On The Quantum Computability Of The 3 Satisfiability Solution Space

Adewale Oluwasanmi

waleoluwasanmi@yahoo.com

Abstract

We present a treatise on a new quantum theoretic algorithm for solving the 3 Satisfiability problem. The presented algorithm is not a standard quantum algorithm in the sense that it is intended solely for true "physical" quantum systems (if it at all can be realized on these systems). Instead, we posit that the 3 Satisfiability problem has an intrinsic complex quantum form that can be programmed in order to build a model of the solution space for satisfiable instances or show that such a model cannot be constructed. This yields surprising results on the ability for classical systems to abstractly simulate general quantum systems. We also present other relevant structures, mathematical and physical, that bear close analogies with the methods and structures presented.

1. Introduction

The 3 Satisfiability problem is a popular problem in the field of computer science notable for being one of the quintessential problems in the NP complete space. This article does not address classical notions of the complexity level of the problem in this regard.

Furthermore, we do not treat the problem in its regular logical form as a conjunctive normal formula over disjunctive unit clauses. We instead cast the problem into a novel quantum computational form or "type". This treatment converts the initial propositional formula into a quantum programmable type (strengthening the propositions as types conception) whose particular type is algebro-combinatorial (an idea to be fleshed out in detail). This assumes that the reader is familiar, if needed, with the normal classically logical formulations of the problem in CNF form as well as how the maximal operational space of that formulation classifies the problem as NP complete in the literature. We do this, because as the reader will suspect, our quantum programmatic formulation ultimately leads to new perspectives on this complexity classification.

The approach we present is intended to address the major source of uncoordinated exponential blow up or "drift towards infinity" in the classical approach, namely that attributable to the worst-case element of brute force search. To address this, we propose an alternative, "renormalizable" model where the original clauses are converted to a discrete solution form and the totality of potential and actual solution spaces are modelled as sets of "evolutions" of combinatorial species along with an associated algebra of mutation/generator functions which extend the species by operating over their interference patterns. The species are quantum "compressions" or representations of partial or complete solution spectra (if any solutions exist) and the generators construct complex "generations" of this space. In the case of unsatisfiable instances and instances with just one solution, the initial generation, termed first generation, is deemed complete. For instances with more than one solution, first generations serve as a complex differentiable base case over which other generations can be computed. All successive species constructed after the initial generation are collectively termed second generation species. Actual sets of satisfying assignments to unit propositions can be regarded as output/base species (that is they are of the same type, species, as the clausal space in our formulation and are complex homeomorphic with other species – if we regard all species as quantum deformable functions, with unit assignments as invertible constant functions as will be demonstrated.

The article is divided into 5 broad sections as follows:

1. The basic algorithm.
2. Mathematical analogs.
3. Physical analogs.
4. Conclusion.

The first section will be comprehensively covered as our aim is to rigorously show that the basic quantum theory of the algorithm is sound and complete for all problem instances. Other sections will be less formal and are simply intended to demonstrate the versatility and power of our reasoning strategy in analyzing these systems/analogs.

The overall style pursued is functional in spirit as opposed to formal since our approach is algorithmic and relies less directly on formal methods established in more rigorous mathematical areas. Hence our main instruments of proof will be descriptions of structures and the functions that operate on these structures along with arguments to prove the correctness (soundness and completeness) of the entire procedure. A main requirement will be for the reader to be able to follow along and understand these (sometimes informal descriptions), the operations that change them and the resulting (further) structures derived from applying the operations.

2. The Basic Algorithm

2.1 Quantum Programs

We construe our algorithm as manipulating quantum programs, hence the term, quantum algorithm. These quantum programs, which we also term quantum injunctions, can be taken to be the programmatic equivalents of standard quantum computational elements, say qubits or qudits, that is, they are morphable and their morphisms can only take on standard quantum values. Ignore the fact that these programs themselves are the problems to be solved, that is, they extend classical quantum computation with the notion of self-solving quantum systems. This makes the programs a form of abstract quantum matter which, with some ingenuity, can be used to study other (physically concrete) quantum systems. Note that because we are dealing with a virtual form, these morphisms will always take on their desired quantum values, that is, we have no probabilistic element in the orientation or "spin" of the programs, providing an abstract and absolute form of topologically secure quantum computation.

2.2 Unit Variables and Literals

Unit literals represent traditional propositions that can be assigned true or false. Each literal presented can be identified by a subscript (natural number) and sign (+ for unnegated literals and – for negated literals). For brevity sake, we only show signs for negated (-) literals and assume that unsigned numbers are positively signed. Every literal and its opposite (inverse) are antipodes the same variable (simply identified by the lower letter "x" attached to their common subscript) which is their algebraic "superposition". The signed representations (literals) are the only elements directly available to functional manipulation but we will continue to use the term "variable" to refer to the abstract sense of assignment so we can talk of assigning a variable, say $x_{3,}$ to one of its assignable literals, 3 or -3 , that is, we make that particular literal true and the opposite literal (same variable, opposite sign), false. The selected notation for literals is similar in spirit to the DIMACS CNF format used by many classical SAT solvers.

2.3 Basic Quantum Program Structures

Each initial disjunctive clause is represented by a Quantum Injunctive Clause, which we will call an X structure. This is a comma separated list of literal values, where the integer part represents the variable subscript and the sign is the OPPOSITE of that carried by the literal in normal CNF format, nested within opening and closing braces. For example:

Clausal form $(\neg x_1 \vee x_4 \vee \neg x_7)$ is transformed to X structural form {1, -4, 7} and

Clausal form $(x_3 \vee \neg x_5 \vee x_6)$ is transformed to X structural form {-3, 5, -6}.

The entire CNF formula itself is represented by a different structure termed a Y structure which is a comma separated list of all (valid) X structures. We will shortly explain the notion of a valid X structure. This gives Y structures a richer, computationally effective (functional versus formulaic) structure. For example:

The formula $(\neg x_1 \vee x_4 \vee \neg x_7) \wedge (x_3 \vee \neg x_5 \vee x_6)$ is transformed to Y structural form {1, -4, 7}, {-3, 5, -6}.

The last class of structures, Z structures, are the actual satisfying solutions to some problem instance S. A Z structure is represented as a comma separated open list (no opening or closing braces) of assignments (literals that have been made true). For example:

The Y structure: {1, -4, 7}, {-3, 5, -6} is satisfied by the Z structure: -1, -4, 7, 3, 5, -6.

Note how in our example, only the first literal in each Y structure has its sign reversed in the Z structure and is the only assignment that satisfies the respective underlying disjunctive clause (because, as we have explained, X structures reverse the sign).

Signs are reversed in X structures for the following reason: Computationally, X structures can then be interpreted as declared conditional programs for which the following axiom must be satisfied:

> Axiom 1: "For some X structure, X1, composed of n distinct elements and for any program P1 with an attached, partially completed Z structure, Z1, that satisfies (must satisfy) the given instance S containing X1, if n-1 of the literals in X1 have already been added to Z1, that is, these n -1 variables in Z1 have already been assigned to literals which do not satisfy the classical disjunctive clause represented by X1, the program P1 must immediately assign the nth, unassigned literal, call it k, in X1 to its opposite Thus, we extend Z1 to a new Z structure, Z2, that satisfies X1. We call Z2 a satisfying completion of X1 over k".

For example:

> The Y structure: {1, -4, 7}, {-3, 5, -6} has the partially completed Z structure: -1, -4, 7, 5 at some time t and at the successive time t + 1 we extend the solution to -1, -4, 7, 5, -6 by adding -6.
>
> Note that the second X structure is unsatisfied at the new time t + 1 and that we must immediately at what we call time t + 2, add 3 (the opposite of -3) to the Z structure, obtaining the satisfying solution: -1, -4, 7, 3, 5, -6.
>
> At time t + 2, we can regard the previous n–1 unsatisfying literal assignments as the INPUT portion for the inclusion of the satisfying literal assignment (the OUTPUT) of the anonymous function locally defined (in space and time) on the X structure. This qualifies us to call each structure X of size n, an n-dimensional program encoding n functions of input size n-1 and output size 1.

Axiom 1 treats X structures as algebraic types from which we can construct a new type via pattern matching. Since Y and Z structures vary along with X structures, they can be regarded as algebraic types themselves. Speaking algebraically then, we think it worthwhile to make the following section closing statements by speaking of the "cycle" structure of the 3 algebraic types:

> Every completed satisfying program P1 outputs a permuted/chained variation of a permutable set of assignments C (that is, the elements of C should be assignable in any order). We say each X structure carries a partial cycle on C chains.
>
> Every X structure of size n that carries a cycle on some combination C, over some output literal j, also carries a (exactly n-1) cycles on other C structures for which the negation (contra-variation) of j is an input assignment. Thus, every X structure is a co-cycle over its n variants, that is, every X1 structure is a partial co-cycle over its n associated C structures.
>
> On the other hand, Y structures must necessarily carry a total co-cycle over every possible combination (where solutions exist), at every instance.

Z structures are just particular chain realizations of some covered combination C.

2.4 Injunctive Space Completion

Here, we introduce an additional axiom and its corollary that both define a partial binary "integration" operation over X structures.

Axiom 2: "Take any 2 valid (again, we will explain invalidation rules shortly) X structures, X1 of size m and X2 of size n. If X1 and X2 share a common variable x_i, such that x_i takes on opposite signs (literals) in both structures AND X1 and X2 share no other variables, DIFFERING IN SIGN, we must generate and add to the global Y structure, a new X structure X3, composed of the m-1 elements in X1 and n-1 elements in X2, differing from x_i".

This can be read more procedurally as: "If Y is to contain only valid X structures, then it must admit only X structures that do not lead to assignment contradictions when applied in parallel since by Axiom 1, if we do not construct X3, x_i will be assigned two contradictory values if it is the last value assigned in both X1 and X2 by some arbitrary program P". Thus, Axiom 2 defines a binary parallelization/synchronization operator on our X structural programs.

Corollary 2.1: Self Contradiction/Reduction: "Take 2 structures X1 and X2 both of size n having the same set of variables and in which n-1 of the variables are the same and agree in sign (same literals) and also having an nth common variable that differs in sign (opposite literals). We must generate a new structure X3, composed of the agreeing n-1 literals. X1 and X2 are now invalid as they have been replaced/reduced by X3, a valid substructure of each.".

2.5 First-Generation Algorithm

i. Take a given 3 SAT problem instance S, convert all of its clauses into X structures and add them to a list called "incoming".

ii. Create a new list called "processed" to add structures that have been picked up from "incoming" and processed as described in step iii.

iii. Process elements of "incoming" in a loop, and for each iteration:
Remove the first "incoming" structure X1 and compare it to each applicable element in "processed" using Axiom 2.
If we generate a new X structure during a comparison with 5 OR FEWER elements, add it to "incoming", else discard it (NOTE THIS STEP).
Add X1 to "processed.
Continue until all elements in "incoming" is empty.

iv. If at any point, we generate two X structures X1 and X2 each of size 1 and each containing the same variable but where each variable is the literal opposite of the other, X1 and X2 are invalid and S is unsatisfiable, otherwise, if we exhaust "incoming" and no such occurrences are recorded, S is satisfiable.

2.6 Proof of Correctness

It should be quite obvious from the description above that we are dealing with a polynomial time algorithm (all considered X structures are bounded by a finite size of 5) with no clear (linear) reason why the procedure should work. The best way to prove the correctness of this algorithm will be to examine its edge cases.

First, we must agree that if we complete our first generation such that by our criteria above, the problem instance S is satisfiable (to be proven), then all valid X structures describe necessary actions that must be taken at "boundaries" by any satisfying program P according to Axiom 1.

Secondly, during second generations, as P assigns variables to literals, any X structure can be rewritten/rescaled with a structure reflecting all completed assignments. For example:

Say we have some X structure {3, -5, -7}.

If we assign the variable x_5 to false, that is, our Z structure contains -5, we can perform the following rewrite:

{3, -5, -7} to {3, -7}.

This ensures that the X structure indicates the new exact boundary rule that satisfies the underlying disjunctive clause.

This is what we mean by X structures being quantum computational elements that can be reprogrammed by taking a quotient on a structure to obtain a valid (and necessary) quantum "adjoined" boundary rule.

Henceforth, we will continue to use particular X structures with clearly defined literals (1, -1, 2, and so on) to demonstrate all described operations in the proof. As the reader will see, this use is abstract enough to demonstrate all necessary points.

Next, consider any two X structures of size 2 such that X1 = {1, 3} and X2 = {1, -3}. Clearly, the variable x_1 cannot be assigned a positive value 1 at any point in the program because this leads to an unsatisfiable condition/assignment dilemma on x_3, as x_3 then cannot be assigned either a positive or negative literal value.

The main line of the proof will show that upon successful completion of a first generation on some arbitrary 3 SAT instance, we never run into the assignment dilemma mentioned (in a very exact, abstract sense which we will demonstrate). This fact couples with the additional fact that because our X structures always denote only valid actions that must be taken at "boundaries", only invalid solutions are ever excluded at each point in time. The result is that our proof is able to show that we can after first generation, continuously assign values to all variables, where some assignments may be free and others bound (to the value necessitated by the boundary rule indicated by some valid X structure).

The approach taken can be used to show that our algorithm is sound and complete (at every point in time) with respect to satisfiable instances. The details are a little subtle and so we ask the reader to pay close attention both individual steps and their combined effect in order to validate the reasoning.

Henceforth we will convert the term first generation into a single formal word – first-generation (note the hyphen). We will also apply this compression to the associated term second generation which will be referred to as second-generation.

Here are the edge cases that prove our approach:

Case 1:

X structures {1, 3} and {1, -3} exist as part of the first-generation completion.

By Axiom 2, this cannot be the case at all since we will have combined both structures into the 1 element "constant" structure {1}.

Case 2:

X structures: {1, -3}, {1, 2}, {-2, 3} exist as part of first-generation completion.

Here assigning the variable x1 to 1 forces the following other assignments by virtue of Axiom 2 (X structures to the left of colon: assignment implications to right of colon):

{1, -3} :3

{1, 2}: -2

{-2, 3}: -3

But by Axiom 2, this cannot be the case as the following actions would have been taken in the first-generation stage:

{1, 3} and {-2, 3} would have yielded {1, -2} which would combine with {1, 2} to yield just {1}.

In this case, this means that the only assignable literal to x_1 is -1 (the inverse of 1).

Case 3:

{1, 3} and {1, -3} exist as part of a rewrite of two independent X structures, X1 and X2 each of size 3.

Say we originally had X1 = {1, 3, 4}, X2 = {1, -3, 5}.

By axiom 2, we would have generated an extra structure X3 = {1, 4, 5}, such that on adding 4 and 5 to the Z structure of a program, we will be forced to add -1 by virtue of Axiom 1. This ensures that the rewrites of X1 and X2, though seeming to be a dilemma have both been invalidated by a "smaller" rule which subsumes them both and that from a holistic algorithmic point of view, no "actual" dilemmas are generated at any point. In fact, upon encountering such a dilemma, we can simply rewrite both "rules" to {-1} by the demonstrated assurance This is a subtle point but is crucial to understanding the correctness of our algorithm.

If in this case, we happen to have another structure {-1, 4, 5}, then we would cancel X3 and 4 and 5 could not be assigned together.

Be convinced that 4 and 5 in this case need not necessarily be different literals, but by virtue of their difference, show the maximal extension of the case.

Case 4:

Slight repeat of case 2 for rewrites.

We have following X structures as a result of rewriting three different 3 element structures: {1, -3}, {1, 2}, {-2, 3}. As shown in case 2, this also generates a dilemma for assigning x_1. Let us again, use a maximally extended case in which the 3 different structures use 3 different extra variables:

X1 = {1, -3, 4}, X2 = {1, 2, 5}, X3 = {-2, 3, 6}

On first-generation, applying axiom 2, we would have generated the following X structures:

{1, -2, 4, 6}, {1, 3, 5, 6} and {1, 4, 5, 6}.

It is again clear that assigning 4, 5 and 6 will explicitly force us to assign x_1 to -1. Again, as before, any seeming intermediate dilemmas are invalidated by the explicit representations of all dependencies.

Case 5 (terminality case):

So far, we've only considered contractions from X structures of size 2 and 3 each. One wonders how 4 and 5 element structures are affected.

Consider the example given in case 4: {1, 4, 5, 6}, This structure ensures that upon assignment of 4, 5 and 6 we immediately add -1. Definitely, if we have some other 4 element structure like {-1, 4, 5, 6}, both structures would be invalidated and it would be impossible to assign all 3 literals in any satisfying assignment.

One more question we can ask is, if only {1, 4, 5, 6} exist and not {-1, 4, 5, 6}, combinatorial-wise, how does the selection of 4,5,6 and -1 "act" on the space. Specifically, is it possible to have the following two 5 element structures:

{-1, 4, 5, 6, 7} and {-1, 4, 5, 6, -7}?

The existence of these 2 structures would mean that we could not assign -1 as expected in addition to 4, 5 and 6 since we would then not be able to select any value for x7 and would therefore need to generate a new structure {-1, 4, 5, 6} which would combine with {1, 4, 5, 6} to yield {4, 5, 6}.

Obviously, the above boundary statement would be true whether we computed up to 5 element structures or not and so we ask, why stop at 5 element structures and not continue on to 6 element structures and larger so we can discover all invalid combinations up to the full signature of the problem (which should be obvious would result in an exponential structure space).

To explain why 5 element structures bound our "quantum loop", note the following:

i. All original structures contain just 3 elements in a standard 3 SAT CNF formula. Other 3 element structures can be derived from the combination of 3 element structures or 3 element and 2 element structures (if one works some examples out by hand, which is recommended to get a feel for how these structures may be manipulated). 3 element structures can become contracted during first-generation and lose literals and so can structures of other sizes. However, we can show that any literals existing in 4 and 5 element structures can ultimately trace their origins to 2 and 3 element structures and when these smaller size structures lose a literal, the loss will propagate to the larger sized structures inductively. In fact, one can in addition say that when any structure representing an original clause (from the initial instance) or any other generated structure for that matter, loses a literal, any combinations (3, 4 and 5 element structure) built up from it also lose that literal. The reasoning behind these two assertions are as follows: Since structures are formed from other structures by what we call the principle of unit contravariance (a new structure is formed when two structures differ in sign over a single variable), we only have two possibilities: the original structure loses the contravariant literal (which renders the derived structure unnecessary) or we lose some other literal and we can simply observe that any structure X' of n elements invalidate any structures of more than n elements of which the n elements of X' form a proper subset. This is because the smaller structure forms a more restrictive boundary condition than the larger structure in this case.

ii. Since we have shown (from case 4), that every 4 structure element will always output a value which does not immediately result in a dilemma over some fifth unknown variable, we can assume that upon the output of a 4 element structure, we can continue to freely assign other variables to literal values from the local point of view of the particular X structure.

iii. From I, ii and iii, we can conclude that using only 3 element (and no more than the structures which are direct conversions of clauses in the original propositional instance S) and all lower sized structures, which together we call the basis set, we can continuously assign every variable constructively to a value because no assignments "cancel" out/result in a dilemma at any point during the process FROM THE ASSURANCE GIVEN BY ALL AVAILABLE INFORMATION. We can further argue (cyclically) that we also need no more information than that which allows us to successfully contract structures from the basis set and other smaller sized (2 elements being the other real possibility) structures without encountering dilemmas. One can say we have the maximally bounded interference-free "wave function" of the entire solution space, forming a closed self-defined recursive loop.

iv. We can then logically conclude that we can always continuously assign variables to literals and also that at no point would we exclude any valid assignment. This shows that our algorithm is sound and complete AT ALL TIMES with respect to all satisfiable instances.

One last set of non-crucial points to observe is that: 1. Freely assignable variables not bound to a represented structure may occur at any point during the course of running the algorithm. We will call these structures radicals and 2: 1 element structures (constants) may occur during first-generation prior to second-generation selections and that in fact all variables over the instance signature may collapse into just one assignable literal each, generating only a single combination instance, in which case there are no possibilities for second-generation modifications. This is a closed loop.

Combining all said proof points, we conclude that if we can't construct a first-generation model (dilemma over some arbitrary variable), our instance S is unsatisfiable, otherwise it is satisfiable and we can generate a sound and complete exact model of its solution space.

2.7 Runtime Analysis

This analysis is straightforward and does not take into consideration the many ways that the algorithm could be made more efficient in practice such as both parallel computation (possible with classical and actual quantum systems) and data partition strategies that reduce the number of structures that would be compared with each other.

Since X structures cannot have more (width) than 5 elements, we can have no more than n^5 distinct X structures overall. If we compare each X structure to every other X structure during first-generation we get n^5 by n^5 total comparisons giving us a total of n^{10} comparisons. All modifications are constant time and so our runtime cannot exceed that needed to make individual comparisons, giving us a maximum of n^{10} total number of operations.

For second generation, if we use only the recommended basis set, similarly, we get no more than n^3 possible X structures. Here, we do not compare structures but instead loop over their collection for a possible maximum of n times to derive a solution. This gives us a possibility for no more than n^4 possible operations.

Considering the totality of all possible operations, we can see that our algorithm has a maximal runtime complexity of $O(n^{10})$.

3. Mathematical Analogs

In this section we briefly explore some mathematical structures that bear a close analogy to structures used by our algorithm. The hope is that mathematicians and computer scientists may be able to use some of the informal insights we provide to address similar problems in their different research areas.

3.1 Isomorphism Space of Proofs and Programs

All structures that we employ can be seen as both program and proof "types" that extend classical propositions, which themselves become a programmatic type in our treatment.

Since programs always declare only "rules" and "facts" which change by quantum differences, they can be viewed as quantum logical declarations (declared quantum logic programs) with the control part simply being the path taken (which would be the "goal" or desired final proof). They can also be viewed as functional programs. All programs can be seen as directly encoding their own continuation form as modifiable data for other programs. This gives a total structure where all programs are dually complex codata (co-compositional) and as well as complex corecursive (co-functional).

3.2 Algebraic Variation

In addition to just looking at structures as algebraic datatypes with pattern matched constructors, one can also view an original SAT formula as an equation for which our algorithm supplies a complex polynomial equivalence relation (which can be seen as more complex than being simply equational) where knowns are the initial clauses and the unknowns are the possibility of a solution and where solutions exist, what they are.

In addition, there is also the notion of a dynamical polynomial in the case where we have multiple solutions and supply, in time, partial or “infinitesimal” knowns which differentiate the space. The total set of solutions at any point in this sense can then be seen as a dynamically algebraic integral/antiderivative unless we somehow know beforehand what the exact knowns will be, in which case we can drop the dynamical qualification. In standard algebraic language, we can term all intermediate solution as complex roots and converted X structures as meromorphic poles of the satisfying function (in the case of single solutions, all structures can be considered completely holomorphic). All structures along the path of a Z structure completion (including the Z structure itself) would also be the complex coefficients of the completed Z structure.

One can also think of all structures as mathematical functions with assignments (solutions) as constant functions and changes over structures and assignments as function composition varieties which are isotropic algebras (provide composition tropes). The composition algebra here has a direct representation in the form of the functions being operated over.

In the case that one takes this view of compositional variation, one can regard the type of space being changed as the type of composition variety: projective for X structures, affine (parallel connection over solution species) for Y structures and smooth for Z structures. If we regard the varieties as algebraic objects (objects represented by the spaces they take up), the entire algorithm itself can be viewed as a polynomial functor over morphisms between algebraic objects.

3.3 Algebro-Topological Space

Y structures with more than one solution directly carry the structure of complex, differentiable 3 (chart size) + 1 (time dimension) manifolds (where X structures are charts which themselves have complex differentiable manifold structure in time) and Z structures carry the structure of real (rationally) differentiable manifolds which differentiate themselves (carry their own transition maps) and also differentiate both X and Y manifolds. Y manifolds can be seen as the algebraic union of X manifolds and also as carrying a closed boundary over other Y submanifolds and Z manifolds of open boundary (an implicit cobordism). Z manifolds in this case can be considered bounded by a somewhat different n+1-manifold version (not the 3-fold made up of X charts) of the Y structure with an implicitly (algebraically) represented chart structure.

In this case we have associate real objects with lines or rational composition along a line (as in the usual case) and complex objects with objects capable of splitting into two or more complex or real objects. This gives Y the informal structure of a complex topology and gives X structures the role of structural sheaves over open Z subsets.

In terms of homology, one can think of Y manifolds in this case as computing a “3-sided” algebraic cohomology product – classifying holes in X manifolds, themselves and Z manifolds.

In general, the manifold abstraction is a useful one as it has physical interpretations via connections with fibrations over locally Euclidean points. These points in our case are not strictly vectors but carry what we call a “spin” form that have a general direction, relative magnitude and for both free variables and 2 element and higher structures, a spin axis which provides a complex coefficient of scale over spins into spaces of lower unit dimensions, like tensors over vectors.

We can consider the operations over these spin forms as algebraic: X structure combinations act like vector addition. Y structures act like complex scalar comultiplications, assignments act like scalar division/quotient in the direction of X manifolds and scalar multiplications in the direction of Z manifolds. This gives a structure that can be called complex coalgebraic.

The superposition of Z manifolds can also be viewed abstractly as acting like a vector space and in this sense, one can regard assignments as deformed (quantum) groupoid elements (in that not all assignments may have an inverse) and the structure of smooth assignability being the groupoid action that computes the individual quantum vectors. Another possibility is to regard first-generations as acting like rings (based on types of available operations) and together with second-generations form an additional structure of an abstract field with regards to the vectors (keeping quotients as divisions and with selecting a particular Z manifold as subtracting out the other possibilities). All of these, as mentioned, is just an abstract variation on already abstract objects.

Lastly, and probably of importance to physicists and quantum programmers who work with Bloch spheres, our manifold representation acts like a kind of Super-Riemann sphere with X structures acting as the complex numbers and their uniting Y structure acting as the point at infinity. Our proof of the algorithm also shows that this sphere has the desired positive definite surface structure (surfaces being the progression of Z structures).

3.3 Variations Over Number Fields

As discussed earlier, we can associate the functions in our space to real and complex forms and by extension, real and complex numbers (where the imaginary portions would be the as of yet undecided modifications on structures). All numbers in this sense can be seen as "Gaussian" because they can be assigned positive integer values.

Since structures act like manifolds and Z structures can be assigned integer values, one can speak of the curvature that computes a Z number. We may not be able to give the exact mathematical character of these curves, but we can consider them as configured over a space in the negative (hyperbolic) direction to the Z number axis. These "hyperbolic" configuration spaces themselves can be said to possess an ellipticity (periodic in two or more directions) in their own positive direction.

One can as well ask about the existence and distribution of prime curves (curves with just one associated number) over the field of all possible SAT instances of a given size n similar to say an algebraic formulation of the Riemann hypothesis.

3.4 Abstract Knots

This would be the most abstract interpretation we think possible: X structures as knotted "membranes", Y structures as vacuous links that connect them into a unified topology and rewrites of X structures as fibrational unknots that cross over into lower dimensional knots or a base (0 dimensional) Z string space. We use the word membrane in a loose sense both to mean a structure that can "discharge" string "potentials" across a complex "organic" space as well in the "brane" sense used in physical String Theory (and this gives us a nice segue into the physics section).

4. Physical Analogs

Like the mathematics section, we hope the speculative analogies here can help physicists answer/investigate more concrete questions and issues in their field.

4.1 Abstract Hilbert Space

Hilbert spaces should normally belong to the section on mathematical analogs except here we only provide computational analogs of numeric distances and angles that may or may not meet the full mathematical definition required by the reader. The analogy though, we think, represents in character, the form of Hilbert spaces able to characterize some abstract physical space. Also, our discussion of this type of space may be the most useful to physicists here as our other analogs are bit more speculative/imaginative.

Say for n variables, the total exponential space of all possible assignments represents an abstract starting point for all other possible subsets of possible assignments. First-generation completed Y structures (computing one of these possible subsets) would be a differentiation from this ground space, that is, they have already diverged (angle) and can be given a numeric value (distance) with respect to all other possible Y structures. In addition, Y structures also represent an algorithmic convergence. Y structures themselves could then be regarded as the inner product over which outer products, that is, further divergences (Z structures with added angle and distance) would be computed.

4.2 Abstractly Classical Quantum Computation

All structures over which we compute can be viewed as “waving” over their respective local spaces. All such computations can then be seen as taking place over the wave representation of collections of base signal forms where our programmatic transformation of these forms align their interference patterns constructively. Y structures can be seen as total propagation mediums where X structures provide a convolutional (reverse shifted) background for the autocorrelated (of the Y structure signal with itself) transport of Z structures (and implicitly associated X structures as well as the Y structure itself). All structures/signals in this case, like true quantum computational entities, are reversible and all signal changes could be considered “quantum relativistic” phase changes.

Also, since we can run second-generations over entities of no more than size 3, we have an abstract analog of a 3 spatial + 1 temporal dimension spacetime similar to standard quantum computation. What we would have given this analogy then, in addition to normal spacetime, is access to the higher dimensional spacetime that topologically guards our computations. We can then regard all spacetimes as derivable from a singly total algebraic continuum of homeomorphic (functional) string-like elements.

4.3 Quantum Gauge Theories

We have touched on the structures we generate as having wavelike properties in the previous analog. We can generalize this to saying that these structures act like wave equations of a field theory where the equation components are action gauges “over” their own actions (making them true or pseudo-particles) as well as the solutions of the total equation system (particles). Y structures would be the system over partial X differential equations and Z structures would be solutions (particles). X structures can also be regarded as individual quantum fields themselves and Y structures as their unification.

One can then ask – field theory of what? And that would depend on the observables, that is, it would be possible to represent any kind of field one wants. Generally, one can note that structures force assignments so can generally be regarded as "force" fields and their union as force fields.

Gravity stands out here, in a relativistic form.

In the case of Y structures with more than one solution, we can think of a modification of the underlying structures during an assignment as a translation of coordinates of the entire Y structure along its axis, given the structure of a force that determines distance.

Another way that we can view Y structures as a gravitational field is that if the set of gauges are seen as physical, we can think of Y structures as some kind of mass density (configured in spacetime units), X structures as gravitational bosons that gauge the geodesic path (Z structure) of some standard object (say abstract photons or quarks) around the field of the Y structure. In this case, one can think of the particular arrangement of assignments s the "real" displacement along the spacetime neighborhood of the associated mass. The entire 5 + 1 (time) dimensional support manifold in this case would encode some kind of supergravity over all possible spacetime paths.

One can also think of all fields and particles in general as self-gauged entities, where the variables are all the sets of quantum properties that can be observed about the particle/field as it moves around some space or interacts with some other arbitrary field/particle with which it shares some or all of the sets of observable properties.

4.4 N Body Orbits

Assume the case that all literals are invertible where opposite literals stand for opposite points on a variable's revolution around a center of gravity (which may just be the pooled gravity of all the variables). We can then continuously move variables around (algebraic but probably unphysical) orbits.

4.5 Absorption/Emission Spectrum

One can also think of all freely assigned variables (general inputs) as absorptions and those forced at boundaries as emissions.

4.6 Torsional Flow

One can think of Y structures as mediums over which solutions flow and are sped up (many resulting forced assignments) or slowed down (fewer forced assignments) as X structure crossings provide the necessary resistance/impedance/torsion points. This would require seeing the X structures as somewhat sequential as opposed to parallel (which would be abstract).

4.7 Abstract Cosmological Evolution

We work this analogy backwards as it is a bit even more abstract than the rest. Here we explicitly assume that we always proceed to second-generation stages.

Say solutions (Z structures) are the real material structures (having mass, energy and constant motion), the basis set of Y structures in this case would be the cosmic “gauges” out of which these structures emerge. Each differentiable Y basis set would be the isotropy that guides the formation of a family of structures and could be said to be the field out of which their masses form. As time evolves, anisotropic sections would drift further from each other in cosmic spacetime. Non basis and higher sized (4 and 5 element) X structures would then serve the function of either massless or unseen (not interacted with) dark masses. First-generation itself would be a sort of scattering/configurational energy (dark) that did not interact directly with Z structures and the initial clause itself would be the progenitor of all structures that emerged from the vacuous creative hole (cosmological vacuum). The whole set of structures together can then be seen as living in some sealed off spacetime (black hole) unless of course some authorial figure could add and delete clauses from time to time., which would make the entire set like a space of learnt functions that say, extend the cognitive work of the authorial agent (like a proof assistant). If two such agents happen to interact (partially or totally over their respective Y structures) they could share learning artifacts via clause/function exchange/insertion/deletion/modification and so on.

5. Conclusion

We have introduced a quantum algorithm that runs on classical computers, a first from what we know. This should prove useful to mathematicians and physicists who aim to study quantum systems and their related mathematical structures and properties. We feel that the greatest contribution we would be able to make in addition to being able to solve the class of problems currently known as NP in the computing literature efficiently (and we hope to have successfully proved is P is equal to NP) is in the scalable and iterative modelling of quantum systems.

We conjecture that problems with existing rational order – mathematical and physical, would be particularly tractable using our algorithm as they would already have some intrinsic quantum computational/algorithmic patterns embedded in their structure. In particular, this should help us better understand quantum systems and their efficient design and improvement. We hope that this spurs a new era in work on both quantum and classical algorithms and systems as we better understand their properties and possibilities on all possible computing platforms.