On Information Conservation and Algorithmic Complexity

Pieter Adriaans[1], Amos Golan[2]

[1] ILLC, IvI, FNWI
University of Amsterdam,
Science Park 107
1098 XG Amsterdam,
The Netherlands
[2] Info-Metrics Institute
American University, 4400 Massachusetts Avenue, NW
Washington, DC 20016-8029
P.W.Adriaans@uva.nl, agolan@american.edu

Abstract. In this paper we investigate standard prefix-free Kolmogorov complexity in the context of Zellner's information conservation principle (ICP). We show that prefix-free Kolmogorov complexity K is not efficient in this sense. We introduce *Information Conserving Algorithmic complexity* K^*, defined on a kernel space of random strings. We prove that this version is efficient in a weak sense. We prove that universal Turing machines do not conserve information in a strong sense, but we conjecture the existence of at least one such machine $\mathfrak{U}$. Because K^* conserves information, the prefix-free aspect of the program code can be ignored as an internal aspect of the representation. This leads to a variant of the universal distribution m^* using a uniform density estimator ξ_U for the distribution of the random strings. This distribution is shown to be smoother than the standard Solomonoff distribution. Of course ξ_U is unknown, but since it 'absorbs' our uncertainty about the distribution m^* uniformly, it leads to a theory that can be applied to small data sets without the intervention of a $O(1)$ factor.

keywords: Kolmogorov Complexity, Zellner's Information Conservation Principle, Universal Distribution

1 Introduction

1.1 Some simple examples

Two well known information measures are: Shannon Information, $I(x) = -\log p_x$, and Kolmogorov complexity $I(x)$ is *the length of the shortest program p that produces x on a computer*. Of these two Shannon is purely extensive: if we combine two mutually independent messages x and y in to a new message we have $I(x + y) = -\log p_x - \log p_y$. Kolmogorov complexity gives more problems.

The shortest program for the string $x = 11111111111111111111111111111111$ is something like "`for i=1 to 30 print:1`" which has 20 characters. The string $y = 1111111111111111111$ still has the same complexity in this approximation "`for i=1 to 20 print:1`". Even the concatenation of x and y has this approximated complexity: "`for i=1 to 50 print:1`". Thus we have the counter intuitive $I(x) = I(y) = I(x \otimes y)$, where $\otimes$ is the concatenation sign. We get no new information when we concatenate x to y. This is partly due to the fact that the strings are not independent, but even for independent strings we have counterintuitive effects. Let $a = 00111110101$ and $b = 110010$ be independent random strings and let $c = a \otimes b$. The shortest programs are "`print:00111110101110010`", "`print:00111110101`" and "`print:110010`". Here we have $I(a \otimes b) < I(a) + I(b)$. On the other hand take the string: $f = 000110001100011000100011$ which is 25 characters long. The shortest program is "`i=1 to 5 print:00011`", which is 20 characters long. so this is an effective compression. Now form f' by removing one bit from the end. Now the shortest program becomes:

"`i=1 to 4 print:00011;print:0001`"

This program has 31 characters which is longer then f', making f' effectively a random string. The fact that we can transform a substantially compressible string in to a random one by removing one bit is quite striking. Of course these effects are largely dependent on the programming language of choice. These observations motivate the following research question: is it possible to develop a variant of Kolmogorov complexity that has the same efficiency as Shannon information and, if not, how close can we get?

1.2 Zellner's information conservation principle and algorithmic complexity

Zellner ([13], [14]) showed that Bayes' rule is efficient in the sense that the amount of input information equals the amount of output information. [4] showed that the maximum entropy principle and the general maximum entropy principle are both 100% efficient. Applying Shannon's criterion for optimal code length $I(x) = -\log \mathcal{P}(x)$ to Bayes' rule we derive:

$$\mathcal{P}(x|y) = \frac{\mathcal{P}(x)\mathcal{P}(y|x)}{\mathcal{P}(y)}$$

$$-\log(\mathcal{P}(x|y)) = -\log(\frac{\mathcal{P}(x)\mathcal{P}(y|x)}{\mathcal{P}(y)})$$

$$\log(\mathcal{P}(y)) + \log(\mathcal{P}(x|y)) = \log(\mathcal{P}(x)) + \log(\mathcal{P}(y|x))$$

$$I(y) + I(x|y) = I(x) + I(y|x)$$

When we use Bayes' rule or the maximum entropy principle, we do not create or lose information in the process of reasoning. Clearly this is a desirable quality.

When we replace Shannon information with Kolmogorov complexity and interpret the expression $I(x)$ as the shortest program that produces the string x on a universal Turing machine U this information efficiency is lost. Strings interpreted as programs differ considerably from probabilities. The class of valid programs is not closed under concatenation, so it is not possible to define information efficiency directly. The standard solution in Kolmogorov complexity is to define a variant that only accepts programs in prefix-free format. This gives not only the benefit of well-defined concatenation of programs, but, via Kraft's inequality, also guarantees the existence of a probability distribution over the set of programs. In this way a universal distribution m for binary strings can be defined. For an extensive discussion of these issues we refer to [11]. Still there are problems with this approach:

- The prefix-free variant of Kolmogorov complexity is not efficient.
- Kraft's inequality is very unrestrictive. Any prefix-free code will do: there is no guarantee that the resulting probability distribution is scientifically useful. It may favor some classes of strings over others without any clear motivation. In fact there is no guarantee that a prefix-free code per se is a good solution for the construction of a universal distribution if we want our result to have any bearing on relatively small binary objects.

There is a third point that is relevant in this context. Zellner's criterion specifies a relation between input and output information. This implies an inherent typing issue that is not well recognized in the literature on Kolmogorov complexity. Let x^* be the shortest program that generates x on U such that $U(x^*) = x$. In this relation x^* is a program, i.e. input, and x is output. x^* is by definition also uncompressible and thus random. But the shortest program that produces x^* as output is something like $U(print : "x^*") = x^*$. In this context x^* is output and $"x^*"$ is the *name* of the object. Note that al this implies that the expression $print : "x^*"$ is a random string, although it looks quite regular.

In the following we start by defining efficient information spaces. We show that standard prefix-free Kolmogorov complexity K is not efficient. We describe the concept of Information Conserving Algorithmic complexity K^* and prove that it is efficient in a weak sense. We prove that universal Turing machines do not conserve information in a strong sense, but we conjecture the existence of at least one such machine $\mathfrak{U}$.

1.3 Efficient Information Spaces

Following Zellner, we define the notion of an efficient information space that satisfies the Information Conservation Principle (ICP). Suppose we have a countable set $\mathcal{I}$ with a general information function $I : \mathcal{I} \to \mathcal{R}^+$ that assigns a positive real number to every element in $\mathcal{I}$. Such a set is called an *Efficient Information Space* if there exists a *Conditional Information Operation* $I(x|y)$ for which the following conservation law holds for all $x, y \in \mathcal{I}$:

$$I(y) + I(x|y) = I(x) + I(y|x) \tag{1}$$

We call such a space efficient because we can always interpret conditional information to be associated with Bayesian probabilities, as was shown in the previous paragraph. In some cases it is useful to assume the existence of a bottom element $\bot$ and/or a top element $\top$. If a space has a bottom $\bot$ element as well as a top $\top$ element it is *closed*. No element contains any conditional information with respect to the top or itself:

$$I(x|\top) = I(\bot|x) = I(x|x) = I(\bot) = 0 \tag{2}$$

A space without a top element is open. A closed information space is finite if:

$$I(\top) = r \tag{3}$$

where r is a real number. The bottom never contains conditional information:

$$I(x|\bot) = I(x) \tag{4}$$

The main results in this paper are independent of those requirements.

Definition 1. *For any two elements $H, D \in \mathcal{I}$ we say that H is a* hypothesis *for D if:*

1. *$I(H|D) = 0$ i.e. $I(H) + I(D|H) = I(D)$, the hypothesis can not be rejected based on the information in D.*
2. *$I(H) = I(D) - I(D|H) > 0$, it contains information that is not supported by D.*

These principles are not very restrictive. Very simple spaces already observe ICP in this sense:

Example 1. Let $\mathcal{I}$ be a powerset P of a countable set U, $\bot = \emptyset$, $\top = U$, $I(X) = |X|$ is the cardinality of X and $I(X|Y) = X - Y$, i.e. the set of elements of X that are not contained in Y. We have: $I(Y) + I(X|Y) = I(X) + I(Y|X)$ which is $|Y| + |X - Y| = |X| + |Y - X|$. This is clearly efficient. This corresponds to a uniform distribution over U. Since the space is efficient, we can, when we know the distribution, reconstruct any unknown subset of U on the basis of its conditional information with other sets.

Example 2. Let $\mathcal{I} = \{0,1\}^*$ be the set of binary strings (i.e. a free monoid with concatenation), $\bot = \varepsilon$, there is no top, $I(x) = |x|$ is the length of x, $I(x|y) = \{|a| : (x = ab, y = bc) \vee (y = cb, x = ba)\}$ i.e. the length of x after subtracting a suffix or prefix that overlaps with y. Clearly we have: $I(Y) + I(X|Y) = I(X) + I(Y|X)$. The corresponding distribution is $\mathcal{P}(x) = 2^{-|x|}$. Since the space is efficient, we can, when we know the distribution, reconstruct any unknown string on the basis of its conditional information with other strings.

This shows the use of efficient information spaces. In these systems we are certain that the probability distributions represent the information and vice versa.

2 Prefix-free Kolmogorov complexity

K is the prefix-free Kolmogorov complexity of a binary string. It is defined as:

Definition 2. $K(x|y) = \min_i\{|\bar{i}| : U(\bar{i}y) = x\}$

i.e. the shortest self-delimiting index of a Turing machine T_i that produces x on input y, where $i \in \{1, 2, ...\}$ and $y \in \{0, 1\}^*$. Here $|\bar{i}|$ is the length of a self-delimiting code of an index and U is a universal Turing machine that runs program y after interpreting $\bar{i}$. The length of $|\bar{i}|$ is limited for practical purposes by $n + 2\log n + 1$, where $n = |i|$. The reason to use $\bar{i}$ lies in the fact that it allows us to separate the concatenation $\bar{i}p$ into its constituent parts, i and p. Here i is the index of a Turing machine which can be seen as capturing the *regular* (structural [12], meaningful [1], model [6], effective [9]) part of the string x, and p describes the input for the machine, i.e. the *irregular* part, e.g. errors in the model, noise and other missing information. We define:

Definition 3. $K(x) = K(x|\varepsilon)$

Where ε is the empty string. This is in fact a one-part code optimization variant K_1 of Kolmogorov complexity that forces all complexity of the information to be stored in the index of the Turing machine. Sometimes it is useful to distinguish a two-part code optimization variant:

Definition 4. $K_2(x) = \min_{i,p}\{|\bar{i}| + |p| : U(\bar{i}p) = x\}$

This version balances the information over an index i and a program p for x.

2.1 Prefix-free Kolmogorov complexity does not define an efficient Information space

It is obvious that prefix-free Kolmogorov complexity does conserve information in a *weak sense*. When we have x and we have an optimal p such that $U(\overline{p}x) = y$ then we can compute y and use this to define (although not compute) an optimal q such that $U(\overline{q}y) = x$. Since this relation is symmetrical we always have $K(y)+K(x|y) = K(x)+K(y|x)$. Here we show that prefix-free Kolmogorov complexity can never conserve information efficiently.

Suppose that we have the following definitions:

$$\mathcal{I} = \{0, 1\}^* \tag{5}$$

$$\forall x, y \in \mathcal{I} : I(x) = K(x), I(x|y) = K(x|y) \tag{6}$$

Lemma 1. *Prefix-free Kolmogorov complexity does not conserve information efficiently.*

Proof: Suppose that the conservation law 1 holds:

$$K(y) + K(x|y) = K(x) + K(y|x)$$

Then we have:

$$K(y|\varepsilon) + K(x|y) = K(x|\varepsilon) + K(y|x)$$
$$\min_i\{|\bar{i}| : U(\bar{i}\varepsilon) = y\} + \min_j\{|\bar{j}| : U(\bar{j}y) = x\}$$
$$=$$
$$\min_k\{|\bar{k}| : U(\bar{k}\varepsilon) = y\} + \min_l\{|\bar{l}| : U(\bar{l}x) = y\}$$

which implies for some optimal i, j, k and l:

$$(|i| + 2\log|i|) + (|j| + 2\log|j|) = (|k| + 2\log|k|) + (|l| + 2\log|l|)$$

This equation has no general solution, unless $|i| = |k|$ or $|i| = |l|$ and this is obviously too restrictive to satisfy ICP. □

This seems counter-intuitive. After all Kolmogorov complexity is an information measure. The expression $K(y) + K(x|y) = K(x) + K(y|x)$ can be read from left to right as: when we have the code for y and the code for x-given-y, then we can construct x, and from that we must be able to construct y-given-x. This is even more embarrassing, when one takes into account that the prefix-free version was introduced to allow for the definition of a universal probability distribution via Kraft's inequality. The downside of this solution apparently is that ICP is violated. One might object that K only introduces exponentially small errors that in most computations have minor consequences. The fact remains, that K is an information measure with the undesirable quality that it *produces* information when we use it. This additional information proliferates uncontrollably in our computations and this makes K virtually useless when analyzing small data sets.

2.2 Information Conserving Algorithmic Complexity

Kolmogorov complexity is non-constructive. It can be approximated by means of dovetailing computations on a universal Turing machine [11]. So let's make this presupposition explicit: it might help to understand the internals of the concept of information conserving algorithmic complexity, when we suppose that we have access to a demon with infinite computational power in a black box. This demon performs calculations of infinite length without showing us the result, but uses them internally. One such thing this demon could do is calculate the optimal representation for a binary object before using it. This allows us to restrict the input of the computations to a kernel domain of random strings.

Definition 5. *Let x be a binary string and let*

$$U : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

be a universal Turing machine. The optimal code *for x is the shortest code that generates x on U:*

$$x^* = \min_i\{|i| : U(\bar{i}\varepsilon) = x\}$$

Note that x^* is a random string and that it is defined independently from any notion of Kolmogorov complexity. If x is random then x is its own optimal code: $x = x^*$. We define the kernel set of random binary strings with respect to U:

Definition 6.

$$\texttt{RANDOM}_U = \{x | x \in \{1,0\}^*, x =_U x^*\}$$

We can restrict the domain of U to this set:

$$U^* : \texttt{RANDOM}_U \times \texttt{RANDOM}_U \to \{0,1\}^*$$

Note that, since all binary objects have an optimally compressed representation this does not restrict the expressiveness of U. This kernel is the basis for the Information Conserving Algorithmic complexity:

Definition 7. *The* Information Conserving Algorithmic Complexity

$$K^* : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}^+$$

of a string x conditional to a string y with respect to a restricted general Turing machine U^* is defined as:

$$K^*_U(x|y) = \min_i \{|i| : U^*(\bar{i}y^*) = x\}$$

Here $\mathbb{N}^+$ indicates the set of positive natural numbers. When possible we will leave the subscript for U out. Again, non-conditional complexity is defined as:

Definition 8. $K^*(x) = K^*(x|\varepsilon)$

The information conserving algorithmic complexity differs at two points from the standard prefix-free Kolmogorov complexity:

- It uses the self-delimiting code $\bar{i}$ internally but reports $|i|$ to the outside world. Thus we lose the possibility of defining a probability distribution via Kraft's inequality.
- It uses the optimal code y^* instead of y itself. All inputs from the domain $\{0,1\}^*$ are compressed to corresponding elements $\texttt{RANDOM}_U$

Definition 9. *$y\dagger x$ is the optimal code that outputs y given x. We have $K^*(y|x) = |y \dagger x|$*

Lemma 2. $K^*(x \dagger y | y, x) = 0$

Proof: $x \dagger y$ is defined given x and y, although it is uncomputable. It can be approximated from below. □

Theorem 1. *Information Conserving Algorithmic complexity does conserve information in the weak sense efficiently.*

Proof: Suppose that the conservation law 1 holds:

$$K^*(y) + K^*(x|y) = K^*(x) + K^*(y|x)$$

Then we have:

$$K^*(y|\varepsilon) + K^*(x|y) = K^*(x|\varepsilon) + K^*(y|x)$$

$$\min_i\{|i| : U(\bar{i}\varepsilon) = y\} + \min_j\{|j| : U(\bar{j}y^*) = x\}$$

$$=$$

$$\min_k\{|k| : U(\bar{k}\varepsilon) = y\} + \min_l\{|l| : U(\bar{l}x^*) = y\}$$

$$|y^*| + \min_j\{|j| : U(\bar{j}y^*) = x\} = |x^*| + \min_l\{|l| : U(\bar{l}x^*) = y\}$$

which implies for the optimal codes j and l:

$$|y^*| + |j| = |x^*| + |l|$$

Now take $j = x \dagger y$ and $l = y \dagger x$. This satisfies ICP, but we have to prove that this equality really is full-filled. I do this for one half of the equality: Given y^* and $x \dagger y$, x^* can be computed. Given x^* and y^* is $y \dagger x$ defined according to lemma 2. □

This proof shows that the conservation law explains the conceptual relation between input and output of computational processes and conditional information. Given the strings x and y we have two ways to specify the exact amount of information in the combination of the two: $K^*(x) + K^*(y \dagger x)$ and $K^*(y) + K^*(x \dagger y)$. Given an infinite amount of time we can always reconstruct one representation out of the other. This is the weak algorithmic interpretation of ICP. To see how this works in practice lets look at an example:

Example 3. Let x and y be two large, and thus highly compressible factorials. Their optimal code is $\overline{p}f_x$ and $\overline{p}f_y$, where p is the factorial program and f_x and f_y are indexes. So we have:

$$|\overline{p}f_y| + \min_j\{|j| : U(\bar{j}\overline{p}f_y) = x\} = |\overline{p}f_x| + \min_l\{|l| : U(\bar{l}\overline{p}f_x) = y\}$$

Given the structure of x and y it is clear that $j = x \dagger y = \overline{q}f_x$ and $l = y \dagger x = \overline{q}f_y$, where q is a program that instructs U to first read a chunk of data and then a program and forget about the rest, such that: $U(\overline{\overline{q}f_x}\overline{p}f_y) = x$ and $U(\overline{\overline{q}f_y}\overline{p}f_x) = y$. Thus we have:

$$|\overline{p}f_y| + |\overline{q}f_x| = |\overline{p}f_x| + |\overline{q}f_y|$$

A next question to ask is whether we can also prove efficient conservation of information in the strong sense. The following lemma shows that this is not the case:

Lemma 3. *Universal Turing machines do not conserve information efficiently in the strong sense.*

Proof: Suppose we have a universal Turing machine U for which $|y^*| + |x \dagger y| = |x^*| + |y \dagger x|$. Now define a new universal Turing machine U' that can call $|x \dagger y|$ as a subroutine with an s such that $|s| < |x \dagger y|$ and $U(\bar{s}y) = x$. This causes U' to violate ICP. □

This observation is a more general variant of the so-called *nickname problem* ([9], [8], [2]). Obviously an information conserving universal machine would be a nice theoretical tool but we have not been able to prove or disprove that it exists, so we formulate the conjecture:

Conjecture 1. A completely efficient information conserving universal Turing machine $\mathfrak{U}$ exists.

The machine $\mathfrak{U}$ would define completely transparent exchange between data and programs. An optimal computational description x^* of x would also define a semantics for x.

In the following we suppose that PREFIX is an infinite set of prefix-free strings. The shift from prefix-free to information conserving algorithmic complexity has some interesting consequences.

- In prefix-free Kolmogorov complexity the universal machine

  $$U : \texttt{PREFIX} \times \{1,0\}^* \to \{1,0\}^*$$

  is defined on heterogeneous domains: the indexes are prefix-free strings, the programs arbitrary strings. The range is the full set of strings. In information conserving algorithmic complexity the universal machine

  $$U^* : \texttt{RANDOM}_U \times \texttt{RANDOM}_U \to \{1,0\}^*$$

  is defined on homogeneous domains of random strings. The range again is the full set of strings.
- In information conserving algorithmic complexity, all inputs have maximal entropy. In principle the index and the program can be interchanged. Thus K^* can be seen as running on the kernel of programs $\texttt{RANDOM}_U$.

3 Discussion

K^* improves K in two fundamental ways:

- It uses an infinite search process to determine the information in the definition of the Turing machine and maps this information onto the set of random strings. In this way we can be sure that accidental compressibility of the indexes does not affect our measurement of the complexity. This in contrast with K that uses the recursive function of the universal Turing machine to 'parse' the indexes and will mistakenly interpret many compressible objects for random strings.

– It does not use the prefix-free representation of K, which implies that indexes for the same information are in general shorter, and that an index for i can never be longer than $\log i$. Although the prefix free programs ensure convergence to distribution m, it is merely a practical solution without any inherent motivation. It adds an additional structure to the complexity measure that in most cases is inconsistent with the phenomena we try to explain. The self-delimiting code that is usually adopted to make the programs prefix-free particularly affects complexity measurements of small sets.

3.1 The universal distribution m^* associated with K^*

Note that all elements in $\texttt{RANDOM}_U$ have maximal entropy. This gives them a natural probability when interpreted as sequences of messages:

$$\forall x \in \texttt{RANDOM}_U : \mathcal{P}(x) = 2^{-|x|}$$

When $x \in \{1,0\}^*$ then our estimate for the probability of x is given by the:

Definition 10. *The* information conserving universal distribution*:* $m^*(x) = 2^{-(1+\xi_U)K^*(x)}$

where $0 < \xi_U < 1$ is a factor that ensures convergence as well as conservation of information (ICP). Note that, since the density of compressible strings is zero in the limit, definition 10 does not converge if $\xi_U = 0$. It can be interpreted as containing information about the density distribution of $\texttt{RANDOM}_U$ in $\{1,0\}^*$ with respect to U. When we compare the information conserving universal distribution with the standard Solomonoff distribution m that is specified via Levin's coding theorem $-\log m(x) = K(x) + O(1)$ ([11]) we observe that for any $0 < \xi_U < 1$ and for any random string x of size large enough:

$$K(x) + O(1) = |x| + 2\log x + O(1) < (1 + \xi_U)|x| = (1 + \xi_U)K^*(x)$$

Thus m dominates m^* on all but a finite number of points.

3.2 Influence of the computational power of U

Suppose we analyze the complexity of factorials. We have two universal Turing machines at our disposal: $U_!$ contains a software routine `fac` with $\texttt{fac}^* = k$ that implements factorials $x!$ and U does not. Let $n = x!$ be a factorial number and let $l =$ "`x!`" be the code length of the program `x!`: the computation in U is described by: $U(\overline{\texttt{fac}}\text{"}\texttt{x!}\text{"}) = n$ which according to

$$K_2(x) = \min_{i,p}\{|\bar{i}| + |p| : U(\bar{i}p) = x\}$$

gives $K_2^U(n) = k+l$. By the same reasoning we have the computation $U(\bar{\varepsilon}x!) = n$ for $U_!$ which gives: $K_2^{U_!} = l + 1$. In other words U needs code that is an additive factor k longer than $U_!$ to code factorials. This implies that U will not recognize

factorials of size $k + l < \log n$ as such, while $U_!$ already starts to compress factorials at $l+1 < \log n$. Generalizing this insight we can observe that increasing the computational power of a universal Turing machine results in the reduction of the frequency of random strings in the initial segment of the set of binary strings and consequently in the resulting universal distribution. This is in line with ideas of Solomonoff where the cumulative learning experience is interpreted as a process of incremental update of the universal distribution [10]. Note that if conjecture 1 is true there would exist a completely efficient universal distribution $\mathfrak{m}$ that codes all possible mathematical knowledge as efficiently as possible. This would be the potential holy grail of any learning effort.

4 Conclusion

This last observation shows that the ξ_U used in the definition of m^* deserves further study. Does it have an asymptotic universal value? What is the quantitative effect of an update of U? Another question that deserves further study is the interpretation of a kernel of random strings as an information conserving lattice.

Out research question at the start of this paper was: is it possible to develop a variant of Kolmogorov complexity that has the same efficiency as Shannon information and, if not, how close can we get? The preliminary answers are:

- We can define a variant of Kolmogorov complexity that conserves information efficiently in the weak sense.
- Universal Turing machines do not conserve information efficiently in the strong sense.
- We conjecture the existence of at least one universal Turing machine that does conserve information efficiently in the strong sense.

Apart from the fact that efficient information processing is in itself desirable, the theoretical constructions presented here give us a better perspective to map concepts of Kolmogorov complexity to other information processing rules (conditional entropies, mutual information).

An example: suppose we want to compare two objects x and y, and we know that they are different but contain the same amount of information. We have $m(x) = 2^{-K(x)+O(1)}$ and $m^*(x) = 2^{-(1+\xi_U)K^*(x)}$. The $O(1)$ term in the equation for m has no sensible interpretation apart from being a factor that regulates the 'resolution' of our theory. It is not even sure that the factor will be the same for y and x giving $K(x) = K(y) + O(1)$ as our best guess. On the other hand the term ξ_U in m^* has a sensible interpretation, which gives $m^*(x) = m^*(y) = 2^{-(1+\xi_U)K^*(x)}$ even if the value of ξ_U is unknown. This implies that K^* can be applied to small data sets even though it is essentially asymptotic.

5 Acknowledgements

This research was partly supported by the Info-Metrics Institute of the American University in Washington, the Commit project and the ILLC and IvI of the

University of Amsterdam and a Templeton Foundations Science and Significance of Complexity Grant supporting The Atlas of Complexity Project.

Bibliography

[1] Adriaans , P.W. , (2009) Between Order and Chaos: The Quest for Meaningful Information, Theory of Computing Systems, Volume 45 , Issue 4 (July 2009), Special Issue: Computation and Logic in the Real World; Guest Editors: S. Barry Cooper, Elvira Mayordomo and Andrea Sorbi, 650-674.

[2] Adriaans, P.W. (2012) Facticity as the amount of self-descriptive information in a data set, http://arxiv.org/abs/1203.2245.

[3] Cover T.M. and Thomas, J.A. (2006), Elements of Information theory, Wiley.

[4] A. Golan (2008) Information and Entropy Econometrics - a Review and Synthesis Volume 2, Foundations and Trends in Econometrics, Now Publishers Inc.

[5] Golan, A. George G. Judge, Douglas Miller (1996) Maximum entropy econometrics: robust estimation with limited data, Volume 16, Series in financial economics and quantitative analysis, Wiley.

[6] Grünwald, P.D. (2007), The Minimum Description Length Principle. MIT Press.

[7] Hopcroft, J. E., Motwani, R., Ullman, J. D. (2001), Introduction to Automata Theory, Languages, and Computation Second Edition. Addison-Wesley.

[8] Foley, D.K. (2010) Notes on Bayesian inference and effective complexity, unpublished manuscript.

[9] Gell-Mann M. and S. Lloyd (2003) Effective complexity. In Murray Gell-Mann and Constantino Tsallis, eds.

[10] Solomonoff, R.J. (1997), The Discovery of Algorithmic Probability, Journal of Computer and System Sciences, vol. 55, nr. 1, 73-88.

[11] Li M., Vitányi P.M.B. (2008), An Introduction to Kolmogorov Complexity and Its Applications, 3rd ed., Springer-Verlag, New York.

[12] Vereshchagin, N.K.Vitányi P.M.B. (2004) Kolmogorov's structure functions and model selection, IEEE Transactions on Information Theory, vol. 50, nr. 12, 3265–3290.

[13] Zellner, A. (1988), Optimal information processing and Bayes' theorem, American Statistician 42, 278-284.

[14] Zellner, A. (2002), Information processing and Bayesian analysis, Journal of Econometrics 107, 41-50.