OPEN ACCESS

BIO-complexity

Research Article

Expected Algorithmic Specified Complexity

David Nemati and Eric Holloway*

Electrical and Computer Engineering Department, Baylor University, Waco, TX, USA

Abstract

Algorithmic specified complexity (ASC) is an information metric that measures meaning in an event, based on a chance hypothesis and a context. We prove expectation of ASC with regard to the chance hypothesis is always negative, and empirically apply our finding. We then use this result to prove expected ASC is conserved under stochastic processing, and that complexity for individual events is conserved under deterministic and stochastic processing.

Cite As: Nemati D, Holloway E (2019) Expected Algorithmic Specified Complexity. Bio-Complexity 2019 (2):1-10. **doi:**10.5048/BIO-C.2019.2.

Editor: William Basener

Received: January 22, 2019; **Accepted:** October 6, 2019; **Published:** October 24, 2019

Notes: A *Critique* of this paper, when available, will be assigned **doi:**10.5048/BIO-C.2019.2.c.

*Email: Eric_Holloway@baylor.edu

1. INTRODUCTION

Algorithmic Specified Complexity (ASC) is a metric of meaning. The metric guarantees that false positives are unlikely to result from chance due to the improbability of ASC result [1]. However, since there is still a possibility of false positives, this raises the question of whether the expected ASC is positive. If the expected ASC is positive, then by the law of large numbers [2] we can expect to observe average positive ASC with enough samples. Consequently, if the expected ASC is positive, the occurrence of ASC is not a surprising event in need of an explanation, and thus doesn't necessarily signify meaning. In this paper, we show the expected amount of ASC from a random variable is always negative. We also use this result to prove conservation bounds on complexity under deterministic and stochastic processing.

2. MEASURING MEANING

To understand the motivation for measuring meaning, we consider the two metrics of standard information theory: Shannon surprisal and algorithmic information. We will see that neither Shannon surprisal nor algorithmic information can measure meaningful information. Instead, we need a hybrid of the two, known as a randomness deficiency, that is measured in reference to an external context.

2.1 Shannon Surprisal

The main reason that Shannon surprisal does not measure meaning is that Shannon specifically chose not to consider meaning in his theory of communication [3].

The Shannon surprisal [3] of an event x with probability p is measured with

$$I(x) := -\log_2 p(x). \tag{1}$$

From (1) we can see why Shannon surprisal does not measure meaning. Consider a fair coin flip, where each side has

$$p(\text{side}) = 0.5. \tag{2}$$

According to Shannon's surprisal, merely knowing that the coin landed heads counts as 1 bit of surprisal for a fair coin, since

$$\begin{aligned} I(\text{side}) &= -\log_2 p(\text{side}) \\ &= -\log_2 0.5 \\ &= 1. \end{aligned} \tag{3}$$

However, unless the coin flip is tied to an event of importance, it is of little interest to us and is meaningless. Meaningful sequences of symbols point to something beyond the sequence, in the same way that a signpost is meaningful because it refers to a destination.

To see the difference between Shannon's concept of surprisal and meaningful information, consider the phrase:

`METHINKS_IT_IS_LIKE_A_WEASEL`

a string of randomly generated letters of the same length:

`HNKRCYDO_BIIIEDWPBURW_OIMIBT`

and the repetitive sequence:

`AAAAAAAAAAAAAAAAAAAAAAAAAAAA`

The random sequence of letters has a higher surprisal than the phrase, but it is meaningless. The repetitve sequence has zero surprisal, and is also meaningless. The phrase is meaningful because there is both an amount of surprisal (range of possible letters) and the arrangement of letters refers to an external context. So, we see that Shannon's surprisal can bound, but not capture meaningful information. Consequently, a number of mathematicians have attempted to better measure meaningful information.

2.2 Algorithmic Information

Always understood in a deterministic setting, algorithmic information theory [4], derived independently by Kolmogorov [5], Chaitin [6] and Solmonoff [7], is based on the insight that the smallest program that will generate a random bitstring is not shorter than the bitstring itself. Consequently, bitstrings with generating programs that are shorter than the bitstring will tend to be orderly and thus could have meaningful structure.

Algorithmic information is the length of the shortest program y^* that generates a bitstring x when executed on a computer $(\mathcal{U})$,[1]

$$K(x) := \min_{y|\mathcal{U}(y)=x} |y|. \tag{4}$$

Chaitin [8] refers to

$$y^* := \arg \min_{y|\mathcal{U}(y)=x} |y| \tag{5}$$

as the *elegant program* for x. These elegant programs, unfortunately, cannot be calculated due to the halting problem.

Algorithmic information can be measured in relation to a context C. Access to context is analogous to having a library in a programming language, or being able to call other programs in the file system. These context files are not counted in the algorithmic information bit count. Conditioning algorithmic information does not cause an increase in information (beyond a constant used to distinguish contextual and context free information),

$$K(x|C) \leq K(x) + O(1). \tag{6}$$

If the context does not help shrink the elegant program length, it is simply not used.

2.3 Prefix-free Coding

We assume that all programs are written in a prefix-free [9] (or self-delimiting [4]) form. "Prefix-free" is a term in coding theory, which signifies a coding standard such that no code word in the standard begins with any other code word in the standard. This allows the codes to be "instantaneous", meaning each code word can be recognized as soon as it is read. Non-instantaneous codes, such as the Morse code, need further information read to signify when a code word ends and another begins. An example of a prefix-free free coding standard is the following set of code words, which do not begin with any other code words in the standard:

- 0
- 10
- 110
- 1110
- 11110
- 11111

In the context of prefix-free Turing machines, "prefix-free" means that no program begins with another program that halts. Algorithmic information defined with a prefix-free Turing machine has a number of useful properties. One important property we will encounter later in this paper is that prefix-free algorithmic information can be used to define a semimeasure, a key element of the paper's foundamental proof.

2.4 Randomness Deficiency

The algorithmic information of a bitstring, by itself, does not indicate whether a bitstring is meaningful or not. For example, if a bitstring has 10 bits of algorithmic information, and the bitstring is only 10 bits long, then the bitstring is random and meaningless. On the other hand, if a bitstring has 10 bits of algorithmic information, and the bitstring is 20 bits long, then it is compressible and might be meaningful. So we see just knowing the algorithmic information of a bitstring is insufficient to differentiate between meaningful and meaningless bitstrings.

To address this difficulty, Kolmogorov proposed the concept of "randomness deficiency" [4] where the algorithmic information of the bitstring is subtracted from the length of the bitstring,

$$\delta(x) := \ell(x) - K(x). \tag{7}$$

Martin-Löf demonstrated randomness deficiency is a universal test for randomness [10].

The downside of this randomness deficiency metric is that while it overcomes the problem of surprisal's inability to distinguish random and non-random sequences, it does not identify meaningful information, as it would provide a higher score for the repetitive sequence over the English text sequence in our opening example.

Levin, a student of Kolmogorov, generalized randomness deficiency [11] as

$$d(x/\mu) := -\log_2(\mu(x)) - K(x). \tag{8}$$

[1] More generally, on a universal Turing machine.

We can see that (8) is a combination of Shannon surprisal and algorithmic information.

More recently, Milosavljević derived a metric called "algorithmic significance" [12], which is similar to the randomness deficiency derived by Levin.[2] The metric differs in that it uses compression instead of algorithmic information, and thus is a computable metric. Milosavljević's metric is used to detect patterns in DNA sequences [13–15], where the randomness deficiency is measured by algorithmic compression.

These later variants of the randomness deficiency by Levin and Milosavljević are an improvement over algorithmic information in regards to measuring meaning. The variants allow us to give a higher score to the English text than to the repetitive sequence by setting $p(x) = 1$ for the repetitive sequence, which implies[3]

$$\begin{aligned} d(x) &= -\log_2 1 - K(x) \\ &= 0 - K(x) \\ &< 0. \end{aligned} \tag{9}$$

Even so, these variants of the randomness deficiency metric do not quite get us to meaningful information. The deficiency metrics prefers rare, highly regular sequences over the more complicated sorts of sequences that we consider meaningful, i.e. English text. However, if we measured algorithmic information in regards to a relevant context, such as the English language, then according to (6) the randomness deficiency will increase when encountering a sequence of English text. The metrics do not account for an independent context, so cannot identify meaning.

2.5 Algorithmic Specified Complexity

Following this insight of the need for an external context in randomness deficiency metrics, a new metric of meaningful information is algorithmic specified complexity (ASC) [16],

$$ASC(x, C, p) := I(x) - K(x|C). \tag{10}$$

Where

1. x is a bitstring generated by some stochastic process,
2. $I(x)$ is the Shannon surprisal of x, also known as the *complexity* of x, and
3. $K(x|C)$ is the conditional algorithmic information of x, also known as the *specification*.

ASC is itself a kind of "specified complexity", which in turn is an even broader generalization of randomness deficiency. Specified complexity is a more general concept since it can in theory also involve uncomputable resources, such as halting oracles [17–20]. The original definition of specified complexity is in [21] and developed in [22].[4] The concept of specified complexity is a refinement of the explanatory filter defined in [24].

The idea of specified complexity is to justify rejecting the chance hypothesis in favor of an alternate hypothesis by using an independent knowledge source to define a rejection region. As long as the knowledge source is independent of the chance distribution, it can be used to define the rejection region after the fact. This is in contrast to traditional Fisherian significance testing [25] where all hypotheses must be defined before observing the event. [5]

ASC is capable of measuring meaning by positing a context C to specify an event x. The more concisely the context describes the event, the more meaningful it is. The event must also be unlikely, that is, having high complexity $I(x)$. The complexity is calculated with regard to the chance hypothesis distribution p, which represents the hypothesis that x was generated by a random process described by p, implying any similarity to the meaningful context C is by luck. ASC has been illustrated by its application to measure meaningful information in images and cellular automata [26, 27].

ASC is probabilistic. Although a positive ASC score could signify meaning, a positive score can also be achieved by chance. The probability of achieving at least α bits of ASC by chance is at most $2^{-\alpha}$ [1]. The improbability of ASC has been generalized for any canonical specified complexity in [28], where the criterion of canonical specified complexity as possession of a *kardis* is also defined in [28].

Since the algorithmic information cannot be calculated, we depend on upper bounding the algorithmic information in ASC with a computable compression algorithm conditioned on a context C, a generalization of Milosavljević's approach in [12]. The computable form of ASC is known as observed ASC (OASC), and is a lower bound on the true ASC. Likewise, the probability of at least α bits of OASC has probability that is at most $2^{-\alpha}$.

2.6 Advantage of Contextualized Randomness Deficiency

The use of context distinguishes ASC from Levin's generalized form of randomness deficiency in (8) and Milosavljević algorithmic compression approach. The fundamentaly advantage is that the use of an independent external context allows ASC to measure whether an event refers to something beyond itself, i.e. is meaningful. Without the context, the other randomness deficiencies perhaps can tell us that an event is meaningful, but cannot identify what the meaning

[2] Thanks to Dr. Tom English for bringing attention to Milosavljević's work.

[3] It must be the case that $K(x) \geq 1$, because if the null symbol is a halting program, then since all programs start with the null symbol no other programs can be halting programs, according to the definition of prefix-free algorithmic information.

[4] Dembski anticipated the incorporation of algorithmic information into CSI in [21]. The use of algorithmic information and randomness deficiency for defining specified complexity was also recommended by Devine in [23], a few years after the derivation of ASC.

[5] Fisher recognized the need for ad hoc hypotheses, but due to not having a formalized definition of randomness, which Kolmogorov et. al later provided, his method was restricted to the *a priori* selection of hypotheses [11].

is. Thus, ASC's use of an independent context enables novel contextual specifications to be derived from problem domain knowledge, and then applied to identify meaningful patterns, such as identifying non-trivial functional patterns in the game of life [26].

3. CONSERVATION OF EXPECTED ASC

Now we turn to the paper's main result. Since there is a probability of generating positive ASC by chance, it is a question whether the expected ASC produced by a chance hypothesis can be positive. However, to prove our result regarding the expectation of ASC we will need to transform $K(x|C)$ into a semimeasure, requiring more technical background

3.1 Universal Probability

The property of $K(x)$ being prefix-free allows us to define a "universal probability". To understand how, we need to understand Chaitin's uncomputable, oracular Omega number.

The prefix-free set of halting programs follows the Kraft inequality [9]

$$\Omega := \sum_{y \in \mathcal{Y}_{halts}} 2^{-\ell(y)} < 1, \tag{11}$$

where $\mathcal{Y}_{halts}$ is the set of all halting programs. Ω is Chaitin's Omega number [9]. Omega is an oracle, because by knowing Omega we could solve the halting problem. However, since the halting problem is unsolvable, we cannot know Omega.

Now we turn to elegant programs. Since the elegant program y^* used to calculate the algorithmic information of x halts when run on the universal Turing machine, elegant programs $\mathcal{Y}_{elegant}$ are a subset of $\mathcal{Y}_{halts}$, i.e. $\mathcal{Y}_{elegant} \subset \mathcal{Y}_{halts}$. We know this to be the case since multiple halting programs of different length output the same bitstring, yet they cannot all be elegant programs. Thus, elegant programs are also prefix-free and, from the Kraft inequality,

$$\sum_{y \in \mathcal{Y}_{elegant}} 2^{-\ell(y)} < \Omega < 1. \tag{12}$$

We can define a semimeasure[6] known as "universal probability"[7] using prefix-free Kolmogorov complexity,

$$\mathbf{m}(x) := 2^{-K(x)}. \tag{13}$$

Similarly, we can express the conditional algorithmic information of the specification as a conditional form of (13),

$$\mathbf{m}_{/C}(x) := 2^{-K(x|C)}. \tag{14}$$

Since $\mathbf{m}_{/C}$ is defined on a subset of the elegant programs, according to Equation (12) $\mathbf{m}_{/C}$ will sum to less than one.

$$\sum_{x \in \mathcal{X}} \mathbf{m}_{/C}(x) = \sum_{x \in \mathcal{X}} 2^{-K(x|C)} < 1. \tag{15}$$

3.2 Conservation of Expected ASC

Using the definition of ASC in Equation (10) and the semimeasure in Equation (14), application of Jensen's inequality shows us the expected ASC is

$$\begin{aligned} E_{\mathcal{X}}[ASC(\mathcal{X}, C, p)] &= \sum_{x \in \mathcal{X}} p(x)(I(x) - K(x|C)) \\ &= \sum_{x \in \mathcal{X}} p(x) \log_2 \left(\frac{\mathbf{m}_{/C}(x)}{p(x)} \right) \\ &\leq \log_2 \left(\sum_{x \in \mathcal{X}} \frac{p(x)}{p(x)} \mathbf{m}_{/C}(x) \right) \\ &\leq \log_2 \left(\sum_{x \in \mathcal{X}} \mathbf{m}_{/C}(x) \right) \\ &< \log_2 1 = 0. \end{aligned} \tag{16}$$

This means a chance hypothesis with distribution p generates negative expected ASC. The reason why (16) is a strict inequality is due to (15), which demonstrates summing the conditional universal probability results in a value less than 1, of which the logarithm is negative.

Since OASC is a lower bound on ASC, then Equation (16) also implies,

$$E_{\mathcal{X}}[OASC(\mathcal{X}, C, p)] \leq E_{\mathcal{X}}[ASC(\mathcal{X}, C, p)] < 0. \tag{17}$$

3.3 Hypothesis Testing Application

The law of large numbers states that the average of a series of experiments will approach the mean [2]. As such, we can test Equation (16) empirically using the average of a large number of events. In the empirical case, we rely on OASC, since ASC is not calculable, so we will use the corollary Equation (17).

As an experiment we will generate bits according to distribution q and compare the results to chance hypothes p, which is not always the same as q. Both distributions are Bernoulli distributions, and a specific distribution is represented by $\mathtt{B}(\alpha)$, where α is the probability of generating a 1.

Equation (17) states that when $q = p$, then the average OASC will drop below zero. On the other hand, when $q \neq p$ the average OASC can be positive. Thus, according to the law of large numbers, after enough samples a positive average OASC indicates $q \neq p$.[8] This approach to rejecting the chance hypothesis is a form of hypothesis testing, which is also addressed in regards to specified complexity in [24], [21], [1], and [28].

We should note that these empirical results are not verifying that expected ASC is always less than zero, since the expected ASC in the following examples is calculated an-

[6] While a probability distribution sums to one, a semimeasure does not sum to one.

[7] $\mathbf{m}$ is called a "universal probability" because it multiplicatively dominates all computable probability distributions [4].

[8] Absolute certainty is impossible due to requiring an infinite number of samples to converge to the true mean.

alytically, not empirically. On the other hand, the results indicate that if we empirically observe positive average OASC then we can reject the chance hypothesis distribution p in favor of the empirical distribution q, since positive average OASC implies positive expected ASC. Of course, the converse does *not* apply: negative average OASC does not imply negative expected ASC. This is because OASC is a lower bound on ASC, so some data may have negative OASC while also having positive ASC.

The general procedure is as follows:

1. Generate 100 bitstrings from each q distribution, each bitstring 512 bits long.

2. Calculate surprisal for each bitstring using the p distribution.

3. Calculate the OASC by subtracting the compression length from the surprisal for each bitstring.

   For compression we use the Lempel-Ziv-Welch algorithm that is trained on a 1,000,000 length bitstring for each q distribution.

4. Average the OASC over the 100 bitstrings for each p and q pair.

The results of the test are shown in Table 1, where we expect the diagonal to be negative after enough trials.

As a sanity check, we also calculate the expected ASC for each tuple, which must always be greater than the average OASC, and consequently will be greater than the average OASC after enough samples.

3.3.1 Shannon Entropy and the Kullback-Liebler Divergence

Before getting into the derivation of expected ASC in this application, some more technical background is required regarding Shannon entropy. The Shannon entropy of a probability distribution p is the expected surprisal provided by that distribution over the set of events $\mathcal{X}$,

$$H(p) := \sum_{x \in \mathcal{X}} -p(x) \log_2 p(x). \tag{18}$$

When the expectation of surprisal of q is taken with regards to the distribution p the result is known as cross entropy,

$$H(p,q) := \sum_{x \in \mathcal{X}} -p(x) \log_2 q(x). \tag{19}$$

The cross entropy between p and q is never less than the entropy of p,

$$H(p,q) \geq H(p). \tag{20}$$

The Kullback-Liebler divergence is the difference between the cross entropy and entropy, and is never negative,

$$\begin{aligned} D(q||p) :=& \sum_{x \in \mathcal{X}} p(x) \log_2 \left(\frac{p(x)}{q(x)} \right) \\ =& H(p,q) - H(p) \geq 0. \end{aligned} \tag{21}$$

3.3.2 Derivation of Expected ASC

With these definitions in hand, we can analytically calculate the expected ASC by using the Kullback-Liebler divergence between q and p. This calculation is based on the following information theory identities.

1. The expected surprisal of bitstring x, generated by independently sampling $n = \ell(x)$ bits from a Bernoulli random variable $\mathtt{B}_q$ defined by q, where $\mathcal{X}$ represents the corresponding series of random variables, with surprisal calculated by p, is the cross entropy $nH(q,p)$,

$$\begin{aligned} E_q[I_p(\mathcal{X})] &= nE_q[I_p(\mathtt{B}_q)] \\ &= n \sum_{b \in \mathtt{B}_q} -q(b) \log_2 p(b) \\ &= nH(q,p). \end{aligned} \tag{22}$$

2. The expected algorithmic information of a sequence x produced by independent samples of a computable Bernoulli distribution q is $H_q(\mathcal{X})$ [4],

$$E_q[K(\mathcal{X})] \leq H_q(\mathcal{X}) + K(q) + O(1). \tag{23}$$

   The inequality in (23) is due to the fact that $H_q(\mathcal{X})$ is a lower bound on expected prefix-free coding of $x \in \mathcal{X}$, thus is a lower bound on $E_q[K(\mathcal{X})]$. With the addition of $K(q)$ the entire distribution for $x \in \mathcal{X}$ is described, taking into account a constant $O(1)$ for overhead, which means $E_q[K(\mathcal{X})]$ cannot be larger than the righthand side.

   Since the terms $K(q)$ and $O(1)$ disappear aspymptotically as n increases, and $H_q(\mathcal{X})$ is the optimal expected compression, we simplify Equation (23) to

$$E_q[K(\mathcal{X})] = H_q(\mathcal{X}) = nH(q). \tag{24}$$

3. By combining Equations (22) and (24), we can conclude that the expected ASC is

$$\begin{aligned} E_q[ASC(\mathcal{X}, C, p)] &= E_q[I_p(\mathcal{X}) - K(\mathcal{X})] \\ &= E_q[I_p(\mathcal{X})] - E_q[K(\mathcal{X})] \\ &= n(H(q,p) - H(q)) \\ &= nD(q||p), \end{aligned} \tag{25}$$

   where $D(q||p)$ is the Kullback-Liebler divergence [9] between q and p.

   Since expected ASC is greater than expected OASC,

this means

$$nD(q||p) \geq E_q[OASC(\mathcal{X}, C, p)]. \tag{26}$$

3.3.3 Results

The calculated expected ASC using Equation (25) is shown in Table 2. Based on the law of large numbers and the fact that OASC is a lower bound on ASC, we expect the difference of Table 1 subtracted from Table 2 to be positive, and when average OASC is positive then expected ASC must also be positive. The difference can be seen in Table 3, and Equation (17) is consistent with the results in all cases.

4. CONSERVATION BOUNDS

Equation (16) shows that a random variable does not have positive expected ASC. However, given that an event has a measure of ASC, is it possible to use some form of stochastic processing to increase the amount of ASC beyond that contained in the original event? For example, what if we apply a function f that always generates the same y for all $x \in \mathcal{X}$, and y is not a member of $\mathcal{X}$?

In this case, given the probability distribution over x, it is always the case that

$$\begin{aligned} p(f(x)) &= p(y) \\ &= \Pr[\mathcal{X} = y] \\ &= 0, \end{aligned} \tag{27}$$

and thus

$$\begin{aligned} I(f(x)) &= -\log_2 p(f(x)) \\ &= -\log_2 0 \\ &= \infty \end{aligned} \tag{28}$$

resulting in infinite ASC for $K(f(x)|C) < \infty$,[9]

$$\begin{aligned} ASC(f(x), P, C) &= I(f(x)) - K(f(x)|C) \\ &= -log_2 0 - K(f(x)|C) \\ &= \infty - K(f(x)|C) \\ &= \infty. \end{aligned} \tag{29}$$

This analysis suggests that ASC is not conserved.

However, Equation (29) only shows that ASC is not conserved if the chance hypothesis is invalid due to incorrect function application. To understand the conservation of expected ASC we have to look at the definition of a random variable, and identify how applying a function to the random variable affects the probability distribution. We then use Equation (16) to prove ASC is conserved under stochastic processing.

[9]This example was originally proposed by Dr. Tom English at http://theskepticalzone.com/.

4.1 Random Variables

We will examine the discrete case, since ASC is defined on discrete probability distributions. A discrete probability distribution p is a countable set of events. The set of events is called the *domain* and signified by ζ. Each event has a probability assigned

$$p := \{\epsilon \in \zeta : 0 \leq p(\epsilon) \leq 1\} \tag{30}$$

such that the sum of all values is equal to one,

$$\sum_{\epsilon \in \zeta} p(\epsilon) = 1. \tag{31}$$

For example, the distribution for the roll of a fair die has six events, each with probability $\frac{1}{6}$, and the sum of all event probabilities is 1.

A random variable $\mathcal{X}$ is a function that assigns labels to the events, i.e. giving each die face a number from 1 to 6. The set of possible labels is called the *range* and signified with Γ. Formally, the random variable $\mathcal{X}$ is

$$\mathcal{X} : \zeta \to \Gamma \tag{32}$$

The subset of the labels $L \subseteq \Gamma$ used by $\mathcal{X}$ is called the *image* of $\mathcal{X}$. Thus, a random variable can also be defined as a set of label/probability pairs, where each pair represents a labeled possible event. To continue the die example, the random variable representing the die is the set of pairs

$$\mathcal{X} := \left\{ \left(1, \frac{1}{6}\right) \left(2, \frac{1}{6}\right) \left(3, \frac{1}{6}\right) \left(4, \frac{1}{6}\right) \left(5, \frac{1}{6}\right) \left(6, \frac{1}{6}\right) \right\}. \tag{33}$$

When we talk about the probability of an element of a random variable, we are talking about a specific label, and are summing the probabilities of all events that have that label. So, when we write $p(x)$ what we mean is

$$\begin{aligned} p(x) &= \Pr[\mathcal{X} = x] \\ &= \sum_{\epsilon \in \zeta | \mathcal{X}(\epsilon) = x} p(\epsilon). \end{aligned} \tag{34}$$

We can create a new random variable from $\mathcal{X}$ by applying a transformation function to each event, essentially relabeling $\mathcal{X}$'s image. For example,

$$Y := f(\mathcal{X}) \tag{35}$$

means

$$Y := \{f(x) : x \in \mathcal{X}\}. \tag{36}$$

Table 1: Average OASC of bitstrings generated from q distribution, measured using p distribution as chance hypothesis

q ↓ p →	$B(2^{-5})$	$B(2^{-4})$	$B(2^{-3})$	$B(2^{-2})$	$B(2^{-1})$	$B(1-2^{-2})$	$B(1-2^{-3})$	$B(1-2^{-4})$	$B(1-2^{-5})$
$B(2^{-5})$	**-29**	-30	23	185	720	1419	1922	2213	2339
$B(2^{-4})$	-20	**-42**	-23	71	475	1045	1476	1731	1844
$B(2^{-3})$	15	-29	**-46**	-23	243	678	1034	1251	1351
$B(2^{-2})$	112	42	-14	**-71**	42	326	599	774	860
$B(2^{-1})$	389	287	179	19	**-67**	22	187	308	375
$B(1-2^{-2})$	878	744	584	322	37	**-69**	-13	54	102
$B(1-2^{-3})$	1373	1214	1014	672	234	-22	**-50**	-25	9
$B(1-2^{-4})$	1869	1688	1452	1039	462	71	-32	**-44**	-23
$B(1-2^{-5})$	2366	2164	1894	1413	705	185	10	-38	**-30**

Table 2: Expected ASC of bitstrings generated from q distribution, measured using p distribution as chance hypothesis

q ↓ p →	$B(2^{-5})$	$B(2^{-4})$	$B(2^{-3})$	$B(2^{-2})$	$B(2^{-1})$	$B(1-2^{-2})$	$B(1-2^{-3})$	$B(1-2^{-4})$	$B(1-2^{-5})$
$B(2^{-5})$	0	9	62	242	780	1510	1965	2229	2378
$B(2^{-4})$	7	0	19	132	536	1133	1520	1750	1883
$B(2^{-3})$	41	16	0	43	305	761	1078	1273	1388
$B(2^{-2})$	135	91	36	0	106	406	644	801	896
$B(2^{-1})$	409	339	234	97	0	97	234	339	409
$B(1-2^{-2})$	896	801	644	406	106	0	36	91	135
$B(1-2^{-3})$	1388	1273	1078	761	305	43	0	16	41
$B(1-2^{-4})$	1883	1750	1520	1133	536	132	19	0	7
$B(1-2^{-5})$	2378	2229	1965	1510	780	242	62	9	0

Table 3: Expected ASC (Table 1) minus Average OASC (Table 2)

q ↓ p →	$B(2^{-5})$	$B(2^{-4})$	$B(2^{-3})$	$B(2^{-2})$	$B(2^{-1})$	$B(1-2^{-2})$	$B(1-2^{-3})$	$B(1-2^{-4})$	$B(1-2^{-5})$
$B(2^{-5})$	29	39	39	57	60	91	43	16	39
$B(2^{-4})$	27	42	42	61	61	88	44	19	39
$B(2^{-3})$	26	45	46	66	62	83	44	22	37
$B(2^{-2})$	23	49	50	71	64	80	45	27	36
$B(2^{-1})$	20	52	55	78	67	75	48	31	34
$B(1-2^{-2})$	18	57	60	84	69	69	49	37	33
$B(1-2^{-3})$	15	59	64	89	71	65	50	41	32
$B(1-2^{-4})$	14	62	68	94	74	61	51	44	30
$B(1-2^{-5})$	12	65	71	97	75	57	52	47	30

so

$$p(y) = \Pr[Y = y] = \sum_{\epsilon \in \zeta | Y(\mathcal{X}(\epsilon)) = y} p(\epsilon). \quad (37)$$

While it seems simple, Equation (37) is important. As an example, if we changed the die labels to "A", "B", and "C", then a possible random variable for the die is

$$\left\{ \left(\mathrm{A}, \frac{1}{3}\right) \left(\mathrm{B}, \frac{1}{3}\right) \left(\mathrm{C}, \frac{1}{3}\right) \right\}. \quad (38)$$

This example demonstrates how function application will affect the surprisal of $\mathcal{X}$. The function is operating on the labels by replacing an existing label with a new label. This means the function can merge labels, i.e. it can take labels "1" and "2" and replace them with "A". But, the function *cannot* split a single label into multiple labels. So, the function cannot take the "A" label and split it back into "1" and "2".

Thus, according to Equation (37) applying a function to a random variable can only produce a new set of labels that have an equal or greater probability than the old set of labels. So, we accordingly define the surprisal of function application to be

$$I(f(x)) := -\log_2 \left(\Pr[f(\mathcal{X}) = f(x)] \right). \quad (39)$$

This means function application can only maintain or decrease surprisal,

$$I(f(x)) \leq -\log_2 p(x) = I(x). \quad (40)$$

4.2 Deterministic Conservation of Complexity

This gives us the basis to see why Equation (29) does not disprove conservation of ASC. More formally, take the original definition of ASC in (10),

$$ASC(x, C, p) := I(x) - K(x|C). \quad (41)$$

and add f to get

$$fASC(x, C, p, f) := I(f(x)) - K(f(x)|C). \quad (42)$$

Applying (40) to the observation that ASC can never exceed the surprisal, we establish the conservation of complexity for ASC,

$$fASC(x, C, p, f) < I(x). \quad (43)$$

This means that if it appears applying a function increases ASC beyond $I(x)$, such as making it infinite in (29), then the error lies in not properly transforming the random variable $\mathcal{X}$ with the function.

Finally, as a corollary, using the new label random variable defined as

$$L := f(\mathcal{X}), \quad (44)$$

the result from (16) still applies,

$$E_{\mathcal{X}}[fASC(\mathcal{X}, C, p, f)] = E_L[ASC(L, C, p)] < 0. \quad (45)$$

Additionally, since we are dealing with a random variable the improbability of ASC still applies,

$$\Pr[ASC(l, C, p) > \alpha] \leq 2^{-\alpha}. \quad (46)$$

4.3 Stochastic Conservation of Complexity

What if f is not deterministic, but instead selected from a set of functions, adding an element of randomness? In this case, the surprisal $I(f(x))$ for some $f(x)$ is potentially greater than $I(x)$ since the probability of selecting that f must also be factored into the information calculation for the event x. Thus, p is now the joint distribution over $\{x, f\}$ and $I(f(x))$ is dictated by the joint probability of f and x. In this case, $p(x)$ and $p(f)$ mean the corresponding marginal distributions.

To simplify notation, we can think of the stochastic processing as a table with the various fs across the top and the xs along the side. Each cell c_{ij} in the table contains a label l that is assigned to the joint event that both f_i and x_j occur together.

Here is a small illustration of the idea:

Table 4: Mapping $f(x)$

X ↓ F →	f_1	f_2	f_3
x_1	a	b	c
x_2	a	a	a
x_3	d	e	f

It is important to note labels can be shared between different events and functions, e.g. $c_{12} = c_{32}$ and $c_{11} = c_{12}$

We can also create a corresponding probability distribution which we'll denote $\mathbb{P}$:

Table 5: Probability Distribution for $f(x)$

X ↓ F →	f_1	f_2	f_3	marg.
x_1	0.25	0.15	0.1	0.5
x_2	0.125	0.075	0.05	0.25
x_3	0.125	0.075	0.05	0.25
marg.	0.5	0.3	0.2	1.0

The set of all labels l in the table is the image of $\mathcal{F}$, and denoted L. The probability of a label occurring is conditioned on x, since the probability of which function is applied to x is dependent on the fact that x occurs. The conditional probabilities $p(l|x)$ are calculated

$$p(l|x) = \sum_{f \in \mathcal{F} | f(x) = l} p(f|x). \quad (47)$$

For example, we can apply (47) to Table 4 and see $p(\mathbf{a}|x_2) = 1$.

The joint probability is

$$p(l,x) = p(l|x)p(x), \tag{48}$$

and the corresponding information of the joint probability for $\{l,x\}$ is

$$I(l,x) = -\log_2 p(l,x), \tag{49}$$

which can be broken apart into

$$\begin{aligned} I(l,x) &= -\log_2 p(l|x)p(x) \\ &= -\log_2 p(l|x) - \log_2 p(x) \\ &= I(l|x) + I(x). \end{aligned} \tag{50}$$

With (47) and (50) in place, and applying the fundamental result from (16), the expected ASC under stochastic processing for a particular event is

$$\begin{aligned} E_F[fASC(x,C,p,F)] &= \sum_{l\in L} p(l|x)(I(l,x) - K(l|C)) \\ &= \sum_{l\in L} p(l|x)(I(x) + I(l|x) - K(l|C)) \\ &= \sum_{l\in L} p(l|x)I(x) + \\ &\quad \sum_{l\in L} p(l|x)(I(l|x) - K(l|C)) \\ &= I(x) + \sum_{l\in L} p(l|x)\log_2\left(\frac{\mathbf{m}_{/C}(l)}{p(l|x)}\right) \\ &\leq I(x) + \log_2\left(\sum_{l\in L}\frac{p(l|x)}{p(l|x)}\mathbf{m}_{/C}(l)\right) \\ &= I(x) + \log_2\left(\sum_{l\in L}\mathbf{m}_{/C}(l)\right) \\ &< I(x) + \log_2 1 \\ &= I(x). \end{aligned} \tag{51}$$

To apply (51) to the ongoing illustration, we need to define a context $\mathbb{C}$ that maps the labels to a prefix-free coding,

Table 6: Code

l	C[l]
a	0
b	10
c	110
d	1110
e	11110
f	11111

Taking the expected fASC for the three x events in our illustration we get the following values, consistent with (51):

Table 7: Expected fASC

x	p(x)	I(x)	E_F[fASC(x, C, P, F)]
x_1	$\frac{1}{2}$	1.0	0.785
x_2	$\frac{1}{4}$	2.0	1.0
x_3	$\frac{1}{4}$	2.0	-1.01

4.4 Stochastic Conservation of Expected ASC

What if we take the expectation over both $\mathcal{F}$ and $\mathcal{X}$? The results in Table 7 suggests this expectation will be positive:

$$\frac{0.785}{2} + \frac{1.0}{4} + \frac{-1.01}{4} = 0.388. \tag{52}$$

However, in this case, we are taking the expectation with regard to the label distribution $p(l)$, which is the new chance hypothesis, and get the same sort of result as in (16),

$$E_{\mathcal{X},\mathcal{F}}[fASC(\mathcal{X},C,p,\mathcal{F})] = E_L[ASC(L,C,p)] < 0. \tag{53}$$

Additionally, since L is a random variable, then the improbability of ASC applies,

$$\Pr[fASC(x,C,p,f) \geq \alpha] = \Pr[ASC(l,C,p) \geq \alpha] \leq 2^{-\alpha}. \tag{54}$$

Doesn't this analytic result contradict our example, which appears to produce a positive expectation in (52)? The key to this apparent contradiction is in our observation in note #2 below Table 4, that labels are not only shared between cells on the same row in Table 4, but also between rows. Thus, when we perform a weighted sum of the results in Table 7 we get the positive result in (52) because we are counting the label a twice: once for the x_1 row and a second time for the x_2 row.

When we only count each label once, we get the following table:

Table 8: Expected Label ASC

l	p(l)	I(l)	K(l|C)	ASC(l,C,P)
a	0.5	1.0	1	0.0
b	0.15	2.74	2	0.74
c	0.1	3.32	3	0.32
d	0.125	3.0	4	-1.0
e	0.075	3.74	5	-1.26
f	0.05	4.32	5	-0.68

Taking the weighted sum of Table 8, we get

$$\begin{aligned} &0.5\times 0 + 0.15\times 0.74 + 0.1\times 0.32 + 0.125\times -1 \\ &+ 0.075\times -1.26 + 0.05\times -0.68 = -0.11 \end{aligned} \tag{55}$$

which is consistent with (53). This resolves the apparent contradiction between (52) and (53), demonstrating ASC is conserved under stochastic processing in the example.

5. CONCLUSION

Standard information theory cannot account for meaningful information. Algorithmic specified complexity can, but it is a probabilistic quantitiy. One question we have addressed is: can the expected ASC be positive? We proved the expected ASC is negative, and provided empirical application of the claim with OASC. We then use the negativity of expected ASC to prove a conservation of expected ASC under stochastic

processing, and determistic and stochastic conservation of complexity for individual events.

ACKNOWLEDGMENTS

Thanks to Dr. Winston Ewert for deriving ASC and explaining its mathematical properties.

Thanks to Dr. Tom English and anonymous reviewers for the critical feedback that helped clarify the article.

Thanks to Prof. Leonid Levin for the work in [11], which provided inspiration for the conservation of complexity under stochastic processing proof.

After writing the deterministic and stochastic conservation of complexity proofs, the authors discovered that Dr. William Dembski made a similar, but stronger, argument in [22].

1. Ewert W, Marks RJ, Dembski WA (2013) On the improbability of algorithmic specified complexity. In 45th Southeastern Symposium on System Theory. IEEE. doi:10.1109/ssst.2013.6524962
2. Papoulis A, Pillai SU (2002) Probability, random variables, and stochastic processes. Tata McGraw-Hill Education
3. Shannon CE (2009) A mathematical theory of communication. In Claude E. Shannon. IEEE. doi:10.1109/9780470544242.ch1
4. Li M, Vitányi P (1997) An Introduction to Kolmogorov Complexity and Its Applications. Springer New York. doi:10.1007/978-1-4757-2606-0
5. Kolmogorov AN (1968) Three approaches to the quantitative definition of information. International Journal of Computer Mathematics 2:157–168. doi:10.1080/00207166808803030
6. Chaitin GJ (1966) On the length of programs for computing finite binary sequences. Journal of the ACM 13:547–569. doi:10.1145/321356.321363
7. Solomonoff R (1964) A formal theory of inductive inference. part i. Information and Control 7:1–22. doi:10.1016/s0019-9958(64)90223-2
8. Chaitin G (2007) The halting probability omega: Irreducible complexity in pure mathematics. Milan Journal of Mathematics 75:291–304. doi:10.1007/s00032-006-0064-2
9. Cover TM, Thomas JA (2005) Elements of Information Theory. John Wiley & Sons, Inc. doi:10.1002/047174882x
10. Martin-Löf P (1966) The definition of random sequences. Information and Control 9:602–619. doi:10.1016/s0019-9958(66)80018-9
11. Levin LA (1984) Randomness conservation inequalities: information and independence in mathematical theories. Information and Control 61:15–37. doi:10.1016/s0019-9958(84)80060-1
12. Milosavljević A, Jurka J (1993) Discovering simple DNA sequences by the algorithmic significance method. Bioinformatics 9:407–411. doi:10.1093/bioinformatics/9.4.407
13. Rivals E, Delgrange O, Delahaye JP, Dauchet M, Delorme MO, Hénaut A, Ollivier E (1997) Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. Bioinformatics 13:131–136. doi:10.1093/bioinformatics/13.2.131
14. Eddy SR (2008) A probabilistic model of local sequence alignment that simplifies statistical significance estimation. PLoS Computational Biology 4:e1000069. doi:10.1371/journal.pcbi.1000069
15. Feldmann H, Aigle M, Aljinovic G, André B, Baclet M, Barthe C, Baur A, Bécam A, Biteau N, Boles E (1994) Complete DNA sequence of yeast chromosome II. The EMBO Journal 13:5795–5809. doi:10.1002/j.1460-2075.1994.tb06923.x
16. Ewert W, Dembski W, II RM (2014) Algorithmic specified complexity. In Engineering and the Ultimate: An Interdisciplinary Investigation of Order and Design in Nature and Craft, 131–151. Blyth Institute Press. doi:10.33014/isbn.0975283863.7
17. Holloway E (2018) Creativity and machines. Communications of the Blyth Institute 1:13–16. doi:10.33014/issn.2640-5652.1.1.holloway.1
18. Holloway E (2018) The logical possibility of halting oracles. Communications of the Blyth Institute 1:56. doi:10.33014/issn.2640-5652.1.1.holloway.3
19. Bartlett J (2014) Using turing oracles in cognitive models of problem-solving. In Engineering and the Ultimate: An Interdisciplinary Investigation of Order and Design in Nature and Craft, 99–122. Blyth Institute Press. doi:10.33014/isbn.0975283863.5
20. Copeland BJ (1998) Turing's o-machines, searle, penrose and the brain. Analysis 58:128–138. doi:10.1093/analys/58.2.128
21. Dembski WA (2005) Specification. Philosophia Christi 7:299–343. doi:10.5840/pc20057230
22. Dembski WA (2006) No free lunch: Why specified complexity cannot be purchased without intelligence. Rowman & Littlefield
23. Devine S (2014) An algorithmic information theory challenge to intelligent design. Zygon® 49:42–65. doi:10.1111/zygo.12059
24. Dembski WA (1998) The Design Inference. Cambridge University Press. doi:10.1017/cbo9780511570643
25. Fisher RA (1992) Statistical methods for research workers. In Springer Series in Statistics, 66–70. Springer New York. doi:10.1007/978-1-4612-4380-9_6
26. Ewert W, Dembski W, Marks RJ (2015) Algorithmic specified complexity in the game of life. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45:584–594. doi:10.1109/tsmc.2014.2331917
27. Ewert W, Dembski WA, Marks RJ (2015) Measuring meaningful information in images: algorithmic specified complexity. IET Computer Vision 9:884–894. doi:10.1049/iet-cvi.2014.0141
28. Montanez GD (2018) A unified model of complex specified information. BIO-Complexity 2018. doi:10.5048/bio-c.2018.4