Holography For Satisfiability

Adewale Oluwasanmi
waleoluwasanmi@yahoo.com

ABSTRACT: We present a discrete differential Nullstellensatz style result for finding satisfiability/unsatisfiability proofs to any 3 SAT formula, along with novel upper bound results. We begin by defining a notion of holographic algorithms sourced mainly from Valiant [1] and a topological interpretation drawn from the holographic principle [2] in physics. In our interpretation, holographic instructions (and their accumulating sequences) live as points on a differential manifold and are (partial with progression towards totality) solutions to the 3 SAT problem expressed in differential form. These forms can be expressed as (implicit) polynomials generated from ring expansions of an algebraic identity which Valiant says holographic algorithms must follow - a definition of algorithms with algorithmic identity based on an algebraic quantity (usually as sets of zeroes of some polynomial), instead of traditionally as a sequence of instructions. This definitional model which is summarized semantically in the following formula for some problem **p**, forgetting the underlying model used to obtain the quantity is:

$$\sum(\text{sets of solutions/zeroes of } \mathbf{p}) \prod(\text{functional constraints on } \mathbf{p})$$

Our result is that, if any solutions exist for a formula $\mathbf{F}$ recast into an initial polynomial $\mathbf{P(x)}$ in some differential ring $\mathbf{PR\{x\}}$, defined by the above identity, then they are multiplicative set extensions of the minimal solutions that our framework provides (they are values in a differentially closed field) and that these values are exactly the zeros of ideals in the ring of differential polynomials generated using $\mathbf{P(x)}$ reflexively itself as a basis.

Restated, our main theorem says that there exists an integration procedure for computing a differential closure as a prime model $\mathbf{M}$, consisting of a single ideal, using $\mathbf{P(x)}$ as a basis, and when $\mathbf{F}$ is satisfiable, $\mathbf{M}$ is either a satisfying saturation model **OR** there exists satisfying saturation models derivable from $\mathbf{M}$, where derivations are multiplicative interpretations, that is for natural numbers $\mathbf{i}$, $\mathbf{n}$ and $\mathbf{k}$ and a set of inducible interpretations $\mathbf{S}$ where all $\mathbf{f}$ are interpretations in $\mathbf{S}$, $\mathbf{t}$ is the continuous time variable and $\mathbf{X}$ is the set of all satisfying solutions:

$$\forall F, \exists M \Rightarrow \forall f, (n, f) \in S \wedge \partial^k f/\partial^k t: M^k \rightarrow X, \Rightarrow M = \int P(x)\partial^i x \partial^i t, \text{ iff } F \text{ is satisfiable.}$$

The differential ideal generated, or more specifically, its field of coeffficients, functions logically as an **assignability** predicate over solutions in the original formula, and in line with Tarski, our procedure provides an implicit model of truth over these predicates not directly expressible within the logic exposed by the predicates themselves. This procedure functions to yield an operator valued second order logic in line with Fagin's theorem, capable of exactly describing the (holographic) algorithms that prove the 3 SAT formulae and therefore can be extended as an efficient logic over the whole **NP** class.

1. Introduction

Before looking in detail at the differential parts of our theory, it is useful to first examine its purely logical and model theoretic aspects.

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. The full classification of the NP space as well as the question of whether P = NP had the earliest developments in the combined works of Cook [3], Levin [4] and Karp [5].

A brief description of the P vs NP problem is that it asks, if every problem whose solution can be recognized in polynomial time by a non-deterministic algorithm can also have the same solution generated in polynomial time (by a deterministic algorithm).

In [3], solutions are encoded as language statements and generation/recognition are treated uniformly as co-acceptance procedures (recognizers are information symmetric to generators). We however propose that the distinction between generation and recognition highlighted in our definition and in the holographic literature already gives us a way to represent these 2 acceptance procedures algebraically in the form of matchgates [6, 7] with language generators nested on the inside (initial endpoint) of a structure that has language recognizers on the outside (final endpoints) and a type of cancelation algebra between paths as we move from the inside to the outside of such a structure (whether such an algebra be polynomial or non-polynomial being of not much relevance at this point).

Valiant is then right, in posing what we call the Holographic Conjecture: That any positive resolution to the P vs NP question would need to address the issue of asymmetry/limits of symmetry between the generation and recognition procedures for this problem, and why the problem cannot be solved efficiently using a holographic structure such as a match gate, a conjecture which we also address here.

In their traditional form, satisfiability clauses take the form of bracketed disjunctions over some subscripted variables, for example:

$$(\mathbf{x}_1 \vee \mathbf{x}_2 \vee \mathbf{x}_3 \vee \mathbf{x}_4 \vee \ldots \vee \mathbf{x}_n)$$

up to any number of **n** variables.

Satisfiability formulae join these clauses by conjunction and allow variables and their negations to be repeated in separate clauses, up to any number of clauses, for example:

$$(\mathbf{x}_1 \vee \neg\mathbf{x}_2 \vee \neg\mathbf{x}_3 \vee \mathbf{x}_4 \vee \ldots \vee \neg\mathbf{x}_n) \wedge (\neg\mathbf{x}_1 \vee \mathbf{x}_2 \vee \neg\mathbf{x}_3 \vee \mathbf{x}_4 \vee \ldots \vee \mathbf{x}_n) \wedge \ldots$$
$$\wedge (\neg\mathbf{x}_1 \vee \mathbf{x}_2 \vee \mathbf{x}_3 \vee \mathbf{x}_4 \vee \ldots \vee \neg\mathbf{x}_n) \wedge (\mathbf{x}_1 \vee \mathbf{x}_2 \vee \neg\mathbf{x}_3 \vee \neg\mathbf{x}_4 \vee \ldots \vee \mathbf{x}_n)$$

Satisfiability problems in this form then ask for a satisfying model, in this case, a set of assignments to the unit variables in the formula's clauses that make all the enclosed clauses true. This satisfying assignment can be considered a "flat" linear model (a model that has a single, completed interpretation).

One natural question to ask at this point is if higher dimensional models may exist that are asymptotically reducible to flat models via interpretations (strings of variable assignments) should the formula be satisfiable, and our answer to this question is yes, a holographic model, which we describe formally. This model is exactly the prime model, that is, the model detailing the least type, which is the differential closure of a derivation procedure over some field K.

From a model theoretic perspective then, first, we assume that the Boolean satisfiability problem has been reduced into its 3-SAT form which is necessary for our procedure to work.

From this, we reduce the problem into a Holant[1] inspired differential form, that is, a form that allows a certain holographic counting formulation (in terms of zeroes of polynomials) as prescribed by Valiant [1] to be applied to the problem. This form is structured so that:

All possible satisfying solutions, at any point in time, count as (assignable) solution sets and satisfy this part of the Holant formula:

$$\sum\textbf{(sets of solutions of p)}$$

Actual assignments count as the constraints that "flatten" the higher dimensional model to select out a subset (and eventually just one) of the satisfying assignments to the CNF formula and satisfy this part of the Holant formula:

$$\prod\textbf{(functional constraints on p)}$$

It turns out that it is indeed difficult to get directly at this positive model, where we assign satisfying values directly to variables, that is, we have no concept of what variables are functionally assignable (or if any valid assignments exist at all) initially. To continue, we have to further borrow from Tarski, the idea of an inductive schema. This leads us to rephrase Tarski's idea in terms of assignability as:

"We can obtain a model of assignability if we can obtain a model of unassignability"

or in more dynamic terms,

"A statement is assignable if and only if it is not unassignable at some given time t"

Thus, the notion of unassignability becomes dual and drives the logic for assignments holographically, and if we are still to adhere to Valiant's algebraic intuition on the clausal level, our main task now becomes to construct an algorithm:

"that induces a base model of unassignability AS A SUM OF IMPOSSIBLE SOLUTIONS, dual to a co-induced model of A SUM OF POSSIBLE SOLUTIONS, AT ALL TIMES, from which assignments which function as CONSTRAINTS can be combined to select one of the satisfiable solutions (contained in the sum of possibilities), where this final selection, considered indexed by the assignments, is the formal PRODUCT of applying the constraints."

Such an algorithm, if it exists, and we show that it does, would be a meta-algorithm as per Tarski. Such an algorithm would also be a variety generated by a second order differential polynomial ideal. The rest of the following sections will detail the differential aspects of our theory, particularly, how to generate the ideal if it exists via integration and how to extend the ideal to particular solutions via differentiation:

2. The Simplest Holography of A Single Witness
3. Towards Complex Topological Union: Bi-Local Linguistic Collisions.
4. The Global Union Operation.
5. Putting It All Together- Witness Completion and Algorithmic Analysis.

Concluding sections include:

6. Implications For Theories of Computational Complexity and Computability
7. Implications For Other Mathematical Fields.
8. Conclusions and Future Directions.

2. The Simplest Holography of A Single Witness.

2.1 Implicit Differential Polynomials and their Zeroes

We start by describing a simple algorithm that always satisfies a single clause. Firstly, note that it is a fundamental process axiom that for any standalone 3 SAT clause, there are always 7 ways of satisfying such a clause. In fact, for any standalone k-depth clause, there are exactly $2^k - 1$ satisfying solutions. This means that out of the 2^k ways of satisfying a clause, one of them is always initially unassignable.

First assume that assignments take place over the binary field:

$$[1, -1].$$

This means functions on the sets of zeroes of an ordinary algebraic polynomial that satisfy the clause take the form:

$$[1, -1] \rightarrow [1, -1]^k$$

Let our test integers come from some set represented by letters, that is, **[a … z]**, so that a positive **(+1)** truth value assigned to some random subscript can be represented by say the letter **a** and a negative (-1) by an oppositely signed letter **-a**.

Now assume the set of assignments **abc** assigns true to all the variable in some original formula, then the set of assignment **-a-b-c** assigns false to all the variables. This statement is unassignable. The singleton set containing just this one statement is our (starting) model of unassignability for a one clause formula.

Now, consider the below procedure executed against the unassignment model for the single clause. This procedure incorporates the notion of FREE and BOUND assignments. Free assignments are those in which at the time when a value is to be assigned to some unit variable, we have a free choice of assigning the variable to either a true or false value and a bound variable

is exactly the opposite scenario, where the framework rules force us to assign the variable to either one of these values:

i. Start by identifying the excluded solution for the given clause. Initially, this solution is of some length **x** where **x > 1**. **x = 3** in our case. For an example, using our generic notation, assume the initial excluded solution is **-a-b-c**.
ii. Assign values freely to unit variables one by one, in a loop.
iii. If the value freely assigned to any variable satisfies the original clause, STOP. For example, if in our starting example, we assign a value **a**, b, or **c**, we can stop.
iv. Otherwise, if the current length of the excluded solution is **GREATER THAN 1**, rewrite the excluded solution by requiring that the only combination of false assignments to the remaining variables be the new excluded solution, that is, the old, excluded solution with one less integer, the integer representing the unsatisfying assignment received. For example, if in our starting example, we first assign a value of **-a**, obviously the underlying clause is still unsatisfiable and so we rewrite the new excluded solution as **-b-c**. Consider this a strengthening (old value is added to the new one). This strengthening is now defined as an explicit **SUM** over unassignable statements.
v. Otherwise, if the current length of the excluded solution is EXACTLY 1, we are now **CONSTRAINED** to change the sign on the excluded solution and satisfy the underlying clause.

Readers familiar with topology will notice the following:

i. The procedure always returns some satisfying solution such that the set of all possible solutions looks like the closed sets of a connected space.
ii. The actual steps that select any solution subtracts or more precisely, quotients out the other solutions. This quotienting in a topological definition, covers the final selection and the subtracted sections together as points form an open set in relation to the final selection.

Now from the point of view of any final solution, we have a topology:

$$(X, \tau)$$

Where **X** is the (not directly countable) set of possible solutions and **τ** is the dynamic algorithmic topology on **X** generated by assignments (free and bound).

Now we go back to define **SUMS** (of solution sets) and **PRODUCTS** (of constraints) in the following topological way:

i. **Sums Of Solution Sets**: A topological sum is defined as the disjoint union of the underlying sets. The sum which we define in the above procedure (step iv) is such a sum because it includes a union of isolated/partitioned unassignable statements. The final value of this sum is always the selected solution and can be considered a sum over interval (unassignable statements) values. It is always dual and additively inverse to the sum of assignable statements.

ii. **Product of Constraints**: A product topology is defined as the coarsest topology on a space, which is the topology that preserves the fewest open sets on the space. Since any set of constraints composes a final solution which excludes all interval values and outputs the selected statement, the process applying the constraints is the same as taking a product over the set of intervals (excluding them). Using this perspective, we can consider the stages before a final selection as computing fractional products. Again, as above, the final value of this product is always the selected solution and can be considered a product over interval (unassignable statements) values.

These two definitions complete our requirements for the algebraic operations needed in a Holant form for a single clause. Since we formally sum over (fractional) products, we can formally describe the algorithm given above as the **HOLOGRAPHIC SUM OF PRODUCTS RULE** for a single clause.

This rule can be interpreted as the application of a derivation rule operator over some implicit differential polynomial ring G[X] in one variable with operational coefficients in the rings of plain polynomials (to be satisfied). Our polynomials would be structured as a product of monomials. Each monomial, mapping exactly to one of the unknowns, for some random variable x, can be written in the form:

$$\mathbf{(1 - x)} \text{ or } \mathbf{(1 + x)}$$

So that polynomials evaluate to zero when satisfied.

A monomial for which either value is unnecessary to the satisfaction of the clause/enclosing polynomial, takes the form:

$$\mathbf{(x^2)}$$

In the following sections, the reader will see how such monomials can be generated from the group evaluation/integration of all polynomials.

Now if we take it that rewriting is still implicitly taking place all the way until we get to a final solution, we can order a representation where these rewrites function as coefficients of our differential operators and the current depth of the operator functions as its exponent. In a future work in which we formalize the cohomology of our sheme, we will further describe these coefficient action as the modification of the time derivative of the differential operator and thus

the function itself (in a reflexive fashion). Our sequences of derivations now form a differential ring by the following reasoning:

i. Solutions form an implicit ring. We can form (polar) pointwise sums and products of the underlying statements.
ii. Solutions are one to one with derivations and so derivations also form a ring of the following type: $\sum f_i g_i$

Where **i** is an integer, $\mathbf{f_i}$ functions come from the ring of algebraic polynomials with monomials of the form **(1 - x), (1 + x)** or **(x^2),**

And $\mathbf{g_i}$ functions are of the form: $\mathbf{\partial^n g(x) / \partial^n t}$ and form the ring of differential operators**,** **n** is the current depth of the rewrite, and final solutions are considered flat, that is are constants having zero depth. We can observe that the time variable **t** unlike the other variables, does not have a discrete definition but emerges (pseudo) continuously from the process logic. This is what gives this variable the status of a continuous variable and lends a differential interpretation to our process.

Note that we have just expressed the schema that turns the points in the space into predicates over solutions like an algebraic scheme. Remember that schemes are ringed spaces that generalize varieties by introducing multiplicities and feature a commutative ring for every open set. Here we have a differential scheme.

2.2 The Fundamental Theorem

Topological algebras, as standardly constructed, define an algebra (a ring with an operator/ a module/vector space with a bilinear mapping) that is also topological space, such that the algebra operations defined are continuous functions over the space.

This definition requires that the operator on the algebra act like a multiplication (product) that linearly extends the multiplication operator on the underlying field, in our case, a ring. For this, the operator on the algebra would be the sum over unassignable statements, which is linear in both the variable holding the existing sum as well as in the new constraint and extends the product operation on the underlying ring.

We now redefine a discrete multiplicative schema as a topological space with rings for all open solutions (positive statements), where the solutions can be considered multiples of each other.

With this additional information, we can now formally describe our sum of products rule as "**a topological algebra acting on a discrete multiplicative schema**" and the holographic algorithm for a single clause as simply the product of this algebra in one multiplicative dimension.

For readers familiar with abstract algebras, one can easily see this is a chiral algebra. Chiral algebras display chirality (handedness – distinction between directions in spacetime) and are usually implemented in physics as vertex algebras – vertex expansions over an ordered structure (lattice). In our case, the vertex algebra described is quite similar (with a higher number than 2 of chiral parts) to the one formally espoused in [7], which formally utilizes an algebra over boundary modules of rings, as we do here (modules are implied from the gluing of spectra).

What we have accomplished here is to describe the process of one WITNESS (associated with a single clause), describing how a single clause may be satisfied from its (accumulated) statement.

Finally, in the rest of the paper, for purposes of easy separation and discussion of concepts, we will simply call elements (constraints/unassignable statements and their sums (of products) of the topology which are not actually satisfying assignments, ANTI-PRODUCTS. This marks them as elements of a product space that we are in some sense "multiplying/producing" away from and are exactly the categorical duals (hyper-duals in some sense) of actual products.

Now we state our first theorem using the language developed here and assumed to be proven:

Theorem 1: *Any single clause C can be transformed into a differential polynomial D with solutions in a differentially closed field F, where D is an ideal in the differential polynomial ring $K[X]$ whose elements vanish on the minimal solution of D and its expansions (subsets).*

Corollary 1: *Any single clause C can be transformed into a differential polynomial D with solutions in a differentially closed field F and any polynomial with a solution in F can be derived from D, that is, all solutions to C are solutions to some polynomial derivable from D.*

2.3 The Global Topological Conjecture:

Having defined a holographic algorithm and thus the theorem for a single clause, we will now attempt to extend our definition to a formula of several clauses, approaching the definition of a true nullstellensatz.

First, we ask if we can extend the topological algebra of the Sum of Products rule to include an arbitrary number of clauses (product dimensions), that is, if we can apply this rule simultaneously to a set of satisfiable solution spaces, so that their products align over all possible variable assignment sets, for one or more satisfying solutions (ideally, all solutions), essentially forming a single coordinated product? If so, we can conjecture the following:

*"For any **n** number of clauses, in a given formula **F** with **n** clauses, if we convert **F** into holographic form of initial unassignable sums, there is a procedure **P**, which turns **F** into an **n** dimensional (multiplicative) schema over the whole formula, if the initial formula is satisfiable, and a zero schema (no reachable open solutions) when it is not".*

Note that this procedure **P**, is not the (local) topological algebra itself, but another procedure (considered an extension) over whose output the topological algebra can be applied. In our case, for those more familiar with topology, **P** can easily be shown to be an extended version of the topological algebra which operates over the exterior of the topology over which our regular topological algebra operates on the boundary.

Our goal now becomes to find this procedure **P**, if it exists. If **P** exists, we have a traversable topology that encodes satisfying solutions. That **P** exists for all satisfiable formula is a conjecture that there exists a globally discrete multiplicative schema as well as a global solution

topology on the associated formula. If such a **P** exists, then there must exist witness procedures for verifying its steps.

The task of proving this conjecture is tantamount to prescribing a process for topological union between the product spaces of individual clauses as well as to provide a set of witnesses that can use this process to describe satisfying holographic algorithms for satisfiable formulae.

If we can achieve the above stated goal, then we can achieve the full description [1] of, holography given by Valiant as the "mapping of solution fragments, many to many, while preserving the patterns of interference among them". We are encouraged that this is possible because of 2 reasons:

i. We have a sum of products formula which is Holant/Holographic and well defined in one dimension and in the first order according to [1].
ii. The String theoretic holographic principle [2] tells us that a description of interactions between elements on the topological boundary of a region of interaction should equate to a description in the interior of the same space for an arbitrary number of elements. Considering the orthogonal single nature of interacting elements in a single clause, this condition is trivially satisfied for what one may call, first order interactions. One can imagine this interaction taking place within a second order across clauses.

The fact that two these theories of holography, developed in two different contexts, can be made to trivially agree, albeit, on a very restricted case in a first order sense gives us a hint of possibility that they might be made to agree over a second order with some careful algebraic analysis.

Achieving the above stated goal would yield the following theorem and its natural corollary that extend the one clause case.

Theorem 2: *Any set of clauses CA can be transformed into a differential polynomial D with solutions in a differentially closed field F, where D is an ideal in the polynomial ring $K[X]$ whose elements vanish on the minimal solution of D and its expansions (subsets).*

Corollary 2: *Any set of clauses CA can be transformed into a differential polynomial D with solutions in a differentially closed field F and any polynomial with a solution in F can be derived from D, that is, all solutions to the conjunction of the clauses in CA are solutions to some polynomial derivable from D.*

This is our nullstellensatz and proving this main theorem is the focus of the following 3 sections.

3. Towards Complex Topological Union: Bi-Local Linguistic Collisions:

Here we attempt to satisfy the global topological conjecture in a limited way for a formula consisting of just 2 clauses by looking at the logical content of their underlying linguistic schemes, that is, the set of current unassignable statements (expressed as sums of products on the purely algebraic side).

Our next consideration would be to look at updating information about the common scheme in order to "unify" their satisfaction algorithms and thus their topologies. This strategy uses the linguistic structure of the schema to modify the algebraic ones of the scheme.

Consider the following case:

> We have 2 sums (of unassignable statements), **A** of depth **m** and **B** of depth **n** in a formula **F**. Assume there is exactly one and only one term in **A** that that has the opposite sign of a term in **B**. Let us call term in **A**, **x** and the term in **B**, **-x**.
>
> Now assume, at some time **t**, we have assigned the underlying clause to other integers, but we have not assigned a value **x** or **-x** to either clause, that is, exactly at time **t**, the only values we have assigned to each variable are exactly the unsatisfying values, that is, the exact integers specified in the unassignable statement
>
> *For example, say **A** initially (at time a **t1** = **0**) has value **125** and **B** has value **34-5** (at time a **t1** = **0**) in a formula **F** and at time **t2** (**t2** > **0**) we have assigned **F** to **1234**, reducing **A** to **5** and **B** to **-5**.*
>
> Since the original clauses remain unsatisfied by the current set of assignments and the two remaining terms are conflicting, we cannot satisfy both clauses at the same time (as required), since by the natural law of assignment, any variable can only either be assigned to an integer or its negation.
> Therefore, the combination of hypothetically assigned terms (as demonstrated in our example) is **GLOBALLY UNASSIGNABLE** in any scheme of satisfaction including both clauses and is a common anti-product for both clauses.

To prevent the situation above from occurring, we must correct the formula F at some time **t** before **t1**. In fact, we need to correct these sums as soon as we can compare the two offending sums, that is, at time **t = 0** relative to both sums. We need to combine all the non **x** terms in both sums into a new sum **S** that becomes the unassignable sum for a new induced clause **C**, which we have just added to the formula. **S** now becomes unassignable in our model.

> *For example, say **A** initially (at time a **t1** = **0**) has value **125** and **B** has value **34-5** (at time a **t1** = **0**) in a formula **F** and at time **t2** (**t2** > **0**) we have to add a new unassignable sum **1234** to our model of unassignable sums. This is because **1234** does not satisfy either clause because of the conflict on **5** and **-5**.*

*Let us further assume that if more than one term in **A** conflict with (an equal number of) terms in **B**, our rule does not apply. This is simply because we can assume (for now, and to be proven) that no assignment process **AT ANY TIME**, would produce a string containing a previous term **X** and its negation **-X** upon arriving at point where we need to assign another variable to another term (**Y** or -**Y**). If indeed, we never run into this situation (as the reader should by the end of article, see that we never will), we can assume this assumption invalid.*

This gives us the flowing bi-local process axiom:

***Axiom 1**: Given 2 clauses A of depth m and B of depth n where exactly one term in A's anti-product conflicts with one term in B's anti-product, the common anti-product is a new term combining the other non-conflicting terms in both anti-products into a new single anti-product.*

One thing to note, is that if A and B are the same depth, and all the other non-conflicting terms are exactly the same, the resulting anti-product replaces the A and B as a single larger (shorter length) anti-product, otherwise, we will always get an additional, smaller (longer length) anti-product.

We will henceforth refer to the operation that writes out a new unassignable model as a result of the assignability evaluation performed, a **DIRECT SUM OPERATOR** or simply, a **DIRECT SUM** of models (viewed as topologies). This is taken from the direct sum concept in abstract algebra which pairs structures so that their underlying operations are compatible between pairs (or collections in our more abstract interpretation) of their individual elements. This direct sum can be considered to be the exact dual of the sum of products rule for every clause (which we can now interpret as a direct product).

Note: One can also look at this as a **DIRECT PRODUCT** of algebras, turning 2 different algebras into the same algebra, similar to group direct products.

This operator can be seen as the integral dual of the differential derivative that computes actual solutions and is a kind of differential form in the following way(now with time extending instead of quotienting and without counting the multiplicities of the derivation operator). **y** and **x** are variable with values in the field of solutions and **t** is the time variable:

$$\int \mathrm{F(x)F(y)} \partial \mathrm{x} \partial \mathrm{y} \partial \mathrm{t}$$

The use of the Dolbeault operator here suggests that our integration is somewhat infinitesimal and considers exact values on boundaries. The open integration is similar to the use of the symbol in statistic where the computational semantics of the operator (in our case, a predicate), confines the values of the integration sample. It should be apparent that for any two clauses whose topologies are united this way, our topological conjecture, and thus a version of

theorem 2, does indeed hold for the formula (containing just the 2 clauses). The set of resulting reductions is indeed a prime model and an ideal in the differential ring. That is, we can trivially select any and all valid assignment products using the procedure of assigning values to the enclosing formula.

4. The Global Union Operation.

So far, we have shown how the topological conjecture holds in the case of any two independent clauses by describing how the common anti-product base of two arbitrary clauses can be schematically unified into a single differential ideal. That is, the satisfaction of a formula of 2 clauses is completely witnessed by our model.

This binary bi-local operation described in the above section will represent the basic operation of topological union which we REPEAT over all possible, relevant pairs in some set **S** (which we allow to grow by addition of new elements until we reach some exhaustive condition to be defined).

Now since we know **S** has the ability to grow (from examining the bi-local operator), we now ask the proper model theoretic question: what controls the maximal size of **S**, so that we know when to stop the growth of **S** so we can run the sum of products rule to successful completion on every clause, thereby satisfying it (that is when does **S** develop stable algebraic identities)? We are essentially asking for an equivalent of a Grobner bases that is exact and applies to a differential instead of just an algebraic ring, which defines an ideal in the manner that would generalize Theorem 2 to an (potentially) infinite number of clauses, that is, an inductive rule.

The first stage – verifying that **S** is stable (according to some criteria **C**))and then proceeding to satisfy the formula) is exactly the **P** procedure we asked for in our topological conjecture.

Proceeding in the second stage towards successful satisfaction is what we will call a complete **random topological walk** (this assumes **S** has more than one solution, an assumption which makes the solution easier to demonstrate).

In essence, what we ask is this: What are the walking boundaries of the assignment operator that we associate with the sum of products rule? We can answer this by asking, what is **C**, the halting criteria for guaranteeing that the application of the rule will work when given a satisfiable formula?

We realize that **C** can be treated as some kind of inductive limit (where we have accumulated enough observations to deduce a general principle) placed on some structure and that when this limit is reached successfully, we know the underlying formula is satisfiable. But what may this structure be?

i. The set itself? – This limit can be set by detecting a common condition among the clauses that indicates that the procedure is now complete.
ii. The depths of sums (witness statements) in the set? – This assumes that we will not need more than sums of a certain depth to detect when a formula is satisfiable.

Now, let us incorporate the idea of such a fixed set and ask the following question about its computability:

> *"Take some SATISFIABLE arbitrary CNF formula **F,** with a set of (original) clauses **C**. Let **CX** at all times be the set of sums associated to the original clauses **C**, regardless of their current values. Can we compute some fixed set **CP** of unassignable sums (anti-products) which extends **CX**, in which all member sums have been resolved bi-locally with every other (relevant) member sum, such that no new sums need to be added to the set, except the modification (expansion) of existing sums of the subset **CX** into full products via (all possible satisfying) sets of assignments, using a unified topology?"*

We end this section by assuming that such a set, **CP,** which is a domain specific differential ideal, can be constructed for any arbitrary satisfiable formula **F** and in the next section, we "reverse-engineer" this set to discover what its properties should be through a novel form of algorithmic analysis. In this reverse-engineered scenario, we must demand that all algorithmic moves be witnessed. Being witnessed implies the following:

i. The assignment value is free – at the time just before we make the assignment, it is possible to assign its negation.
ii. **OR** The assignment value is bound - An assignment value is bound whenever its negation is blocked. We introduced a simple blocked scenario in our section on the simplest holography where we assigned a variable to a value when our unassignable statement has reached a depth of 1 and we are constrained to assign its negation. In the analysis section below, we introduce additional scenarios in which a variable can become blocked. Being witnessed now means that we can guarantee that no assignment procedure **P** will ever be blocked on both an assignment value and its negation AND that P. will satisfy the underlying formula.

5. Putting It All Together- Witness Completion And Algorithmic Analysis.

We concluded the last section by assuming that some stable inductive ideal set **CP** exists (generated by **CX**) and can be computed for every satisfiable formula **F**, after which we can successfully generate a successful assignment to the formula.

Generating a successful assignment to a formula **F** essentially means the following:

i. At no point during the walk should we be blocked on assignment to a variable. This means any variable should always be assignable to either true or false.
ii. We should include a procedure, that at any point **t1**, can check if assigning any variable **x** to some truth value **a** would cause some other variable **y** to become blocked, if so, assign **x** to **-a**, we call this **UNBLOCK** rule. The success of the **UNBLOCK** rule should always be guaranteed by the presence of some higher order sum in **CP**. Indeed, this is the absolute test of the set **CP** as a witnessing

set. Also, note that whenever we unblock a variable, the assignment to the variable always satisfies the underlying clause(s).

iii. At no point **t2**, if indeed we have such a set **CP**, should the procedure in **ii** indicate that we are blocked on both possible assignments to the variable **x**.

In essence, we will eventually describe a procedure **P** that uses **CP** to witness/verify that it can complete a successful assignment to a satisfiable formula using both free and bound assignments as well as the **UNBLOCK** rule.

To correctly determine what **CP** should then be, we first need to identify every case in which a topological walk can become blocked, then include a refutation that such a case would never exist if a certain type of **CX witness** (thus fixing what properties **CX witnesses** should have) or that we have a sum (a supporting **witness**) included in **CP,** a **CP witness,** that can be used to justify the use of the **UNBLOCK** rule.

If indeed this procedure exists, which encodes such witnesses, then we can guarantee that we can generate at least one random satisfying assignment to any satisfiable **F** using our proposed algorithm, that is we can verify that **CP** is indeed a differential ideal for some satisfiable form of our clause. From then we can show that **CP** is indeed a complete/total ideal on which every satisfiable polynomial vanishes.

5. 1 Obstruction Analysis

In this section, for ease of initial analysis, we assume that our formula is multiply satisfiable (by more than one solution) and that we have generated enough witness sums (in **CP**) to conclude that this indeed so. Our goal here then is to analyze ALL potential obstructions to satisfaction procedure, if such a set **CP** exists. The criteria of such a set would then be that for any kind of potential obstruction to assignment completion that we can think of, we can also provide a refutation for satisfiable formulae – that either the scenario is impossible or that there is a witness procedure which ensures that the obstruction can be remediated according to an unblock procedure which we define. This refutation squarely rests on the fact that the set **CP** *contains enough witnesses to verify that the obstruction is impossible or can be removed* by a procedure compatible with the topological algebra. Overall, we define 4 distinct types of witnesses to be identified:

immediate witnesses*, of which there are two kinds,* ***direct*** *and* ***indirect****.*
intermediate witnesses *which are always indirect.*
boundary witnesses *which are always indirect.*

The act of analyzing obstructions on a space (manifold – which we have here) that proceeds linearly can be seen as a type of homological algebra for mathematically inclined readers. Here is where our choice of the use of schemes will be seen as particularly ingenious in the way it allows us to both blend topology and abstract homology (as cohomology in our case) in proving our main conjectures.

Remember that our core problem is to prove the satisfiability of the main formula, that is, the conjunction over the original clauses. Also, remember that we required our initial problem to be in 3 SAT format. This means that our satisfying procedure can run the **SUM OF PRODUCTS,** rule, that is, the differential derivation procedure on only sums of depth 3 and smaller to confirm satisfiability of the main formula **F**. Higher dimensional sums will only be used as background witnesses (as we will show) to show that our ideal is sufficient. In terms of our underlying topological algebra, this means that we can use just 3- depth sums and less as the kernel of sums for our algebraic morphisms.

We break our analyses down by highly structured uses cases organized in order of structural complexity. We determine this complexity first by sum depth, deeper sums are more complex than shallower sums, as well as structured combinatorial complexity. These are organized into 3 main Use Cases (UC) that cover every *abstractly* computable satisfying walk pattern and number 29 in total. ***These cases will gradually introduce our cast of necessary witnesses***, until we have identified every possible type of witness we will need to complete our model. The depth of the sums in each case we will consider will be assumed to be the depth that they have when we have determined that we have computed all the initial sums we need in order to start assigning variables to values. This means that after we obtain these sums, there will be no further, direct bi-local interactions between unassignable sums – we have a complete starting holographic algorithm.:

i. **<u>UC 1 - Sums of depth 1</u>**

For example, an original **n**-depth sum reduced down to a single unsatisfying variable assignment, whose negation must be assigned to satisfy the underlying clause. This case is trivial since we can recognize that these sums indeed would represent the limit of sum taking and on a propositional algebra level would themselves be unsatisfiable assignments. That such are not negated by any other sum of depth 1 would represent the weakest possible preconditions for satisfying the associated formula. This can be considered an implicit axiom. If no such conflicts occur, the satisfying assignments to the individual clauses would be the propositional negation of the (non-conflicting) unsatisfiable sums.
Refutation: It is impossible to block on sums of depth 1 in a satisfiable formula. If we get two sums **a** and **-a** at any point during bi-local comparisons, we will regard the formula as unsatisfiable (more on this in our section on algorithmic analysis).

ii. **<u>UC 2 - Sums of depth 2</u>**

These can be immediately satisfied or reduced to a sum of depth 1 by some assignment if the associated assignment does not satisfy the clause.

We can identify 3 scenarios in which a variable assignment would cause the remaining formula to become unsatisfiable: Take **x, y, z** and **i** as integers and **-x, -y, -z** and **i** as their respective negations. The assignment of interest is to **x** (cancelling out **-x**). Each scenario will be followed by a refutation which affirms the impossibility of its existence from some supposedly complete process of enumerating the sums.

Scenario 1: There exists 2 sums **xy** and **x-y**. An assignment to **x** would leave two conflicting sums.
Refutation: Our bi-local operator would combine both clauses into a single **x** sum (making **x** unassignable) and so any set containing 2 such clauses will change and so is not complete.

Scenario 2: There exist 3 sums **xy, z-y** and **x-z**. An assignment to **x** would leave 2 sums **y** and **-z**. We now have to assign **-y** and **z** to satisfy the underlying clauses (first and third). This immediately unsatisfies the second clause (this is exactly the sum of its unsatisfiable statements).
Refutation: The **DIRECT SUM** would combine the first two sums into **xz** which will then combine with the third and so make **x** unassignable. So, the above set contains a change and cannot be considered to have been complete.

Scenario 3: There exist 4 sums **x-z, xi, z-y** and **y-i**. An assignment to **x** would directly leave 2 sums **-z** and **i**. We now have to assign **z** and **-i** to satisfy the underlying clauses (first and second). To satisfy the third clause, we now have to assign **y** but to satisfy the fourth clause, we must assign **-y**, which conflict, making both clauses unsatisfiable.
Refutation: The **DIRECT SUM** would combine the two sums **xi** (second) and **y-i** (fourth) into **xy** which will then combine with the third, **z-y**, to form **xz** which will then combine with the first, **x-z**, to make x unassignable. So, the above set contains a change and cannot be considered to have been complete.

To see that is the limit for depth 2 cases, we note that depth 2 sums can only relate between two variables and that an assignment to some value would force by principle, only one bound assignment. Verifying that these bound assignments do not conflict among any number **n** (here limited to 4) of related clauses is exhausted by the scenarios we explored above.

iii. **UC3 - Sums of depth 3**: These can be immediately satisfied or reduced to a sum of depth 2 if the associated assignment does not satisfy the clause.

It can be seen that sums of this depth require slightly more sophisticated refutation for most cases (except for the very first case presented).

The main refutation strategy used here is to show how we remediate the situation when we are forced into the 3 scenarios (refuted for that smaller case) mentioned in the above use case for sums of depth 2, that is, when an assignment leaves any of the following forms:

1. **xy, x-y**
2. **xy, x-z, z-y**
3. **x-z, xi, z-y, y-i**

As explained previously, assigning the underlying clause to **x** in any of these cases will cause the whole formula to become unsatisfiable.

Here our scenarios have multiple possibilities (that scale combinatorically), and we will review all of them in order in order to provide the right refutation of impossibility and/or point to the right witness that lets us execute the UNBLOCK rule for that possibility.

Scenario 1: We have scenario 1 of the UC 2 (**xy, x-y**) after assigning some integers **a** and **b**. We now identify the possibilities that can lead to this scenario as well as their refutations.
Possibility 1: The 2 original UC -2, Scenario 1 sums were of the forms **axy** and **ax-y** (here assignment to b is irrelevant).
Refutation: The **DIRECT SUM** would combine these into scenario 1 of UC 2 inheriting that refutation So the above set contains a change and cannot be considered to have been complete.

Possibility 2: The 2 original UC -2, Scenario 1 sums were of the forms **axy** and **bx-y**, if assigned to **x**, formula becomes unsatisfiable.
Analysis: We have no rule indicating that **a** and **b** should not be assigned together, and in fact, there is the very explicit possibility that they may both be required as forced arguments withing some assignment block.

If we analyze the **DIRECT SUM** operation, we see that the following sum must have been generated: **abx**

This means that we have an explicit rule/witness saying that if we assign **a** and **b**, we have to assign **-x**. Since **abx** is a 3- depth sum, it would be a part of **CX**, the set

of sums to be satisfied and that it can be satisfied makes it a **direct immediate witness**.

But this is only the view we get from looking at the two sums explicitly stated. If we assume that it is possible for our set CX may contain the cancelling sum, that is: **ab-x (witness cancellation)**.
Then we would have generated **ab**, so the premise in our possibility would be cancelled.
Refutation: We never need to assign to **x**, since we can UNBLOCK the variable by assigning it to **-x**, using an available witness, otherwise, **ab** becomes unassignable if there is a cancelling sum.

ANALYSIS INTERLUDE: As one can readily observe above, we can switch the clauses in which **a** and **b** appear to include one more test case. But notice that since we are using anonymous (letter) variables, this would amount to an abstract repetition of the above scenario.

As the following cases get more complex than those we have already looked at, that is, we have scenarios that involve 2 or more variables spanning 3 or more clauses, we will adopt an economical approach in listing down the complete list of applicable scenarios. Our approach would be thus: Where variables appear just once in a scenario, their positions will not matter, just like above. When a letter occurs twice or more, say **x** times, among **n** number of clauses, we need to "permute" the appearance of the variable among all possible **x** positions it can take in **n** slots. The reader will see, as we go along, that this will suffice to abstractly evaluate all the possible obstruction patterns generatable among 2 or more variables.

Scenario 2: We have scenario 2 of the UC 2 (**xy**, **z-y** and **x-z**) after assigning some integers **a**, **b** and **c**. We now identify the possibilities that can lead to this scenario as well as their refutations.

Possibility 1: The 3 original UC -2, Scenario 2 sums were of the forms **axy**, **az-y** and **ax-z**.
Refutation: The **DIRECT SUM** would combine **axy** and **az-y** into **axz** which would then combine with **ax-z** to give **ax** and so the original set was not complete (since we had **axy**).

Possibility 2: The 3 original UC -2, Scenario 2 sums were of the forms **axy**, **az-y** and **bx-z**. (**a** is in positions 1 and 2)

Analysis: If we analyze the **DIRECT SUM** operation, we see that the following should take place: **axy** and **az-y** combine to give **axz** which combines with **bx-z** to give **abx (AN UNBLOCK WITNESS)**. Since this witness must be satisfiable, this means once we assign **a** and **b**, we have to assign **-x**. If **ab-x** (**witness cancellation**) were also included in the set, we would get **ab**.
Refutation: We would not need to assign **x** to the formula (after assigning **a** and **b**), since we can assign to **-x** (and satisfying the underlying clauses), otherwise, **ab** becomes unassignable.

Possibility 3: The 3 original UC -2, Scenario 2 sums were of the forms **axy**, **bz-y** and **ax-z** (**a** is in positions 1 and 3).
Analysis: The **DIRECT SUM** combines **axy** and **bz-y** into **abxz** (4- depth sum-indirect witness) which combines with **ax-z** to give **abx (witness)**. If **ab-x** were also included in the set, we would get **ab**.
Refutation: We would not need to assign to **x**, since we can assign to **-x**, otherwise, **ab** becomes unassignable.

Possibility 4: The 3 original UC -2, Scenario 2 sums were of the forms **bxy**, **az-y** and **ax-z** (**a** is in positions 2 and 3).
Analysis: The **DIRECT SUM** combines **bxy** and **az-y** into **abxz** (4- depth sum-indirect witness) which combines with **ax-z** to give **abx (witness)**. If **ab-x** were also included in the set, we would get **ab**.
Refutation: We never need to assign to **x**, since we can assign to **-x**, otherwise, **ab** becomes unassignable.

Possibility 5: The 3 original UC -2, Scenario 2 sums were of the forms **axy**, **bz-y** and **cx-z** (as mentioned earlier, this becomes position agnostic).
Analysis: This analyzes the maximal context of 3 spanning assignments. The operation of the **DIRECT SUM** yields the sequence: **axy** combines with **bz-y** to give **abxz** (a 4- depth sum) which combines with cx-z to give **abcx**.
Here we note that we now have a rule, **abcx**, that says once we assign **abc** we must assign **-x**. Since this sum has a depth greater than 3, we would not try to satisfy it directly. However, we still execute it by unblocking the corresponding variable. For this, we will call **abcx** an **indirect immediate witness** and **abxz** an **intermediate witness**.
If we have a corresponding rule that says **abc-x**, then we would combine with **abcx** to generate **abc** meaning we could not assign the three variables together. When we invoke the UNBLOCK rule based on this witness, that is a witness that does not explicitly get rewritten (since we are only trying to satisfy clauses of

depth 3 or less), how can we know if we have not blocked another variable somewhere?
Explicitly, there may be a rule like **abc-xi** if we allow our sum generation algorithm go up to depths of 5. In this case, we may discover rules like **abc-x-i**, which would then give us **abc-x** which would help us reduce the original sum down to **abc** thus barring us from assigning **abc** together.
If **abc-xi** did exist and not **abc-x-i, abc-x** becomes a valid witness to an **UNBLOCK** rule (because we have verified it doesn't block any other variable) and since **UNBLOCK** rules would only ever have a maximum of 3 terms in their antecedents (reader should verify that this is indeed correct), we need not consider the effect of running more than 4- depth sum product rules as witnessable executions. Here, 5- depth sum rules are always necessary witnesses to the running of 4= depth sum witnesses but 5- depth sum rules themselves are never "run". For this we will call 5-depth sum rules, **boundary witnesses**, as they witness the "boundary" of the computation.
Refutation: If we allow sums of up to depth 5, from Axiom 2, we never need to assign to **x**, since we can assign to **-x**, otherwise, **abc** becomes unassignable.

Scenario 3: We have scenario 3 of the UC 2 (**x-z, xi, z-y** and **y-i**) after assigning some integers **a**, **b, c** and **d**. We now identify the possibilities that can lead to this scenario as well as their refutations. One can by now hypothesize the number of possibilities **p** here to be algebraically related to the maximum number of assignable integers **n**. This appears from the above cases to be about $\mathbf{p} = 2^{\mathbf{n}-1}$ These would be correct as we show below for the 16 possibilities for a maximal assignment to 4 variables **(a, b, c** and **d).**

Possibility 1: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **axi, az-y** and **ay-i**.
Refutation: The **DIRECT SUM** would combine **ax-z** and **az-y** into **ax-y** which would then combine with **ay-i** to give **axi** which then combines with **ay-i** to give **ax** and so the original set was not complete (since we **ax-z**, **axi**).

Possibility 2: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **axi, az-y** and **by-i**. (**a** in positions 1, 2 and 3)
Refutation: The **DIRECT SUM** would combine **ax-z** and **az-y** into **ax-y** which would then combine with **by-i** to give **abi** which then combines with **by-i** to give **aby** so that after assigning **ab** we must assign **-y**, breaking the unsatisfiability condition.

We can also consider a different cancelation path (and indeed we can for majority of the cases listed): **axi** combines **by-i** to give **abx**y which combines with

az-y to give **abxz** which combines with **ax-z** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

This immediately shows us that depending on the order we encounter the original clauses, we may get different additional rules, giving the idea of a non-commutative algebra. We will however see that whatever eventual signature we arrive at for a satisfiable formula, all of its possible signature will be quantitatively equivalent, that is, they will preserve the same set of total solutions. This we will demonstrate in detail when we consider the soundness and completeness of our final algorithm.

Going forward, we will only consider one of the many inductive paths that can be used to refute the scenarios we present (just as we have done in earlier cases).

Possibility 3: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **axi, bz-y** and **ay-i**. (**a** in positions 1, 2 and 4)
Refutation: The **DIRECT SUM** would combine **ax-z** and **bz-y** into **abx-y** which would then combine with **ay-i** to give **abx-i** which then combines with **axi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 4: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi, az-y** and **ay-i**. (**a** in positions 1, 3 and 4)
Refutation: The **DIRECT SUM** would combine **ax-z** and **az-y** into **ax-y** which would then combine with **ay-i** to give **ax-i** which then combines with **bxi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 5: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **axi, az-y** and **ay-i**. (**a** in positions 2, 3 and 4)
Refutation: The **DIRECT SUM** would combine **bx-z** and **az-y** into **abx-y** which would then combine with **ay-i** to give **abx-i** which then combines with **axi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 6: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **axi, bz-y** and **by-i**. (**a** (one random letter) is in positions 1 and 2, and **b** in positions 3 and 4).
Refutation The **DIRECT SUM** would combine **ax-z** and **bz-y** into **abx-y** which would then combine with **by-i** to give **abx-i** which then combines with **axi** to give

abx so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 7: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **axi,** **bz-y** and **ay-i**. (**a** (one random letter) is in positions 2 and 3, and **b** in positions 1 and 4).
Refutation: The **DIRECT SUM** would combine **bx-z** and **bz-y** into **bx-y** which would then combine with **ay-i** to give **abx-i** which then combines with **axi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 8: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi,** **az-y** and **by-i**. (**a** (one random letter) is in positions 1 and 3, and **b** in positions 2 and 4)
Refutation: The **DIRECT SUM** would combine **ax-z** and **az-y** into **ax-y** which would then combine with **by-i** to give **ax-i** which then combines with **bxi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 9: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **axi,** **az-y** and **by-i**. (**a** (one random letter) is in positions 2 and 3, and **b** in positions 1 and 4)
Refutation: The **DIRECT SUM** would combine **bx-z** and **az-y** into **abx-y** which would then combine with **by-i** to give **abx-i** which then combines with **axi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 10: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi,** **bz-y** and **ay-i**. . (**a** (one random letter) is in positions 1 and 4, and **b** in positions 2 and 3).
Refutation: The **DIRECT SUM** would combine **ax-z** and **bz-y** into **abx-y** which would then combine with **ay-i** to give **abx-i** which then combines with **bxi** to give **abx** so that after assigning **ab** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 11: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **axi,** **bz-y** and **cy-i**. (**a** is in positions 1 and 2, **b** and **c** occupy just one position).
Refutation: The **DIRECT SUM** would combine **ax-z** and **bz-y** into **abx-y** which would then combine with **cy-i** to give **abcx-i** which then combines with **axi** to

give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 12: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi,** **az-y** and **cy-i**. (**a** is in positions 1 and 3, **b** and **c** occupy just one position).
Refutation: The **DIRECT SUM** would combine **ax-z** and **az-y** into **ax-y** which would then combine with **cy-i** to give **abcx-i** which then combines with **bxi** to give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 13: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi,** **cz-y** and **ay-i**. (**a** is in positions 1 and 4, **b** and **c** occupy just one position)
Refutation: The **DIRECT SUM** would combine **ax-z** and **cz-y** into **acx-y** which would then combine with **ay-i** to give **acx-i** which then combines with **bxi** to give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 14: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **axi,** **az-y** and **cy-i**. (**a** is in positions 2 and 3, **b** and **c** occupy just one position)
Refutation: The **DIRECT SUM** would combine **bx-z** and **az-y** into **abx-y** which would then combine with **cy-i** to give **abcx-i** which then combines with **axi** to give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 15: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **axi,** **cz-y** and **ay-i**. (**a** is in positions 2 and 4, **b** and **c** occupy just one position)
Refutation: The **DIRECT SUM** would combine **bx-z** and **cz-y** into **bcx-y** which would then combine with **ay-i** to give **abcx-i** which then combines with **axi** to give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 17: The 4 original UC -2, Scenario 2 sums were of the forms **bx-z**, **cxi,** **az-y** and **ay-i**. (**a** is in positions 3 and 4, **b** and **c** occupy just one position)

Refutation: The **DIRECT SUM** would combine **bx-z** and **az-y** into a**bx-y** which would then combine with **ay-i** to give **abx-i** which then combines with **cxi** to give **abcx** so that after assigning **abc** we must assign **-x**, breaking the unsatisfiability condition.

Possibility 18: The 4 original UC -2, Scenario 2 sums were of the forms **ax-z**, **bxi, cz-y** and **dy-i**. (**all 4 assignments are in a single position**).
Refutation: The **DIRECT SUM** would combine **ax-z** and **cz-y** into a**cx-y** which would then combine with **dy-i** to give **abcdx-i** which then combines with **cxi** to give **abcdx** so that after assigning **abcd** we must assign **-x**, breaking the unsatisfiability condition.
Post-Analysis : We started out by stating that this was a procedure to provide walking instructions for depth 3 sums and less. From above, we see that in this one case, we need a 5- depth sum indirect witness. As we stated earlier, this witness would also need a higher order witness so that if we have two sums, **abcdxk**
and **abcdx-k**, we can make **abcdx** unsatisfiable. Hence we also need 6- depth sums.

Summary: So far it appears that to walk successfully while abiding by our holographic formula, we only need to provide witnesses of up to sum depth 6.

We should now state the second axiom of our theory:

Axiom 2*: Computing unsatisfiable sums up to 6- depth orders gives us the complete set of 4 and 5 depth implicit rules needed to verify that 3 depth rules and less can be run successfully using the UNBLOCK mandate, for satisfiable formula.*

5.2 Implementation

We are now ready to construct an algorithm that builds on the axiom we just stated and that is GUARANTEED as we promised, to produce a random satisfying assignment to some arbitrary satisfiable formula **F** when run. The primary goal of this algorithm would be to produce a satisfying set of assignments to **F** if it is satisfiable and then **OUTPUT YES** or indicate that it is not and **OUTPUT NO**. We leave the reader to verify that the moves made by the algorithm are indeed permitted based on the body of techniques and refutations we have stated:

i. Start by assigning an initial unassignable statement to each starting clause. Label these a starting set of sums, **S**.
ii. Expand **S** to the limit as follows.
iii. For each element currently in the set and that is added to the set, evaluate its bi-local union via the **DIRECT SUM** with every other relevant sum (sums for which it shares exactly one conflicting integer value). IGNORE any generated sums whose is greater than 6 (we will not need these witnesses).

iv. Stop either when no sums can be generated or when we have reduced two sums to their minimum, but they conflict over this minimum value. If there is a conflict of this type, the original formula is unsatisfiable, **OUTPUT NO**, otherwise:
v. If there are any (non-conflicting) minimum sums, assign their negations.
vi. Are all variables assigned (bound)?, if so, **OUTPUT YES**, otherwise:
vii. Start the free assignment of variables to values.
viii. After each free assignment, execute the necessary reductions to each relevant unassignable sum (step iv of the **Holographic Sum Of Products Rule)**.
ix. Examine the field of 2-depth sums. If any variable is blocked as indicated above (**Scenario**s **1**, **2** and **3** of **UC2**), assign its negation (Using the **UNBLOCK** rule as supported by our body of refutations in **UC3**).
x. Repeat steps v to vii until all variables have been assigned.
xi. **OUTPUT YES**.

5.3 Soundness and Completeness

Now we examine the correctness of our algorithm in terms of the topological conjecture, and our assumption that the set generated is indeed an ideal in the way we described, that is, can be extended/expanded to **EVERY** solution that satisfies the associated formula, if **ANY** solutions exist. We justify this notion of correctness in terms of the following questions, in order of increasing complexity:

i. Will the algorithm OUTPUT NO if the formula is not satisfiable (Soundness)?
   - YES. Up to step iv of the algorithm, all we have computed is the **DIRECT SUM** of ***unassignable statements***. Unless we miscategorized some statement as unassignable, which we have not, then this is the correct sum (and so is its dual) giving our operational definitions. If the algorithm outputs NO at this point, the formula is not satisfiable.

ii. Will the algorithm OUTPUT NO if the formula is satisfiable (Soundness)?
   - NO. By converse of the argument above, if after we complete step iv, the program can only output YES because we know the algorithm will proceed to the end and find a flat model using free assignments and the UNBLOCK rule.

iii. Will the algorithm compute an incorrect product, that is, a set of assignments that do not satisfy the formula (Soundness)?
   - NO. Since every product is a result of the **Holographic Sum Of Products Rule**, every product will satisfy each formula.

iv. Is there some correct product that satisfies some formula that we will be unable to compute (Completeness)?
   - NO.
   - First, assume that up to step iv, such a set has not been eliminated because we have only computed the correct sum of unassignable statements.

- Now, let us assume that such a set exists, we start at step v, by assigning some of the variables to the values given in the set until we get our first blockage. This means that if we assign the supposed value, we get conflicting values for some other variable.
- But we already have a value for that variable from our example and so our rule is wrong.
- But if our rule is wrong about this one randomly set, then the inductive process is not sound with regards to the structure of this random set.
- But we know that it is sound for determining unsatisfiability.
- And we know that it is sound for determining satisfiability for instances with exactly one solution. They are not unsatisfiable, and all assignments will be bound.
- If our algorithm breaks on some random set, then the set must have some type of extra structure!
- We know we can take any formula with more than one solution and conjoin it with other clauses in such a way that we reduce the number of solutions down to 1.
- We can take our original formula which contains this problematic solution and modify it so that it has only one solution, the problematic solution.
- We know that we will now find this solution in the altered formula. It is the only solution.
- Therefore, we would have found the solution in the original formula also and so the set could not have induced the blocking rule.
- Therefore, our original answer is right and our algorithm and therefore our topological conjecture is proven complete for all possible products.

The conclusion of this analysis proves Theorem 2 (and its corollary) as we originally stated it.

5.4 Runtime And Optimizability Analysis

The essence of this algorithm essentially compares each member of a set with every other member of the same set with which it shares a single conflicting variable. Since we know that the maximal depth of elements cannot exceed 6, the set size cannot exceed the number of ways to choose random 6 digits from n, which is exactly n^6. The maximal running time for our algorithm is indeed easy to compute. It is the combinatorial time needed to complete the necessary set of comparisons required, which the maximal size of the set, n^6, in a cartesian product with itself, which yields $\mathrm{O}(\mathrm{n}^{12})$. Subsequent steps to assign n^3 number of clauses in exactly that amount of time, $\mathrm{O}(\mathrm{n}^3)$, and to check for blocked variables requires comparing depth

two sums with each other in a cartesian product giving a runtime of $O(n^4)$. So, in fact, the worst-case running time for our algorithm remains $O(n^{12})$.

It is easy to see algorithms in practice effectively partitioning data so that elements need only to look at so many elements, indexed by some structure which looks at their constituent integer values. Thus, for problems already highly structured (probably where the internal propositions represent rational terms of some domain of thought), running times may well go below the worst case pictured here.

Also, the fact that the common number of terms in two elements need not exceed 5 is potentially a factor that some algorithmically innovative technique may exploit in a novel way to further constrain running times.

These considerations, along with the fact that the very form of the algorithm promises to be hardware (hardware meaning even physical systems we do not currently consider as computational) implementable for faster processing of domain specific computational problems, makes the optimization of this algorithm beyond its current form, look promising.

All these possibilities offer hope the algorithm here may indeed be extended and implemented in practice in its exact form in ways more efficient than presented here.

One exception we see here may be in the feasibility of constructing increasingly effective prime factoring algorithms. We will show in future work, when we address the formal cohomology that integer prime factorization may still be the most naturally difficult problem to crack with our system and may be still even harder than simply using other known, traditional methods.

6. Implications For Theories of Computational Complexity and Computability

To the computational complexity theorist, the implications for this algorithm are easy to see: That the class of **NP** problems does reduce to the class of **P** problems via this approach. To do this, we implicitly followed descriptional complexity approach of extending unordered (implicit) first order logic by an induction operator, where order is defined according to some monotone progression. This has precedence in the complexity literature for example, in the work of Dawar [9]. In fact, since we are also able to prove unsatisfiable cases, we really do have an extension of known nullstellensatz styled algebraic proof systems [10] and indeed have a proof that **NP = coNP**.

More importantly, by virtue of invoking Tarski's principle of inductivity, certain issues of self-reference and the impossibility of computing semantic (satisfiability here is realized as a semantic property of logical formulae) properties of formulae/programs invoked in the proof of Rice's theorem, particularly related to the halting problem, become addressable, as follows:

It is a known fact that any computer program can be represented in circuit form which can then be converted to CNF form and then to a 3 SAT form. This same construction can also be applied to any type of computable inputs to some arbitrary program.

Rice's theorem in terms of the halting procedure states that it is impossible to know if some arbitrary input will cause a program to terminate. This is simply because we do not have,

nor can we demonstrate a logical model of termination that can be represented in the first order language normally used to address this problem.

In our model, this is remedied. A termination of a program in holographic form can simply be interpreted as the holographic reduction to a final product when the model is fed a series of operations from some input space. This is possible because our model guarantees that the meta language is responsible for interfacing between inputs and our program space. In this case, non-termination can simply be read as a program which hangs because the sequence of input operations has not yielded a product (the original formula has become unsatisfiable OR the set of input operations assigns valid values to only a subset of the variables.

This description is quite condensed but should be sufficient to show that we are able to achieve, in what would seem in computational terms, a relative Turing jump for what may be termed computably enumerable but uncomputable sets. Instead of working with explicitly defined arithmetic schemes, what we are able to achieve is what we may call meta-order dualization. In future work, we show that the arithmetic scheme used in the syntactic construction of programs is dual to a higher order scheme used to specify the programs semantically. We hope this is an area in which our findings here may be able to help invigorate and advance the study of computable functions and computability in general.

7. Implications For Other Mathematical Fields.

Where we seem to have ignored, or not explored in very much detail, the potentials of many mathematical intricacies inherent in our model in pursuit of our definition and proof of existence of a product space, a very algebraic object, it is our hope that working mathematicians may take some of these ideas in a more fruitful direction.

One apparent type of structural object we are working with is that of a naturally dual functor that associates an unassignable model to an assignable one (or more precisely a model and an ideal in a ring) and our unions between statements are adjunctions between functors with an overall algebraic identity. In essence, what we have can be seen as a type of abstract definition of self-adjoined operator.

And what exactly about this manifold (representable as a tuple of algebraic integers, and thus Euclidean), which we never explicitly construct, makes it have these special properties from which a highly structured operator can be defined? We contend there is a property – fibered symmetric categoricity – which we borrow from category theory and homological algebra, which gives our manifold these properties.

Symmetric categoricity, informally defined here is the inherent mappability of an initial power object to all possible final objects via computable morphisms. The anti-product we define is a natural co-product of what we may call fibered symmetric k-categories, where k is the depth of the clause. If we ask what the objects that we categorize are, these would be the directed constraints, that is, the intervals/open sets, and together they constitute topological data about some closed set, which makes the point over which the topological algebra induces objects and their morphisms, a sheaf (that is the point being multiplied) – locally defined data (inductively) attached to the open sets of a topological space. The category of sheaves on a topological space, generated from a small category (clause) on a site with a big category (formula) is the definition of topos found in most literature. Here we seem to have the case of localized, schematic topos,

a discrete analogy that could be of interest to mathematicians that study these objects. A place to begin investigations into the kind of underlying abelian categories used, that is additive categories (pre-additive category with all finite biproducts – product/anti-product pairs), those most alike in structure to the underlying types of categories implied above would be the essential study of Grothendieck categories [11].

The full analyses of these methods of investigation lay beyond the scope of this paper but one can say that all of these readily available mathematical tools in the field of topology, topological algebra and differential and algebraic topology coupled with our initial reference to the holographic principle, does point strongly to the fact that this theory can be used to model physical processes and it is itself a kind of abstract physical process.

Another strong indication of the above fact is from the field of logic itself in the form of linear logic. We recall that essentially, linear logic is an improvement to proof and truth-based logics that emphasizes the use of resources. Our method is indeed a form of complex linear logic in emphasizing satisfiability/assignability as a resource with similar operations for object formation, introduction and eliminations. Linear logic has since been identified as a very suitable logic for quantum processes, a thing which is readily seen from the (implicit Lagrangian) manner in which our theory regulates degrees of freedom over super-posable probabilistic end states.

Finally, the nature of the manifolds we encounter strongly suggest some variant of String Theory. Briefly, they seem from a surface inspection to have a Kähler form, with the following identifiable properties:

i. Symplectic (differential) form on unassignable statements.
ii. Complex form on assignable statements AND
iii. Riemannian metric over the product space obtained as a positive definite, continuous expansion mode over the entire manifold, leading up to a solution (for satisfiable instances).

We intend to fully delineate this structure in future work where we consider the scheme cohomology. We contend that the String/Membrane (with surface regulation of potential) theory we get from a full pursuit of this analytic method, beyond what can be easily gleaned as above, also yields a very powerful analytic form that may be able to shed light not only physical initial conditions (like the ones suggested by the Big Bang theory) but may well be used to explain processes (as we have shown directly in the case of 3 satisfiability) as diverse as to how virtual particles (of information/energy) push/scatter on and around each other (applying weights via some field law/theory) to acquire mass (permanence) and become real, relative to each other under the influence of gravity (attraction to similar regions of space). It can be shown that this explanatory modelling power can also be applied to the intricacies of other natural processes such as the evolution of these kind of systems along the lines of intelligent evolution.

In fact, to buttress the point about its evolutionary modelling power, an algorithm generated the way we described functions as a domain specific evolutionary calculus over combinatorial species according to Joyal [12]. This would be a calculus with a direct signature of these operations (+, ×) and their inverses as well as indirect operations of function exponentiation and composition. A full discussion of the physical and evolutionary implications of the results

obtained here will be fully addressed in a future work, as they are closely tied with viewing the process itself as an elementary form of machine reasoning with shared signal processing.

8. Conclusions and Future Directions

As evident from the set of implications considered above, holography as elucidated seems to be a promising foundation for the exploration of the solution spaces to many complex problems as well their model implications. We hope that the algebraic and algorithmic ideas pursued here can be further developed by other researchers along these and other promising lines.

The personal interest of the authors is in pursuing this line of reasoning, that is, in addition to providing computationally satisfying descriptions of natural processes as explained above, is also to study the form as the building block of a powerful new form of machine reasoning, as indicated in the closing of the last section, a topic which in itself demands further in-depth investigation and research.

It is our belief that the methods explored here can be used and further developed in the pursuit of the resolution of many important scientific, engineering and technological problems.

References

[1] Valiant, Leslie (17–19 October 2004). Holographic Algorithms (Extended Abstract). FOCS 2004. Rome, Italy: IEEE Computer Society. pp. 306–315. doi:10.1109/FOCS.2004.34. ISBN 0-7695-2228-9.

[2] Bousso, Raphael (2002). "The Holographic Principle". Reviews of Modern Physics. 74 (3): 825–874. arXiv:hep-th/0203101. Bibcode:2002RvMP...74..825B. doi:10.1103/RevModPhys.74.825. S2CID 55096624.

[3] Stephen A. Cook. The complexity of theorem-proving procedures. In Proceedings of the 3rd ACM Symposium Theory of Computing, pages 151–158, Shaker Heights, Ohio, 1971.

[4] Leonid A. Levin. Universal search problems. Problemy Peredachi Informatsii, 9(3):265–266, 1973.

[5] Richard M. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations, (R. Miller, J. Thatcher eds.), pages 85–103, 1972.

[6] L. G. Valiant. Expressiveness of Matchgates. Theoretical Computer Science, 281(1): 457-471 (2002). See also 299: 795 (2003).

[7] J. -Y. Cai, V. Choudhary and P. Lu, "On the Theory of Matchgate Computations," Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07), San Diego, CA, USA, 2007, pp. 305-318, doi: 10.1109/CCC.2007.22.

[8] Alexander Beilinson, Vladimir Drinfeld, Chiral Algebras, Colloqium Publications 51, Amer. Math. Soc. 2004.

[9] Dawar, Anuj (1993). Feasible computation through model theory. University of Pennsylvania ProQuest Dissertations Publishing, 1993. 9321378.

[10] R. Impagliazzo, J. Krajíček, P. Pudlák, A. Razborov and J. Sgall (1995/1996), Proof Complexity in Algebraic Systems and Constant Depth Frege Systems with Modular Counting. Computational Complexity 6 (1995/1996) 256-298.
[11] Grothendieck, Alexander (1957), "Sur quelques points d'algèbre homologique", Tohoku Mathematical Journal, Second Series, 9: 119–221, doi:10.2748/tmj/1178244839, ISSN 0040-8735, MR 0102537
[12] Joyal, André (October 1981). "Une théorie combinatoire des séries formelles". Advances in Mathematics. 42 (1): 1–82. doi:10.1016/0001-8708(81)90052-9.