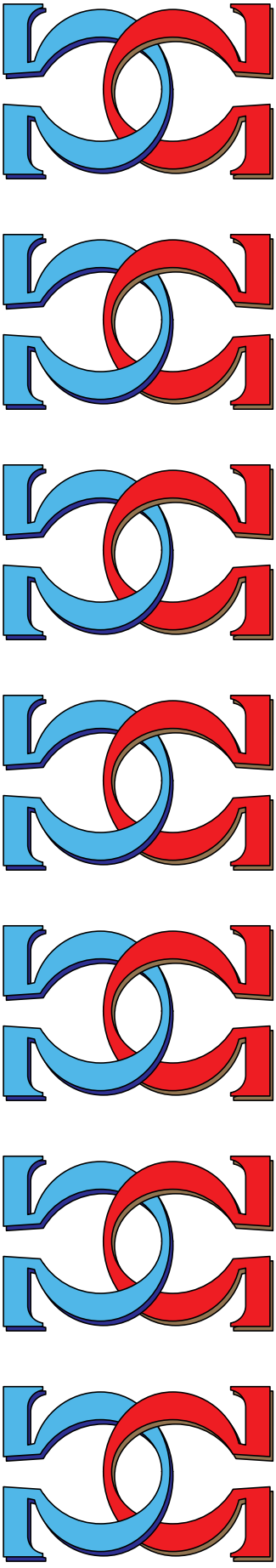


CDMTCS
Research
Report
Series

T-Complexity and T-Information Theory – an Executive Summary, 2nd revised version

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

CDMTCS-286
October 2006

Centre for Discrete Mathematics and
Theoretical Computer Science

T-Complexity and T-Information Theory – an Executive Summary, 2nd revised version

Ulrich Speidel

October 25, 2006

Abstract

This paper describes the derivation of the T-complexity and T-information theory from the decomposition of finite strings, based on the duality of strings and variable-length T-codes. It further outlines its similarity to the string parsing algorithm by Lempel and Ziv. In its first version [15], it was intended as a summary of work published mainly by Titchener and Nicolescu. Apart from minor corrections, the present extended version incorporates feedback from previous readers and presents new results obtained since.

1 A brief introduction to T-codes

This paper first gives an introduction to T-codes and their construction technique, as this is fundamental for the understanding of the T-decomposition algorithm that underpins the T-complexity measure. It also assists in its physical interpretation.

T-codes [3, 4, 11] are similar to Huffman codes in that they are codes with variable-length codewords. Depending on the source symbol probabilities, a Huffman code may also be a T-code. In fact, any T-code set could theoretically have been constructed as a result of a Huffman code construction algorithm. By inference, every T-code set is *complete*, a property also called *exhaustive* by some authors. In complete codes, all internal nodes of the (Huffman) decoding tree are fully populated. Moreover, like Huffman codes, T-codes are *prefix codes*, again a property that is also – somewhat confusingly – known as *prefix-freeness*, implying that no codeword in the set is a proper prefix of another.

Similarities largely end there, however. Most Huffman codes are not T-codes, and at least from today's perspective, the significance of T-codes is not in Huffman-like source coding. Rather than letting the source symbol probabilities determine the length of the codewords and the structure of the decoding

tree, T-codes are constructed without regard to symbol probabilities. Their construction focuses instead on a recursive tree structure.

A T-code set is constructed as follows:

1. Start with a finite alphabet S (e.g., the binary alphabet, where $S = \{0, 1\}$). Every finite alphabet is a (trivial) T-code set by default, with the letters being primitive codewords.
2. Given a T-code set, another T-code set may be derived from it by a process called "T-augmentation". This involves choosing an arbitrary codeword from the existing T-code set, called the "T-prefix", and a positive integer, called the "T-expansion parameter"[3, 4, 11] or "copy factor"[14]. Any T-code set may thus be derived from an alphabet in a series of n T-augmentations using a series of T-prefixes $p_1, p_2, \ldots, p_n$ and a series of T-expansion parameters $k_1, k_2, \ldots, k_n$. The resulting set is denoted $S^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ and is said to be a T-code set at "T-augmentation level" n.

The T-augmentation itself is performed according to the following equation:

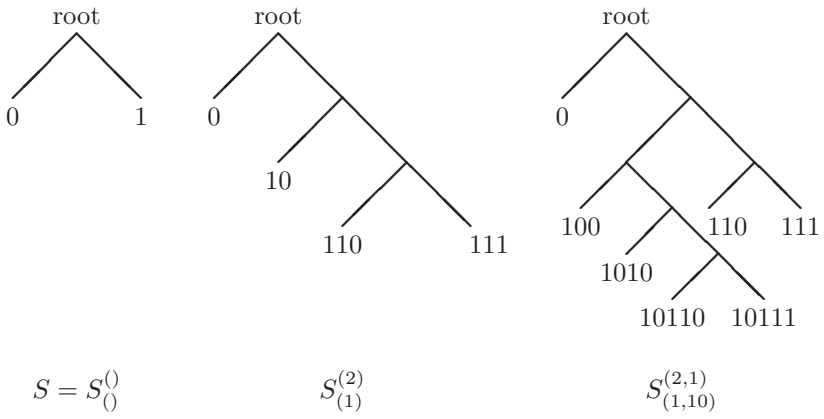
$$S^{(k_1,k_2,\ldots,k_{n+1})}_{(p_1,p_2,\ldots,p_{n+1})} = \bigcup_{i=0}^{k_{n+1}} \{p^i_{n+1}s | s \in S^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)} \backslash \{p_{n+1}\}\} \cup \{p^{k+1}_{n+1}\} \qquad (1)$$

where $S^{()}_{()} = S$ and $p^i_{n+1}s$ donates the concatenation of i copies of the T-prefix p_{n+1} with s.

In the tree picture, this is equivalent to the following operation:

1. select a T-prefix p_{n+1} from the existing set $S^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$, which has been derived from an alphabet S by $n \geq 0$ T-augmentations.
2. select a T-expansion parameter $k_{n+1} \geq 1$.
3. make k_{n+1} additional copies of the decoding tree.
4. Successively concatenate the new copies and the original of the tree via the T-prefix chosen, i.e., the root of the first copy attaches to p_{n+1}, that of the second copy to $p_{n+1}p_{n+1}$, etc.

Figure 1 depicts the construction of the T-code set $S^{(2,1)}_{(1,10)}$ starting from the elementary decoding tree for the alphabet S. The latter consists of a root node and two branches denoting the "reception" of a 0 and 1 respectively. The branches terminate in two leaf nodes, which correspond to the two trivial codewords 0 and 1 in S.



.

Figure 1: T-augmentation as a copy-and-append process of decoding trees

In the first T-augmentation in this example, the tree for S is copied three times and the three copies are linked to each other via the respective leaf nodes corresponding to the codeword 1. The second T-augmentation links two copies of the tree for $S^{(2)}_{(1)}$ via the leaf node 10.

Another example, using codeword lists rather than trees, is shown in Table 1. Note here that each codeword, once created, remains in the subseqent sets unless it is used as a T-prefix, at which point it disappears for good. By T-augmenting a set over and over again, we can generate arbitrarily large sets.

By choosing the T-prefixes and the T-expansion parameters wisely, a T-code tree may be shaped to suit a particular source. However, this is not of prime relevance here as this paper is not about Huffman-style source coding.

It is worth noting here that some T-code sets may be constructed with more than one set of T-prefixes and T-expansion parameters. E.g., the set $S^{(3)}_{(0)}$ is the same as the set $S^{(1,1)}_{(0,00)}$. A set of T-prefixes and T-expansion parameters used in the construction of a T-code set is called a "T-prescription". As the example illustrates, a T-code set may have more than one T-prescription. The concept of a T-augmentation level is thus only defined with respect to a particular T-prescription. However, all T-prescriptions for a given T-code set can be derived from each other with ease. For the purposes of this paper we shall assume that, if several T-prescriptions exist, we will always refer to the one T-prescription for which the T-expansion parameters are maximised (anti-canonical T-prescription). For a detailed discussion of this topic see [2] or [8]. A less rigorous version of the proof may also be found in [11].

	T-augmentation level			
n	0	1	2	3
k_n	n/a	1	1	3
set	S	$S^{(1)}_{(1)}$	$S^{(1,1)}_{(1,10)}$	$S^{(1,1,3)}_{(1,10,0)}$
	0	0	0	$\emptyset$
	1	$\not{1}$	–	–
		10	$\not{1}\emptyset$	–
		11	11	11
			100	100
			–	–
			1010	1010
			1011	1011
				$\emptyset\emptyset$
				–
				–
				011
				0100
				–
				01010
				01011
				$\emptyset\emptyset\emptyset$
				–
				–
				0011
				00100
				–
				001010
				001011
				0000
				–
				–
				00011
				000100
				–
				0001010
				0001011

Table 1: T-augmentation from the binary alphabet S via the intermediate T-code sets $S^{(1)}_{(1)}$ and $S^{(1,1)}_{(1,10)}$ to the final set $S^{(1,1,3)}_{(1,10,0)}$. The columns show the codewords in the respective T-code sets. The "deleted" strings are the now internal nodes of the new decoding tree.

2 The significance of the longest codewords

Consider the two longest codewords in the sets of the examples in Table 1 and Figure 1. In each set, the longest codewords differ by exactly one letter, the last letter, and they all belong to the same copy of the elementary tree for S. Thus, the number of longest codewords is given by the cardinality of the alphabet S. In our binary sets, for example, there are exactly two longest codewords per set. Furthermore, the longest codewords contain – in reverse succession – all the T-prefixes that were used in the construction of the set. The length of the runs of the T-prefixes in the longest codewords equals the T-expansion parameter (in the T-prescription for which the T-expansion parameter is maximised).

As it turns out, it is possible to derive each T-code set from any of its longest codewords. The algorithm for this is described in the next section. For the moment, let us simply note that it exists.

Furthermore, given an arbitrary finite string over an arbitrary finite alphabet, it is always possible to find a T-code set for which this string is one of its longest codewords. This set is unique, i.e., there is no other T-code set for which the same string is also one of the longest codewords. For a proof of this theorem see [2] or [8].

Since the longest codewords in any T-code set are identical except for the last letter, we may regard their common part as an identifier for this T-code set.

This duality between strings and T-code sets means that we can use the T-code set construction algorithm not only to construct codes, *but also as a string construction (production) algorithm*. The T-augmentations are the steps in this algorithm.

If we consider the string production aspect, we can create (or lengthen) a string as follows:

1. take an existing string (which may consist of just one letter) and consider the T-code set $S^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ for which it is one of the longest codewords.

2. pick a codeword p_{n+1} from $S^{(k_1,k_2,\ldots,k_n)}_{(p_1,p_2,\ldots,p_n)}$ and append k_{n+1} copies of it to the left of the string. The new string is now one of the longest codewords from $S^{(k_1,k_2,\ldots,k_{n+1})}_{(p_1,p_2,\ldots,p_{n+1})}$.

If we repeat this as often as we desire, we can generate arbitrarily long strings of the format:

$$p_n^{k_n} p_{n-1}^{k_{n-1}} \ldots p_2^{k_2} p_1^{k_1} a \tag{2}$$

where $a \in S$.

Consider now the choice of T-prefix codeword and T-expansion parameter: if we choose a long p_{n+1} to append to the left, we create a longer resulting string than by choosing a short codeword. However, the number of steps needed to create the string remains constant - and the patterns that the "extra" bit of string includes are indeed all patterns we have already seen as T-prefixes in the initial set (or, in other words, as substrings in the original string).

Similarly, by choosing a large k_{n+1}, one does not add much extra information, but merely repeats an already occurring pattern. Note that a similar theme of construction steps and focus on recurring patterns is found in parsing algorithm underpinning the Lempel and Ziv production complexity [1].

Before we take this theme further, however, we need to discuss how an existing string can be parsed to yield the corresponding T-code set.

The next section describes how one arrives at the T-augmentation construction recipe for a string, i.e., at a T-prescription for the corresponding T-code set. The parsing algorithm used to obtain this T-prescription is called "T-decomposition".

3 T-decomposition

Suppose that, for a given string x and a letter a from the alphabet S, we want to find the T-code set for which xa is one of the longest codewords. Consider the following algorithm:

1. Set $m = 0$.
2. Decode xa as a string of codewords from $S^{(k_1,k_2,\ldots,k_m)}_{(p_1,p_2,\ldots,p_m)}$.
3. If xa decoded into a single codeword from $S^{(k_1,k_2,\ldots,k_m)}_{(p_1,p_2,\ldots,p_m)}$, set $n = m$ and finish.
4. Otherwise, set the T-prefix p_{m+1} to be the second-to-last codeword in the decoding over $S^{(k_1,k_2,\ldots,k_m)}_{(p_1,p_2,\ldots,p_m)}$.
5. Count the number of adjacent copies of p_{m+1} that immediately precede the second-to-last codeword. Add 1 to this number, and define it to be the T-expansion parameter k_{m+1}.
6. T-augment with p_{m+1} and k_{m+1}.
7. Increment m by 1 and goto step 2 above.

Example: Let $x = 011000101010$ and $a = 0$, and let $xa = 0110001010100$ be the longest codeword in some T-code set. Decoding xa over $S = \{0, 1\}$, we

obtain the following codeword boundaries indicated by dots:

$$xa = 0.1.1.0.0.0.1.0.1.0.1.0.0.$$

From these, we identify $p_1 = 0$ and $k_1 = 1$. Decoding xa over $S^{(1)}_{(0)}$ we obtain

$$xa = 01.1.00.01.01.01.00.$$

i.e., $p_2 = 01$ and $k_2 = 3$. Hence, decoding over $S^{(1,3)}_{(0,01)}$, we get

$$xa = 011.00.01010100.$$

such that $p_3 = 00$, $k_3 = 1$, and subsequently we obtain $p_4 = 011$ with $k_4 = 1$. The reader may wish to verify that $xa = 0110001010100$ is indeed one of the longest codewords of $S^{(1,3,1,1)}_{(0,01,00,011)}$.

One can also regard this mechanism as a succesive elimination of lower-level codeword boundaries, as shown in the following graphic for a different string:

S	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1)}_{(1)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1)}_{(1,10)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1,3)}_{(1,10,0)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1,3,1)}_{(1,10,0,11)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1,3,1,1)}_{(1,10,0,11,1010)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1,3,1,1,1)}_{(1,10,0,11,1010,1010011)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1
$S^{(1,1,3,1,1,1,2)}_{(1,10,0,11,1010,1010011,100)}$	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1

A more detailed treatment of T-decomposition using the current notation may be found in [11]. As mentioned above, the original proof may be found in [2, 8].

The reader may have noticed that the T-decomposition algorithm described above has a time complexity of $O(|x|^2)$. In a more recent result [17], Yang and the author have shown that the above algorithm can been simplified to run in both $O(|x| \log |x|)$ time and space. The key to this improvement is the insight that only codeword boundaries following the respective p_{m+1} need to be removed in each decoding pass. By adding the decoded codewords to a special

data structure with intertwined doubly-linked lists, it is possible to line up the codewords like "beads on a thread". One thread represents the string as such and the other threads each represent a series of identical codewords in the string. One then only follows the "thread" for the respective T-prefix in each decoding pass, skipping all other codewords in the process.

4 T-complexity

When Lempel and Ziv [1] proposed their production complexity, they recognised that the number of parsing steps would give a meaningful measure of string complexity.

Titchener pursued a similar thought and proposed a "T-complexity" measure $C_T(xa)$ as follows [10, 7, 9, 13]:

$$C_T(xa) = \sum_{i=1}^{n} \log_2(k_i + 1) \tag{3}$$

where the k_i are the T-expansion parameters found in the decomposition of xa. The units of C_T are effective T-augmentation steps, or *taugs*. Note that if all $k_i = 1$, we have $C_T(xa) = n$.

Note further that C_T is invariant under change of T-prescription, and that it a "physical" interpretation. $2^{C_T(xa)}$ equals the number of internal nodes in the decoding tree for the T-code set defined by xa, and $C_T(xa)$ this denotes the number of bits required to address each of these nodes. It is thus a measure for the state space required by a T-code decoder for the set.

The C_T measure was published in [7] and has since been discussed in several other papers by Titchener [9, 10, 13] and in a paper by Titchener, Fenwick, and Chen [12]. A further paper by Ebeling, Steuer, and Titchener [14] shows experimentally that T-complexity is closely related to the Kolmogorov-Sinai entropies arising from bipartition of the logistic map. This establishes a strong link to Shannon entropy. More recent results [19] compare various entropy estimators in the same context as Ebeling, Steuer, and Titchener's paper. Here, T-complexity-based estimates outperform those obtained from Shannon's n-gram entropy and show about the same accuracy as estimates based on the Lempel-Ziv production complexity.

5 Bounds on the T-complexity

Titchener further considered the question which strings would deliver maximal/minimal T-complexity for a given length.

For a given string length $L = xa$, minimal T-complexity is delivered by strings that contain $L - 1$ copies of a single letter, repeated over their entire length, followed by an arbitrary letter. If L is the length of the string xa, then we have a single step and $C_T(xa) = \log_2 L$.

Maximal T-complexity for some L seems to be attained by strings that satisfy both of the following two conditions:

1. A T-prescription can be found for the string such that all T-expansion parameters in the T-prescription are equal to one, and
2. all of the T-prefixes in that T-prescription are shorter than the shortest codeword from the T-code set that the string represents.

This ensures a maximum number of T-augmentations for a given length of string. Note that these two conditions together may be sufficient but are certainly not necessary: If we calculate the T-complexity of all strings of length L, then there there must obviously always be at least one string among them whose T-complexity is maximal for the given L and $\#S$. However, there are certain L for which no strings exist that satisfy the two conditions above (e.g., for a binary alphabet, strings with $L = 3$ and $L = 9$ that satisfy the conditions exist, but no such strings exist for $L = 6$). To get to $L = 6$ with T-expansion parameters of 1, one would need to T-augment either:

1. twice with a T-prefix of length 1 followed by a T-prefix of length 3, which leaves three codewords of length 2 in the set and violates the second condition, or
2. once with a T-prefix of length 1 followed by two T-prefixes of length 2, which leaves a codeword of length 1 and also violates the second condition.

The construction rule for strings with maximal T-complexity implies that there must be an upper bound for C_T for a given alphabet. Titchener found by experimentation that the logarithmic integral $li(L \ln \#S) = \int_0^{L \ln \#S} \frac{dq}{\ln q}$ seems to provide such an upper bound, asymptotically. Moreover, he found that the maximal T-complexity appears to converge rapidly towards this bound for strings that are only a few dozen letters long. A proof of this theorem has yet to be given — it is currently based largely on experimental evidence, with the exception of a partial result recently proven in [16].

This latter work, by Titchener, Gulliver, Nicolescu, Staiger and the author in [16], points at the possibility that the upper bound may be given by a series expansion where each term contains a logarithmic integral and a coefficient, the non-zero coefficients are given by the prime factors of L. The higher order terms seem to have a tendency to cancel out, yielding the asymptotic bound.

Experimental evidence also indicates the following:

- If all strings of a given length L are analyzed, they show on average a high but sub-maximal C_T. This is easily explained in that "random" strings, during their production in a random process, are predominantly extended by short T-prefixes rather than long ones, and longer T-prefixes occur only occasionally. Most of the strings of a given length L have T-complexities that fall within a very narrow band. As L increases, the distribution of C_T values for strings of length L seems to become increasingly peaked and the bulk of the distribution *seems to drop away from the maximum value of* C_T *for that length*. Note that this result has a caveat attached to it: Since it is not computationally feasible to check C_T for all strings of length L unless L is small, any "random" sample of strings of length L is necessarily subject to any "random sample generator" bias.

- Except for very small L, most T-expansion parameters in strings of length L are equal to 1.

- Strings that would generally be regarded as being non-random (e.g., representations of rational numbers) fall outside of this peak. Irrational numbers such as π, $\sqrt{2}$ etc. or strings produced by natural random processes such as radioactive decay seem to fall inside the band.

6 T-information and T-entropy

For practical purposes, i.e., comparisons between strings of different length and their substrings, a nonlinear concave function such as the logarithmic integral is a bit unwieldy.

One can argue that a string with maximal C_T that is T-augmented a number of times, each time with one of the – respectively – shortest T-prefixes available, has information added to it at a high and – approximately – constant rate over its length.

Applying the inverse logarithmic integral to C_T thus results in "linear-looking" curves for both strings with maximal C_T and for strings that fall within the narrow band mentioned above.

Titchener recognized this [13] and thus defined an information measure I_T as:

$$I_T = \mathrm{li}^{-1}(C_T) \tag{4}$$

where li^{-1} is the inverse logarithmic integral.

He further defined a T-entropy as $H_T = \Delta I_T / \Delta L$, i.e., the rate of change of I_T along the string. $\bar{H}_T = I_T / L$ denotes the "average T-entropy".

7 Other observations

Other observations indicate that there is a link between T-codes and the theory of necklaces and irreducible polynomials [20].

8 Applications

T-complexity and its derived measures are useful in entropy estimation, similarity measurement and event detection as well as other classification problem. They are currently being used in network event detection [18] as well as in a medical classification application.

9 Conclusions

The T-complexity definition seems reasonable given a similar approach to string complexity by Lempel and Ziv [1]. From this, the derivation of T-information and T-entropy also seem to be reasonable steps to take.

Experimental evidence suggests that they produce "meaningful" results. For example, the results by Ebeling, Steuer, and Titchener show that T-entropy and the Kolmogorov-Sinai entropy seem to be closely related in nonlinear (symbolic) dynamics [14].

References

[1] A. Lempel and J. Ziv: *On the Complexity of Finite Sequences*. IEEE Trans. Inform. Theory", 22(1), January 1976, pp. 75-81.

[2] R. Nicolescu: *Uniqueness Theorems for T-Codes*. Technical Report. Tamaki Report Series no.9, The University of Auckland, 1995.

[3] M. R. Titchener: *Generalized T-codes: an Extended Construction Algorithm for Self-Synchronizing Variable-Length Codes*, IEE Proceedings – Computers and Digital Techniques, 143(3), June 1996, pp. 122-128.

[4] U. Guenther: *Data Compression and Serial Communication with Generalized T-Codes*, Journal of Universal Computer Science, V. 2, N 11, 1996, pp. 769-795. `http://www.iicm.edu/jucs_2_11`

[5] U. Guenther, P. Hertling, R. Nicolescu, and M. R. Titchener: *Representing Variable-Length Codes in Fixed-Length T-Depletion Format in Encoders and Decoders*, CDMTCS Research

Report no.44, Centre of Discrete Mathematics and Theoretical Computer Science, The University of Auckland, August 1997. http://www.cs.auckland.ac.nz/CDMTCS/researchreports/044ulrich.pdf.

[6] U. Guenther, P. Hertling, R. Nicolescu, and M. R. Titchener: *Representing Variable-Length Codes in Fixed-Length T-Depletion Format in Encoders and Decoders*, Journal of Universal Computer Science, 3(11), November 1997, pp. 1207–1225. http://www.iicm.edu/jucs_3_11.

[7] M. R. Titchener: *A Deterministic Theory of Complexity, Information and Entropy*, *IEEE Information Theory Workshop*, February 1998, San Diego.

[8] R. Nicolescu and M. R. Titchener, *Uniqueness Theorems for T-Codes*, Romanian Journal of Information Science and Technology, 1(3), March 1998, pp. 243–258.

[9] M. R. Titchener, *A novel deterministic approach to evaluating the entropy of language texts*, *Third International Conference on Information Theoretic Approaches to Logic, Language and Computation*, June 16-19, 1998, Hsitou, Taiwan.

[10] M. R. Titchener, *Deterministic computation of string complexity, information and entropy*, *International Symposium on Information Theory*, August 16-21, 1998, MIT, Boston.

[11] U. Guenther: *Robust Source Coding with Generalized T-Codes*. PhD Thesis, The University of Auckland, 1998. http://www.tcs.auckland.ac.nz/~ulrich/phd.ps.gz

[12] M. R. Titchener, P. M. Fenwick, and M. C. Chen: *Towards a Calibrated Corpus for Compression Testing*, Data Compression Conference, DCC-99, Snowbird, Utah, March 1999.

[13] M. R. Titchener: *A measure of Information*, IEEE Data Compression Conference, Snowbird, Utah, March 2000.

[14] W. Ebeling, R. Steuer, and M. R. Titchener: *Partition-Based Entropies of Deterministic and Stochastic Maps*, *Stochastics and Dynamics*, 1(1), p. 45., March 2001.

[15] U. Guenther: T-Complexity and T-Information Theory – an Executive Summary, CDMTCS Research Report no.149, Centre of Discrete Mathematics and Theoretical Computer Science, The University of Auckland, February 2001. Note: Author's surname changed to "Speidel" after marriage in late 2002. http://www.cs.auckland.ac.nz/CDMTCS/researchreports/149ulrich.pdf.

[16] M. R. Titchener and A. Gulliver and R. Nicolescu and U. Speidel and L. Staiger: *Deterministic Complexity and Entropy*, *Fundamenta Informaticae*, v. 64(1-4), 443-461, 2005

[17] Jia Yang, Ulrich Speidel: *A T-Decomposition Algorithm with $O(n \log n)$ Time and Space Complexity*. Proceedings of the IEEE International Symposium on Information Theory, 4-9 September 2005, Adelaide, pp. 23–27.

[18] R. Eimann, U. Speidel, N. Brownlee: *A T-Entropy Analysis of the Slammer Worm Outbreak*, Proceedings of the 8th Asia-Pacific Network Operations and Management Symposium (APNOMS), 27-30 September 2005, Okinawa, Japan, pp. 434–445.

[19] U. Speidel, M. Titchener, J. Yang: *How well do practical information measures estimate the Shannon entropy?*. P120, Proceedings of the 5th International Symposium on Communication Systems and Digital Signal Processing (CSNDSP) 2006, 19-21 July 2006, Patras, Greece.

[20] A. Gulliver and U. Speidel: On T-Codes and Necklaces, CDMTCS Research Report no.290, Centre of Discrete Mathematics and Theoretical Computer Science, The University of Auckland, October 2006. `http://www.cs.auckland.ac.nz/CDMTCS/researchreports/290ulrich.pdf`.