The World and Mind of

Computation and Complexity

Using the Schafftarian hypothesis to recreate the Turing's Machine in a more complex, but efficient way.

Published by Smashwords

Introduction

It is hard to imagine the virtual world as the world of reality, where objects are tangible and one can these objects with the existence of sight without using mathematical terms and using symbols to describe the world just to see what it looks like. However, the virtual world is a world where nothing exists in observation, but only in the mathematical perception where there are hypothetical identities and analogies which are needed to describe them. At any state, one could look at the virtual world as if it is our world of reality; there are limitations, however with those limitations, ideas and concepts can be used to create wondrous things. Imagine that the virtual world, as you all of the sudden appear in a dark space with nothingness. You can see that everything is darkness, however you can think to create whatever can be created, however you must, with a certain language of mathematics, speak what you want to create.

Schafftarian Hypothesis

A specific language is needed to create something for if there is no language in existence, how can one determine what to create in the first place? The cyber world, as it is referred to in common day society, works the same way where it relies off a common mathematical language to carry out its daily tasks. On of the languages used is called Binary. Binary, consisting of only ones and zeros, is a mathematical computerized language that is used to calculate values and carry out other tasks, like how machines carry out the processes that they are programmed to do for human society. Mathematics itself, which the cyber world relies on, is a world that does not consist of concrete objects, but of hypothetical objects that represent the objects that exist in reality.

In this way, we can represent language as a concrete object, the dimensions more specifically, which aren't really concrete concepts, but yet are things that exist in reality. There are one dimensional languages, two dimensional languages, three dimensional languages, and so forth with the other languages existing in other dimensions yet not understandable by our current human knowledge. For example, binary is a one dimensional language, where it only works with one layer considering the *Schafftarian hypothesis*.

$$S_h \Rightarrow [0, 1, 1, 0, 1, 0, 0, 0]$$

One example of existent hypothetical machines that use this one dimensional language is the Turing's Machine. The Turing's Machine relies on only working on one dimension, which means that it can only carry out only one process per exponential process that is carried out. Computers use this type of method, only carrying out one process per exponential step. In our current times, the improvement of computers only leads to computing faster these exponential steps by speeding up how fast it takes to carry out these steps. However, it isn't really the focus

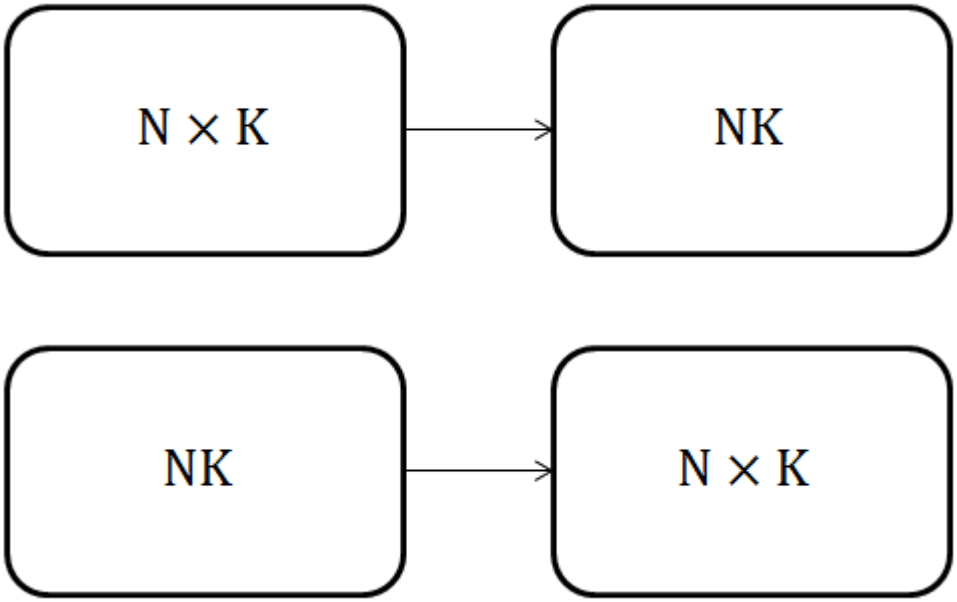
of discussion when carrying out multiple steps per process. This means having a language that works on multiple dimensions. Here is an example of a multiple dimensional language using only ones and zeros.

$$S_a \Rightarrow [0,1,1,0,1,0,0,0]$$
$$S_b \Rightarrow [1,0,1,1,0,0,0,0]$$
$$S_c \Rightarrow [1,1,0,0,0,0,1,0]$$
$$S_d \Rightarrow [0,1,0,0,1,0,1,0]$$
$$S_e \Rightarrow [0,1,1,0,1,0,1,1]$$
$$S_f \Rightarrow [0,0,1,0,1,1,0,1]$$

Now, if one were to use the Schafftarian matrix algorithm to make it appear on a one layered scale, here is what the result would be.

$$S_h = \begin{vmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & & \end{vmatrix}$$

With the applied algorithm to the sequence, the sequences becomes more complex in its structure, however there is a way to reverse the doings of the algorithm by using the inverse of the Schafftarian matrix algorithm, which essentially is taking each sequence element and carrying out the algorithm to bring it back to its original state. This, even though with its simplicity, could reverse a problem that was earlier solved. For example, if one was to take two numbers, *n* and *k*, and these numbers were multiplied together to get a specific value you could also take that specific value to get the inverse. Therefore, you would be able to complete the inverse in the same amount of time it took to solve the problem in the first place.



As seen here, *n* and *k* are the representations of the factors of the solution. However, it is not as simple as this. Since *n* and *k* can represent multiple numbers, that means there can be multiple solutions to the inverse of the problem. However, with the matrix algorithm each solution, though very similar, is unique to its two factors. The factors, in this case, do not refer to the values that are represented, but by the matrix factor that represent the values. For applying the Schafftarian algorithm to the problem, the variables *n* and *k* are to represent the matrix representations of the factors. Though, these two variables will not multiply together as matrices would be.

When multiplying the two matrices, in this case, the Schafftarian algorithm must be applied where instead of multiplying the elements of *n* and *k*, the corresponding elements actually are combined as two elements. Once that is completed, the Schafftarian matrix algorithm will be applied again with the new sets, to get the solution. To retrieve the factors of the solution, one merely carries out the inverse of the algorithm and applies it to the solution get the two factors

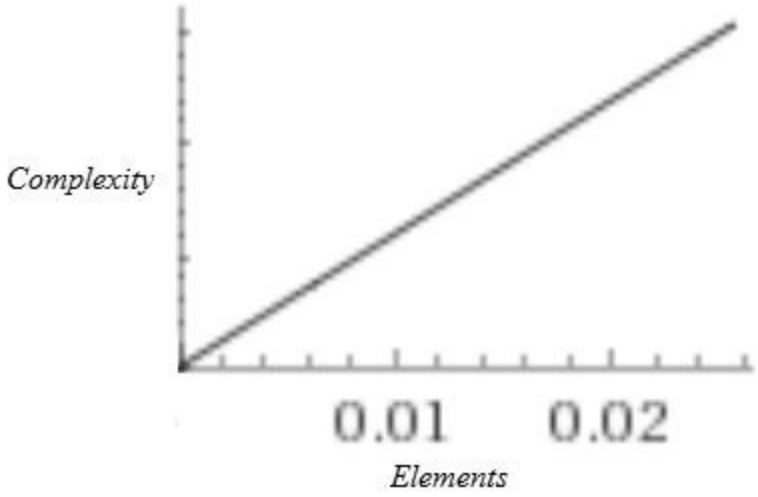
that were involved in the solution. Therefore, finding the solution and finding the factors of the solution can be done in the same amount of time. However, a problem with this method is finding multiple ways to get the same solution.

With the difficulty of finding the multiple implications of different elemental differentiations, this method faces a problem. Since one solution should have multiple factors, especially for problems, such as multiplications, this method may not work due to the strictness to its bounds. With this ideal in mind, there could be a way to solve this issue. This would be to allow solutions to have multiple outputs, or expanding the bounds of which a solution can be found. This would mean allowing certain solutions to have multiple interpretations. However, this would make the problem more complex, therefore ruining the point of a simple solution to an issue, which is to find the factors of a multiplication solution. One way to solve this complexity solution is to increase the amount of levels, or layers, that the problem is solved on. With the Schafftarian hypothesis, there is the Schafftarian matrix, which is represented with $S_{n\times k}$, where the variable *n* symbolizes the number of layers of levels and the variable *k* symbolizes the length or complexity of the problem. To determine the complexity of the problem, this formula would be used.

$$S_{n\times k_c} = \frac{E_t k}{n}$$

This formula states that the complexity of the Schafftarian matrix *S*, with its dimensions being *n* by *k*, is equal to the product of the exponential time needed to solve the problem and the amount of elements within the problem divided by the amount of levels or layers used to solve the problem to reach the solution. If complexity is linear, this formula would predict complexity because with complexity being linear, that means there is a pattern within complexity, which

means there should be a pattern within pattern within probability as well. However, if complexity is non-linear this would also mean that probability is non-linear. This means that complexity and probability is unpredictable, although it seems that both are linear at the time. The graph below represents the linear growth of complexity if there is one layer or level within the Schafftarian matrix.



As seen in the graph, with the increase of elements within a six leveled or layered problem, there is an increase of complexity within the problem. In order to lower complexity, the amount of elements has to be lower than the amount of layers of the problem, which is represented by $n > k$. This has correlation with P and NP problems.

In order to determine whether the value of S_c is either classified as being a *P* problem, a problem that can be verified and solved within exponential time, is equal to an *NP* problem, a problem that can be only verified in polynomial time, a scale is used. In this scale for complexity, there is a range from zero to infinity. If the Schaffter Complexity, also abbreviated as *sc*, is less than 100,

then it is to be classified as a *P* problem. However, if *sc* is either equal to or larger than 100, it is to be classified as an NP problem.

$$S_c = P \leftrightarrow S_c < 100$$
$$S_c = NP \leftrightarrow S_c \geq 100$$

Therefore, this means that *P* cannot always equal *NP*. This is due to the fact that with the increase of elements or levels, exponential time will also increase no matter the case. However, there is always the possibility that the Schafftarian complexity value is equal for both the P and NP problem even with the increase of elements and levels. However this type of circumstance is unknown currently or may not exist at all. The reason is explained mathematically, where the exponential time has to be proportional to the amount of levels and elements and being proportional to the machine that is carrying out the process. There is also a formula to include the factor of speed that the machine carrying out the process, which would be the following:

$$S_c = \frac{E_t km}{n}$$

With the prior definition of the variables within this formula, the variable *m* symbolizes the average speed that the machine carrying out the process has. The original formula ignores the idea of machine speed, therefore generalizes complexity, assuming that all machines carry out the process at the same exact speed. However, this formula assumes that all machines run at different speeds, therefore making a more accurate prediction based on the machine itself. However, if one were to try to calculate the general complexity of a problem, using the general formula would be more accurate due to how even fast machines have to carry out the same amount of steps in order to complete the process the user wants completed. Therefore, it is more accurate to use the first formula because it makes a depiction of complexity from the

perspectives of all speeds rather than the speed of one specific machine. Complexity, if to be useful in analysis, needs to be looked at without the speed of the machine, but with what problem is being solved by the machine and the exponential time needed to complete the process that is needed to solve a problem.

In order to be able to use the Schaffter Complexity formula, there has to be a way to calculate the exponential time needed to solve or evaluate the problem. In order to do this, the exponential time formula would be needed, if applied to the Schafftarian matrix.

$$E_t = \frac{kn^2}{2}$$

This formula, however, implies that each level or layer of the Schafftarian matrix is equal amounts of elements. Therefore, if the amount of elements in each level or layer is unequal, that means this formula retrieves the approximate exponential time needed to carry out the full process in order to solve or evaluate the problem to find the solution. In this case, if the variable E_t is replaced with its formula; here is what the mathematical formula for Schaffter Complexity would be.

$$S_{n \times k_c} = \frac{E_t k}{n}$$

$$S_c = \frac{k(\frac{kn^2}{2})}{n}$$

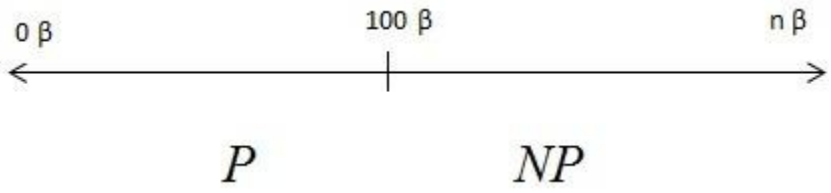
$$S_c = \frac{\frac{k^2n^2}{2}}{n}$$

$$S_c = \frac{k^2n^2}{2} \times \frac{n}{1}$$

$$S_c = \frac{k^2 n^3}{2}$$

This version of the formula, however, assumes that the exponential time needed to solve or evaluate a problem is predictable by the exponential time formula. In any case, not all things are predictable, for there are unexplained phenomena, such as being able to solve a problem in different ways. If this is the case, this means that predictability is non-linear in complexity. With non-linear complexity, it is implausible to make an accurate mathematical formula or equation to predict complexity or be able to predict probability because probability follows similar principles as complexity. Therefore, this would mean that *P* can, but not always equal *NP*. This would bring about unpredictability within complexity because if it cannot be predicted whether a problem is a *P* or *NP* problem, there is no way to prove whether *P* equals *NP* or not. In such a case, this means that there is no mathematical way to predict complexity or probability, therefore it only can be represented by formulas and equations that can make far approximations of a problem with a value of complexity or probability. However, there are ways to make equations or formulas that in some ways make close approximations to real values. In any case, with the unpredictability of complexity and probability, it renders almost all equations or formulas that exist to be only accurate to a certain extent determined by the complexity of the problem. Even with equations for the mass of objects and the gravitational pull of planets and celestial bodies, there is a great deal of accuracy and a great deal of approximation. This kind of unpredictability can be seen within quantum mechanics, especially with Heisenberg's uncertainty principle. The more complex the problem becomes, the less accurate a formula or equation becomes. Within the bounds of a problem, the formula needed to find the solution becomes more complex as the size of the problem increases. With this idea of complexity, two sections of complexity can be made

to classify certain problems, which are known as the *P* and *NP* classification, as explained before.



With this scale, beta represents *sc*. Having zero beta would not normally be used in a problem because in order to solve a problem it requires some form of work in order to process the solve or evaluate the problem in the first place. However, since this scale applies to all instances, zero beta is included on the scale. In order for this scale to work with complexity and probability, there has to be the assumption that complexity and probability are both linear. The reason for this is due to the way complexity values work, where proportionality is an important factor of complexity values. With a non-linear complexity system, predictability upon a complexity scale would be almost useless due to the non-linear nature of complexity values, thus rendering complexity measurement useless. Due to the lack of evidence or proofs leading to the conclusion of a non-linear system, it must be assumed at the time that complexity and probability are linear and proportional to their factors.

However, if it is true that complexity and probability are non-linear, then the predictability of events and values is mainly due to coincidence and sensitivity of a specific item to be solved. Quantum mechanics is one area in which sensitivity is a major issue with the equations and formulas that are used in that particular field. With complexity and probability being rendered non-linear, equations and formulas of all scientific fields would be rendered only having the ability to predict the unpredictable with only such accuracy that the “needed” values are reached,

however the accuracy is only leading to an approximate solution. Though this would be the downfall of non-linear complexity, the good thing with non-linear complexity and probability is *P* can, in certain circumstances, equal *NP* if the right factors are met, where the complexity value is low enough to be considered a *P* problem.

The 100 beta is what separates each side of the complexity scale, with each side of the scale being off balanced because above 100 beta complexity can reach infinity. One question, however, is whether a problem with infinite complexity can be evaluated or solved. If a problem had infinite complexity, this would mean that it had either infinite exponential time, rendering the problem improbable so solve, having infinite elements, or even infinite levels or layers.

$$S_c = \frac{k\infty}{n}$$

$$S_c = \frac{E_t\infty}{n}$$

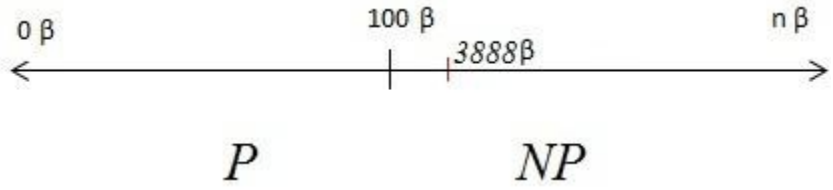
$$S_c = \frac{E_t k}{\infty}$$

The one extraordinary thing about this type of situation is how the formula works with infinite levels or layers. If there are infinite levels or layers within a problem, meaning that the variable *n* is equal to infinity, if the exponential time or element count is not equal to infinity, this means that the complexity value would be 0, which means that a problem being worked on has no complexity at all. These following conditions, however, are not always possible because all variables have to be proportional to the problem.

$$S_h = \begin{vmatrix} A_1 & B_1 & C_1 & D_1 & E_1 & F_1 \\ A_2 & B_2 & C_2 & D_2 & E_2 & F_2 \\ A_3 & B_3 & C_3 & D_3 & E_3 & F_3 \\ A_4 & B_4 & C_4 & D_4 & E_4 & F_4 \\ A_5 & B_5 & C_5 & D_5 & E_5 & F_5 \\ A_6 & B_6 & C_6 & D_6 & E_6 & F_6 \end{vmatrix}$$

$$E_t = \frac{6(6^2)}{2} = 108 \rightarrow S_c = \frac{6^2(6)^3}{2} = 3888$$

$$S_c = 3888\beta sc$$



With the given work, this means that a problem done with six elements on a six layered Schafftarian matrix, the complexity would be 3888 beta. Therefore, the example problem would be an *NP* problem, since the complexity value is larger than 100 beta. With these formulas, different instances can be predicted, such as the amount elements that were in the initial problem or how many layers or levels were used to complete the problem and evaluate or solve it. This example, however, does not include the factor of solving speed. The following formula would be used to get exponential time that includes the factor of machine speed.

$$E_t = \frac{kn^2}{2m}$$

Therefore, if one were to want to include machine speed into the complexity formula, the following would be the result.

$$S_c = \frac{k(\frac{kn^2}{2m})}{n}$$

$$S_c = \frac{\frac{k^2n^2}{2m}}{n}$$

$$S_c = \frac{k^2 n^3}{2m}$$

In this case, the complexity value using this formula cannot be the exact value, but only the approximate because machine speed is not part of exponential time. Now, the question is what is the speed of a machine? The speed of a machine is the average time it takes to solve a set of problems with certain amount of elements. The speed of a machine is as follows:

$$m = \frac{e}{A_t}$$

In this formula, machine speed, represented by *m*, is equal to the amount of elements in the set of similar sized problems, which is represented by *e*, divided by the average time it takes to carry out these problems, which is represented by A_t. This formula, however, can only approximate the machine speed due to unpredictable factors, but can give enough accuracy to give a good complexity value. When combining the formula for machine speed and formula for complexity, here is the following result.

$$S_c = \frac{k^2 n^3}{2m}$$

$$S_c = \frac{k^2 n^3}{2(\frac{e}{A_t})}$$

$$S_c = \frac{k^2 n^3}{\frac{2e}{A_t}}$$

$$S_c = \frac{k^2 n^3}{1} \times \frac{A_t}{2e}$$

$$S_c = \frac{A_t k^2 n^3}{2e}$$

With these sets of formulas, many different things can be predicted. For example, the following formulas represent their corresponding descriptions.

$$n = \frac{\sqrt[3]{2S_c k}}{k}$$

This formula represents the *n* variable, which is the width of the Schafftarian matrix.

$$k = \frac{\sqrt{2S_c n}}{n^2}$$

This formula represents the *k* variable, which is the height of the Schafftarian matrix.

$$2 = \frac{k^2 n^3}{S_c}$$

This formula is a proof formula, which allows one to prove that any given solution is accurate for that specific situation.

$$S_{ca} = \frac{\sqrt{2S_c n}}{n}$$

This is one formula to obtain the area of a Schafftarian matrix problem.

Most of the formulas presented here are ways to obtain values and predict complexity values. These can be useful if one is given the complexity value and one of the dimensions. For example, if the goal is to obtain the value of *k*, which represents the amount of elements within a given problem. A sample of using these formulas is if the given complexity value is 1008, where the amount of layers or levels of the problem is 6, the following would be done.

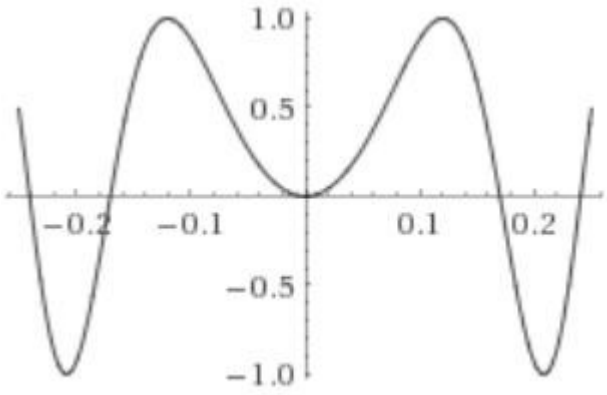
$$k = \frac{\sqrt{2(1008)(6))}}{(6)^2}$$

$$k = \frac{\sqrt{12096}}{36}$$

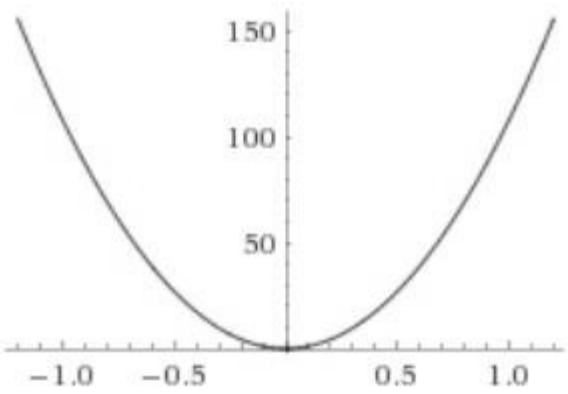
$$k = 3.055$$

Since *k* is 3.055, this means that there are approximately three to four elements within each layer or level of the problem. In order to obtain the whole amount of elements within the problem, multiply the approximate *k* value by *n*. This means that there are approximately 18 elements within this problem. This is based on exponential time because exponential time is a determining factor of the problem and how fast it is solved in general. Again, these formulas assume that complexity and probability are linear.

The only thing to find from here is determining whether complexity and probability are linear or non-linear. If complexity is linear, $P \neq NP$ would be proven to be correct, where if complexity is non-linear, this would mean that *P* does, in fact, equal *NP* in certain circumstances.



Non-linear graph of complexity



Linear graph of complexity

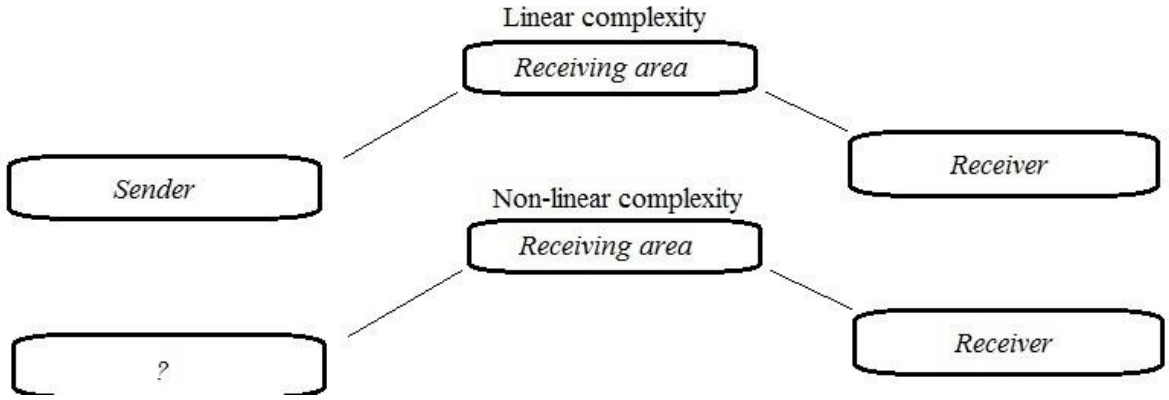
With the linear version of complexity, the complexity value increases on a slope that doesn't change, therefore the limitation of complexity lies under a certain bounds. However, with non-linear complexity, there are infinite arrays of problems that are *P* problems. With non-linear complexity, predictability is unpredictable, especially since there are an infinite amount of equations and formulas that fit within the *P* and *NP* problem spectrum. Within that spectrum, there are, however, areas of negative complexity. This doesn't seem possible because negative complexity would mean that there are problems that can "solved themselves". This seems impossible because in order to solve a problem there needs to be an intelligence that solves the problem in the first place. This is why it seems least likely that complexity is non-linear due to its improbable nature of complexity. With linear complexity, there is an existence of proportionality, which means that the complexity value will correspond with only one set of values for *n* and *k*.

One major difference between linear and non-linear complexity is the existence of cause and effect. Linear complexity states that cause and effect must exist due to proportionality in its

nature and the way it works through systems of probability. On the other hand non-linear complexity states that cause and effect are not needed to complete a specific set of problems. With lack of term, these problems can be looked as "imaginary" problems because they lie within the spectrum of "non-existent" problems. In any case, there also can be a specific system where both linear and non-linear complexity exists. This would mean that there are problems that need an intelligence to solve while there are problems that do not need an intelligence to solve. However, the reason that "imaginary" problems don't seem to exist is due to how existent problems cancel out these "imaginary" problems. This would the same as how there is anti-matter and matter. Since matter exists, anti-matter is hard to find because of its existence with matter. If it is non-linear complexity that exists, this means that in everyday life there are "imaginary" problems. This assumes, however, that there are infinite *P* problems within complexity. In order for infinite *P* problems to exist, non-linear complexity would have to be symmetrical. However, currently there is no proof that non-linear complexity is not symmetrical, since there is a seemingly infinite array of problems that fall within the *P* spectrum of problems. Therefore, if complexity is non-linear, this would mean that there are infinite amounts of *P* problems within the spectrum of problems within complexity and probability. The only real known way to prove that non-linear complexity does not have an infinite array of *P* problems is to form a function that represents more accurately the form of non-linear complexity.

One other issue with non-linear complexity is the idea that with legitimate information to a receiver must always have a sender. With non-linear complexity, for some problems there is no need for a sender, but only a receiver, which is a representation of the solution. Linear complexity supports the existence of a sender in order for there to be a receiver. With this in mind, if non-linear complexity is proven to be the correct form of complexity, this would also

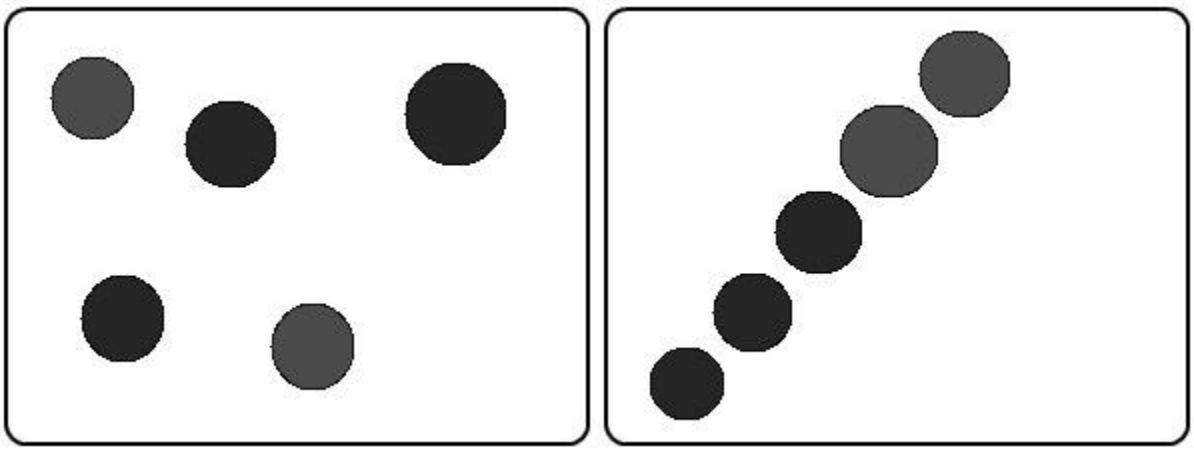
prove the existence of one-way functions. With one way functions, there is a way to retrieve the solution, but then not be able to retrieve the problem itself. Non-linear complexity would be the same concept, where there is no sender of the received information, therefore there is no way to reverse the process of finding the problem using the solution of the problem.



For linear complexity, the sender can be found easily with using the received problem. However, for non-linear complexity, it can only be predicted whether there is a sender of the problem or none at all. With this in mind, it would be improbable to determine whether there was a sender of the problem or not. If non-linear complexity is the correct form of complexity, that would render all solutions unable to link themselves to a sender at all because there would be no known way to determine if there is a sender of the solution at all. This would mean that everything dealing with mathematics is a concept in order to understand a truth that is unable to be found because there is no real way to prove that the mathematical solution come from the formulas or equations that are known today in mathematics and science. This would also mean there would be no way to prove that current mathematical formulas or equations are correct in the first place, but these equations and formulas only will give an approximation close enough to the true solution of a problem. This is one of the many connections between non-linear complexity and quantum mechanics. - Many of the equations and formulas seen within the field quantum mechanics deal with mainly

predicting values and finding probabilistic locations of particles. Quantum mechanics mainly deals with predictions and probability, using predictions to find the locations of specific particles within an area. Non-linear Complexity can be viewed the same way as quantum mechanics, where equations and formulas can only predict the outcome of specific problems, and being unable to retrieve data from the original problem.

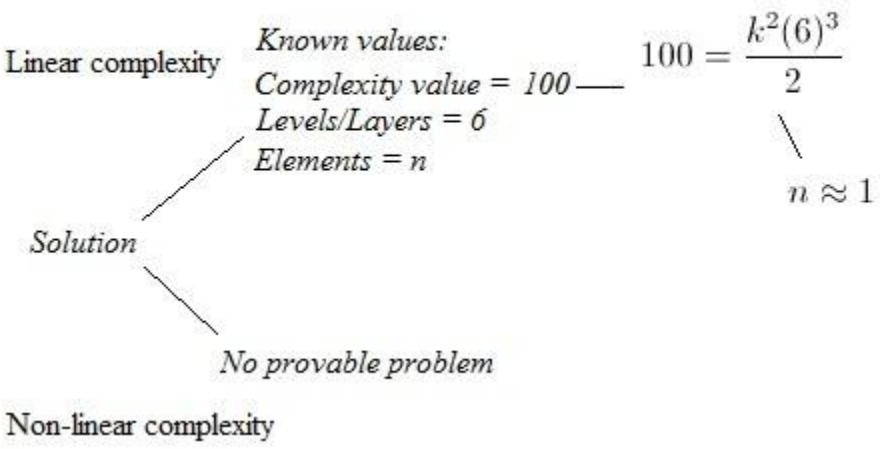
It is also difficulty to determine whether a value is a solution to a problem if complexity is non-linear, since it would require predictability or probability to find an approximate solution. With there being one-way functions within non-linear complexity, there would be no apparent way to prove that a solution came from a specific problem unless it is known already that the solution belongs to that problem. However, if non-linear complexity is the true form of complexity, this would not mean that chaos exists within complexity and probability. Chaos assumes that there is no order within information and within problems and solutions. With non-linear complexity, there is an order that exists, but is not as apparent as it would seem for linear complexity.



Non-linear complexity *Linear complexity*

With non-linear complexity, there is no apparent form of order. With the chaotic view of non-linear complexity, there is no simple way to form a function that represents the positioning and properties of objects in that position. Non-linear complexity would have to follow predictability rather than accurate solutions to a problem. Linear complexity follows a different mathematical function of finding solutions to a problem and goes against non-linear complexity. With linear complexity, a function would be possible to form that describes complexity, being able to form formulas and equations that can accurately give a solution to a problem or set of problems. Finding the function for linear complexity could even help predict equations for science and mathematics, which is being able to predict and prove whether an equation is correct or not by determining if it follows along with the function of linear complexity. The issue with linear complexity, however, is how it contradicts what is known about the Universe, especially with the field of quantum mechanics. Since mathematics is all based off of the patterns and order of nature, there must be a connection between complexity and reality. Therefore, non-linear complexity would seem more likely in this case because of its connection to quantum mechanics. Many of quantum mechanic's formulas and equations rely on predictability and follows along with the philosophy that everything can only be determined through predictability. This would also mean that complexity and probability are either identical or even the same thing. This would combine the concepts of probability and complexity, therefore identifying solutions as only predictions of what exists in reality. It would also mean that *P* does and does not equal *NP*. In certain cases, determined by the graph of non-linear complexity, there are certain cases where *P* does equal *NP* while in other instances *P* does not equal *NP*. This is a similar concept in quantum mechanics, where it can be said that a particle is in a certain location while it is also not in that same location. A similar concept applies to complexity, where *P* can equal *NP* while it also

cannot equal *NP* at the same time. With this idea, it will be difficult to prove that complexity is non-linear because there is no specific point where non-linear complexity can be proven with. Also, with non-linear complexity, proportionality would be rendered only coincidence because of its unpredictable nature. In nature, it is proven that proportionality is very common with natural entities, especially on cosmic scales. With linear complexity, proportionality is a valid because with linear complexity the complexity value is proportional to the amount of elements in the problem and the levels or layers within the problem.



Information tree for complexity, both of linear and non-linear complexity

An information tree can be developed to show the main differences between linear and non-linear complexity. The information tree can be formed the same way as the sender and receiver chart can be formed. Linking solutions to their specific problems is part of linear complexity and being able to find what the original problem was using the solution. Non-linear complexity states that the original problem must be known in order to determine the source of the solution.

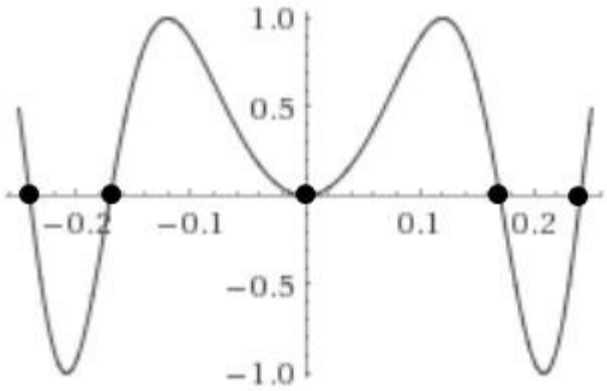
Modern day thinking of the world, however, seems to agree that linear complexity is the true form of complexity because everything can be broken down into other forms or substances that

exist. For example, protons of atoms can be broken down into other elementary particles that exist. Solutions to problems, *P* or an *NP* problem, can be also broken down into smaller pieces. This, however, is not the point that is derived from the argument of whether complexity is linear or non-linear. The real point of arguing the existence of a linear or non-linear complexity is whether *P* and *NP* can be equal. With the complexity formula being involved, here is how *P* equaling *NP* would look like.

$$P\frac{E_t k}{n} = NP\frac{E_t k}{n}$$

With the linear complexity model, this equation would show that *P* and *NP* cannot be equal because the values must be proportional to their specific problem. Therefore, linear complexity argues against the idea of *P* being equal to *NP*. In order for these two to be equal, complexity would have to be non-linear, being that the values are not proportional.

$$E_t kn = E_t kn$$



With the existence of non-linear complexity, the formulas would have to include variables of unpredictable values. This would mean the formula would have to be modified in a way that includes all factors involved within the non-linear complexity.

$$S_c = \frac{C E_t k}{n}$$

In this non-linear complexity formula, *C* represents the unpredictable factor within non-linear complexity. Even with an unpredictability variable *C*, both *P* and *NP* would not be equal in certain circumstances because the unpredictability variable would have to be equal on both sides, which would cause the gap of difference between *P* and *NP* no bigger or smaller.

$$S_c = P\frac{C_1 E_t k}{n} \qquad S_c = NP\frac{C_1 E_t k}{n}$$

$$S_c = P\frac{C(6 \times 12)}{6} \rightarrow C = 10 \qquad S_c = NP\frac{C(6 \times 1200)}{6} \rightarrow C = 1$$
$$S_c = 1200 \rightarrow C = 10 \qquad S_c = 1200 \rightarrow C = 1$$

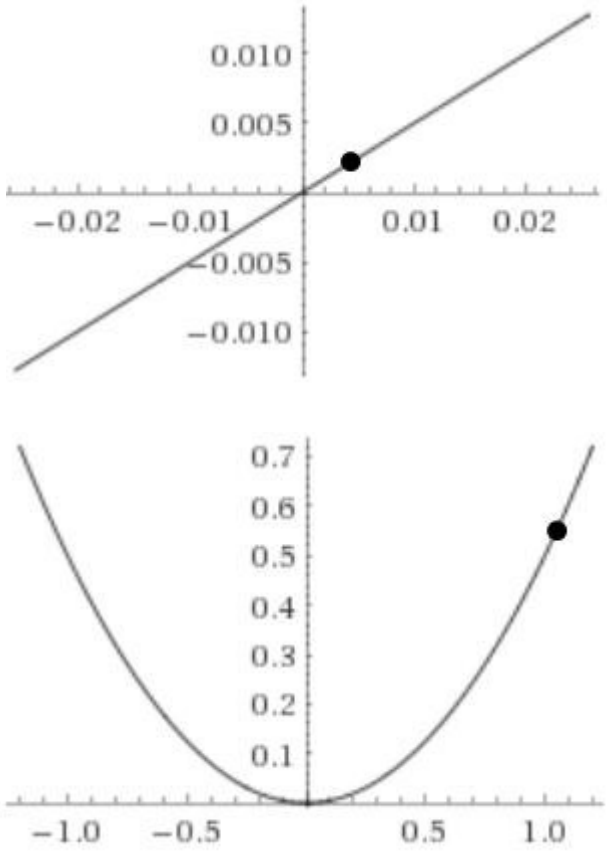
Therefore, the unpredictability variable could equalize these two types of problems. However, the question would be what this unpredictability variable represents. With non-linear complexity, there exists infinite ways to solve a problem. This means being able to use different methods to find the solution to a problem, which then ultimately defies the idea of sequential complexity, meaning linear complexity. The only problem with this idea is how each method of solving a problem must have similar build up as the other. For example, there may be multiple ways to calculate the value of π, yet each way to calculate the value follows along the same path as the

other methods, such as using infinite series or using integrals. Though infinite series and the use of integrals are particularly different methods, they still fall along the lines of sequences that can be used to calculate more accurately the value of π. This system applies to non-linear complexity. Non-linear complexity provides insight into an infinite realm of ways to solve problems that lie within the spectrum of *P* and *NP* problems. Linear complexity, on the other hand, goes along with the philosophy that there are problems that only can be solved with certain methods, formulas, and equations. Many fields also follow along with this idea of exact formulas and equations of linear complexity such as general physics. In general physics, most equations and formulas supposedly give accurate depictions of mathematical concepts of nature and the many forces that exist within reality. However, the problem with this notion of accuracy is how inaccurate the values are compared to exact values. For example, the equation for gravity, provided by Isaac Newton, is the following:

$$F = G\frac{M_a M_b}{r^2}$$

His equation is approximately accurate; however it is not an accurate equation as other physicists have proven by forming other equations, such as Albert Einstein's field equations dealing with gravity. With the advancement of science, more accurate equations are found and form better ways of predicting natural values. However, all these equations have one thing in common; they are all approximations. Some may argue that even though they are approximations to real values they are still within the lines of being correct values due to the precision needed to make an accurate value considered a valid value. However, the reason for their accuracy is due to the other values that help make approximations to exact values. An example of this would be imagining a circle of a particular size on a graph, where on the graph this circle moves on a slope of *m* on the graph. One can make a formula that predicts the slope of *m* by determining the

positions of the circle with certain values of *x* or *y*, however the accuracy of that formula or function may be very slim. Questions about the problem could arise where the circle's slope may be changing very *n* changes in the *x* value.



With particular functions with this specific problem, it would be hard to predict the location accurately for this circle due to the lack of information that is given to solve such a problem. Since the problem has the given that the circle is increasing by *m*, with the increase of *x*. With an increase in knowledge of the path of this circle, more accurate formulas or equations that predict the path of this circle may be developed, but yet the formula or equation that exists for this type of problem is still an approximation of the path of this circle because of other factors becoming involved that may not be known to the one who is solving the problem to come upon a solution

and evaluation of the path of the circle. Example formulas that could be formed from are the following:

$$y = \frac{1}{2}x$$

With this formula that follows the $y = mx + b$ formula, there is a somewhat accurate approximation of the path of the circle within the problem. However, a better and more accurate approximation can be formed from the following formula:

$$y = \frac{1}{2}x^2$$

This is a more accurate approximation of the path that the circle is following, but there is an even more accurate approximation using infinite series.

$$P = \sum_{n=0}^{\infty} \frac{1}{2}x^2 = \frac{1}{2}(0)^2 + \frac{1}{2}(1)^2 + ... + \frac{1}{2}(n)^2$$

With the continuation of being able to find more and more formulas and equations that can provide more accurate approximations of the path of this circle, the problem becomes more complex. One could make the case that since there are more formulas and equations that become more accurate than the previous formula or equation, which then leads to the conclusion that there must be an infinite array of formulas and equations that more accurately predict the solution to the problem. Therefore, this would give the assumption that complexity is non-linear.

In order to determine the accuracy of a formula or equation through complexity, there is a formula to determine the accuracy value, which is an approximation of accuracy. Of course, this assumes that the true path of the circle is actually known to the one who is solving the problem.

$$A_c = \frac{D_p}{S_c}$$

The accuracy value, represented by A_c, is equal to the quotient of the decimal places that are accurate, represented by D_p, and the complexity value that is calculated when calculating the complexity of the calculations to get the decimal value, represented by S_c. To apply this, an example would be an equation or formula for calculating π. One equation to calculate π is to use the following equation that uses the infinite series, but with *k* equaling ten:

$$\pi \approx \sum_{n=0}^{k=10} \frac{4}{n} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \ldots + \frac{4}{n}$$

In order to calculate accuracy, the amount of decimals accurate to π would have to be counted. In this case, the amount of accurate decimals within this equation is one.

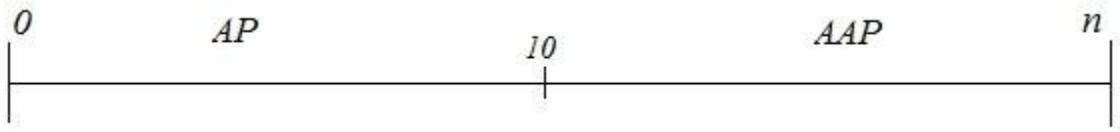
$$A_c = \frac{1}{\frac{10^2(1)^3}{2}}$$

$$A_c = \frac{1}{\frac{100}{2}}$$

$$A_c = \frac{1}{50}$$

$$A_c = 0.02$$

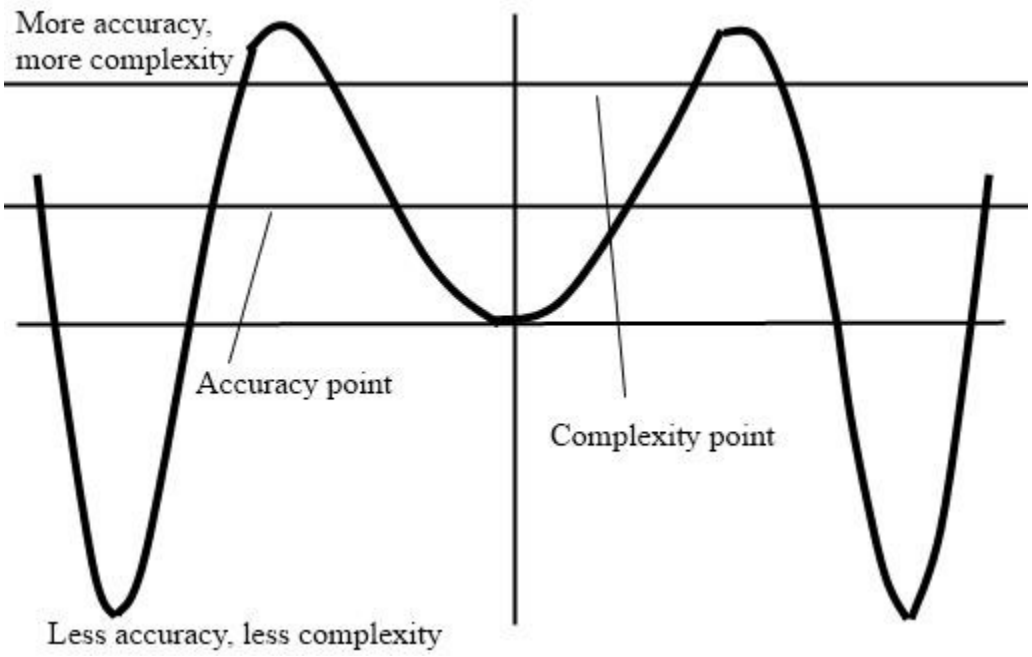
In this problem, the accuracy of this equation is *0.02*. This, however, only applies to this series because the amounts of steps taken were only ten. For an infinite series, however, the accuracy would also be infinite because of the amount of steps taken is, in fact, infinite. It may be confusing to some because accuracy may be depicted as a measurement of complexity. It is in some cases, but the value of accuracy does not show specifically how complex a problem is, but how accurate the solution is compared to the problem. With calculating accuracy, it also can be measured on a scale, and with that there are two specific spectrums of accuracy. These two kinds of accuracy are *AP* and *AAP*, where *AP* stands for approximation, while *AAP* stands for accurate approximation. The determining factor between whether a solution is AP or AAP is whether the accuracy value is greater than ten.



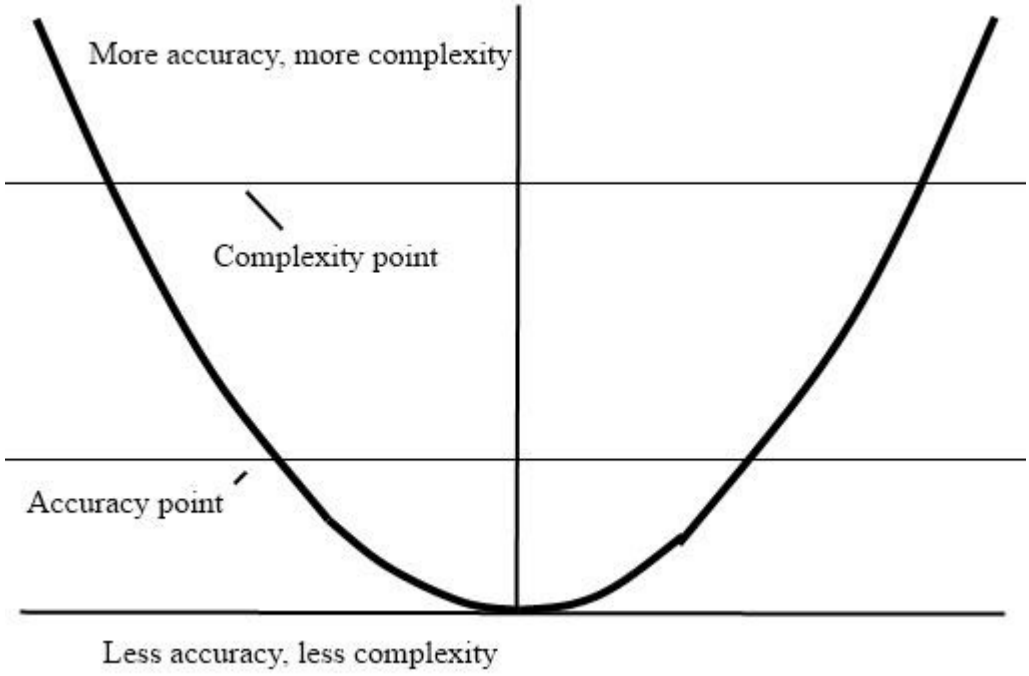
With the calculation of π, the equation, in the specific case it was put in, would belong in the *AP* spectrum of accuracy because it is below the value of ten. Now, it may be weird for zero to exist on an accuracy spectrum because zero accuracy would seem irrelevant because the assumption is if it had zero accuracy it couldn't be considered an approximation. However, this is a false assumption because even with a value of zero accuracy there are some distinct similarities with the zero accuracy approximation and an accuracy value that may be higher. For example, with π,

since the number is irrational and goes on infinitely, there is no way for the accuracy value to reach the *AAP* spectrum because the number is infinitely large in decimal length. Accuracy values and the spectrum of *AP* and *AAP* have a close connection to the *P* and *NP* spectrum. This connection deals with how there is to be a proportion between accuracy and complexity, especially with linear complexity. As weird as it sounds, there could also be the same connection within non-linear complexity. The uniqueness does not come from within non-linear complexity, but with the connection between non-linear complexity and proportionality. Non-linear complexity seems to deny the existence of a proportion between the *P* and *NP* problems, however it does, in many cases, have a proportion with accuracy and approximation. This, in itself, is counter-intuitive due to the nature of non-linear complexity. Non-linear complexity follows along the lines of being able, in certain cases, to have both *P* and *NP* problems equal, providing insight to the realities of complexity where accuracy cannot be determined because of complexity is non-linear that means there is no way to define accuracy. In non-linear complexity, however, accuracy is a big standing point within this type of complexity because it relies on having accuracy as determining whether *P* is equal to *NP* or not. With non-linear complexity, it can never be known whether *P* is exactly equal to *NP* because accuracy cannot be determined within non-linear complexity. Therefore, the accuracy value is a big part of non-linear complexity. Non-linear complexity provides the idea that there are infinite areas whether *P* can equal *NP*, which then leads to the concept that *P* equaling *NP* is not an accurate, but approximate statement about the two spectrums of complexity. This is due to how sometimes *P* and equal *NP*, while at other times they cannot be equal. This is specifically shown by the non-linear complexity graph. At times, *P* can equal to *NP* with precision, however at other times *P* cannot equal *NP*. This idea of non-linear complexity and its correlation with accuracy also follows along

the lines of quantum mechanics. Accuracy is not seen much in quantum mechanics because one does not think of the accuracy as a way to determine the approximation of a value, or in the case of quantum mechanics the location of a particle. However, accuracy values are a big concept in quantum mechanics because quantum mechanics follows along the lines of non-linear complexity and yet there are times when predictions lead to accuracy that is higher than other values of accuracy. This means that quantum mechanics is a big part of non-linear complexity, where accuracy deals with the fact that *P* can sometimes equal *NP*, but in other cases it cannot equal *NP*.



Non-linear complexity graph comparison to accuracy



Linear complexity graph comparison to accuracy

With non-linear complexity, there are infinite times where the complexity value passes the complexity and accuracy point, while with linear complexity passes both the complexity and accuracy point are passed by the complexity value only once. This shows that with non-linear complexity there are more problems that fall within both the *P* and *AP* spectrum than within linear complexity.

This would mean that non-linear complexity seems to be more probable type of complexity that exists because linear complexity is such a strict form of complexity, which is not what is observed within many fields of science, such as quantum mechanics. With these classifications of complexity, there is even the probability that complexity can both linear and non-linear. This does not mean that complexity is both at the same time, but can take forms of being linear and non-linear. This follows along the point where with non-linear complexity *P* can at times equal

NP while in other circumstances *P* cannot equal *NP*. This is weird concept because of the way this concept follows along the lines of how non-linear complexity treats both *P* and *NP* problems. If complexity can take two different forms with *P* and *NP* problems also taking two different forms this would mean that an infinite loop of logic comes out of complexity. Since *P* can at times equal *NP* but in other cases it cannot equal *NP* within non-linear complexity, but yet if complexity can take on the form of both linear and non-linear complexity this would mean that complexity could still work with the idea of *P* sometimes equaling *NP* while at other times *P* does not equal *NP*.

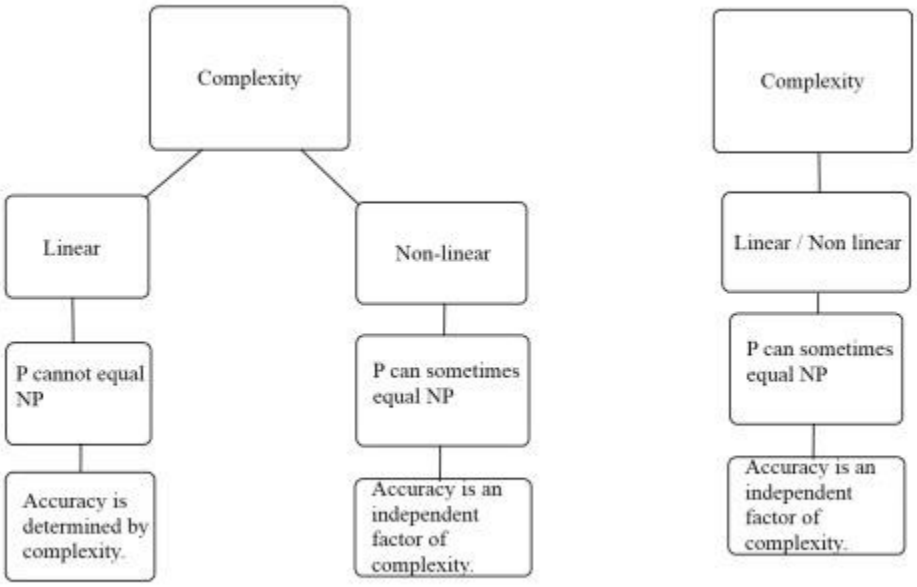


Chart displaying different concepts of complexity

There is also another way that complexity can be looked at. Instead of looking at complexity as only being able to be linear, non-linear, or both, complexity can be looked at as being all three depending on the type of problem. However, it would not matter if complexity took all three

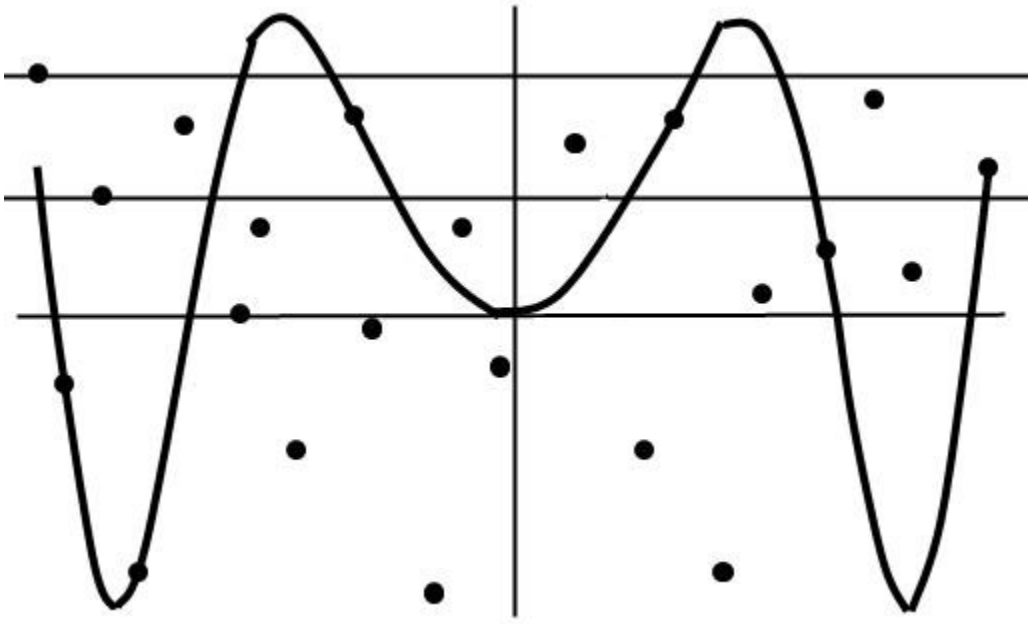
forms because the only difference would be that linear complexity relies on strict order and sequence. With strict order and sequence, patterns must emerge from randomness because randomness relies on probability to reach a random set of values or instances. Therefore, within linear complexity true and unique randomness of sets cannot occur. However, within non-linear complexity, complete and unique randomness is possible due to how *P* can equal *NP* and at other times not equal to *NP*. Proving that complexity is non-linear would also prove that there is an algorithm that could give outputs with values that are specifically unique and do not appear to form the patterns that regular known algorithms do themselves.

Randomization is one particular area of complexity that can get difficult, even with such a simple idea of randomness. The difficult part with randomness is the idea of finding an algorithm that, with sequential processes, can give outputs are totally unique to previous and future values that occur within that algorithm.

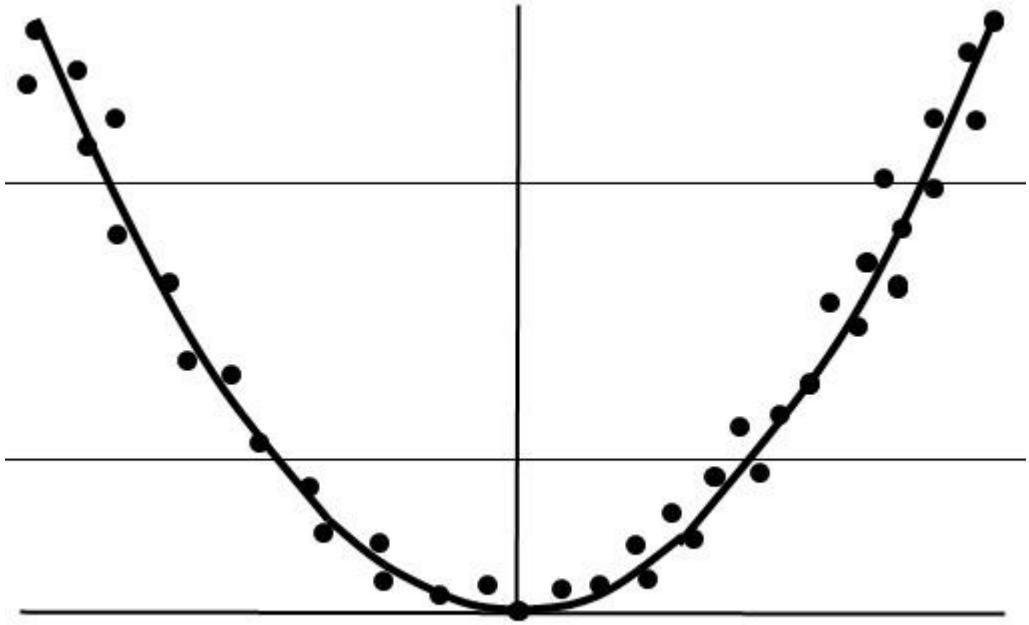
$$R = \frac{n(k+2)}{k}$$

However, the problem with these type of algorithms is their inability to give unique values as outputs. In order to have uniqueness, there would have to be a uniform non-linear probability involved, which would lead to the idea of non-linear complexity. With algorithms mostly known to following sequential proportionality, linear complexity would be most probable with these kinds of algorithms because of the nature of algorithms. With algorithms, there has to be a step process carried out in order to consider it an algorithm, which would mean that sequential values would follow as the output. The problem with sequential outputs is there won't be unique values

that proceed from the outputs that are made from the algorithm itself. Therefore, if linear complexity there is no particular way to make an algorithm that would produce unique outputs. In order to be able to form an algorithm that completes the task of outputting unique values, complexity would have to be non-linear. With non-linear complexity, the randomization algorithm would have to travel across the spectrum of complexity, forming a wave like pattern on the complexity spectrum. The big challenge, if complexity is non-linear, is being able to form an algorithm that can travel across the spectrum with only sequential development.



Randomization with non-linear complexity



Randomization with linear complexity

With non-linear complexity, the complexity value and accuracy value can travel across the spectrum without strict limitation, however with some limitations that non-linear complexity has. However, with linear complexity, there is a stricter limitation to randomization because the complexity value and accuracy follows a linear course of the spectrum. Even with non-linear complexity, however, there is still limitation to randomization because it still has the limitations, and, with limitation, there is no way to find a complete and unique algorithm that produces outputs that are uniquely randomized. Therefore, even with non-linear complexity, complete randomization is improbable using an algorithmic sequence. In any case, non-linear complexity can form a better algorithm that carries out randomization because linear complexity has a stricter form of randomization due to the limitation of its spectrum. With the non-linear form of complexity, the randomization point exists on multiple wavelengths of complexity, while, with linear complexity, an order or sequence of the randomization point is found. This means that

with randomization, there is a following sequential order that is formed from a randomization algorithm. Therefore, one must assume that complete randomization with the use of an algorithm is improbable because of the nature of complexity.

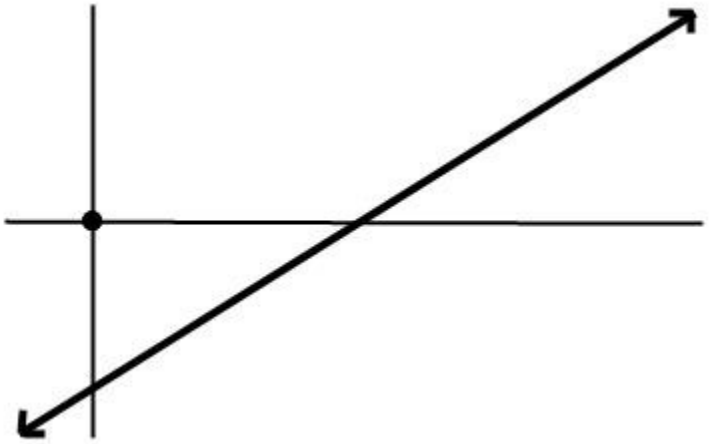
There are ways, with any form of complexity, that randomization is possible if one uses multiple algorithms to achieve unique randomized values. Using multiple algorithms to complete the goal is possible because with multiple algorithms the complexity value is dependent on *n* amount of algorithms rather than only one particular algorithm. To measure randomization, the following formula could be used to determine whether the value, compared to the algorithm, is unique or not.

$$R_a = \frac{V_1 - V_2}{ES_t}$$

The formula, known as the Relative Randomization Formula, states that the first value produced by the algorithm minus the second value produced by the algorithm divided by the product of the amount of elements within the algorithm and the amount of steps it takes to carry out the algorithm if and only if the first value is larger than the second value. To apply this to an algorithm that produces values where the second value is larger than the first value, the following formula would be used.

$$R_a = \frac{V_2 - V_1}{ES_t}$$

In order to be able to predict the randomization formula to randomization values, a graph can be shown to predict randomization values for linear complexity.



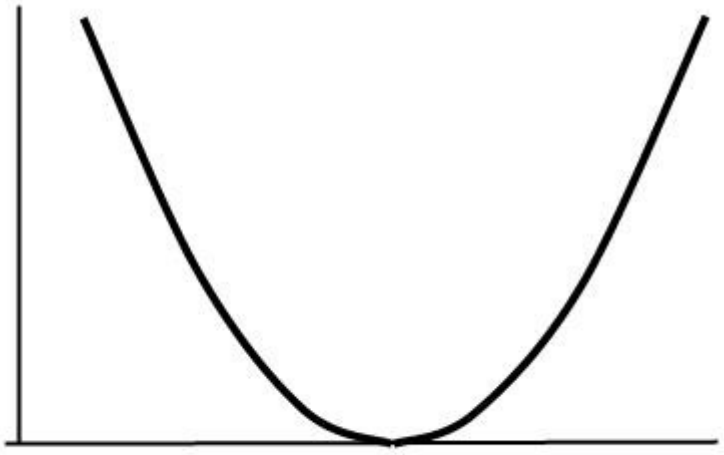
Randomization of linear complexity

In order to apply randomization to non-linear complexity, the following formula would have to be made.

$$R_{al} = \frac{(V_1 - V_2)^2}{ES_t}$$

$$R_{al} = \frac{(V_2 - V_1)^2}{ES_t}$$

To be able to predict non-linear randomization of complexity, the graph would look like the following.

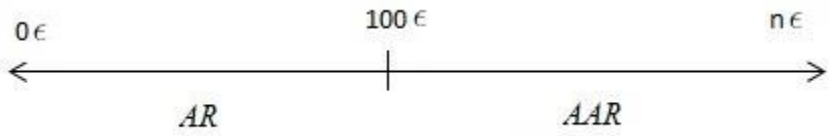


Randomization of non-linear complexity

The non-linear version of complexity brings about the idea that randomization has more accuracy within non-linear complexity than in linear complexity. This accuracy brings about the idea that randomization is more intuitive in non-linear complexity than in linear complexity. With randomization, it takes more algorithmic sequence and size in order to complete a randomization sequence with linear complexity, as seen by the graphs shown. However, with non-linear complexity, it takes less sequence and size for an algorithm to complete a randomization that is unique to each of the values produced by it. However, even with this accuracy, there is no way with either linear or non-linear complexity that a simple algorithm can be produced to have outputs of unique values compared to the other outputs produced by the algorithm. However, this does not mean that it is not possible to develop an algorithm that produces outputs similar to unique randomization values.

There are two classifications of randomization spectrums, which are *AR*, which stands for approximate randomization, and *AAR*, which stands for accurate approximate randomization.

The relevance of accuracy to randomization is due to how accuracy is measured within complexity. Accuracy deals with how much of the goal was completed with the problem in the first place. Therefore, there is a spectrum of accuracy for randomization. The following is a scale to show how to determine whether the randomization algorithm is *AR* or *AAR*.



In order to classify a randomization algorithm, using the randomization determination formula, one has to determine whether the randomization value is either less than 100 or if it is greater than or equal to 100. If a randomization value is smaller than 100, the randomization algorithm is to be classified as *AR*. However, if the randomization value of an algorithm is greater than or equal to 100 the algorithm is to be classified as *AAR*. Randomization algorithms, however, have a proportionality to them, which is shown by the randomization algorithm. This, again, is against the idea of no proportionality with non-linear complexity because, as stated before, non-linear complexity is against the idea of proportionality due to how it acts with problems being classified as either *P* or *NP*. Since the output values of a randomization algorithm are the results of an algorithm, there has to be a connection, or proportionality, with the algorithm that had produced it. Therefore, randomization algorithms have to fall within the lines of linear complexity.

The issue with having unique randomization within linear complexity is the way that randomization has been defined. Randomization is defined as the production of values of which

are unique to other values that were produced within the event of randomization, but having connection with the even or algorithm that had produced that output value in the first place. Since non-linear complexity deals with having no sender of information from which a receiver had received the information, a randomized set of values would have connection with each other because they had come from an algorithm, no matter how complex the difference is between the values. This defies the idea of non-linear complexity because of the idea of a sender and receiver. Therefore, randomization and non-linear complexity could not fit together to make an understandable unity between the two concepts. This, in some sense, would not make any particular understanding of the two because both non-linear complexity and randomization follow along with the same concepts, which is the production of the two, will result in non-linear values.

Mathematically, there is no particular way to unite the two concepts, however, philosophically, they should be able to be united as one concept or at least be able to correspond to one another. Both randomization and non-linear complexity are both made of similar, but different substance of ideology. In order for randomization to fit in with non-linear complexity, it would have to be produced from multiple algorithms, which disputes the point of randomization. The dispute with multiple algorithms used for randomization is where randomization is supposed to be done with only one specific algorithm, where the randomization comes from one specific source. However, when multiple sources are needed to complete randomization, it defeats the purpose of randomization where the set of output values is randomized relative to a specific algorithm or set of sequences and steps.

However, this does not mean that using more than one algorithm is not a way to produce randomization, but simply states that the efficiency is less, especially when calculating the complexity value of the steps required producing the output values from those specific algorithms. The point, however, is to be able to produce a single algorithm that can produce output values that are unique to themselves without any connection between the values. This means that algorithms for randomization are not completely randomized, but produce a larger difference between the output values. However, it also can be conclusive that, even with no implication to do so, all algorithms have a randomization value. However, another form of the randomization formula would have to be used because proportionality must exist with the algorithms, especially with linear complexity.

$$R_a = \frac{V_1 - V_2}{E S_t \mu}$$
$$\textit{where} \quad \mu_E = E_1 \times E_2 \times E_3 ... \times E_E$$

With this formula, an extra factor includes the values of the elements within the algorithm. This factor is known as the exponential variable value. This value takes into perspective the values that are outputted from an algorithm. To calculate the exponential variable value, the first *n* amount of values must be found, which n must be equal to *E*, which represents the amount of elements outputted from an algorithm. This formula takes into account the size of the elements within a specific algorithm, which becomes more accurate because all factors are important to calculating randomization. The other form of the formula only brings into account the amount of elements and the output values of the algorithm, but doesn't bring into account the specific values of the elements of that specific algorithm. With the existence of other algorithms that have

different implications, there is still a randomization that occurs within the algorithm. All algorithms have a specific randomization value; however the point of the algorithm may not be to output randomized values. Therefore, all algorithms fit within the spectrum of randomization, which shows the relation between complexity and randomization. This relation between complexity and randomization brings into perspective the idea that randomization may actually be complexity. This major similarity between randomization and complexity also brings up the issue of whether complexity is linear or non-linear. With this in mind, it would also be an issue to determine whether randomization is linear or non-linear.

$$R_a = S_c$$

With an existence of both linear and non-linear randomization, there would also be the question of whether non-linear randomization belongs to non-linear complexity or whether non-linear randomization belongs to linear complexity, or whether the types of randomization belong with the corresponding form of complexity. Issues with combining these types of randomization and complexity are the contradictions that occur, especially with non-linear complexity. With non-linear randomization, every output value is based on multiple algorithms rather than just one particular algorithm. Non-linear complexity relies on the idea that one simple algorithm would be able to carry out randomization, which is why these two ideas contradict. The ideologies, however, bring about new concepts of randomization, where randomization is not actually randomization, but a complex sequence that produces huge differences within the output values. This would mean that complete randomization, by definition, is not possible because by the definition of complexity, especially linear complexity, there must be a sender of information and a sender of information. With the existence of a sender, all the output values produced by the

sender are visible and even the output values show similarities between each other, like a hint that can appear that shows the existence of a sender and the sender's build up or processes.

This idea of sender and receiver information connection is repetitive within the types of complexity because the topic deals with being able to determine accuracy, complexity, and randomization. With the existence of linear and non-linear forms of these three concepts, only the linear forms of the concepts support the idea of a sender and receiver. On the other hand, non-linear forms of these three concepts either deny the existence of a sender or state that a sender is not needed in order for information to exist. However, with the lack of evidence, non-linear complexity only hangs upon observations that are made. Linear complexity has more evidence to its structure because of the structure of algorithms, which depend upon sequential development rather than randomized systems. The problem also, with the combination of complexity being linear at some instances, while being non-linear in other instances, is the fact that current single algorithms cannot output such randomized and non-proportional values. Within the observable Universe, there are many links between linear complexity and the way that the world is formed, with the only exception being quantum mechanics. Quantum mechanics is the only weak link of complexity being non-linear. This could change; however, as there is more knowledge gained by the research in the quantum mechanical fields. With this in mind, it would likely be assumed that complexity is linear, however this does not mean that non-linear complexity is not possible. If, in any case, there is proof that an algorithm with such a large randomization value does exist, this would mean that non-linear complexity is the true form of complexity. This would also mean that *P* problems can also equal *NP* problems. This would mean that a lot of something can come from a small amount of something. This defies the idea of equality, where if you have a certain amount of something the same amount is received from it.

Physics follows the same types of philosophy where the equality of substance must be kept. With non-linear complexity, this would mean that equality happens within spectrum moments, where only a spectrum of values will work within the given problem. While non-linear complexity deals with uncertainty within accuracy and randomization, linear complexity deals with certainty within randomization and accuracy. With non-linear concepts, it would be a breakdown of certainty because with non-linear complexity there are spectrums of areas where randomization and accuracy are not proportional to complexity. Proportionality is a popular observation within nature, where the values and properties of objects have similarities. For example, within physics, mass and gravity are proportional proven by the equations that scientists have formulated over the years.

Proportionality is a big concept with complexity and the other concepts discussed. However, the formulas presented for these concepts only describe those specific concepts. Proportionality has its own formula as well, but in order to measure proportionality both complexity and randomization have to be taken as factors. In order to form a formula that is specifically meant for proportionality, both the complexity and randomization formula have to be combined as one particular formula.

$$R_a = \frac{(V_1 - V_2)}{ES_t}$$

$$S_t R_a = \frac{E_t k}{n} \times \frac{(V_1 - V_2)}{ES_t}$$

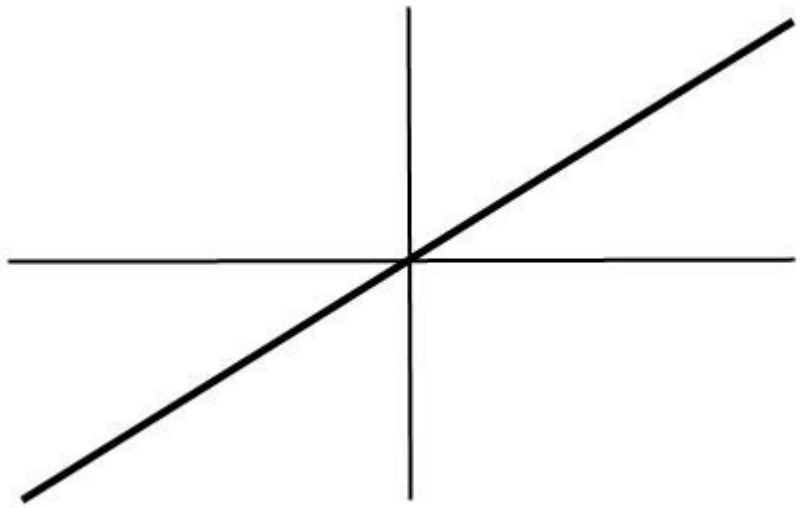
$$S_t R_a = \frac{E_t k (V_1 - V_2)}{ES_t n}$$

$$S_t R_a = \frac{E_t V_1 k - E_t V_2 k}{E S_t n}$$

$$\rho = S_t R_a$$

$$\rho = \frac{E_t V_1 k - E_t V_2 k}{E S_t n}$$

Proportionality is represented by the Greek letter rho. The reason why proportionality is the product of complexity and randomization is due to how proportionality deals with the differences between values and to their parent algorithm. With the increase of complexity and randomization, there is an increase in thee proportionality because with the difference between two values produced by a specific algorithm, there is the connection also with the amount of elements within the algorithm and the product of all the factors of the algorithm, which the includes the factor of how many levels or layers this algorithm is carried out on. If the graph for proportionality for linear complexity, it would look like the following.



With the graph, this version of the formula for proportionality has many similarities with the linear form of complexity. Of course, this formula only applies to the linear concepts. Non-linear properties of each concept would cause the formula to form an uncertainty due to how the formulas work for each concept.

$$S_c R_{al} = \frac{(E_t k)^2}{n} \times \frac{(V_1 - V_2)^2}{ES_t}$$

$$S_c R_{al} = \frac{(E_t k)^2 (V_1 - V_2)^2}{ES_t n}$$

$$S_c R_{al} = \frac{E_t^2 k^2 (V_1 - V_2)^2}{ES_t n}$$

$$S_c R_{al} = \frac{E_t^2 k^2 (V_1^2 - V_1 V_2)}{ES_t n}$$

$$S_c R_{al} = \frac{E_t^2 V_1^2 k^2 - E_t^2 V_1 V_2 k^2}{ES_t n}$$

$$\rho_l = S_c R_{al}$$

$$\rho_l = \frac{E_t^2 V_1^2 k^2 - E_t^2 V_1 V_2 k^2}{ES_t n}$$

Non-linear proportionality provides a more accurate stance on complexity, but still lacks ability to predict output values of algorithms. Non-linear concepts are also more complex and complicated due to the lack of proportionality that exists for non-linear concepts. This complexity makes non-linear complexity would cause the idea of equality to become a complicated ideology because if proportionality is the factor between equality, this would mean

that with many equations and formulas equality does not always exist between two or more factors within both *P* and *NP* problems. This idea of something not always being equal follows along with non-linear complexity because within certain spectrums of non-linear complexity, *P* can at times be equal to *NP* while at other times *P* cannot equal *NP*.

Proportionality is a very important part of mathematical fields, especially Geometry and Trigonometry, where in order to find segment values and the sides of specific shapes proportion is an important part of finding those values. It would not possible to be able to find these proportions if complexity is non-linear. This would mean that complexity would have to be linear. However, this is a misconception because for something complexity is linear while in other parts complexity is non-linear. This idea of linear and non-linear complexity is what may separate all the fields of science, especially between general physics and quantum mechanics. General physics follows the idea that many things that are in the physical realm can be calculated with certainty with just using formulas and equations, where quantum mechanics is the complete opposite. With quantum mechanics, calculations are uncertainty within predicting locations of particles with formulas and equations, however these equations and formulas predict with some accuracy because the equations and formulas bring into perspective the factors that are involved within the problem. One issue with measuring proportionality is being able to measure the complexity of that proportion. This would require a formula that takes into perspective both the complexity of that proportion and the proportionality value. To measure proportional complexity, the following formula would be used.

$$C_p = \frac{S_c}{\rho V_1 - \rho V_2}$$

In this formula, the proportional complexity is equal to the quotient of the complexity value and the difference of the product of the proportional value and the first value and the product of the proportional value and the second value. This shows the connection between the complexity value of an algorithm with its proportionality factor and output values. Another way to look at this formula is by replacing all the variables with their corresponding formulas, which would be the following.

$$C_p = \frac{S_c}{\rho V_1 - \rho V_2}$$

$$C_p = \frac{\frac{k^2 n^3}{2}}{(\frac{E_t V_1 k - E_t V_2 k}{E S_t n})V_1 - (\frac{E_t V_1 k - E_t V_2 k}{E S_t n})V_2}$$

$$C_p = \frac{\frac{k^2 n^3}{2}}{\frac{V_1(E_t V_1 k - E_t V_2 k)}{E S_t n} - \frac{V_2(E_t V_1 k - E_t V_2 k)}{E S_t n}}$$

$$C_p = \frac{\frac{k^2 n^3}{2}}{\frac{E_t V_1^2 k - E_t V_1 V_2 k}{E S_t n} - \frac{E_t V_1 V_2 k - E_t V_2^2 k}{E S_t n}}$$

$$C_p = \frac{\frac{k^2 n^3}{2}}{\frac{E_t V_1^2 k - E_t V_1 V_2 k - E_t V_1 V_2 k - E_t V_2^2 k}{E S_t n}}$$

$$C_p = \frac{\frac{k^2 n^3}{2}}{\frac{E_t V_1^2 k - E_t V_2^2 k}{E S_t n}}$$

$$C_p = \frac{k^2 n^3}{2} \times \frac{E S_t n}{E_t V_1^2 k - E_t V_2^2 k}$$

$$C_p = \frac{k^2 n^3 (ES_t n)}{2(E_t V_1^2 k - E_t V_2^2 k)}$$

$$C_p = \frac{ES_t k^2 n^4}{2E_t V_1^2 k - 2E_t V_2^2 k}$$

$$C_p = \frac{ES_t k n^4}{2E_t V_1^2 - 2E_t V_2^2}$$

This would be the final form of the formula for calculating proportional complexity if the only factors of the proportional complexity value. However, randomization is also one of the factors of complexity; therefore another formula would have to be made to bring into perspective the form of complexity. However, these formulas are only for linear complexity. The following would include the randomization factor.

$$C_p = R_a \frac{ES_t k n^4}{2E_t V_1^2 - 2E_t V_2^2}$$

$$C_p = \frac{(V_1 - V_2)}{ES_t} \times \frac{ES_t k n^4}{2E_t V_1^2 - 2E_t V_2^2}$$

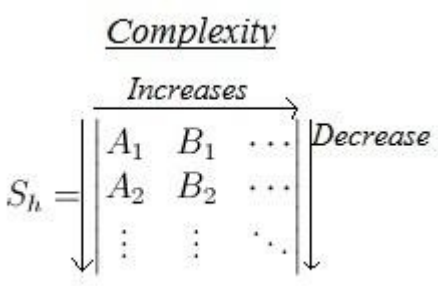
$$C_p = \frac{ES_t k n^4 (V_1 - V_2)}{ES_t (2E_t V_1^2 - 2E_t V_2^2)}$$

$$C_p = \frac{ES_t V_1 k n^4 - ES_t V_2 k n^4}{2ES_t E_t V_1^2 - 2ES_t E_t V_2^2}$$

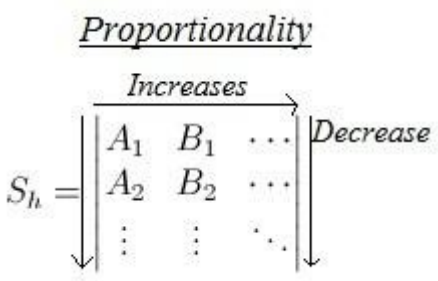
$$C_p = \frac{V_1 k n^4 - V_2 k n^4}{2E_t V_1^2 - 2E_t V_2^2}$$

With this formula, randomization is included as a factor of the proportional complexity value. This is a more accurate view of proportions because the randomization values represent the main difference between values and their uniqueness. The farther apart the two values get, the more complex the proportionality is for one specific algorithm, hence the name “proportional complexity”.

All of these formulas and equations can all be properties of the Schafftarian matrix. With the increase in *n*, the complexity value decreases. However, with the increase of *k*, complexity increases.



What is interesting about this idea is how it links to proportionality. With proportionality, the same system would apply where *k* causes proportionality to increase, while the increase of *n* decreases the proportionality of an algorithm.



This means that with the increase of levels that the problem is done one, the less proportionality the problem has. This would also mean that with the increase of levels, the less likely *P* will be equal to *NP*. In order to convert an *NP* problem to a *P* problem there must be at least enough levels or layers to carry out the problem on, but have too many layers or levels it will become, again, an *NP* problem. The amount of layers or levels that the problem is solved on must be relative to the complexity of the problem. To determine this, the problem must be assumed that it is, at first, being solved with only one layer. From this relative point, dependent on the complexity value, add layers to the problem. This would allow the problem to become a *P* problem, therefore moving the problem from an *NP* problem to a *P* problem on the spectrum. This idea causes confusion because with linear complexity, the spectrum of complexity is strict because of it being linear. This leaves complexity and proportionality in question because they both follow linear properties, but yet together they form a non-linear bond of concepts. This becomes an idea where a linear to linear conceptual bond becomes a non-linear concept. This adds a new view on complexity because now complexity types can be determined on how much of complexity or proportionality there is within the algorithm or problem. Through the proportional complexity formula, however, both the increase of *k* and *n* will bring about more of it. Proportional complexity is very dependent on the ratio between how many elements are with

the Schafftarian matrix and how many elements are within the problem combined with the product of all the factors within the algorithm.

$$\underline{\textit{Proportional complexity}}$$

$$S_h = \overset{\textit{Increases}\ \rightarrow}{\left\downarrow \begin{vmatrix} A_1 & B_1 & \cdots \\ A_2 & B_2 & \cdots \\ \vdots & \vdots & \ddots \end{vmatrix} \right\downarrow} \textit{Increase}$$

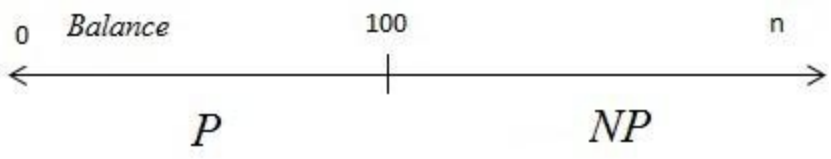
With the Schafftarian matrix, when there is an increase in *k*, there is an increase in complexity, proportionality, and proportional complexity. However, when there is an increase with *n*, there is a decrease in complexity and proportionality, but the proportional complexity increases. The issue with this is the fact that complexity and proportionality decrease with *n*, yet proportional complexity increases with *n*. These two concepts, even when they are similar, are contradictory mathematically. However, this doesn't mean that they aren't correct concepts. Non-linear complexity would work with this concept because according to non-linear complexity the equations and formulas do not have to be proportional. This would have to mean that proportional complexity is only a valid concept within non-linear complexity, not linear complexity. Another problem with proportional complexity being in the spectrum of linear complexity is the way the amounts of elements within the algorithm interact with complexity. With the proportional complexity formula, the amount of elements and the element's values would decrease the proportional complexity value; therefore the increase in elements decreases the proportional complexity of the algorithm. This defies linear complexity because with the increase of elements there should be an increase in complexity. Since, however, it works the

opposite; the concept must fit within non-linear complexity. This would mean that if non-linear complexity is proven to be the right form of complexity, this would prove even more that within the spectrum of non-linear complexity *P* can be equal to *NP*. Proving that complexity is non-linear would prove that *P* can, in fact, equal *NP*. However, if it were to be proven that complexity, in any way, is linear this would automatically prove that *P* cannot equal *NP* in any circumstance. This would also prove that proportion, with non-linear complexity, is non-linear as well because of how it has an effect on both complexity and proportional complexity. Non-linear complexity does not follow balance as linear complexity does. Balance and proportional complexity have similarities, but also have differences. Balance is a concept of linear complexity while proportional complexity is a concept belonging to non-linear complexity.

$$B_c = \frac{E_{sct2}U_{sc1}}{E_{sct1}U_{sc2}}$$

$$\textit{Where} \quad U_{sc1} = V_1 + V_2 + V_3 ... + V_n$$
$$\textit{Where} \quad U_{sc2} = V_1 + V_2 + V_3 ... + V_n$$

This is the approximate formula for balance within linear complexity, where both variables of *E* represent the amount of elements within two algorithms. The point of this balance equation is to determine how well two algorithms are balanced with each other, assuming that complexity is linear. This balance equation brings into perspective the idea of comparing two algorithms to see if they fit within the lines of either *P* or *NP*. The scale for balance is very similar to the scale for complexity, except the balance scale is more linear than the complexity scale in a sense that balance only deals with factors, such as the length of the algorithm and the factors involved.



Balance follows linear complexity because balance will always say that two pieces of a problem are always equal no matter what. In this concept, balance is very sensitive within linear complexity because if two things are equal they must be equal. However, this balance is not what is thought of as balance. In definition, balance is where two things are equal no matter what. This type of balance falls within spectrum and scale. In a certain area of the scale, things are within balance and falls within the linear complexity. While proportional complexity has a broader spectrum of equality, balance in definition with linear complexity has a more strict view of equality. However, these formulas only give approximate values to algorithms with linear and non-linear complexity. In order to give an exact value with complexity, there must be a variable involved with linear and non-linear complexity. However, oddly enough, this variable varies between algorithms. For example, randomization algorithms will have either similar or the exact variable, while algorithms dealing with other jobs will have a different variable involved.

$$\omega_c = \sum_{n=1}^{\infty} \frac{V_n V_{n+1}}{U_n U_{n+1}} = \frac{V_1 V_2}{U_1 U_2} - \frac{V_2 V_3}{U_2 U_3} + \frac{V_3 V_4}{U_3 U_4} \cdots - \frac{V_n V_{n+1}}{U_n U_{n+1}}$$

This variable is known as the complexity constant of algorithmic sequences. This variable, represented by omega with a subscript of *c*, is a variable that shows classification for certain types of algorithms that exist. This variable can be used to classify algorithms into specific categories, since all algorithms, no matter their intention, have a randomization value. This

randomization value classifies all algorithms as being proportionality between the approximations of exact values that exist in reality. Balance and equality is within the range of spectrum rather than exact, which also applies to proportional complexity. Proportional complexity not only focuses on the idea of approximate equality, but also focuses on the complexity of this approximation. This mainly asks the question of how in-balanced is both sides of the equation, meaning the complexity of this in-balance. This insight into equality and proportionality will allow the observation of equations from the perspective that uncertainty is at hand, even when the intention is for equality within mathematics.

These formulas, equations, and concepts apply to mainly algorithms and Schafftarian matrices that follow the dimensions of only *k* and *n*. However, this whole ideology of complexity becomes more complicated with the existence of a Schafftarian matrix on the third dimension. With this type of Schafftarian matrix, there are now three variables involved. These variables, known as the dimensions of the Schafftarian matrix, are known as *n*, *k*, and *d*. The *d* variable, in this case, represents the depth of the Schafftarian matrix within the third dimension.

$$S_{\Delta} = \overset{d}{\underset{k}{\overset{n}{\begin{matrix} [E_n][E_n] \\ [E_n][E_n] \\ [E_n][E_n] \\ [E_n][E_n] \end{matrix}}}}$$

With correlation of the virtual world with the reality, the third dimension of the Schafftarian matrix becomes more complex with the newly added *d* variable of depth. Now, the formulas for the second dimensional Schafftarian matrix have to be modified to fit the environment of the third dimensional Schafftarian matrix. The formulas are, fortunately, not much different than the formulas for the second dimension version of the Schafftarian matrix. The reason for the

complexity is because of variation of the Schafftarian matrix within the third dimension. In geometry, there are many types of forms a third dimensional volume can take, including a sphere, cube, prism, pyramid, and cylinder. The complexity involved here is the fact that the amount of layers or levels can, in fact, vary when each section of a third dimensional Schafftarian matrix is formed.

$$S_{\Delta} = \begin{matrix} d \\ n \\ k \end{matrix} \begin{bmatrix} [E_n][E_n] \\ [E_n][E_n] \\ [E_n][E_n] \end{bmatrix}$$

Rules of interpretations for this type of Schafftarian matrix can vary because of the complexity of the algorithm itself within this type of dimension. The third dimension has more uncertainty than the second dimension because at least, for the second dimension, there are stricter rules that apply. However, the variable of d is not to be added directly. If a third dimension is added to the Schafftarian matrix, not only does the complexity increase, but the amount of calculations involved will double depending on how the calculations are done. However, this does not mean that the amount of steps taken to calculate a problem has increased. This idea would assume that steps are all being done on one layer or level, which this assumption is incorrect. With the existence of more layers and levels, multiple calculations can be done at the same exact time. With more levels or layers, efficiency increases as well within the bounds of complexity, as proven by the formulas mentioned before. Even though the second dimensional Schafftarian matrix may be more efficient than the first dimensional Schafftarian matrix, the third dimensional Schafftarian matrix is more efficient than the second dimensional Schafftarian

matrix. There is a balance between complexity and efficiently with these systems, which leads to the issue of whether efficiency or complexity is the goal of the system.

Less complexity
More efficiency

$$S_{\Lambda} < S_{h} < S_{\Delta}$$

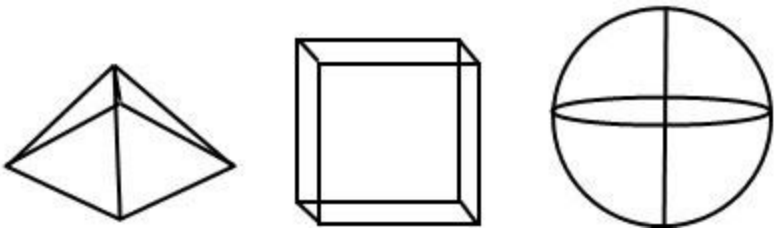
More complexity
Less efficiency

In this case, S_{Λ} resembles a one dimensional Schafftarian matrix, S_{h} resembles a two dimensional Schafftarian matrix, and S_{Δ} resembles a three dimensional Schafftarian matrix. Even though the formulas for each dimension must change, there is not much a difference between the formulas. For the formulas, the following will represent the Schafftarian matrix for each dimension.

$$S_{c\Lambda} = \frac{E_t k}{2}$$

$$C_{p\Lambda} = \frac{V_1 k - V_2 k}{2E_t V_1^2 - 2E_t V_2^2}$$

These are only sample formulas for a one dimensional Schafftarian matrix. For the formulas of the third dimensional Schafftarian matrix, all that would be need to be done is to multiply the amount of "faces" on the Schafftarian matrix by the formulas and solve the formulas from there. Here are a set of geometrical volumes that exist.



Some geometrical shapes

Though the Schafftarian matrix does not resemble the shapes in Geometry, they have a strong connection with the shapes of these geometrical volumes. For example, with a simple Schafftarian matrix existing on the third dimension, it is only a repeat of a two dimensional figure, except with multiples of the faces of the Schafftarian matrix. This relates to the geometrical shape of a cube. There are ways to form Schafftarian matrices that resemble the shapes of a pyramid or sphere, but the issue with these Schafftarian matrices is their complexity. In order to symbolize these types of Schafftarian matrices mathematically, the variable symbol would be $S_{\Delta n(a_1,a_2,a_3,...a_n)}$, where *n* represents the amount of faces the Schafftarian matrix consists of. A similar instance can be done with a two dimensional Schafftarian matrix. This would be done where $S_{h(a_1,a_2,a_3,...a_n)}$ and *a* represents the amount of elements in each layer or level and its subscript *n* represents the layer or level number within the Schafftarian matrix. This would mean that the default identity for a Schafftarian matrix is $S_{Dn(a_1,a_2,a_3,...a_n)}$, where the variable *D* represents the amount of dimensions that the Schafftarian matrix is working on. This, however, is not the complete identity of a Schafftarian matrix to be presented mathematically. There are other properties that could be described within the identity, such as complexity, proportionality, and proportional complexity. There is also the type of problem and whether it

would be a *P* or *NP* problem. This would mean that the whole identity would be $S_{DTC_p P_t P_c n(a_1,a_2,a_3,\ldots a_n)}$. This identity, however, does not have to be written out fully because it truly depends on the intention of addressing the identity of the problem with the Schafftarian matrix. The point of this identity is to shorten what is needed to be written without having to write out the whole problem within a matrix. This is due to how the complexity, proportionality, and proportional complexity representing both the amount of elements within the Schafftarian matrix and the amount of layers or levels that the problem was worked on.

It is hard to imagine how binary language can fit into the Schafftarian matrix, since it is complicated to see how binary language fits only on a layer within a matrix even when it is not intended to be like this. However, binary is, in fact, a single layered language because it works on one layer to carry out a task. Though binary seems like an efficient language, it is not efficient enough. The problem with binary is each element of the language is only used once to carry out a process. There is an even more efficient way to process information, like how DNA is processed within the human body. In the language of DNA, each nucleobase is not just used once, but is, in fact, used at least six times to create structure, according to the Schafftarian hypothesis. This is not only efficient, but shows how nature itself accomplishes processes and tasks. DNA can be viewed as a Turing's machine, but consisting of six tapes that have not only two, but four factors. The main difference between binary and the language of DNA is not only the efficiency, but because of the fragility of both languages. With binary, even with one change not much is affected by the change. However, with genetic coding, even one change can change the structure of the human body. One nucleobase change can change at least six structural sequences within the DNA sequence, which means at least six characteristics would be changed.

There is one factor that becomes a very specific and important factor of the Schafftarian matrix is the amount of types of factors that are used within the Schafftarian matrix. This normally does not apply to regular algorithmic analysis, but rather works with languages and computation language mainly seen within mathematics. With binary, there are only two factors involved. However, with genetic language, there are four factors involved, which are *A*, *T*, *G*, and *C*. Many of the formulas for the Schafftarian matrix deal with how many elements or layers are dealt with when solving a problem or finding the properties of both linear and non-linear complexity, but the formulas do not include the factors that are involved within the Schafftarian matrix. The reason for this is the formulas are meant for analysis of algorithms rather than languages. This is why there is a separate section of formulas and equations that calculate the complexity of the Schafftarian matrix. One of these formulas is a formula that calculates the complexity of a language with the use of a Schafftarian matrix.

$$S_{ch} = \frac{E_t nk}{F_t}$$

The complexity formula for computation languages is very similar to the complexity formula for the Schafftarian matrix that analyzes algorithms, but instead of dividing by *n* the division is by the amount of factors involved with the language. Since this is not an algorithmic analysis of the Schafftarian matrix, the variable for exponential time is not to be measured the same. In this case, exponential time is to be measured by dividing the product of *n* and *k* by the average time it takes to interpret the sequence or message.

$$E_t = \frac{nk}{T_a}$$

If these two formulas were to be combined, this would be the result.

$$S_{ch} = \frac{E_t nk}{F_t} \, E_t = \frac{nk}{T_a}$$

$$S_{ch} = \frac{\frac{nk}{T_a}(nk)}{F_t}$$

$$S_{ch} = \frac{\frac{n^2k^2}{T_a}}{F_t}$$

$$S_{ch} = \frac{n^2k^2}{F_t T_a}$$

This shows that by dividing the product of *n* squared and *k* squared by the product of the amount of factors involved and the approximate time it takes to interpret the sequence, the complexity value of a language, within the bounds of a sequence, can be found. The complexity value is the only formula that can be similar to the algorithm analysis version of the Schafftarian matrix because proportionality and proportional complexity is not relevant to computation language. More specifically, with the analysis of computation language, there is no need to compare the proportionality because there is no analysis of equality with computation language, but only with the algorithmic analysis. Genetic language is one example that could be observed this way with the Schafftarian matrix. These formulas, however, have one main goal, which is to observe and analyze complexity of algorithms and computation languages. In order to use the Schafftarian

matrix, there are specific formulas that are needed to find the properties needed, such as locations of elements in order to achieve a language.

$$P_{\frac{nk-nu}{2}} = E_x$$

In this formula, *P* is equal to the element that is at the position of its subscript. In this formula, the element variable is a representation of the element at the position of the subscript of *P*. This gives a default definition for the Schafftarian matrix. However, if one is to define the properties of the Schafftarian matrix, the following formula would be used.

$$P_{\frac{nk-nu}{S_y}} = E_x$$

In this formula, the variable *S* with subscript *y* represents how the language will be computed with the Schafftarian matrix. For example, if a Schafftarian matrix has the variable *n* equaling six and *k* equaling five, with *u* equaling two the following would be the result.

$$P_{\frac{30-12}{2}} = E_x$$

$$P_{\frac{18}{2}} = E_x$$

$$P_9 = E_x$$

In this case, there is a way to determine the value of an element at a certain position using the values of *k* and n along with the variable *u* and *S* subscript *y*. However, in order to put this to

good use, there needs to be a way to address this value within an equation or formula. In order to give this element an identity, the following would be the format.

$$P_{\frac{nk-nu}{S_y}} + P_{\frac{nk-nu}{S_y}} = S_m$$

This equation is an example of how the format would look like when carrying out a mathematical operation with two elements within a Schafftarian matrix. However, not only mathematical operations can be carried out using this type of format. With genetic coding, the same is applied to the Schafftarian hypothesis where the matrix is split into two sections, dividing the variable *n* by two, to carry out the Schafftarian matrix algorithm on DNA sequences. This same type of algorithm can be used to create more efficiency with calculations using binary combined with the Schafftarian matrix.

The main point of the Schafftarian matrix algorithm is efficiency. This type of efficiency can immediately be evident within genetic code, where there are only four base pairs involved and all the base pairs, when in specific locations on the DNA strand, have more than one function. Even with this amount of efficiency, there are also very negative consequences of this. Even with on single base pair change a huge structural change can happen, which is especially proven with how the human body reacts with even the slightest change in genetic information. This same concept can apply to other machines that follow within the same area of complexity with the use of multiple layers or levels. Since, with multiple levels or layers, there are these sensitive areas that exist; even with such perfection there is also the greatest probability of negative consequence. Even with this complexity of “perfection”, there is the area in which the greatest

consequences can thrive, fortunately and unfortunately. In this area of complexity, error within a multiple layered Schafftarian matrix is like a bomb landing on ground and exploding. Though, the point of target was based on one specific point, the area around is greatly affected. This can also be known as the butterfly effect, but this is not the best analogy to apply to this. This brings big issues with using the Schafftarian matrix for improving artificial intelligence. With such great sensitivity, one simple error can be found as one of the most destructive forces within the computation within the language analysis of the Schafftarian matrix. This inability to adapt to an error can bring things into question of whether efficiency is justifiable to say that the risk should be taken to use the Schafftarian matrix for language computation, especially in artificial intelligence. The proof to the reason why the risk should be taken is the human body itself. The human body, which is structured with DNA, is a very complex system and maybe even one of the most complex systems that humans themselves know of. This complexity can be lead back to the existence of genetic information, or DNA. Humans are complex machines consisting of a computational machine, known as the brain. Even the human brain is considered still a mystery to the field of science. With such close perfection, there is the greatest consequence of failure within the structure of DNA. This follows along with the idea of being the biggest and having the hardest fall. With the increase of complexity, the smallest error can become the biggest mistake within a system. The main understanding of a complex machine is understood to mean that the system can withstand such mistakes when encountering the mistake in the first place. However, this is an incorrect assumption. The more complex something is, the worse a small error can affect an entire system, while with more simplicity there is less effect of a small error upon the system.

With such power of computation follows a higher price of error. The complexity formula itself could be used to calculate the impact of error upon the system. This has a very close connection within philosophy, where the increase of power comes along the higher impact of error upon a system. The more complex a system becomes, the more viable it is for error. This is one of the reasons why nature tends to carry out its processes with the most efficient process. The efficiency of process is the saving point of a system. Without this efficiency, nature is always probable to error. With this ideal, even proven by the values of physics and its fine tuning, if these values were not as they were the Universe would not exist. Existence relies on this simplicity where efficiency, not complexity, is the goal of computation with the Schafftarian matrix. Even with the increase of how many levels a problem is solved with, there must be a balance. Nature achieves or tries to achieve this balance. There is proof of this existence of fragility of the Schafftarian matrix. Human effects on a system are very visible to statistics taken to show our impact upon nature, politics, and economy. The more complex a system becomes the more probable an error is to occur. This is very visible in economics and politics. This concept is especially seen within nature. This may be confusing due to the definition of error within this ideology. In this ideology, error is the difference of norm existence within a spectrum amount of time, according to a relative standard. With the complexity of economics, there is more probability of error. With the complexity of politics, the more probability of error there is within the system. What should be the goal of the entirety of politics and economy should be simplicity. With simplicity, there is little error or a less probable area of error.

To back up from this idea, there has to be a thorough analysis of error. It can be argued that error is relative to perspective. However, this definition of error does not follow the regular definition of error. This definition of error is relative to the norm, where a change is existence due to either

natural or artificial effects. In statistics, there is a situation where complexity has become the greatest enemy of both economics and politics. Complexity and simplicity are at a “tug-of-war” with each other in systems, where the increase of complexity brings upon more probability of error while increase of simplicity brings upon less probability of error. There is this false notion that exists that the more complex something becomes the more accurate it becomes in the very end. This notion is very dangerous and can cause the existence of error to enrage and spread like a disease. This is why nature follows the route of efficiency. When humans do become involved within nature and its processes, nature has to go the more complex route. This does not make humanity the enemy of nature, but it makes complexity the enemy of nature. The Universe is a working system that works to make the most efficient move that exists. It is like a live being that works to make the most efficient decision. When another object or species enters a system, the system must adapt or even evolve to this change in the system. This causes the system to have to become more complex. It is like being in a train station, where there are thousands of people that walk across everywhere going in straight lines and finding the most efficient direction to the destination. However, if some obstruction all of the sudden comes into existence in the middle of the building, the people have to change their path in order to adapt to the system, which makes the system more complexity and even slow. If more obstruction were made, the people would have to figure out a more efficient path in order to follow the system, however at a certain amount of obstructions the system will eventually stop and self-destruct. The Schafftarian matrix follows the same philosophy, especially with genetic information. It can even be evident that any system has to follow the same philosophy in order to live on as a system. A system must meet standards that even nature has to follow. Systems must follow properties in order to live on to exist. For a system to exist, it must be able to adapt within the bounds of complexity. In order for

a system to exist, it must be efficient and less than complexity and a system cannot accept massive change without being able to adapt to the change before the change to occur. A system is like a living being, where it must keep to its environment. Algorithms, themselves, follow the same type of properties. With even a small change to an algorithm, a huge effect is put upon the solution to the algorithm. All systems exist with the same properties no matter the intention of the system.

Complexity of a system, however, does not rely on what dimension the Schafftarian matrix is on with the system. For example, the natural algorithm, as it could be called, may in fact work on either the second or third dimension. This depends on the efficiency of the algorithm being on which dimension, but the point being is how a three dimensional Schafftarian matrix is carried out. When on multiple dimensions, it is assumed that all layers are being worked on at the same time, meaning the no time is lost by adding a new layer to the problem to get a solution. However, bringing the algorithm to the third dimension would bring issues with efficiency because with the rule of the third dimensional Schafftarian matrix there must also be another rule involved in the algorithm. If nature would be to go the most efficient route, nature would most likely follow the two dimensional Schafftarian matrix. Evidence of this, of course, is the human genetic make-up. With research on DNA with the Schaffter hypothesis, it all works on a two dimensional Schafftarian matrix that has variable n equaling six of course. This shows that nature follows a fine-tuned system of efficiency with genetics, and even with the other processes of nature. With nature, there is the question of whether both humans and other mammals follow the same algorithm closely as human genetic code does. Another area of evidence that proves this connection between nature and efficiency is the existence of evolution. Evolution is one of nature's ways of dealing with complexity, where when something does not fit into it this item or

factor is removed and disposed of. This shows that nature disposes of obstructions to its improvements and adaptations. With natural selection, there is an algorithmic sequence that detects error within the properties of nature and completely removed it. However, this assumes that evolution is, in fact, a system that exists as a force without the species that exist in nature. This force exists within the species and organisms that exist and within the DNA, which follows the idea of simplicity. This process of evolution is not a force, but is a result of the approach of simplicity by genetic information of organisms that exist. Since nature follows efficiency, there would be less efficiency if, in fact, evolution is a separate force. Having evolution as a result of genetic information built into organisms would be more simplistic and more efficient that would require less time and forces to carry out nature's processes. Having evolution as a force itself would bring upon a more complex system, which would mean that efficiency would not be the goal of nature. This would bring upon the destruction of a system of nature and with it the destruction of the Universe, not only by the non-efficiency of evolution, but by the non-efficiency of the Universe the Universe would not exist at all. This fine tuning of the Universe is caused by this efficiency, even with all the matter that exists. The only complexity of the Universe is caused by its simplicity that involves finding the most efficient way to manage the matter that exists in the Universe. The reason why the efficiency with simplicity is more complex is because the human mind challenges itself to understand complexity. When understanding complexity comes to meet understanding simplicity, the human mind cannot understand the difference between simplicity and complexity. The human mind, in the modern day world, is trained to understand mainly complexity because it is to be thought that the Universe is complex in its build up, however this becomes a false assumption that blocks the human mind from discovering simplistic ideologies.

Quantum mechanics, itself, is a complex system that, however, goes along with the idea of complexity and uncertainty. It follows along the paths of non-linear complexity, consisting of uncertainty of reality. This goes against the idea of efficiency for nature and for the Universe. Non-linear complexity does not follow the array of efficiency, but follows the exact opposite. Since there are multiple ways for something to be equal, there would have to be a way to determine those equalities. This is not efficient; therefore linear complexity must be the true form of complexity, at least the only natural form of complexity. This does not mean non-linear complexity does not exist. Artificially, non-linear complexity could be used to compare *P* problems and *NP* problems, relying on the idea that they can at times be equal, but other times they cannot be equal. However, the problem with non-linear complexity, as wanting to be achieved, is it does not follow efficiency. Efficiency is only possible with linear complexity, which is why it is called linear. Non-linear complexity is not efficient because of its standards of equality, hence it being non-linear. Again, efficiency is the goal of the Schafftarian matrix because with the measurement of complexity, there are many factors involved in finding the most efficient solution to a problem. The reason for nature's complicated appeal is due to its simplicity. Simplicity, in many forms, is sometimes perceived incorrectly of being complex. This is one of the main reasons why DNA, with reading it directly as a one-layered Schafftarian matrix, is so complex in its build up. What happens is the way the user reads it does not follow along with the idea of efficiency. Efficiency means to use everything without only just using it once and being able to recycle elements of the problem, just like how in genetic information, most of the nucleobases are used at least six times in a sequence. This kind of usage is very common in nature, where materials are efficiently used and not disposed of till the last particle is used up. For example, if an animal dies in nature it is not disposed of as just dead and entering

the ground as waste as it gets eaten alive by the many small organisms that exist in nature. When the organisms do eat up the dead animal, other animals may come to eat it and take the resources it needs to survive. Then, the little organisms will start eating away at the body of the dead animal, eating away at what remains. After these little organisms eat what they can of the body, other processes may occur that nature carries out to use efficiently the substances that come from the dead body, which then after the leftover particles are absorbed into particles as nutrients into the ground for plants to grow. This kind of efficiency is necessary in order for the food chain to carry on its natural processes. Even one small change to the food chain can cause a huge effect upon the existence of nature. This kind of effect is very common within the natural algorithm, even in the genetic makeup of organisms. Evolution is not just a "force" that exists in nature that governs which animals die and which ones live on as a species. It is a built in process of genetics that allows nature itself, within the organisms, to decide what lives and what dies. Nature is also a living organism, even when it does not seem like it. Evolution, in this case, is also about the process of conservation of matter. Instead of letting matter going to waste, nature uses every last particle of it. That is why it is shut a perfect system; however humans must also be involved in the system in order to make sure drastic changes, which are enemies of a system, do not occur. Humans have a big effect upon the algorithm of nature and can cause either really positive or negative changes to nature. For example, when too many trees are cut, the system has to continue to change drastically. Without the system being able to "adapt" to this change, there is no way for the system to continue. The system goes to a halt and soon begins to break down until a new condition can be met for this specific system. Just like an animal's system, where since the animal's body cannot adapt to either violent or dies with non-adaptable occurrences, it also comes to a halt and dies, which then the system of nature must adapt to this change by disposing

of the material produced. Even if the system of nature can adapt to such changes, there is a bound for systems. If too many drastic changes occur, the system begins to fall apart. The system of nature consist of a perfect balance of complexity and simplicity, which shows that nature tries to achieve the most efficient outcome if a drastic change occurs. However, even with this ability, errors can cause the system to change drastically.

The Schafftarian matrix is also an example of a system. With one simple change of the matrix, the algorithm itself also changes to fit the change that occurred. In order to be able to fit this change, the algorithm must change slightly in order to survive the change, which means that the algorithm relates to the original form, but adapts to the change that occurred. However, this type of process does not always happen. As stated before, with such perfection there is also a negative consequence that can occur even with one error. With such complexity in a system, the more likely errors are to occur because of the structure of a system. That is why, with the Schafftarian matrix, it is the goal to decrease the complexity value of an algorithm or language. With less efficiency, it is most likely that the system is not simplistic, however with more efficiency it is more likely that the system is simplistic, even if it does not seem like it is. With simplicity, predictability becomes more of a feasible task than when predicting complexity. This has to do with the proportionality value and the proportional complexity value, where the increase in these values means the more complicated it is to predict events within algorithms, especially algorithms found in nature. Quantum mechanics is an example of a complex algorithm because it fits within the range of complexity, especially in non-linear complexity. Quantum mechanics follows the non-linear way of thinking, which is why it follows a complex type algorithm. With quantum mechanics, the location of particles is always changing due to its sensitivity of Universal changes. Quantum mechanics is one of the main examples of what the consequences

are when the complexity of it is so high on the spectrum. That is why predictability with quantum mechanics is complicated in the nature of uncertainty. However, this assumes that quantum mechanics follows an uncertainty principle. Quantum mechanics could actually be simplistic with simple rules, leading to a complex design. The issue here addresses whether simplicity is the result of complexity and that complexity is the result of simplicity. This would mean that *P* problems are actually a result of *NP* problems while *NP* problems that are a result of a *P* problem.

This would also have to mean that *P* problems are almost the same as *NP* problems in the first place. The result of *P* problems being *NP* problems would have to mean that the results of *P* are equal to *NP* while the results of *NP* are equal to *P*. This would have to mean that both linear and non-linear complexity exist, but within bounds. While linear complexity exists on the higher levels, non-linear complexity exists at the core of what follows linear complexity. This would mean that, since complexity by this ideal is both linear and non-linear, *P* can equal *NP*, or rather the results of these types of problems is the exact opposite type that the problem imposed. In other words, having complete efficiency is impossible; however the goal of efficiency is not impossible. Nature's system relies on the ability of trying to achieve a purely efficient system that leads to nature's seemingly complex system. Evolution is this attempt to perfect the system of nature's algorithm, where the goal is to achieve perfect efficiency. If the point of a system does not include finding the most efficient algorithm to accomplish a task, the system will simply fall apart and be disposed from existence. The Universe is a growing and living system that attempts to achieve perfection. Humanity, in fact, is an example of this attempt. Humans try to find the most efficient algorithm to achieve daily events. With a growing human race, the algorithmic sequence of humanity is increasing in complexity. This complexity can be resembled

by road systems. If a road system attempts linear course, the less traffic there is. However, if road systems begin to enter curvature, or non-linear complexity, the whole system slows down and soon becomes so weak that the system cannot work anymore until linear complexity is achieved. This would mean limiting the route or path of an algorithm to lessen the amount of processes or steps needed to accomplish a task that occurs in a system. For example, if there is a problem to be solved, the first step is to lower the size of the problem, which then makes the problem within the bounds of simplicity. Simplifying a problem is the major step of finding the solution.

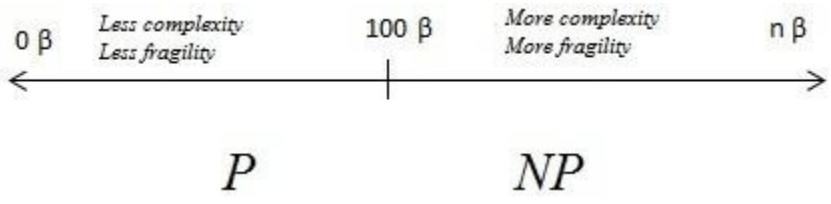
With a more complex problem, it is more likely an error will occur. With this, the range of simplicity and complexity is also a scale of fragility with a problem. This is especially visible within the uncertainty principle of quantum mechanics. Quantum mechanics has a fragility that makes it hard to predict events with particles using the equations that are a part of the uncertainty principle. This leads to the idea of a spectrum of certainty and uncertainty, which then can be calculated mathematically if all the factors are accounted for. Both certainty and uncertainty can fall within their own areas of complexity. Certainty would fall within the range of linear complexity while uncertainty falls within the idea non-linear complexity. Certainty, as by definition, states that linear complexity knows and can predict exact values within a system. On the other hand, uncertainty states that within non-linear complexity predictions with equations, formulas, and algorithms are not always correct and have to adjust to the specific area of occurrence within a system. If this uncertainty also applies to the natural algorithm, many things with the natural process would be uncertain, therefore the natural algorithm would not work.

Fragility of an algorithm can, in fact, be calculated by finding the values of an algorithm and finding the ratio and proportionality between then.

$$F_t = \frac{V_n}{V_{n+1}} \times \frac{S_c}{E}$$

$$F_t = \frac{V_n S_c}{V_{n+1} E}$$

This formula brings into perspective the values that come as outputs form an algorithm, the complexity value of the algorithm, and the amount of elements involved in the algorithm. The point of it is to measure the fragility or sensitivity of an effect upon the algorithm and the impact a change would have upon the algorithm the calculation is done on. The Schafftarian matrix can also use this as a formula to calculate the impact of a change upon a specific language. Fragility has a big part involved in both linear and non-linear complexity, where the more complex an algorithm or language becomes the more fragile the algorithm or language is to change. For example, with a language, the more complex it is, even with one simple change, the whole meaning of a sequence used by the language would change. One a scale, the fragility is proportional to complexity, where the more complexity there is the more fragility there is. On the other hand, less complexity means less fragility.



With *P* problems, change does not cause a huge impact upon the problem and what the solution will be. However, *NP* problems are affected massively by change and are more fragile than P problems. This would mean that *P* problems are different than *NP* problems. This would have to mean that P and NP problems cannot be equal, which is a contradictory concept between linear and non-linear complexity.

If, with non-linear complexity, *P* and *NP* can sometimes be equal, this would have to mean that their properties are either similar or identical. This becomes an issue because quantum mechanics proves the existence of non-linear complexity, but the concepts of non-linear complexity would have to be considered contradictory because of the idea of similar properties between *P* problems and *NP* problems. However, since they do not have the same properties they cannot be equal. The problem with this notion is the assumption that both cannot change properties and treats them as separate instances when they are only classifications of problems. Classifications, in this case, can have different properties when put into certain instances or situations. With this in mind, *P*, as a classification, can equal *NP* if these are considered classifications of problems. Therefore, these two types of concepts are no longer contradictory. This would mean that the classification of *P* can equal the classification of *NP*, as proposed by non-linear complexity. There is another problem, however, with the classification of P equaling the classification of *NP*.

If *P* problems can only equal *NP* some of the time, that would mean that there would have to be a certain definition of what these classifications would be because non-linear complexity deals with this whole realm of uncertain equality. If uncertain equality is the truth of mathematics, then nothing can be proven because there is an uncertainty involved that limits the idea of proofs. This uncertainty also leads to the idea that some may take the assumption that randomness is how the natural algorithm goes by, this is incorrect. Randomness, in this sense, has an order, but

falls within the range of proportional complexity. This randomization value represents how the difference between values produced by an algorithm does have, in fact, proportionality to it. This leads to the existence of a proportional complexity value, where there is proportionality between the values, but the proportionality is so complex that it cannot be detected easily. As stated before, out of simplicity there is an output of complexity while out of complexity there is the output or result of simplicity.

This can lead to the misconception of simplicity and complexity, especially with *P* and *NP* classification. Simplicity, in itself, is not always a result of complexity. The same thing applies to complexity where it is not always the result of simplicity. Simplicity and complexity are both of different concepts and can be derived on their own.

Both fragility and efficiency are main issues with linear and non-linear complexity. Achieving an algorithm or language that can adapt to intense changes while becoming complex enough to complete a task that resolves around complexity would be difficult because with the equations this is almost improbable to do and even if it can be done the problem would most likely fall within the spectrum of non-linear complexity, which already defies the ideas of complexity. Balance would have to be formed in order for this to work. Even in nature, there is not perfect efficiency because of the many changes that occur, whether human or not human. All organisms have an involvement in forming this natural algorithm that exists; therefore all organisms can have an effect upon the natural algorithm. For example, with the growth and extinction of species, the natural algorithm has to adapt to these changes with the process of evolution. In this case, if an organism begins to inhabit another organism's habitat the natural algorithm has to adapt to this change in structure within this certain area of environment. This brings a connection

between the natural algorithm and the genetic algorithm. The genetic algorithm has very many similarities with the natural algorithm.

These similarities deal with the impact of change upon the genetic information. As stated before, following the Schafftarian hypothesis, each nucleobase is used at least six times within sequences to form the structure of an organism. This type of algorithm can be applied in artificial means as well, improving the speed and efficiency. With computer development, there is always the existence of limits because in order to improve technology with the computer language called binary, it requires processors to increase in speed with a one-layered Schafftarian matrix. This causes an issue with the improvement of technology also because of the problem with complexity. Referencing from previous issues with complexity, with the increase of complexity there is an increase in the likely chance of error. This would mean that with the increase of complexity in our technologies, the more error that comes into existence the bigger impact that changes have upon these technologies. The solution the problem is not to increase the speed of processors through means of compacting so much on a processor, but to improve the methods of how processors process the information, such as increasing the amount of levels that the processors process the information. This method would not increase the complexity, therefore - making the method an efficient way to transferring and storing information. The Schafftarian matrix method would definitely increase both speed and size of memory and processing without having to increase the complexity of memory and processing. This would reduce the complexity of processors and add simplicity to the system of processors and memory.

For memory, the Schafftarian matrix algorithm could help where instead of storing information on one specific level it can be stored on multiple levels, which would take up less space. This

technique is also used in genetic information. In DNA, especially human DNA, all the genetic information can be stored in small chromosomes, which are smaller than even the technologies that we can develop. This is due to how genetic information works with each nucleobase and attempting to be efficient with using these nucleobases to form a structured organism. If this type of technique can be used to form better technologies for computation, high-powered computers can be generated that are the size of or smaller than your finger. This improvement in technology would bring about new ideas among the technologies that exist in our modern day world. It would also improve the development of artificial intelligence because information can be stored more efficiently than with a one-layered system. With artificial intelligence, processing of information would be quicker because of how many layers the information could be stored on at the same time. Instead of dealing with the limitation of step-by-step processes, there could be multiple processes happening at the same time. Instead of increasing the speed to carry out the processes done on one particular level, the calculations and processes should be done on multiple levels, more specifically six levels.

$$S_{\Lambda} = [0,1,0,1,1,0,0,0,0,1,1,0]$$

$$S_h = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{vmatrix} \qquad S_{\chi} = \frac{N_b}{k}$$

This switch from a complex system to a simplistic system becomes very useful for competition between compacting processes into faster systems and begins to support efficiency. This aim for simplicity and efficiency brings more abilities to technologies and fast computations. This not

only brings ability to artificial technologies, but also allows biological computations. Using biological materials to compute algorithms would be easier because they are easily disposable and are less expensive than producing the technologies that take longer to make and compute less, but still cost more because of the technologies involved in the production of these technologies.

The human brain, being a highly complex system that can do computations way faster than a regular artificial computer, is still a mystery to how it is able to store memory with all the information that is inputted into the brain. Binary code is, obviously, not the type of system that the brain uses to store information and it is not likely that the brain stores memory on a one-layered system. In this case, the brain may even use a six-layered Schafftarian matrix to store memory. Since the time that the memory inputted into the Schafftarian matrix is identical to all the layers, the input values are more efficiently stored. With the ability to store and process memory with such efficiency, the mind can store more memory in very little space. Since the Schafftarian matrix is what follows this idea of multiple levels, efficiency is what the human brain tries to achieve. Since the human brain follows, hypothetically, the Schafftarian hypothesis, this would mean there are an already defined set of formulas and equations to calculate the properties of the human brain with complexity values, randomization values, proportional values, and proportional complexity values.

However, there are even more equations that can result from the Schafftarian matrix being the set model of the inner workings of the human brain. If this is true, it would make predicting human thought more easily done than ever before. For example, referring to a specific area of memory within the Schafftarian matrix would look as such.

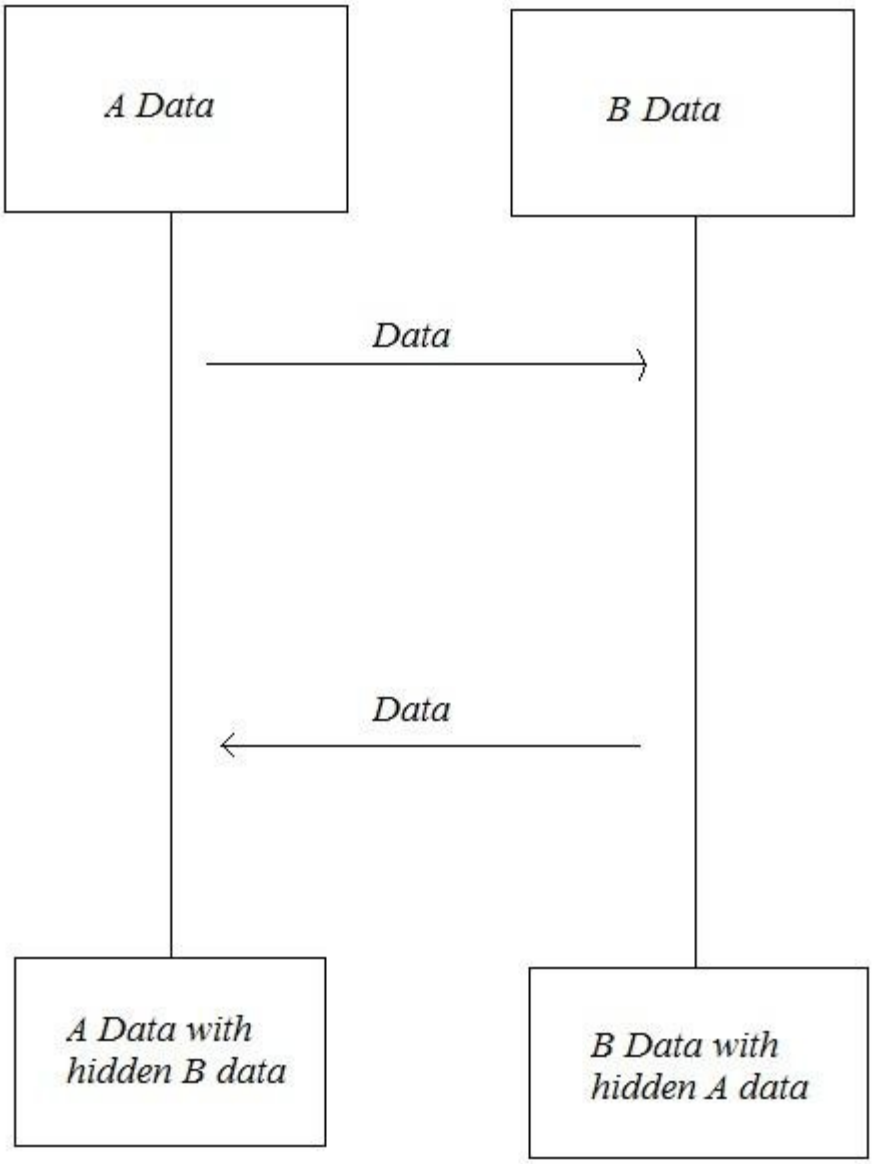
$$A_{\omega} = P_{\frac{nk-nu}{Sy}}$$

This formula pinpoints where a memory within the Schafftarian matrix is stored. However, this assumes that the human brain follows a second dimensional Schafftarian matrix. The problem this notion of a second dimensional Schafftarian matrix is though it assumes efficiency, it also has less memory that can be stored within its properties. Therefore, the other possible solution to the brain functionality with memory storage and development, the Schafftarian matrix could actually be a third dimensional Schafftarian matrix. The three dimensional Schafftarian matrix would provide enough memory and processing room in order to store information quickly and efficiently. A three dimensional Schafftarian matrix would allow more maneuverability for information and allows more unique algorithmic rules than the two dimensional Schafftarian matrix, meaning that the brain has some uniqueness to its memory storage and processing. The only issue with this depiction of the brains fundamental method of memory and storage is the efficiency. After the second dimensional Schafftarian matrix, complexity and efficiency begin to break down where the usual formulas for the Schafftarian matrix do not work for the third dimensional Schafftarian matrix.

Other issues involved in the three dimensional Schafftarian matrix include whether it follows linear or non-linear complexity. The two dimensional Schafftarian matrix, with most of its formulas, shows a depiction of a linear complexity involved in the complexity and proportionality within algorithms and languages. The three dimensional Schafftarian matrix may show different signs of complexity, where it could most likely follow non-linear complexity.

Since there are multiple rules that could be followed for a three dimensional Schafftarian matrix, there are millions of algorithms that could be used to interpret information for this method. If there are multiple algorithms that could be used for the three dimensional Schafftarian matrix, there would be no core algorithm to base the interpretation on. Therefore, efficiency is not involved in the three dimensional Schafftarian matrix, making it more complex than it needs to be. Unlike the three dimensional Schafftarian matrix, there is at least a set algorithm based on the structure of the matrix itself.

Now with the existence of a three dimensional Schafftarian matrix, it may be assumed that there are no more versions of the Schafftarian matrix that could be developed. This, in fact, is incorrect. Though all the spatial dimensions of the matrix have been implemented, there is another factor that could be added in to make the algorithm even better. Imagine that there are two senders of information that both use the three dimensional Schafftarian matrix to send information to an unknown source. If both senders send both of their pieces of data at the same time to a specific destination it is known as a network. Though the assumption is these pieces of information can only reach their destination with no other effect impacting their data structure, there is something hidden with network information such as this being sent to a vast database of information. These two pieces of information could some form of interaction between them, affecting their data structure, which then these pieces of information are like living organisms within a system. This is where this type of Schafftarian matrix comes into play. With data interactions, these pieces of data can pick up bits of other pieces of data. This would mean that pieces of data could adapt to specific changes with the change of a system.



Data can pick up bits and pieces of other interacting data

In reality, data is interacting all of the time. This occurs especially within quantum mechanics where information from particles is always being entangled with other particles caused by the entanglement of particles. This type of system acts similar to what happens when someone touches an object with their hand. When a person touches an object, data is actually transferred to the object because when the person touches the object data, such as their genetic information

or fingerprint, is transmitted onto the object they touched. Even though it is not noticed in reality, data interaction occurs with humans as well. When someone passes by another person, even when they did not intend to interact, there is information that is left on the first and second person. For example, the person passing by the other may get a glimpse at what this person looks like. The same thing occurs with the other person as well, getting a glimpse at what the other person looks like. Even on the quantum level, there is information interacting with other particles that interact with the information. There are good and bad things about this type of interaction between pieces of data. With the interaction of data, there could be ways to predict easily what the source was for piece of data just by looking at other pieces that had interaction with it.

However, there are some negative sides to this concept. With computer security, such as firewalls and such, when people attempt to break into a computer by bypassing the security, the security software involved in protecting the computer filters out threats that attempt to break into the computer and stops the security threat from there. However, with this concept, this only threatens the computer even more. Since data interaction causes bits and pieces of data to fall upon other pieces of data, the security threat could attempt to break in the first time and get the information needed to break into the computer. It is common for this to happen in modern day technologies, where hackers attempt to carry out a dDoS, or denial service, attack and when the security or server that is attacked has to restart, the information with rebooting the security or server is used by the user who is hacking and the user carrying out the attack gets access to the information and then carries out what they attempt to do. However, there are more things involved, but it will not be covered because it is not as important with discussing this concept. Therefore, with this idea, it would become more dangerous when developing security to protect a

computer because the more powerful security software is the harder the impact will have when someone attempts to break into the computer.

Besides the negative effects of the concept, it could actually make the transferring of network data easier and even safer. Since data follows a more complex line of transferring, it would take too long to decrypt the data. Instead of having security with filters, have data with filters. With certain signatures, the data could be accessed, but if out the signature reading the encrypted data would be close to impossible. This is where linear and non-linear complexity becomes a problem with this technique of secure data with the Schafftarian matrix. With two competing ideologies that are the complete opposite of each other's concepts, if linear complexity is the true form of complexity there is no problem with this technique of secure data because signatures would be able to compare to the signature of the piece of data without a problem. However, if non-linear complexity is the form of complexity it would be more difficult to compare a signature with a piece of data. Even if both linear and non-linear complexity were the true forms of complexity, there would still be a problem with determining if only one signature works with deciphering the piece of data to be read.

Linear complexity

$$S_g = D_1 \Rightarrow \begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix}$$

Non-linear complexity

$$S_g \approx D_1 \Rightarrow \begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix}$$

Another issue with non-linear complexity being the form of complexity with this idea is with approximations. Since the signature, with non-linear complexity, can only be an approximation, there would be no way to determine if the approximation was the true form of the signature to be able to read the data. However, it would not mean that if non-linear complexity is the true form of complexity that it would be impossible, though the efficiency would decrease massively. This would be due to how the system, in this case the data, would have to adapt to changes in certain parts of the signature, meaning that the data itself would have to change without changing the meaning of the data.

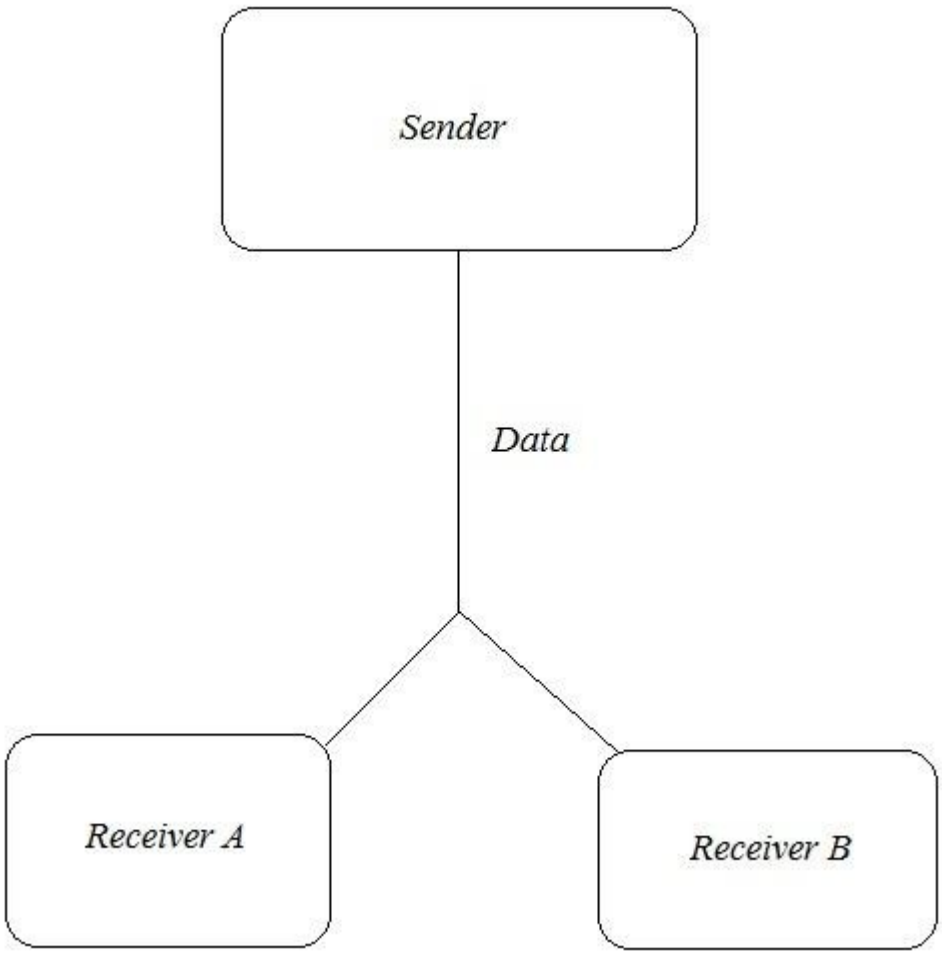
$$S_g \approx D_1 = \begin{vmatrix} \Delta A & \Delta B & \Delta C \\ \Delta D & \Delta E & \Delta F \\ \Delta G & \Delta H & \Delta I \end{vmatrix}$$

$$S_g \approx D_1 \to \sum_{n=0}^{\infty} \begin{vmatrix} \Delta A & \Delta B & \Delta C \\ \Delta D & \Delta E & \Delta F \\ \Delta G & \Delta H & \Delta I \end{vmatrix}$$

The data would have to infinitely change according to the small change occurring in the signature. This would be highly inefficient because there would be no way to determine if the signature was coming from the correct receiver or not.

The main problem here is determining if the receiver of the information was the correct receiver of that information. Though a simple question, it is a complex and philosophical question of whether the receiver of the information was the correct receiver, or rather if the receiver should have received the information in the first place. The question would be whether there is a mathematical way to be able to determine if the receiver was the correct receiver of the information. One way to do this would be to have the sender always being able to check if the receiver was the correct receiver of the information, but the problem with this is being able to send information without having to worry whether the information got to its correct destination. A major problem with the idea is in order to determine if the sender desired that information to go to a specific destination it would require that there was a hidden piece of data within the information that was sent that determined whether the receiver had actually received the information or not, which is similar to the idea of signatures. With the idea of signatures, it the

issue of whether other receivers would have access to that signature. It would be a difficult process of being able to determine if the signature came from the right receiver.



To whom does the data belong?

Again, being able to determine the correct receiver mathematically would mean understanding the algorithm of human thought. Analyzing human thought and discovering the human thought algorithm would efficiently allow a mathematical equation to determine to whom or what that data really belongs. This, of course, would only have to be done mainly if non-linear complexity

is the correct form of complexity. Even with linear complexity, however, there would still be issues with this problem, which in itself falls within the spectrum of *NP* problems.

Conclusion

Computation with a hypothetical Turing's machine seems quite simple, with the language of binary being used and only consisting of two types of inputs, which are ones and zeros. Once more complex problems occur, however, the efficiency of computation becomes far less than what should be used to accomplish to task with. Binary, being a one-layered language, is only useful when the amount of computation needed below the bounds of complexity for binary. In order to increase the amount of complexity a language can withstand, new levels or layers have to be given to a language. Genetic information is one of these examples, where the goal is to fit as much information into very little volume as possible. Genetic information relies on six layers as a language to carry its function within the human body and in many other organisms. Interesting enough, the number of layers being six is the perfect number of layers needed to make such a complex, but efficient language. Having more layers for a language, especially genetic language, would have caused more complexity and less efficiency because in order to fit so much information into one small volume or area there must a perfect structure that can withstand being "squeezed" into such a small space.

With such a simplistic build-up, it would require such a complex intelligence to design such a structure with such perfection. This would back up the claim of linear complexity because with linear complexity must have a sender in order for there to be any feasible information for the receiver. In order for there to be intelligence behind information, there must be intelligence behind the information. This would also include the concept of data interaction. With the

interaction with one piece of data with another piece of data, there will always be bits and pieces of data clinging onto the other piece of data, which then leaves behind a fingerprint of the original piece of data. This would mean that with a sender that has a form of intelligence the information will always have a form of intelligence, while a sender with a chaotic system will come out with the corresponding type information, which would be chaotic information or known as randomness. Of course, with non-linear complexity, since *P* problems can and cannot equal *NP* problems, the same could be said about the connection between the information and its sender. If *P* can at times equal *NP* while at other times not equal *NP*, this would also mean that the information at times shows a connection with its sender while at other times it does not. This would mean that there would be no way to prove that information came from an intelligent source or not, which would also mean that nothing could be proven. This concept would have to be incorrect currently because there is no evidence that the information cannot always be linked to its sender. Also, without the proof that one-way functions exist there is no way to prove that information can at times not have any connections with its sender. This would have to mean that information cannot be lost, since the information of the sender can never be lost, according to linear complexity.

This brings up the question of whether information can ever be lost when it comes into existence. The idea could be similar to the concept of conservation of substances, where nothing is gained or destroyed but simply moved or just modified. With information, no data is simply just created or destroyed, but moved from location to location or modified to not look similar to the original piece of information. This conservation of information would prove to be useful, especially to finding “missing” information where it may least likely be found. For example, when a magnet is hovered over a hard drive, the information on the hard drive becomes scrambled and is

unreadable. However, the information still exists with it being scrambled and modified the way it is. Conservation of information is a very interesting concept because how information, no matter how complex, is still readable in any form because the only thing that needs to be done is find the correct way to interpret the information. Genetic information is a very good example of this situation. With the complexity of genetic information, it is difficult to understand that there is feasible information within DNA. However, there is, in fact, information hidden through the sequences which allows the human body to carry out its processes. This applies to all information that exists, where information is never lost but only moved or modified.

Conservation of information is also one of the properties of linear and non-linear complexity. Linear complexity accepts the idea of conservation of information because it follows the existence of one-way functions, meaning that evaluating a problem and finding the solution would result in not being able to reverse the solution to the problem. Non-linear complexity does not support the concept of conservation of information do to how non-linear complexity handles data. Since the input, with non-linear complexity, does not always have proportionality with the output there does not have to be any "fingerprint" left behind by the information that was affected by a source.

In this particular case, the examples presented for linear complexity would be the opposite, where if someone touches an object a fingerprint is not left or when a magnet is hovered over a hard drive that data is lost forever, which is not the case. Since there is no concrete proof that information can be lost, this would be one proof that non-linear complexity is not the true form of complexity. Even in the world of computers, data is always having interaction with other pieces of data with the existence of networks and connections between computers that share

information. In systems that have data interaction, the data that interacts seems to become blended in with the data that it has interacted with. This is why, when you change an environment of a system, the system adapts to the change that had occurred. When a system changes, data interaction occurs and the data grows connections with the other pieces of data within the system. This happens in nature all the time where a new species is introduced to a certain habitat, and while the new species evolves slowly to survive the habitat begins to form connections with the new species that was introduced into the environment. This not only applies to natural systems. It can apply to many other systems that exist. Mathematical systems follow along the same type of process, where a new variable is introduced and the equation or formula grows connections with that specific variable. It can also apply to computer systems, where with the increase of new data the computer has to adapt to the amount of data that is on the computer and the increase in new software that enters the computer.

Though, the issue with current artificial intelligence is the limitations of being able to make connections with its environment and being able to adapt to these changes. When developing artificial intelligence, the issue with it is not being able to connect with its environment, especially when developing human-like artificial intelligence. The main reason why artificial intelligence has not been able to do this is due to the limitations in memory. Unlike computers, humans have the ability to store millions of bits of data that comes from the senses, such as sight, hearing, taste, and the other sense that humans have. However, with the two or three dimensional Schafftarian matrix, the memory problem might be solved, which can bring about the existence of data interacting artificial intelligence. With this concept, it could also bring about the development of data interacting software. This would mean that software is able to interact with the other software that exists on the same computer or interacting with the user of the computer.

This would improve technologies today because software could adapt to the changes of a computer without having to deal with strict bounds of error caused by the slightest change. If the bounds and limitations are removed, there are many more possibilities for artificial intelligence.

Not only do these concepts improve the development of artificial intelligence, but they also improve the ways of security. With new innovative ways to develop security, it becomes more difficult to break security on machines. In today's technology, most algorithms of security rely on sequences that do not produce completely unique codes to protect a machine. If an algorithm were to be able to produce completely unique values for security, the algorithm used would not be as predictable. With the increase in ability of adaptive systems, the strength of security will improve, providing safer and more reliable machines that can protect themselves from possible threats, such as viruses. Like the human body, this new concept of interactive data would provide a machine with the capability to have its own immunity to viruses. With this ability, there would be no need for continuous updating of antivirus software because the computer would adapt to these new viruses and grow immune to them.

With the growth of new technologies, there has to be a time for new development and more efficient methods of developing these technologies. If the same path is kept for developing technologies, there will soon be a slowdown in innovation and soon no more improvements can be made with the same methods of development. Binary is the main language for computation with artificial intelligence, but the problem with it is the inefficiency caused by the limitations of binary. There is another language that goes along with binary, which is a quantum mechanical language called Ternary. With this language, there are infinite possibilities of sequences that could be made from this language because it works on the quantum level, where there are an

infinite amount of possibilities. Therefore, it follows along with the concept of non-linear complexity. The issue with the Ternary system is the fact that there are an infinite amount of possibilities that more than one sequence means the same thing as another set of sequences. Again, it falls in the category of multiple equalities, where something can at times be equal and at other times not be equal. Time and time again, this is proven to be an inefficient system because of how non-linear complexity works with efficiency.

As there is a growth in technology, there needs to be a growth in innovation. Without one, the other cannot improve. New methods and ideas are needed to improve technologies because with following the same method of computation and when the only goal is to improve the current day technology, there will be a limitation to these specific methods where there is only so much that can be improve upon a method of computation. This type of ideology also can be applied to the Schafftarian hypothesis, where even with this new view of efficiency there is still room for improvement. Once Schafftarian hypothesis is improved upon, where efficiency has reached its limit, new ideas must be presented supporting innovation and new idea to make technology better than its original form. This follows the ideologies of science, where the goal is to continue to research the inner workings of the Universe and never stopping until the truth is known about how our Universe truly works. Innovation should be strived for, not just strive to improve technologies. Technology is an important aspect of innovation, but without both working with each other neither of them will be improved, creating a halt in the production of a better society. A major connection with the improvement of technology and the increase in innovation is the idea of complexity. With the improvement of technologies, there is an increase in complexity. Innovation attempts to decrease that complexity caused by the improvement of technologies. Without the increase of innovation, there is a major problem with complexity within

technologies. With the increase of complexity, there is an increase in the probability of error. In order to decrease that probability, there needs to be a decrease in complexity, which means an increase in innovation of this technology.

With the increase in the production of technology, there will be an increase in the amount of error within technology. Error is a huge issue with the increase in production of technology, especially with an increase in demand for better technologies to better improve society. To improve society with technology, efficiency is the main goal. Since the point of technology is to increase efficiency, the goal needs to be efficiency, which can be accomplished with the increase of innovation. Only then will the probability of error with technology decrease. There needs to be a balance between technology and innovation to achieve this. Efficiency can only arise from the balance between two concepts, both being equally as important as the other. Without this ideology, the improvement of society cannot be achieved. Not only will the increase in innovation make a physical improvement upon society, but also a mental improvement. When new technologies arise from science, people must learn both the good and bad consequences arising from these increases of new technologies. It must be realized that even with the increase in technology there must be responsibility. This responsibility arises from the philosophy that people should new ideas and inspiration to help others, not to bring down the existence of humanity.

Implications of this technology

Technology is a good thing for the advancement of society because it allows humanity to observe its morality and understand how much of an impact technology is on the world. With the advancement of science, there must also be an advancement of morality. There must be a balance

between science and morality, neither over running the other. Equality between both will bring justice to both science and the existence of moral standards. With the continuation of science and its development, there is an increase in the “left mindedness” of society. When humanity strays away from understanding the consequences of science, only destruction will arise from humanity. Though one should reach for the heights and discover new things about our world, there must be a responsibility behind it. Responsibility and moderation are two main factors behind the existence of morality. Responsibility with our actions and taking understanding that technology should be used for the good of humanity, not to turn it into a tool of mass destruction. Moderation with our technologies must also be watched, where we must moderate ourselves to what we should and should not do with our innovations and inspirations. Science is meant for the discovery of understanding of our Universe, not to prove a point.

When science and personal belief become combined, it only corrupts the main foundation of what science truly is. Efficiency should be the goal of science and technology, not to spread a personal point of view. There must be a balance between science and moral because if science becomes the main focus then the true point of science becomes lost and there is no reason for the creation of new ideas. If society can achieve this ideology then the speed of innovation and technological development will increase rapidly. Only when this occurs will society progress and expand, which will show that the society is ready for expansion. If this does not occur, society will fail to continue on and soon will fall apart, defeating the purpose of unifying humanity. Unification is a big concept in society, where the goal to unite everyone despite difference. Technology is a big characteristic of society that has unified humanity, where there is a big network of communication between everyone all around the world. This unification of humanity brings about the fullest inspirations and ideas of the human race, where people are free to share

their ideas without the limitations of distance or time taken to spread that information across the world. Data interaction is a huge step in speeding up the process of science, innovation, and development of technologies. With the concept of data interaction, new ideas are formed, combined, and then used to design new technologies that could help improve society.

Science is a field of exploration and innovation, finding new ways to improve technology that exists in the modern day world. Not only does it allow people to discover and innovate, but it also allows people to explore moral implications. With the growth of humanity and the continuation of development intended for the good of humanity there are troubling situation that can occur. With the increase of pride on the increase of intelligence, there is a bubble of isolation that forms, where humanity thinks so highly of itself that it feels the need to separate itself from what is most important, which is to use science for the improvement of society. For example, developing artificial intelligence with human-like features should not be done with the intention of either destructive or immoral conduct. The intention of this technology should be to better improve society, develop cures for diseases, and better the lives of people. With the increase both technology and innovation, the more humanity needs to increase in responsibility of using the technology and ideas for the correct purposes.

The information of this hypothesis can be used to develop artificial intelligence and practically can help find ways to cure diseases with the genetic part of the hypothesis. While it can be used for good purposes, it could also be used for bad reasons. With being able to develop biological instances with the genetic algorithms, both biological medicines and biological weapons could be made by programming cells to either help destroy bad cells or to go to healthy cells and completely destroy them. Also, with artificial intelligence, new forms of computation could be

done with the hypothesis. With this in mind, new software programs can gain themselves an artificial intelligence, where software could soon think on its own without any other user guidance. Also, observation of humans would finally not be limited because instead of having to use humans as the case to study, human representations that are artificial intelligence could help predict statistics and be able to predict events with human interaction. This same concept of experiment could be done to study systems of either virtual or physical organisms. With virtual organisms, experiments could be done to compare the interactions with virtual organisms with other virtual organisms, and then perform an experiment with a virtual organism and a physical organism. These types of experiments could help develop better artificial intelligence because with the examination of interactions it could help better understand data interaction with change and development of systems, especially systems that go through rapid change. This could also help with statistical analysis, which would analyze human statistic to get a better understanding of human systems and how humans interact with other humans and even with artificial intelligence.

With concepts of interaction, there is this hint of a type of system, where everything is actually unified into one system of language or code. In data interaction, there is this algorithmic sequence that controls systems with equality, where with the connection between two pieces of data there is unification between these two pieces of data creating a system with a specific personality. With data interaction, the code involved between the two or more “organisms” becomes unified as one piece of code, where changes occur within the system due to how the organisms interact or react with each other. Systems, after a period of exponential time, begin to develop a personality with the amount of changes within the system and how the organisms interact and react with each other. However, each system, even if developed in similar ways, will

have a probability of a very small percentage of being identical. This is what makes systems strange. The question would be what is the constant factor involved in the development of systems. This constant is what is behind the development of all systems. The constant is a small algorithm that is behind the existence of all systems. Think of systems as crystal formations, where there are millions of crystals within a cave. The crystals themselves represent the systems, where their formations represent these elements within the system. In this case, the cave represents the area of all existence for systems, where there is a set algorithm behind all these systems. There could be millions of these cases that have different algorithms, which would form systems on that specific algorithm. With this new concept, there could be new ways to experiment with the development of systems with the existence of artificial intelligence. This would allow us to gain a better understanding of how systems work and find the driving properties of systems.

With this information, it could help develop human-like artificial intelligence, where these forms of artificial intelligence can be self-aware and develop without the guidance of humans. This could help in the development in understanding human intelligence and interaction. This understanding would help develop new forms of safety, being able to predict human actions with the testing of virtual organisms that act similar to humans. In this range of development, observation of systems also could bring a better understanding how systems of organisms are formed in the first place and whether there must be some force behind this interaction between organisms that form environments. This implication of artificial intelligence would bring an evolution in technology that better helps humans without the guidance of humans. Even if this is all theory, there is a very likely change that it is the driving force behind the existence of systems, where data interaction becomes a huge driving force behind these systems.