity among all equivalent T-prescriptions, i.e., if $L(P) = L(P')$, then $\text{arity}(P) \leq \text{arity}(P')$. Nicolescu and Titchener [12] have previously shown that a canonical T-prescription is *unique* in its equivalence class, i.e., if P and P' are canonical T-prescriptions

and $L(P) = L(P')$, then $P = P'$. Thus *every* T-code set S has a *unique canonical* T-prescription, P_S, and define its T-complexity:

$$\boxed{\mathrm{C_T}(S) = \mathrm{C_T}(P_S).} \tag{8}$$

It can be further shown that the T-complexity for T-prescriptions is *equivalence invariant*, which means that any T-prescription, P, may be used to evaluate the T-complexity of a T-code set, S:

$$L(P) = S \Rightarrow \mathrm{C_T}(S) = \mathrm{C_T}(P). \tag{9}$$

Example 2.2. Examples of T-code sets based on the alphabet $A = \{0, 1\}$. Observe:

$$\begin{aligned} S_1 &= A^{(1,1,1)}_{(0,1,00)} = \{0000, 0001, 00100, 00101, 0011, 01, 100, 101, 11\}, \\ S_2 &= A^{(2,1)}_{(0,01)} = \{000, 001, 01000, 01001, 0101, 011, 1\}. \end{aligned}$$

$P_1 = \langle A, (0, 1, 00), (1, 1, 1)\rangle$ is the canonical T-prescription that generates the set S_1, $L(P_1) = S_1$; P_1 is systematic. $P_2 = \langle A, (0, 01), (2, 1)\rangle$ is the canonical T-prescription that generates the set S_2, $L(P_2) = S_2$; P_2 is not simple and therefore cannot be systematic. Calculating the T-complexities we obtain $\mathrm{C_T}(S_1) = C_T(P_1) = \log_2(8) = 3$ and $\mathrm{C_T}(S_2) = C_T(P_2) = \log_2(6) = 2.58$.

Example 2.3. An example of a T-code set with multiple equivalent T-prescriptions based on the alphabet $A = \{0, 1\}$. Observe:

$$S = A^{(3,1)}_{(0,1)} = A^{(1,1,1)}_{(0,00,1)} = \{0000, 0001, 001, 01, 10000, 10001, 1001, 101, 11\}.$$

The set S can be generated in two ways: either using the canonical T-prescription, $P_1 = \langle A, (0, 1), (3, 1)\rangle$, or else the non-canonical T-prescription, $P_2 = \langle A, (0, 00, 1), (1, 1, 1)\rangle$. P_1 is not simple; P_2 is simple but not systematic. Calculating the T-complexities, we obtain $\mathrm{C_T}(S) = C_T(P_1) = C_T(P_2) = \log_2(8) = 3$.

3. String Decomposition

A symmetry implicit in the T-code trees allows one to discover the sequence of steps for construction of the tree from any one of the longest codewords in the T-code set. As illustrated in Figure (2), the intermediate trees generated in the course of a recursive T-code construction appear not only at the top (root) of the tree, as one expects from the copying process, but also at the bottom. This means that the longest codewords in the set traverse the tree through the root of each intermediate subtree. Thus implicit in the longest codewords are the steps for the construction of the tree, including the nodes of the tree which are not traversed by the codeword.

The algorithm for decomposing a string into its constituent copy patterns and copy factors is here demonstrated by way of example only. Though our example assumes a binary alphabet, the process is may be extended to cover an alphabet of any size. The foundations of this algorithm are described by Nicolescu and Titchener [12].

Example 3.1. Assume the alphabet, $A = \{0, 1\}$, and the string, $x = 0100010101101$. We use the period (.) as a delimiting marker between codewords to facilitate reading along the string.

1. We start with the T-code set, $S_0 = A^{()}_{()} = A$, and we decode the string, x, over the codewords in S_0:
$$x = 0100010101101 = 0.1.0.0.0.1.0.1.0.1.1.0.1$$
We identify the penultimate codeword which will be our first copy pattern, $p_1 = 0$, and, as this is not replicated immediately to its left, we take $k_1 = 1$. Conceptually, we construct a new T-code set, $S_1 = A^{(1)}_{(0)} = \{00, 01, 1\}$, and decode the string, x, over the codewords in S_1, In practical terms, we don't actually construct the set, but achieve the same by grouping each occurrence of p_1 with its next codeword:
$$x = 01.00.01.01.01.1.01$$

2. We select the penultimate codeword, $p_2 = 1$, and, since this does not repeat immediately to the left, we take $k_2 = 1$. Conceptually, we again construct a new T-code set, $S_2 = A^{(1,1)}_{(0,1)} = \{00, 01, 100, 101, 11\}$, and decode the string, x, over the codewords in S_2. In practice, we simply group each occurrence of p_2 with its next codeword:
$$x = 01.00.01.01.01.101$$

3. We select the penultimate codeword, $p_3 = 01$, and, since this repeats twice immediately to its left, we take $k_3 = 3$. Conceptually, we again construct a new T-code set, $S_3 = A^{(1,1,3)}_{(0,1,01)}$, and decode the string, x, over the codewords in S_3. In practice, we simply group each occurrence of p_3 with its next codeword, or codewords, up to 3 times, if p_3 repeats itself:
$$x = 0100.010101101$$

4. We select the penultimate codeword, $p_4 = 0100$, and, since this does not repeat to its left, we take $k_4 = 3$. Conceptually, we again construct a new T-code set, $S_4 = A^{(1,1,3,1)}_{(0,1,01,0100)}$, and decode the string, x, over the codewords in S_4. In practice, we simply group each occurrence of p_4 with its next codeword:
$$x = 0100010101101$$
We have reduced the initial string, x, to a single codeword, which indicates completion of the cycle.

We thus have a T-code set, $S = S_4$, and canonical T-prescription, $P = \langle A, (0, 1, 01, 0100), (1, 1, 3, 1)\rangle$. Computing the complexity: $\mathrm{C_T}(S) = \mathrm{C_T}(P) = \sum_{i=1}^{4} \log_2(k_i + 1) = 5$ *taugs*.

Nicolescu and Titchener [12] show that, for any given string $x \in A^+$, the algorithm illustrated above will always return a canonical T-prescription for a T-code set that contains x as one of its longest codewords.

Current implementations [21] cater for alphabets of up to $r = 256$ symbols and may process strings of more than $4 \cdot 10^7$ symbols, though the time to process a string of length n runs in time $\mathcal{O}(n^2)$. Speidel and Yang [22] have made performance improvements, culminating in a more recent unpublished algorithm running in time $\mathcal{O}(n)$.

4. String T-Complexity

Though T-augmentation was previously defined as a *set construction* process, it may also be more narrowly viewed as a *string production* process for the *longest codewords* in the set. For example, the T-augmentation given in Example (3.1) may be viewed as a process that produces the string 0100010101101. As another example, the T-augmentation depicted in Figure (2) may be viewed as a process that produces the strings 1001011110110 and 1001011110111.

Nicolescu and Titchener [12] show that, for *every* string, $x \in A^+$, there exists a *unique* T-code set, T_x, for which x is a maximal-length codeword. Thus for every string, $x \in A^+$, we can compute a corresponding T-complexity by way of the T-complexity of the associated unique T-code set, T_x, and its unique canonical T-prescription, P_x:

$$\boxed{\mathrm{C_T}(x) = \mathrm{C_T}(T_x) = \mathrm{C_T}(P_x).} \tag{10}$$

5. Lower Bound

For any $n > 0$, the T-complexity of a length n string, x, is *minimal* and equal to $\log_2 n$ when in Eqn (3) $q = 1$, with a single copy pattern, $p_1 \in A$, repeated $k_1 = n - 1$ times. Thus the lower bound is $\log_2 n$, which proves the following theorem:

Theorem 5.1. (Lower Bound)
The T-complexity function is bounded below by the logarithm base 2 of the string length.

$$\boxed{\log_2 n \leq \mathrm{C_T}(x), \text{ for all } x \in A^n.} \tag{11}$$

We note that the alphabet size, r, does not figure in this bound which is not altogether surprising. A string of one hundred repeating 1's (or repeating 5's for that matter) may be formed from any number of alphabets, but by our definition its T-complexity is independent of the alphabet size, which is not the case for the maximal bound.

6. Codeword Length Distributions

The maximal bound is more difficult to derive. For this we need to know something about how the codeword length distributions grow with each choice of copy pattern, for these ultimately determine what choices may follow in the construction process. We may visualize the *distribution of codeword lengths* in a T-code set by an infinite vector, $\boldsymbol{d}$, whose elements after index n are all 0, where n is the length of the longest codewords:

$$\boldsymbol{d} = (d_1, d_2, \ldots, d_n, 0, \ldots).$$

We introduce the following notations with reference to a codeword length distribution, $\boldsymbol{d}$:

- $|\boldsymbol{d}|$ is the distribution size, i.e., the total number of codewords, $|\boldsymbol{d}| = \sum_{i=1}^{\infty} d_i$.
- $\mathrm{First}(\boldsymbol{d})$ is the index of the first non-zero element in $\boldsymbol{d}$; it is the minimum length of all codewords in $\boldsymbol{d}$, $\mathrm{First}(\boldsymbol{d}) = \min\{i \mid i \in \mathbb{N}^+, d_i \neq 0\}$.

- $\mathrm{Last}(\boldsymbol{d})$ is the index of the last non-zero element in $\boldsymbol{d}$; it is the maximum length of all codewords in $\boldsymbol{d}$, $\mathrm{Last}(\boldsymbol{d}) = \max\{i \mid i \in \mathbb{N}^+, d_i \neq 0\}$.

- $\mathrm{MaxAt}(\ell)$ is the largest codeword count that can appear at index, $\ell \in \mathbb{N}^+$, over all possible distributions that have ℓ as the index of the first non-zero element; $\mathrm{MaxAt}(\ell) = \max\{d_\ell \mid \forall \boldsymbol{d}$ s.t. $\mathrm{First}(\boldsymbol{d}) = \ell\}$.

- $\mathrm{MinLast}(\ell)$ is the smallest index of the last non-zero elements over all possible distributions that have ℓ the index of the first non-zero element; $\mathrm{MinLast}(\ell) = \min\{\mathrm{Last}(\boldsymbol{d}) \mid \forall \boldsymbol{d}$ s.t. $\mathrm{First}(\boldsymbol{d}) = \ell\}$.

Where the distribution $\boldsymbol{d}$ is obvious, we use the abbreviations: $\ell = \mathrm{First}(\boldsymbol{d})$, $m_\ell = \mathrm{MaxAt}(\mathrm{First}(\boldsymbol{d}))$, $n_\ell = \mathrm{MinLast}(\mathrm{First}(\boldsymbol{d}))$.

Fact 6.1.

- For an arbitrary T-prescription, $d_{\mathrm{Last}(\boldsymbol{d})} = r$.

- For an arbitrary T-prescription, $d_\ell \leq m_\ell$ and $n_\ell \leq \mathrm{Last}(\boldsymbol{d})$.

- For an *exhaustive* T-prescription, $d_\ell = m_\ell$ and $n_\ell = \mathrm{Last}(\boldsymbol{d})$.

- Any *systematic* T-prescription, for which either $d_\ell = m_\ell$ or $n_\ell = \mathrm{Last}(\boldsymbol{d})$, is an *exhaustive* T-prescription.

- The set of all lengths n_ℓ is $\mathcal{N}_X$, the set of all exhaustion lengths.

- The mapping $\mathbb{N}^+ \to \mathcal{N}_X$ defined by $\ell \to n_\ell$ is a monotone bijection, thus invertible, and we can express n_ℓ as a function of ℓ, and vice-versa.

Example 6.1. Examples of code distributions; items 1, 2, 4 correspond to exhaustive T-prescriptions, while 3 does not.

1. For the binary alphabet, $\boldsymbol{d} = (2, 0, \ldots)$, $\ell = 1$, $m_\ell = 2$, $n_\ell = 1$.

2. For the T-code set S_1 of Example (2.2), $\boldsymbol{d} = (0, 2, 2, 3, 2, 0, \ldots)$, $\ell = 2$, $m_\ell = 3$, $n_\ell = 5$.

3. For the T-code set S_2 of Example (2.2), $\boldsymbol{d} = (1, 3, 1, 2, 0, \ldots)$, $\ell = 1$.

4. For the T-code set S of Example (2.3), $\boldsymbol{d} = (0, 2, 2, 3, 2, 0, \ldots)$, $\ell = 2$, $m_\ell = 3$, $n_\ell = 5$.

We use a generating function to algebraically work with the distributions resulting from T-augmentations, writing:

$$\boldsymbol{d}(z) = d_1 z^1 + d_2 z^2 + d_3 z^3 + \ldots + d_n z^n + \ldots = \sum_1^\infty d_j z^j.$$

Prefixing all elements in a T-code set using a copy pattern of length ℓ results in a distribution represented by the product, $\boldsymbol{d}(z) \cdot z^{\ell}$. Assuming an initial distribution, $\boldsymbol{d}(z)$, for a T-code, S, from Eqn (1), the distribution for $S_{(p)}^{(k)}$ is $\dot{\boldsymbol{d}}(z)$, as defined by:

$$\dot{\boldsymbol{d}}(z) = \sum_{j=0}^{k} \boldsymbol{d}(z) \cdot z^{\ell \cdot j} - \sum_{j=1}^{k} z^{\ell \cdot j} = \boldsymbol{d}(z) \sum_{j=0}^{k} z^{\ell \cdot j} - \sum_{j=0}^{k} z^{\ell \cdot j} + 1, \tag{12}$$

where $\ell = |p|$. Thus:

$$\dot{\boldsymbol{d}}(z) - 1 = (\boldsymbol{d}(z) - 1) \sum_{j=0}^{k} z^{\ell \cdot j}. \tag{13}$$

If $k = 1$, Eqn (13) reduces to:

$$\dot{\boldsymbol{d}}(z) - 1 = (\boldsymbol{d}(z) - 1)\,(1 + z^{\ell}) = \boldsymbol{d}(z)(1 + z^{\ell}) - z^{\ell} - 1. \tag{14}$$

If we repeat the T-augmentation with another copy pattern also of identical length ℓ, then we obtain a new distribution $\ddot{\boldsymbol{d}}$, as defined by:

$$\ddot{\boldsymbol{d}}(z) - 1 = \left(\boldsymbol{d}(z)(1 + z^{\ell}) - z^{\ell}\right)(1 + z^{\ell}) - z^{\ell} = \boldsymbol{d}(z)(1 + z^{\ell})^2 - z^{\ell}(1 + z^{\ell}) - z^{\ell}. \tag{15}$$

7. Code Distributions for Exhaustive T-Prescriptions

We now consider the principal case which was empirically observed to drive the growth of the T-complexity in terms of the string lengths. Exhaustive T-prescriptions yield codewords that grow minimally with each T-augmentation step, which means that, their T-complexity relative to the string lengths tends towards the maximum. In this and the next section we concern ourselves mainly with exhaustive T-prescriptions. In Section 9 we show how the results for exhaustive T-prescriptions may be extended to cover the systematic and simple T-prescriptions. Though examples of T-prescriptions may be found which have T-complexities that exceed systematic or simple T-prescriptions with the same maximum length strings, these differences are empirically observed to be, relatively speaking, small perturbations to the fundamental behaviour.

Consider $\boldsymbol{d}^{\ell}(z)$, the distribution associated with an exhaustive T-prescription P, where all $m_{\ell-1}$ copy patterns of length $\ell - 1$ are consumed. Thus the smallest available copy patterns in $\boldsymbol{d}^{\ell}(z)$ are of length ℓ. We may now repeat the T-augmentation process m_{ℓ} times and we obtain a new exhaustive T-prescription with the distribution, $\boldsymbol{d}^{\ell+1}(z)$. We have:

$$\begin{aligned}
\boldsymbol{d}^{\ell+1}(z) &= \boldsymbol{d}^{\ell}(z)(1 + z^{\ell})^{m_{\ell}} - \left[(1 + z^{\ell})^{m_{\ell}-1} + (1 + z^{\ell})^{m_{\ell}-2} + \ldots + (1 + z^{\ell})^{1} + 1\right] z^{\ell} \\
&= \boldsymbol{d}^{\ell}(z)(1 + z^{\ell})^{m_{\ell}} - \left[\frac{(1 - (1 + z^{\ell})^{m_{\ell}})}{(1 - (1 + z^{\ell}))}\right] z^{\ell} \\
&= \boldsymbol{d}^{\ell}(z)(1 + z^{\ell})^{m_{\ell}} - (1 + z^{\ell})^{m_{\ell}} + 1 \\
&= (\boldsymbol{d}^{\ell}(z) - 1)(1 + z^{\ell})^{m_{\ell}} + 1.
\end{aligned} \tag{16}$$

Evaluating the highest order term yields $z^{\ell m_\ell}\boldsymbol{d}^\ell(z)$, from which we conclude:

$$\boxed{n_{\ell+1} = n_\ell + \ell m_\ell.} \tag{17}$$

Unrolling Eqn (16) to the initial alphabet (having r symbols), successively using m_i copy patterns of length i, for $1 \leq i \leq \ell$, yields:

$$\boldsymbol{d}^{\ell+1}(z) - 1 = (rz-1)\cdot\prod_{i=1}^{\ell}(z^i+1)^{m_i} \tag{18}$$

$$= (rz-1)\sum_{i_\ell=0}^{m_1}\binom{m_1}{i_1}z^{i_1}\sum_{i_2=0}^{m_2}\binom{m_2}{i_2}z^{2i_2}\cdots\sum_{i_\ell=0}^{m_\ell}\binom{m_\ell}{i_\ell}z^{\ell i_\ell} \tag{19}$$

$$= (rz-1)\sum_{i_1=0}^{m_1}\sum_{i_2=0}^{m_2}\cdots\sum_{i_\ell=0}^{m_\ell}\binom{m_1}{i_1}\binom{m_2}{i_2}\cdots\binom{m_\ell}{i_\ell}z^{i_1+2i_2+\cdots+\ell i_\ell}. \tag{20}$$

8. Upper Bounds for Exhaustive T-Prescriptions

In this section, we consider a given exhaustive T-prescription P and with this P derive the upper bound of the T-complexity for the special case of exhaustive T-prescriptions. For the first ℓ coefficients d_i^ℓ of the distribution polynomial $\boldsymbol{d}^\ell(z) = \sum_{i=0}^{\infty} d_i^\ell \cdot z^i$ we have

$$d_0^\ell = \cdots = d_{\ell-1}^\ell = 0 \text{ and } d_\ell^\ell = m_\ell\,. \tag{21}$$

Consider $\mathcal{P}^\ell(z) = \prod_{i=1}^{\ell-1}(z^i+1)^{m_i}$. According to Eqns (18) and (21), its first coefficients, $\pi_0^\ell, \ldots, \pi_\ell^\ell$, satisfy the identities:

$$\begin{aligned} \pi_0^\ell &= 1, \\ \pi_k^\ell &= r\cdot\pi_{k-1}^\ell, \text{ for } k = 1,\ldots,\ell-1, \text{ and} \\ \pi_\ell^\ell &= r\cdot\pi_{\ell-1}^\ell - m_\ell\,. \end{aligned} \tag{22}$$

Consequently, $\pi_k^\ell = r^k$, for $k = 0,\ldots,\ell-1$ and $\pi_\ell^\ell = r^\ell - m_\ell$. Differentiating Eqn (18) yields:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}z}\boldsymbol{d}^\ell(z) &= r\cdot\mathcal{P}^\ell(z) + (rz-1)\sum_{i=1}^{\ell-1}\left(\frac{i\cdot m_i\cdot z^{i-1}}{z^i+1}\cdot\mathcal{P}^\ell(z)\right) \\ &= \mathcal{P}^\ell(z)\left(r + (rz-1)\sum_{i=1}^{\ell-1}(i\cdot m_i\cdot z^{i-1}\cdot\sum_{j=0}^{\infty}(-1)^j z^{ji})\right). \end{aligned} \tag{23}$$

For the last sum, we obtain:

$$\begin{aligned} \mathcal{Q}^\ell(z) &= \sum_{i=1}^{\ell-1}(i\cdot m_i\cdot z^{i-1}\cdot\sum_{j=0}^{\infty}(-1)^j z^{ji}) = \sum_{i=1}^{\ell-1}(i\cdot m_i\cdot\sum_{j=0}^{\infty}(-1)^j z^{i(j+1)-1}) \\ &= \sum_{k=0}^{\infty}\sum_{i=1}^{\ell-1} i\cdot m_i\cdot(\sum_{i(j+1)-1=k}(-1)^{\frac{k+1}{i}+1})\cdot z^k \\ &= \sum_{k=0}^{\infty}(\sum_{\substack{i\mid k+1 \\ i\leq k}}(-1)^{\frac{k+1}{i}+1}\cdot i\cdot m_i)\cdot z^k. \end{aligned}$$

In order to achieve our result, we consider only the first ℓ coefficients in Eqn (23), i.e., we consider Eqn(23) modulo z^ℓ:

$$\begin{aligned}\ell \cdot m_\ell \cdot z^{\ell-1} &= \frac{\mathrm{d}}{\mathrm{d}z}\boldsymbol{d}^\ell(z) \pmod{z^\ell} \\ &= r\cdot\mathcal{P}^\ell(z) + ((rz-1)\cdot\mathcal{P}^\ell(z))\cdot\mathcal{Q}^\ell(z)) \pmod{z^\ell} \\ &= \sum_{k=0}^{\ell-1} r^{k+1}z^k - \mathcal{Q}^\ell(z) \pmod{z^\ell},\end{aligned} \tag{24}$$

because, $\mathcal{P}^\ell(z) = \sum_{k=0}^{\ell-1} r^k z^k \pmod{z^\ell}$ and $(rz-1)\cdot\mathcal{P}^\ell(z) = -1 \pmod{z^\ell}$. Comparing the coefficient of $z^{\ell-1}$ yields:

$$\ell \cdot m_\ell = r^\ell - \sum_{\substack{i|\ell \\ i<\ell}} (-1)^{\frac{\ell}{i}+1}\cdot i \cdot m_i. \tag{25}$$

Now the length of the longest string $n_\ell \in \boldsymbol{d}^\ell(z)$, is $n_\ell = \sum_{j=1}^{\ell-1} j\cdot m_j + 1$, and therefore by Eqn (25):

$$\begin{aligned} n_\ell &= \sum_{j=1}^{\ell-1} r^j + \sum_{j=1}^{\ell-1}\sum_{\substack{i=1 \\ i|j}}^{j-1} (-1)^{\frac{j}{i}}\cdot i\cdot m_i + 1 \\ &= \left(\frac{r^\ell-1}{r-1} - 1\right) + \sum_{i=1}^{\ell-2}\left(\sum_{\substack{j=i+1 \\ i|j}}^{\ell-1}(-1)^{\frac{j}{i}}\right)\cdot i\cdot m_i + 1 \\ &= \frac{r^\ell-1}{r-1} + \sum_{i=1}^{\ell-2}\left(\sum_{k=2}^{\lfloor\frac{\ell-1}{i}\rfloor}(-1)^k\right)\cdot i\cdot m_i. \end{aligned} \tag{26}$$

A *lower bound* on n_ℓ is obtained by assuming that every inner sum is 0:

$$n_\ell \geq \frac{r^\ell-1}{r-1}. \tag{27}$$

An *upper bound* is obtained by assuming that every inner sum is 1:

$$\begin{aligned} n_\ell &\leq \left\lfloor\frac{r^\ell-1}{r-1}-1\right\rfloor + \left\lfloor\frac{r^{\lfloor(\ell-1)/2\rfloor+1}-1}{r-1}-1\right\rfloor + \left\lfloor\frac{(r^{\lfloor(\ell-1)/4\rfloor+1}-1)}{r-1}-1\right\rfloor + \cdots + 1 \\ &\leq \frac{r^\ell + r^{\lfloor(\ell-1)/2\rfloor+1} + r^{\lfloor(\ell-1)/4\rfloor+1} + \cdots - \lfloor\log_2(\ell)\rfloor - (r-1)\lfloor\log_2(\ell)\rfloor}{r-1} \\ &< \frac{r^\ell + r^{\lfloor(\ell-1)/2\rfloor+1} + r^{\lfloor(\ell-1)/4\rfloor+1} + \cdots}{r-1} < \frac{r^\ell + \sum_{i=1}^{\lfloor(\ell-1)/2\rfloor+1} r^i}{r-1} \\ &< \frac{r^\ell}{r-1} + \frac{r^{\lfloor(\ell+3)/2\rfloor}}{(r-1)^2}. \end{aligned} \tag{28}$$

Bringing Eqns (27) and (28) together:

$$\boxed{\frac{r^\ell - 1}{r-1} \;\leq\; n_\ell \;<\; \frac{r^\ell}{r-1} \;+\; \frac{r^{\lfloor(\ell+3)/2\rfloor}}{(r-1)^2}.} \tag{29}$$

Now, selecting the *upper* and *lower* bounds on $n_{\ell+1}$ and n_ℓ, respectively, gives an upper bound on $n_{\ell+1} - n_\ell$. Conversely, selecting the *lower* and *upper* bounds on $n_{\ell+1}$ and n_ℓ, respectively, gives a lower bound on $n_{\ell+1} - n_\ell$ and, from Eqn (17), yields:

$$\left[\frac{r^{(\ell+1)}-1}{r-1}\right] - \left[\frac{r^\ell}{r-1} + \frac{r^{\lfloor(\ell+3)/2\rfloor}}{(r-1)^2}\right] < \ell m_\ell < \left[\frac{r^{\ell+1}}{r-1} + \frac{r^{\lfloor(\ell+4)/2\rfloor}}{(r-1)^2}\right] - \left[\frac{r^\ell - 1}{r-1}\right].$$

Combining terms and dividing by ℓ gives:

$$\frac{r^\ell}{\ell} - \frac{r^{\lfloor(\ell+3)/2\rfloor} + r - 1}{\ell(r-1)^2} < m_\ell < \frac{r^\ell}{\ell} + \frac{r^{\lfloor(\ell+4)/2\rfloor} + r - 1}{\ell(r-1)^2}. \tag{30}$$

For our exhaustive T-prescription, P, Eqn (3) reduces to a count of the codewords consumed as copy patterns, i.e., $\mathrm{C_T}(P) = \sum_{i=1}^{\ell-1} m_i$. From Eqn (30):

$$\begin{aligned}
\sum_{i=1}^{\ell-1}\frac{r^i}{i} - \sum_{i=1}^{\ell-1}\frac{r^{\lfloor(i+3)/2\rfloor}}{i(r-1)^2} - \sum_{i=1}^{\ell-1}\frac{1}{i(r-1)} \;&<\mathrm{C_T}(P)<\; \sum_{i=1}^{\ell-1}\frac{r^i}{i} + \sum_{i=1}^{\ell-1}\frac{r^{\lfloor(i+4)/2\rfloor}}{i(r-1)^2} + \sum_{i=1}^{\ell-1}\frac{1}{i(r-1)}, \\
\sum_{i=1}^{\ell-1}\frac{r^i}{i} - \delta_L(\ell) \;&<\mathrm{C_T}(P)<\; \sum_{i=1}^{\ell-1}\frac{r^i}{i} + \delta_U(\ell),
\end{aligned} \tag{31}$$

where

$$\begin{aligned}
\delta_L(\ell) \;&=\; \frac{r}{(r-1)^2}\sum_{i=1}^{\ell-1}\frac{r^{\lfloor(i+1)/2\rfloor}}{i} + \frac{1}{(r-1)}\sum_{i=1}^{\ell-1}\frac{1}{i}, \\
\delta_U(\ell) \;&=\; \frac{r^2}{(r-1)^2}\sum_{i=1}^{\ell-1}\frac{r^{\lfloor i/2\rfloor}}{i} + \frac{1}{(r-1)}\sum_{i=1}^{\ell-1}\frac{1}{i}.
\end{aligned} \tag{32}$$

From Fact (2.3), Eqn (33) holds also for $\mathrm{C_{T,S}}(n_\ell)$, where $n_\ell \in \mathcal{N}_X$:

$$\boxed{\sum_{i=1}^{\ell-1}\frac{r^i}{i} - \delta_L(\ell) < \mathrm{C_{T,S}}(n_\ell) < \sum_{i=1}^{\ell-1}\frac{r^i}{i} + \delta_U(\ell),} \tag{33}$$

where ℓ is considered a function of n_ℓ, as indicated in Fact (6.1). The upper and lower bounds on $\mathrm{C_T}(n_\ell)$ computed from Eqn (33) for a binary alphabet ($r = 2$) are plotted in Figure (1), as functions of n_ℓ.

9. Asymptotic Behavior for Simple T-Prescriptions

In this section, we derive an analytic expression for the asymptotic bound for simple T-prescriptions. Though the proof holds only in the limit, the resultant expression is in practice an adequate approximation over the whole range of string lengths.

Fact 9.1. $\mathrm{li}(r^b) - \mathrm{li}(r^a) = \int_a^b \frac{r^u}{u} du.$

Proof:
The proof uses the definition of li and the substitution, $v = r^u$.

$$\int_a^b \frac{r^u}{u} du = \int_{r^a}^{r^b} \frac{v \ln r}{lnv} \frac{dv}{vlnr} = \int_{r^a}^{r^b} \frac{dv}{lnv} = \mathrm{li}(r^b) - \mathrm{li}(r^a).$$

□

Fact 9.2. $\lim_{\ell\to\infty} \left(\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^\ell)\right) = \infty.$

Proof:
The proof uses Fact (9.1).

$$\begin{aligned}
\lim_{\ell\to\infty} \left(\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^\ell)\right) &= \lim_{\ell\to\infty} \int_\ell^{\ell+1} \frac{r^u}{u} du \geq \lim_{\ell\to\infty} \int_\ell^{\ell+1} \frac{(1+1)^u}{u} du \\
&\geq \lim_{\ell\to\infty} \int_\ell^{\ell+1} \left(\frac{u}{u} + \frac{u(u-1)}{2u}\right) du \geq \lim_{\ell\to\infty} \int_\ell^{\ell+1} \frac{u-1}{2} du \\
&\geq \lim_{\ell\to\infty} \frac{\ell^2}{4} - \frac{(\ell-1)^2}{4} = \lim_{\ell\to\infty} \frac{2\ell-1}{4} = \infty.
\end{aligned}$$

□

Fact 9.3. $\lim_{\ell\to\infty} \frac{\delta_L(\ell)}{\mathrm{li}(\ln r^{n_\ell})} = 0, \lim_{\ell\to\infty} \frac{\delta_U(\ell)}{\mathrm{li}(\ln r^{n_\ell})} = 0.$

Proof:
The proof uses Cesaro-Stolz, L'Hôpital's rule; and Eqn (29); the details are omitted here. □

Fact 9.4. $\lim_{\ell\to\infty} \frac{r^\ell/\ell}{\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^\ell)} = \frac{\ln r}{r-1}.$

Proof:
Using L'Hôpital's rule.

$$\begin{aligned}
\lim_{\ell\to\infty} \frac{r^\ell/\ell}{\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^\ell)} &= \lim_{\ell\to\infty} \frac{r^\ell/\ell \cdot (\ln r - 1)/\ell)}{\frac{r^{\ell+1}.\ln r}{\ln r^{\ell+1}} - \frac{r^\ell.\ln r}{\ln r^\ell}} = \ln r \cdot \lim_{\ell\to\infty} \frac{1/\ell}{\frac{r}{(\ell+1)} - \frac{1}{\ell}} = \ln r \cdot \lim_{\ell\to\infty} \frac{1/\ell}{\frac{\ell r - \ell - 1}{\ell(\ell+1)}} \\
&= \ln r \cdot \lim_{\ell\to\infty} \frac{(\ell+1)}{\ell r - \ell - 1} = \frac{\ln r}{r-1}.
\end{aligned}$$

□

Fact 9.5. $\lim_{\ell\to\infty} \dfrac{\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^{\ell})}{\mathrm{li}(\ln r^{n_{\ell+1}}) - \mathrm{li}(\ln r^{n_{\ell}})} = \dfrac{r-1}{\ln r}.$

Proof:
Using L'Hôpital's rule and Eqn (29); the details are omitted. □

Fact 9.6. $\lim_{\ell\to\infty} \dfrac{\sum_{i=1}^{\ell-1} \frac{r^i}{i}}{\mathrm{li}(\ln r^{n_{\ell}})} = 1.$

Proof:
Using Cesaro-Stolz.

$$\begin{aligned}\lim_{\ell\to\infty} \frac{\sum_1^{\ell} r^i/i}{\mathrm{li}(\ln r^{n_{\ell+1}})} &= \lim_{\ell\to\infty} \frac{r^{\ell}/\ell}{\mathrm{li}(\ln r^{n_{\ell+1}}) - \mathrm{li}(\ln r^{n_{\ell}})} \\ &= \lim_{\ell\to\infty} \frac{r^{\ell}/\ell}{\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^{\ell})} \cdot \frac{\mathrm{li}(r^{\ell+1}) - \mathrm{li}(r^{\ell})}{\mathrm{li}(\ln r^{n_{\ell+1}}) - \mathrm{li}(\ln r^{n_{\ell}})}.\end{aligned}$$

The rest of the proof follows from application of Facts (9.4) and (9.5). □

Lemma 9.1. (Asymptotic Upper Bound)
For all simple T-prescriptions, $\mathrm{li}(\ln r^{n_{\ell}})$ is asymptotic to the upper bound of the T-complexity, $\mathrm{C}_{\mathrm{T,S}}(n_{\ell})$, for all exhaustion lengths, n_{ℓ}.

$$\boxed{\lim_{\ell\to\infty} \frac{\mathrm{C}_{\mathrm{T,S}}(n_{\ell})}{\mathrm{li}(\ln r^{n_{\ell}})} = 1.} \tag{34}$$

Proof:
We start by dividing all terms of Eqn (33) by $\mathrm{li}(\ln r^{n_{\ell}})$.

$$\begin{aligned}\sum_{i=1}^{\ell-1} \frac{r^i}{i} - \delta_L(\ell) \;&<\; \mathrm{C}_{\mathrm{T,S}}(n_{\ell}) < \sum_{i=1}^{\ell-1} \frac{r^i}{i} + \delta_U(\ell) \\ \frac{\sum_{i=1}^{\ell-1} \frac{r^i}{i}}{\mathrm{li}(\ln r^{n_{\ell}})} - \frac{\delta_L(\ell)}{\mathrm{li}(\ln r^{n_{\ell}})} \;&<\; \frac{\mathrm{C}_{\mathrm{T,S}}(n_{\ell})}{\mathrm{li}(\ln r^{n_{\ell}})} < \frac{\sum_{i=1}^{\ell-1} \frac{r^i}{i}}{\mathrm{li}(\ln r^{n_{\ell}})} + \frac{\delta_U(\ell)}{\mathrm{li}(\ln r^{n_{\ell}})}.\end{aligned}$$

The rest of the proof follows from Facts (9.3) and (9.6). □

We note that the concavity of $\mathrm{li}(\ln r^n)$ ensures this asymptotic bound is valid not only for the exhaustion lengths, $n_{\ell} \in \mathcal{N}_X$, but for all lengths $n \in \mathcal{N}_Y$, corresponding to maximal-length strings of systematic T-prescriptions. More so, the simple but non-systematic T-prescriptions are also asymptotically bounded by $\mathrm{li}(\ln r^n)$, because the choice of a longer copy pattern has the effect of reducing the T-complexity relative to the string length. These arguments can be used to prove the following main theorem. The proof is omitted here, but can be found in [20].

Theorem 9.1. (Asymptotic Upper Bound)
For all simple T-prescriptions, $\mathrm{li}(\ln r^n)$ is asymptotic to the upper bound of the T-complexity, $\mathrm{C}_{\mathrm{T,S}}(n)$, for all string lengths, $n \in \mathbb{N}^+$.

$$\boxed{\lim_{\ell\to\infty} \frac{\mathrm{C}_{\mathrm{T,S}}(n)}{\mathrm{li}(\ln r^n)} = 1.} \tag{35}$$

If r is the alphabet size, and n the length of a string, we recognize that r^n is simply the number of distinct strings of length n. $\ln r^n$ is the minimum amount of information in *nats*, required to uniquely distinguish each of these strings.

We surmise from this that the T-complexity of a finite string is simply the logarithmic integral of the total information content of the string. In the next section, we define the T-information and T-entropy with this assumption in mind.

Figure (3) shows a magnified view of Figure (1), near the origin. It shows the lower T-complexity bound, represented by the function $\log_2 n$. The "staircase" represents the upper bound for simple T-prescriptions. The leading "step-edges" of the staircase correspond to systematic T-prescriptions, and the vertical drop-lines, at exhaustion lengths, 3, 5, 9, 15, 39, correspond to exhaustive T-prescriptions. Though only proved to be a bound asymptotically, the expression, $\mathrm{li}(\ln 2^n)$, is close to the upper bound, even for small string lengths.

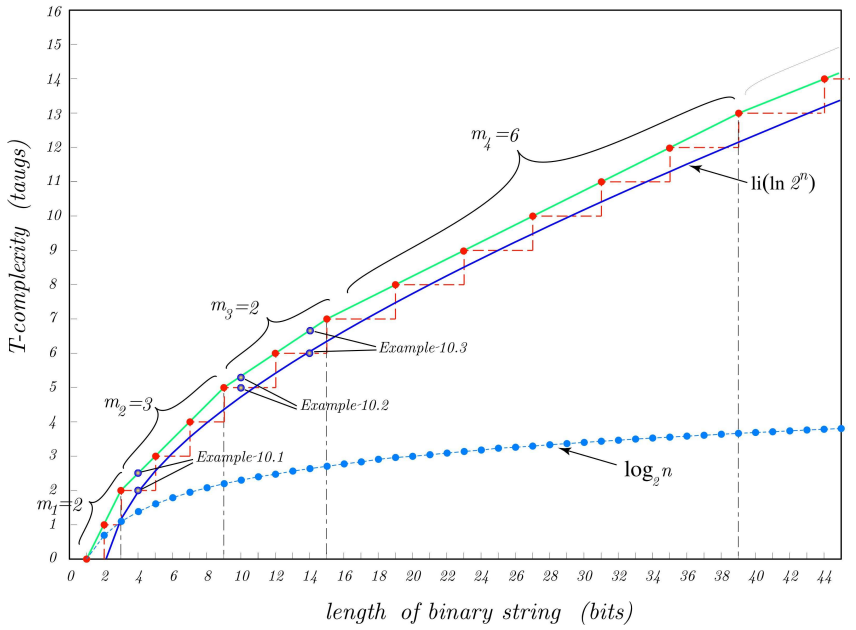


Figure 3. A magnified view of the T-complexity bounds, near the origin.

10. T-Information and T-Entropy

We define the *T-information* denoted $\mathrm{I_T}(x)$ of a string, $x \in A^+$, as that quantity whose logarithmic integral yields the corresponding T-complexity of the string. Whereas the T-complexity is measured in *taugs*, the T-information of the string is measured in *nats*. The defining relationships are thus:

$$\begin{aligned} C_T(x)\ \text{(taugs)} &= \mathrm{li}(\mathrm{I_T}(x)\ \text{(nats)}), \\ I_T(x)\ \text{(nats)} &= \mathrm{li}^{-1}(\mathrm{C_T}(x)\ \text{(taugs)}). \end{aligned}$$

The T-entropy is simply defined as the average T-information rate:

$$\mathrm{H_T}(x)\ \text{(nats/symbols)} = \frac{\mathrm{I_T}(x)\ \text{(nats)}}{|x|\ \text{(symbols)}} = \frac{\mathrm{li}^{-1}(\mathrm{C_T}(x)\ \text{(taugs)})}{|x|\ \text{(symbols)}}.$$

Ebeling, Steuer, and Titchener [3] have empirically shown that the T-entropy is close to the Kolmogorov-Sinai entropy for certain simple non-linear dynamical systems. Further work continues in this area.

11. Open Problems

For many string lengths, some non-simple T-prescriptions achieve higher T-complexity values than all simple T-prescriptions.

Claim 11.1. These are some of the first string lengths whose corresponding simple T-prescriptions do not achieve maximum T-complexity values.

$$4, 6, 8, 10, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, \ldots$$

Empirically, the relative differences are observed to be small, and all T-complexity values seem to lie between the bounds derived in this paper. Further work is necessary.

Example 11.1. Consider two canonical T-prescriptions that generate longest codewords of length 4: $P_1 = \langle\{0,1\},(1,10),(1,1)\rangle$, $P_2 = \langle\{0,1\},(1,0),(2,1)\rangle$, $L(P_1) = \{0, 100, 1010, 1011, 11\}$, $L(P_2) = \{00, 010, 0110, 0111, 10, 110, 111\}$. Their complexities are: $\mathrm{C_T}(P_1) = \log_2(2{\cdot}2) = \log_2(4)$, $\mathrm{C_T}(P_2) = \log_2(3 \cdot 2) = \log_2(6)$. Observe that P_1 is simple, but P_2 is not, and yet $\mathrm{C_T}(P_2) > \mathrm{C_T}(P_1)$.

Example 11.2. Consider any two canonical T-prescriptions that generate longest codewords of length 10: $P_1 = \langle\{0,1\}, \boldsymbol{p_1}, \boldsymbol{k_1}\rangle$, $P_2 = \langle\{0,1\}, \boldsymbol{p_2}, \boldsymbol{k_2}\rangle$, where $\boldsymbol{p_1}$ has lengths $(1,1,2,2,3)$, $\boldsymbol{p_2}$ has lengths $(1,1,2,2)$, $\boldsymbol{k_1} = (1,1,1,1,1)$ and $\boldsymbol{k_2} = (4,1,1,1)$. Their complexities are: $\mathrm{C_T}(P_1) = \log_2(2 \cdot 2 \cdot 2 \cdot 2 \cdot 2) = \log_2(32)$, $\mathrm{C_T}(P_2) = \log_2(5 \cdot 2 \cdot 2 \cdot 2) = \log_2(40)$. Observe that P_1 is simple, but P_2 is not, and yet $\mathrm{C_T}(P_2) > \mathrm{C_T}(P_1)$.

Example 11.3. Consider any two canonical T-prescriptions that generate longest codewords of length 14: $P_1 = \langle\{0,1\}, \boldsymbol{p_1}, \boldsymbol{k_1}\rangle$, $P_2 = \langle\{0,1\}, \boldsymbol{p_2}, \boldsymbol{k_2}\rangle$, where $\boldsymbol{p_1}$ has lengths $(1,1,2,3)$, $\boldsymbol{p_2}$ has lengths $(1,1,2,2,3,4)$, $\boldsymbol{k_1} = (4,4,1,1)$ and $\boldsymbol{k_2} = (1,1,1,1,1,1)$. Their complexities are: $\mathrm{C_T}(P_1) = \log_2(5 \cdot 5 \cdot 2 \cdot 2) = \log_2(100)$, $\mathrm{C_T}(P_2) = \log_2(2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2) = \log_2(64)$. Observe that P_1 is not simple, but P_2 is simple, and yet $\mathrm{C_T}(P_1) > \mathrm{C_T}(P_2)$.

The authors have identified the following open problems:

- To prove, or disprove, that there is an infinity of string lengths n, where $\mathrm{C_T}(n) > \mathrm{C_{T,S}}(n)$, i.e., where non-simple T-prescriptions achieve higher T-complexity values than all simple T-prescriptions.
- To prove, or disprove, that our bounds hold in general, for $\mathrm{C_T}$, not only $\mathrm{C_{T,S}}$, i.e., for all T-prescriptions, and not just for simple T-prescriptions.

12. Summary and Conclusions

Practical computable information measures are becoming increasingly important in areas, as diverse as medical instrumentation, bio-informatics, dynamical systems, communications, network management and monitoring, coding, compression and data mining. In this paper, we have described a particular set of measures based around the structured coding properties implicit in the T-code construction. Following the example of Lempel and Ziv, we have proposed a string production process from which the complexity of individual strings may be computed. We have derived bounds on the complexity measure for systematic T-prescriptions. We have observed from an extensive body of empirical evidence that the mathematically less tractable general case is at least consistent with the derived bounds. The upper and lower bounds for the upper T-complexity bound are dominated by a series which we further proved to be asymptotically equivalent to an analytical expression involving the logarithmic integral. The significance of the latter is that it allows us to propose linearized measures of the complexity, that reflect the additivity properties and units more usually associated with information and entropy.

We further identified open problems in respect of the derivation of a generalized proof of the bounds. However the growing weight of empirical evidence encourages us to believe that a full proof will possible in the course of time. The close correspondence between our measures and the Kolmogorov-Sinai entropy of well studied dynamical systems like the logistic map provides evidence of the connections between Shannon's probabilistic definitions and our own deterministic formulations. The duality that exists between stochastic and deterministic descriptions of non-linear dynamical systems appears to suggest that ultimately a formal proof of equivalence with Shannon's probabilistic information theory will be forthcoming.

13. Acknowledgements

The authors wish to acknowledge the fundamental contributions by Prof Solomon Marcus, who has pioneered new directions in the study of information, entropy and complexity measures [7, 8, 9, 10]. We are grateful for his generous advice and encouragement.

We are indebted to John Morris and Ioan Tomescu for their many thoughtful suggestions. Of course, the authors take full responsibility for all remaining shortcomings in this article.

References

[1] Abramowitz, M., and Stegun, I. A.: *Handbook of Mathematical Functions*, Dover Publications, Inc., NY, 1965.

[2] Chaitin, G.: *A theory of program size formally identical to information theory*, Rep. RC 4805, IBM, Yorktown Heights, N.Y., April 1974.

[3] Ebeling, W., Steuer, R., Titchener, M. R.: Partition-based entropies of deterministic and stochastic maps, *Stochastics and Dynamics*, **1**(1), 2001, 45–61.

[4] Günther U.: *Robust source coding with generalized T-codes*, PhD Thesis, The University of Auckland, 1998, `http://www.citr.auckland.ac.nz/~ulrich/phd.ps.gz`.

[5] Kolmogorov, A. N.: Three approaches to the quantitative definition of information, *Probl. Inform. Transmission*, **1**, 1965, 1–7.

[6] Lempel, A., and Ziv, J.: On the Complexity of Finite Sequences, *IEEE Transactions on Information Theory*, **22**(1), 1976, 75–81.

[7] Marcus, S.: Entropie et energie poetique, *Cahiers de Linguistique Théorique et Appliquée*, **4**, 1967, 171–180.

[8] Marcus, S.: On types of meters of a poem and their informational energy, *Semiotica*, **4**(1), 1971, 31–36.

[9] Marcus, S.: The poetic relevance of the information energy, in: *Studies in Probability and Related Topics* (M. Demetrescu, M. Iosifescu, Eds.), Nagard, Roma, 1983, 355–360.

[10] Marcus, S.: Symmetry as periodicity and complexity as absence of symmetry, *Proceedings of the 2nd International Conference on Symmetry and Antisymmetry in Mathematics, Formal Languages and Computer Science*, Satellite Conference of the Third European Congress of Mathematics, Brasov, June 29 – July 1, 2000, 17–19.

[11] Nicolescu, R.: *Uniqueness theorems for T-codes*, TR.9, Tamaki Report Series, Auckland, September 1995.

[12] Nicolescu, R., Titchener, M. R.: Uniqueness theorems for T-codes, *Romanian Journal of Information Science and Technology*, **1**(3), 1998, 243–258.

[13] Solomonoff, R. J.: A formal theory of inductive inference, *Information and Control*, Part I: **7**(1), 1964, 1–22, Part II: **7**(2), 1964, 224–254.

[14] Titchener, M. R.: A measure of information, *Proceedings Data Compression Conference*, Snowbird, UT, 2000, 353–362.

[15] Titchener, M. R.: Deterministic computation of complexity, information and entropy, *Proceedings IEEE International Symposium on Information Theory*, 16–21 Aug 1998, 326.

[16] Titchener, M. R.: A novel deterministic method for evaluating the entropy of language texts, *3rd Conference in Information-Theoretic Approaches to Logic, Languages, and Computation*, Hsi-tou, Taiwan, June 1998.

[17] Titchener, M. R.: A deterministic theory of complexity, information and entropy, *IEEE Information Theory Workshop*, San Diego, CA, February 1998, 80.

[18] Titchener, M. R.: Generalised T-codes: extended construction algorithm for self-synchronising codes, *IEE Proceedings – Communications*, **143**(3), 1996, 122–128.

[19] Titchener, M. R.: Digital encoding by means of new T-codes to provide improved data synchronization and message integrity, *IEE Proceedings – Computers and Digital Techniques*, Technical Note, **131**(4), 1984, 51–53.

[20] Titchener, M. R., Gulliver, A., Nicolescu, R., Speidel, U., Staiger, L: *Deterministic Complexity and Entropy*, TR 255, CDMTCS, Auckland, 2004.

[21] Wackrow, S., Titchener, M. R., Günther U.: *Tcalc.c*, Source code under GNU licence, The University of Auckland, 1999, `http://tcode.auckland.ac.nz/~mark/tcalc.c`.

[22] Yang, J., Speidel, U.: An improved T-decomposition algorithm, *Fourth ICICS and IEEE PCM Proceedings*, Singapore, December 2003, Vol.3, 1551–1555.