GE Global Research

Kolmogorov Complexity Estimation and Analysis

S.C. Evans, J.E. Hershey and G. Saulnier

2002GRC177, October 2002
Class 1

Technical Information Series

GE Global Research

Technical Report Abstract Page

Title	Kolmogorov Complexity Estimation and Analysis		
Author(s)	S.C. Evans J.E. Hershey G. Saulnier*	**Phone**	(518)387-7014 8*833-7014
Component	Information and Decision Technologies, Niskayuna		
Report Number	2002GRC177	**Date**	October 2002
Number of Pages	6	**Class**	1

Key Words Source Coding; Lempel Ziv compression; Kolmogorov Complexity

Methods for discerning and measuring Kolmogorov Complexity are discussed and their relationships explored. A computationally efficient method of using Lempel Ziv 78 Universal compression algorithm to estimate complexity is introduced.

Manuscript received June 26, 2002

*RPI - Rensselaer Polytechnic Institute, Troy, NY

Submitted to Sixth World Conference on Systemics, Cybernetics and Informatics, July 14-18 2002, for session on Complexity Theory and its applications to Systems, Networks and Information Assurance

Kolmogorov Complexity Estimation and Analysis

Scott C. Evans, John E. Hershey and Gary Saulnier

Abstract—**Methods for discerning and measuring Kolmogorov Complexity are discussed and their relationships explored. A computationally efficient method of using Lempel Ziv 78 Universal compression algorithm to estimate complexity is introduced.**

I. Introduction

Kolmogorov Complexity is a fundamental measure of information with growing applications and importance [2], [4]. Estimation of Kolmogorov Complexity is key to objective information system monitoring and analysis. References [7], [2]-[4] contain many applications of Kolmogorov Complexity; also see [1] for background on this subject. All applications of Kolmogorov Complexity are limited due to its incomputable nature and are impacted by improvements or innovations in the ability to estimate Kolmogorov Complexity well.

In this paper we review a generic method for estimating complexity – the Lempel-Ziv 78 (LZ78) [11] universal compression algorithm, discuss its limitations in estimating complexity, and derive a computationally efficient method of using this algorithm to estimate complexity. We then develop two measures for the estimation of complexity – power spectral density based estimation and expected time of sequence production. We discuss the relationships between these methods of estimation and other estimators. Additionally we introduce a third parameter related to complexity – SPAN. Relationships between these measures of complexity are compared and discussed, and their relationships to compression-based estimates of Kolmogorov Complexity are explored.

II. Kolmogorov Complexity

A. Background

Kolmogorov Complexity is a measure of descriptive complexity contained in an object. It refers to the minimum length of a program such that a universal computer can generate a specific sequence. A good introduction to Kolmogorov Complexity is contained in [1] with a solid treatment in [7]. Kolmogorov Complexity is related to Shannon entropy, in that the expected value of K(x) for a random sequence is approximately the entropy of the source distribution for the process generating the sequence. However, Kolmogorov Complexity differs from entropy in that it relates to the specific string being considered rather than the source distribution. Kolmogorov Complexity can be described as follows, where φ represents a universal computer, p represents a program, and x represents a string:

$$K_{\varphi}(x) = \left\{ \min_{\varphi(p)=x} l(p) \right\}.$$

Random strings have rather high Kolmogorov Complexity – on the order of their length, as patterns cannot be discerned to reduce the size of a program generating such a string. On the other hand, strings with a large amount of structure have fairly low complexity. Universal computers can be equated through programs of constant length, thus a mapping can be made between universal computers of different types. The Kolmogorov Complexity of a given string on two computers differs by known or determinable constants. The Kolmogorov Complexity K(y|x) of a string y, given string x as input is described by the equation below:

$$K_{\varphi}(y \mid x) = \left\{ \begin{array}{c} \min_{\varphi(p,x)=y} l(p) \\ \infty, \text{ if there is no } p \text{ such that } \varphi(p,x) = y \end{array} \right\},$$

where l(p) represents program length p and φ is a particular universal computer under consideration. Thus, knowledge or input of a string x may reduce the complexity or program size necessary to produce a new string y.

The major difficulty with Kolmogorov Complexity is that you can't compute it. Any program that produces a given string is an upper bound on the Kolmogorov Complexity for this string, but you can't compute the lower bound.

III. Complexity Estimation

A. Empirical Entropy

Entropy is calculated from the source distribution producing a given string [10]. When the source distribution is not known,

The authors are with GE Global Research, Niskayuna, NY and RPI in Troy, NY. Lockheed Martin Systems Integration Owego, NY, funded this work, technically transitioning ideas developed under DARPA Information Assurance and Fault Tolerant Networks Projects contract F30602-01-C-0182 and managed by the Air Force Research Laboratory (AFRL) Information Directorate.

calculation of entropy is not possible until a distribution is measured. This is essentially the same problem as estimating the Kolmogorov Complexity of a given string in that we are trying to find the smallest set (or probability density function) from which a string is a typical element.

Empirical Entropy is entropy measured from the data itself. Considering the case of binary sequences, first order empirical entropy compares the number of ones and zeros and develops a probability density function. Second order empirical entropy notes the frequency of occurrence of pairs 01, 11, 10, 00. Third and following order empirical entropies develop probability density functions based on larger length patterns.

Empirical entropy as a measure of complexity can be extremely inaccurate. For example, the string 101010101010..., N times has maximal first order empirical entropy since it has an equal number of ones and zeros, yet intuitively this string is extremely simple in descriptive complexity. Still, even crude complexity estimation such as this has been shown to be useful in classifying data and behavior (see for example [6]).

B. Lempel Ziv 78 for Complexity Estimation

LZ78 has been used as an estimator for complexity for various applications, including DNA sequence analysis [9] as well as information security [3]. Kolmogorov Complexity is the ultimate compression bound for a given finite string, thus a natural choice for estimation of complexity is the class of universal compression techniques. In [11], Lempel and Ziv define a measure of complexity for finite sequences rooted in the ability to produce these sequences from simple copy operations. The LZ78 universal compression algorithm harnesses this principle to yield a universal compression algorithm that can approach the entropy of an infinite sequence produced by an ergodic source. However, as discussed in [5], any universal sequential code will fall short for some individual sequences.

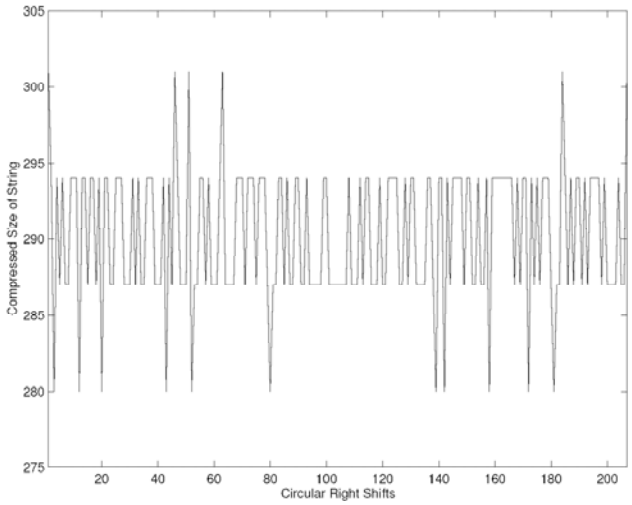

Figure 1: LZ78 Compression of Circular Right Shifts of a String. Simple shifts of a string result in significant variation in the strings compression.

For example, consider the binary string created by concatenating the first 52 unique binary strings together (from the sequence 0, 1, 00, 01, 10, 11, 000 …) is compressed using LZ78. This string of length 218 bits can be encoded using LZ78 in 315 bits – a net loss in compression. However, as shown in Figure 1, simple circular right shifts changes the compressibility of this string dramatically – a variation of up to 28 bits. Since a circular right is a very simple operation that can be implemented in a Turing machine, the Kolmogorov complexity of a string undergoing such a shift should change minimally. Thus, better estimators than LZ78 must be found to truly harness complexity as a usable parameter.

C. Improved LZ78 Complexity Estimation

Despite the difficulties discussed above, LZ78 is among the more accessible universal complexity estimators. However, complexity estimation using LZ78 usually amounts to performing the entire compression process and comparing inverse compression ratios as a measure of complexity. In fact, the simple Lempel Ziv partition contains enough data to estimate complexity without performing the entire compression encoding process.

Central to the LZ78 algorithm is the partitioning scheme introduced by Ziv and Lempel in [8]. The LZ78 algorithm partitions a string into prefixes that it hasn't seen before, forming a codebook that will (given a long enough string with enough repetition) enable long strings to be encoded with small indexes. Consider an example to illustrate how this algorithm works: LZ partitioning of the string:

101101001001101001001110100100110010

is performed by inserting commas each time a sub-string that has not yet been identified is seen. The following partition results:

1,0,11,01,00,10,011,010,0100,111,01001,001,100,010.

This can be represented by the binary tree shown in Figure 2.

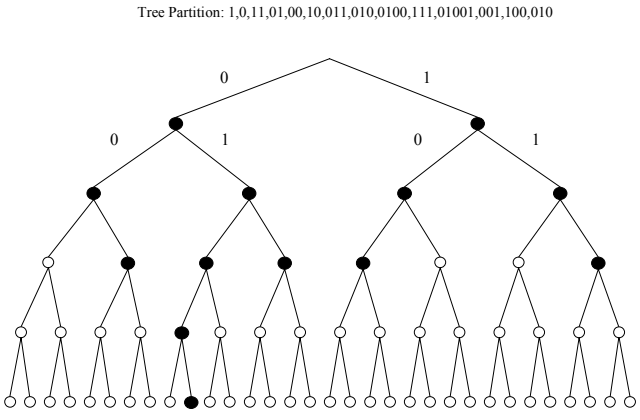

Figure 2: LZ78 binary tree representation of the partition for the binary string: 101101001001101001001110100100110010. Nodes contained in the partition are colored in black.

The nodes marked in black of the five level tree shown are nodes contained in the LZ78 partition of the example string. Nodes that are not filled in indicate code words or phrases that are not contained in the LZ78 partition. Each node or phrase occurs exactly once in the string with the exception of the last phrase which may be a repeat of a previously seen node. Good compression (low complexity estimation) results when the LZ78 partition contains a deep, sparse tree, while poor compression (high complexity estimation) results from strings that are less deep and more completely populated at each level

Maximum compression of LZ78 is achieved if all code words are children of the same branch, for example, the string:

11010110111011010110011011000

partitioned as 1,10,101,1011,10110,101100,1011000 will be highly compressed by LZ78. However, the string

10101101001001011101110000001

= 1,0,10,11,01,00,100, 101, 110, 111, 000, 001

will not be compressed by LZ78. The binary trees corresponding to these cases are shown in Figure 3.

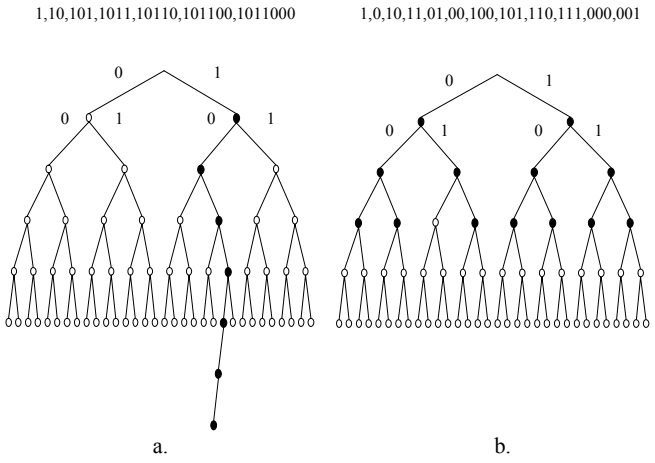

Figure 3. LZ78 partition for a: highly compressible string and b. string not compressed by LZ78. Nodes contained in the partition are colored in black.

Since the performance of LZ78 will be determined by the partition, by concentrating exclusively on the tree partition aspects of the algorithm we can achieve better efficiency when using LZ78 to estimate complexity. The essential metric is the number of phrases in the partition.

The minimum number of sub-strings (commas) in an LZ78 partition is the number M such that each sub string is one bit longer than the previous sub string:

$$\sum_{m=1}^{M} m = L = \frac{M(M+1)}{2} = \frac{M^2}{2} + \frac{M}{2}$$

Solving this quadratic equation and taking the positive solution for M we have:

$$M^2 + M - 2L = 0 \Rightarrow M = \frac{-1 + \sqrt{1+8L)}}{2}$$

For strings of any substantial length, the constant terms become insignificant and a good estimate of the lower bound results from ignoring the additive constant terms:

$$M \geq \sqrt{2L}$$

Since we know the minimal number of phrases a string of length L can have, we can normalize the number of phrases in the LZ78 partition based on this minimum for use in defining a normalized complexity estimator. We define a the metric *C* as an estimator of complexity using the LZ78 partition given a string of length L bits and an LZ78 partition of M phrases:

$$C \equiv \frac{M}{\sqrt{2L}}$$

This metric enables use of the LZ78 partitioning algorithm to estimate complexity, normalized by length, providing an estimator similar to compression ratio, but without the need for the overhead to actually complete the LZ78 compression. Under this metric complexity strings a and b from Figure 3 would have a complexity estimate metrics shown below.

$$C(a) = \frac{7}{\sqrt{2(28)}} = .93, \quad C(b) = \frac{13}{\sqrt{2(28)}} = 1.73$$

Thus the complexity estimation metric for string b is almost twice as high as that for string a. This metric has the desirable properties of LZ78 based complexity estimation without the need for completing the entire compression process as well as a normalized by length estimator with intuitively pleasing scale – higher numbers represent higher complexity.

IV. Alternate Approaches

A binary sequence of n-bits probably has little meaning if considered "in a vacuum" so to speak. Rather, meaning attaches when the sequence is viewed in a context of genesis. As discussed previously, one such framework is that of Kolmogorov complexity. What this framework requires is the least complex Turing machine that could produce the sequence. This framework is certainly a fundamental and intellectually satisfying framework as it provides a metric of complexity that requires no reference outside of itself. The computation, or assessment of complexity, is, however, impossible to perform for general n.

Therefore, we are motivated to look at other, more utilitarian, probes of complexity in the hope that they will

serve adequately and, in some limit, be related to the Kolmogorov complexity assessment.

A. SPAN

The concept of SPAN relates to other forms of complexity estimation. In their groundbreaking paper leading to development of ubiquitous compression algorithms Lempel and Ziv relate complexity to the ability to produce subsequent words in a string from previous words through simple copy operations with modification only of a single, last bit [8]. The SPAN concept relates to producing subsequent string components from the string's history but considers more intricate operations than simple copy operations.

Turing machines are infinite state machines since a Turing memory tape has unlimited storage. Finite state machines are, of course, bounded by the limitations of the physical and there are a number of paradigms of sequence generators in this machine class. The popularity of these paradigms is directly tied to their ease of implementation and their contribution to making important technical operations happen with little hardware and cost. One popular paradigm is the generation of sequences from a seed sequence. One probe of a sequence can thus be to determine the extent, or span, of the seed. The span is easily calculated and, to some extent, seems intuitively intertwined with sequence complexity.

Specifically, let us denote the zero/one n-bit binary sequence as

$$s_1, s_2, \ldots, s_n$$

We say the sequence has span=k if and only if k is the smallest natural number for which there exists a Boolean function, f, such that

$$s_i = f(s_{i-1}, \ldots, s_{i-k}) \quad , \quad i = k+1, \ldots, n$$

Note that the span is extremely sensitive to slight changes in a bit stream.

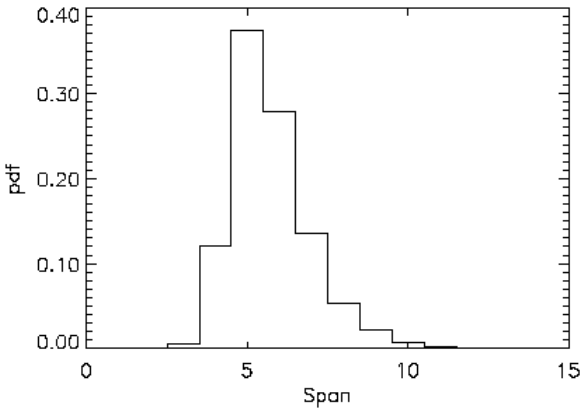

Figure 4: Probability density function as a function of SPAN for 15 bit sequences.

The span is thus a brittle measure. Also, the span is not invariant to sequence reversal. For example, the span of the sequence

10000

is 1 but the span of the reversed sequence

00001

is 4.

Figure 4 is the probability density function (pdf) of the spans computed for a large set of 15-bit sequences generated (pseudo) randomly.

B. Waiting Time

A binary sequence has to come from somewhere. Moving from the deterministic genesis viewpoint to that of probabilistic genesis, we suggest that another dimension of consideration is the determination of the mean waiting time that a balanced binary Bernoulli source would take to first produce a particular n-bit sequence. Using results developed in [1] we calculate the mean waiting time, T, for the n-bit sequence (1) by evaluating T(2) where

$$T(x) = \sum_{r=1}^{n} e_r x^r$$

and

$$e_r = \begin{cases} 1 & \text{if } s_1 s_2 \ldots s_r = s_{n-r+1} \ldots s_{n-r+2} \ldots s_n \\ 0 & \text{otherwise} \end{cases}$$

C. Power Spectral Density

1) PSI

As previously discussed, due to its non-computable nature, estimates of K(x) are difficult. Numerous techniques for estimating K(x) are discussed in [4]. The task of estimating K(x) is related to the task of assessing string structure. A new primitive approach to this related issue is introduced based on the power spectral density of a string's auto-correlation. This approach highlights the ability to gain knowledge of K(x) without any higher knowledge about the system producing string x or the meaning of the information.

Recognizing that the complexity of a binary string may be defined in many ways. A useful complexity measure may be related to properties of the string's non-cyclic auto-correlation. Specifically, given an n-bit binary string, S, where:

$$S = \{s(i)\}, 0 \le i < n ,$$

and

$$s(i) \in \{\pm 1\} \ \forall i .$$

Define the non-cyclic auto-correlation, R, as:

$$R = \{r(i)\}, 0 \le i < n$$

where

$$r(i) = \sum_{j=0}^{n-i-1} s(j)s(i+j) \quad .$$

From R, calculate the sequence's non-negative power spectral density, Φ_i, by multiplying the Fourier transform of R by its conjugate. The measure for binary string complexity that is formed is denoted by Ψ and is defined as

$$\Psi = \frac{1}{norm.\ factor} \sum_i \Phi_i \log \Phi_i \quad .$$

The motivation to this approach is found in the rich and venerable field of synchronization sequence design. Sequences that have an auto-correlation whose side-lobes are of very low magnitude provide good defense against ambiguity in time localization. Such an auto-correlation function will approximate a "thumbtack" and its Fourier transform will approximate that of band-limited white noise.

The authors of this paper expect that Ψ will be of utility in assessing complexity as it relates to the compressibility of a binary string. To begin the testing of this hypothesis, strings are generated from the Markov process diagrammed in Figure 5. A series of binary sequences of 8000 bits were generated, each for different values of p. Ψ was computed for each of these strings and also packed into 1000-kilobyte files. These were subjected to the UNIX compress routine. The Inverse Compression Ratio (ICR) was computed which is the size of the compressed file normalized to its uncompressed size, 1000 kilobytes in these cases.

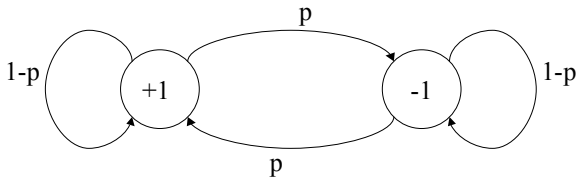

Figure 5. Markov model for string generation.

The hypothesis is that Ψ and the ICR should vary in a similar manner and that Ψ might be a useful measure of sequence compressibility and hence complexity. The graph in Figure 6 below seems to endorse this hypothesis and further research is motivated.

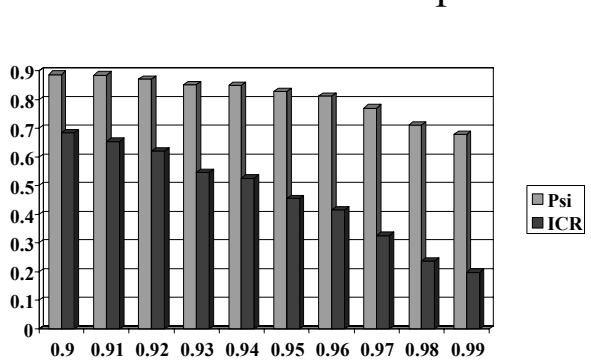

Figure 6. Variation of Psi and ICR with p.

As the above results illustrate, the parameters of sequence auto-correlation power spectral density and compressibility are related and follow similar trends. These fundamental metrics are possible candidates for measuring trend of increase or decrease in K(x). However, also illustrated by these results (the unequal rate of change between the two metrics) are the loose bounds within which estimates of K(x) are related. Other methods of estimating K(x) are described in [4]. In the next section we introduce a method for attacking the issue of loose bounds in order to make complexity metrics useful for the purposes of assessing and providing information assurance

D. Examples

We present six 15-bit sequences and compute the values of the three probes for each of them. The results are displayed in Table 1. The sequences (a)-(f) are:

(a) the all zero sequence
(b) a sequence built from a simple basic pattern
(c) – (e)
(f) an m-sequence, a phase of the sequence produced by the primitive trinomial $x^4 + x^3 + 1$.

ID	Sequence	Span	Wait	Ψ
a	000000000000000	1	65534	0.30283
b	010101010101010	1	43690	0.15560
c	010010111001111	5	32768	0.43152
d	010011100010001	5	32772	0.41345
e	011110100110110	5	32770	0.41419
f	100010011010111	4	32770	0.38764

Table 1: Probe Values for 6 15-bit Binary Sequences

SPAN properties can be understood by the relationships between tree nodes. Figure 7 illustrates the binary tree representation of the LZ78 partition for strings a and f from Table 1. As expected from previous analysis the tree for the simple sequence a is narrow and deep, with only 5 phrases in

the partition, while the tree for the more complex sequence f is shallow with 7 phrases.

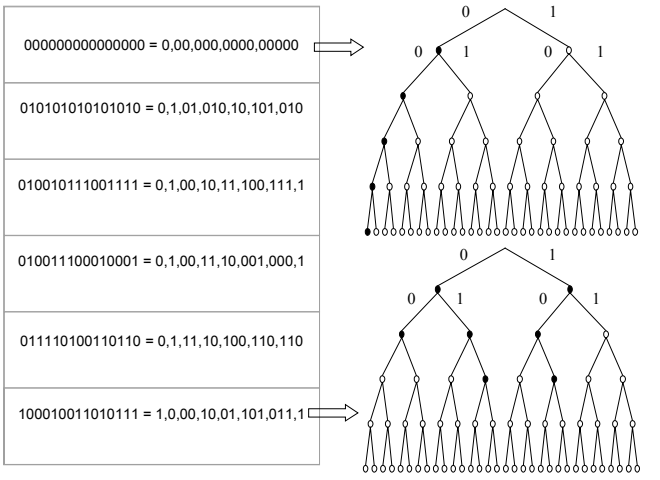

Figure 7: LZ78 Partitions for 15 bit Binary Sequences

Additional properties of binary trees can provide further illumination and potential for complexity analysis. Some nodes in the binary tree contribute to other nodes or branches, while certain nodes are disjoint, or are not part of the production sequence of other nodes. Disjoint nodes have maximal SPAN (SPAN equal to their length). The amount that the SPAN of a particular phrase deviates from the maximum represents the potential for a particular node to be used to construct other phrases

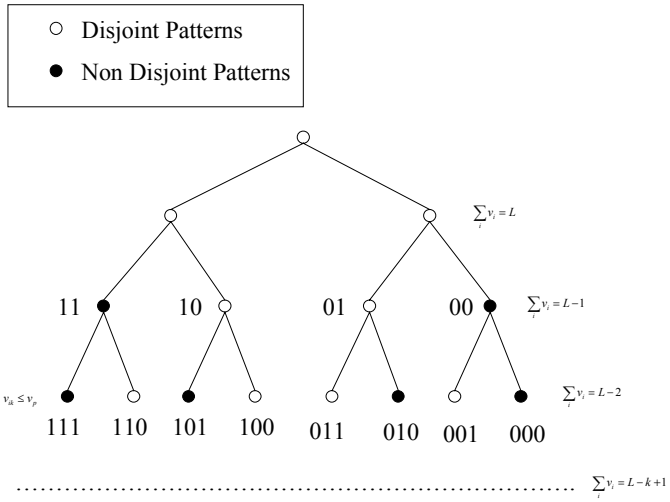

Figure 8: Identification of disjoint and non-disjoint patterns or nodes for a binary tree of depth 3.

Figure 8 illustrates these inherent relationships for a three level binary tree. The pattern 111 is non-disjoint, in that it can be produced from a simple copy operation from the preceding string 11, while the string 110 is disjoint and has maximum span. These and other conserved relationships inherent in binary trees motivate the possibility of generating alternate partitions of a binary string from an LZ78 partition. A complexity estimation algorithm based on finding the optimal binary tree composition has been developed [4]. This algorithm, entitled the Optimal Symbol Compression Ratio algorithm (OSCR), will be partnered with LZ78 partitioning and binary tree SPAN and other relationships to develop improved complexity estimators in future work.

V. Conclusion

Use of LZ78 to estimate complexity can be performed more computationally efficiently by concentrating on the size of the partition. An upper bound on the number of phrases in this partition has been derived and a normalized LZ78 based complexity estimation metric defined. Alternate concepts for evaluating and estimating complexity have been developed and their relationships to one another explored. Future work will expand these results and consider optimal string partitions as a method for achieving improved estimation of Kolmogorov Complexity.

References

[1] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley, NY, 1991

[2] Evans, S, Bush, S. F., and Hershey, J., “Information Assurance through Kolmogorov Complexity”, DARPA Information Survivability Conference & Exposition II, 2001, Proceedings Vol 2, pp 322-331.

[3] Evans, S. C. Barnett, B. “Conservation of Complexity for Network Security”, accepted for publication in MILCOM 2002.

[4] Evans, S. C, and Bush, S. F. ``Symbol Compression Ratio for String Compression and Estimation of Kolmogorov Complexity,'' GE Research Technical Report 2001CRD159, November 2001.

[5] Kieffer, J. C. and Yang, E. “Sequential Codes, Lossless Compression of Individual Sequences, and Kolmogorov Complexity,” IEEE Transactions of Information Theory, Vol 42, 1 January 1996

[6] Kulkarni, A. B., Bush, S. F. and Evans, S. C. ``Detecting Distributed Denial-of-Service Attacks using Kolmogorov Complexity Metrics,'' GE Research Technical Report 2001CRD176. December, 2001.

[7] Li, M. and Vitányi, P. An Introduction to Kolmogorov Complexity and Its Applications, Springer, NY 1997

[8] Lempel, A. and Ziv, J. “One The Complexity of Finite Sequences,” IEEE Transactions of Information Theory, Vol IT 22, January 1976, pp 75-81.

[9] Powell, D. R, Dowe, D. L., Allison, L. and Dix, T. I. "Discovering simple DNA sequences by compression", Monash University Technical Report, monash.edu.au.S

[10] Shannon, C. E. ``A mathematical Thoery of Communication,'' Bell Systems Technical Journal, Vol 27, pp. 379-423, 623-656, October, 1948.

[11] Ziv, J. and Lempel, A. “Compression of individual sequences via variable length coding,” IEEE Trans. Inform. Theory, vol IT-24, pp. 530-536, 1978.

S.C. Evans
J.E. Hershey
G. Saulnier

Kolmogorov Complexity Estimation and Analysis

2002GRC177
October 2002