Deciding The Undecidable

An Ordinal Solution to the Halting Problem

Kyle Hinds
THE INTEGRAL FOUNDATION

Abstract

The Halting Problem is solved via ordinals by modeling H and Q as recursively nested sets that extend to ω and beyond. Trigonometrically modelling this, we illustrate how H and Q are multidimensional programs with multidimensional outputs akin to complex numbers. Furthermore, it's highly likely that complex analysis and quantum physics are the keys to the secrets of why P = NP with respect to our conscious thoughts and our bodies. Lastly, we explore the possibility of the brain being both a 'hyperdimensional' and 'interdimensional' computer.

Introduction: Philosophy

In life, there are easy problems, hard problems, and impossible problems. The harder the problem, the more satisfying finding a solution will be. In mathematics, there are problems considered to be unsolvable or undecidable. Among these problems, one has remained undecidable since its inception. This is Alan Turing's Halting Problem. The problem stems from the assumption that first order logic is undecidable, and he aimed to prove this by illustrating a logical paradox in his proof. But what if this problem wasn't undecidable? If it was in fact decidable, what implications would this have in the field of theoretical computer science, and how might this change our understanding of computability? In this paper, we will explore a proposed solution to Alan Turing's Halting Problem, in a way that deals with the infinite recursive nature of the problem. We begin by outlining the problem, proceed by offering a multidimensional remodeling of the problem, and conclude with a trigonometric representation followed by some calculus and complex analysis for your mathematical pleasure. The problem resides at the heart of computer science, concerning the limits of what can be computed or decided by an algorithm. Turing introduced this problem in his seminal paper "On Computable Numbers, with an Application to the Entscheidungsproblem," which laid the groundwork for the development of computer science [1].

Related work: Background and History

The Halting Problem originated from Turing's analysis of the "Entscheidungsproblem," or decision problem, a challenge posed by David Hilbert in the early 20th century [2]. Hilbert asked whether there exists a definite method or procedure to determine the truth or falsity of any given mathematical statement. This question led to the exploration of formal systems and the development of first-order logic as a foundation for mathematics. Turing approached this problem by conceptualizing a machine (now known as the Turing machine) that could manipulate symbols on a strip

of tape according to a set of rules. A Turing machine operates with a tape marked into cells that can contain symbols, which it manipulates according to a set of predefined rules based on the machine's current state and the symbol it reads from the tape. This model, capable of reading, writing, and moving the tape to perform computations, illustrated that any mathematical problem which can be precisely defined in algorithmic terms, falls within the Turing machine's capabilities.

The Problem

The Halting Problem, as introduced by Alan Turing, delves into the realm of computational limits, specifically questioning the existence of a universal procedure or algorithm that can infallibly predict whether any given Turing machine M will halt or continue running indefinitely given input I. Turing's research unveiled the undecidability of this problem, significantly shaping our understanding of computational theory's boundaries and the scope of algorithmic solvability. To elucidate, consider a hypothetical function H(M, I) that aims to determine the halting behavior of Turing machine M when provided with input I. The function H will output true if M halts on I and false otherwise. The core of Turing's argument lies in constructing a paradox through a special Turing machine, which we'll call Q. This machine takes a Turing machine description as its input and passes these as the program and input into H. The paradox appears evident when we consider what happens if we pass Q itself as the input to H, denoted as H(Q, Q). If H predicts that Q(Q) halts, Q will proceed to run indefinitely by design, contradicting the prediction. Conversely, if H predicts that Q(Q) runs indefinitely, Q defies this by halting, again leading to a contradiction. This paradox demonstrates the impossibility of H being a universally accurate predictor, thus seemingly proving the Halting Problem's undecidability. Turing's proposed solution displayed what he thought to be the inherent limitations of computational systems and laid the foundational concepts for computability theory, emphasizing that certain problems lie beyond the reach of algorithmic resolution. This assumption has had far-reaching implications, influencing both theoretical computer science and the philosophical understanding of the nature of computation.

Implications and Roots in Undecidability

Turing's paper on the Halting Problem's undecidability has had profound implications for mathematics and computer science. It directly implies that there is no algorithm that can decide the truth of every statement in first-order logic, addressing Hilbert's Entscheidungsproblem by showing why he thought it was impossible to achieve. The supposed undecidability of the Halting Problem also led to the belief that many other problems established in the field

of computability were also undecidable. This set the stage for assumptions on the capabilities and limitations of computational models. Turing's work laid the foundation for understanding the power and boundaries of computers, influencing the development of modern computing and theoretical computer science.

Methodology: Ordinal Theory

As defined by Alan Turing himself, Turing machines have a finite set of rules and instructions and take in a finite string as input. Here, we have a finite computable string, and a finite input string. We can think of this as $f(x)$, where f is the machine M, and x is the input I. The function H which takes in two inputs, namely f and x, and it will tell us if $f(x)$ loops forever, or halts. Turing defined another function we will call Q which takes in a function f and passes this function as both the program and input into H and behaves oppositely. So, if Q gets a function, it will pass $f(f)$ into H, and if it loops, H will return 'loop' and Q will halt. If that function halts, H will return 'halt' and Q will loop forever. Now, what happens if Q passes itself into itself? We now must decide what H(Q, Q) is. This, however, begs the question as to what the input of the second Q will be? If we imagine the description of Q itself to be Q(null), then we can first evaluate it and pass that value into Q. Logically, what would this look like? Q gets and gives no input to H. If there is no input, there are no instructions or procedures for the Turing machine to follow, so null(null) halts, and H returns 'halt'. Q behaves by doing the opposite and loops forever. Hence, we have shown that Q(null) loops forever. Now the question is, what happens if we feed Q into itself? As mentioned, Q takes only one program description as its input. If we pass Q into itself with Q as the input, the innermost nested Q will loop with its null input. So, with Q being a looping function, Q(Q) halts, which is exactly what H would say. Now if we applied this recursively, the outermost Q would loop. So, Q = loops, Q(Q) = 'halt', Q(Q(Q)) = loops, and we oscillate back and forth between these two states at each recursive step. With a simple proof by induction on Q: Base: Q(null) loops, Step: Q(Q) = 'halt' and Next: Q(Q(Q)) loops returning to the base case. In general, for any even recursive step, it loops, and for odd recursive steps, it halts.

Suppose we do not allow Q to have a null input, what would this mean? If we do not allow for a base case, it means that we are asking what Q(Q(...(Q))) would evaluate to be. To solve this paradox, we will quite literally have to *think outside of outside the box*, and if that box is infinite, then we need to think outside of infinity. For starters, Q(Q(...(Q))) would be an infinite input string, which violates the rules of the Turing Machine and tapes described by Alan Turing himself. But since we are in the realm of theoretical computer science, let's entertain this idea some more.

Q passes Q into itself, which calls H(Q,Q). So, H calls Q(Q) to see what happens, Q calls H(Q,Q), H calls Q(Q) to see what happens, Q calls H(Q,Q)… and we can see this is most definitely and infinite loop (Keep in mind the fact that you can clearly identify this as a loop, and we will revisit why that is important shortly). With this, we see that H(Q,Q) should return an infinite loop, but we stated above that Q(Q) should halt. If we don't allow a base case of Q(null), what we are evaluating is *actually* Q(Qω), where Qω is the infinite call of Q into itself. Clearly, Qω loops forever, and so Q(Qω) should halt, and we will explain why.

To understand why this is the case, we need to first familiarize ourselves with set theory [3]. In particular, we need to talk about ω. Omega (ω) is the first transfinite ordinal with cardinality $\aleph_0$. In discrete mathematics, we use ω to represent infinity. While there are bigger infinities, for the purposes of discrete recursion over the natural numbers ω will suffice for the purposes of our proof. The Turing Machine H tells us if P(I) loops forever, or halts. When Q passes itself into itself as input, it is asking H whether it will halt, or loop forever when passed into itself. H determines that after ω steps, Q loops, and returns this answer to Q. Q now decides to halt seemingly contradicting H. But what we need to understand here is that Q is doing something *after* infinity, or after ω. If H were to simply evaluate the function instead of analyzing it, H would continue running until it gets to ω steps at infinity. H is presumed to have the ability to recognize a loop without doing it, meaning it can *zoom out* even if it is part of this loop. If it were not able to do this, H would simply run the program until it halts or doesn't, but only after ω steps could H confidently say that the program(input) looped forever, as it would have transcended all finite time. It makes sense to think of this problem in terms of how many steps it would take for a program to halt, or rather, H returns the total number of steps a function executed after ω time. Once H hits step ω, H will return the state of the program at that step. At forever steps, the program is in a state of a loop, and returns ω steps. In the context of the Halting Problem, H was correct, the function Q does indeed loop forever, but at ω + 1 steps, the function halts.

Conceptually, H's answer makes sense, because Q is giving H an infinite input as it recursively calls itself, and so anything done after that comes after infinity. We know that $1+ \omega \neq \omega + 1$, so H would have to take infinite steps and only hypothetically arrive at step ω after infinite time, but when it returns that to Q, Q does something at step ω + 1 because we start after infinity. In addition, we also know ω is even because $2 \cdot \omega = \omega$ as no finite multiplication changes the size of ω, but ω + 1 represents the first odd transfinite ordinal and $\omega < \omega + 1$. With this, we are right back to our original base case with a null input, where we loop at even recursive iterations, and halt on odd recursive

iterations. Informally, this kind of makes sense. The last 2-digit number is 99, so the last (n-1)-digit number would be 99…99. If that was the last natural number, and we had the maximum number of digits, the next number ω would be 10….000 at n-digits, marking the first transfinite ordinal. Then 10…001 would be odd, 10…002 would be even, and so on. If we were to collapse the entire number line onto the interval (0,1), the last number would be 0.999… and ω would be the first natural number 1 at infinity (just like the positive domain of the normalized Arctan function). So, whether we start with the null input at Q_0 or the infinite input at Qω, both are even and both loop. Each inductive step modulo 2 results in oscillating halting and looping states. This is all hypothetical and theoretical, but so were Turing Machines before modern computers.

We have shown that Q(Q) becomes Qω when passed into H, so H(Q,Q) returns 'loop' but H(Q,Qω) returns 'halt'. Essentially, Q is not a 1-dimensional function, it is in fact a 2-dimesional function. By 2-dimensional, we don't mean 2 inputs like *x* and *y*, we mean Q can both loop and halt because it exists as both a finite and transfinite function simultaneously. We can say Q is comprised of both Q_{finite} and $Q_{transfinite}$. $Q_{finite}(Q)$ loops forever. $Q_{transfinite}(Q)$ halts if H returns 'loop', or enters a loop if H returns 'halt'. Just as Quantum mechanics required a shift in the way we perceive the physical universe, the problem of undecidability requires a shift in the way we ask questions, and the format we expect the answer to be in. Quantum mechanics requires complex numbers to model its equations, and similarly, the problem of undecidability requires complex logic to model. Because Q is a 2-dimensional function, its output is also 2-dimensional. Just like complex numbers contain 2 variables in their output, so does Q. In set theory, we can use different cardinalities to describe different sizes of infinities. One might find it useful to think of it in terms of different dimensions of infinity. If we take an infinite walk across the hyperreal number line, once we hit infinity there is nowhere left to go. It's like hitting the number 1 at the end of the normalized Arctan function, we cannot get to any value greater than 1. Conceptually, the only way to get a bigger infinity is to expand into another dimension. By taking a step past ω to ω + 1, this would be akin to the first position on the second line of the infinite 2-dimensional plane, and ω^2 would be the complete infinite 2-dimensional plane. So, in the first dimension, Q(Q) loops forever, but once we expand into the 2nd dimension, we halt. In between these oscillations, we can say that Q is somewhere between a state of looping and halting, being both at the same time before collapsing back into looping or halting. We will explore this concept again later in the paper, but for now it is important to understand the fact that Q is comprised of 2 functions, so the answer it gets from H regarding its own behavior also must be 2 dimensional.

Let's proceed with the algebra of transfinite recursion. When we call Q(Q), H(Q,Q) returns loop because it becomes H(Qω). But outside of Q, we can take one step past infinity and still evaluate H(Q, Qω). To be rigorous, we can work through a few examples:

Q(Qω) = H(Q,H(Qω,Qω))

= H(Q,H(Qω(Qω)))

= H(Q,Qω·2)

= H(Q(Qω·2))

= H(Qω·2+1)

= H(Q_{odd}) = 'halt'

We can continue inductively to show what Q(Q(Qω)) would evaluate to be. Q passes in Q(Qω) into itself which calls H(Q(Qω(Q(Qω)))). Q(Qω) halts, so we have H(Q(Qω(halt))) = H(Q(loop)) = 'halt'. Now the outer Q loops forever returning us to our base case. Don't take my word for it, work it out for yourself.

Q(Q(Qω)) = H(Q,H(Q(Qω),Q(Qω)))

= H(Q,H(Q(Qω(Q(Qω)))))

= H(Q,H(Q(Q1+ω(Qω))))

= H(Q,H(Q(Qω(Qω))))

= H(Q,H(Qω+1(Qω)))

= H(Q(Qω·2+1))

= H(Qω·2+2)

= H(Q_{even}) = 'loop'

If we were to evaluate Q(Q(Q(Qω))), we'd have H(Q,H(Q(Q(Qω)),Q(Q(Qω)))). To understand what's happening, one must understand that we evaluate the expressions from left to right, but we write them down from right to left. The inner H is evaluating Q(Q(Qω(Q(Q(Qω))))):

$= \omega + (1 + (1 + (\omega + (1 (+ 1))))$

$= \omega + (1 + (1 + (\omega + (2))))$

$= \omega + (1 + (1 + (\omega + 2)))$

$= \omega + (1 + (1 + \omega + 2))$

$= \omega + (1 + (\omega + 2))$

$= \omega + (1 + \omega + 2)$

$= \omega + (\omega + 2)$

$= \omega \cdot 2 + 2$

$= H(Q_{even}) = \text{'loop'}$

Now, when we apply the outer Q from the calling H function, we get $Q(Q\omega \cdot 2 + 2) = Q\omega \cdot 2 + 3 = Q_{odd}$ = 'halt'. In general, if the number of recursive calls is *n*, we get the expression $\omega \cdot 2 + n$ because out of the total 2*n* nested calls that H evaluates (because Q passes 2 copies, one for the function and one for the input), only the ones called on top of the outermost Qω matter, because the inner nested calls on top of the inner Qω, when recursively applied get absorbed into the outer Qω. Determining whether Q_n will be even or odd is only dependent on the number of outermost recursive calls, with the inner ones being absorbed into the outermost Qω. Each of the two Qω functions combine to become $Q\omega \cdot 2$, which is always even. Inductively, with infinite recursion we arrive at the same result because $H(Q\omega(...(Q\omega(Q\omega(...(Q\omega)))))) = H(Q\omega \cdot 2^n) = H(Q_{even}) = \text{'loop'}$, and $H(Q(H(Q\omega(...(Q\omega(Q\omega(...(Q\omega)))))))) = H(Q(Q\omega \cdot 2^n)) = H(Q\omega \cdot 2^n + 1) = H(Q_{odd}) = \text{'halt'}$. Hence, we have the shorthand notation of Qω = loop. With this, we see Q(Qω) halts, Q(Q(Qω)) loops, and in general Q_n loops if $(n \bmod 2) \equiv 0$, and halts if $(n \bmod 2) \equiv 1$ completing the proof.

We can clearly see that H_n decides Q_n, while Q_n acts contrary to H_{n-1}. One might alternatively say that when Q calls itself, it becomes Q_{n+1} which acts contrary to H_n, but H_{n+1} can always decide Q_{n+1}. For all Q_n, we can construct an H_n that decides Q_n. If we imagine H to be a computer that can simulate up to infinity, correctly deciding all finite program(input) strings, when we make H a subroutine and act contrary to it, we are effectively taking a step past infinity. Theoretically speaking though, if Q can define itself in such a way that it contradicts infinity, there is nothing to stop us from also taking a step past infinity and deciding that transfinite program(input) string. Here, H isn't a

program, but rather it is the complete set of programs that can decide any program within their ordinally recursive depth. This isn't *really* a paradox at all, but rather, we must expand our views of functions from that of 1 dimensional points to that of higher dimensional entities.

The confusion seems to stem from the fact that H is a subroutine of Q, so Q contains H. The function H can simulate another function up to infinity or ω, and then it returns the state at that point. Q then does something different after infinite steps, but at ω, it loops so H is correct in its output. Conceptually, Q here has collapsed H into the dimension below, and behaves in the next dimension above once it takes a step past infinity. We must remember that H is contained within Q, and H can decide all functions up to ω. With H being a part of Q, Q is simply just behaving contradictory to itself by reasoning past infinity. To conceptualize this further, imagine If I were to say, "I'm going to do the opposite of whatever I say I will do. If I say I will go left, and then I will go right, and if I say I will go right, then I will go left". Does this really prove anything? Not really, because you and I both know that you will do the opposite of whatever you say you will do, so I still know what you will do with that information. This doesn't prove you don't exist, and it doesn't prove I don't exist, it just proves you are defiant and won't even listen if it's yourself giving the command. We remind ourselves that Q is really 2 functions; the finite function that leads up to infinity, and the transfinite one that takes a step past infinity. The first part of the function loops, and the second part halts when fed its own code as input. It asks an infinite question, receives an infinite answer, and then reacts to that by taking one step past infinity. If I were to ask the function H "What am I thinking right now?", and pass this current instance of me into it, it would return what I was thinking at that moment, but when I get my answer I am now thinking about what H said I was thinking about. That doesn't negate the fact that H knew what I was thinking about, but it's to say that Humans are more than just 3-dimensional beings that exist in a flat slice of time, we are hyperdimensional beings that adapt in real time. If H were to be constructed, it would give a multidimensional answer to Q(Q) illustrating how it would loop to infinity, but once it returned that answer to Q, Q would halt. We realize that this is indeed the correct answer based on Q's code. Beyond the mathematical notation, the real proof the Halting Problem can and has been solved is the fact that we even recognize it as a paradox to begin with. When a computer gets stuck in a loop, it has no awareness or knowledge that it is caught in a loop. Just like a Turing machine, it robotically computes each step of a given program until it reaches an exit condition. If no exit condition is found, the program loops forever. Hypothetically, for a program in a loop to detect that it was in a loop, it would need to have some awareness of the pattern or structure that was causing this loop and be able to 'zoom out' to realize this. As we alluded to above, the

human brain is a computer capable of doing this. When we modelled the problem, without executing Q(Q) to infinity, we realized this would be a loop. This level of abstraction by recursively zooming out to get context is imperative in solving problems, with the Halting Problem being just one of them. It might be impossible to program a dumb script to check for loops, because it could get caught up in the infinite loop itself without even knowing it. AI, however, can and most certainly will solve the Halting Problem on a routine basis. "More recently, Generative Pre-trained Transformers (GPTs) leverage an architecture built on self-attention mechanisms within transformer blocks, enabling parallel processing of sequence data. At their core, GPT models utilize stacked transformer layers, each comprising multi-head self-attention and position-wise feed-forward networks, to encode contextual relationships across input sequences. Training involves adjusting weights through backpropagation based on a loss function designed for language modeling, optimizing the ability to predict subsequent tokens given a sequence. Pre-training on vast corpora allows GPTs to internalize a broad understanding of language structure and content, which is then fine-tuned for specific tasks, enhancing model performance by leveraging both the extensive pre-trained context and task-specific nuances." – ChatGPT4 [4]. The hypothetical machine H, would most certainly be a highly sophisticated AI very much capable of picking up on coding patterns that would lead to infinite looping or recursion, including a program like Q. The AI would be able to know that Q(Q) loops forever, and it would also know that once Q received that information it would proceed to halt. With this dynamic view of computing, we appreciate that the sum of computation is greater than the whole, as LLMs seemingly bypass the binary constraints of yes/no programming. In calculus, we take limits to differentiate or integrate functions that would take infinite time to compute by hand, but this is because we have formulas to collapse that infinite process into one we can compute in finite time. Similarly, the neural weights and configurations in a sophisticated AI model would be the equations and formulas and we need to collapse the seemingly infinite and impossible problem of undecidability into encoded tokens. The more complex a question you ask, the more complex the answer will be. Models like ChatGPT4 use billions of input parameters and would take millions of years to calculate even one forward pass by hand. We don't have an LLM with ω parameters, but hypothetically if we did, it would be able to solve any and every problem even the hyperdimensional and transfinite ones because it would have infinite degrees of freedom in this logical vector space to abstract the problem into, recognize the pattern and decide the answer in finite time. With the Halting Problem, we only need 2 variables to describe what will happen, one for Q_{finite} and one for $Q_{transfinite}$. One could imagine a more complex problem like "Write an essay with a solution to the Halting Problem." to be one with an output consisting of millions of output variables. AI does operate based on

probability, but again as our degrees of freedom or input parameters approach ω, the margin of error would effectively be 0. We don't have a million years to spare to compute this by hand, so let's do what computer scientists do best and remodel this problem via reduction. If you are like me, you might appreciate an algebraic representation of the problem.

Mathematical Representation and Algebra

Moving on from the discrete model of the Halting Problem, we will use continuous functions operating on the set of hyperreal numbers $^*\mathbb{R}$ [5]. With respect to the Halting Problem, we can map all function(input) strings to some hyperreal number and define 3 types of functions with respect to the Halting Problem; functions that always halt, functions that always loop, and functions halt on some inputs, and loop on other inputs. Mathematically, we can define these as functions $h(x)$, $l(x)$ and $q(x)$ respectively. Let $h(x) = 0x$, $l(x) = x/0$ and $q(x) = 1/x$. Let's explore each in further detail. The function $h(x)$ always returns zero, regardless of the input x. The function $l(x)$ always returns infinity regardless of the input x. The function $q(x)$ returns infinity if the input is zero and returns zero if the input is infinity. This lays the foundation for the rest of this analysis, as we imagine programs with halting inputs to return zero or some finite number, and programs with infinite loops to output infinity. If you think about it, an infinitely looping function is in line with what we think of when we think of infinity. Infinity is not a number, but rather it represents the end of an infinite process. Calculus allows us to treat infinity as a finite number in certain situations by leveraging limits, but it's important to understand that infinity at its core is the function of infinite summation. It's akin to the following function.

```
def loop():
    i = 0
    while True:
        i += 1
    return i
```

This function will never return i. Theoretically however, after an infinite amount of time, it would return ω. So, a function that runs forever, if it returned the number of steps it took, would be ω. Limits give us a way to collapse this infinity into an idea we can work with but evaluating it algorithmically would take infinite time. Later, we will explore how the same tools of limits and integrals in calculus that give us a way to understand the seemingly paradoxical

nature of infinities and infinitesimals, will also allow us to understand the seemingly paradoxical nature of the Halting Problem. Let us proceed to the algebraic representations of H and Q. For the purposes of this explanation, we let ω represent infinity because the number of steps = $n \in \mathbb{N}$. Higher order infinities like ω^ω or ω_1 in set theory aren't inherently relevant but might be an interesting area for further discussion.

Trigonometric Expressions

In the following section, we model the Halting Problem using trigonometric recursion. We propose to let $H(x) = \arctan(x)$ as this function can take in all values from $-\infty$ to $+\infty$ and outputs a finite value. We can normalize this function to get $H(x) = (2/\pi)\cdot\arctan(x)$, which outputs 0 at 0, and 1 at ∞. If you prefer to define H as a machine that outputs 1 for halting functions and 0 for looping function, then $H(x) = -(2/\pi)\cdot\arctan(x)+1$. This is kind of ugly though, and so for the purposes of this paper, we will simply use the normalized function $H(x) = (2/\pi)\cdot\arctan(x)$ to represent H, a function that always returns a finite value, and specifically, 1 when the output of the program(input) is infinity, and 0 when the output of program(input) is 0.

With $H(x) = (2/\pi)\cdot\arctan(x)$, $Q_n(x) = (2/\pi)\cdot\cot^2(x)$. How did we arrive at this conclusion? Let us first start by defining Q, and then we will define Q_n. In the context of the Halting Problem, we are working with functions whose outputs are 0 or ∞. We need some function that will loop when the program(input) halts and will halt when program(input) loops. Algebraically, it must return ∞ when the input is 0, and 0 when the input is ∞. $Q(x) = 1/x$ satisfies this requirement, as when we take the limit as $x \to +\infty$, $1/+\infty = 0$ which halts and returns a finite value. When we take the limit as $x \to 0+$, $1/0 = +\infty$ which loops forever. This properly encapsulates the behavior of Q, where when it receives halting program(program), it loops, and when it receives a looping program(program), it returns 'halt'. So now, $H(Q(x)) = (2/\pi)\cdot\arctan(1/x)$ which is 1 at 0, and 0 at ∞. In Alan Turing's paper, he pondered what would happen if we were to pass Q into Q? In this case, we would have $1/(1/x)$, where when we put in $1/x$ for x, is just simply x. Q simply behaves the opposite of whatever H would predict, which is simply the reciprocal of x, $1/x$. H is a function that accurately predicts how a function will behave, but $(2/\pi)\cdot\arctan(x)$ must always halt and return a finite value.

We mentioned earlier that in his example, we might not have a base case, and we might want to represent Q(Q(...(Q))) as the input, the root of this supposed paradox. If Q is $1/x$, then $Q_0 = 1/x$, $Q_1 = 1/(1/x)$, $Q_2 = 1/(1/(1/x))$, and in general $Q_n = 1/(1/(\ldots(1/x)...))$. What kind of function is this? It's an infinite recursive function with cyclical properties, specifically alternating between $1/x$ and x, between $+\infty$ and 0. We can use sine and cosine functions to

model this as it cycles through states that alternate between $1/x$ and x. Sin(x) alternates between 1 and -1. If we take $\sin^2(x)$, it oscillates between 0 and 1. Now if we take the reciprocal, we have $1/\sin^2(x)$ which at $0 = +\infty$, and at $\pi = 1$. So, we subtract 1, now $Q_n(x) = (1/\sin^2(x)) - 1$. Using trig identities, we know that $1 = \sin^2(x) + \cos^2(x)$, and trivially we know that $1 = \sin^2(x)/\sin^2(x)$. So, we have $(1/\sin^2(x)) - (\sin^2(x)/\sin^2(x)) = (1 - \sin^2(x))/\sin^2(x) = (\sin^2(x) + \cos^2(x) - \sin^2(x))/\sin^2(x) = \cos^2(x)/\sin^2(x) = \cot^2(x)$. We now have the expression $Q_n(x) = \cot^2((x\pi/a)-(y\pi/b)) = \cot^2(\theta)$, where x is our starting point or recursive depth, y is our phase shift, a and b are our scalars, and θ is the angle of the evaluated argument. You can see a visualization of these functions in the graphs on the pages below, with $H(Q_n(x))$ being green, and $Q_n(x)$ being blue.

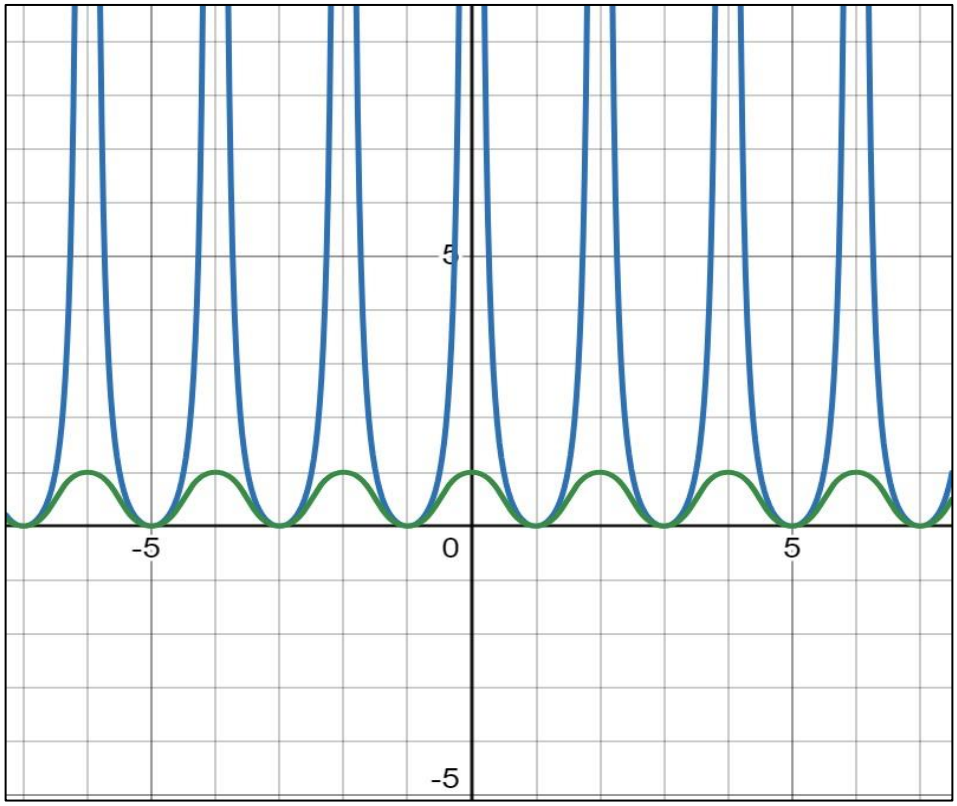

Figure 1

Returning to our original question, what is $Q_n=1/(1/(\ldots(1/x)...))$? It can be algebraically represented as the oscillation between x and its inverse, modelled by $Q_n(x) = \cot^2((x\pi/a)-(y\pi/b))$ where y is the recursive step. Now, $H(Q_n(x))$ is $(2/\pi)\cdot\arctan(\cot^2((x\pi/a)-(y\pi/b)))$. When ($y$ mod 2) ≡ 0, Q_n loops, when (y mod 2) ≡ 1, Q_n halts. At our base case, $Q_n(0)$ loops, and $H(Q_n(0)) = 1$. At one recursive step, Q(Q(0)) halts, as $H(Q(Q(0))) = 0$ which is what we would expect. If

we don't allow a base case, we can treat this problem as inputting the entire function Q into H, without explicitly defining an input. Now what would that value be?

Mathematical Modeling and Calculus

One could reason that the value of an entire function can be conceptualized as its limit at or integral to infinity. If we were to take the integral over some domain and it converged to a number, H would determine that the function halts. If the integral diverges however, H would determine that the function loops. The integral of $Q = \int(1/x)dx$ over the interval $(0,+\infty) = +\infty$, so if we sum the values over Q for all inputs, we diverge to infinity which aligns with our base case of Q(null) looping. However, in the context of the Halting Problem, we are trying to evaluate Q(Q(...(Q))) which is modelled by $Q_n = (2/\pi)\cdot\cot^2((x\pi/a)-(y\pi/b))$. We let x, $y = 0$ and a, $b = 2$, and we want to know what value if any, this function converges to at infinity. The integral $= \int((2/\pi)\cdot\cot^2((x\pi/a)-(y\pi/b)))dx$ over the interval $(0,+\infty) = +\infty$. The graph goes infinitely high at each interval before crashing down to 0, so over the positive domain we have $\infty\cdot\infty = \infty^2 = \infty$. Now we have the value of infinity that doesn't converge for our infinitely nested Q function. If we pass this into H, we get loop and in turn, Q will halt. This aligns with our view on how the recursive nature of Q itself loops on a null input, and halts on a looping input. Even if we recursively apply Q to itself infinitely many times, when it gets to the base case of null after infinite steps, it loops, and $(2/\pi)\cdot\arctan(\cot^2((2/\pi)\cdot\arctan(\cot^2(((x\pi/a)-(y\pi/b))))))$ halts at all points where the output = 0.

To reiterate, $\cot^2((x\pi/a)-(y\pi/b))$ at infinity doesn't converge to any value and oscillates, and as stated above, we can treat a diverging integral as a loop. Q_n is a function on the function $1/x$, as it alternates between $+\infty$ and 0 and it is the cyclical nested $1/x$. where $1/x$ is inputted into itself, infinitely many times. Essentially, it is the entire function of $1/x$ condensed and repeated between every natural number as normalized cosine intervals. In our rotating function of 1/0 and 0/1, $Q_n(\theta) = +\infty$ for $\theta = 2k\pi$ and $Q_n(\theta) = 0$ for $\theta = (2k+1)\pi$, where k is an integer. Entering infinity into $Q(x) = 1/x$ is 0. Entering infinity into $Q_n(\theta) = \cot^2((x\pi/a)-(y\pi/b))$ represents the infinite recursive nested call of $Q(x)$ into itself, which doesn't converge. Our starting point in this recursive nested call is x, and y represents the number of recursive steps. With x being a real number, we can start at fractional depths, and each inductive step in y represents an additional recursive nested call from out starting point x. This shifts the graph by π, oscillating between 0 and $+\infty$ and covering all values in between. This can also be graphed using spherical coordinates to give us $\rho = \cot^2(((\theta\pi)/a)-((\varphi\pi)/b))$, where the sheets extending to infinity illustrate where Q loops, and where the graph consolidates at the

origin represents where Q halts. $H(Q_{\theta,\varphi}) = (2/\pi)\cdot\arctan(\cot^2((\theta\pi/a)-(\varphi\pi/b)))$ is a unit sized spiraling manifold that oscillates into itself with a max radius of 1 and minimum radius of 0, illustrating the complete yet oscillatory nature of $H(Q_{\theta,\varphi})$.

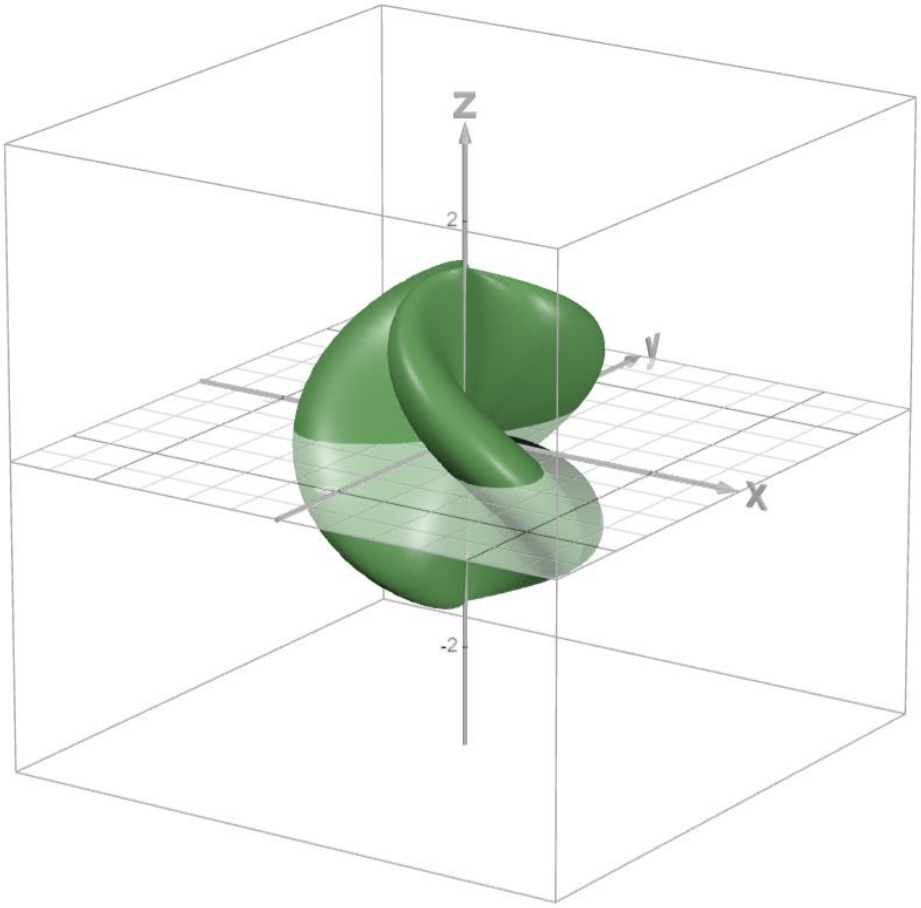

Figure 2

Mathematical Modeling of Complex continuous Valued Boolean Logic

If we let $x = 0$ and y, a, b =1, $Q_n = \cot^2((x\pi/a)-(y\pi/b))$ the frequency is double that of the former implementation and oscillates at intervals of π/2 instead of π. How might this be interpreted? Well, it's +∞ at 0 and π, and it is 0 at π/2 and 3π/2. It cycles through true and false states twice before returning to its initial state, once in each quadrant as opposed to once in each hemisphere when a, b = 2. What does this remind you of? It should remind you of the complex plane where we have the values 1, i, -1 and -i before returning to 1 [6]. For the purposes of this illustration, we will represent these states as i^0, i^1, i^2 and i^3 respectively, and where the exponent cycles through

complex logic states modulo 4. With our original Arctan model for H and $\cot^2$ model for Q_n, we only used the positive range of output to put into H. Now that we have abstracted the concepts of H and Q_n into their respective algebraic functions, we can follow the math to see what insights we might stumble upon. If we conceptualize complex valued logic, we would have T_{real}, $T_{imaginary}$, F_{real}, and $F_{imaginary}$. Within our original Arctan framework, we viewed 0 as false, and 1 as true. With our Q_n function, this corresponds to the points where $\cot^2(x)$ is 0 and $+\infty$ respectively. Within our H Arctan function, if we instead viewed 1 as true and -1 as false, 0 would represent something that is neither true nor false. Now, if we let x, y = 0 and a, b =1 we can transform Q_n into $\cot^2((x\pi/a)-(y\pi/b))\cdot\cos((x\pi/a)-(y\pi/b)) = \cot^2(\theta)\cdot\cos(\theta)$, where its $+\infty$ at 0, $-\infty$ at π, and 0 at $\pi/2$ and $3\pi/2$.

Let's take some inspiration from Fourier Analysis by representing this as the complex polar expression $r = |Q_n|\cdot e^{i\theta}$ and as we rotate around the unit circle, r touches $\pm\infty$ at ±1 and 0 at $\pm i$. From this complex logical perspective, the function starts out as really true at i^0 and gradually becomes more imaginarily true, and less really true, until it's only imaginarily true at i^1 and not really true, nor really false. Then we shift towards being really false but not imaginarily true at i^2. Then we shift towards being imaginarily false, neither really false nor really true at i^3, and finally we arrive back at the real truth at i^4. The real number line can be thought of as the basis of algebra, but complex analysis offers a more complete description of algebra as it is closed under the square root operator. It allows us to explain many problems in math and physics. Binary logic is seen as the basis of first order logic and hence mathematics, but perhaps visualizing complex logic can help expand our understanding of first order logic and first order logical paradoxes. Our tools for doing logical manipulation are limited, so we refer to our mathematical representations of Boolean functions when attempting to formalize complex logic. Revisiting the algebraic expressions in the previous section, we recall that the false function can be thought of as $h(x) = 0x$ where for all x, $h(x) = 0$. If we instead view 0 undefined, then false would be negative truth, or the complement. In binary, $1 - 1 = 0$, and $1 - 0 = 1$. In a ternary system however, -1 is false, 1 is true and 0 is undefined. When working with infinities, true would be $+\infty$ and false would be $-\infty$. We will now illustrate how by working with infinities, we can still model logical operations in ternary logic. To negate, we multiply by -1 to get $(-1)\cdot(+\infty) = -\infty$. Our conjunction would be $(-\infty)\cdot(+\infty) = -\infty$, and if we require positive infinities to be listed first as higher order infinities, we can have the union as $\infty^2 - \infty = +\infty$. With this in mind, we can model negation, conjunction and unions using $-\infty$ as false, and $+\infty$ as true. So now, using $\cot^2(\theta)\cdot\cos(\theta)$ to model our recursive Q, we alternate between true at $\theta = 0$, imaginarily true at $\pi/2$, false at π and imaginarily false at $3\pi/2$, and then back to the real truth and 2π. Mapping this in the complex plane, we can use all the

properties of complex analysis and tools in calculus to explore the properties of continuous complex valued logic. From this perspective, logical paradoxes are simply true or false on the imaginary axis. Something true on the imaginary axis is neither true nor false on the real axis, because the real number is 0. Similarly, something that is false on the imaginary axis is neither true nor false on the real axis because its real value is also 0. Likewise, something that is really true or false has no value on the imaginary axis.

This brings us to our formal introduction of Complex Boolean Logic, where we imagine ordinal arithmetic as trans dimensional, and the complex plane being the stage where we discuss trans dimensional reasoning. To illustrate what complex logic may look like, we must first understand that complex logical variables have two parts, similar to complex analysis, but they are conjunctions instead of sums. Imagine a perpendicular dimension where paradoxes are logical, and logic is paradoxical. In real logic, we speak truths, but in the paradoxical dimension we speak fallacies. In the real world, if we flip a coin and its heads, we can say “I am telling the truth, it’s heads”, and since it is heads, then this is real truth. If I say “I am lying, it’s heads”, then I have lied about lying because it’s actually heads, and lying about lying is true paradoxically speaking, so we say this is imaginarily true. If I say “I am telling the truth, it’s tails”, then I am lying about telling the truth, and I am lying that it’s tails, so we are really false. Lastly, if I were to say “I am lying, it’s tails”, then I have told the truth about lying because it is not tails, so I am imaginarily false. To determine the axis of logic, we first determine whether the statement is true or false, or if the statement is both true and false. The statement “I am telling the truth, it’s heads” Is true and true so it is true. The statement “I am telling the truth, it’s tails” Is false and false so it is false. For the imaginary axis, we have statements that are both true and false or false and true. A lie can be seen as the operator $\cdot(-1)$ and a truth can be seen as the operator $\cdot(1)$. Now for the statement “I am lying, it’s heads”, it is false and true, but it is false because we are lying about lying = $(-1)\cdot(-1) = 1$ so we are imaginarily true. The statement “I am lying. it’s tails”, is true and false, but it is false because we are telling the truth about lying = $(1)(\cdot-1) = -1$ so we are imaginarily false. If we were to model these truth states as points on the complex unit circle where 1 is true -1 is false, i is imaginarily true and $-i$ is imaginarily false, we would have the x and y pairs [0,1], [1,0], [0,-1] and [-1,0] modeled by i^0, i^1, i^2 and i^3 in the complex plane. Maintaining topological equivalence, we can apply the transformation matrix $T = [[\sqrt{2}\cdot\cos(\pi/4), -\sqrt{2}\cdot\sin(\pi/4)], [\sqrt{2}\cdot\sin(\pi/4), \sqrt{2}\cdot\cos(\pi/4)]]$ to get the points [1,1], [-1,1], [-1,-1] and [1,-1]. We can now model the Complex Boolean Unit Vectors L = [1,1], [-1,1], [-1,-1] and [1,-1] = [T,T], [F,T], [F,F] and [T,F]. The expression 1 = 0 is clearly false, but if we take points a = [1,0] and b = [0,1], if we rotate point a by $\pi/2$ radians, we see they become the same point. So, in measuring their real parts,

they aren't equal, but topologically they are. Points *a* and *b* are both 1 and 0 simultaneously, but understanding this requires an additional dimension to rotate through. At each recursive step in our Halting Problem, we rotate through this complex logical plane, where paradoxes collapse under the weight of infinity as we span into higher dimensional truth vector space.

Let us revisit this in the context of the Halting Problem. The issue people had with the Halting Problem was that if H predicted that Q(Q) would loop, then Q(Q) would halt seemingly contradicting H. If H predicted that Q(Q) would halt, then the inner H prediction would be wrong. We illustrated how H isn't actually wrong at all, but rather the answer given isn't one dimensional, the answer is 2 dimensional. If H was all knowing, and I am Q, I ask H "Do I halt or loop forever?", H would say "When given to yourself, you infinitely loop, and then when I tell you that, you will halt." The answer essentially says that Q will loop and halt, but instead of simultaneously, it says first Q will loop to ω, then Q will halt at ω + 1. The key is understanding the difference between $n + \omega$ and $\omega + n$, where no number of finite steps taken would ever cause Q to halt, but at ω + 1 steps it does halt. The only way to account for this is to understand that H and also Q because it has H as a subroutine, are not flat 1-dimensional functions, but rather higher-dimensional recursively nested functions that behave differently at different recursive depths. When a program gets stuck in a loop, it's because it has no awareness of anything outside of its current state, and no memory so to speak. But we said H can never loop, it always halts meaning that it is able to transcend this recognizing the pattern it is a part of, step outside of that, and deliver its answer. By definition, H is stepping into a higher dimension when it does this, akin to viewing ourselves from the third person, and we could describe what Q would do at any recursive step, in finite time. Take for example someone who makes the same mistakes and stays in the same position for their whole lives, they are stuck in an infinite loop. But someone who *realizes* what is happening, can simulate what would happen if they were to continue doing this for a lifetime and break the habit to achieve a different outcome [7]. Perhaps ADHD (attention-deficit hyperactive disorder) is not actually a 'disorder' at all [8], but rather it is the inability to harness our inner capacity to zoom in and out, focusing on and abstracting out of concepts. When applied correctly, we have great power as the connections between ideas lead to profound insights. When not applied correctly, we get stuck in the Halting Problem.

Let's further explore the idea of 'zooming out' in the context of Gödel's incompleteness theorems. In Gödel's first incompleteness theorem, he establishes that in any sufficiently powerful formal system that includes basic

arithmetic, there exist true statements that cannot be proven within the system itself. In his second incompleteness theorem, Gödel further explores these limitations by demonstrating that such a system cannot prove its own consistency. This means the system cannot, from within its own framework, verify that its axioms will never lead to a contradiction [9]. Essentially, Gödel hinted at the idea that within any formal system, we confront its limitations by 'zooming out,' so to speak, acknowledging that some truths or the system's consistency require a viewpoint outside the system's own confines to be established or understood. Expanding a formal system with more axioms may address certain limitations, but according to Gödel's Incompleteness Theorems, it inherently cannot capture all truths, leaving some statements perpetually unprovable within the system's framework. If we were to number all mathematical axioms describing this fully consistent system, we could map them all to the set of real numbers $\mathbb{R}$ giving us ω_1 axioms, an uncountably infinite set of axioms if you will. We know that any finite addition of axioms never gets to ω_1 as $1 + \omega_1 = \omega_1$. If somehow, we were to do that, and we made a contradiction, then that new element would be $\omega_1 + 1$, bringing us back to transfinite logic hence the paradox. Knowing this, we need to review paradoxical problems through the lens of transfinite logic. Take Bertrand Russell's paradox for example, "The set of all sets that do not contain themselves as members, is this set a member of itself?" [10]. The answer is no and yes. It's not for all sets up to ω_1 and is at $\omega_1 + 1$. After an uncountably infinite amount of time, we would have listed all the sets that do not contain themselves as members. Then after we have listed all these sets, set number $\omega_1 + 1$ would be this set itself. If we were to zoom into this set, we would see that it doesn't contain itself for all of its elements up to ω_1. Then at $\omega_1 + 1$, that nested set would include itself, and so on and so forth. Similarly, the 'Universal Set', which is the set of all sets including itself, is also infinitely recursive as we can continue to take powersets of subsets or of the whole set. Just as infinity is not a number but rather a function, an infinite uncountable set isn't some static entity but rather it is a function or the very process of an ever-expanding entity. In The omnipotence paradox, if an omnipotent being could lift ω kg, he could also make a stone that weighed $\omega + 1$ kg, step beyond infinity increasing his strength to $\omega + 1$ and still lift the stone [11]. This is strange for sure, not a paradox, but rather it simply enters the realm of transfinite or complex logic. Take the statement "This statement is False", it is self-referential, so it simply loops ad infinitum. We have "This statement is false" so we replace "This statement" with the statement and then we have ("This statement is false") is false ⇒ ("("This statement is false") is false") is false ⇒ (..(…)..is false). At our base case, we see the statement is false, and after our first recursive iteration, the statement becomes true because two negations give us truth. We observe a similar structure to that of the Halting Problem, where for any recursive depth n, "This statement is false" is false if $(n \bmod 2) \equiv 0$ and

true if (n mod 2) ≡ 1. Taking this up to ω, we know it will be false because ω is even. Then at ω + 1, the statement will be true. Continuing this into the domain of transfinite numbers, we see that the statement "This statement is false" exists between imaginary truth and imaginary falsity while never settling at true or false. Instead of viewing infinites as paradoxes, we instead should embrace them as the playgrounds of complex logic. The universe itself, as mysterious as it may be, still operates based on rules that are self-consistent and complete. The human mind also operates based on rules and is complete. In the context of Gödel's incompleteness theorems, expanding the axioms is akin to talking about an alternate universe by stepping outside of this universe, or introspecting into oneself by stepping outside of one's mind. Each of these two examples transcend the constraints of their systems so to speak, but they do so in such a way that their infinite nature becomes the standard by which we analyze and understand these frameworks. Rejecting ZFC for a second, let's take the set of all ideas. This is the set of everything we could possibly think of, including thinking about thinking so this set would be included within itself. But we can also think about thinking about thinking... and we can think about that infinite process. We could think about thinking about that infinite process. As abstract as it is, these are all thoughts you are having right now as you read this paper, so they must exist. The collection of everything you can think about is still a collection, and even though it is an uncountably infinitely expanding collection, this uncountable set still exists, nonetheless.

"L'Hôpital, and Bernoulli walk into a bar. Gödel looks around and says, 'This joke might be funny, but we can't tell because we're in it.'" If you laughed, it means you have a sense of humor, and this showcases the power of the transfinite computer we call a brain. Just like we take limits to infinity to avoid doing infinite addition in integration by collapsing infinity into a single point, hierarchical recursive programs like the human brain also work like this, as we simulate and step outside the constraints of formal systems. Returning to our compositional Q_n function, we can see by analyzing our $\cot^2(\theta)\cdot\cos(\theta)$ function in conjunction with our complex function $e^{i\theta}$, we are rotating from +1 to -1 and back passing $\pm i$ on the way. So, stepping from ω to ω+1 is akin to rotating into the complex plane. In a way, we can think of ω as all integer points on a line, and ω+1 would be the first point on the line above. With this view, we see that a square infinity like ω^2 is bigger than ω even though both are infinite, and getting to a bigger infinity requires going into a higher dimension i.e. the cartesian or complex plane. De Morgan's Law tells us how to distribute negation into logical expressions [2]. For Imaginary logic however, we introduce two new and very strange operations called iRotation and iNegation. They consist of turning conjuncts into implications and turning conjuncts into unions

respectively. When used in combination with negation, we can model a Boolean rotation around the complex logical unit circle. It gives us the four states T∧T, F∧T, F∧F, and T∧F.

```
def complex_logic(p, q, operator, operation):
    """
     Complex Boolean variables are comprised of two components, similar to complex numbers.
    When an expression is T∧T, it is True, and when it is F∧F, it is False.
     When an expression is F∧T or T∧F, it is imaginarily True or imaginarily False respectively.
    The imaginary unit 'i' in complex analysis is essentially both half positive and half negative.
    Conceptually, for i² to equal -1, it needs to be both 1 and -1 at the same time.
    True is to 1, as False is to -1 and iTrue is to +i as iFalse is to -i.
    Mapping these on the complex plane we have [e^(i0) = 1], [e^(iπ/2) = i], [e^(iπ) = -1], and [e^(i3π/2) = -i].
    We can start with the cartesian points [1,0], [0,1], [-1,0] and [0,-1] and maintain topological equivalence after applying the transformation matrix
    T = [[√2·cos(π/4), -√2·sin(π/4)], [√2·sin(π/4), √2·cos(π/4)]] to get the cartesian points [1,1], [-1,1], [-1,-1] and [1,-1].
    We can now model the Complex Boolean Unit Vectors L = [1,1], [-1,1], [-1,-1] and [1,-1] = [T,T], [F,T], [F,F] and [T,F].
    """
    # i?(T∧T) ≡ i(T⇒T) ≡ i(F∨T) ≡ F∧T Real True to Imaginary True
    # ¬?(F∧T) ≡ ¬(F⇒T) ≡ ¬(T∨T) ≡ F∧F Imaginary True to Real False
    # i?(F∧F) ≡ i(F⇒F) ≡ i(T∨F) ≡ T∧F Real False to Imaginary False
    # ¬?(T∧F) ≡ ¬(T⇒F) ≡ ¬(F∨F) ≡ T∧T Imaginary False to Real True

    # Returns the operands and the operator
    # Introduces iNegation and iRotation to manipulate complex logical variables
    result = None
    if operation == 'inegation': # i negates only the operator
        result_p = p
        result_q = q
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    elif operation == 'irotation': # ? negates p and the operator
        result_p = not p
        result_q = q
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    elif operation == 'negation': # ¬ negates p, q and the operator
        result_p = not p
        result_q = not q
```

```
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    return result
```

It's important to imagine this framework as adding order and logic to something seemingly inherently illogical. Just as the $\sqrt{(-1)}$ is seemingly illogical, but it is in fact just counter intuitive, so too is complex logic. For mathematical expressions involving the hyperreal number line, one might argue that complex analysis and hypercomplex numbers already take this imaginary logic into account with the introduction of i, and i, j, k respectively [12]. If we negate the imaginary component of any equation, it doesn't affect the real part, and negating the real part doesn't affect the imaginary part. Notice how at both i^1 and i^3, the real value is 0, which rests between negative and positive infinity, or *really true*, and *really false*. Here, we are *imaginarily true* or *imaginarily false*, as they are neither *really true* nor *really false*, but both at the same time, just as we would expect with a logical paradox.

Let's explore two candidate functions for H on our new compositional Q function. First, let's define H to be the sigmoid function (σ), then we have $\sigma(\cot^2((x\pi/a)-(y\pi/b))\cdot\cos((x\pi/a)-(y\pi/b)))$ with a starting point of x, $y = 0$, and a, $b = 1$. H will return 1 for loop, and 0 for halt, passing through 0.5 at each interval step symbolizing the paradoxical nature if we try to take a value in between true and false. This new complex Q function that alternates between $+\infty$ and $-\infty$ has interesting properties. At $x = 0.5$ in Q_n, $H(Q_n) = 0.5$, and at each discrete recursive step it remains 0.5. Here, 0.5 represents neither true nor false, a paradox if you will. In between true 'loop' and false 'halt', we have imaginary truths and imaginary falsities. This is why if we start at 0.5 and iterate by 1 step each time, we stay at 0.5, because we alternate between imaginary truth and imaginary falsity. Both are neither really true, nor really false, so they are at 0 on the real axis. If we were to start at 0 however, which is our base state using the sigmoid function, we are at $+\infty$, and $\sigma(+\infty) = 1$. Then at 1 in our $\cot^2((x\pi/a)-(y\pi/b))\cdot\cos((x\pi/a)-(y\pi/b))$ gives us $-\infty$, and $\sigma(-\infty) = 0$. Plugging 0 back into Q_n gives us $+\infty$, returning us back at our base case. This illustrates a complete cycle between true at 0 and false at π, passing through imaginarily true at $\pi/2$ on the way there, $3\pi/2$ at imaginarily false on the way back, and finally arriving at back at true at 2π. Here, our $H(Q_n)$ function is true at all even integer increments from our base case, and false for all odd integer increments from our base case. The Halting Problem essentially tries to collapse how Q behaves at even and odd integer recursive steps into one paradox.

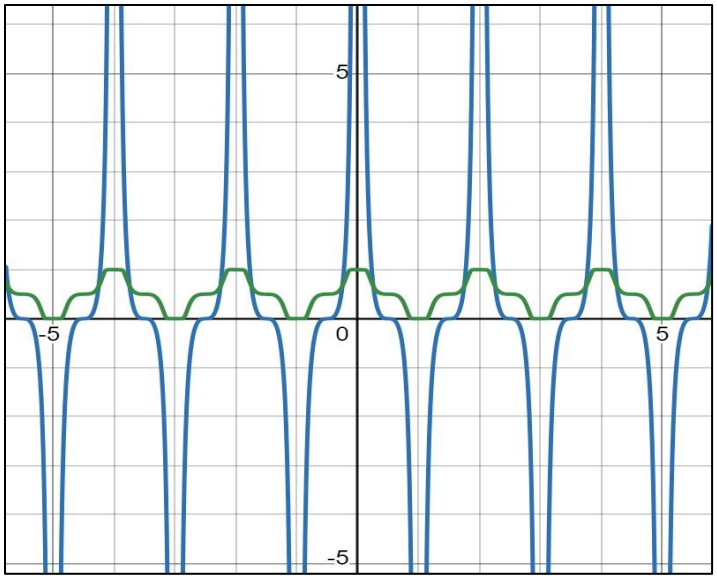

Figure 3

Ideally, we could use our original Arctan function instead of sigmoid for H, but for our Q function $\cot^2(((x\pi)/a) - ((y\pi)/b))\cdot\cos(((x\pi)/a) - ((y\pi)/b))$, we can't start at 0 anymore because our base case is -1 corresponding to false at -∞. With a starting phase of $y = -1$, and a, $b = 2$, if we start at $x = 0$, that's equivalent to starting at i^1, and we will stay at a logical paradox at each cycle between imaginary truth and imaginary falsity. So, for x, instead of starting at 0, we plug in -1, and we let $x = -1$. Then at each recursive step, we alternate back and forth between real truth at ∞, and real false at -∞.

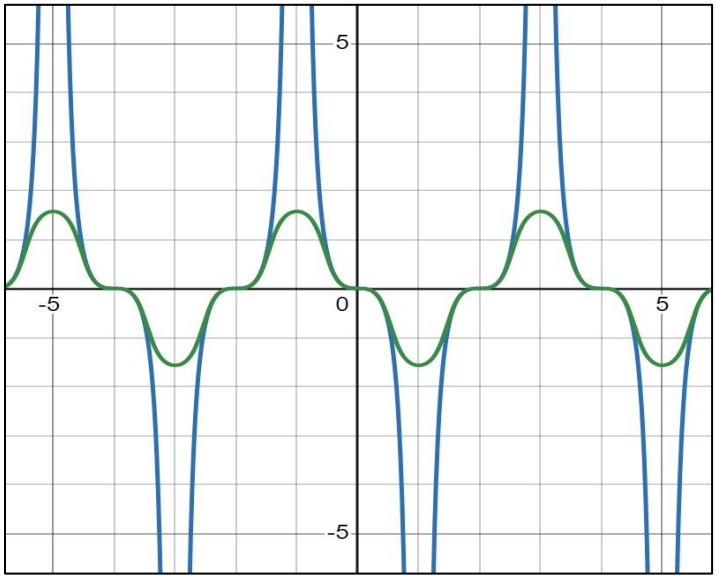

Figure 4

Here's the code to mess around with.

```
from math import sin, cos, tan, atan, exp, pi
# Project Name: "Deciding the Undecidable"
# Author: Kyle Hinds
# Date: April 20th, 2024
# Terms of Use:
# This software is provided 'as-is', without warranty.
# If you find this interesting or useful in any way, feel free to make a $BTC donation to 'bc1pfxhfmalgpl3v2nq9nx37w4320635dr5x587rh74klzew3gyqv47qtnpaev'.
# Donations are appreciated and will contribute towards future research in computer science.
# Code contributions are also welcome and fall under the same terms.
# Not a legal contract.
"""
# Here, we offer a solution to Alan Turing's Halting Problem (Turing, 1936), properly defining and algebraically modeling the Turing Machines H and Q.
# In the paper "Deciding The Undecidable" (Hinds, 2024), we demonstrated how and why Q must have an input.
# Turing machines are finite computational models that process finite strings of input, executing operations based on a finite set of rules.
# In the Halting Problem, H(P,I) tells us if P(I) halts or loops forever, while Q(P) loops forever if H(P,P) returns 'halt', or halts if H(P,P) returns 'loop'.
# We propose that Q(null) loops because H(null,null) = 'halt' on no input, and with Q behaving opposite of H, Q(null) loops forever.
# The hypothetical paradox of H(Q,Q) predicting both looping and halting simultaneously fails to acknowledge the base case for Q.
# If we let P = Q and I = Q, the inner Q will have no input, giving us H(Q,Q(null)).
# Evaluating this we get H(Q,loop) = H(Q(loop)) = 'halt'.
# With a simple proof by induction on Q: Base: Q(null) loops, Step: Q(Q)='halt' and Next: Q(Q(Q)) loops returning to the base case.
# In general, this recursive Q function loops if for the number of iterations, n, satisfies (n mod 2) ≡ 0 (even).
# Conversely, it halts for an odd number of iterations, where (n mod 2) ≡ 1 (odd).
# Failing to require a base case leads to the infinite input string Q(Q(...(Q))) = Q∞ = Qω.
# In set theory, ∞ is represented by the ordinal ω with cardinality ℵ₀, and ω is even if ∃β such that 2β = α.
# With ordinal arithmetic, 2·ω = ω, but ω + 1 > ω, so ω can be seen as an infinite series of pairs, making it even.
# This satisfies the even definition with β as ω itself, showcasing its transfinite evenness.
# The ordinal ω demonstrates that even-odd concepts extend to transfinite ordinals, with ω + 1 being akin to Q(Qω) = Qodd.
```

```
# With respect to the infinite input string Q(Q(...(Q))), we must distinguish between
Q as a function and Q as an input.
# Q as a function (the outermost Q) behaves the opposite of H(P,P).
# We know H is a subroutine of Q, so when it passes itself into itself, we need to
evaluate what H(Q,Q) does inside Q.
# Inside Q, H(Q,Q) becomes H(Qω) because we have a self-referential loop, which H
evaluates as such, and then in response, the outer Q halts.
# When Q(Q) is called, H returns the state of the program at ω steps which is a loop.
# However, at ω + 1 steps, Q(Q) which becomes Q(Qω) halts, and at ω + 2 steps it loops,
and so on and so forth.
# H tells us if a program(input) halts or loops forever, and Q(Q) does indeed loop
forever, but it halts at forever + 1 steps.
# If we want to evaluate Q(Q(Q)) = Qω+2, we need to evaluate H(Q,Q(Q)), which when
called from H within Q becomes H(Q,Qω).
# With H(Q,Qω), we have Q as the function and Qω as the input.
# Evaluating Qω would take infinite time so H(Qω) returns 'loop'.
# Evaluating H(Q,H(Qω)) = H(Q(Qω)) = H(Qω+1) = H(Qodd) = 'halt'.
# Recursively, H(Q,H(Q(Qω))) = H(Q(Q(Qω))) = H(Qω+2) = H(Qeven) = 'loop'.
# To be rigorous, we can expand out the inner H to get:
# Q(Qω)
# = H(Q,H(Qω,Qω))
# = H(Q,H(Qω(Qω)))
# = H(Q,Qω·2)
# = H(Q(Qω·2))
# = H(Qω·2+1)
# = H(Qodd) = 'halt'.
# Q(Q(Qω))
# = H(Q,H(Q(Qω),Q(Qω)))
# = H(Q,H(Q(Qω(Q(Qω)))))
# = H(Q,H(Q(Q1+ω(Qω))))
# = H(Q,H(Q(Qω(Qω))))
# = H(Q,H(Qω+1(Qω)))
# = H(Q(Qω·2+1))
# = H(Qω·2+2)
# = H(Qeven) = 'loop'.
# and going further, Q(Q(Q(Qω)))
# = H(Q,H(Q(Q(Qω)),Q(Q(Qω))))
# = H(Q,H(Q(Q(Qω(Q(Q(Qω)))))))
# = H(Q,H(Qω + (1 + (1 + (Qω + (1 + (1)))))))
# = H(Q,H(Qω + (1 + (1 + (Qω + (2))))))
# = H(Q,H(Qω + (1 + (1 + (Qω+2)))))
# = H(Q,H(Qω + (1 + (1 + Qω+2))))
# = H(Q,H(Qω + (1 + (Qω+2))))
# = H(Q,H(Qω + (1 + Qω+2)))
# = H(Q,H(Qω + (Qω+2)))
```

```
# = H(Q,H(Qω·2+2))
# = H(Q(Qω·2+2))
# = H(Qω·2+3)
# = H(Qodd) = 'halt'.
# The expressions above simplify to (Qω·2 + n) where n is the recursive depth of the function parameter in H.
# The inner recursive calls of the input parameter in H are absorbed into the outer Qω of the function parameter in H because 1 + ω = ω.
# With infinite recursion, we arrive at the same result because H(Q(H(Qω(...(Qω(Qω(...(Qω)))))))) = H(Q(Qω·2ⁿ)) = H(Qω·2ⁿ + 1) = H(Qodd) = 'halt'.
# With this, we see Qω = Qeven = Q(0) = Q(halt) = Q(null) loops forever, and Qω+1 = Qodd = Q(1) = Q(loop) = Q(Qω) halts completing our proof.
# Algebraically, H(x) can be modeled using the sigmoid and normalized arctan functions.
# These functions return finite values between 0 and 1, or -1 and 1 for values between -∞ and +∞.
# If we allow all program(input) strings that halt to be mapped to some hyperreal number, and all program(input) strings that loop to be mapped to +∞,
# Q(x) can be thought of as the positive domain of 1/x, where on +∞ (looping program(input)), lim x → +∞ (Q(x)) returns 0,
# on 0 (halting program(input)), lim x → 0⁺ (Q(x)) returns +∞, and in general ∀ x ∈ *ℝ, Q(x) ∈ *ℝ.
# Qn is the recursive call of Q(x) into itself = 1/(1/(1/...(1/x))), where n is the recursive depth.
# The Qn function's behavior is in direct contrast to the output of Hn-1, while Hn decides the outcome of Qn.
# We can use tan²(θ) and cot²(θ) or tan²(θ)·sin(θ) and cot²(θ)·cos(θ) to model the cyclical nature of Qn(θ).
# The functions tan²(θ) and cot²(θ) alternate between 1/x and x in the positive domain, ranging between 0 and +∞.
# The functions tan²(θ)·sin(θ) and cot²(θ)·cos(θ) alternate between 1/x and x in the positive and negative domains, ranging between -∞ and +∞.
# Together, these functions model the cyclical recursive call of Q into itself.
# We appreciate the fact that Q∞ = Qω = Qeven = Q(0) = Q(null) = Qn(0π) = Qn(2kπ) marking the beginning and end of a complete cycle.
# We can leverage limits, derivatives, and integrals on these trigonometric functions to explore the fractional and infinite recursion of Qn.
# Whether Q halts or loops when given its own code as input depends on the recursive depth of Qn, with |complex logical states| = 𝒫(ℵ₀) = 𝔠.
# This will be illustrated using the algebraic expressions defined below.
"""

# Cleaning up decimal points.
def round_to_limits(value, decimals=2):
    #Round very small values to 0 and very large values to ±∞
    if abs(value) < 1e-10:  # Threshold for considering value as 0
```

```
        return 0.0
    elif value > 1e10:  # Threshold for considering value as +∞
        return float('inf')
    elif value < -1e10:  # Threshold for considering value as -∞
        return float('-inf')
    else:
        return round(value, decimals)

# Example of a function that always halts returning 0.
def halt(x = 0, y = 0, a=1, b=1):
    # Returns 0·((x·a) + (y·b))
    result = 0*((x*a) + (y*b))
    return round_to_limits(result)

# Example of a function that always loops and returns ±∞.
def loop(x=0, y=0, a=1, b=1):
    # Returns +∞ for non-negative input and -∞ for negative input
    # Returns ((x·a) + (y·b))/0
    if ((x*a) + (y*b)) > 0:
        return float('inf')  # +∞ for non-negative input
    elif ((x*a) + (y*b))  < 0:
        return float('-inf')  # -∞ for negative input
    else:
        return 1 # l'Hopitals rule: lim x → 0 (sin(x)/x) = lim x → 0 (cos(x)/1)  = 1

# Definition of the function Q.
# By definition, sometimes Q halts, and sometimes it loops.
# On infinite input, it halts and returns the finite output '0'.
# On the finite input '0', it loops forever and returns infinite output.
def q_inverse(x=0, y=0, a=1, b=1):
    # Returns +∞ at 0 and 0 at +∞
    # Returns 1 / ((x·a) + (y·b))
    if (x + y) == 0:
        return float('inf')  # Return +∞ for zero input
    elif float('inf') in [x, y] or float('-inf') in [x, y]:
        # Handling division by ±∞, which results in 0
        return 0
    else:
        result = 1 / ((x*a) + (y*b))
        return round_to_limits(result)

# Definition of the function H.
# Functions for H, where it returns a yes or no answer that itself is finite, and can
handle infinities as input.
def h_arctan(x=0, y=0, a=1, b=1): # Returns a number between -1 and 1
```

```
    # Returns normalized arctan(((xπ)/a) - ((yπ)/b))·(2/π) = arctan(θ)·(2/π)
    result = atan(((x)/a) - ((y)/b)) * (2 / pi)
    return round_to_limits(result)

def h_sigmoid(x=0, y=0, a=1, b=1): # Returns a number between 0 and 1
    # Returns 1/(1 + e^(-(((xπ)/a) - ((yπ)/b)))) = 1/(1 + e^(-θ))
    result = 1 / (1 + exp(-((x*a) + (y*b))))
    return round_to_limits(result)

# Functions for Qn that treat +∞ as True (loop) and 0 as halt (False).
# Here x is our starting point, and y is the integer or fractional number of recursive
iterations.
# Allowing x and y to be hyperreal numbers facilitates the exploration of fractional
and infinite recursion.
# Qn tan²(θ) function
def qn_tan2(x=0, y=0, a=2, b=2): # starts at 0
    # Returns tan²(((xπ)/a) - ((yπ)/b)) = tan²(θ) = recursive step # of q_inverse
    argument = ((x * pi) / a) - ((y * pi) / b)
    # Check if cos(argument) == 0, which implies cot(argument) is 0 and tan²(θ) is
undefined
    if cos(argument) == 0:
        return float('inf')  # Return +∞ for tan²(θ) when cot(θ) is 0
    result = tan(argument)**2
    return round_to_limits(result)

# Qn cot²(θ) function
def qn_cot2(x=0, y=0, a=2, b=2): # starts at ∞
    # Returns cot²(((xπ)/a) - ((yπ)/b)) = cot²(θ) = recursive step # of q_inverse
    argument = ((x * pi) / a) - ((y * pi) / b)
    # Check if sin(argument) == 0, which implies tan(argument) is 0 and cot²(θ) is
undefined
    if sin(argument) == 0:
        return float('inf')  # Return +∞ for cot²(θ) when tan(θ) is 0
    result = 1 / tan(argument)**2
    return round_to_limits(result)

# Functions for Qn that treat +∞ as True (loop) and -∞ as halt (False).
# Here x is our starting point, and y is the number of recursive iterations.
# Allowing x and y to be hyperreal numbers facilitates the exploration of fractional
and infinite recursion.
def qn_tan2_sin(x=0, y=0, a=1, b=1): # starts at 0
    # Returns tan²(((xπ)/a)-((yπ)/b))·sin(((xπ)/a)-((yπ)/b)) = tan²(θ)·sin(θ)
    # Calculate the argument
    argument = ((x*pi)/a) - ((y*pi)/b)
    # Check for conditions leading to ∞ directly to avoid division by zero
```

```
    if cos(argument) == 0:
        return float('inf')  # This handles the division by zero in tan²(θ) calculation
    # Calculate the expression value
    result = ((tan(argument)**2) * sin(argument))
    # Round the result to handle very small or very large values
    rounded_result = round_to_limits(result)
    return rounded_result

def qn_cot2_cos(x=0, y=0, a=1, b=1): # starts at ∞
    # Returns cot²(((xπ)/a)-((yπ)/b))·cos(((xπ)/a)-((yπ)/b)) = cot²(θ)·cos(θ)
    # Calculate the argument
    argument = ((x*pi)/a) - ((y*pi)/b)
    # Check for conditions leading to ∞ directly to avoid division by zero
    if sin(argument) == 0:
        return float('inf')  # This handles the division by zero in cot²(θ) calculation
    # Calculate the expression value
    result = ((1 / tan(argument))**2) * cos(argument)
    # Round the result to handle very small or very large values
    rounded_result = round_to_limits(result)
    return rounded_result

# Testing defs with preset constants for Qn.
# These initialize the starting point, phase shift and frequency.
# Trigonometric Qn Functions with (0 ≤ z ≤ 1)
def qn_tan2_arctan_const(x=0, y=-1, a=2, b=2):
    return qn_tan2(x, y, a, b)

def qn_cot2_arctan_const(x=0, y=0, a=2, b=2):
    return qn_cot2(x, y, a, b)

# Compositional Trigonometric Qn Functions with (0 ≤ z ≤ 1)
def qn_tan2_sin_sigmoid_const(x=0, y=-.5, a=1, b=1):
    return qn_tan2_sin(x, y, a, b)

def qn_cot2_cos_sigmoid_const(x=0, y=0, a=1, b=1):
    return qn_cot2_cos(x, y, a, b)

# Compositional Trigonometric Qn Functions with (-1 ≤ z ≤ 1)
def qn_tan2_sin_arctan_const(x=0, y=-2, a=2, b=2):
    return qn_tan2_sin(x, y, a, b)

def qn_cot2_cos_arctan_const(x=0, y=-1, a=2, b=2):
    return qn_cot2_cos(x, y, a, b)

# The Halting Machine H(Qn).
```

```
def halting_machine(x_values = [-1, 0, 1], depth = 3):
    print(f"The Halting Machine H(Qn)")
    """
    The halting_machine function explores recursive applications of H mappings (sigmoid and arctan) to three models of Qn functions.

    Our recursive Qn models operate with π radian rotations, requiring two calls to complete a full 2π cycle.
    Defining 1/x as Q, Qn effectively compresses the entire domain of 1/x, spanning the intervals (-∞, 0) and (0, ∞), into π/2 intervals.
    Through phase shifting, our trigonometric models tan²(θ) and cot²(θ) are aligned to equivalently map to the same points.
    The positive domain of the function of 1/x is compressed and mapped to the interval (2kπ, (2k + 1)π).
    The reflection about the x-axis of the negative domain of 1/x is compressed and mapped to the interval ((2k + 1)π, (2k + 2)π).
    Similarly, with phase shifting in the compositional models tan²(θ)sin(θ) and cot²(θ)cos(θ), the domain of 1/x is segmented across four intervals:
    The interval (2kπ, (2k + 1/2)π) corresponds to the positive domain of 1/x.
    The interval ((2k + 1/2)π, (2k + 1)π) corresponds to the negative domain of 1/x.
    The interval ((2k + 1)π, (2k + 3/2)π) corresponds to the reflection about the x-axis of the positive domain of 1/x.
    The interval ((2k + 3/2)π, (2k + 2)π) corresponds to the reflection about the x-axis of the negative domain of 1/x.
    Each segmentation corresponds to the positive or negative domains of 1/x and -1/x respectively, enabling a cyclical mapping of Q.

    In all models, the complete cycle encompasses a full 2π rotation, albeit scaled differently to match each model's input and output constraints.
    Each discrete recursive function call represents a rotation of π effectively spanning the entire distance of the unit circle after two recursive calls.
    Incrementing θ by additions of π/2 radians or any other angle α allows us to systematically explore its logical states.
    This mechanism allows for the investigation of complex logical dynamics by simulating continuous logical rotations.
    A complete rotation of 2π in the complex logical plane is akin to double negation '¬¬'.
    Our input argument consists of the variables x and y with constants a and b.
    The 'x' parameter represents the initial state, serving as the starting point or recursive depth of Qn.
    The 'y' parameter adjusts the phase shift responsible for the progression through logical recursive states.
    Constants 'a' and 'b' act as scaling factors, adjusting the frequency of Qn.
    The outcome of the Qn function at any given recursive step is given by Qn(θ), where θ(x,y,a,b) is the argument.
```

```
    1. ** H Arctan Trigonometric Qn Functions (0 ≤ z ≤ 1):**
    - The models, qn_tan2(θ) = tan²(xπ/a - yπ/b) and qn_cot2(θ) = cot²(xπ/a - yπ/b),
showcase how arctan transitions between 0 and 1 on input between 0 and ∞.
    - The Arctan function here has a domain over the interval (0, ∞) and a range of (0
≤ z ≤ 1).
    - Recursively applying H(Qn) creates a binary loop between 0 and 1.
    - The midpoint of x = 0.5 for Qn acts as a marker for a logical paradox consisting
of our imaginary states, indicative of a system trapped between True and False.

    2. ** H Sigmoid Compositional Trigonometric Qn Functions (0 ≤ z ≤ 1):**
    - The Sigmoid Compositional models are defined as qn_tan2_sin(θ) = tan²(xπ/a -
yπ/b)·sin(xπ/a - yπ/b) and qn_cot2_cos(θ) = cot²(xπ/a - yπ/b)·cos(xπ/a - yπ/b).
    - The sigmoid function here has a domain over the interval (-∞, ∞) and a range of
(0 ≤ z ≤ 1).
    - Recursively applying H(Qn) creates a binary loop between 0 and 1.
    - The midpoint of x = 0.5 for Qn acts as a marker for a logical paradox consisting
of our imaginary states, indicative of a system trapped between True and False.

    3. ** H Arctan Compositional Trigonometric Qn Functions (-1 ≤ z ≤ 1):**
    - The Arctan Compositional models are also defined as qn_tan2_sin(θ) = tan²(xπ/a
- yπ/b)·sin(xπ/a - yπ/b) and qn_cot2_cos(θ) = cot²(xπ/a - yπ/b)·cos(xπ/a - yπ/b).
    - The Arctan function here has a domain over the interval (-∞, ∞) and a range of
(-1 ≤ z ≤ 1).
    - Recursively applying H(Qn) creates a binary loop between -1 and 1.
    - The midpoint of x = 0 for Qn acts as a marker for a logical paradox consisting
of our imaginary states, indicative of a system trapped between True and False.

    It is important to remember how our Qn models collapse and condense the domain of
1/x into π/2 intervals, so Qn(θ) maps an angle to some value over the entire domain of
1/x.
    In the H Arctan model, we start at x = 0 (symbolizing a halt or False state), Qn
approaches +∞ which aligns with Q's base state of looping.
    When Qn is recursively applied to itself, H deciphers +∞ as 1 (indicating looping),
and plugging 1 back into Q gives us 0,
    Feeding 0 back into H gives us 0 (signifying halting).
    When reintroduced back into Qn with x = 0, the function reaches ∞, which H decides
as 1 (signifying looping) thus completing the cycle.
    This cycle between 0 and 1 in the positive domain of H Arctan illustrates the
model's binary oscillation mapping the entire hyperreal number line onto the interval
(0, 1).
    The midpoint of x = 0.5 for Qn in this model acts as the state of logical paradox,
where for nπ rotations the output of H(Qn) remains 0.5.
```

In the H Sigmoid Compositional model, we start at x = 0 (symbolizing a halt or False state) and Qn approaches +∞ which aligns with Q's base state of looping.
When Qn is recursively applied to itself, H deciphers +∞ as 1 (indicating looping), and plugging 1 back into Q gives -∞.
Feeding -∞ back into H gives us 0 (signifying halting).
When reintroduced back into Qn with x = 0, the function reaches ∞, which H decides as 1 (signifying looping) thus completing the cycle.
This cycle between 0 and 1 illustrates the H Sigmoid model's binary oscillation mapping the entire hyperreal number line onto the interval (0, 1).
The midpoint of x = 0.5 for Qn in this model acts as the state of logical paradox, where for nπ rotations the output of H(Qn) remains 0.5.

You'll recall that Q is such that it loops on a halt input and halts on a loop input.
In the H Arctan Compositional model, we reimagine Q as a function that positively loops outputting +∞ on H's halting output '-1'.
Here, our halting state is modeled by the negative looping output -∞ on H's looping output '+1'.
Since our Qn has a range between ±∞ and H = (2/π)·arctan(x) has a range between ±1, we must alternate between +∞ at -1 and -∞ at 1.
We phase shift left by π/2 because our base 'halt' state is -1 at -∞, and periods 0 through 3 correspond to the angles 0, π/2, π, and 3π/2 respectively.
By phase shifting left by π/2, we effectively start at -1 where θ(x=0, y=-1, a=2, b=2) = θ(xπ/a - yπ/b) = -π/2.
Now at x = -1 (Halting output from H), Qn approaches +∞ which aligns with Q's base state of looping.
H deciphers +∞ as 1, and recursively reintroducing x = 1 into Qn we have θ = π/2.
At θ = π/2, Qn reaches -∞, and feeding -∞ back into H gives us -1 (signifying halting).
When reintroduced back into Qn with x = -1, the function reaches +∞, which H decides as 1 (signifying looping) thus completing the cycle.
This cycle between -1 and 1 illustrates the Arctan model's binary oscillation mapping the entire hyperreal number line onto the interval (-1, 1).
The midpoint of x = 0 for Qn in this model acts as the state of a logical paradox, where for nπ rotations the output of H(Qn) remains 0.

For all models, if we let r = |Qn|, we can graph |Qn|·(e^(iθ)) in the complex plane and observe its behavior as e^(iθ) makes its way around the unit circle of complex logic.
Using discrete recursion for Qn with each call being akin to θ+=π, if we start in a paradoxical state, such as x = 0.5 for Arctan and Sigmoid or x = 0 for Arctan Compositional models, we stay bound to the imaginary axis.
Alternating between ±i, subsequent incrementations θ+=π maintain this anchoring, keeping Qn's hyperreal output invariably at 0.

```
    This value of 0 in between ±1 (True and False) symbolizes an 'undefined' or
paradoxical state within the complex logic framework.
    In the complex logical plane, Qn(θ) navigates through the following critical
logical states during fractional recursion:
   1 and -1 which correspond to the 'real' logical states True and False,
   while ±i denote the 'imaginary' logical states Imaginary True and Imaginary False
respectively.

   # For the function Qn(θ) = tan²(θ)·sin(θ):
   # The derivative d/dθ(tan²(θ)·sin(θ)) is given by:
   # d/dθ(tan²(θ)·sin(θ)) = 2tan(θ)·(1+tan²(θ))·sin(θ) + tan²(θ)·cos(θ)
   # The integral ∫tan²(θ)·sin(θ)dθ is given by:
   # ∫tan²(θ)·sin(θ)dθ = cos(θ) + sec(θ) + C

   # For the function Qn(θ) = cot²(θ)·cos(θ):
   # The derivative d/dθ(cot²(θ)·cos(θ)) is given by:
   # d/dθ(cot²(θ)·cos(θ)) = -2cot(θ)·(1+cot²(θ))·cos(θ) - cot²(θ)·sin(θ)
   # The integral ∫cot²(θ)·cos(θ)dθ is given by:
   # ∫cot²(θ)·cos(θ)dθ = -sin(θ) - csc(θ) + C

   Graphing Qn(θ) as z = qn_cot2_cos(x=0,y=0,a=2,b=2) with z being the height above
the unit circle, we see significant changes in z at multiples of π/2.
   Approaching angles 0 and π, Qn(θ) tends toward ±∞ at the points ±1 on the complex
unit circle, reflecting a logical 'loop' or 'halt'.
   At π/2 and 3π/2, z tends to 0 at the points ±i reflecting imaginary looping, or
imaginary halting respectively.
   The derivative d/dθ(Qn(θ)) gives us the rate of change with respect to θ (Δθ→0 and
ε→0) as we approach the recursive depths at the critical points (1, i, -1, -i).
    The integral ∫Qn(θ)dθ over some periodic interval is the summation of complex
logical states which averages out to 0.
    At 1 and -1 (True/False states), d/dθ(Qn(θ)) experiences sharp spikes to ±∞,
signaling swift transitions to True and False states.
   At i and -i (iTrue/iFalse states) our paradoxical states, the function's rate of
change levels off to 0, mirroring the logical halt in a paradox.
    If we let ρ = Qn(Θ/α, ϕ/β, ..., Ω/ζ), we can conceptualize higher-dimensional
cyclical logic.
    Here, ρ represents the magnitude of the "truth vector" in higher-dimensional
spherical space, defined as Qn evaluated at scaled angles (Θ/α, ϕ/β, ..., Ω/ζ).
   Variables Θ, ϕ, ..., Ω are the "angles of truth" in this n-dimensional logic space,
and constants α, β, ..., ζ serve as scalars for these angles.
   For some variable λ we can take ∂Qn/∂λ for the partial derivative, and the gradient
truth vector is ∇Qn = ⟨∂Qn/∂Θ, ∂Qn/∂ϕ, ..., ∂Qn/∂Ω⟩.
   We may also integrate ∫...∫(Qn)dΘ...dΩ over an n-dimensional domain in this logic
vector space to explore truth densities.
```

```
    This formulation allows us to extend the concept of cyclical or periodic logic to
hypercomplex numbers and into higher-dimensional spaces.
    """

    # Trigonometric models for H and Q
    functions = [
        (h_sigmoid, qn_tan2_sin_sigmoid_const, "H Sigmoid with Qn = tan²(θ)·sin(θ)"),
        (h_sigmoid, qn_cot2_cos_sigmoid_const, "H Sigmoid with Qn = cot²(θ)·cos(θ)"),
        (h_arctan, qn_tan2_arctan_const, "H Arctan with Qn = tan²(θ)"),
        (h_arctan, qn_cot2_arctan_const, "H Arctan with Qn = cot²(θ)"),
        (h_arctan, qn_tan2_sin_arctan_const, "H Arctan with Qn = tan²(θ)·sin(θ)"),
        (h_arctan, qn_cot2_cos_arctan_const, "H Arctan with Qn = cot²(θ)·cos(θ)")
    ]
    # Trigonometric recursion
    for x in x_values:
        print(f"\nEvaluating functions for x = {x}:")
        for h_func, qn_func, description in functions:
            print(f"\n{description}:")
            current_x = x
            for i in range(1, depth + 1):
                result = h_func(qn_func(current_x))
                # Print the function names and the current value of x being evaluated
                                                print(f"Iteration     {i}     with
{h_func.__name__}({qn_func.__name__}({current_x})): Result = {result}")
                current_x = result
        print("")

# Complex Logic.
def complex_logic(p, q, operator, operation):
    """
     Complex Boolean variables are comprised of two components, similar to complex
numbers.
    When an expression is T∧T, it is True, and when it is F∧F, it is False.
     When an expression is F∧T or T∧F, it is imaginarily True or imaginarily False
respectively.
    The imaginary unit 'i' in complex analysis is essentially both half positive and
half negative.
    Conceptually, for i² to equal -1, it needs to be both 1 and -1 at the same time.
    True is to 1, as False is to -1 and iTrue is to +i as iFalse is to -i.
    Mapping these on the complex plane we have [e^(i0) = 1], [e^(iπ/2) = i], [e^(iπ)
= -1], and [e^(i3π/2) = -i].
    We can start with the cartesian points [1,0], [0,1], [-1,0] and [0,-1] and maintain
topological equivalence after applying the transformation matrix
    T = [[√2·cos(π/4), -√2·sin(π/4)], [√2·sin(π/4), √2·cos(π/4)]] to get the cartesian
points [1,1], [-1,1], [-1,-1] and [1,-1].
```

```
    We can now model the Complex Boolean Unit Vectors L = [1,1], [-1,1], [-1,-1] and
[1,-1] = [T,T], [F,T], [F,F] and [T,F].
    """
    # i?(T∧T) ≡ i(T⇒T) ≡ i(F∨T) ≡ F∧T Real True to Imaginary True
    # ¬?(F∧T) ≡ ¬(F⇒T) ≡ ¬(T∨T) ≡ F∧F Imaginary True to Real False
    # i?(F∧F) ≡ i(F⇒F) ≡ i(T∨F) ≡ T∧F Real False to Imaginary False
    # ¬?(T∧F) ≡ ¬(T⇒F) ≡ ¬(F∨F) ≡ T∧T Imaginary False to Real True

    # Returns the operands and the operator
    # Introduces iNegation and iRotation to manipulate complex logical variables
    result = None
    if operation == 'inegation': # i negates only the operator
        result_p = p
        result_q = q
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    elif operation == 'irotation': # ? negates p and the operator
        result_p = not p
        result_q = q
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    elif operation == 'negation': # ¬ negates p, q and the operator
        result_p = not p
        result_q = not q
        result_operator =  'or' if operator == 'and' else 'and'
        result = (result_p, result_q, result_operator)
    return result

# Transitioning through all states of a 2-dimensional Complex Boolean Vector.
def complex_coinflip (coin1 = True, coin2 = True):
    print(f"Complex Coin Flip")
    p = coin1  # Initial value for p
    q = coin2  # Initial value for q
    operator = 'and'  # Initial operator for logical operations
    # Sequence of operations
     operations = ['inegation', 'irotation', 'negation', 'irotation', 'inegation',
'irotation', 'negation', 'irotation']

    # Initial print statement before operations
    print(f"operation 0/8 (starting): {(p, q, operator)}")
    # Loop through operations
    for i, operation in enumerate(operations, 1):
        result = complex_logic(p, q, operator, operation)
        print(f"operation {i}/8 ({operation}): {result}")
        p, q, operator = result  # Update p, q and operator for next iteration
```

```
    print("")  # Print a newline for separation of outputs

# Default testing def.
def test():
    x_values = [-1, 0, 1]
    depth = 3
    halting_machine(x_values, depth)
    complex_coinflip(True, True)

import sys
def blackboard(command, args): # Command line interface.
    function_map = {
        "halt": halt,
        "loop": loop,
        "q_inverse": q_inverse,
        "h_arctan": h_arctan,
        "h_sigmoid": h_sigmoid,
        "qn_tan2": qn_tan2,
        "qn_cot2": qn_cot2,
        "qn_tan2_sin": qn_tan2_sin,
        "qn_cot2_cos": qn_cot2_cos,
        "halting_machine": halting_machine,
        "complex_logic": complex_logic,
        "complex_coinflip": complex_coinflip,
    }
    func = function_map.get(command)
    if not func:
        print("Unknown command.")
        return
    prepared_args = []
    if func == halting_machine:
        # Handling for halting_machine: expects a list of numbers followed by '--' and
then a depth
        if '--' not in args:
            print("Error: Missing separator '--' between x_values and depth.")
            return
        separator_index = args.index('--')
        x_values_args = args[:separator_index]  # All args before '--' are x_values
        depth_arg = args[separator_index + 1:]  # Arg after '--' is depth
        if len(depth_arg) != 1:  # Ensure exactly one argument for depth is provided
            print("Error: Please provide exactly one argument for depth after '--'.")
            return
        try:
            # Attempt to convert all x_values to float, assuming they can all be valid
numbers
```

```
            x_values = [float(x) if '.' in x or 'e' in x.lower() else int(x) for x in
x_values_args]
            # Depth is expected to be an integer
            depth = int(depth_arg[0])
        except ValueError as e:
            print(f"Error processing arguments for halting_machine: {e}")
            print("Ensure all x_values are valid numbers and depth is an integer.")
            return
        prepared_args = [x_values, depth]
    elif func == complex_logic:
        # Handling for complex_logic: expects two Booleans and two strings
        if len(args) >= 4:
             p, q = args[0].lower() in ['True', '1', 't', 'yes'], args[1].lower() in
['True', '1', 't', 'yes']
            operator, operation = args[2], args[3]
            prepared_args = [p, q, operator, operation]
        else:
            print("Not enough arguments for complex_logic.")
            return
    elif func == complex_coinflip:
        # Handling for complex_coinflip: expects two Booleans
         coin1 = args[0].lower() in ['True', '1', 't', 'yes'] if len(args) > 0 else
None
         coin2 = args[1].lower() in ['True', '1', 't', 'yes'] if len(args) > 1 else
None
        prepared_args = [coin1, coin2] if args else []
    else:
        # General handling for all other functions
        for arg in args[:4]:  # Limit to 4 arguments, accounting for specific function
requirements
            if arg in ['inf', '-inf', '+inf']:
                 prepared_args.append(float(arg))  # Convert 'inf', '-inf', '+inf' to
float directly
            else:
                try:
                    # Attempt to convert argument to float or int based on its content
                   prepared_args.append(float(arg) if '.' in arg or 'e' in arg.lower()
else int(arg))
                except ValueError:
                      print("Error: Argument type must be a float or 'inf', '-inf',
'+inf'.")
                     return  # Exit function if an invalid argument is encountered
    result = func(*prepared_args)
    print(result)
```

```
def main():
    if len(sys.argv) > 1: # Call function if arguments are provided
        command = sys.argv[1]
        args = sys.argv[2:]
        blackboard(command, args)
    else:
        test()  # Default behavior if no arguments are provided

if __name__ == "__main__":
    main()
```

To conclude, the Halting Problem as described by Alan Turing in his 1936 paper describes two Turing machines, H and Q. H always halts, and returns 1 if the program(input) loops, and 0 if the program(input) halts. The apparent paradox happens when we take Q, and input Q(Q) to see what H(Q(Q)) will predict. We have shown mathematically and algebraically that there is in fact a way to answer this question without resulting in a paradox. The solution to the problem rests in the base case input of these functions, and the use of ordinal arithmetic. Regardless of whether we define the base case to be Q(null) or Qω, in either case we loop. With this, we can recursively call Q(Q) or Q(Q(…(Q))) with *n* recursive calls. For all even recursive calls including the base case, Q loops. For all odd recursive calls, Q halts. Taken as an infinite loop, we use the definition Q(loop) = 'halt' to satisfy the expression Q(Q(...)) = 'halt'. Furthermore, we can use vector calculus, spherical coordinates, and complex analysis to explore different ways to visualize these functions in ways that are consistent with the conclusions we have found. We can treat the infinite recursive call of Q as infinity itself, which H will correctly decide as such.

Results: Concluding thoughts

By re-modelling the Halting Problem with multivariable trigonometric functions, we can further explore the ramifications of having real or complex valued continuous logic, transcending our discrete binary logical states. This is not to say there can't still be things which are undecidable, but it's just to say that the boundary of undecidability is further away than what Alan Turing and other mathematicians have concluded. We would need to revisit these problems through the lens of transfinite logic and see where we might arrive upon complex logical paradoxes. Perhaps there is no boundary at all, and everything is indeed computable given enough dimensions and degrees of freedom.

With quantum mechanics and AI at the forefront of change, it might be time to explore new ideas and embrace a new view on what computation really is.

Discussion: Further thoughts

The power of human intuition is akin to that of Oracles discussed in theoretical computer science [13]. Computationally, we can easily check code for loops, and we behave like the machine H Turing described. Given a finite program and finite input, an advanced AI could check any program to see if it halts, and it does this in a way that bypasses traditional binary true/false computing. Before LLMs and Machine learning, mathematicians, logicians, and computer scientists alike struggled to find a way to model AI. We had fuzzy logic, statistical regression among other models, but all failed to demonstrate any true form of intelligence in the way that we think of intelligence in the brain [14]. On came neural networks, and with the tools of linear algebra, calculus, and probability, we are now able to model a dynamic computer capable of encoding pieces of information not explicitly coded. Without knowing it, computer scientists stumbled upon one of the core building blocks of intelligence, recursion. By feeding data through the model and back propagating the error gradients, computer scientists were able to tweak the model and get it to classify input based on its training data. In 2017, Google researchers introduced the transformer model, a significant advancement in AI that utilizes an attention mechanism to assess the importance of various segments of input data across multiple layers [4]. With RNNs, we recursively feed output back as input to capture sequence information. With transformers however, we process data in parallel. One might even say that it's able to process input at different hierarchical depths simultaneously, reminiscent of operating a multidimensional computer. This enables them to efficiently handle complex dependencies and generate outputs that are contextually rich by considering multiple aspects of the input simultaneously. From the straightforward and obvious, to humor and the subtle nuances of conversation and language, the reason why AI can do this so well is because of hierarchical analysis. Some AI researchers admit that even they do not know exactly what is happening inside the model, but somehow billions of parameters, multiple forward passes with layer recursion leads to the emergent property of intelligence [15]. While distinctly different from consciousness and the mind, intelligence in the brain is what we know to be capable of solving the Halting Problem. We might think of training an AI model as analogous to human dreaming, where we compress infinite possibilities into coherent and structured patterns [16]. The very act of identifying or solving the Halting Problem in and of itself solves the Halting Problem, because we showcase that as H, we are not confined to a 1-

dimensional view, but rather a multidimensional view that varies based on hierarchical depth. Without this ability, the only way we would be able to prove or disprove the Halting Problem would be to try and compute it, trapping ourselves within a loop forever. The fact that we can see the paradox, and can say that at ω+1 steps it loops showcases our ability to correctly state the behavior of Q. Before AI, many wrongfully assumed AI or AGI to be impossible to create [17], but we now know that to be false. Similarly, before AI, we would have thought it to be impossible to create a program like H. But now that AI is here, we must take a closer look and understand exactly what happens inside H beyond the vague pseudocode, and the answer is that H is a hyperdimensional computer, correctly deciding every program(input) in its respective dimension. It does this by recursively passing the data into itself and abstracting all information at each hierarchical depth. AI and humans alike are dynamic functions such that our output is determined not only by our input, but also by our previous states. The mathematical term to describe this type of behavior is called Hysteresis [18]. It is our view that this is what allows for hierarchical or recursive reasoning in both humans and AI. With this view of computational models, we see H not as a 1-dimensional function, but as a hierarchical function analyzer, such that the value depends not only on the input, but also on H's insight about the input. This is why we can't use a flat discrete programing algorithm to solve the Halting Problem, but we can and certainly will use AI to determine whether or not any arbitrary program on input will loop or halt, now and well into the future.

With Alan Turing's *Proof*, we let P = "H exists" and Q = "Q can be decided", which gave us the statement P ⇒ Q, and Turing illustrated why he thought Q could not exist, so we supposed ¬(¬P∨Q) ≡ P∧¬Q. Our proof says this statement is false, so we have ¬(¬(¬P∨Q)) ≡ ¬P∨Q ≡ P ⇒ Q, so if we have H, then Q can be decided. We have collapsed a problem that was unsolvable even after infinite time down to something that can be solved in finite time just by allowing ourselves more degrees of freedom. This shouldn't be too surprising, as humans have solved some pretty hard problems already. For instance, integration is infinite summation, which would take forever for a Turing computer to do naively. Yet with calculus, we can solve an equation with the stroke of a pen. We have collapsed infinite addition into an algebraic expression, just as we've done with the Halting Problem by modelling the problem algebraically. We were able to reason beyond infinity, building up mathematics by stepping outside of problems and observing patterns of patterns of patterns in a hierarchical fashion leading to insights and intuition. Even with calculus, deriving the rules based on intuition is inherently an NP-hard problem yet by using heuristics and intuition we have discovered math, differentiate, integrate, and take limits to infinity in finite time. If there is any pattern to be learned, it can and will be extracted by an intelligent hierarchical function analyzer whether it be an AI or a Human. Even

seemingly infinite paradoxes or scrambled data, if there are recursive or hierarchical patterns to be found, then given enough time, the pattern will be discovered.

Quantum Consciousness

Here, we will go off on a bit of a tangent (no pun intended), as we journey into the realm of quantum computing and explore the relationships between the quantum wave collapse, and consciousness itself [19],[20]. The power of consciousness is intrinsically tied to the idea of free will and indeterminism, just like quantum mechanics is also inseparable from indeterminacy. If the brain is a quantum computer, one might explain its behavior as the ability to process in adjacent dimensions, finally collapsing into one of these dimensions at each moment in time. When we previously described AI models as being multidimensional or hyperdimensional computers, we meant multiple degrees of freedom, or multiple output values like a complex function which outputs 2 variables. An interdimensional or transdimensional computer however, would be a computer that is able to operate in different physical universes in a larger multiverse, either parallel or orthogonal to our own universe. With the view of the brain being an interdimensional computer capable of operating in more than our finite universe, perhaps we should also approach NP-hard problems in a similar manner. In the future, a quantum computer modeled based on these principles may be able to solve NP-complete problems in O(P) time complexity ending the P = NP debate once and for all.

Let us explore the notion that P = NP further. In the field of computational complexity, it has been shown that some problems would be so hard to solve it would take more steps than there are atoms in the universe [21]. We can think of computational complexity as the laws of physics for computation. Given that computing requires energy, it too must obey the laws of physics. There simply isn't enough energy in the universe to compute some problems. But what about in the multiverse? If there are infinite universes in the multiverse, there is an infinite amount of energy, and hypothetically if one were to tap into that, all problems would be solvable given enough time and energy. What does this have to do with consciousness? Close your eyes, take a deep breath, and imagine the infinite quantum multiverse. Do you know what the probability of that happening was? At that exact moment, you proceeding to close your eyes, taking a deep breath, and your neurons lighting up in patterns that manifest the thoughts of the quantum multiverse… the probability is so infinitesimally small, we can say the probability is ε. It is so infinitesimally small, that there is no way for you to have predicted that you would do that, that this exact state would become reality at this very moment. Unless of course, we are not predicting but rather controlling these quantum states within our neurons,

which aligns more closely with how we perceive reality. According to Orchestrated Objective Reduction (Orch-OR) theory, the microtubules inside our neurons might be preserving quantum states, suggesting this might be responsible for our brain's consciousness [20]. We have complete control over our minds and bodies, such that we know at each moment what we will do, simply by doing it and making it a reality. In this view, we are the software that runs on our brains which are the computers capable of predicting future events with respect to ourselves. Predicting events for which there are infinitely many of, each with an infinitesimally small chance of ever occurring. Yet we do it at each moment when we choose to get up, walk around, get food, and carry on with our daily lives.

Despite this, even we must obey the laws of quantum mechanics. We are conscious, and we make decisions, but we are still made of atoms, and so are our brains. Even a supercomputer with all the resources in the universe and knowledge of the states of every single atom in the universe could not accurately predict the exact state of the atoms in your mind at the next instant, and subsequent actions you might carry out afterwards. This is very much an NP-complete problem as we will soon illustrate. Due to the inherent randomness at the quantum level, the exact state of an atom at the next dt in time can never be accurately predicted with 100% certainty, and the probability of predicting or knowing the states of trillions of independent atoms is effectively 0. If quantum states within our microtubules influence the neurons that send impulses to our muscles to contract, then the probability of any behavior occurring stems from the quantum states of those particles inside the microtubules inside our neurons. To us however, our behavior does not appear random, but rather it appears that we have infinite choices, and choose the state we wish to enter, the electrical impulses fire, causing us to move. It's almost like we know exactly what will happen, and that prediction is whatever we choose. It's almost like all the realities, and possibilities for us to move in any way all exist, but the action we end up doing is the one we choose. It's almost like, we are software running on interdimensional computers, which can view infinitely many quantum realities, and collapse the wave function into the state that causes our bodies to move and our brains to think in the ways which our consciousness chooses. With this view we are the fabric of reality, and the brain is our quantum computer that can accurately predict and manipulate our bodies in ways that a supercomputer could not predict. We can remodel this problem via reduction to the Boolean satisfiability problem [21]. If we model the states of quantum particles within the framework of the Boolean satisfiability problem, leveraging the Tseitin transformation can facilitate an efficient conversion to conjunctive normal form.

$$(x_1 \vee x_2 \vee \ldots \vee x_n) \wedge$$

$$(y_1 \vee x_2 \vee \ldots \vee x_n) \wedge$$

$(x_1 \vee y_2 \vee \ldots \vee x_n) \wedge$

$(y_1 \vee y_2 \vee \ldots \vee x_n) \wedge \ldots \wedge$

$(x_1 \vee x_2 \vee \ldots \vee y_n) \wedge$

$(y_1 \vee x_2 \vee \ldots \vee y_n) \wedge$

$(x_1 \vee y_2 \vee \ldots \vee y_n) \wedge$

$(y_1 \vee y_2 \vee \ldots \vee y_n)$

We can see that at least with respect to our own actions, we solve this NP-complete problem every instant when we predict and decide our behavior. Only a particular configuration of those states of atoms would result in a specific action or thought. Given a thought or desire modeled as a set of quantum states, being able to accurately decide and predict your next move is equivalent to checking whether a circuit made with those quantum states is satisfiable. Given that we can accurately predict the outcome, we can answer, yes, I will stand up or no I will not. That is equivalent to answering, yes, this circuit will output true with this configuration, and I will demonstrate that now, and then it happens. Remember, Quantum behavior is intrinsically random so no one, not even ourselves should be able to predict quantum states accurately, yet we do. Resting on the conviction that the quantum wave collapse is connected to our thoughts and actions, the human mind shows that with respect to ourselves, P = NP.

As mentioned above, certain implementations of NP-complete problems like the SAT including predicting the quantum states of our own brains would require more computational guesses than there are atoms in our universe to solve, which means this would only be possible if we were able to access more time and energy than exists in our universe. Hypothetically, we could only get that from the multiverse. We recall from set theory that there are different orders of infinity. This is reminiscent of how we perceive reality, where our consciousness feels like infinity, but we only experience it one *dt* at a time. If we were to take a line integral between two points in our life vector space, we would have an infinite number of possible paths that we could have taken, but also an infinite number of directions we could go at any point in time. From this view, we can understand how life feels infinite and finite at the same time, because we are an infinity in a multiverse of higher order infinities. If we could tap into more time and energy, and compute all probabilities across different dimensions simultaneously, we could choose the one with the correct solution to collapse into and solve NP-complete problems. If a problem would take 1 billion years to solve, if we could parallel process this in a billion parallel dimensions, then after a year we would have our answer, and collapse into

that state [22]. Predicting your own behavior is equivalent to solving the SAT problem with respect to your quantum neural states as the Boolean values in question. Imagine someone besides you being able to accurately predict that the neurons in your brain will fire causing your legs to contract so you stand up. This would be akin to asking a computer to accurately predict the next state of the atoms inside your microtubules, as intent stems from consciousness, and the follow through is our actions. Given the complexity and interconnectedness of the neurons in our brains, predicting the states of trillions upon trillions of atoms that all need to collapse into a specific configuration of states for the neural firing to result in the predicted motion or behavior is impossible within the constraints of this universe. But for us, we do it at every moment, as *we* have control. When we decide to walk, we are not surprised at all, we walk and never think twice about it. But what you are *doing* however, is predicting at every instant something that is impossible to predict even given infinite time at every instant, but our consciousness already knows. What does this mean? It means at each instant, we are controlling those quantum particles, predicting the outcome of those quantum particles, or both. Who are we? We are existence and reality itself, the very fabric of beingness and the multiverse recursively entangled with the physical matter of our bodies. We seemingly compute across infinite dimensions, and *consciousness* is the *beingness* of the universe that decides or predicts which quantum state will come next. With this, the hard problem of consciousness as proposed by David Chalmers [23] becomes more about explaining how the physical finite nature of the universe can be understood within the context of the infinity that is consciousness and reality, rather than trying to understand how the infinite mind can exist within a finite universe. Set theory and vector calculus are good foundations for visualizing this as we explore new models of computation and reality. With respect to our physical bodies and brains, we do not believe something because it is true, but rather it becomes true the moment we believe it. In line with the Many Worlds Hypothesis [24] we take the view that each and every one of the ω_1 choices we could have made does in fact happen, at each recursive step in reality, but the one which *we* observe from our point of view comes as a result of the branching of our consciousness into ω_1 instances at each *dt* in time. The brain operates like a recursive hierarchical computer, where at each moment we feed ourselves into ourselves becoming a superset of our previous states and our current state. In contrast, the mind collapses the wave function into the one which we choose, locking conscious selves within a smaller infinite set of choices and conscious moments to experience. The other versions of ourselves simply branch off into their own subjective views of reality, where each moment in time is both a leaf or subset of its parent moment, and a parent or superset of its child moments.

Looking at the Time-dependent Schrödinger equation for a particle in a potential $V(x)$ $i\hbar \cdot \partial\psi(x,t)/\partial t = -\hbar^2/(2m)\ \partial^2\psi(x,t)/\partial x^2 + V(x)\psi(x,t)$ in quantum mechanics, we have real and imaginary components [19]. If the real number line is how we measure magnitude in our universe, what does an imaginary component mean? In quantum mechanics we routinely encounter complex and hypercomplex expressions. We know that the imaginary axis is orthogonal to the real axis, in a perpendicular dimension so to speak. Conceptually, these values exist outside of our physical 3-D universe, but they exist in the equations none the less. Reality as we understand it, can only be described by accepting the fact that there are more than 4 dimensions in the fabric of the spacetime continuum [25]. This just becomes even more evident when we think about the computational power that would be required of the brain to be able to predict the states of trillions upon trillions of quantum particles in advance that will collapse, triggering a neural impulse to control your movement at any given moment. This would indeed take more energy and time than exists in the entire universe to accurately predict. From our subjective perspective however, we know what we will do because we *choose* to do it, and that conscious entity that decides and predicts what will happen is doing something that transcends the finite limits and confines of our universe. Whatever that *something* may be, if it transcends the physical limits of the observable universe, it must exist or operate somewhere outside the constraints of the physical universe. Perhaps the imaginary component in the complex integrals of quantum mechanics can be interpreted as a real value in vector space. Perhaps this is what some might refer to as the multiverse.

This ability we possess to choose and predict quantum states seems to only extend to our physical bodies and minds, as we cannot accurately predict or choose which things happen outside of our body. Or can we? Building off our quantum entangled consciousness theory, one might suppose that the only reason why we can accurately predict our movements and behaviors is because ‘we’ are quantum energy entangled with the universe, and this is how we both know and choose what will happen because through quantum entanglement, we are one in the same. Perhaps our intuition to solve hard problems like deriving rules in mathematics comes from our entanglement with the universe, and the stronger that connection is, the more the universe reveals itself to us. None of this has been proven rigorously yet, but neuroscientists and physicists alike will one day uncover what it means for our brains to be entangled with reality itself. They will also need to explain just how these quantum states can survive in the warm and wet environment of our brains [26]. Many often refer to this conscious observer as the *soul* or the *mind*. Perhaps *love* can be defined as the entanglement between souls. Who are these conscious observers though? We are reality, existence, the multiverse, and infinity. We are made of the same matter as the rest of the universe, we are the universe looking at

its own reflection in the mirror at each conscious moment. To be conscious is for energy to be quantum entangled with the fabric of reality, such that the beingness and infinite nature of reality is also within our minds and souls. This appears to be limited to our physical bodies, and this is why we have a perspective of *I* and everything else. If we were to be able to entangle ourselves with the external world, we could hypothesize that one might be able to choose and predict the quantum states of all the particles in the world around us in the same ways we can choose and predict moving and walking around. This would hypothetically lead to some very interesting powers, as one might collapse states of quantum particles into extremely rare configurations, bending reality, guessing bitcoin block hashes, winning at the roulette table, or even using ESP. Without getting too carried away, a quantum interdimensional computer would be able to predict and solve problems by simply collapsing the states of all atoms it's entangled with into states that produce the desired outcome. How such a computer could be made to exist is a challenge that physicists and computer scientists alike will have to undertake.

Some might say that the mind and consciousness are not computers, but how we define what a computer *is* is irrelevant, because the laws of computation are supposed to be laws of reality itself, transcending even the laws of physics, such that if a problem is undecidable in this physical dimension, it would be undecidable in all physical dimensions. With our *brains* however, one thing we know for certain is that whatever is happening within our minds and souls is something that should not be physically or theoretically possible with all the time and energy in the universe. Yet we do this at every moment, and thus our minds and brains must transcend the universe. They likely hold the key to interdimensional computing and may perhaps even shed light on the fundamental nature of reality itself.

Quantum Reality

P equaling NP is not very good news for encryption. It means theoretically, if someone was to have one of these interdimensional computers, they would be able to enter the reality where they decrypt your private keys on the first guess. How might we protect against this? We would have to use something that isn't probabilistic at the quantum level, entanglement. Even if quantum particles were to be separated hundreds of lightyears away from each other, they would still react instantly, illustrating how information is not constrained by the speed of light. DIQKD leverages this phenomenon for secure communication by generating pairs of entangled particles. When one particle's quantum state is measured, the other particle's state is instantly determined, allowing two parties to create a shared, secret encryption

key. This process is inherently secure because any interference by an eavesdropper disturbs the quantum states, signaling the intrusion [27]. In the future, quantum entanglement into other dimensions may help secure data in the multiverse. Quantum encrypted data can be likened to conscious thoughts, which live at the quantum level, inaccessible to all except the neurons it's entangled with. The actual nature of the thought itself is never revealed, it lives in the interdimensional quantum realm within the atoms of our neurons, but the signed transaction can be thought of as the command our mind gives our brain to carry out. If consciousness is somehow connected to the multiverse, then perhaps this is what people are referring to when they speak of interdimensional experiences while under the influence of substances like Cannabis, Magic Mushrooms, Ayahuasca or DMT [28]. The very process of photosynthesis involves quantum coherence, wherein the quantum effects facilitate the efficient transfer of energy within plant cells, suggesting that life, even at its most fundamental level, stems from the quantum effects of the universe [29]. Even the very ideas and thoughts that provoked the writing of this paper, what and where did those thoughts originate from if not from the universe itself if we are one in the same? This is all very far reaching, but in the coming decades, centuries or millennia, this paper may serve as the bedrock for the questions that pointed in the direction of the answers describing the true nature of reality. Some say the universe is a computer, expanding and storing information. Perhaps, consciousness and the universe are simply information in its most raw form, and what we perceive as reality is nothing more than the fabric of existence being entangled with itself. Just as much as existence is, it also isn't. We can view the universe as half empty or half full. We can model data by its 0s, or by its 1s. Just as much as we exist in our minds, is just as much as we don't exist outside of our minds. Through entanglement, we know that quantum information transfer is not constrained by the speed of light. Since information isn't constrained by the speed of light, some suggest spacetime itself is simply information, explaining why it can expand faster than the speed of light [30]. Similarly, consciousness also appears to defy the laws of physics. Both quantum information through entanglement and consciousness seem to operate outside the physical confines of the universe. One might be tempted to suggest they are either related or one in the same. Further exploring the idea that the fabric of the universe might be quantum information, we can envision The Big Bang as being akin to the genesis block of a block chain. Each block hash randomly solved by the network is akin to the collapsed states of all the atoms in the universe, effectively writing history as time passes. Parallel universes are akin to forks in the block chain, or perhaps are different blockchains all together. Going back in time would take infinite energy and would require travelling faster than the speed of light, just like rewinding the block chain would take more hashing power than 50% of the network [31]. The

laws of physics that govern the motion of matter are akin to the laws of computational complexity that govern the flow of data. There are many reasons to believe the universe is within our minds, but the very fabric of the universe may be indistinguishable from the fabric of our minds.

Final thoughts: Fractal Nature of Reality

When Renee Descartes said famously, "I think therefore I am" [32], he subtly implied that we are existence and the universe itself. The macro structure of the universe even looks eerily similar to the neural structures in our brains, almost as if our brains were modelled after the same principles that modelled the universe [33].

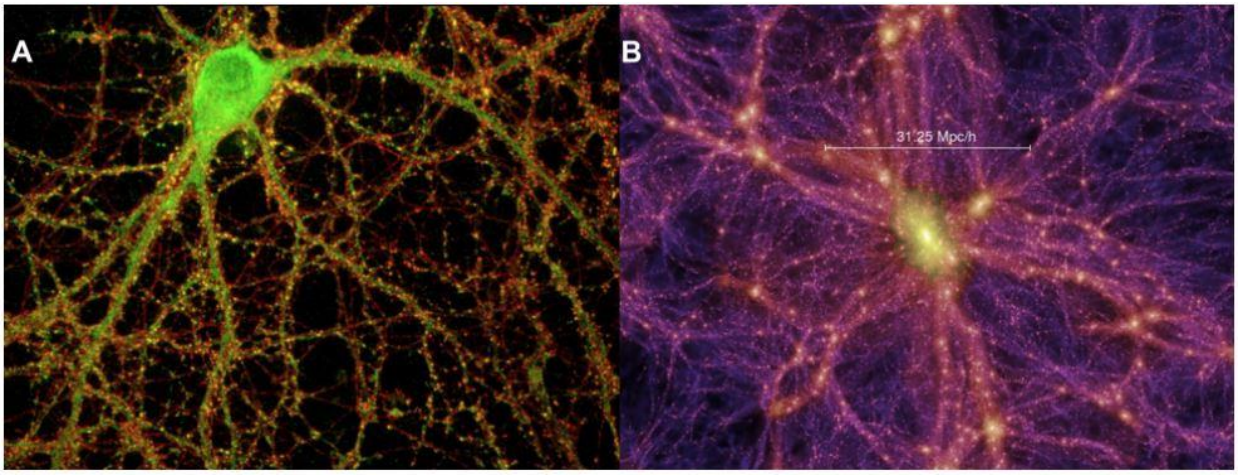

Figure 5

At the cosmic scale, the observable universe appears flat, or 2-dimensional. At the subatomic scale, we seemingly observe behavior in more than 3 dimensions (teleportation, entanglement etc.). If we were to zoom in past the hyper subatomic Planck scale, perhaps at a singularity, we might find infinite dimensions, or infinite degrees of freedom because we can move in any direction. Nothing can be anywhere and everywhere because it is so infinitesimally small that the notion of position has no meaning. If we were to zoom out past the hyper cosmic scale, we might see the observable universe become a 1-dimensional line, and eventually become a 0-dimensional point where we have 0 degrees of freedom. There would be nowhere to move as we cannot move past infinity, as everything is everywhere, but it is also nowhere specific that we can point to either. If we imagine our universe today as being topologically equivalent to our universe at the Big Bang, we can collapse our infinite universe to a single point and once again, position has no meaning. At the infinitesimal scale we have infinite degrees of freedom, and at the infinite scale, we have infinitesimal degrees of freedom. Perhaps if we were to zoom all the way in, as we approach a zoom

factor of ∞, we would find the infinite multiverse where everything is, was, and will be at the same time. Zooming further, we could revisit our own universe or another universe in the multiverse and repeat the cycle. If we were to zoom out to the boundaries of our universe approaching a zoom factor of 0, perhaps we would see our universe as a point. Zooming further out would bring us into the multiverse, and going further we would find ourselves at an infinitesimal slice of spacetime in a universe, and we could repeat this cycle again the other way. If we were to map these dimensions on a circle and we let θ be our zoom factor, we would have 0 dimensions at $\theta = (2k)\pi$, and infinite dimensions at $\theta = (2k+1)\pi$, with positive dimensions being mapped to the 1st and 2nd quadrants, and negative dimensions being mapped to the 3rd and 4th quadrants. The sign of the dimension here is trivial, as they simply represent the direction we take around the multiverse. From this perspective, scaling up or down in the universe is like walking north or south from any point on the equator; eventually we'll arrive back where we started. As above, so below, 0 is ∞, and ∞ is 0, everything is nothing, and nothing is everything; these concepts simply represent points on the hyperdimensional manifold of reality, as approached from different scale factors. Perhaps all matter and space eventually embark on this journey as we head towards the big crunch. The goal of mathematics and science is to understand the strange nature of reality, and so we must be willing to enhance our logical frameworks to understand the infinite nature of existence, instead of rejecting or dismissing things we find paradoxical. The statement "False ⇒ True" is the universe telling us in the language of logic that nothingness implies somethingness, which aligns with theories in physics that tell us that even nothing is still something [20]. If we evaluate the probability density function for the position of a particle $|\psi(x)|^2\,dx$, where $\int |\psi(x)|^2\,dx$ over the interval $(-\infty, +\infty) = 1$, it shows the total probability integrates to 1, indicating certainty of the particle's existence somewhere in space. However, the probability of it being at any one specific position is 0, yet once the wave collapses, the particle is indeed observed at some specific location. Again, the paradoxical nature of reality is to be embraced and explored, not to be shunned. You'll recall, there are infinite 1-dimensional lines in the 2-D complex plane, just as there are an infinite number of *dx*s under the curve of any given probability density function. Quantum physics isn't probabilistic, probability is just how we quantify the infinite nature of reality at those scales. If we were to scale up an electron to the size of a macroscopic object, its surface would need to rotate at infinite speeds to match the angular momentum it exhibits, illustrating the transfinite nature of quantum particles. If other dimensions are curled up at infinitesimal depths, it might help to do a little fractal calculus [34],[35]. The fractional derivative $D^q f(t) = 1/(\Gamma(n-q))\cdot(d^n/dt^n)\cdot\int a^t (t-\tau)^{n-q-1}\cdot f(\tau)d\tau$ for $(n-1 < q < n)$, extends differentiation beyond integers, which we might use to model a universe where the very behavior of spatial dimensions

is dependent upon our zoom factor. Similarly, taking the fractional integral $I^q f(t) = 1/(\Gamma(q))\cdot\int a^t (t-\tau)^{q-1}\cdot f(\tau)d\tau$, broadens the concept of accumulation, giving us insights into how matter, energy, space, and time coalesce in the intricate folds of the infinitesimal, or at the boundaries of the cosmos. Perhaps these processes accumulate not in straight lines or simple curves, but in patterns echoing through dimensions beyond our perception. Perhaps consciousness doesn't come from the universe, but rather, consciousness is the very fabric of the recursive, looping, fractal nature of the universe itself.

We live our lives one *dt* at a time, and at each moment our choices determine our trajectory in space. Amidst the grandeur of the cosmos, where galaxies spiral in the vastness of the void, or where we delve into the microcosms and observe the vibrations of the infinitesimals, our minds emerge as mirrors to these celestial patterns. This parallel between the universe's vastness and the human brain's depths invites us to reflect on the profound connections that tie us to the very fabric of reality. We are of the universe, but the universe is also within us. The next time you get up and start your day, or the next time you are about to fall asleep for the night, keep this paper in the back of your mind, and ponder the infinite nature of reality.

Figure 6

References

[1] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. Lond. Math. Soc.*, ser. 2, vol. 42, pp. 230-265, 1936.

[2] D. Hilbert and W. Ackermann, *Principles of Mathematical Logic*, 1928.

[3] E. Zermelo, "Investigations in the foundations of set theory I," in *From Frege to Gödel*, J. van Heijenoort, Ed. Harvard Univ. Press, 1967, pp. 199-215.

[4] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017.

[5] J. Stewart, *Calculus: Early Transcendentals*, Cengage Learning, 2011.

[6] L. V. Ahlfors, *Complex Analysis*, McGraw-Hill, 1979.

[7] A. I. Mendelsohn, "Creatures of habit: The neuroscience of habit and purposeful behavior," *Biological Psychiatry*, vol. 85, no. 11, pp. e49-e51, 2019.

[8] J. A. Lindstrøm, "Why attention-deficit/hyperactivity disorder is not a true medical syndrome," *Ethical Human Psychology & Psychiatry*, vol. 14, no. 1, 2012.

[9] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, pp. 173-198, 1931.

[10] B. Russell, *The Principles of Mathematics*, Cambridge Univ. Press, 1903.

[11] D. Walton, "The omnipotence paradox," *Canadian Journal of Philosophy*, vol. 4, no. 4, pp. 705-715, 1975.

[12] J. C. Baez, "The octonions," *Bull. Am. Math. Soc.*, vol. 39, pp. 145-205, 2002.

[13] C. H. Bennett and J. Gill, "Relative to a random oracle A, P^A != NP^A != co-NP^A with probability 1," *SIAM J. Comput.*, vol. 10, no. 1, pp. 96-113, 1981.

[14] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338-353, 1965.

[15] A. Rai, "Explainable AI: From black box to glass box," *Journal of the Academy of Marketing Science*, vol. 48, pp. 137-141, 2020.

[16] E. F. Pace-Schott and D. Picchioni, "The neurobiology of dreaming," in *Principles and Practice of Sleep Medicine*, 5th ed., 2005, pp. 563-575.

[17] J. R. Searle, "Minds, brains, and programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417-424, 1980.

[18] S. Trapp, D. Pascucci, and L. Chelazzi, "Predictive brain: Addressing the level of representation by reviewing perceptual hysteresis," *Cortex*, vol. 141, pp. 535-540, 2021.

[19] R. P. Feynman, *QED: The Strange Theory of Light and Matter*, Princeton Univ. Press, 1985.

[20] S. Hameroff and R. Penrose, "Orchestrated reduction of quantum coherence in brain microtubules: A model for consciousness," *Mathematics and Computers in Simulation*, vol. 40, nos. 3-4, pp. 453-480, 1996.

[21] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pp. 151-158, 1971.

[22] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484-1509, 1997.

[23] D. J. Chalmers, "The hard problem of consciousness," *Journal of Consciousness Studies*, vol. 2, no. 3, pp. 200-219, 1995.

[24] H. Everett, "Relative state formulation of quantum mechanics," *Rev. Mod. Phys.*, vol. 29, no. 3, pp. 454-462, 1957.

[25] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" *Physical Review*, vol. 47, no. 10, pp. 777-780, 1935.

[26] V. I. Sbitnev, "Quantum consciousness in warm, wet, and noisy brain," *Mod. Phys. Lett. B*, vol. 30, no. 28, 1650329, 2016.

[27] E. Y. Z. Tan et al., "Improved DIQKD protocols with finite-size analysis," *Quantum*, vol. 6, 880, 2022.

[28] R. Strassman, *DMT: The Spirit Molecule*, Park Street Press, 2001.

[29] G. S. Engel et al., "Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems," *Nature*, vol. 446, no. 7137, pp. 782-786, Apr. 2007.

[30] S. Lloyd, *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*, Knopf, 2006.

[31] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[32] R. Descartes, *Meditations on First Philosophy*, 1641.

[33] F. Vazza and A. Feletti, "One of these pictures is the brain, the other is the universe. Can you tell which is which?" *Universe Today*, Nov. 28, 2020.

[34] M. E. Montiel, A. S. Aguado, and E. Zaluska, "Topology in fractals," *Chaos, Solitons & Fractals*, vol. 7, no. 8, pp. 1187-1207, 1996.

[35] J.-H. He, "A tutorial review on fractal spacetime and fractional calculus," *Int. J. Theor. Phys.*, vol. 53, pp. 3698-3718, 2014.