Answers Research Journal 18 (2025): 290–323.
https://assets.answersresearchjournal.org/doc/v18/signification_distinction.pdf

Significant Distinction: A Proposed Metric for an Objective Measurement of Instructional Information, and Its Inference from Functional Complexity

Peter Rankin, 11398 Highway 76 W, Laurens, South Carolina 29360

Abstract

How can we objectively distinguish between the prose of a monkey's banging on a keyboard versus the poetry of Longfellow? Does a 30-minute 100-megabyte video of kittens playing with a ball of yarn truly contain a thousand times more information than the plain text of a highly intelligent book that takes up less than 0.1% of the same disk space on a computer? Or is there a metric which can more accurately compare the two? How should we measure for repetition and redundancy within a message? This article explores the concept of "significant distinction" as an objective measure of information which aligns closely with our intuition. Significance refers to the data's representation of something outside itself, such as a set of instructions for building a piece of functional equipment. Distinction refers to the degree to which the data is not internally redundant. The conjunction of the two attributes is the metric called significant distinction, or its information content under the definition here, when applied to data. Using the same principle, a reasonable method for inferring information from the functional complexity of the end product is given. It is then compared against other concepts, such as Shannon information, Kolmogorov complexity, specified complexity, and coded information systems theory. Possible applications are considered, including arguing for a young and specially created human genome.

Keywords: information; information theory; DNA; genetic entropy

Introduction

In the first century, Paul the apostle discussed the meaningful communication of information in 1 Corinthians 14. He stressed two critical attributes: distinction, or clear differences within the communication of a message (for example, verse 7, "except they give a distinction in the sounds"); and significance, or the symbolic nature of a message (for example, verse 10, "There are, it may be, so many kinds of voices in the world, and none of them is without signification"). The conjunction of these two attributes, when fleshed out more fully, provides a new metric for measuring the information content of a message. In many cases, this metric can be quite objective, especially when the end (or purpose) of the data is objective.

One common criticism from evolutionists is that creationists do not have a clear and objective definition of what we mean by "information." The purpose here is to explore significant distinction as another objective measurement of information which quantifies our meaning in a specific sense, a sense which is common and intuitive. Other metrics within information theory do exist. Here is a summary of some popular ones:

- **Shannon information** measures the level of "randomness" (or entropy) in a data source. It was created to find ways of communicating the same data with less transmission by picking up on patterns in the sender. For example, in English, the letter "e" appears most frequently, and so it should not require as much space in data to send as a less common letter. Combinations like "th" and "sh" are also very common, thus having lower Shannon information.
- **Kolmogorov complexity**, or algorithmic complexity, is a theoretical measure of the shortest computer program capable of generating the data in question. This means that some data sets seemingly very complex, such as images of fractals, contain remarkably little Kolmogorov complexity.
- **Specified complexity** has been described by William Dembski as the difference between the Shannon information of the data and the Kolmogorov complexity of its description (Dembski 2024, under "Specified Complexity as a Unified Information Measure," paragraph 6, beginning with "With Ewert's lead"). "Specificity" means the simplicity in describing the data, and "complexity" basically means improbability, or Shannon information. For example, a highly improbable set of DNA instructions which produces something with a very simple description, like "flagellar motor," would have high specified complexity per Dembski's metric.
- **Werner Gitt's definition** consists of five "layers" of information: statistics (for example, number of letters, their frequencies, etc.), syntax (for example, how letters are allowed to be joined together), semantics (the meanings represented by the symbols in the text), pragmatics (how the message is to be carried out practically), and apobetics (purpose, meaning, teleology). (Gitt 2000, 50–82, chapter 4, "The Five Levels of the Information Concept").

- **Coded Information Systems (CIS)** treats the coded message as part of a broader system. The metric is concerned with how much the system's behavior is refined to attain a goal as the result of the coded message and its interpretation. Not only the coded message, but also the design details of the system itself, can contribute to the measurement of information under this metric.

The new metric proposed here borrows from existing discussions and metrics. For example, creationists have long discussed the idea of "meaning" in data, which is essentially our term "significance." Creationists have also stressed that there is a difference between the highly ordered yet simple patterns which sometimes form spontaneously in nature (such as patterns in mineral crystals or snowflakes) and the complex information found in DNA; this consideration is similar to our term "distinction."

If we abstract the concepts of "significance" and "distinction," we can apply the underlying principles not only to data, but to physical systems themselves; we will call this "functional complexity." Functional complexity is the sum of all the details of a physical system which contribute to the carrying out of its function, such as, for example, all the intricate physical components of a car engine which aid in its ability to turn the driveshaft. It is the analog, in the physical realm, of significant distinction in the realm of data.

This discussion has three parts. First, we will define and illustrate the concept of significant distinction. Second, we will discuss how to infer the existence of significant distinction in a set of instructions (the coded message) by analyzing the functional complexity of the resulting physical system. For example, we can infer that there exists much information (significant distinction) in a dragonfly's DNA by analyzing the functional complexity of the resulting insect. Third, we will compare this metric of significant distinction with several popular metrics, both in information theory and in creationist discussions more broadly. The goal is to provide an objective metric for the intuitive sense of the term "information" which can then be incorporated into broader arguments, including, for example, the evidence that DNA has a Creator, the objectivity of the decay of genetic content in this fallen world, and by extension, the evidence of its recent creation.

Defining Significant Distinction

The definition of this proposed metric is as follows: *Information content can be measured as distinct data significant to some end.*

This definition hinges on two key concepts: (1) significance and (2) distinction. The former is a philosophical concept requiring abstract conceptual terminology and discussion, the latter is mathematical. The former can be detected only through mental activity, the latter through statistical computation. The former requires an identified end outside of the message, while the latter analyzes only the message itself. Both steps are needed for this metric, the philosophical and the mathematical.

For example, suppose that you have a furniture assembly booklet. Reading it, you recognize its end (that is, "purpose," broadly speaking), which is to direct the building of a desk for writing. How would you measure its significant distinction toward that end? First, you would mark all the text in the booklet which aids in building the writing desk in some way. For example, you can exclude the legal disclaimers, other things like the table of contents, etc.; because although they may be significant in other ways, they do not help you build the desk (the end product). This is how you would measure raw significant data. Next, you would account for repetition or redundancy in the remaining data to determine the amount of uniqueness, or distinction. The result is the amount of information you have detected under this metric; that is, the amount of significant and distinct data. Consider the following illustrations:

- **Insignificant indistinction**:
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  (This text carries no meaning and thus lacks significance; and since all the letters are the same, there is no distinction or variety.)
- **Significant indistinction** (relative):
  (Sports fan) "Go Tigers! Go Tigers! Go Tigers!…"
  (These cheers are significant relative to the fans and to the team; yet it is mostly repetitive, and so it has little distinction.)
- **Insignificant distinction**:
  "cSxeWuYDbX xXP LfibUzvOR; bCmSGFpG wyMQQklqRT, ROGNuD,…"
  (This text was generated randomly and has no significance or meaning; however, it contains almost no pattern, and thus it has extremely high distinction.)
- **Significant distinction**:
  "I shot an arrow into the air, It fell to earth, I knew not where…"
  (This text has meaning in English and is significant; it also is non-repetitive, except for repetitions of syntax, and thus it has a high level of distinction.)

Basic definitions

Before defining significance and distinction more precisely, we should define a few preliminary terms. First, by "detail" is meant any physical property of a system. Details about a desk include its weight, width, length, height, materials, color, the types of joints between the pieces, and so on. Details can be very broad, such as the size of the desk relative to the room; or very narrow, such as the position of individual fibers of wood in a panel, or even the position of each molecule in the desk.

By "data," we mean sequential digital data; that is, a sequential series of definite integers or symbols. On a computer, everything is numeric at its foundation and is ultimately a series of simple bits (0 and 1). If you type a sentence on the computer, each letter is stored as a number in an 8-bit "byte" (or multiple bytes) which correlate to that letter. We can do something similar for any language with a definite alphabet, because we can map each symbol to its own integer. For instance, we could decide to map A=1, B=2, etc.

By "serialization," we mean the process of converting physical details into data. For example, when you measure a desk and write down its specifications, you are serializing the physical details of the desk into written data. (More broadly, this is called "quantification;" but our purpose is more precise, since we desire a serial string of numbers, as we will see.) Detail refers to the raw physical properties, such as the literal length of a desk, while data refers to the representation of this length as a sequence of integers or letters.

Distinguishing detail from data is important for a few reasons. First, raw details (or properties) can be represented as symbols in a variety of ways. You could describe the length of a desk as "4 ft 1 in" or "49 in," and both would mean the same thing. We could use English or metric measurements (inches vs. centimeters) or use Spanish or Japanese. Computers have different ways of storing numbers with decimal places. Precisely the same detail could be represented in many different ways in data. Second, data can represent detail in a kind of terse "shorthand" if the receiver knows what the terms mean. Third, data can meaningfully symbolize non-physical concepts, including emotion and spiritual matters. Thus, distinguishing between data and detail is important for our purposes.

Additionally, we should distinguish between the data and the communication channel. English text can be communicated via an e-mail on a computer, on a letter written in cursive with a fountain pen, or over a radio broadcast, for example. In each case, the same digital English data is being communicated, only over different channels.

By the term "end," we mean that toward which something is working. (Since the section on "significance" is conceptual or philosophical, we use these words in a conceptual sense, consistent with standard dictionary definitions, and not in a technical engineering sense.) "End" is like the term "purpose," except that it does not necessarily imply intelligent intent, which allows us to use our metric in an argument demonstrating intelligent intent without having to circularly assume it beforehand. One of the ends of the DNA of a dragonfly is to give its wings the ability to propel it through the air, as an example. The "product" is built to achieve the end (such as the wings, nerves, etc.). For instance, if you look at the blueprints of a car engine, the "end" is to turn a driveshaft, while the "product" is the engine itself. In this case, the end is very simple (rotation of a shaft), while the product is highly complex (all the varied components of the engine to accomplish this end).

Significance

Next, we come to defining the two core concepts, significance and distinction. The first, significance, is the philosophical and conceptual side of the metric, and it requires mental activity. Significance cannot be measured using computation and statistical features of the data alone. To infer significance, first, we must identify an appropriate "end" for the data. For instance, studying the blueprints of a car engine, with sufficient knowledge, we can recognize that its end is to turn a driveshaft. Once we identify the end, we can mark all data which contributes in some way toward that end, or all the data which helps in turning the driveshaft. For our purpose, we do not care about the "degree" of significance; we simply place all significant data into a set.

Notice that before we can apply our metric at all, we must recognize an appropriate end of the data upfront. Measuring information (specifically, the "significance" portion) in our intuitive sense here can never be done by looking at the data in complete isolation. Paul stresses this point (1 Corinthians 14:11): "Therefore if I know not the meaning of the voice, I shall be unto him that speaketh a barbarian, and he that speaketh shall be a barbarian unto me." In "Information, Genetics and Entropy," Barrios (2015) stresses the need of the observer to understand the code to infer semantic value, or meaning, indicating that semantic value cannot be inferred from a data set in complete isolation, which agrees with Paul's point. Further, the objectivity of our measure is therefore limited by the objectivity of the identified end; in cases like beauty and art, it is likely that these are presently far too subjective for our metric. Other times, we may not adequately recognize the significance of the data (Daniel 12:4; 1 Peter 1:11).

However, although our goal here is not to argue the philosophy of value or ultimate meaning as such, we should note that functional ends are quite clear and uncontroversial. Microbiology has made it clear that DNA is significant to the survival of an organism. Survival and reproduction are very simple and objective ends that even evolutionists will acknowledge as having value in their theory; and thus, so are all subordinate ends, such as the organism's ability to see, hear, move, and digest. As a result, all DNA which contributes toward these ends is significant in that respect, and we can apply our metric.

Further, we need not circularly assume intelligent intent to have a valid "end" of the data (we use the words "goal" or "purpose" as synonyms only in the loose sense of these terms). As Dembski (2025, under "5 A Note About Targets" [in submission]) notes in his paper "The Law of Conservation of Information: Natural Processes Only Redistribute Existing Information," a "target" (essentially the same basic concept as the term "end" used here) does not require the assumption of purpose; targets can be natural or neutral, such as with functionality; and Dembski notes that even prominent evolutionists refer to the same concept from a naturalistic standpoint. Function is a particularly good candidate as an end (or target) against which we can objectively measure significant distinction for our purposes here as well.

As a practical example, consider a chapter in a construction book about concrete foundation footers, written in English. You know the language, and you also know something of construction; and so with relative ease, you identify the end of this data, which is to create a strong platform to support a structure. Thus, all data in this chapter which contributes to the strength of the foundation is significant to that end. If the foundation will be weaker without rebar, then the data concerning rebar is significant; and if the way the concrete is allowed to cure affects its strength, then it is significant, too.

It is impossible to prove the absence of significance in data. Again, Paul makes the point that if he does not know the meaning of a language, to him, it sounds like gibberish (1 Corinthians 14:10–11). Something might sound like gibberish and yet be perfectly understood by another person who knows the language. Similarly, the events of our lives often seem very chaotic and without significance; but we know that if we love God, He is working all these things together for our good (Romans 8:28; Proverbs 16:33). In general, we cannot say that a data set has no significance unless we know that the ultimate source is truly random. While information often leaves a mathematical signature in the data (for example, traces of detectable patterns found in English text), this is merely the shadow, and not the substance, of information.

In summary, with objective ends (like turning a driveshaft), our metric can also be objective. The way alterations to the engine affect its torque and overall performance is often a rather simple matter. While we may not be able to exhaustively identify all information, we can identify much of it; and the metric here allows for incremental detection.

Literal vs. symbolic data

There is a critical distinction that we need to make for our purposes, which is between literal and symbolic data. Put simply, data is symbolic when it is full of symbols to things outside itself. It is literal when the pieces of data reference each other in mathematical or visual "webs" of interconnectivity. Failure to make this distinction can result, for instance, in unrealistically optimistic expectations regarding the ability of matter, via the varied environments and laws of chemistry, to self-organize into "networks" to form life and information (see Cronin and Walker 2016).

For example, in ASCII art, people "draw pictures" using only the letters and symbols on the keyboard with a monospace font (where every character is the same width). People can be very creative within these limitations. For an illustration of an elephant using only ASCII characters, please see fig. 1.

```
                             _
                           .' `'.__
                          /      \ `'"-,
         .-''''--...__..-/ .     |      \
       .'                ; :'     '.  a   |
      /                  | :.       \     =\
     ;                   \':.      /  ,-.__;.-;`
    /|     .              '--._   /-.7`._..-;`
   ; |       '                |`-'      \  =|
   |/\        .   -' /     /  ;         |  =/
   (( ;.       ,_  .:|     | /     /\   | =|
    ) / `\     | `""`;     / |    | /   / =/
      | ::|    |      \    \ \    \ `--' =/
     /  '/\    /       )    |/      `-...-`
    /    | |  `\    /-'    /;
    \  ,,/ |    \   D    .'  \
jgs  `""`   \  nnh  D_.-'L__nnh
             `"""`
```

Fig. 1. ASCII art elephant. Art by Joan G. Stark. https://www.asciiart.eu/animals/elephants.

What is the value of this data? It is in the literal way that the symbols appear in physical proximity to each other when printed on a page or screen. For instance, the letter "a" was chosen as the elephant's eye because it looks visually like an eye with an eyelid in that font; and this character was surrounded by spaces because of the visual effect; and the positions of the various lines and marks have to do with how they literally work together to produce the visual picture of an elephant. The two-dimensional shape mimics the physical shape of the elephant rather than using symbolic representation. The pieces of data are strongly self-referential. This is a good example of what we mean by literal data.

By contrast, symbolic data is full of symbols to things outside itself. Werner Gitt (2000, 85) limits the domain of information to those systems dealing with abstract references to reality (that is, coded messages, or symbols), as opposed to observations of reality which are direct. For example, consider the word "elephant." This word is not used for any literal similarity between the letters and the animal. The letters are combined merely as a symbol which stands for the concept. The term "significance" includes the idea of symbol, especially in the more classical use of the word, where to "signify" meant to stand for something outside itself. In semiotics, the terms "signifier" and "signified" are still used in this sense. Thus, in a shallow sense, the concept of "significance" can be as simple as the meaning of an individual word in isolation (such as "elephant" standing for the animal); but this same idea of significance, in a deeper sense, can refer to the way in which words, put together, help further the end goal. For instance, the message "Feed the elephant a handful of peanuts for obedience" can have deeper significance to the zookeepers to the extent that it furthers the elephants' obedience, thus promoting the overall success of the zoo. Language allows such symbols to be joined in complex relationships to reference real-world objects and concepts.

Data can be literal or symbolic to varying degrees. While the ASCII art is mostly literal, there is a sense in which you could say that it stands for the elephant by visual analogy when taken as a whole, even as opposed to an exact photograph, and is thus at least a little symbolic. Some words are onomatopoeic, meaning that they are chosen for sounding like the things they stand for, and are thus slightly literal (for example, "sizzle" or "zoom"). Typically, data is either almost entirely symbolic, or almost entirely literal. Our definition of "significance" assumes symbolic data, and thus excludes literal data from consideration (for example, we exclude images of fractals, or the elephant ASCII drawing). This difference will become important later in distinguishing our metric from Kolmogorov complexity.

Distinction

The second key concept in this proposed metric is distinction. While the first part, significance, is conceptual and philosophical, the second part, distinction, is entirely mathematical. Distinction is the uniqueness of the data, or its lack of internal repetition; that is, the non-repetitiveness of the letter patterns in the text. When all data being analyzed has already been identified as significant (per the mental, philosophical method above), then the level of distinction correlates to the information density of the significant text. It is an entirely intrinsic measurement, distinguishing it from classic Shannon information, which is based on the probabilities of the source which generates it. For our purposes, we do not need to know the probability distribution of the source or the general statistics of the language. We need only the message itself to measure its distinction. (Measurements are meaningful over full-length messages, rather than over small, isolated pieces of text.)

To calculate the distinction of the message, we iterate through each character (the "cursor") and determine its distinction, measured in bits. The distinction of the message is the sum for all individual characters, as seen in eq. 1.

$$d = \sum_{i=1}^{m} -\log_2\left(F\left(i\right)\right) \qquad \text{(eq. 1)}$$

where
i=the 1-based cursor position of the character under consideration. For the first letter in the message, i=1, etc.
m=the total message length.
$F(i)$=the expectation fraction for that cursor position. (We will discuss how to calculate this next.)

That is, for each cursor character (at position i), we convert an expectation fraction ($F(i)$) into distinction bits by using the negative log, base 2, to get distinction in bits. This will look familiar compared with Shannon information, except that we are not dealing with probabilities in that sense, but only with the patterns intrinsic to the message itself.

To calculate the cursor expectation fraction $F(i)$, we must take any applicable prefixes into account for each cursor character (position). A prefix is simply the text that comes immediately before the cursor character being considered. Let a given prefix length be notated with the variable j (that is, with a two-letter prefix, j=2). We need to calculate the prefix target fraction (t), which is the ratio of times a given prefix results in the same letter as the cursor. For example, if half the times s occurs in a message, it is followed by the cursor letter h, then the prefix fraction will be roughly 0.5 ($^1/_2$). This prefix target fraction (the variable t) is calculated using eq. 2.

$$t = \frac{x+1}{x+y+1} \qquad \text{(eq. 2)}$$

where
x=the number of other prefix matches (or, the number of other times this prefix occurs and is also followed by the cursor character). For instance, if the cursor is a [SPACE] character, and the prefix under consideration is "the" (j=3), then x is the number of other times the text "the" occurs within the message, while also followed by a [SPACE].
y=the number of other prefix occurrences which do

not match the cursor, or in our example, the number of times "the" is found in the message while being followed by some character other than [SPACE].

In eq. 2, we include the cursor instance itself in the calculation (hence the +1 in both the numerator and denominator) to get the prefix fraction.

Next, we must calculate a "certainty fraction" (c) for the prefix under consideration. Roughly, the more a prefix match is likely to occur by "chance" alone (to use the term loosely), the less the certainty fraction. Patterns are less special the more "likely" they are. We calculate this certainty fraction c using eq. 3.

$$c = 1 - a^x \cdot (1-a)^y \cdot \frac{(x+y)!}{x!\,y!} \qquad \text{(eq. 3)}$$

where

a=the character distribution fraction of the cursor. For example, if the cursor character is E, and if there are 5 Es in the message, and the message's length is 79 characters total, then a= 5/79 for that cursor character.

x, y=the number of other prefix matches/non-matches (see notes on eq. 2).

The odds of a single other prefix matching the cursor by "chance" (loosely using the term) is simply the value a, since that is the frequency of the cursor character relative to the other characters in the message. The odds of a single prefix not matching is the inverse, (1–a). The odds of a given particular constellation of matches/non-matches is therefore $a^x \cdot (1-a)^y$. However, there are many different ways we could arrange these matches and non-matches (permutations), which is what the factorial portion is considering. For all combinations of whole numbers (x, y) where $x+y=z$, the sum of the probabilities calculated here is exactly 1; or put another way, the probabilities are divided across all possibilities of matches/non-matches, with some more or less likely. Again, we use the term "probability" loosely, referring really to the patterns within the message contents.

To calculate the cursor expectation fraction $F(i)$, we use the iterative equation; see eq. 4. Let n be the maximum applicable prefix length for the cursor (discussed below); the cursor's expectation fraction will be the iterative result of eq. 4, where $j=n$. This iterative approach also quantifies the intuition that the longer a pattern continues, the more "specific" it gets, the stronger the pattern, and the lower the distinction going forward.

$$f_j = f_{j-1} - (f_{j-1} - t_j) \cdot c_j, f_0 = a \qquad \text{(eq. 4)}$$

where

j=the prefix length under consideration.

f_{j-1}=the expectation fraction so far; that is, the result of the previous iteration.

t_j=the target prefix fraction calculated in eq. 2, for prefix length j.

c_j=the prefix certainty fraction calculated in eq. 3, for prefix length j.

We begin with j=0 (no prefix) and increment j, terminating when either of these conditions is true:

1. We terminate if the prefix contains no patterns in the next character which follows it. This is because we seek to measure redundancy. For example, suppose the prefix is "the;" there must be at least two instances of this prefix which are followed by the same character (they don't have to be consecutive). If the prefix results in something different every time, then there is no pattern to measure.
2. We terminate if this is the first prefix "match" (the first time this prefix has resulted in the same character as the cursor). We must measure the internal distinction of all content, including patterns the first time they occur; only in subsequent repetitions does it make sense to factor them in as redundancies. The exception is if "$t_j<f_{j-1}$;" in which case, we do factor this prefix in, because it is an "outlier" in the prefix pattern. This means the other prefix occurrences will have reduced distinction for repetition; and so, we should factor in the rarity of this prefix occurrence (just as we assign greater distinction to rare individual characters relative to common ones).

To see an example using the text of John 1:1, please refer to Appendix B, which details the measurement of select letters for illustration. It is interesting to compare different languages of the same translated text. For instance, John chapter 3 (the entire chapter) measures at 7,325.6 bits of distinction in English, 6,645.4 bits in Greek, and 6,082.3 bits in Latin (all caps). This is an average of 1.81 bits per character in English, 0.99 in Greek, and 1.79 in Latin. For calculation results using this computer program over various texts, including chapters or small books of the Bible, as well as random binary and hexadecimal data, see Appendix C. For the PHP computer program which measured these distinctions programmatically, see Appendix A. Appendix D contains quantitative comparisons of short sample texts among four metrics suitable for such comparison.

Under this definition, we can note a couple of interesting things that may happen when combining multiple messages into one. It is likely that combining message A and message B will yield a total distinction which is less than the sum of their individual distinctions when taken separately. For example, the string "abcdefg" has a total distinction of 19.65 (2.81 per character). The string "bcdefg" has a total distinction of 15.51 (2.58 per character).

Yet when combining them into a single string ("abcdefgbcdefg"), the total distinction is 22.8 (1.75 per character), which is less than the combined distinction of each string taken alone. This is because the internal repetition is increased with the shared data between these two strings.

However, it is also possible for the total distinction of messages to be greater than the sum of each message taken separately. As an example, the string "aaaaa" has 0 distinction, as does "bbbbb." But combining them ("aaaaabbbbb") yields a distinction of 6.80 (0.68 per character). This actually aligns with our intuition. On the one hand, if you have heard instructions before, a repetition adds little to the "information" of the message. On the other hand, having to constantly switch contexts (such as switching between English and Latin constantly in older texts) would be more mentally taxing, having a higher degree of distinction. Generally, in such cases, Latin (for example) is italicized to set it apart from the English, making it more predictable and easier to read.

Consider the letter 'A' repeated a million times in a row. Although this is a relatively large amount of data, it has no variety or distinction. If we increase the repetition from a million to a trillion (a million million), although we have increased the amount of raw data a millionfold, we have increased the distinction by nothing. On the other hand, truly random data would have very high distinction, with little pattern.

Information contained in raw (uncompressed) intelligent messages generally falls between these two extremes, containing a decent level of distinction, but also a lot of repetition. (While this is a practical reality in real-world information, it is not an essential characteristic for our purpose.) If you text your friend, "I will be there in fine minutes," your friend knows that you meant "five minutes;" but this would not be possible if the message were maximally compressed. For instance, if you used numbers instead of spelling out the word and wrote, "I will be there in 6 minutes," your friend would not know for sure whether you made a mistake or meant to send a precise time. The presence of redundancy is one safeguard against miscommunication, and natural languages make heavy use of it.

For reference, Genesis chapter 1 has about 4.9 kilobits of distinction, with an average distinction of about 1.19 bits per character. John chapter 3 has about 7.3 kilobits, with 1.8 bits per character. Both passages were measured using sentence casing, no verse numbers, with paragraphs. Genesis 1 contains more redundancy than John chapter 3, or less relative distinction given its length.

Also, when we speak of distinction, we are not referring to Kolmogorov complexity. (For reference, Kolmogorov complexity is measured by the shortest computer program which can generate the text.) For instance, a picture of the Mandelbrot set has a high degree of visual complexity and yet can be generated with relatively simple computer instructions. However, these considerations do not apply in our case, because we require data which is full of symbols to things outside itself to be "significant." While each pixel stands for the result of the calculation at those coordinates, the data is still interconnected in a mathematical "web," related by literal mathematical and visual relationships. However, symbolic data does not work this way. For example, consider the word "car" in English. You have assigned integers to each letter: A=1, B=2, etc. If you were to mathematically modulate this word by adding 1 to each letter, that would translate as: c-to-d, a-to-b, r-to-s; and so "car"+1 (adding one to each letter) becomes "dbs." Any meaning of the new symbol would not be because of their mathematical connections. That works only with literal data.

Unwrapping data

When we measure significant distinction, it is against data in its "unwrapped" state. Data can be wrapped in layers of encoding and representation, but our concern is only with the final data which is intrinsically symbolic of things outside itself. In other words, when we say "significance," we imply immediate extra-data significance; and thus, by our definition, we exclude all data wrapping from consideration. We care only about the heart of the data, which is immediately symbolic of things outside itself.

For example, suppose you have reserved a table at a restaurant, and your parents are on their way, and your mother sends you a text message, "Almost there, 5 minutes." That's 23 bytes of raw data (a byte is 8 bits; a bit is either a 0 or a 1), if encoded in UTF-8 (for illustration; UTF-8 is a way of encoding letters into bytes). Now suppose instead of sending this text message, she sends you an audio file in the chat with 2 seconds of recorded length where she says the same thing. For simplicity, let's assume a relatively high-quality MP3 recording, with a data rate that remains constant at 128kbps (128 kilobits per second, or 128,000 bits of data per second of audio). At 2 seconds, that is 256 kilobits, or 32,000 bytes. This is over 1,300 times the size of the text, and yet they both communicate the same sentence. Intuitively, we know that the MP3 does not contain over a thousand times more information than the text; and this is due to data wrapping. The MP3 may include additional data such as tone of voice and background noise, but these things extend beyond the realm of our discussion, which is with strictly digital and objective data significant toward a definite end (in our case, English text meant to inform).

In this example, the English text "Almost there, 5 minutes" is transmitted as an analog signal, wrapped further in digital representation:

Wrap Layer 2: Digital MP3 (31 kilobytes, 256k bits/(8 * 1024))
Wrap Layer 1: Analog audio
Raw Data: Digital English text (23 bytes)

Significant distinction is always measured in relation to data in its fully unwrapped state relative to its end. The same concept as the MP3 would apply to video files and pictures. A video of kittens playing with a ball of yarn has much less information than a book written by Einstein, even if the book, in text format, is less than 0.1% the file size. Both audio and video recordings are essentially analog, even if they are wrapped digitally. (How the brain interprets such analog data into digital signals is likely inaccessible to us, placing it out of reach of more objective measurements of significant distinction.) No matter how many layers data is wrapped in, our concern is with the fully unwrapped data itself, in this case, in the form of digital English text.

As another example, security video footage contains much data, but it cannot be measured for information content unless an extra-data end is identified, beyond the mere preservation of the video files themselves. But if video footage becomes important in solving a case of bank robbery, then the unwrapped digital data that specifically helped in this case becomes significant to that end; but to measure its significant distinction, it must be unwrapped from its encoding as video, which can be somewhat subjective. One way to do this would be to describe the significant details of the video in a written language; that is, "At 4:21pm, a masked man 6 ft 2 in tall walks through the front door holding a gun…", etc. Historical video recordings may contain meaning of a sort, but we cannot apply our metric until an end transcending the data is first identified. Otherwise, it does not fall within the domain of significant distinction.

Content vs. profit

We should note that the measure of information content is different from a measure of its profit (or value). If you have a bucket of sand, you can measure it by its volume, or you can measure it by its weight and density relative to other materials. Information is a different kind of entity, and our measure is analogous to volume in some ways; but we also factor in distinction, which is analogous to density or weight. But there is another key aspect to information which is outside our measure, which is profitability.

For example, Don Batten gives the example that the sentence "She has an automobile" contains 21 characters; yet so does the phrase, "Sue has a red Porsche" (Batten 2017, under "Some thoughts about 'new information'"). The second sentence, however, is much more specific; and thus, it conveys more "information" in a sense, if we are talking about profit. This is because the purpose of these statements is to increase knowledge; and in the realm of knowledge acquisition, in this case, the specific is more valuable than the general. A detective also tends to value the specific over the general, as he must find out who robbed the bank or pulled the trigger. Yet sometimes, the general is more valuable, such as saying, "Every household in this subdivision has two automobiles," which is more general yet applies more broadly, thus increasing the amount of "information" transmitted in a more concise way than listing each house in the subdivision one by one.

The measure of profit is highly subjective to the receiver. Suppose you lived in the year 1920 and heard, "Bob has purchased a new Model T." This might sound quite specific to us living today, but in 1920, about half the cars in America were Model Ts, and so it is not all that specific. If later, you heard that Bob had purchased a "black" Model T, that would add almost no informational value, because the Model Ts produced at that time were all black. When the end of information is to increase knowledge, then the measure of profitability, and even of specificity, are highly subjective to the receiver's assumptions, and not strictly based on the raw nature of the facts being conveyed.

The point, however, is that significant distinction is a measure of information content, and not of information profit. The information structure found in the DNA molecule, for example, utilizes such a dense and highly profitable language that it must greatly surpass English text in its profitability; but such determinations are outside our scope here. We are dealing only with the parts of DNA which are interpreted sequentially, or as one digital character after the other. (It does not matter if it is left-to-right, right-to-left, or any other direction; it matters only that the letters are processed in sequence, one after the other.) Any additional considerations, such as the parts of DNA facilitating dynamic 3D folding, when they are not necessarily interpreted sequentially by the cell, likely contain additional sorts of information which transcend our measure here.

Application to functional complexity

While significance and distinction can apply to data, the concepts, when abstracted, can also apply to physical details of existing systems. This allows us to reasonably infer a minimum amount of significant distinction (instructional information) in the underlying code which generates the systems.

The usefulness of this will be discussed later.

Functional complexity can be seen as the analog in the physical realm of significant distinction in the realm of data. (This analogy parallels the comparison between "detail" and "data" made earlier.) We can define functional complexity as the sum total of the distinct details of a physical system which contribute toward it carrying out a particular end (or function). For example, the functional complexity of an engine includes all the details of the physical engine which aid it in turning the driveshaft.

Applying these concepts to functional complexity requires some abstraction in the ideas of "significance" and "distinction," however. With physical systems, we are no longer speaking of symbol, and so we must use the term "significance" in a broader, more general sense if we wish to apply the same basic concept. Details are "significant" to the function (in this more abstract sense) if they contribute toward the performance of that function (or end). For instance, if you look carefully at a car engine and figure out how it works, even if you do not have a blueprint, you can infer that a great many details are "significant" to the engine's proper running. You could measure the diameter of the pistons and then run experiments, either in reality or possibly using knowledge of chemistry and physics equations, to determine the range of acceptable tolerances before the engine's performance degrades. The precision of tolerance, starting with the current actual diameter, gives us the specification for a significant detail about the engine.

We can also measure the distinction of physical systems. However, this is more subjective and requires us to recognize physical repetitions or patterns, and to serialize them accordingly, so that these patterns will be detected and accounted for. For instance, putting significance aside for a moment and speaking only of distinction, you could infer that the distinction of the shape of well-ordered ice crystals is very low compared with water before it freezes, because with well-ordered crystals, the molecules generally follow a repeating pattern. In Cronin and Walker's Assembly Theory, for instance, since identical building blocks can be reused in this case and given the physical constraints of the low temperature which naturally forms these building blocks, the ice crystals would require little selection (Sharma et. al. 2023). Snowflakes contain more distinction, but even then, most of their beauty is from the repetition of the patterns; and thus, they have far less distinction than the random position of water molecules before the crystals formed. On the other hand, the individual grains of a pile of sand poured onto a table have a great deal of distinction under this definition, since each grain has its own orientation and position, and with little pattern. However, we should note that they are similar in respect to all being grains of sand, and the distinction is limited only to their individually unique shapes and orientations. This is like Assembly Theory's emphasis on whether individual building blocks are highly reusable.

Thus, we can apply the underlying concept of significant distinction either to details or to data. It is more objective when applied to data. When in relation to physical details, we will call this concept "complexity" and when in relation to data, we will call this "information." These definitions align closely with the general distinctive use of each word in everyday speech: complexity is to a car engine as information is to a blueprint. When referring to how a mousetrap works, we speak of complexity, whether little or much; but when we speak of the instructions for building a mousetrap, we speak of information, whether little or much.

Inferring instructional information from functional complexity

It is highly reasonable to infer a minimum level of instructional information from functional complexity. The complexity of a firefly implies a great deal of information in the DNA. Yet we must subtract external contributive preexisting complexities (part of the surrounding environment), which we will look at in a moment.

This does not work for ends which are themselves representations of digital data. For instance, to use our previous example, this does not work with pictures generated by a Mandelbrot set function. In this case, the complexity is far greater than the instructional information, but the end is data-bound; that is, the end is in the data itself. With functional ends, this is not applicable. While we can get emergent visual complexity from a simple mathematical formula (the Mandelbrot picture), we cannot get emergent functional complexity in that way. This needs more research and thought, but there are likely two main reasons.

First, functional requirements in the real world are rather simple. To make an abstract argument, to produce extreme functionality from simple information (as with the Mandelbrot set), it would require the surrounding context of the functioning system to be a sort of "inverse" of this mathematically emergent complexity; in which case, the resulting functionality would fit its surroundings as a hand fits into a glove. Both environment and machine would be based on relatively simple mathematical principles, merely as inversions of each other. However, this does not seem to be the way the real world is set up; and if it were in some case, we could probably detect it without too much difficulty. For instance, for a dragonfly's wing (and all that goes into it, such as the nervous system, muscles, etc.) to propel it forward, it must do so in an air space which is, at best, of soup-like consistency, and at worst, very chaotic. The functional complexity of its wings simply cannot be mathematically emergent, because the consistency of the air cannot be so, in that way.

Second and related, functionally complex machines (such as car engines) are not strongly patterned the way that the Mandelbrot data is. If we analyze the Mandelbrot images, we will see a lot of repetitions, variations, and inversions of the same basic shapes. With functional complexity in the real world, we can reasonably rule out emergent mathematics from its lack of these kinds of patterns.

Thus, it is reasonable to infer a minimum level of instructional information by taking the functional complexity of the object and subtracting any contributive complexity. Royal Truman, in his theory of Coded Information Systems (CIS) goes into great depth on preexisting resources and other considerations which supplement the coded message itself (Truman 2012). One example of contributive complexity is that of the agents carrying out the instructions, to the extent it is used. For example, if the instructions are, "Make a kitchen table of X width and length," because the carpenter already has a lot of knowledge gained by making past tables, these simple instructions rely on this inherent contributive complexity in the carpenter's skillset. In general, if the agents are intelligent (for example, human factory workers), creativity and cleverness can also be dynamic contributive complexities, making up for a deficiency of instruction; but these are probably unquantifiable in many cases. Mindless agents, such as the components of a cell, are theoretically far easier to quantify in this regard.

Another type of contributive complexity is that of the surrounding environment, to the extent it applies. Truman notes that things like cell membranes can add to the coded message itself during interpretation (Truman 2012, under heading "Messages vs sensors in CIS theory"). This is outside the scope of our definition of information here, since our definition applies only to the digital data (the coded messages, in Truman's model), not considering additional contributions by the receiver during interpretation. Thus, Truman's definition of "information" is broader than ours here. But for example, if the product is coded to allow trial and error to adapt to its environment, then certain parameters can fine-tune themselves to the environment by its contribution. For example, God created dogs with the genetic ability to have hair of varying length through genetic recombination. Coupled with natural selection (and likely other mechanisms), this causes adjustments in the hair length of a community of dogs to suit their environment. In a cold region, dogs with longer hair survive better, and eventually, the genetic variants for shorter hair are filtered out of the gene pool. The preference of which hair length should dominate was not originally specified in the information, but it was tuned through input by the environment. However, this is trivial compared to the information needed for hair in the first place, long or short, and compared to the information allowing for genetic variety at all.

The level of functional complexity is almost always superseded by the amount of instructional information required to build it; which, in turn, is almost always superseded by the amount of intelligence required to generate the instructions. For example, to describe a car engine is difficult; but to understand all the instructions for assembling one from scratch is far more difficult; and to write these instructions in the first place is the most difficult task of all. In general, like a waterfall, intelligence cascades down into information, which cascades down further into functional complexity. Thus, given a pool of substantial functional complexity, it is most reasonable to infer that it cascaded down from a higher plane of information, which itself cascaded down from a higher plane of intelligence.

This principle of inferring information from functional complexity can be very helpful, for example, when discussing the information in DNA. One way to measure the information content in DNA would be to look at the data itself; but we can also infer DNA information by looking at the resulting complexity of the functional product, less any epigenetic or environmental contributions. This remaining complexity, which is the vast majority, implies at least this same amount of information content in the DNA, and almost certainly immensely more. For example, by studying a dragonfly, we notice all the design details that must be in place for it to function: the shape of the wings, the eyes, its sense of balance, the tail, and a host of other variables. If we can serialize these variables to their proper tolerances and couch them in a well-formed language, we can infer at least this amount of information in the original instructions. One benefit of this metric is that it need not be exhaustive. We can detect part of the information content without needing to quantify the entire amount.

As a simple example, perhaps you have been to a fast-food restaurant where the lid did not quite fit onto the fountain drink cup. Suppose that the rim is 10 cm in diameter, and that the tolerance for the diameter of the lid's rim is 1 mm before the fit degrades. This is a window of 2 mm (1 mm either way), or a 2% total variation. Thus, the amount of distinction for this literal number is ~5.6 bits ($-\log_2(0.02)$). If we know that the diameter of the lid was set via instructional information, and not by trial-and-error or a self-referencing specification (for example, by creating the lid using the cup as a kind of mold), then we can reasonably infer at least 5.6 bits of information to specify the diameter of the lid. This is a conservative measurement, since it takes only relative diameter into account and not the absolute starting point (that is, it is likely that more than six yes/no questions, or bits, are required to get a diameter which is functional for the cup). See Truman 2012 for more detailed thought on such considerations. Further, we should factor in many more parameters: the thickness of the lid's plastic, the material, the width of the cuts into which to insert the straw, the specifications for the bubbles on the top to punch in the type of drink, the circularity, the creases to add strength, etc. And this does not even consider the instructions surrounding these raw parameters.

Comparisons with Other Metrics

In his book *Signature in the Cell*, Stephen Meyer (2009, 362, 371) discusses many of these concepts. He speaks of "functional significance in the pattern of letters" and says, "If an improbable sequence produced a functional outcome, then it was also specified in the sense that Dembski's method required." This is the basic idea of what we mean here by significance. In his book *In the Beginning Was Information*, Werner Gitt (2000, 55) says, "*Shannon's* theory of information is suitable for describing the statistical aspects of information, e.g. those quantitative properties of languages which depend on frequencies. Nothing can be said about the meaningfulness or not of any given sequence of symbols." This also refers to the concept of significance as defined here.

In a paper entitled "Information as Distinctions: New Foundations for Information Theory," David Ellerman (2013, 3.6) explores a metric he says is dual and convertible to and from Shannon information: "By solving the dit-count and the bit-count for p0 and equating, we can derive each measure in terms of the other… Thus the two notions of entropy are simply two different ways, using distinctions (dit-counts) or binary partitions (bit-counts), to measure the information in a probability distribution." The sense of "distinction" by Ellerman is not what is presented here, although, as with Shannon information, it has similarities.

Shannon information

Shannon information shares similarities with our measurement of distinction, but it has key differences. Here are a couple of examples. First, classic Shannon information deals with probabilities regarding the sender or source, whereas we are looking only at the message on its own merits when it comes to distinction (though to first identify significance, we must use knowledge extrinsic to the data sequence itself).

For example, in their paper "Shannon Information and Kolmogorov Complexity," Peter Grünwald and Paul Vitányi (2010, 2) write that with Shannon information, "it is only the characteristics of that random source that determine the encoding, not the characteristics of the objects that are its outcomes." They go on to quote Shannon, who explains that semantics are excluded from consideration, and that his system is designed before the specific message to be sent is known (Shannon 1948).

Our measure of distinction is an intrinsic measurement, whereas probability, properly speaking as meant by Shannon, is extrinsic to the specific data set. For instance, the text "zzzzzzz" might have just as much Shannon information as a random block of text, or even more, given the rarity of the letter "z," provided the random source's probability distribution does not typically result in such orderly sequences. By contrast, significant distinction does not take the source into account at all, and thus measures no distinction, or no variety; and if this completely repetitive string is taken alone, it has no significant distinction.

A second example of a difference with Shannon information is that we exclude conditional probability for prefixes during that pattern's first occurrence in a message. This differentiates our measurement here from others which use Shannon entropy in the classical sense. For instance, Clément Pit-Claudel outlines a Python script which calculates entropy of a text using n-grams, but the author notes that contamination can occur with longer values of n because this calculation does not make a special case for the first occurrence of a pattern. For instance, Pit-Claudel (2013, under "Experimental results") writes, "In the graph presented above, it is therefore likely that the estimates for n over 5 or 6 are already plagued by a significant amount of sampling error; indeed, for any given n there are in total 27^n possible n-grams consisting only of the 26 alphabetic letters and a space, which exceeds the size of the corpus used in this experiment as soon as $n>5$." (See also, Cover and Thomas 2006, 168–169.) For our purposes, the first occurrence cannot be considered a redundancy, but only subsequent occurrences. Thus, while our measure shares similarities with Shannon information, there are key differences.

Kolmogorov complexity

Kolmogorov complexity is a measure of the theoretically minimum underlying algorithmic complexity of a data set. That is, it is a measure of the shortest possible computer program which could generate the data. Data sets which appear highly complex can sometimes have very little Kolmogorov complexity. As mentioned previously, an extremely detailed image of the Mandelbrot can be generated by a relatively short computer program, since the math for generating these images is rather simple. Our concept of distinction is different, however, because it deals with simple repetition of data, and not with algorithmic modulation.

Earlier, we considered why Kolmogorov complexity does not measure what we desire here. In summary, if the virtue of a data set is in its modulated mathematical relationships, then it is literal. In contrast, symbolic data stands for things outside itself; and if such a mathematical relationship did exist in the data, it would be incidental to our purposes and could be ignored anyway. Further, Kolmogorov complexity cannot be calculated in most cases due to the halting problem, but an intuitive metric of detected information must be accessible at a basic level.

Coded information systems

Royal Truman's Coded Information Systems (CIS) theory measures the behavioral refinement of the target system through use of some coded message (Truman 2012, Figure 3). Truman's metric is broader, and more abstract, than significant distinction.

Because CIS quantitatively measures final behavioral refinement, the metric will include any contributions of preexisting resources (that is, contributions of the target machinery implementing the message). By contrast, significant distinction excludes these preexisting resources, interested in the measure of information content in the coded message alone.

One interesting advantage of the CIS metric is that it is likely to factor out redundancies in the original coded message automatically. Because it measures behavioral refinement, by the time the coded message reaches this stage of effect, any extraneous redundancies in the message are likely to have been ignored (or combined) by the target system during interpretation and thus filtered out of the final metric. Significant distinction, on the other hand, must use math to filter out redundancy.

Werner Gitt's definition

Significant distinction shares many similarities with Werner Gitt's definition of information. His definition consists of five "layers" (or levels) of information. "Significance" corresponds rather well to the last three layers (particularly to semantics and apobetics, and to a lesser extent, to pragmatics). However, with significant distinction, we do not presuppose purpose or teleology but instead refer neutrally to the "end" to which the data is working. We can thus use significant distinction in a broader argument to demonstrate intelligent intent as a conclusion without having to assume it circularly beforehand. "Distinction" corresponds to the first two layers, and particularly to the first, statistics. It corresponds only incidentally to the second, syntax, because syntax forms the basis of the patterns or repetitions within the text, and these patterns affect its measure of distinction.

Specified complexity

A concept that appears quite similar at first glance is specified complexity. For convenience, in this section, SC will stand for specified complexity, and SD will stand for significant distinction. These two concepts are highly congruent in application; what yields a high score in one will likely be high in the other, and vice versa. However, they are essentially distinct, and their purposes are different as well. They are measuring completely different things.

First, SC relies on classic Shannon information as part of its formula. William Dembski (2024, under "Shannon and Kolmogorov Information," paragraph 5) writes, "The complexity in specified complexity is therefore Shannon information." However, SD uses a modified intrinsic metric which is distinct from Shannon information. This makes SC more source-focused, while SD is more data-focused. For instance, William Dembski gives the example of ten people in a room who all confirm that their birthdays are January 1; in this case, the sequence has high complexity in his sense ("complexity" in SC refers to improbability; less probable events are considered more complex). In our sense, however, ten identical numbers (birthdays) would have no distinction, and thus, no information content as we mean it, if taken alone. SC is about the probability of the event being produced under some hypothesis, whereas SD does not consider this at all.

Second, with SC, the Kolmogorov complexity of the description (specification) is inversely correlated to (subtracted from) the specified complexity of the message (Dembski 2024, under "Specified Complexity as a Unified Information Measure," paragraph 6 beginning "With Ewert's lead"; Dembski and Ewert 2023, under 6.4 "Specification and Complexity"). In other words, the simpler the specification for the data, the lower the description's Kolmogorov complexity, and the higher the specified complexity. That is, short descriptors are more "specific" (in this sense) than long descriptors. By contrast, SD does not use Kolmogorov complexity at all. (The halting problem is not an issue with SC, because the relation is inverse, applied to the descriptor and not to the data; that is, the halting problem could result in a more conservative measurement of SC at times, but never in a more liberal one.)

Third, SC can apply to data-bound data (i.e., when the data's value is in the patterns within the data itself), whereas SD requires extra-data significance to be considered information. A set of Fibonacci numbers qualifies as SC by virtue of its intrinsic data patterns, whereas it would not qualify as SD, not having extrinsic significance by itself. The same would go for 25 coin tosses of "heads" with a fair coin; it is very interesting and improbable, but taken alone, it contains no SD.

Fourth, SC focuses on descriptors of the products for specificity, whereas SD focuses on the ends. These are similar but not the same. For instance, in SD, the end may be to "turn the driveshaft;" but in SC, the descriptor may be "an internal combustion engine." The complexity of describing the product is not a factor in SD specifically, whereas it can be a factor in SC.

In short, SC is concerned with improbability and simplicity of specification, whereas, strictly speaking, SD is concerned with neither. (However, it is probable that ends worth measuring in SD will be simple.) SD is concerned only with quantifying information in a particular sense which is often meant in everyday, intuitive speech, but with this sense only, and nothing more, taken alone. Though both concepts are highly congruent in interesting ways, they are fundamentally distinct measurements; and thus, they are compatible and can be used together in the same argument.

Conclusion

Significant distinction can be useful, including in some young-earth Christian creation arguments. First, it is a simple and objective measure which aligns well with intuition. Multitools can be very useful, but perhaps sometimes, a tool with a narrower purpose can be more precise. Maybe by separating out the concept of mere intuitive information from other factors, it can then be combined with others for greater force in certain settings. For example, perhaps it can more easily quantify what it would take to add new information to the human genome, or which mutations are objectively removing information; and thus, the argument of genetic entropy can be strengthened even more, further demonstrating deterioration and the necessity of recent creation. It could also provide one additional metric to support the claim that "information" as meant by creationists is indeed an objective entity at a basic level in many cases. Measuring the significant distinction of portions of DNA would require the expertise of geneticists to quantify portions which are significant and interpreted sequentially. For instance, the kinds of mutations which are beneficial by virtue of breaking or deactivating functionality reduce significant distinction objectively, and thus, they could not be used as examples to explain how evolution accounts for the rise of significant distinction in the first place.

Second, the concept of significant distinction aligns closely with the biblical ideas mentioned by Paul. What Paul describes in 1 Corinthians 14 includes communication even by lifeless entities like musical instruments, possibly making this definition especially applicable to describing DNA, which is transmitted and carried out by unconscious chemical processes, yet while being true information, nonetheless. In the Bible, God seems to have included the basic keys for many amazing concepts; and this may be one such case.

Third, it is probable that we can easily add concepts to significant distinction to aid in an inference of intelligence. For example, perhaps the simpler the end, and the more complex the product, the stronger the implication of intelligence. Flight is a very simple end for a dragonfly (that is, this idea—of darting freely through the air in any direction—is simple enough to grasp as a concept); and yet the actual mechanics involved to achieve this, or the product, including its wing design, eyesight, body, tail, etc., are complex in the extreme. To imagine flight is easy, but to effect it is arduous. This argument would incorporate specified complexity, and it would also parallel irreducible complexity, which is where many parts work together such that if any did not do their job, the entire machine would fail. The famous example of irreducible complexity given by Michael Behe is the simple mousetrap; to function properly, it needs the platform, the hammer, the spring, the catch, and the holding bar. If any one of these is missing, the entire mousetrap's function becomes useless (Behe 2006, 42).

In conclusion, significant distinction may be a biblical, intuitive, simple, incremental, and objective measure of information content. Presuming that the concept laid out here is accurately built upon the biblical criteria discussed in 1 Corinthians 14, this would be another example of the Bible's supernatural insight, not only in the subjects of history, science, human relationships, and spiritual matters; but also, in the topic of information theory, and written in the first century.

References

Barrios, Julio Ernesto Rubio. 2015. "Information, Genetics and Entropy." *Principia: An International Journal of Epistemology* 19, no. 1: 121–146.

Batten, Don. 2017. "Nylon-Degrading Bacteria: Update: Nylonase does not Support Microbes-to-Mankind Evolution." May 19. https://creation.com/nylonase-update.

Behe, Michael J. 2006. *Darwin's Black Box: The Biochemical Challenge to Evolution*. New York, New York: Free Press.

Cover, Thomas M., and Joy A. Thomas. 2006. *Elements of Information Theory*. 2nd ed. Hoboken, New Jersey: John Wiley & Sons, Inc.

Cronin, Leroy, and Sara Imari Walker. 2016. "Beyond Prebiotic Chemistry: What Dynamic Network Properties Allow the Emergence of Life?" *Science* 352, no. 6290 (3 June): 1174–1175.

Dembski, Bill. 2025. "The Law of Conservation of Information: Natural Processes Only Redistribute Existing Information." *Bio-Complexity* [under submission].

Dembski, Bill. 2024. "Specified Complexity Made Simple." Bill Dembski. February 26. https://billdembski.com/intelligent-design/specified-complexity-made-simple/.

Dembski, Bill, and Winston Ewert. 2023. *The Design Inference: Eliminating Chance through Small Probabilities*. 2nd ed. Seattle, Washington: Discovery Institute Press.

Ellerman, David. 2013. "Information as Distinctions: New Foundations for Information Theory." University of California/Riverside. January 23, 2013. https://ellerman.org/wp-content/uploads/2013/01/InfoAsDits.pdf

Gitt, Werner W. 2000. *In the Beginning Was Information: A Scientist Explains the Incredible Design in Nature*. Ulm, Germany: Ebner.

Grünwald, Peter and Paul Vitányi. 2010. "Shannon Information and Kolmogorov Complexity." *Centrum Wiskunde and Informatica*, July 22. https://homepages.cwi.nl/~paulv/papers/info.pdf.

Meyer, Stephen C. 2009. *Signature in the Cell: DNA and the Evidence for Intelligent Design*. New York, New York: HarperCollins Publishers.

Pit-Claudel, Clément. 2013. "An Experimental Estimation of the Entropy of English, in 50 Lines of Python Code." *Code Crumbs*. December 8. http://pit-claudel.fr/clement/blog/an-experimental-estimation-of-the-entropy-of-english-in-50-lines-of-python-code/.

Shannon, C.E. 1948. "The Mathematical Theory of Communication." *Bell System Technical Journal* 27 (July, October): 379–423, 623–656.

Sharma, Abhishek, Dániel Czégel, Michael Lachmann, Christopher P. Kempes, Sara I. Walker, and Leroy Cronin. 2023. "Assembly Theory Explains and Quantifies Selection and Evolution." *Nature* 622, no. 7982 (12 October): 321–328.

Truman, Royal. 2012. "Information Theory—Part 3: Introduction to Coded Information Systems." *Journal of Creation* 26, no. 3 (December): 115–119.

Appendix A. PHP computer program to calculate internal distinction of text.

```php
<?php // Appendix A: A PHP program to measure internal distinction of text

// File name: distinction.php
// Sample usage:
//
//   php distinction.php "IN THE BEGINNING WAS THE WORD ..."
//   php distinction.php "../path/to/text/file.txt"

/**
 * The statistics of a particular prefix text within a message
 */
class PrefixStats
{
    /**
     * The total number of times this prefix occurs within the message, with at least
     * one character following (i.e., cannot be on the very end to be a prefix)
     */
    public int      $Occurrences;

    /**
     * The count of the next letters immediately following this prefix within
     * this message; e.g., ["a" => 2, "e" => 12, ...], meaing that "a" follows
     * this prefix twice in the message, "e" 12 times, etc.
     */
    public array    $NextCharCounts = [];

    /** The positions of the first occurrences of each "next char". The ordinal
     * is measured at the start of the associated prefix occurrence. E.g.,
     * ["a" => 1055, "e" => 145, ...], meaning that the first "a" following this
     * prefix comes after the prefix beginning at position 1055, etc. Positions
     * are 0-based (0 is the first position).
     */
    public array    $NextCharFirstUses  = [];

    /**
     * The position of the first occurrence of this prefix within the message.
     * Positions are 0-based (0 is the first letter of the message).
     */
    public int      $FirstUse;

    /**
     * Get the fraction of times the prefix is followed by the specified
     * character. I.e., if 1 out of 4 prefix occurrences is followed by
     * an "a", then this fraction would be 0.25.
     */
    public function getNextCharFraction(string $char): float {
        $ncc = $this->NextCharCounts[$char] ?? 0;
        return (float)$ncc / (float)$this->Occurrences;
    }

    /**
     * When building these prefix stats, register the occurrence of a character
     * that follows the prefix (keeping track of the totals and first occurrences)
     *
     * @param string $char The character following the prefix
     * @param integer $prefixOrdinal The position of the prefix occurrence under
     *   consideration
     */
    public function registerNextChar(string $char, int $prefixOrdinal) {
        $ncc = $this->NextCharCounts[$char] ?? 0;
        if ($ncc == 0) {
            $this->NextCharFirstUses[$char] = $prefixOrdinal;
        }
        $ncc += 1;
        $this->NextCharCounts[$char] = $ncc;
    }

    /**
     * Check whether this prefix contains any repetition (patterns after the prefix). If
     * not, it is not applicable in our measurement.
     */
    public function containsRepetition(): bool {
        foreach ($this->NextCharCounts as $char => $count) {
            if ($count > 1) return true;
        }
        return false;
    }
}

/** Measure the distinction of a string of text */
class Distinction
{
#region Private variables
    /** The string being measured for distinction */
    private string  $s;

    /** Cache the length of the string for speed */
    private int     $strlen;

    /** An associative array of each character to its total count in the string;
     * e.g., ['a' => 4, 'b' => 2, ...] */
    private array   $charCounts         = [];

    /** An associative array of character distribution fractions, or the fraction
     * that each character takes of the entire message; e.g., ['a' => 0.03, ...] */
    private array   $charDistributions  = [];
#endregion

#region Public variables
    /** The calculation precision in digits (1k should be plenty for most situations) */
    public int      $CalcPrecision      = 1000;

    /** Very large unreduced prefix fractions take up a lot of calculation power per
     * factorials; to help, the program can auto-reduce fractions to a given denominator
     * for approximate measurements. About 400 should be plenty. */
    public int      $ApproximatePrefixFraction = 400;

    /** Whether to write tracing output to the console */
    public int      $TraceLevel         = 5;

    /** Cap the cache entries at this position (to avoid too much memory usage) */
    public int      $CacheCap           = 1000000;

    /** The maximum prefix length that will be taken into consideration (performance) */
    public int      $MaxPrefixLength    = 100;

    /** Log text for each character, which may be printed to the output depending on
     * settings. Also useful to allow printing of overview stats at the beginning. */
    public array    $CharLogs           = [];

    /** @param string $s The string to analyze for distinction */
    public function __construct(string $s) {
        // Normalize string for regex purposes
        $s              = str_replace(["\n", "\r"], '`', $s);
        $this->s        = $s;
        $this->strlen   = strlen($this->s);

        // Get the number of occurrences for each character
        for ($i = 0; $i < $this->strlen; $i++) {
            $char = $this->s[$i];
            if (!isset($this->charCounts[$char])) {
                $this->charCounts[$char] = 0;
            }
            $this->charCounts[$char] += 1;
        }

        // Calculate the character distribution fractions
        foreach ($this->charCounts as $char => $count) {
            $this->charDistributions[$char] = (float)$count / $this->strlen;
        }
    }

    /**
     * Get the distinction of the string per this measurement, in bits
     *
     * @return float The distinction, in bits, of the given string
     */
    public function getDistinction(): float {
        $total = 0.0;

        // Loop calculating and summing the distinction of each character in the message
        for ($cursor = 0; $cursor < $this->strlen; $cursor++) {
            $charLog        = [];
            if ($cursor % 1000 === 0) echo "."; // Display progress every 1k chars

            $char = $this->s[$cursor]; // The character being analyzed
            $a    = $this->charDistributions[$char]; // The character's frequency fraction
            $f    = $a; // Which is also the initial expectation fraction

            // Loop through any applicable prefixes, factoring in their "pulls" on the
            // expectation fraction
            for ($prefixLength = 1;
                $prefixLength < $this->MaxPrefixLength;
                $prefixLength++
            ) {
                // Make sure the prefix length is valid and applicable
                $prefixPosition = $cursor - $prefixLength;
                if ($cursor - $prefixLength < 0) break;
                $prefix      = substr($this->s, $prefixPosition, $prefixLength);
                $ps          = $this->getPrefixStats($prefix, $prefixLength);
                $firstMatch = $ps->NextCharFirstUses[$char];
                $target      = $ps->getNextCharFraction($char);
                $x           = $ps->NextCharCounts[$char] - 1; // Number of _other_ matches
                $y           = $ps->Occurrences - 1 - $x; // Number of non-matches
                $terminate  = false;
                if ($ps->containsRepetition() == false) {
                    $terminate = true; // No redundancy to measure
                } else if ($firstMatch   >= $prefixPosition) {
                    // If this is the first _match_, then we use the prefix fraction as
                    // the target only if it falls below the char distribution fraction
                    // (to account for the extra specificity needed in the rarity of this
                    // character); otherwise, this prefix is not applicable.
                    if ($target >= $f) {
                        $terminate = true;
                    }
                }

                // Run the calculation and adjust the expectation fraction
                $certainty  = $this->getCertainty($a, $x, $y);
                if ($this->TraceLevel >= 7) {
                    $charLog[] = "\n      "
                        . ($terminate ? '[terminate] ' : '')
                        . "[j=$prefixLength "
                        . ', x=' . $x
                        . ', y=' . $y
                        . ", t=" . number_format($target, 3)
                        . ', f[j-1]=' . number_format($f, 3)
                        . ', c=' . number_format($certainty, 3)
                        . ']';
                }
                if ($terminate) break;

                $f = $this->pullNumber($f, $target, $certainty);
            }

            // Convert the expectation fraction into distinction bits
            $d      = -log($f, 2);
            $total  += $d;

            // Log the results, if applicable
            if ($this->TraceLevel >= 3) {
                $charLog = array_merge([
                    'd=' . number_format($d, 2)
                    . ' f=' . number_format($f, 3)
                    . ' pl=' . ($prefixLength - 1)
                ], $charLog);
            }
            $this->CharLogs[] = $charLog;
        }

        echo "\n";
        return $total;
    }

    /** Get the line-item detail logs to print */
    public function getDetailLogs(): string {
        $details = [];
        for ($cursor = 0; $cursor < $this->strlen; $cursor++) {
            $c = $this->s[$cursor];
            $showCursor = $cursor + 1;
            $detail = "[$showCursor] $c - " . implode(" ", $this->CharLogs[$cursor]);
            $details[] = $detail;
        }
        return implode("\n", $details);
    }

    /** Associative array of cached [string Prefix => PrefixStats, ...] */
    private array $prefixCache = [];
    /**
     * Get the stats for a given prefix
     *
     * @param string $prefix The prefix whose stats to get
     * @param integer $prefixLength For performance, provide length of prefix
     * @return PrefixStats The prefix stats
     */
    private function getPrefixStats(string $prefix, int $prefixLength): PrefixStats {
        $ps = $this->prefixCache[$prefix] ?? null;
        if (!$ps) {
            // This prefix hasn't been cached yet, and so build it
            $ps         = new PrefixStats();

            // Use lookaheads to include overlapping matches (more important in binary),
            // and assure that at least one character follows in the message
            $regex      = '/(?=' . preg_quote($prefix, '/') . '.)/';
            $numMatches = preg_match_all($regex, $this->s, $matches, PREG_OFFSET_CAPTURE);
            if ($numMatches === false) throw new \Exception("Regex failed: " . $regex);
            if ($numMatches === 0) throw new \Exception("No matches at all for prefix.");

            $ps->FirstUse       = $matches[0][0][1];
            $ps->Occurrences    = $numMatches;
            foreach ($matches[0] as $match) {
                $ordinal    = $match[1];
                $lastChar   = $this->s[$ordinal + $prefixLength];
                $ps->registerNextChar($lastChar, $ordinal);
            }

            $this->prefixCache[$prefix] = $ps;
        }
        return $ps;
    }

    /**
     * Pull a number from one toward another linearly by a given fraction
     *
     * @param float $original The original number
     * @param float $target The target number
     * @param float $byFraction The fraction by which to pull
     * @return float The number after being pulled toward the target
     */
    private function pullNumber(
        float $original,
        float $target,
        float $byFraction
    ): float {
        return $original + (($target - $original) * $byFraction);
    }

    /** Cache the certainty calculations for combinations of (a, x, y) */
    private array $certaintyCache = [];
    /**
      * Get the certainty fraction for a given combination of (a, x, y), per Equation 2.
     *
     * @param float $a The character distribution fraction within the message
     * @param integer $x The number of other prefix matches with the cursor
     * @param integer $y The number of prefix non-matches with the cursor
     * @return float The corresponding certainty fraction
     */
    private function getCertainty(float $a, int $x, int $y): float {
        // Serve the cached value if available
        $cacheKey = number_format($a, 15) . ":" . $x . ":" . $y;
        if (isset($this->certaintyCache[$cacheKey])) {
            return $this->certaintyCache[$cacheKey];
        }

         // Large fractions can have excessive factorials; if configured, automatically
        // reduce them to a reasonable denominator.
        if ($x + $y > $this->ApproximatePrefixFraction) {
            $total  = $x + $y;
             $div    = (float)$total / $this->ApproximatePrefixFraction;
            $x      = floor($x / $div);
            $y      = floor($y / $div);
        }

        // Calculate the certainty fraction given the probability of a specific
        // occurrence, taking into account permutations (see Equation 2).
        $permutations = $this->getPermutations($x, $y);
        $a            = number_format($a, 20);
        $probability  = floatval(bcmul(
            bcmul(
                bcpow($a, $x, $this->CalcPrecision),
                bcpow(
                    bcsub(1.0, $a, $this->CalcPrecision), $y,
                        $this->CalcPrecision), $this->CalcPrecision),
            $permutations, $this->CalcPrecision));
        $certainty = 1.0 - floatval($probability);

        $this->certaintyCache[$cacheKey] = $certainty;
        return $certainty;
    }

    // Cache permutations for sets of (x, y), for faster processing speeds
    private array $permutationsCache = [];
    /**
     * Get the permutations for given values of (x, y).
     *
     * @param integer $x The number of other prefixes which match with the cursor
     * @param integer $y The number of other prefixes which do not match
     * @return string The number of applicable permutations
     */
    private function getPermutations(int $x, int $y): string {
        if (isset($this->permutationsCache[$x . ':' . $y]) == false) {
            // We can simplify (x+y)!/(x!y!) somewhat by dividing out the
            // greater number in the denominator from the numerator; to,
            // product(y+1, ... y+x)/x!, or the opposite, if x is greater.
            $g               = max($x, $y);
            $l               = min($x, $y);
            $num             = 1;
            $denominator     = 1;
            for ($i = $g + 1; $i <= ($g + $l); $i++) {
                $num  = bcmul($num, $i, $this->CalcPrecision);
            }
            for ($i = 1; $i <= $l; $i++) {
                $denominator = bcmul($denominator, $i, $this->CalcPrecision);
            }
            $val = bcdiv($num, $denominator);
            $this->permutationsCache[$x . ':' . $y] = $val;
        }

        return $this->permutationsCache[$x . ':' . $y];
    }
}

// Command prompt interaction
if (!isset($ignoreCommandPrompt)) { // Allow calling this programmatically
    $text = $argv[1] ?? '';
    if (strlen($text) == 0) die("Must enter a string to analyze.\n");
    if (strlen($text) < 2000 && str_ends_with(strtolower($text), ".txt")) {
        if (!file_exists($text)) {
            die('Could not find file: ' . $text . "\n");
        } else {
            $text = file_get_contents($text);
```

```php
        }
    }
    $dc                   = new Distinction($text);
    $dc->MaxPrefixLength = 500;
    $dc->TraceLevel       = 10;
    $distinction          = $dc->getDistinction();

    $summary  = "Distinction detected:     " . number_format($distinction, 10) . "\n";
    $summary .= "Per character:            "
        . number_format($distinction / strlen($text), 10) . "\n";
    $summary .= "Character count:          " . strlen($text) . "\n";

    $details    = $dc->getDetailLogs();

    echo $summary . "\n" . $details . "\n" . $summary;
}

```

Appendix B.
Distinction calculation overview for text of John 1:1.

John 1:1 text:
IN THE BEGINNING WAS THE WORD, AND THE WORD WAS WITH GOD, AND THE WORD WAS GOD.

Breakdown of the distinction calculation for some of the cursor characters, for illustration:

• [1] "I": No prefixes, and so the expectation fraction is simply the character distribution fraction ("a"). Since this message contains 4 I's, and there are 79 letters total, a=4/79. The negative log base 2 is ~4.3 bits of distinction.
• [2] "N": Out of 3/4 times "I" is used in this message, it also results in an "N", as here; and so, for j=1 (prefix length of 1), t=0.75 per our equation. Since this is the first occurrence of this pattern, we can take it into account only if it is an outlier (greater distinction), and so we must terminate and ignore this pattern until the next time it is used. This is because a pattern does not lose all distinction simply for being repeated; the first occurrence must still be measured for distinction for accuracy.
• [3] "[SPACE]": Here, a=16/79 (16 spaces out of a message of 79 letters). The prefix "N" (j=1) is used elsewhere with a slight pattern (the letter "D" follows this prefix twice). Further, x=0 (no other prefix occurrences result in a "T"), y=5 (5 other occurrences result in something else). Plugging these into our formulas, we get t=0.167 and c=0.677; this is the first occurrence of the prefix, but it is somewhat of an outlier (t<f[j-1]), and so we count it, and so f(1)=0.178. The next prefix length (j=2), prefix "IN", has no redundancy (it results in something different each time), and so we terminate, and d(i)=-log2(0.178) = 2.49.
• [4] "T": 5/79, 3.98 bits. (We terminate at the first prefix because t>f(0).)
• [5] "H": 5/79, 3.98 bits. (We terminate at the first prefix because t=1, and t>f(0).)
• [6] "E": 5/79, 3.98 bits. (We terminate at the first prefix for the same reason; first pattern occurrences ought not count, for accuracy.)
• [7] "[SPACE]": 16/79, 2.3 bits. We terminate at the first pattern (j=1) for the same reason; E usually precedes a space, meaning this is a pattern, but it is the first occurrence.
• [8] "B": This contains a prefix (the space) which was used at least once before, but it resulted in a different outcome (the letter "T"). However, in this case, t>f(0), and so we terminate and ignore this prefix. a=0.013 (1/79), and for j=1, x=0, y=15, t=0.063, which is greater than 0.013, and so we terminate, as this is the first (and only) pattern match.
• (Skipping to position 72 for a good illustration of many applicable prefix lengths...)
• [72] "W" (of last word "WAS" in verse). There are 7 W's, and so a=0.089 (7/79). The applicable prefixes extend all the way back to 16 characters long (the text "D, AND THE WORD " was used once prior in the message). For most of these (lengths 3-16), the target is 1 (100% of the time, they resulted in a "W"); and the certainty for these is about 0.9 (90%). This means that the expectation fraction will end up being almost exactly 1 by the time we factor in prefix length 16, and so distinction will be essentially 0. We terminate at j=17 because it never occurs elsewhere in the message (x=0, y=0), and thus no redundancy.

Complete Computer Output

For each letter, first the distinction is given in bits, followed by the expectation fraction ("f=") and the maximum applicable prefix length ("pl="). If there are applicable prefixes, then beneath these, each one is printed in brackets on its own output line. "pl=" stands for that particular prefix length; "c=" indicates the certainty fraction; and "f=" indicates the expectation fraction before this prefix was considered.

```
Distinction detected:    151.3426959325
Per character:           1.9157303283
Character count:         79

[1] I - d=4.30 f=0.051 pl=0
[2] N - d=3.72 f=0.076 pl=0
      [terminate] [j=1 , x=2, y=1, t=0.750, f[j-1]=0.076, c=0.984]
[3]   - d=2.49 f=0.178 pl=1
      [j=1 , x=0, y=5, t=0.167, f[j-1]=0.203, c=0.677]
      [terminate] [j=2 , x=0, y=2, t=0.333, f[j-1]=0.178, c=0.364]
[4] T - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=3, y=12, t=0.250, f[j-1]=0.063, c=0.947]
[5] H - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=4, y=0, t=1.000, f[j-1]=0.063, c=1.000]
[6] E - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=3, y=1, t=0.800, f[j-1]=0.063, c=0.999]
```

```
[7]   - d=2.30 f=0.203 pl=0
      [terminate] [j=1 , x=3, y=1, t=0.800, f[j-1]=0.203, c=0.973]
[8] B - d=6.30 f=0.013 pl=0
      [terminate] [j=1 , x=0, y=15, t=0.063, f[j-1]=0.013, c=0.174]
[9] E - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=0, y=0, t=1.000, f[j-1]=0.063, c=0.000]
[10] G - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=0, y=4, t=0.200, f[j-1]=0.051, c=0.188]
[11] I - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=0, y=3, t=0.250, f[j-1]=0.051, c=0.144]
[12] N - d=0.44 f=0.739 pl=1
      [j=1 , x=2, y=1, t=0.750, f[j-1]=0.076, c=0.984]
      [terminate] [j=2 , x=0, y=0, t=1.000, f[j-1]=0.739, c=0.000]
[13] N - d=3.72 f=0.076 pl=0
      [terminate] [j=1 , x=0, y=5, t=0.167, f[j-1]=0.076, c=0.326]
[14] I - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=0, y=5, t=0.167, f[j-1]=0.051, c=0.229]
[15] N - d=0.44 f=0.739 pl=1
      [j=1 , x=2, y=1, t=0.750, f[j-1]=0.076, c=0.984]
      [terminate] [j=2 , x=0, y=0, t=1.000, f[j-1]=0.739, c=0.000]
[16] G - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=0, y=5, t=0.167, f[j-1]=0.051, c=0.229]
[17]   - d=2.30 f=0.203 pl=0
      [terminate] [j=1 , x=0, y=3, t=0.250, f[j-1]=0.203, c=0.493]
[18] W - d=3.50 f=0.089 pl=0
      [terminate] [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
[19] A - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
[20] S - d=4.72 f=0.038 pl=0
      [terminate] [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
[21]   - d=2.30 f=0.203 pl=0
      [terminate] [j=1 , x=2, y=0, t=1.000, f[j-1]=0.203, c=0.959]
[22] T - d=2.06 f=0.240 pl=1
      [j=1 , x=3, y=12, t=0.250, f[j-1]=0.063, c=0.947]
      [terminate] [j=2 , x=0, y=2, t=0.333, f[j-1]=0.240, c=0.123]
[23] H - d=0.00 f=1.000 pl=2
      [j=1 , x=4, y=0, t=1.000, f[j-1]=0.063, c=1.000]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=1.000, c=1.000]
      [terminate] [j=3 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[24] E - d=0.00 f=1.000 pl=3
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.063, c=0.999]
      [j=2 , x=3, y=1, t=0.800, f[j-1]=0.799, c=0.999]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.800, c=1.000]
      [terminate] [j=4 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[25]   - d=0.00 f=1.000 pl=4
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.203, c=0.973]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=0.784, c=0.992]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.998, c=0.992]
      [j=4 , x=3, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [terminate] [j=5 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[26] W - d=1.19 f=0.437 pl=1
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [terminate] [j=2 , x=2, y=1, t=0.750, f[j-1]=0.437, c=0.979]
[27] O - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
[28] R - d=4.72 f=0.038 pl=0
      [terminate] [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
[29] D - d=3.50 f=0.089 pl=0
      [terminate] [j=1 , x=2, y=0, t=1.000, f[j-1]=0.089, c=0.992]
[30] , - d=5.30 f=0.025 pl=0
      [terminate] [j=1 , x=1, y=5, t=0.286, f[j-1]=0.025, c=0.866]
[31]   - d=2.30 f=0.203 pl=0
      [terminate] [j=1 , x=1, y=0, t=1.000, f[j-1]=0.203, c=0.797]
[32] A - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=1, y=14, t=0.125, f[j-1]=0.063, c=0.620]
[33] N - d=3.72 f=0.076 pl=0
      [terminate] [j=1 , x=1, y=3, t=0.400, f[j-1]=0.076, c=0.760]
[34] D - d=3.50 f=0.089 pl=0
      [terminate] [j=1 , x=1, y=4, t=0.333, f[j-1]=0.089, c=0.694]
[35]   - d=2.30 f=0.203 pl=0
      [terminate] [j=1 , x=3, y=3, t=0.571, f[j-1]=0.203, c=0.916]
```

```
[36] T - d=2.06 f=0.240 pl=1
      [j=1 , x=3, y=12, t=0.250, f[j-1]=0.063, c=0.947]
      [terminate] [j=2 , x=1, y=2, t=0.500, f[j-1]=0.240, c=0.833]
[37] H - d=0.00 f=1.000 pl=2
      [j=1 , x=4, y=0, t=1.000, f[j-1]=0.063, c=1.000]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=1.000, c=1.000]
      [terminate] [j=3 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
[38] E - d=0.00 f=1.000 pl=3
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.063, c=0.999]
      [j=2 , x=3, y=1, t=0.800, f[j-1]=0.799, c=0.999]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.800, c=1.000]
      [terminate] [j=4 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
[39]   - d=0.00 f=1.000 pl=4
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.203, c=0.973]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=0.784, c=0.992]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.998, c=0.992]
      [j=4 , x=3, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [terminate] [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
[40] W - d=0.42 f=0.750 pl=5
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [j=2 , x=2, y=1, t=0.750, f[j-1]=0.437, c=0.979]
      [j=3 , x=2, y=1, t=0.750, f[j-1]=0.743, c=0.979]
      [j=4 , x=2, y=1, t=0.750, f[j-1]=0.750, c=0.979]
      [j=5 , x=2, y=1, t=0.750, f[j-1]=0.750, c=0.979]
      [terminate] [j=6 , x=1, y=0, t=1.000, f[j-1]=0.750, c=0.911]
[41] O - d=0.00 f=1.000 pl=6
      [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
      [j=2 , x=2, y=4, t=0.429, f[j-1]=0.412, c=0.954]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.428, c=0.996]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=0.998, c=0.996]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.996]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.996]
      [terminate] [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
[42] R - d=0.00 f=1.000 pl=7
      [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.595, c=0.999]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.999]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=7 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [terminate] [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
[43] D - d=0.00 f=1.000 pl=8
      [j=1 , x=2, y=0, t=1.000, f[j-1]=0.089, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.993, c=0.992]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=7 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=8 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [terminate] [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
[44]   - d=0.89 f=0.540 pl=1
      [j=1 , x=3, y=3, t=0.571, f[j-1]=0.203, c=0.916]
      [terminate] [j=2 , x=1, y=1, t=0.667, f[j-1]=0.540, c=0.677]
[45] W - d=1.19 f=0.437 pl=1
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [terminate] [j=2 , x=1, y=2, t=0.500, f[j-1]=0.437, c=0.779]
[46] A - d=1.23 f=0.428 pl=2
      [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
      [j=2 , x=2, y=4, t=0.429, f[j-1]=0.412, c=0.954]
      [terminate] [j=3 , x=1, y=0, t=1.000, f[j-1]=0.428, c=0.937]
[47] S - d=0.00 f=1.000 pl=3
      [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.595, c=0.999]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.999]
      [terminate] [j=4 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
[48]   - d=0.00 f=1.000 pl=4
      [j=1 , x=2, y=0, t=1.000, f[j-1]=0.203, c=0.959]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.967, c=0.959]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.959]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.959]
      [terminate] [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
[49] W - d=1.19 f=0.437 pl=1
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [terminate] [j=2 , x=0, y=2, t=0.333, f[j-1]=0.437, c=0.169]
[50] I - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=0, y=6, t=0.143, f[j-1]=0.051, c=0.268]
[51] T - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=0, y=3, t=0.250, f[j-1]=0.063, c=0.178]
[52] H - d=0.00 f=1.000 pl=1
      [j=1 , x=4, y=0, t=1.000, f[j-1]=0.063, c=1.000]
      [terminate] [j=2 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[53]   - d=2.32 f=0.200 pl=2
      [j=1 , x=0, y=4, t=0.200, f[j-1]=0.203, c=0.596]
      [j=2 , x=0, y=4, t=0.200, f[j-1]=0.201, c=0.596]
      [terminate] [j=3 , x=0, y=0, t=1.000, f[j-1]=0.200, c=0.000]
[54] G - d=4.30 f=0.051 pl=0
      [terminate] [j=1 , x=1, y=14, t=0.125, f[j-1]=0.051, c=0.633]
[55] O - d=3.98 f=0.063 pl=0
      [terminate] [j=1 , x=1, y=2, t=0.500, f[j-1]=0.063, c=0.833]
[56] D - d=3.50 f=0.089 pl=0
      [terminate] [j=1 , x=1, y=3, t=0.400, f[j-1]=0.089, c=0.732]
[57] , - d=1.99 f=0.251 pl=1
      [j=1 , x=1, y=5, t=0.286, f[j-1]=0.025, c=0.866]
      [terminate] [j=2 , x=0, y=1, t=0.500, f[j-1]=0.251, c=0.025]
[58]   - d=0.05 f=0.967 pl=2
      [j=1 , x=1, y=0, t=1.000, f[j-1]=0.203, c=0.797]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.838, c=0.797]
      [terminate] [j=3 , x=0, y=0, t=1.000, f[j-1]=0.967, c=0.000]
[59] A - d=0.01 f=0.996 pl=3
      [j=1 , x=1, y=14, t=0.125, f[j-1]=0.063, c=0.620]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.102, c=0.937]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.943, c=0.937]
      [terminate] [j=4 , x=0, y=0, t=1.000, f[j-1]=0.996, c=0.000]
[60] N - d=0.00 f=1.000 pl=4
      [j=1 , x=1, y=3, t=0.400, f[j-1]=0.076, c=0.760]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.322, c=0.924]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.949, c=0.924]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.996, c=0.924]
      [terminate] [j=5 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[61] D - d=0.00 f=1.000 pl=5
      [j=1 , x=1, y=4, t=0.333, f[j-1]=0.089, c=0.694]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.259, c=0.911]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.934, c=0.911]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.994, c=0.911]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=0.999, c=0.911]
      [terminate] [j=6 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[62]   - d=0.00 f=1.000 pl=6
      [j=1 , x=3, y=3, t=0.571, f[j-1]=0.203, c=0.916]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.540, c=0.797]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.907, c=0.797]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.981, c=0.797]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=0.996, c=0.797]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=0.999, c=0.797]
      [terminate] [j=7 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[63] T - d=0.00 f=1.000 pl=7
      [j=1 , x=3, y=12, t=0.250, f[j-1]=0.063, c=0.947]
      [j=2 , x=1, y=2, t=0.500, f[j-1]=0.240, c=0.833]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.457, c=0.937]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.966, c=0.937]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=0.998, c=0.937]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [terminate] [j=8 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[64] H - d=0.00 f=1.000 pl=8
      [j=1 , x=4, y=0, t=1.000, f[j-1]=0.063, c=1.000]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=1.000, c=1.000]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [terminate] [j=9 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[65] E - d=0.00 f=1.000 pl=9
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.063, c=0.999]
      [j=2 , x=3, y=1, t=0.800, f[j-1]=0.799, c=0.999]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.800, c=1.000]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [terminate] [j=10 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[66]   - d=0.00 f=1.000 pl=10
      [j=1 , x=3, y=1, t=0.800, f[j-1]=0.203, c=0.973]
      [j=2 , x=3, y=0, t=1.000, f[j-1]=0.784, c=0.992]
      [j=3 , x=3, y=0, t=1.000, f[j-1]=0.998, c=0.992]
      [j=4 , x=3, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [terminate] [j=11 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[67] W - d=0.00 f=1.000 pl=11
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [j=2 , x=2, y=1, t=0.750, f[j-1]=0.437, c=0.979]
      [j=3 , x=2, y=1, t=0.750, f[j-1]=0.743, c=0.979]
      [j=4 , x=2, y=1, t=0.750, f[j-1]=0.750, c=0.979]
      [j=5 , x=2, y=1, t=0.750, f[j-1]=0.750, c=0.979]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=0.750, c=0.911]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=0.978, c=0.911]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=0.998, c=0.911]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [terminate] [j=12 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[68] O - d=0.00 f=1.000 pl=12
      [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
      [j=2 , x=2, y=4, t=0.429, f[j-1]=0.412, c=0.954]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.428, c=0.996]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=0.998, c=0.996]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.996]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.996]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [terminate] [j=13 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[69] R - d=0.00 f=1.000 pl=13
      [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.595, c=0.999]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.999]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=7 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.999]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [terminate] [j=14 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[70] D - d=0.00 f=1.000 pl=14
      [j=1 , x=2, y=0, t=1.000, f[j-1]=0.089, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.993, c=0.992]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=5 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=6 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=7 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=8 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.992]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [terminate] [j=15 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[71]   - d=0.00 f=1.000 pl=15
      [j=1 , x=3, y=3, t=0.571, f[j-1]=0.203, c=0.916]
      [j=2 , x=1, y=1, t=0.667, f[j-1]=0.540, c=0.677]
      [j=3 , x=1, y=1, t=0.667, f[j-1]=0.626, c=0.677]
      [j=4 , x=1, y=1, t=0.667, f[j-1]=0.653, c=0.677]
      [j=5 , x=1, y=1, t=0.667, f[j-1]=0.662, c=0.677]
      [j=6 , x=1, y=1, t=0.667, f[j-1]=0.665, c=0.677]
      [j=7 , x=1, y=1, t=0.667, f[j-1]=0.666, c=0.677]
      [j=8 , x=1, y=1, t=0.667, f[j-1]=0.667, c=0.677]
      [j=9 , x=1, y=1, t=0.667, f[j-1]=0.667, c=0.677]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=0.667, c=0.797]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=0.932, c=0.797]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=0.986, c=0.797]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=0.997, c=0.797]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=0.999, c=0.797]
      [j=15 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [terminate] [j=16 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[72] W - d=0.00 f=1.000 pl=16
      [j=1 , x=6, y=9, t=0.438, f[j-1]=0.089, c=0.999]
      [j=2 , x=1, y=2, t=0.500, f[j-1]=0.437, c=0.779]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.486, c=0.911]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.954, c=0.911]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=0.996, c=0.911]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=15 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [j=16 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.911]
      [terminate] [j=17 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[73] A - d=0.00 f=1.000 pl=17
      [j=1 , x=2, y=4, t=0.429, f[j-1]=0.063, c=0.954]
      [j=2 , x=2, y=4, t=0.429, f[j-1]=0.412, c=0.954]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.428, c=0.937]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=0.964, c=0.937]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=0.998, c=0.937]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=15 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=16 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [j=17 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.937]
      [terminate] [j=18 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[74] S - d=0.00 f=1.000 pl=18
      [j=1 , x=2, y=2, t=0.600, f[j-1]=0.038, c=0.992]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.595, c=0.999]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.999]
      [j=4 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=15 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=16 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=17 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [j=18 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.962]
      [terminate] [j=19 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[75]   - d=0.00 f=1.000 pl=19
      [j=1 , x=2, y=0, t=1.000, f[j-1]=0.203, c=0.959]
      [j=2 , x=2, y=0, t=1.000, f[j-1]=0.967, c=0.959]
      [j=3 , x=2, y=0, t=1.000, f[j-1]=0.999, c=0.959]
      [j=4 , x=2, y=0, t=1.000, f[j-1]=1.000, c=0.959]
      [j=5 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=6 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=7 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=8 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=9 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=10 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=11 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=12 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=13 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=14 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=15 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=16 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=17 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=18 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [j=19 , x=1, y=0, t=1.000, f[j-1]=1.000, c=0.797]
      [terminate] [j=20 , x=0, y=0, t=1.000, f[j-1]=1.000, c=0.000]
[76] G - d=3.36 f=0.098 pl=1
      [j=1 , x=1, y=14, t=0.125, f[j-1]=0.051, c=0.633]
      [terminate] [j=2 , x=0, y=2, t=0.333, f[j-1]=0.098, c=0.099]
[77] O - d=0.05 f=0.964 pl=2
      [j=1 , x=1, y=2, t=0.500, f[j-1]=0.063, c=0.833]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.427, c=0.937]
      [terminate] [j=3 , x=0, y=0, t=1.000, f[j-1]=0.964, c=0.000]
[78] D - d=0.01 f=0.995 pl=3
      [j=1 , x=1, y=3, t=0.400, f[j-1]=0.089, c=0.732]
      [j=2 , x=1, y=0, t=1.000, f[j-1]=0.316, c=0.911]
      [j=3 , x=1, y=0, t=1.000, f[j-1]=0.939, c=0.911]
      [terminate] [j=4 , x=0, y=0, t=1.000, f[j-1]=0.995, c=0.000]
[79] . - d=6.30 f=0.013 pl=0
      [terminate] [j=1 , x=0, y=6, t=0.143, f[j-1]=0.013, c=0.074]
Distinction detected:    151.3426959325
Per character:           1.9157303283
Character count:         79
```

Appendix C.
Sample Distinction Measurements for Selected Texts.

Content	Format	Length	Distinction	Average per Character
John 3	English	4,045	7,325.6	1.81
	Binary (from English ASCII)	32,360	8,035.2	0.25
	English (CAPS)	4,053	7,028.1	1.73
	Greek	6,682	6,645.4	0.99
	Latin (CAPS)	3,394	6,082.3	1.79
Genesis 1	English	4,117	4,914.9	1.19
Genesis 5	English	2,759	2,949.9	1.07
Colossians	English	11,017	20,366.8	1.85
	Binary (from English ASCII)	88,136	21,299.3	0.24
Gospel of John	English	98,288	122,281.3	1.24
DNA	Yeast gene	5,026	8,079.4	1.61
Random Data	Binary	200	160.2	0.80
		500	411.0	0.82
		50,000	41,645.7	0.83
	Hexadecimal	500	1,824.9	3.65
		5,000	17,994.5	3.60
		50,000	178,554.1	3.57

Bible passages were exported from the open-source computer program Xiphos, available from Crosswire at: https://xiphos.org/ . Translations used were: English, King James Version (KJV); Greek, Elzevir Textus Receptus (1624); Latin, Vulgate. For the first two entries regarding John 3, the English export was altered to be paragraph-format with no verse numbers, with a matching binary-formatted text file of chars "0" and "1" based on ASCII values for the corresponding text file. For all other Bible exports, a standard Xiphos export was used with minimal metadata output, and the translation information metadata message removed from the beginning of the text. Random data was taken either from freely available random data sources online or generated using PHP's built-in cryptographically secure random_int function. The yeast gene measured was taken from a sample GenBank database entry available from the United States National Center for Biotechnology Information (available on the website for the National Institute of Health, NIH): 2025, https://www.ncbi.nlm.nih.gov/genbank/samplerecord/ . Note that we only measure distinction here as an exercise; we do not seek to establish the significance of the entire dataset, which would require analysis by someone with the appropriate knowledge of genetics.

Appendix D. Comparative Results of Selected Metrics of Information.

Here, we will compare the information measure of the following messages:

1. The letter "z" repeated 100 times
2. A random sequence of 100 letters and numbers
3. The first 1,000 members of the Fibonacci sequence
4. Text of John chapter 3
5. Phone call with a request

We will compare these messages using four metrics: significant distinction, Shannon information, Kolmogorov complexity, and specified complexity. These four are well suited for quantitative comparison. We will not here consider Coded Information Systems (CIS) because it is a broader measure, extending beyond the coded message itself to include information added by the mechanisms of the receiver, etc. We will also not consider Werner Gitt's definition in this appendix because it is multi-faceted (with five "levels" of information) and not suited for this type of comparison (for example, see Gitt 2000, 124, under question "Q4: Please give a brief definition of information").

The Letter 'z' repeated 100 times.

Summary of measurements:

1. Significant distinction: 0 bits (N/A, no significance, no distinction)
2. Shannon information: 1,040 bits
3. Kolmogorov complexity: <= 248 bits (PHP)
4. Specified complexity: >= 960 bits

Significant distinction

Because this message has no meaning outside itself, it contains no significance. Further, because all the letters are the same, it contains no distinction. It thus contains no information under this metric.

Shannon information

The frequency of the letter "z" (uppercase or lowercase) in English is about 0.074%, or about 1 in 1,350 letters. This is 10.4 bits per occurrence if assuming constant probability distribution for each character. Multiplied by 100, this yields 1,040 bits of Shannon information. (For simplicity, we will not use n-grams for this calculation.)

Kolmogorov complexity

We can output 100 zs in sequence through the following PHP computer program:

```
<?php echo str_repeat('z',100);
```

This program is 31 characters long, or 248 bits if using PHP.

Specified complexity

If assuming the probabilities given by the Shannon calculation above (as 1,040 bits), we then subtract the bits necessary to specify the sequence. We could write the English text, "z repeated 100 times." If using 20 bits per word (which is like how Dembski calculates specified complexity in some examples), we end up with roughly 80 bits to specify this sequence. We subtract this and get a lower bound of 960 bits.

Random Sequence of 100 Characters

Let this be a random sequence of 100 characters with equal probabilities for all English letters (uppercase or lowercase) and numbers.

Summary of measurements:

1. Significant distinction: 0 bits (N/A, no significance)
2. Shannon information: 595 bits (letter frequencies)
3. Kolmogorov complexity: <= 912 bits (PHP)
4. Specified complexity: 0 bits

Significant distinction

Unless we can detect significance in a data set, we cannot measure it for significant distinction. Random text is therefore outside the scope of significant distinction unless some sort of significance can be defined (that is, a randomly generated encryption key, for example).

Shannon information

If assuming a source known to choose randomly between uppercase and lowercase letters and numbers, then the bits of each character will be about 5.95 (26 uppercase, 26 lowercase, and 10 numbers=62 characters; -log2(1/62)=~5.95). Thus, the Shannon information of 100 such characters is 595 bits.

Kolmogorov complexity

Since this data is random, the shortest PHP program capable of outputting the sequence will likely be:

```
<?php echo "[TEXT]";
```

This is 14 characters besides the text itself to be outputted; meaning it is 114 bytes long, or 912 bits.

Specified complexity

Because the text is random, it has no specificity (in the sense meant by specified complexity). Thus, there is no specified complexity, since the bits required to describe the sequence will match or exceed the improbability of the sequence itself.

First 1,000 Members of the Fibonacci Sequence

Summary of measurements:

- Significant distinction: 0 bits (N/A, no significance, literal and not symbolic data)

- Shannon information: 525,000 bits (equal probabilities for 0-9 and SPACE)
- Kolmogorov complexity: <= 592 bits (PHP)
- Specified complexity: >= 525,283 bits

Significant distinction

The Fibonacci sequence, taken alone, is not significant of anything outside itself. (See the section in the paper on literal vs. symbolic data.) Thus, it is outside the realm of significant distinction.

Shannon information

For simplicity, we will assume an even chance for each of the 10 numeric digits (0-9) and the space, making the probability of each character 1/11, or about 3.46 bits. The first 1,000 Fibonacci numbers consist of 151,864 digits total (including spaces separating each number), or 525,363 bits. (By the end of the sequence, numbers are over 300 digits long!)

Kolmogorov complexity

We can write this with a relatively short PHP computer program:

```
<?php $a=0;$b=1;for($i=0;$i<1000;$i++){echo $b."
";$a=$b;$b=bcadd($b,$a);}
```

This is a total of 74 bytes, or 592 bits.

Specified complexity

We could specify the sequence as: "first thousand Fibonacci numbers"; or, using the estimate of 20 bits per word, a specification of 80 bits. Thus, the specified complexity is the Shannon information minus the specification bits, or 525,363–80=525,283 bits (assuming even probability distribution for each digit).

Text of John Chapter 3

Summary of measurements:

- Significant distinction: 7,325.6 bits
- Shannon information: 8,560.9 (using single-word frequency)
- Kolmogorov complexity: <= 17,784 bits (PHP, gzuncompress/base64_decode)
- Specified complexity: 8,500.9

Significant distinction

As Christians, we accept the significance of the entire Bible on faith; and we also experience the significance of much of it through personal experience. Thus, we take the entire text of John chapter 3 as significant. All that remains is to calculate its distinction. Running the text of John chapter 3 through the computer program, we get a measurement of 7325.6 bits.

Shannon information

Shannon information is based on the probabilities of the source, or sender; and thus, there are different ways of calculating it. Using a simple word-frequency method from a freely available database online, the "probability" of each word in isolation was taken, the bits calculated from that, and the results summed. N-grams of varying lengths could be used instead and would reduce the bits measured, provided a calculator was programmed. For this paper, it was programmed to use simple word frequency for simplicity, yielding 8560.9 bits.

Kolmogorov complexity

To calculate this, run the text of John 3 through PHP's "gzcompress" method, and the output of that through "base64_encode." Next, create a PHP program which simply reverses the procedure:

```
<?php echo gzuncompress(base64_
decode('. . .'));
```

A smaller file size could probably be achieved by using raw binary instead of base64-encoded text. As it is, this yields an upper bound of 17784 bits.

Specified complexity

Presuming that the specification of "John chapter 3" is sufficient, using the conservative 20 bits per specification word, this leaves us with 60 bits of specification subtracted from the Shannon information of the text for 8500.9 bits.

Phone Call

(Neighbor) Hi, I am out of town and need to ask you to do me a favor for my dog. Hold on, someone is at the door. Who is it? It's the pizza delivery man, hold on a second. Great, thanks, yes, these pizzas look correct. Here's a tip. You're welcome! Okay, let me set these pizzas down on the table. Okay, I'm back. Could you feed my dog for me tonight? The food is in the garage in the large plastic bucket. Thanks!

Summary of measurements:

- Significant distinction: 648 bits
- Shannon information: 919 bits (single-word frequencies)
- Kolmogorov complexity: <= 3,256 bits (PHP)
- Specified complexity: >= 669 bits

Significant distinction

Identifying the end of this phone call as helping the neighbor's dog, we can eliminate the entire part of the conversation regarding the pizza:

(Neighbor) Hi, I am out of town and need to ask you to do me a favor for my dog. ~~Hold on, someone is at the door. Who is it? It's the pizza delivery man, hold on a second. Great, thanks, yes, these pizzas look correct. Here's a tip. You're welcome! Okay, let me~~

~~*set these pizzas down on the table. Okay, I'm back.*~~ *Could you feed my dog for me tonight? The food is in the garage in the large plastic bucket with the lid. Thanks!*

This leaves us with the text:

(Neighbor) Hi, I am out of town and need to ask you to do me a favor for my dog. Could you feed my dog for me tonight? The food is in the garage in the large plastic bucket with the lid. Thanks!

Running this through our PHP program to perform our calculation, we get a distinction of 648 bits. For a detailed example of performing this calculation by hand using another text, please see Appendix B. Here, we will demonstrate the distinction calculation for a few example letters. To start with, we will analyze the second and final occurrence of the word "dog" (positions 100-102):

- "d": There are 9 d's in a message with a total length of 194; making the expectation of "d" 9/194=0.046 expectation fraction with no prefix. The 1-letter prefix is [SPACE]; in two other cases, [SPACE] is followed by "d", and in the other 39 cases, it is followed by something else (x=2, y=39). Thus the 1-character prefix expectation t=0.071. Using our iterative equation, this adjusts our expectation fraction to 0.064. For a prefix length of 2, or "y[SPACE]", both times this prefix occurs, it results in the same letter, "d"; thus, x=1, y=0, t=1, c=0.954; increasing the expectation fraction to 0.957, per our equation. By the time the maximum applicable prefix length is reached (length 4, since "[SPACE]my[SPACE]" occurs elsewhere), the expectation fraction has reached 1.00, resulting in virtually no distinction for the letter "d" in this instance.
- "o": The same applies to this letter, except that the maximum prefix length reaches 5. The only other time "[SPACE]my[SPACE]d" occurs (earlier), it also results in an "o", thus having virtually no distinction.
- "g": Same (virtually no distinction), except the maximum applicable prefix length is 6.
- . . .
- [Position 183, "l" for "lid"] "l": There are 4 ls (lowercase L) out of 194 total characters; making the no-prefix expectation 4/194=0.021. With prefix length 1, the [SPACE] is followed once elsewhere by the same letter "l", and 40 other times by a different letter; thus x=1, y=40, t=0.048, c=0.633. With a prefix length of 2, x=1, y=6, t=0.250, and c=0.873, since the two-letter prefix ("e[SPACE]") occurs 7 other times, once matching the cursor, 6 times not matching. We continue with this until we reach the maximum applicable prefix length of 5, where x=1, y=1, t=0.667, c=0.960. The six-letter prefix occurs nowhere else in the message, and thus we terminate the iterative function, with the expectation fraction ending at 0.666, and a distinction of 0.59 bits.

When we add together all the bits for each character, we measure 648 bits of distinction total.

Shannon information

If using the single word probabilities, and taking the entire message (since with Shannon information, we do not discriminate based on significance), we calculate 919 bits. Another method would be to calculate the probabilities of each letter based on letter frequency, which would result in a higher measurement; or we could use n-grams, which would likely result in a lower measurement. For simplicity, we use the single-word probability calculation based on open data available online.

Kolmogorov complexity

Using the same method as the previous example (using base64_encode and gzcompress), the program is 407 bytes, or 3,256 bits.

Specified complexity

Presuming that a suitable specification can be found in 10 words, with 20 bits per word as a safe estimate, this would subtract 200 bits, leaving at least 719 bits. Assuming 2^50 other possible messages matching a suitable specification, we are left with 669 as a conservative lower bound. However, this is based on guesswork and depends on the chance hypothesis chosen (e.g., word probabilities, character probabilities, n-grams, or some other chance hypothesis). Yet this rough estimate should be suitable for conceptual comparison.