Fractality and Coherent Structures in Satisfiability Problems

Theophanes E. Raptis[abc]

[a]National Center for Science and Research "Demokritos", Division of Applied Technologies, Computational Applications Group, Athens, Greece.

[b]University of Athens, Department of Chemistry, Laboratory of Physical Chemistry, Athens, Greece

[c]University of Peloponnese, Informatics and Telecommunications Dep., Tripolis, Greece

Abstract: We utilize a previously reported methodological framework [5], to find a general set of mappings for any satisfiability (SAT) problem to a set of arithmetized codes allowing a classification hierarchy enumerable via integer partition functions. This reveals a unique unsatisfiability criterion via the introduction of certain universal indicator functions associating the validity of any such problem with a mapping between Mersenne integers and their complements in an inclusive hierarchy of exponential intervals. Lastly, we present means to reduce the complexity of the original problem to that of a special set of binary sequences and their bit block analysis via a reduction of any expression to a type of a Sequential Dynamical System (SDS) using the technique of clause equalization. We specifically notice the apparent analogy of certain dynamical properties behind such problems with resonances and coherencies of multi-periodic systems leading to the possibility of certain fast analog or natural implementations of dedicated SAT-machines. A Matlab toolbox is also offered as additional aid in exploring certain simple examples.

1. Introduction

The Boolean Satisfiability (SAT) problems comprise a central part of both computer science, mathematical logic and complexity science with applications ranging from Artificial Intelligence (AI) as well as circuit design and automated theorem proving. It is also one the first proven cases of Non-Polynomial (NP) problems [1], [2] for which no general algorithm for solving every possible case has been reported as yet. A modern review of the problem can be found in [3] and [4]. In this report we present a new methodology for tackling these problems which largely departs from traditional methods by making use of some analytical approaches following certain reduction protocols. The emphasis is not so much in accelerating computations for specific cases as it is towards deepening our understanding by looking at possible symmetries and recursive structures of certain supersets of all possible such expressions.

In previous work [5], we introduced the notion of Inductive Combinatorial Hierarchies (ICH) to provide the means for an organized search of questions regarding symbolic patterns and their manipulations in a variety of situations including abstract computations and complexity measures. One of the main points behind this new setting relies in the empirical observation that many indicator functions defined over successive exponential integer intervals present a ubiquitous structure akin to arithmetic fractal sequences the origin of which can be traced back to the period mixing of counter sets. This can also be interpreted as a discrete analogue of interference with the main characteristic of the associated set of periods forming a lacunary series.

Given that indicator functions play a major role in both the so called, functional [6] as well as purely functional programming framework [7] as well as in other areas of machine learning for uniform computability as in neural networks where characteristic functions are approximated via polytopes [8], [9], [10] the existence of as yet unexplored constraints imposing a hidden recursive structure manifesting as scale invariance over a hierarchy of domains and to a large extent independently of the particular problem at hand, poses a serious challenge in further understanding its origin and implications for simplifying a variety of problems and hence, achieving an ultimate reducibility in terms of complexity measures, either in the sense of Kolmogorov [11], [12], [13] or Rademacher [14], [15] or any other possible.

An essential step towards this direction is to recognize two major classes as in principle two foreign sets of possible morphisms, the first one being given always as an algebraic mapping of integers to integers, depending solely on some functions over the quantitative content ("*value*") of the integers, while the second being applied strictly symbol-wise on their expansion in some alphabet base (otherwise known as "*radix*") *b*. A complete arithmetization of the second class is any attempt in finding a correspondence between a member of the second class with one or more of the first one, via a set of pattern recognition attacks on a hierarchical lay out of the resulting sequences of outputs encoded as integers over successive exponential intervals, whenever these can be resolved by some analytical methods. As a matter of fact, the one dimensional orderly projection used in ICH intends to act as a replacement of probabilistic

approaches which in cases of a fractal order being present could turn out being recursive.

The current effort is focused on a double target as part of a larger program including (*a*) applications of the "*et pluribus unum*" principle introduced in [16] for finding ways to perform digital computations without digital design with possible applications in analog and natural computing, (*b*) reexamination of the status of complexity theory under the light of the same principle where large integers are treated like "black boxes" with pattern mappings performed via direct arithmetic operations and, (*c*) lastly, the possibility of using such techniques for formal verification [17], [18].

That these same techniques are directly applicable to the case of the general satisfiability (SAT) problem is not directly obvious and it gives an opportunity for demonstrating the power of the arithmetization technique of which one of the main benefits is a serious reduction of the Kolmogorov complexity of search algorithms. While the SAT problem has a long history and the number of algorithms in circulation are many and constantly improving [19], [20], [21] we will concentrate on an attempt to completely arithmetize any SAT instances proceeding constructively in a series of progressively more general expressions of the overall problem starting from the more symmetric ones.

To this aim, we first introduce the ICH toolbox in section ***2*** while the types of symmetry leading to an overall hierarchical classification and enumeration of all possible SAT instances are appropriately quantified in section ***3***. In section ***4*** we introduce a set of universal indicator functions admitting an analytical formulation in terms of bitwise operators. These can easily be generalized for problems with a redundant set of atomic variables and arbitrary negation literals of which the action becomes equivalent to that of a set of logical masks. In section ***5***, we provide the additional means for treating the set of all possible negation codes resulting in a Global Truth Table as well as for treating overlapping logical variables. In section ***6***, we perform a reduction via the introduction of a "*clause equalization*" protocol leading to an association of each *SAT* instance with the trajectories of a Sequential Dynamical System. These are then reduced to analytic series representations of some number theoretic functions which finally reveal any such instance to be

resolvable via the product of at most three appropriate indicator functions. The relevant Matlab package *SATbox* is examined in detail in section *7*. In the last section we further discuss the significance of fractal sequences that are revealed through the previous analysis and their close association with wave-like phenomena which we suggest that it calls for a new axiomatic framework.

2. Inductive combinatorial hierarchies: a How-To

We consider an inclusive sequence of integer intervals S_N:[0,…,2^N-1] satisfying $S_1 \subset S_2 \subset \ldots \subset S_N$ and the associated lexicographically ordered word dictionaries of the binary patterns for each integer in these intervals laid as a hierarchy of N x 2^N matrices $W_1 \subset W_2 \subset \ldots \subset W_N$ known also as binary designs [22]. Alternatively, any such dictionary is identical with the ordered set of unfolded paths of a binary tree with L+1 nodes (with the root node not "visible").These then constitute the members of a well formed, naturally self-similar hierarchy of which the rows correspond to an increasing set of periodic counters of periods $2^k, 0 < k \leq N$ and the columns correspond to the decoded binary expression of each integer index in any interval with the least significant bit at the first row and the maximal one given by the binary logarithm defined as $l_2(\nu) = 1 + \lfloor \log_2(\nu) \rfloor$. We will consider these two directions as complementary with the row-wise direction being termed the direction of "*uniform computability*" and the column-wise as that of "*individual computability*". Any function which contains a coordinatization of the form $f(v,k)$ will then be considered as both uniformly and individually computable in which case we shall call it *completely resolvable* or simply *complete* over the hierarchy. An example of this construct is shown in figure *1*.

In problems of satisfiability, bounded length L expressions result in an exponentially large search space which is often treated via the introduction of a hypercube $\{0,1\}^L$. We shall avoid this technique here and we shall instead use a type of "*uniform deployment*" over any interval S_N forming an integer sequence and in particular a large binary word corresponding to any indicator function defined over the same interval. We shall also associate every W_L dictionary with a discrete time flow of the underlying set of counters which is then the equivalent of a "*successor*" function in the uniform direction while the individual

direction serves as a "*rank*" map providing an ordered sequence of bits corresponding to the associated index of the discrete time flow. An "*unrank*" map then inverts this operation by giving back the index of a sequence. This allows a representation corresponding to a hierarchy of Boolean Dynamical Systems (BDS) with the operator formula serving as the "program" and the Word Dictionary as the underlying BDS providing the driving data. The essential property of any such BDS is that for any level of a hierarchy with members $\{S_L,\ W_L\}$, the individual power exponents are turned into periods often resulting in an arithmetic fractal structure of the resulting sequence. An important distinction concerns two classes of sequences which can be characterized as "*amending*" and "*non-amending*" with respect to their evaluation across any level of the hierarchy. This depends on the possible sensitivity of any operator formula on a new symbol added on top of each level of $\{S_L,\ W_L\}$ which forces reevaluation on the whole interval. As a trivial example, simple sequences like that of the digit-sum function are non-amending while for SAT indicators the opposite must be true.

The resulting fractality seems ubiquitous and it can be observed in a variety of settings including such abstract cases as the set of allowed words of a Dyck language [23] (a concrete example can be seen in https://github.com/rtheo/Easy-Dyck). This is also an example of an amending sequence. An arithmetized version of such indicator functions will then be any individually resolvable analytical expression as a function of the integer index solely which avoids any use of the pair of *rank* and *unrank* bijections entirely. If a unique formula can be proven inductively for any two consecutive steps then it is universal which is especially useful if a generating function can also be extracted. We shall also need to introduce a special map required for the block analysis of indicator functions especially in the case of an underlying fractal order.

This is given via some bit block counting method which performs a lossless mapping into a higher alphabet of the alternating blocks of same bits the simplest such encoding given as an alternating sign sequence $\{c_i\}$ of varying length satisfying $\sum_{i=1}^{N_C} |\, c_i \,| = L, 0 < |\, c_i \,| \leq L, \forall S_L$ with N_C as the "*block dimension*" while the sign follows the (*mod* 2) periodicity for the

first counter across any S_L. We notice that the sign convention can also be transferred to a single additional bit given by the (*mod 2*) operator while the first and last counters with negative sign are equivalent to the *leading* and *trailing zeros* as often mentioned in computer science [24]. By construction, the set of block counters for any S_L is a subset of the integer partitions of L [25]. In fact, one can construct the whole set of the L-partitions by choosing a sufficiently larger L order.

The main rule of thumb in all such cases is that an individual formula trades time complexity over space complexity in that recursive formulas for uniform computability are often simpler at the expense of exponential memory consumption, hence satisfying a generic type of inequality like $M\tau > C$ where C some arbitrary constant. This is also strongly associated with the fact that the uniform direction at any level of a hierarchy is always absolutely ordered while any individual direction appears increasingly randomized at a varying degree of complexity on different zones of any dictionary. Given recent advances in the Fourier analysis of Boolean functions [26], [27], it might be possible to promote the previous into a kind of "uncertainty" principle but this lies beyond the scope of the present report.

In what follows, we also introduce an additional tool that we shall call "*arithmetization*". The fundamental arithmetized operation in the above defined hierarchy can be given as the identity $Id_{\mathrm{N}}(\nu) - \nu = 0$ where we define the identity operator via the polynomial representation as

$$Id_{\mathrm{N}}(\nu) = \sum_{k=0}^{L-1} 2^k \sigma(\nu, i) \tag{1}$$

In (1), we use the abstract symbol $c(v,i)$ for any bit decoding operator at a row-wise position k of which a frequently used analytical expression is given by $\sigma(\nu,k) = \mathrm{mod}\left(\lfloor \nu / 2^k \rfloor, 2\right)$ with $\lfloor\ \rfloor$ denoting the floor function. In what follows, we shall introduce a much simpler, division-free equivalent expression which can be constructed from a restriction of a general asymmetrically 2-periodic square pulse of the form

$$\eta(x; p_0, p_1) = \Theta\left(p_1 - \mathrm{mod}(x, p_1 + p_0)\right) \tag{2}$$

The η function represents an asymmetric pulse with an excitation period p_1 and a silent period p_0 giving a total period $p = p_1 + p_0$, while Θ stands for the modified Heavyside function with $\Theta(0) = 1$. We notice in passing that the function is generic even for $x \in \mathbf{C}$ where it creates a regular lattice for $p_{0,1} \in R$ while there is an equivalent residue based expression as $\Theta\left(p_1 - p \bmod(x/p,1)\right)$. It is then possible to reproduce the bit decoder via the choice $p_0 = p_1 = 2^k, k = 0,1...$ and either a phase shift of the main period or by taking its complement as $c(\nu,k) = 1 - \eta(\nu,2^k,2^k)$. We shall heretofore adopt the simplified notation $\overline{\eta}(\nu,k) = 1 - \eta(\nu,2^k,2^k)$ which we shall call, the restricted η function. Since periods are symmetric, we can also provide the equivalent residue form $\Theta\left(2 \bmod(x2^{-k} - 1,1)\right)$. Any matrix W_N can then be reconstructed as a concatenation of complete constructors given as

$$W_{jk} = \overline{\eta}\left(j,k\right) \quad j \in S_N, 0 \le k < N \tag{3}$$

An alternative expression can be given with the aid of a phase shift $\nu_0 = 2^k - 1$ performing a bit complement as well as with inversion of the Θ argument

$$W_{jk} = \eta\left(j + \nu_0, k\right) = \Theta\left(\bmod(j,2^k) - 2^{k-1}\right), \quad j \in S_N, 0 \le k < N \tag{4}$$

Other elementary examples include the well known case of the "*Digit-Sum*" function $s_2(\nu)$ [28] and its trivial derivative, the parity function $mod(s_2(\nu), 2)$ for which we can immediately provide a complete formula as

$$s_2(\nu) = \sum_{k=0}^{l_2(\nu)} \overline{\eta}(\nu,k) \tag{5}$$

The case of $s_2(\nu)$ has already being proven to be also equivalent to a fractal function F as $s_2(\nu) = \nu \log(\nu)/2 + F$ [29]. Similar formulas can be written for other important quantities as the Hamming distance which simply replaces the single term in (5) with the absolute difference of two terms. We should stress here the fact that objects like W_L matrices are here treated as samplings of an underlying dynamics in the uniform

direction which can be periodically extended in an infinite interval. This then can also be interpreted as an extension of the well known orthonormal Rademacher system [30] consisting of a sampling of a so called, lacunary sequence of harmonic oscillator signs. This also allows generalizing the Walsh system comprising products of the previous [31], [32]. Hence, any of the operators and formulas defined below can also be considered as particular filters sampling an underlying continuous flow.

A set of important object expressible in a similar way are bitwise operators that play a major role in the expression of some universal indicator functions and their reduction to equivalent BDS in the analysis that follows. We may give beforehand a compact representation of the basic form of any bitwise operator denoted here as $R(\nu,\mu)$ as a binary expansion of the form

$$R(\nu,\mu)=\sum_{\pi=0}^{L-1}2^{i}\,r_{\{\wedge,\vee\}}\left\{\overline{\eta}(\nu,\pi),\overline{\eta}(\mu,\pi)\right\},\quad \nu,\mu\in S_L \tag{6}$$

In (6) we use r to denote either a standard Boolean AND ($\wedge$) and OR ($\vee$) operator. Using the periodic η function representation as in (8) we can perform a dissection of the above formula for any S_{L} to show that all terms are superpositions of the same scaled square grids which then attains the form of an "interference" pattern as

$$R_{\wedge}(\nu,\mu)=\sum_{\pi=0}^{L-1}2^{i}\overline{\eta}(\nu,\pi)\overline{\eta}(\mu,\pi) \tag{7a}$$

$$R_{\vee}(\nu,\mu)=\nu+\mu-R_{\wedge}(\nu,\mu) \tag{7b}$$

The second formula is derived from the known identity $x\vee y=\neg\left(\neg x\wedge\neg y\right)\cong 1-(1-\eta_{\nu})(1-\eta_{\mu})$ which when applied to each summand takes the form $\eta_{\nu}+\eta_{\mu}-\eta_{\nu}\eta_{\mu}$. The two operators commute over any pair of integers $\{\nu,\mu\}$ with the additional "edge" properties over any intervals

$$R\vee(\nu,0)=0,R\wedge(\nu,0)=\nu,R\vee\left(\nu,2^{L}-1\right)=2^{L}-1,R\wedge\left(\nu,2^{L}-1\right)=\nu$$

and the fixed points $R_{\wedge,\vee}(\nu,\nu)=\nu$. Interestingly, (7b) may be used to define the notion of a "*complement over a plane*". Computation over any hierarchy has the amending property.

A mostly useful operator in the SAT analysis that follows is the bitwise difference or exclusive disjunction operation which can be defined via the expansion

$$R_{\oplus}(\nu,\mu)=\sum_{\pi i=0}^{L-1}2^{i}\,\overline{\eta}(\nu,\pi)\oplus\overline{\eta}(\mu,\pi) \tag{8}$$

where $x_i \oplus x_j$ denotes the individual XOR of each pair of bits. It is possible to derive it algebraically from a product of the two previous operators in (7) corresponding to a product of summands as $|\eta_\nu-\eta_\mu|^2=\eta_\nu+\eta_\mu-2\eta_\nu\eta_\mu$. Using the explicit forms from (7) results in the identity

$$R_{\oplus}(\nu,\mu)=\nu+\mu-2R_{\wedge}(\nu,\mu) \tag{9}$$

Hence, the bitwise exclusive disjunction is identical with the difference operator $R_{\vee}-R_{\wedge}$. With the aid of (7a-b) and (9) we can then write the equivalent forms

$$R_{\vee}(\mu,\nu)=\frac{1}{2}\left(\mu+\nu-R_{\oplus}(\mu,\nu)\right) \tag{10a}$$

$$R_{\wedge}(\mu,\nu)=\frac{1}{2}\left(\mu+\nu+R_{\oplus}(\mu,\nu)\right) \tag{10b}$$

An important property of many such objects when expressed as 2^L x 2^L matrices over exponential integer intervals has already been recognized as the origin of many arithmetic fractals by others [33], [34], [35]. Some uses of arithmetic fractals appeared recently in [36]. The block analysis of successive rows of fractal matrices reveals a complicated pattern of modulated block periods with no obvious expressions for p_0 and p_1 and leads to the need for special interpolation methods.

A fundamental property of these operators associates (8) with a special subgroup of the permutation group. Let then a triplet of integers as $\{\lambda, \mu, \nu\}$ such that $\lambda + \mu = \nu$. Let also their binary expansions be given as $\{\sigma_\lambda,\sigma_\mu,\sigma_\nu\}$. Since the two first form a restricted 2-integer partition of ν, it must hold that

$$\sigma_\lambda(i)+\sigma_\mu(i)=1,\forall\sigma_\lambda(i)\neq\sigma_\mu(i)$$

We may then call λ and μ the "*partial negation codes*" of v as each of them can be used to invert certain bits of v via $|\sigma_\lambda-\sigma_v|$ or $|\sigma_\mu-\sigma_v|$. The action of any negation bits is a bitwise operation in any pair (μ, v) which is identical with a bitwise exclusive disjunction hence it is straightforward to extract the total action of any bitwise negation in S_L via an additional operator applied as a transformation on all inputs of (21a-b) in the form $\nu\to\overline{\nu}_\mu=R_\oplus(\mu,\nu)$. This also leads to the simultaneous identities $h(\mu,\nu)=s_2\left(R_\oplus\left(\mu,\nu\right)\right)=s_2\left(\overline{\nu}_\mu\right)$for the Hamming distance h of any two positive integers.

Let then, an arbitrary exponential interval S_L and a negation mask coded as an integer $\mu\in S_L$ with a binary logarithm $\lambda = l_2(\mu)$. For the case of the marginal value of $\mu_L=2^L-1$, it acts as a total negation operator satisfying the known identity $\overline{\nu}=2^L-\nu-1$ while for all other cases it corresponds to a partial negation. Let then v_μ be the greatest integer in any S_L such that for any v in the same interval we have

$$\nu=\nu_\mu+\left\lfloor\frac{\nu}{2^\lambda}\right\rfloor \tag{11}$$

Then v_μ has the same most significant bit at the level of any partial negation the rest of the bits of any v being invariant under μ. Let then $\overline{\nu}_\mu$ be the result of any partial negation in which case we immediately have the following obvious identities

$$\overline{0}_\mu=\mu,\ \ \overline{\mu}_\mu=0,\ \ |\overline{S}_\lambda|=|S_\lambda|-\mu,|S_\lambda|=2^\lambda-1 \tag{12}$$

For any 2^L x 2^L matrix over all pairs of (v, μ) values, (12) fixes the main diagonal and the first and last columns of inverse order. We may also classify separately the cases of $\mu = 2^i$, i=0,…,L-1 for which the sole action is the reversal of the corresponding period at the same significance level i resulting always to a permutation class analogous to the corresponding period. Hence, for μ = 1 we get an action isomorphic to a swap matrix between even and odd indices giving simply $\overline{\nu}_1=\nu+(-1)^\nu,\nu\in S_L$.

Similarly, for μ = 2, for which we can write a scaled version as $\tilde{\nu}_2 = \nu - 2[2\eta(\nu;2,2)-1], \nu \in S_L$ and so on. For all other cases, we will have a mixture of swaps creating a total permutation. For instance, in case of μ= 3 we first find the uniform deployment of all partial 1-negations as the sequence *1 0 3 2 5 4 7 6…* followed by the partial 2-negations giving as a total the new sequence *3 2 1 0 7 6 5 4…* This way every one of the intermediate partial negation codes in every S_L can be classified accordingly. The level of the most significant bit also determines the "locality" or neighborhood dependence for each particular μ code. An alternative treatment is given in App. *A*, where a direct relationship is proven between individual permutation elements and 2-integer compositions of any μ code.

An additional property directly associated with the internal mirror symmetry of any W_L array can be given through the restricted *2*-integer partitions of any $2^L - 1$ integer of which the binary expansions are always the total bitwise not-complement of each other. When the total complementation is used in any of the arguments of any of the three *R* operators defined previously, it performs either a left right or an up-down flip of the resulting matrix for all 2^L inputs. We can numerically verify that any total summand of the form $R(\mu,\nu)+R(\overline{\mu},\nu)+R(\mu,\overline{\nu})+R(\overline{\mu},\overline{\nu})$ will evaluate in a constant matrix with its unique value being always $s(2^L-1)$ with *s* = 3, 2, 1 for bitwise disjunction, exclusive disjunction and conjunction respectively. We notice that the *s* integer is the number of truth values in the elementary symbol-wise version of each operator. The three separate summands that result are equivalent to three linear functional equations. Using also the decompositions (10a-b), one can verify that they are compatible with the total summands. One is then left with a unique functional equation for the exclusive disjunction operator of the form

$$R_{\oplus}(\nu,\mu)+R_{\oplus}(f(\nu),\mu)+R_{\oplus}(\nu,f(\mu))+R_{\oplus}(f(\nu),f(\mu))-2^{L+1}+2=0 \quad (13)$$

In (13) we used $f(\nu)=2^L-\nu-1$ for the total complement over any S_L to emphasize the resulting functional forms while the order *L* can also be taken as $l_2(\max\{\nu,\mu\})$. We would like to stress the importance of (10) and (13) as well as the restricted integer partition property of App. *A* for further development of a bitwise calculus that could reveal the existence

of register-less logical machines with as yet unknown types of fast, analog implementations.

Last, we provide a useful extension of the binary hierarchy on higher alphabets that can be given with an inclusive sequence of larger interval $S_L^{(b)}:[0,...,b^L-1]$. A complete hierarchy shall comprise the objects $\{S_L^{(b)},W_L^{(b)}\}$where $W_L^{(b)}$ shall be an associated matrix from a lexicographical ordering of multiple step counters with a scaling of order b^i. To provide an analytical form we expand the previous definitions appropriately. Consider then any set of b-counters as constructors of an associated $W^{(b)}$ matrix given as

$$W_{ij}^{(b)}=\mathrm{mod}\left(\left\lfloor\frac{i}{b^j}\right\rfloor,b\right) \tag{13}$$

We can replace (13) with a new division-free version of a generalized η_b function where a standard η function replaces the scaling factor as

$$W_{ij}^{(b)}=\eta_b(i,j)=\sum_{k=1}^{b-1}\overline{\eta}\left(i;kb^{j-1},(b-k)b^{j-1}\right) \tag{14}$$

In the above we use an accumulator of ones for the reconstruction of each symbol while periods form a restricted 2-integer partition of the scaling factor b^{j-1}. Using the residue form of the original η function we can also write the expansion terms as $\Theta\left(b\bmod((x/k)b^{1-j}-k,1)\right)$.

The generic expression in (14) leads to a curious algebraic structure when one uses the alternative expression of the Heaviside function in the original definition of the η function as $\Theta(x)=\left(1+x/|x|\right)/2$, by noticing the fact that the absolute value in the denominator of the sign equivalent is restricted to the ternary alphabet thus making possible the identification $|x|\cong\eta_3\left(x,j(x)\right)-1$. Using the original definition as $\overline{\eta}(\nu,p,p')=1-\left(1+x/|x|\right)/2$ where x stands for the internal function $p-\mathrm{mod}(\nu,p+p')$ then leads to an additional identity where every base expansion term η_b can be expressed with the aid of a modulated version of η_3 solely, including η_3 itself. Using an appropriate match $f(j,k)$ for the original periods in (10) and the equivalent (*mod* 1) form of η results in a summation formula as

$$\sum_{k=1}^{b-1} \frac{k - b \bmod(ib^{1-j}/k,1)}{\eta_3(i, f_b(j,k)) - 1} = \frac{b(b-1)}{4} - \eta_b(i,j)$$

The presence of poles in the above suggests that these objects should be studied as a set of merorphisms in the complex plane rather than the restricted integer domain we are interested here. The significance of the above goes beyond the scope of the present work so we only mention it here for the integrity of the presentation of the new generalized η_b functions.

Let now $s_b(\nu)$ be the higher order digit-sum function exhausting all values in any bounded interval $\{0,\ldots,\mathrm{Nb}\} \forall \nu \in S_N^{(b)}$. We shall also need to define a pair of *"even/odd digit-sums"* satisfying $s_b(\nu) = s_b^1(\nu) + s_b^0(\nu)$ written symbolically as

$$s_b(\nu) = \sum_{j=0}^{l_b(\nu)-1} \eta_b\left(\nu; kb^j, (b-k)b^j\right) \tag{15a}$$

$$s_b^\sigma(\nu) = \sum_{j=0}^{l_b(\nu)-1} \sum_{k=1}^{b-1} \overline{\eta}\left(\nu; kb^{2j+\sigma}, (b-k)b^{2j+\sigma}\right), \sigma \in \{0,1\} \tag{15b}$$

We shall also define another auxiliary sequence as $\delta^{10} s_b(\nu) = s_b^1(\nu) - s_b^0(\nu)$ .As these are relevant for the SDS analysis of section ***5***, we are interested in resolving them based on their internal regularities which have a natural recursive structure and self-similarity. These allow their uniform deployment via concatenative operators and simple reproducing maps $\{K_1,\ldots,K_m\}$ acting on an initial core sequence s_0 as

$$s_n \leftarrow \left[s_{n-1}, K_1(s_{n-1}),\ldots, K_m(s_{n-1})\right]$$

It is then possible to resolve their individually computable expressions via a set of standard elementary number theoretic functions where modulo operations capture periodicities and different types of scaling are used for repetitive indices. Using these techniques one can isolate successive subsequences until the complexity of the original sequence is exhausted. Applying this technique into the uniform evaluation of both $s_b(\nu)$ and $\delta^{10} s_b(\nu)$ sequences shows that they can be interpolated with the formulas

$$s_b(\nu) = \mathrm{mod}(\nu, b) + \left\lfloor \frac{\nu}{b} \right\rfloor - 2\,\mathrm{mod}\left(\left\lfloor \frac{\nu}{b^2} \right\rfloor, b \right) \tag{16a}$$

$$\delta^{10} s_b(\nu) = \mathrm{mod}(\nu, b) - \mathrm{mod}\left(\left\lfloor \frac{\nu}{b} \right\rfloor, b \right) + \mathrm{mod}\left(\left\lfloor \frac{\nu}{b^2} \right\rfloor, b \right) \tag{16b}$$

We have heretofore, described a toolbox that shall be of aid in an effort to make a full transcription of *SAT* problems into their arithmetized forms where further analytical manipulation can be made possible.

3. Hierarchical classification of SAT problems

Next, we proceed to a hierarchical classification of both Conjuctive and Disjunctive Normal Forms (CNF/DNF) for any SAT problem which is to be applied at any level of the ICH defined in the sense of the previous section. To this aim we have to introduce an intermediate assignment map as follows. Let us assume a set of n_C "slots" for each clause over L total positions and a set of n_A arbitrary Boolean variables referred to as the "atoms" of a CNF/DNF expression to be assigned to each slot accommodating a number of $\{n_\mathrm{i}\}$ positions conforming to one of the integer partitions of L

$$L = \sum_{i=1}^{n_C} n_i$$

Then any CNF/DNF proposition can be abstracted to a set of n_C parentheses as {(x,y,…),(z,…),…,(w,…)} before the specific order of logical operators being applied for evaluation which is effectively, only one more bit. Such a case corresponds to a diagrammatic network characterized by a specific connectivity matrix under the additional constraint that each map of input atoms to the literals codomain of any clause being non-injective and surjective but with many-to-one connections disallowed. The presence of negation operators in the translation of atoms to literals ('x' vs '~x') inside any clause can then be taken care of by the use of a weighting factor {-1,+1} in any connection arrow from atoms to slots.

Given any particular connectivity diagram we may then discriminate between different cases with the following characteristics. For any set of

n_C slots we can have either a 1-1 mapping with no overlap between clauses or a redundant mapping where several initial atoms are used in multiple positions in different clauses. Any set of expressions can also be parametrized by the number of negative weights so that for any expression we can have up to 2^L distinct "*negation codes*" which can be encoded as integers including the identity of atoms and literals (0-code).

To properly characterize any given SAT instance we need to separate between instances of exactly k distinct atoms in all clauses (k-SAT) or less. To this aim we introduce a total overlap ratio $\lambda_i = n_A / L, \;\; 0 < \lambda_i \leq 1$ where n_A the number of input atoms with the case λ =1 corresponding to a unique 1-1 assignment. This particular parameters serve to define an amount of non-uniqueness in the set of unique Boolean variables entering any such expression with the symmetric ones being given for λ = 1. Notably, due to a well known Local Lemma of Lovasz [37], any k-CNF formula is satisfiable if every one of its clauses has a common overlap in at most 2^{k-2} positions or equivalently, the same number of non-zero entries in a connectivity matrix rows. Recent work by Moser [38] as well as Moser and Tardos [39] revealed a randomized algorithm based on a new lemma according to which there is always some constant k_0 such that when the overlap does not exceed 2^{k-k0} the k-CNF problem is satisfiable and a satisfying assignment can be found in polynomial time $\tau \sim p(n_C)$.

We can now separate SAT instances into at most four distinct classes which allow their parametrization to be given in terms of appropriate indicator functions and integer codes over any member of a combinatorial hierarchy. Specifically, we may discriminate between (*a*) 1-1 k-SAT expressions with equal length clauses, (*b*) 1-1 SAT with unequal clauses, (*c*) k-SAT with overlapping, equal clauses and, (*d*) SAT with overlapping and unequal clauses. The additional parametrization of each expression with a Boolean negation code can be given as an independent dimension of each problem via a separate integer encoding. This allows the direct visualization of the influence of all possible negation codes as a 2^L x 2^L Boolean matrix. In order to make this algorithmically tractable we need to devise a method of applying each expression as an appropriately parametrized operator to all the elements of a W_L matrix as defined in (3) or (4) for each separate negation code corresponding to a row of the final matrix. To this aim, we utilize the weight encoding of the original

assignment map which can be brought to a strict binary format in $\{0,1\}^{L}$ with each one denoting the presence of a negation operator in the associated literal.

To provide a complete integer encoding for the previous classification we start from the two 1-1 classes (*a*) and (*b*) with the observation that for each total length L for any interval S_L, the set of all possible clauses coincides with the set of integer partitions of L, hence for each and every member $\{S_L, W_L\}$ of the associated binary ICH, the number of possible SAT expressions grows according to the well known integer partition function $P(v)$ [25] (sequence A000041 in Sloan's OEIS). If we use the notation $\{v\}$ for the set of tuples $\{n_1,\ldots,n_k\}$ comprising all members of the set of partitions and $\{v\}_k, 0 < k \le v$ for any particular member of k elements from an ordered list then it is possible to provide a fully arithmetized code for each member of a SAT class with L total atoms as a unique pair $(L, \{L\}_k)$ mapping from the set of all such tuples to the integers

$$\{L\}_k \cong \sum_{i=0}^{k} n_i L^i,\ 0 < n_k < \ldots < n_0,\ \ n_0 = L-1 \qquad (17)$$

For instance, the partitions of 4, given as {3+1, 2+2, 2+1+1, 1+1+1+1} are mapped to the sequence $\{4\} \to \{7,10,22,85\}$ with the maximal element always giving the trivial, all ones partition in class (*a*). Notice that an inverse sorting of n_i resulting in a different subset of pairs of partitions per base L. In figure *3* we show the result of computing all such valid encodings for different bases $L > 2$ for both sorting types. The same code can be used as a partition generator under the constraints of monotonicity and non-zero coefficients as in (14). We also checked the inverse method of producing SAT codes from each partition with an independent partition generator based on block counting revealing a similar distribution as in figure(3). Details of the methods are reported in section ***6***. The above method characterizes uniquely and exhaustively all SAT expressions of the two first classes corresponding to any particular member of a hierarchy, the length of any particular partition member given also as the L-ary decoding of the associated integer code. Hence, all members of the classes (*a*) and (*b*) can be derived from the tuple $\{\{L\}_k, \mu\}, \mu \in S_L$ where μ stands for the associated negation code. We mention in passing that the presence of fractality in the set of integer

partitions has been recently proved by Folsom *et al.*[40], as well as Brunier *et al.* [41]. A uniform indicator function over any exponential interval becomes now a map $\{L\}_k \times S_L \to \{0,1\}$.

For the last two classes (*c*) and (*d*), the numbers of n_A input atoms, n_C clauses, and separate populations of literals per clause $\{p_i^C\}_{i=1}^{n_C}$ are independent parameters. In such a case, the connectivity matrix of the assignment map ceases being a mere permutation. Let then, m_i stand for the multiplicity degree (number of one bits) in any column of the n_A x L connectivity matrix in the associated assignment map from atoms to literals. For κ such degenerate rows with $n_A - \kappa$ single atom inputs taking into account the structure of map from atoms to literals not allowing many-to-one connections, every m_i must count a unique literal so that we must have

$$\sum_{i=1}^{n_C} n_i = (n_A - \kappa) + \sum_{i=1}^{\kappa} m_i = L \tag{18}$$

This simplifies using the overlap ratio λ as

$$\sum_{i=1}^{\kappa} m_i = (1-\lambda)L + \kappa \tag{19}$$

We can now associate members of the (*c*) and (*d*) classes with the set of partitions $\{L+\rho\}, 0 < \rho < n_A$ where we have introduced the parameter $\rho = n_A - \kappa$ as a new dimension of the total search space. Notice that in the case of $n_A = \kappa$, we have just two different partitions of L which is still an incomplete description. To find a complete mapping to any level of $\{S_L, W_L\}$ we need to complete a tuple $\{\{L\}_k, \{L+\rho\}, \mu\}, \mu \in S_L$ with an additional encoding sufficient to cover all the information of a degenerate assignment map uniquely identifying each and every row of the relevant connectivity matrix. Each and every such row maps an atom to one or more literals. If all such rows are interpreted as separate, distinct integers $\{v_i\}$ then, the previously introduced multiplicities are associated as $m_i = s_2(v_i)$ while every such integer is sampled from the corresponding

combinatoric subset $\binom{L}{m_i}$ in the relevant expansion of 2^L when unconstrained. W can now prove that for any such matrix ***w*** and a pair of an *1* x *L* vector ***v*** with elements $v_i = 2^i$, $I = 0,\ldots,L\text{-}1$, and a n_C x *1* vector **1,** the identity $\mathbf{1} \cdot \mathbf{w} \cdot \mathbf{v} = 2^L$ holds.

The proof is a direct consequence of the map from atoms to literals in each clause corresponding to rows and their integer codes v_{i}. By the same token that led us to the identity (17), each separate row of any such random matrix must not have one bits in the same column for all corresponding columns. Moreover, the previous property of no common one bits is essentially the same as the requirement in App. *A* on the Hamming distances between all $\kappa(\kappa\text{-}1)/2$ pairs satisfying

$$\left\{h\left(\nu_i,\nu_j\right)=\left\|s_2(\nu_i)-s_2(\nu_j)\right\|=\left\|m_i-m_j\right\|\right\}_{\{i,j\}}^{\kappa(\kappa-1)/2} \tag{20}$$

Hence, addition of any pair of such integers increases the sum of digits additively as already mentioned in App. *A*. As a result, any integer codes for the rows of a connectivity matrix must be a subset of the restricted 2-integer compositions of 2^L-1. We shall denote the maximal subset of these partitions as $\left\{\left\{2^L-1\right\}\right\}$.

It is then in principle possible to construct a sieve using (20) as a filter which given two partitions of *L* as $\{L\}_C \times \{L+\rho\}_m$ the first being used to decode the number of clauses n_C and the second for deriving the multiplicities m_i. The filter can be used to extract only the relevant partitions out of the $\left\{\left\{2^L-1\right\}\right\}$ subset. The most direct way for exhaustively enumerating the subset of all possible matrices can be given through a special indicator function over the interval $S_{n_A L}$ where strings are interpreted as a sequential concatenation of each such matrix rows, each of them being valid when it satisfies the condition

$$\sum_{k=0}^{n_A L} 2^{\mathrm{mod}\left(k,n_A\right)}\overline{\eta}(\nu,k)=2^L-1 \tag{21}$$

These can be further classified according to the number of isolated periods (powers of 2) in each substring of length n_A. Results for the case of such a classification are shown in figure ***4*** for an example of $n_A = 3$ and $L = 4$**.** A complete identification of the last two classes can then be given with a tuple $\{\{L\}_C, \{L+\rho\}_m, \{\{2^L-1\}\}, \mu\}$. In Table 1, we review the basics of the classification presented here with some additional properties that shall be proven in the next section where we also complete the examination of appropriate indicator functions for all the classes separately.

4. Universal Indicator Functions

We proceed with the construction of analytical formulas of appropriate indicator functions for the arithmetized version of any CNF/DNF expression. To begin with, we shall ignore overlaps and consider only the first two classes *SAT0-1* following the nomenclature of Table 1. Evaluation of any similar expression is a two stage process, the first part being the assignment mapping atoms to literals in a sequence of n_C parentheses $(\sigma_j,...,\sigma_k), j = p_{i-1}+1, k = p_i$ followed by evaluation of each. At the second stage, bits from each clause evaluation shall form a new word of n_C bits for final evaluation. Each parenthesis can then be interpreted as a binary expansion associated with a unique new integer $\mu_i, i = 1,...,n_C$ such that all clauses can be arithmetized. The complete arithmetization of the input expression will then be given by the total code $\nu_S = \mu_1 + 2^{p1}\mu_2 + ... + 2^{S(pi)}\mu_{nc} \in S_L$ where we used $S(p_i)$ to denote the partial summands of all clauses atom populations with the convention $S(n_0) = 0$. Negations can be encoded as separate integer codes or logical masks translating the original $\{\pm 1\}$ weights of the assignment map into {0, 1} codes and identify them as a new unique integer. Overlap codes can also be given as separate integers with ones indicating all pairs of logical variables with the same index. In this section we provide a pure analytical expression for indicator functions over any S_L in the absence of negations and overlaps restricting attention to the relevant classes *SAT0* and *SAT1*, while the rest shall be treated separately in the next section with one additional special indicator associated with unsatisfiability.

Each clause number μ_i has to be processed separately so that we need an inverse map for extracting the whole tuple from any integer in S_L via the below equivalent operators

$$\sigma(\nu;i,j)=\mathrm{mod}\left(\left\lfloor\frac{\nu_S}{2^i}\right\rfloor,2^{j-i}\right)=\sum_{k=i}^{j}2^k\overline{\eta}\left(\nu_S,k\right),\ 0\le i<j\le l_2(\nu_S) \quad (22)$$

Processing each clause integer separately requires finding an arithmetic alternative for a recursion of the form $x_{n+1}\leftarrow x_n\circ\sigma_m, x_1=\sigma_j, m=j+1,...,k$. To this aim, we introduce a generalization of the parity function which is normally derived from the sequential exclusive disjunction of successive bits in any binary word, introducing a "*C-parity*" and a "*D-parity*", denoted as $p_\vee(\nu), p_\wedge(\nu)$ which can be typically written as

$$p_\vee(\nu)=\prod_{i=0}^{l_2(\nu)-1}\overline{\eta}(\nu,i),\quad p_\wedge(\nu)=\prod_{i=0}^{l_2(\nu)-1}\left(1-\overline{\eta}(\nu,i)\right)$$

The second form results from the algebraic expression of the logical identity $x\vee y=\neg\left(\neg x\wedge\neg y\right)$ which when applied simultaneously to all bits in an expansion turning *D*-parity to an equivalent *C*-parity for total complements. It is then trivial to show that for any interval S_L the only non-zero value of $p_\vee(\nu)$ is a Mersenne number 2^L-1 as any other pattern contains zeros which falsify the specific clause. Similarly, only the zero word satisfies $p_\wedge(\nu)$. To construct a global indicator function over any S_L we shall then need two auxiliary characteristic functions over the integers as

$$\chi_C(\nu,l)=\begin{cases}1, & \nu=2^l-1\\ 0, & 0\le\nu<2^l-1\end{cases},\quad \chi_D(\nu,l)=\begin{cases}0, & \nu=0\\ 1, & 0<\nu\le 2^l-1\end{cases}$$

An equivalent analytic form of a Mersenne number detector can be given as $\chi_C(\nu,l)=\lfloor\nu/(2^l-1)\rfloor$ and the use of complements makes χ_C sufficient for both cases. We then straightforwardly identify $p_\vee(\nu)$ with $\chi_C\left(\nu,l_2(\nu)\right)$ and $p_\wedge(\nu)$ with $\chi_D\left(\nu,l_2(\nu)\right)=1-p_\vee(\overline{\nu})$. The use of χ_C only is suggested from the crisp truth values over any interval allowing the use of a Mersenne number detector.

The previous definitions allow us to derive analytical expressions for the two main indicators in a purely functional approach by noticing that the second processing stage is simply $p_{\vee}\left(\{p_{\wedge}(\mu_i)\}_{i=1}^{nc}\right)$and $p_{\wedge}\left(\{p_{\vee}(\mu_i)\}_{i=1}^{nc}\right)$for the *CNF* and *DNF* cases respectively, resulting in the following explicit forms for any $\nu_s \in S_L$

$$\mathbf{1}_{S_L}^{(CNF)}(\nu_S) = \chi_C\left(1-\sum_{i=1}^{n_C} 2^i \chi_C\left(\beta_i - \sigma(\nu_s; S(p_{i-1}), S(p_i), p_i), n_C\right)\right) \quad (23a)$$

$$\mathbf{1}_{S_L}^{(DNF)}(\nu_S) = 1-\chi_C\left(a_C - \sum_{i=1}^{n_C} 2^i \chi_C\left(\sigma(\nu_s; S(p_{i-1}), S(p_i), p_i), n_C\right)\right) \quad (23b)$$

$$a_C = 2^{n_C}-1,\ \ \beta_i = 2^{p_i}-1 \quad (23c)$$

The forms in (23) can be further simplified by noticing that it is sufficient for each separate indicator per clause to be one for the *CNF* case or their summand to be nonzero for the *DNF* case. Using also the cumulant summand $S(p_i - 1)$ of all clause values reduced by one to fit into the form of the σ operator of (22) leads in the expressions

$$\mathbf{1}_{S}^{(CNF)}(\nu_S) = \left\lfloor \frac{1}{n_C}\sum_{i=1}^{n_C}\left\lfloor \frac{1}{\beta_i}\bmod\left(\left\lfloor \frac{\nu_s}{2^{S(p_i)}}\right\rfloor, 2^{p_i}\right)\right\rfloor\right\rfloor \quad (24a)$$

$$\mathbf{1}_{S}^{(DNF)}(\nu_S) = \sum_{i=1}^{n_C}\left\{1-\left\lfloor 1-\frac{1}{\beta_i}\bmod\left(\left\lfloor \frac{\nu_s}{2^{S(p_i)}}\right\rfloor, 2^{p_i}\right)\right\rfloor\right\} > 0 \quad (24b)$$

Since (24a-b) contains only summands of single bits and all divisions correspond to single bit shifts the overall complexity depends mainly on the *modulo* function. Current implementations for arbitrarily large integers and infinite precision have a logarithmic complexity such that we can propose an estimated total complexity of the above indicators of the order of $\sim \mathrm{O}\left((\log \nu)^{n_C}\right)$ apart from some minor overhead. We notice that there is a kind of "resonance" mechanism involved in the evaluation of (21) in that every exponential interval represents a full "*phase coherency interval*" for the underlying counters while any intermediate uniform

column of any W_L will also have partial coherency intervals at positions given by a sequence like $k2^{i+j}, 0 < k \le 2^{L-j-1}$ for any intermediate column in bit positions $(i,...i+j), i, j < L$. Falsification of the total expression can only result as a loss of all coherencies between the n_C individual indicators.

5. Negations, overlaps and unsatisfiability

Next we move to examine the significance of the additional negation and overlap arithmetized codes where any such code distorts the original coherencies of the $\{S_L, W_L\}$ hierarchy thus acting as a cut-off filter by exclusion of certain subsets and nullification of the original indicator functions in the remaining allowed sub-intervals. To this aim, we keep the original direct association of any SAT expression clauses with the rows of a W_L array without removing the redundant variables. By doing so we choose to work again with an overall search space of 2^L instead of $2^{\lambda L}$, which is to be restricted with the introduction of some intermediate maps $M(\nu)$over any S_L interval acting as logical masks in the uniform direction and filtering out the appropriate subset. These can be computed for any specific overlap ratio $\lambda < 1$, and any particular assignment map under the demand that specific columns of the W_L must conform to the condition of certain bits being identical at different significance levels for each column. We keep the negation as a mapping between indices and apply both operations as a total mapping $\nu \to \overline{M}_{\mu}(\nu) = R_{\oplus}(\mu, \nu \mathbf{1}_S^O(\nu))$ where μ stands for the arithmetized negation code using the bitwise formalism introduced in section ***2*** and the product with the overlap indicator discarding indices out of the particular sub-class.

Assume then $m_i - 1$ independent pairs of literals each with a degree of degeneracy d_i, among all clauses that have to be constrained with the disjunction or conjunction of a total of logical conditions of the form $\{x_i = x_j\}, 0 < i, j \le L$ applied in every individual column of any W_L. This is equivalent to the not-complement of the exclusive disjunction of any two bits which leads to the additional logical mask as a product over all the numbers of pairs in the form

$$\mathbf{1}_S^O(\nu) = \prod_{\{i,j\}}^{N_p} \left(1 - \left\| \overline{\eta}(\nu, i) - \overline{\eta}(\nu, j) \right\|\right), \nu \in S_L \tag{25a}$$

$$N_p = \frac{1}{2}\sum_{i=1}^{\kappa} d_i(d_i - 1) \tag{25b}$$

Since, any uniform deployment turns different significance levels into periods this can be redefined as a total synchronization function or multiple bitwise conjunctions between remote periods for each literal. The above can be greatly simplified with m_i bitwise conjunction operators acting on each integer index in any S_L via a set of arithmetized overlap codes. These are to be extracted by the direct translation of the associated m_i binary rows in the connectivity matrix of the original assignment map into their integer values as $\left\{\kappa_i^O\right\}_{i=1}^{m_i}$. Both one bit and zero bit coincidences can be incorporated utilizing the internal mirror symmetry of the total S_L interval to obtain

$$\mathbf{1}_S^O(\nu, m_i^{\{d_i,l_i\}}) = \left\lfloor \frac{1}{m_i}\sum_{i=1}^{m_i-1}\left\{R_\vee\left(\kappa_i^O, \nu\right) + R_\vee\left(\kappa_i^O, \bar{\nu}\right)\right\}\right\rfloor \tag{26}$$

Having completed the construction of the appropriate indicators and the necessary logical masks, we proceed with the construction of Global Truth Tables (GTT) as 2^L x 2^L matrices corresponding to a map $S_L \times S_L \times S_L \to \{0,1\}$ where the second S_L domain will contain all possible assignments of atoms to literals, the associated new integers $\mu \in S_L$ standing for the *"negation code"* and the third one containing the relevant overlap codes so as to cover the additional *SAT2* and *SAT3* classes. We use σ = 0, 1 for *CNF* or *DNF* forms from now on. A definition comprising all the necessary problem data can be given as a functional composition of the form

$$\mathbf{T}^{(\sigma)} = \mathbf{1}_{S_L}^{(\sigma)}\left(R_\oplus\left(\mu, \nu\mathbf{1}_{S_L}^O\left(\nu, m_i^{\{d_i,l_i\}}\right)\right)\right) \tag{27}$$

We shall readily show that each such matrix is an ordered object with the main information contained in the first "*root*" raw corresponding to μ = 0, the rest being simply permutations.

The uniform permutation classes guarantee that for all possible negation codes on the same instance of any SAT expression the initial number of truth values is conserved thus leaving any true values untouched.

Let $f(v)$: $Z \rightarrow \{0,1\}$ be any Boolean function and consider the composite $f\left(R_{\oplus}(\mu,\nu)\right)$. For μ=1, the action of any negation code is trivial consisting of swapping permutations between adjacent bits of the output of f for any uniform deployment. The block analysis of f in any S_L shall evidently remain invariant if no block counts start and/or end at odd positions with every swapping taking place internally to each block otherwise the block vector length shall increase. Any such permutation depends on the binary expansion of μ and commutes always a block of numbers of exponential order analogous to the periods in the μ expansion. Thus, for any block structure to contain an invariant block at position i, the partial summands must satisfy

$$\sum_{n=1}^{i-1} |\, c_n \,| > 2^{l_2(\mu)},\ |\, c_i \,| < 2^{l_2(\mu)}$$

Using any large random binary vectors one can verify numerically that the above results in a fractal structure for any matrix over all μ in any S_L. We conclude that all SAT expressions in the first two classes must be satisfiable if they are satisfiable for μ=0 and the addition of any negation code is in fact a redundant operation. It is only after imposing additional constraints due to overlapping clauses that satisfiability is threatened.

Next we observe that the additional overlap filter in (26) is increasingly restrictive as any term of lower periodicity is trimming the total interval by excluding certain subsets much like the recursion leading to a Cantor set but in more irregular ways. This in turn reveals a simple criterion for unsatisfiability immediately retrievable by inspection of the connectivity matrix structure in the assignment map. Specifically, we recall that all overlap codes and their binary expansions are by construction made so as to form a restricted integer composition of $2^L - 1$, having no common one bits. Whenever the overlap codes cover all significance levels of $2^L - 1$ the trimming filter in (26) will exclude all input indices in any S_L interval apart from the zero and ones words of length L. Hence, we conclude that the sole responsible for complete unsatisfiability of any *SAT* expression in the last two classes is the case of the overlap codes leaving no non-empty subset for applying the indicators in (23) or (24).

Both the interaction of periodic functions like η, as well as the trimming operators of (26) always results into multi-periodic structures which then act as filters on a similar structure produced by the universal indicator functions making satisfiability a kind of "resonance" effect where one or more periods preserve a degree of coherency between them. These effects pointing again to a certain relation with dynamical models and BDS implied a particular reduction technique in the next section adding more flexibility in handling arbitrary SAT expressions by turning them into trajectories of a special class of dynamical systems.

6. Sequential Dynamical Systems and clause equalization

We now attempt to bring every SAT instance for all four classes under a common format of a special class of discrete dynamical systems which will allow processing all literal values in tandem, a mathematical analogue of parallel processing via a compact, arithmetic fractal object defined as the bitwise operators of section ***2***. Notably, this class belongs to the more general case of so called, Sequential Dynamical Systems (SDS) of which an in depth analysis can be found in [43] and [44]. The latter also defines the important notion of permutation complexity which here takes a particularly simple form. The transcription requires an indirect relation with the previously introduced hierarchy in order to use the bitwise operators along different clauses and for this we will have to bring all clauses to an equivalent encoding of equal length, a feat which can be accomplished with different protocols for each class as follows.

Let S_m be the interval associated with $m = \max(p_i)$, the maximal clause length. Then we ask to find a method for reducing any unequal clauses without overlaps to equal clauses and the same for unequal, overlapping clauses. Following the nomenclature reciewed in Table 1, we have the following two cases

1. *SAT1 → SAT0:*

   If we consider only positive weights for the assignment map, atoms and literals are identical and the numbering of input variables allows them to be separated into n_C distinct groups of N literals with $L = n_A = mn_C$. All possible inputs can then be put into a direct correspondence with a sequence of integers $\{\nu_1,...,\nu_{nc}\} \in S_m^{nc}$.

For the case of unequal clauses of *SAT1* we can use the invariance of the truth values $\mathbf{1} \vee (x_1 \vee ... \vee x_k) = x_1 \vee ... \vee x_k$ to show that we can replace all missing entries for a maximal clause in every literal with units until we have $N = \max(n_i)$ for all clauses. To give an example of the procedure we take an arbitrary CNF formula as for instance $x_1 \wedge (x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$. In Table 2 we show the resulting reduction to the maximal number of three bits per clause. Notice that the order of appearance of X_3 and X_4 is irrelevant when taken uniformly and it could be exchanged so as to have all atom indices sorted. Evidently, the resulting S_N^3 bitwise product of integers for such a case can be filtered so that only certain bands inside each interval will be used with a logical mask stabilizing some of their bits. Logical masks for the excluded subspaces can be easily precomputed out of the symmetrized 2^9 search space by noticing that the particular sorting of significant levels chosen in Table 2, for any instance of SAT1, since all atom indices are distinct can always be brought in front of the one blocks so as to be bounded as $2^{P\min} - 1 \le \nu_i \le 2^{P\min + p_i} - 1, P_{\min} = m - p_i$ where p_i the original number of unique atom indices in any clause.

2. <u>*SAT3 → SAT2:*</u>

Any such one-to-many assignments can be accommodated with the demand of certain intermediate bits of the encoding integers being identical in any interval S_N or the equivalent method of (26) and (27) for building a logical mask. With regards to the previous example we may now write a similar formula for such cases as $x_1 \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3)$. Again, we show the corresponding reduction scheme in Table 3.

With the aid of the bitwise operators we can now construct a basic recursive formula for a universal Conjunctor following the definitions and identities of section ***2***, where we also use an additional bit σ for *CNF* (0) and *DNF* (1) as

$$U_0(\nu_1,...,\nu_m) = R_\vee(\nu_m,,...,R_\vee(\nu_2,\nu_1)) \tag{28a}$$

We can similarly write a complementary universal Disjunctor directly.

$$U_1(\nu_1,...,\nu_m) = R_\wedge\left(\nu_m,...,\overline{R}_\wedge\left(\nu_2,\nu_1\right)\right) \tag{28b}$$

Satisfiability is now translated in the following conditions for the *m*-step trajectories while the same conditions hold for each individual step

$$U_0(\nu_1,...,\nu_m) = 2^m - 1,\ \ \exists\,\nu_i \in \{\nu_1,...,\nu_m\} : \nu_i > 0 \tag{29a}$$

$$U_1(\nu_1,...,\nu_m) > 0,\ \ \{\nu_1,...,\nu_m\} = 2^m - 1 \tag{29b}$$

Inclusion of non-positive weights for literals is directly applicable via the same methods used in the previous section. We notice the complementarity between *CNF/DNF* conditions which appear extremely restrictive on the set of either the terminal or the initial conditions respectively. We can now reformulate the answer to the general satisfiability problem as the existence of a basin of attraction of each of the two complementary domains for all trajectories starting from their complement. This can also be interpreted with conditions (28a-b) taken as a sequence of binary filters for the uniform expressions of the $R_{\wedge,\vee}$ associated matrices. In figure *7*, we show the form of the bitwise conjunctive operator for both conditions of (28) while the equivalent form for the bitwise disjunction is trivial with only one zero or non-zero position at the boundary value $R(2^m\text{-}1,2^m\text{-}1)$. Since, the problem is answerable for both *SAT0* and *SAT1* classes; the reduction performed via the equalization process only leaves the *SAT2* case to be explored.

To understand the particular type of dynamics associated with (28a-b) we recall that both bitwise conjunction and disjunction can be reduced algebraically into the single bitwise difference (exclusive disjunction) operator $R_\oplus = R_\vee - R_\wedge$ which has been proven equivalent to uniform permutations.

The system of (28a-b) has a unique and interesting property that the total map can be split into a linear combination of a pair of simpler “local” and a non linear, “non-local” map, of which the first additive one acts “locally” in any individual trajectory while the second, is conservative only when taken as a permutation in the “non local” uniform direction. In

App. *A*, we discuss how the second part could also be localized by replacing the difference operator with a restricted 2-integer partition of one of the two arguments as $(\nu_{i+1}+\nu_i \pm \kappa_{10}(\nu_{i+1},\nu_i) \mp \kappa_{01}(\nu_{i+1},\nu_i))/2$ but no efficient sieve has been found as yet for choosing the appropriate localizing pair out of the total set.

We observe that the permutations are controlled in a different manner depending which of the two arguments is used as the equivalent negation code so that given a pair of vectors **v** = [0,…,2^m-1] we can write the equivalent expressions for the uniform row-wise or column-wise action of $R\oplus$ respectively

$$R_{\oplus}(\nu_{i+1},\nu_i)=\left(\prod_{j=1}^{s_2(\nu_{i+1})} P[\overline{\eta}(\nu_{i+1},j)]\right)\cdot \mathbf{v}=\left(\prod_{j=1}^{s_2(\nu_i)} P[\overline{\eta}(\nu_i,j)]\right)\cdot \mathbf{v}$$

The matrices $P[\overline{\eta}(\nu,j)]$ are permutation matrices of order controlled by the successive bits in the binary expansion of each argument and each method differs in complexity unless $s_2(\nu_i)=s_2(\nu_{i+1})=k$ when the two integer expansions belong to the same combinatoric subgroup $\binom{m}{k}$. We can still choose all trajectories to conform to a minimal complexity definition by switching between the alternative branches of (30). Then the total permutation complexity for any trajectory as defined in [44] is given by the frequencies of different permutation classes which are here directly analogous to the digit-sum. Since Shannon entropies are also analogous to the same quantity for binary expansions, the overall permutation complexity for all trajectories is analogous to $H(\{\nu_i\})$.

Since the totality of (28) is a conservative operation for any uniform deployment, one can consider an alternative expression of (28a-b) as $(\nu_{i+1}+\nu_i \pm \nu_k)/2$ with k here being a dummy index running on all possible triplets with the constraint that the first 2-summand must belong to the same congruency (*mod* 2) with ν_k, a property which was built in the original definition of the bitwise operators by construction. We can then reconstruct a hierarchical classification of all possible trajectories with the aid of the higher radix hierarchies defined near the end of section ***2***, as

$$\{S_m^{(b)}, W_m^{(b)}\} : b_{\max} = 2^P, \left|S_m^{(b)}\right| = b_{\max}^m, P = l_2\left(\max\{p_i\}\right)$$

for any *SAT* expression with *m* equalized clauses of maximal atoms population. Since the new intervals and dictionaries are defined on powers of *2*, they are "resonant" with an equivalent binary dictionary $S_{mP}^{(b)}$. This makes any total encoding of a *SAT* expression as $\nu_S = \nu_1 + ... + \nu_m b^{m-1}$ equivalent to a simple concatenation of all v_i binary expansions.

Next, we create a new mapping Π_σ from any number of equalized clauses to any trajectory of the form of (28a-b) as an index interpolation scheme of the form

$$\nu_s \to \nu_0, \nu_1, \nu_2, \nu_3 ... \xrightarrow{\Pi} \mu = \nu_0', \nu_1', \nu_2', \nu_3', \nu_4', \nu_5', ... \tag{30a}$$

$$\nu_2' = R_\sigma(\nu_0, \nu_1),\ \ \nu_3' = \nu_2,\ \ \nu_4' = R_\sigma(\nu_2', \nu_2), ... \tag{30b}$$

The new mapping associates any member of any original hierarchy with an extended one as

$$\Pi_\sigma : \left\{S_j^{(b)}, W_j^{(b)}\right\}, j = 2, ... \to \left\{S_k^{(b)}, W_k^{(b)}\right\}, k = 2j - 1$$

The representation in (30a-b) is privileged in that any answer to any *SAT* expression becomes directly visible in the uniform deployment in the new enlarged alphabet. Thus for any two clauses, the answer can be directly "read" from the third digit, for three clauses we get the answer at the fourth digit, and so on. For any number of *j* clauses, the extended sequence will contain exactly *j*-1 steps of the original recursions in (18a-b). The simplicity of the above scheme allows derivation of the original conditions (29a-b) in the simpler symbolic forms for any total codes $\nu_S \in S_j^{(b)}, \mu_\sigma(\nu_S) \in \Pi_\sigma\left(S_j^{(b)}\right)$ as

$$\eta_b\left(\mu(\nu_S), 2j-1\right) = 2^P - 1, \{\nu_i\}^{(b)} > 0 \tag{35a}$$

$$\eta_b\left(\mu(\nu_S), 2j-1\right) > 0, \{\nu_i\}^{(b)} = 2^{jP} - 1 \tag{35b}$$

The conditions can be compacted into a compound, universal Boolean discriminant as

$$U_{\sigma}(\nu_S)=\Theta\left(\frac{\sigma-(-1)^{1-\sigma}\left\lfloor 2^{2P(1-j)}\mu_{\sigma}(\nu_S)\right\rfloor}{2^{P}-1}\right) \tag{36}$$

The above immediately provides an effective algorithm for locating all possible answers given any method by which the Π_{σ} map would be at least uniformly if individually computable. A more technical analysis of the algebraic structure of the original recursion and its relation with the sequences (32a-b) is given in App. *B*.

Finding an efficient, fast sieve for Π_{σ} appears to be of paramount importance for the generic *SAT* problem given the above redefinitions of the original problem where the overall *SAT* complexity reduces to the study of the irregularities in the block analysis of the two $\mu_{\sigma}(\nu_S)$ sequences. Another step to this direction can be made by noticing once again the significance of restricted *2*-integer compositions of any ν_s in any relevant interval which can be described with the aid of three additional integers' κ, λ, μ in the same interval with the expansions

$$\begin{aligned}
&\kappa \xrightarrow{\eta} \nu_1\nu_2 0\nu_4 0\nu_5 ...\\
&\mu \xrightarrow{\eta} 00\nu_3 0\nu_5 0..\\
&\lambda \xrightarrow{\eta} 00R(\nu_1,\nu_2)0R(R(\nu_1,\nu_2),\nu_4)0...
\end{aligned} \tag{37}$$

Obviously any integer in the same interval can be analyzed as $\nu = \kappa + \mu$, $\nu \in S_m^{(b)}$ with a subset for which $\mu = \lambda$, standing for the ν_s subset.

A direct method to construct a simple numerical sieve can be given through the product of three indicators representing the constraints in (37) where the first separates all κ strings that are ($mod\ 2^{jp}$) for all indices $j = 2k+1$, k =1,2,…, the second extracts the complements that are simultaneously ($mod\ 2^{ip}$) for all indices i = 0, 1 and $i = j$ - 1, while the third examines the equality of μ and λ. We extracted the *2*-composits for some simple cases as in figure *8*. There is ample visual evidence for increased symmetry and correlations despite the fact that each integer is examined as an independent entity. Moreover, one recognize that the simultaneous application of the constraints in (32b) are equivalent with a

product of independent indicators for each of them and can be computed separately for further block analysis. Doing so, results in periodic repetitions of block sizes, alternating as $\ldots,|c_i|,-|c_{i+1}|,\ldots$, satisfying $|c_i|+|c_{i+1}|=2^k$ for some integer k. No formal proof is known at the moment for this or similar properties.

To explore further, we experimented with this sieve in several bases. An example of such a subset indicator for the cases of 4 clauses and maximal length 3 (octal) as well as 3 clauses and length 4 (hexadecimal) is shown in figure 9 via its block analysis where all valid sequences and their indices appear in isolated small islands with the *CNF* and *DNF* cases complementary. The resulting sequences are not entirely regular and they vary with the alphabet base while no single interpolating formula is known to the author. Using the wavelet based methods developed in [45] and [46], the Hurst exponents were extracted from the block sequences of the resulting indicators and a detrending fluctuation analysis was performed using the methods developed in [47]. A characteristic feature of all similar block counting structures is that they always comprise almost stationary positive and strictly non-stationary negative subsequences respectively. Analyzing them separately show the same increasingly correlated structure with a characteristic exponents of the order of a_{H} with an associated Haussdorf dimension estimated as $2 - a_{\mathrm{H}}$ for the whole sequence and the positive and negative parts respectively as in Table 5. The existence of persistent correlations in any uniform deployment shows the inheritance of the underlying order of the counters discrete time flow which leads to an important conjectured principle of non-independence of arbitrary integers in the case of symbol-wise morphisms being also one of the main reasons for departing from purely probabilistic treatments in this work. This is further discussed in the last section.

7. SATbox: a MATLAB® toolbox for exploring SAT problems

Our effort in all previous sections was to provide methods that could be useful from a purely functional programming standpoint. We also provide prototype code for experimentation in the github account *github.com/rtheo/SATbox*. The package contains a number of basic utilities reviewed in Table *6,* for the *SAT* as well as other similar

problems that can be examined in the context of the combinatorial hierarchies and the resulting sequences block analysis. It should be stressed that this prototype demo serves only as an example of the programming and analytical techniques explained here and it is not a true production code for expressions with thousands or millions of variables. Such would demand both special infinite precision codes as well as at least an MPI protocol for efficient parallelization where several sequences of which the uniform deployment is required for filtering should be broken into large chunks. Moreover, any true speed up should be based on the efficient use of permutations and solely arithmetic operations for extracting the appropriate indicators in large alphabets.

The main codes are given by the functions *sat* containing an appropriate interface for transcription of a standardized format into the arithmetized codings presented in ***3*****,** *satassign* which performs the assignment map by explicit construction of the associated connectivity matrix performing also the necessary arithmetization of rows for overlapping variables if present, and the core evaluation function *sateval* with internal calls of the *UIeval* function realizing the indicators of (25)-(26) plus the *trimmer* function that delivers subsets of any total S_L interval whenever the *satassign* routine reports the presence of overlaps. The *sat* routine reads a simplified file format without distinction between *DNF/CNF* expressions since both are treated equally by the *sateval* function which delivers back whole GTTs. This is given as a set of rows representing atom indices with different clauses represented by different rows written in separate lines. Any negation operators are denoted with a minus sign in front of any individual atom index. Certain example files and the resulting circulant fractal GTT matrices are offered with the package in the github account.

There are two additional Boolean flags immediately after the input filename that *sat* currently understands. The first is for discriminating between the cases of a single expression evaluation followed by its block analysis for a particular user specified negation plan if 0, or computing a whole GTT in which case any negation signs are ignored if 1. The last, is for choosing between two alternative methods of computation the first being the standard indicator functions of (25), (26) if 0 or the *satsds* applying the equalization protocol if 1. At the current version, only individual expressions are computed via the SDS method.

The additional *satnegation* offered in the utilities folder, is an interface for the *unixor* routine which implements the uniform bitwise difference (XOR) operator as a set of successive permutations and is in fact equivalent with the internal built-in *bitxor* function. We notice that the particular structure of swapping permutations associated with negations is in fact equivalent to separate phase shifts of each counter in the uniform dimension of each constructor of a binary word dictionary W_L and as such they could alternatively be implemented in a similar manner as the one used in identity (4) of section ***2*** to express the difference between the original η function and its complement.

8. Discussion and Conclusions

We presented a new, methodological framework for the treatment of complex problems based on the complementarity of lexicographically ordered word dictionaries which allow trading between individually computable formulas and a multi-periodic, self-similar totality resembling a discrete time flow of a sequential dynamical system, and which in certain cases appears to being able of significant reduction of the complexity of requested computations. This in turn, allows an argument on a kind of relativity of complexity which may appear different when a totality of objects is treated in a holistic frame. This comes from a very simple, fundamental observation that in many cases of discrete structures when these can be resolved as mappings from the integers to the integers, expressions for obtaining the totality of a codomain might exhibit lower complexity than those for obtaining or "choosing" particular subsets, yet finding the same subsets becomes also easier only after resolution of the totality. This also brings about the issue of "locality" which we discuss later on with a concrete, mechanistic example.

The author believes that the intuitive approach introduced here, is complementary to standard approaches in a manner akin to the way analytical geometry is complementary to the standard Euclidean one. The method when applied into the satisfiability problem, leads naturally to certain purely functional, analytical expressions for a set of universal characteristic functions totally describing the subset of all solutions and opening new avenues in the search for optimized algorithms. It is the hope of the author that future research in this direction will permit to visualize the totality of SAT problems in a particularly elegant and useful

way given more specialized codes than the elementary examples presented herein.

Since, there exist at least two analytical reformulations of any SAT based either on special indicator functions or the equalization protocol and the SDS presented in the last section, it is possible to expand this theme by further studying the fractality and the possible regularities of the resulting sequences in future work for real time savings in actual applications. A symbolic algorithm for efficient interpolation of fractal sequences following methods as those explained in the previous sections is under investigation. Notably, certain kinds of indicators as for instance, the case of $\Pi\left(S_{2m+1}^{(b)}\right)$subset show an increasing irregularity when moving to higher alphabet bases. This suggests the possibility of using Machine Learning methods as possible predictors for large samples of similar sequences. Notably, many such methods can be considered as purely functional machines due to their use of convex polytopes as a means to approximate true indicator functions of subsets of solutions.

An important issue with possible applications in parallel computations concerns the observations of the last section for certain mappings of the associated dynamical system reductions exhibiting simultaneously a kind of both local and non-local behavior along their discrete time evolution where large scale permutations intervene across the uniform discrete time flow axis mixing “future” and “past” states. We stress the fact that no formal proof has been given for this particular case and it could turn out that there is always a possibility of interpolating with some contrived expressions that would resolve such non-localities, yet the example can be abstracted in a much more general and potentially useful way, using a particular class of parallel computing machines with a special “*pseudo-quantum*” protocol, the choice of the term being due to an intriguing similarity between this mechanism and the well known, Feynman interpretation [48].

Assume then, a set of interconnected simple processing cores to which a certain execution tree is delivered for multiple evaluations on a large domain. Assume also an internal, localized protocol shared by all cores estimating an objective cost function such that for as long as an individual computation is less costly, each core can choose not to share any data and

proceed on its own up to a point where the costs exceed some threshold in which case the particular core ceases to follow a specific path and instead, it immediately delivers control to a supervising mechanism that shares information from all the other same level nodes of the execution tree before returning control to individual cores. Moreover, one could add to the definition of the cost as used here, the avoidance in using encoders and decoders for accessing individual digit patterns at intermediate stages for any pre-specified radix.

Additionally, the observations at the end of section ***5***, relate to the particular classification of maps over the integers that led to the arihtmetization strategy as originally introduced in section ***2***. Given two distinct classes of algebraic and symbol-wise maps, the existence of any direct equivalence between members of the two delineates the existence of a special algebraic sub-class that can be used to affect similar changes into underlying patterns while still treating its inputs as "solid", indecomposable objects. The appearance of correlations in any uniform arrangement of seemingly independent entities which is in fact inherited from a "hidden" underlying flow could still be manifested under different cases of random samplings thus making this sub-class important in any physical applications that could hide equivalent pattern based computations. This possibility is also in accord with a previous proposal for turning back to analog machines with non-Leibnizian architectures [16]. The prospect of building analog SAT machines then presents an appealing possibility for hybrid A.I. and general cyber-physical systems [49] with the internal capacity of automated proofs and it will be further explored in future work.

References

[1] S. Cook, *"The complexity of theorem proving procedures"*, Proc. 3rd Ann. ACM Symposium, Theory of Computing. (1971) 151–158.

[2] L. Levin, *"Universal search problems"*, in "A survey of Russian approaches to *perebor* (brute-force searches) algorithms". *Annals of the History of Computing*. **6**(4), (1984) 384–400.

[3] E. Giunchiglia, A. Tacchella, (2004) "Theory and Applications of Satisfiability Testing". Lecture Notes in Computer Science. **2919**. E. Giunchiglia, A. Tacchella. eds.

[4] Y. Vizel, G. Weissenbacher, S. Malik, "*Boolean Satisfiability Solvers and Their Applications in Model Checking". Proc. IEEE*. **103**(11) (2015).

[5] T. E .Raptis, "*'Viral' Turing Machines, Computation from Noise and Combinatorial Hierarchies*" (submitted) (2017). Preprint: arxiv:1702.06000 [cs.AI].

[6] P. Hudak, "*Conception, evolution, and application of functional programming languages*", ACM Comp. Surveys, **21**(3), (1989), 359 – 411.

[7] A. Sabry, "*What is Purely Functional Language ?"*. J. Functional Programming. **8**(1), (1993) 1–22.

[8] V. N. Vapnik, (1989). *Statistical Learning Theory*. Wiley-Interscience.

[9] V. N. Vapnik, (2000). *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer-Verlag.

[10] A. Feraz, "*Polytopes as vehicles of informational content in feedforward neural networks"*, Phil. Psych., **29**(5), (2015).

[11] A. Kolmogorov, "*On Tables of Random Numbers*".Sankhyā Ser. A. 25 (1963) 369–375.

[12] Kolmogorov, A.N. (1965). *"Three Approaches to the Quantitative Definition of Information*". Problems Inform. Transmission. **1**(1): 1–7.

[13] M. Li, P. M.B. Vitányi, (2009). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Science & Business Media.

[14] P. L. Bartlett, S. Mendelson *Rademacher and Gaussian Complexities: Risk Bounds and Structural Results*. J. M. L. Res, **3**, (2002) 463-482.

[15] S. Shalev-Shwartz, S. Ben-David, (2014). *Understanding Machine Learning - from Theory to Algorithms*. Cambridge univ. press.

[16] T. E. Raptis, "*Spectral Representations and Global Maps of Cellular Automata Dynamics*", **91**, (2015), 503 – 510.

[17] E. Clarke, O. Grumberg, K. McMillan, X. Zhao. "*Efficient generation of counterexamples and witnesses in symbolic model checking.*" Proc. of Design Automation Conf., 1995.

[18] F. DeMarco, J. Xuan, D. Le Berre, M. Monperrus, *"Automatic Repair of Buggy If Conditions and Missing Preconditions with SMT*".Proc. 6th Int. Workshop on Constraints in Software Testing, Verification, and Analysis (CSTVA 2014).

[19] J. Argelich, C. M. Li, F. Manya, J. Planes, "*Analyzing the Instances of the MaxSAT Evaluation.*" SAT 2011: 360-361

[20] J. Argelich, C. M. Li, F. Manya, J. Planes, "*Experimenting with the Instances of the MaxSAT Evaluation.*" CCIA 2011: 31-40

[21] J. Argelich, C. M. Li, F. Manya, J. Planes, "*The First and Second Max-SAT Evaluations.*" J. Sat., Bool. Modell. Comp., **4** (2008), 251-278.

[22] D. Raghavarao, Damaraju, L.V. Padgett, (2005) *Block Designs: Analysis, Combinatorics and Applications.* World Sci.

[23] J. Daintith, E. Wright, (2008) *"A Dictionary of Computing"*, 6th ed., Oxford Univ. press.

[24] M. E. Jane (2008), *The Chicago Guide to Writing about Numbers*, Chicago Univ. Press

[25] G. E. Andrews, K. Eriksson (2004). *Integer Partitions*. Cambridge Univ. Press.
[26] R. De Wolf, "*A brief Introduction to Fourier Analysis on the Boolean Cube*", Theory of Computing Library, Graduate Surveys, TCGS (2008), 1-20.

[27] R. O'Donell, (2014) *Analysis of Boolean Functions,* Cambridge Univ. Press.
[28] P. J. Grabner, T. Herendi, R. F. Tichy, *"Fractal Digital Sums and Codes."* Appl. Algebra Engrg. Comm. Comput. **8**, (1997) 33-39,
[29] H. DeLonge, H. Delange, Sur la fonction sommatoire de la fonction "Somme des Chiffres",Enseign.Math.(2)21(1975)31-47.

[30] E. Grosswald, *"Rademacher Functions"* in Encyclopedia of Statistical Sciences, (2006) Wiley Online Library, DOI: 10.1002/0471667196.ess2155.pub2.
[31] J. L. Walsh, "*A closed set of normal orthogonal functions*". Amer. J. Math. **45** (1923) 5–24
[32] N. J. Fine, (1949). "*On the Walsh functions*". Trans. Amer. Math. Soc. **65** (1949) 372–414.
[33] H. Peitgen, H. Jurgens, D. Saupe, (2004) "*Chaos and Fractals"*, 2nd ed., SpringerVerlag, NY.
[34] Konvalina, "*Combinatorial fractal geometry with a biological application*", Fractals, **14** (2006) 133-142.
[35] M. J. Patitz, S. M. Summers, "*Self-assembly of discrete self-similar fractals*", Natural Computing, **9**(1), (2010) 135-172.

[36] V. Garcia-Morales, "*Fractal surfaces from simple arithmetic operations*", Physica A **447**, 535 (2016).

[37] N. Alon, J. H. Spencer, (2000). *The probabilistic method* (2nd ed.) Wiley-Intersci.

[38] R. A. Moser. "*A constructive proof of the lovasz local lemma*." In STOC '09: Proc. 41st annual ACM symp. on Theory of computing, NY, (2009) 343–350.
[39] R. A. Moser and G. Tardos. "*A constructive proof of the general lovasz local lemma.*" J. ACM, 57(2):1–15, 2010.
[40] A. Folsom, Z. A. Kent, K. Ono, "ℓ-Adic properties of the partition function.", Adv. Math, **229**(3), (2012) 1586-1609.
[41] J. H. Bruinier, K. Ono, "*Algebraic formulas for the coefficients of half-integral weight harmonic weak Maas from*.", Adv. Math. **246**(20) (2013) 198-219.
[42]C.Kimberling, "*Fractal sequences and interspersions*",. Ars Combinatoria. **45**,(1997). 157–168.
[43] H. S. Mortveit, C. M. Reidys, (2008) *"Introduction to Sequential Dynamical Systems"*, Springer Sci. NY.
[44] J. M. Amigo, (2010), *"Permutation Complexity in Dynamical Systems"*, Springer Series in Synergetics, Springer-Verlag, Berlin.
[45] I.Simonsen, A.Hansen, O.Nes, "*Determination of the Hurst exponent by use of wavelet transforms"*, Physical Review E 58, (1998) 2779-2787.
[46] R.Weron, I.Simonsen, P.Wilman (2004) *Modeling highly volatile and seasonal markets: evidence from the Nord Pool electricity market*, in "*The Application of Econophysics*", ed. H. Takayasu, Springer, 182-191.
[47] M. Little, P. McSharry, I. Moroz, S. Roberts (2006), "*Nonlinear, Biophysically-Informed Speech Pathology Detection"* in 2006 IEEE Int. Conf. Acoustics, Speech

and Signal Processing, 2006. ICASSP 2006 Proc., Toulouse, France. pp. II-1080- II-1083.
[48] R. P. Feynman, A. R. Hibbs, (1965) *Quantum Mechanics and Path Integrals*. NY, McGraw-Hill.
[49] S. K. Khaitan, J. D. McCalley, "*Design Techniques and Applications of Cyber Physical Systems: A Survey*", IEEE Sys. J., (2014).

Appendix A: swapping permutations and integer compositions

We notice that the negation operation can be written as an individual abstract formula of the form

$$\overline{\nu} = \nu + \kappa_{01}(\nu, \mu) - \kappa_{10}(\nu, \mu) \tag{A1}$$

The essence of (A1) is that in the general case, different one bits of μ will affect simultaneously both zeros and ones of the v expansion allowing to split μ in two integers κ_{01} and κ_{10} forming an appropriate subset of the restricted 2-integer compositions $\mu = \kappa_{01} + \kappa_{10}$. Since negation is here equivalent with the bitwise XOR operator we also find from () in section the complementary action of the bitwise AND operator as

$$R_{\otimes}(\mu, \nu) = \frac{1}{2}\left(\mu - \kappa_{01}(\nu, \mu) + \kappa_{10}(\nu, \mu)\right) \tag{A2}$$

Eliminating each pair member using the composition property we find the important constraint

$$2R_{\otimes}(\mu, \nu) = 2\left(\mu + \kappa_{01}(\nu, \mu)\right) = 3\kappa_{10}(\nu, \mu) \tag{A3}$$

There are certain important constraints on the choice of these pairs from the total of μ possible 2-decompositions including the degenerate ones $(\mu,0)$ and $(0,\mu)$ the most important being that of non-coincidence of bits in the expansion of any such pair leading to a natural equality of their Hadamard L_1 distance and their sum of digits as

$$h(\kappa_{10}, \mu - \kappa_{10}) = s_2(\mu) \tag{A4}$$

Using a known equivalent expression for the *lhs* and the additivity forced in the digit-sum for this particular case we also have

$$s_2\left(R_{\oplus}(\kappa_{10},\kappa_{10})\right)=s_2(\kappa_{10})+s_2(\kappa_{01}) \tag{A5}$$

Finding the roots of either leads naturally to another fractal matrix which can be given as one branch of a multi-labeled indicator introduced to separate the two degenerate cases as

$$J_2(\nu;\mu)=\begin{cases}+1, \sum\sigma_\nu^\mu = s_2(\mu)\\ 0, \sum\sigma_\nu^\mu < s_2(\mu)\\ -1, \sum\sigma_\nu^\mu = 0\end{cases} \tag{A6}$$

In (A6), the shorthand $\sum\sigma_\nu^\mu$ simply denotes the selective summand of μ-masked bits of ν. The middle branch then immediately identifies all mixed cases where a non-trivial 2-integer composition should apply. From the data of the resulting matrices it becomes evident that the trivial cases diminish with increasing μ. No interpolation scheme is known as yet to the author that would allow direct use of (A1) and no efficient sieve for individual computability of the appropriate pair for each ν to replace uniform permutation classes seems to exist. Any further progress shall be reported elsewhere.

Appendix B: Series expansion of the equalized clause recursions

To gain further understanding on the structure of the Π_σ map we examine the algebraic structure of any individual trajectories. To complete the derivation we shall need another intermediate map $\Pi_\oplus$ which we define as in (32a-b)

$$\nu_s \to \nu_1,\nu_2,\nu_3,\nu_4\ldots \xrightarrow{\Pi} \lambda=\nu_1',\nu_2',\nu_3',\nu_4',\nu_5',\nu_6',\ldots \tag{C1-a}$$

$$\nu_3'=R_\oplus(\nu_1,\nu_2),\nu_4'=\nu_3,\nu_5'=R_\oplus(R_\oplus(\nu_1,\nu_2),\nu_3),\ldots \tag{C1-b}$$

The essential difference in (C1) is that it does not contain the results of any recursion directly but allows deriving their algebraic form. Manipulation of the recursions in (30a-b) by repetitive evaluation of their algebraic forms leads to a type of series of dyadic fractions as shown in Table 4 where now the digits ν_i correspond to the *rhs* of (35a). These are again members of an extended $\left\{S_{2j-2}^{(b)},W_{2j-2}^{(b)}\right\}$ hierarchy for any initial

clause index *j*. In figure *9*, we show the block analysis of the relevant $\Pi_{\oplus}$ indicator.

Due to the regular appearance of all symbols with different exponential weightings it is preferable to rephrase them as summands over the fundamental sequences $\{s_b(\nu), s_b^1(\nu), s_b^0(\nu)\}$ which have been resolved in section ***2***. We can do so by writing the same expressions as differences of the form

$$2^{-2}(\nu_1+...+\nu_3)+2^{-1}(\nu_1+...+\nu_5)-2^{-1}(\nu_1+...+\nu_3)$$
$$2^{-4}(\nu_1+...+\nu_3)+2^{-3}(\nu_1+...+\nu_5)-2^{-3}(\nu_1+...+\nu_3)+2^{-2}(\nu_1+...+\nu_7)-2^{-2}(\nu_1+...+\nu_5)$$
……

Regrouping same terms leads to coefficients from common powers of 2 and can be denoted using the digit-sum function and the auxiliary notation

$$\nu\big|_k=\begin{cases}\nu, & ,\nu<b^{k-1}\\ \mathrm{mod}\left(\nu,b^{k-1}\right) & ,\nu\geq b^{k-1}\end{cases} \tag{C2}$$

to denote the number of significant symbols present as

$$\left(2^{-4}-2^{-2}\right)s_b(\nu\big|_3)+(2^{-2}-2^{-1})s_b(\nu\big|_5)+2^{-2}s_b(\nu\big|_7)$$

The *DNF* formulas can be treated similarly with the additional restriction that the first term has to be extracted due to a gap in signs so as to make full use of both even and odd digit-sums. Introducing the auxiliary function $\delta^{10}s_b(\nu)=s_b^1(\nu)-s_b^0(\nu)$ we can find inductively the terminal value of any trajectory for any set of *j* equalized clauses as a series expansion for any member of the set $\lambda(\nu_S)\in\Pi_{\oplus}\left(S_{2j-2}^{(b)}\right)$ in the form

$$U_0(\lambda)=\frac{1}{2}s_b(\lambda)-\delta_{CNF}(\lambda,j) \tag{C3-a}$$

$$\delta_{CNF}(\lambda,j)=\frac{1}{2^{j+1}}\sum_{n=1}^{j}2^n s_b\left(\lambda\big|_{2j}\right) \tag{C3-b}$$

$$U_1(\lambda)=\frac{1}{2}\delta^{10}s_b(\lambda)+\frac{\mathrm{mod}(\lambda,b)}{2^{2j}}-\delta_{DNF}(\lambda,j) \tag{C3-c}$$

$$\delta_{DNF}(\lambda, j) = \frac{1}{2^{j+1}} \sum_{n=1}^{j} 2^n \delta^{10} s_b\left(\lambda\big|_{2n}\right) \tag{C3-d}$$

Since, any terminal values of (37a) and (37c) by construction must have the same properties as those of (33a-b) we obtain the summation formulas

$$s_b\left(\lambda(\nu_S)\right) - \sum_{n=1}^{j} 2^{n+1} s_b\left(\lambda(\nu_S)\big|_{2j}\right) = 2^{j+2}\left\lfloor \frac{\mu_{\vee}(\nu_S)}{2^{P(2j-3)}} \right\rfloor \tag{C4-a}$$

$$\delta^{10} s_b\left(\lambda(\nu_S)\right) - \sum_{n=1}^{j} 2^{2j+n} \delta^{10} s_b\left(\lambda(\nu_S)\big|_{2j}\right) + \mathrm{mod}(\lambda(\nu_S), b) = 2^{3j+1}\left\lfloor \frac{\mu_{\wedge}(\nu_S)}{2^{P(2j-3)}} \right\rfloor \text{(C4-b)}$$

The relations (38) constrain the mapping between the two different sieves $\Pi_{\oplus}\left(S^{(b)}_{2j-2}\right)$ and $\Pi_{\wedge,\vee}\left(S^{(b)}_{j}\right)$.

Table 1

Classes	Clauses	Overlap	1-1	n_A/L	Connectivity Matrix	$\{S_L, W_L\}$ Satisfiability
(*a*) SAT0	Equal	0	1	1	Permutation	Full
(*b*) SAT1	Unequal	0	1	<1	Non-Square Permutation	Full
(*c*) SAT2	Equal	1	0	1	Square	Partial
(*d*) SAT3	Unequal	1	0	<1	n_A x L	Partial

Table 2: Clause Equalization

	2^0	2^1	2^2	$v_i \in S_N$
1st Clause	X_1	X_2	X_4	v_1
2nd Clause	1	X_3	X_5	v_2
3rd Clause	1	1	X_6	v_3

Table 3: Clause equalization with overlaps

	2^0	2^1	2^2	$v_i \in S_N$
1st Clause	X_1	1	1	v_1
2nd Clause	X_1	X_2	1	v_2
3rd Clause	X_1	X_2	X_3	v_3

Table 4: SDS recursive series

Recursion Depth	*CNF*	
	Expression over $W_{nc}^{(b)}$ *row*	*Index in* $S_{2m-1}^{(b)}$
2	$2^{-2}(v_1+v_2+v_3)+2^{-1}(v_4+v_5)$	$\nu = v_1 + \ldots + v_5 b^4$
3	$2^{-3}(v_1+v_2+v_3)+2^{-2}(v_4+v_5)+2^{-1}(v_6+v_7)$	$\nu = v_1 + \ldots + v_7 b^6$
	DNF	
2	$2^{-2}(v_1+v_2-v_3)+2^{-1}(v_4-v_5)$	$\nu = v_1 + \ldots + v_5 b^4$
3	$2^{-3}(v_1+v_2-v_3)+2^{-2}(v_4-v_5)+2^{-1}(v_6-v_7)$	$\nu = v_1 + \ldots + v_7 b^6$

Table 5: Hurst Exponents and Fractal Dimensions

		Block Seq.	*Positive Blocks*	*Negative Blocks*
(4, 3)	a_H	0.787	0.620	0.828
	2 - a_H	1.217	1.380	1.172
	D.F.A.	0.5619	0.630	0.605
(3, 4)	a_H	0.781	0.642	0.802
	2 - a_H	1.218	1.358	1.218
	D.F.A.	0.613	0.641	0.654

Table 6: SATbox utilities

Function	***Description***
blockanalysis	Block counting method for large binary sequences.
composits	Extracts 2-integer compositions of the recursive SAT-SDS sequences.
eta	The η function of section ***2***
ipart	Generator of integer partitions
isMersenne	Equivalent to *isShift*($x + 1$)
isShift	Locates all powers of two in any interval S_L
logb	*b*-ary logarithm of integers.
rotor	A special permutation map serving as individual decoder of any integer in arbitrary radix.
Rsymmetries	Used to find the symmetries of bitwise operators leading to (13) of section ***2***.
satnegate	Alternative realization of bitwise exclusive disjunction using permutations.
Sdb	Uniform deployment of the Digit-Sum function in arbitrary radix.
trimmer	Realization of the overlap filter (26) of section ***5***.
unixor	Core routine of *satnegate* for all swapping permutations.
WD	Word Dictionary constructor in arbitrary alphabet bases (η_b function, the binary version being equivalent to built-in *ff2n*)
WDserial	Alternative Dictionary constructor (Switching Automaton) avoiding large memory use and overflows

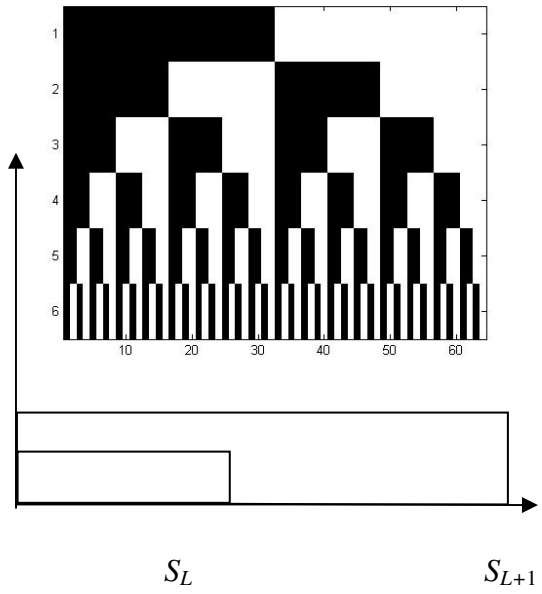


Fig. 1. Schematic of two consecutive members of an Inductive Combinatorial Hierarchy with an example of a lexicographically ordered L x 2^L Word Dictionary $W_{L=5}$ attached in every interval S_L.

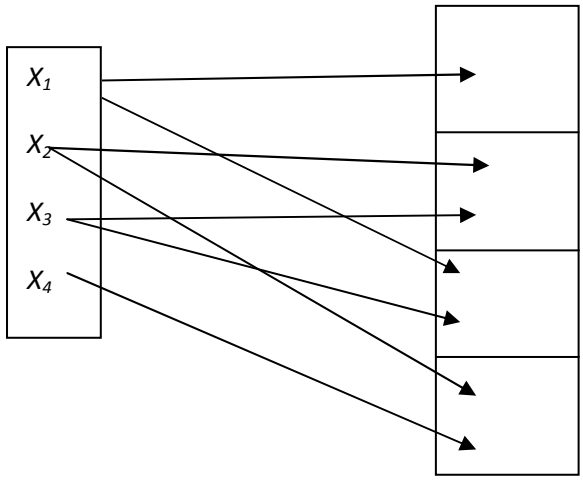


Fig. 2. Example of an Assignment Map from Atoms to Literals inside every clause in the 2-SAT sentence $\left(\left(x_1 \cdot x_4\right) \circ \left(\left(x_2 \cdot x_3\right)\right) \circ \left(\left(x_1 \cdot x_3\right)\right) \circ \left(\left(x_2 \cdot x_4\right)\right)\right)$ with the symbols $\{\cdot, \circ\}$ serving as substitutes for alternating conjuction and disjunction symbols for both *DNF/CNF* expressions. The set of slots on the right is in 1-1 correspondence with the columns of W_8 in a hierarchy like that of Fig. 1.

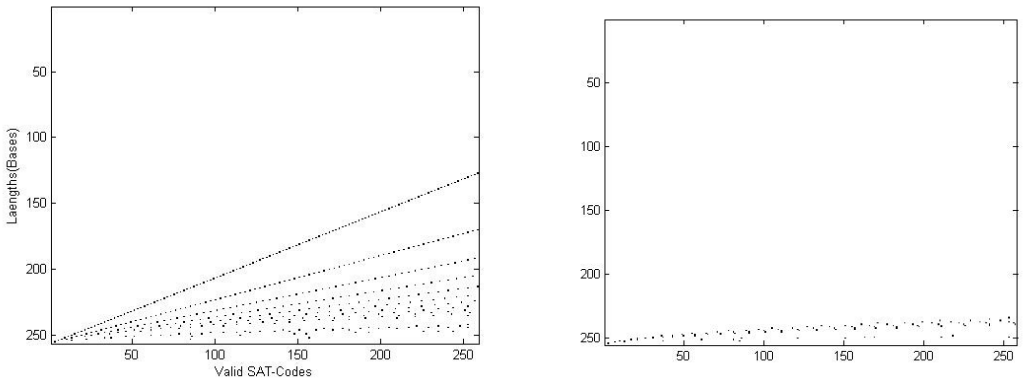


Fig. 3. The infinite subset of valid integer encodings of the two first SAT(0-1) classes with inverted colormap (black for 1, white for 0) limited in the $[0 \times 255]^2$ space.

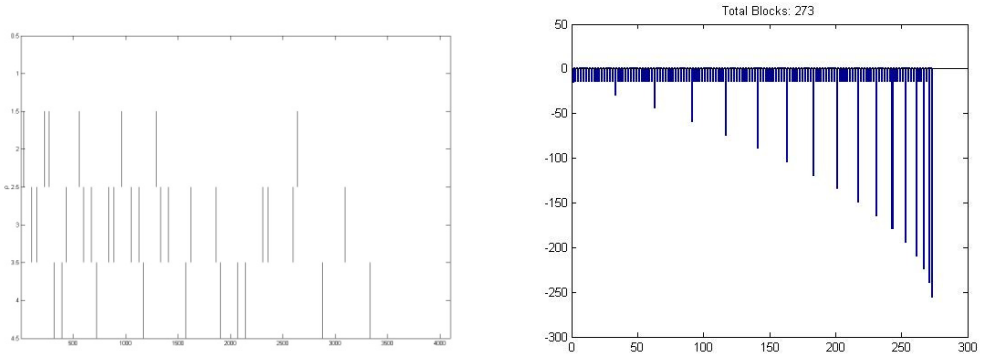
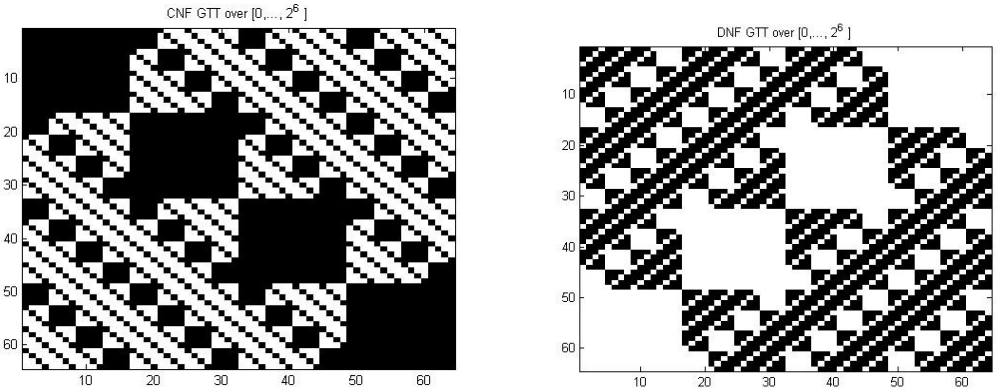
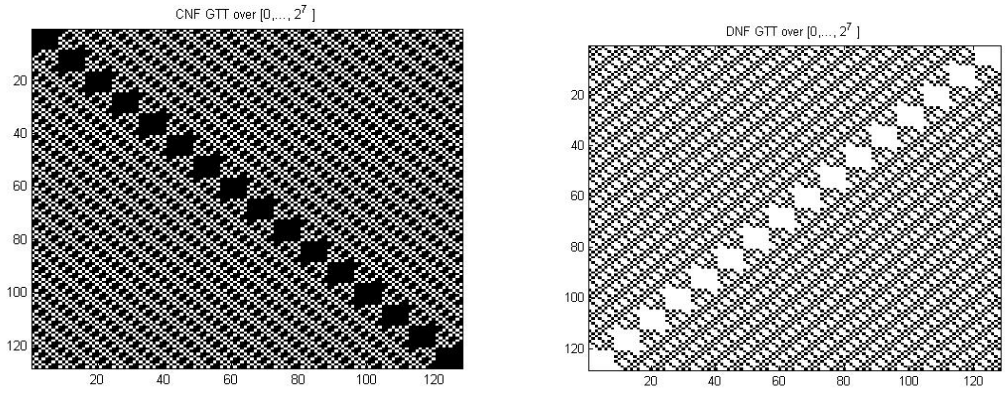


Fig. 4. (*a*) An example of 2-integer compositions as representations of connectivity matrices for the last two SAT(2-3) classes, further classified according to their complementary multiplicity parameter ρ.(*b*) Block analysis of the indicator function over S_{12}.



(a) (b)



(c) (d)

Fig 6. Examples of DNF and CNF truth tables for the *SAT0* and *SAT1* cases of ((1,2), (3,4), (5,6)) in (a)-(b) and ((1), (2,3), (4,5,6,7)) in (c)-(d) expressions respectively over all 64, 128, possible negation codes including zero words, showing certain repetitive regularities and fractality due to the underlying swapping permutations.

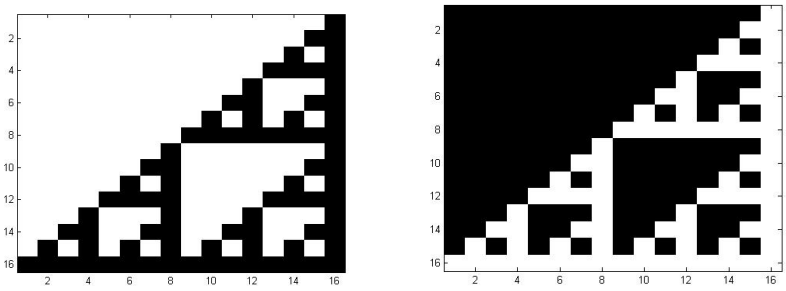


Fig 7. The χ_C filtered bitwise conjunction and disjunction matrix for conditions (27a) and (27b) respectively as not complements of each other.

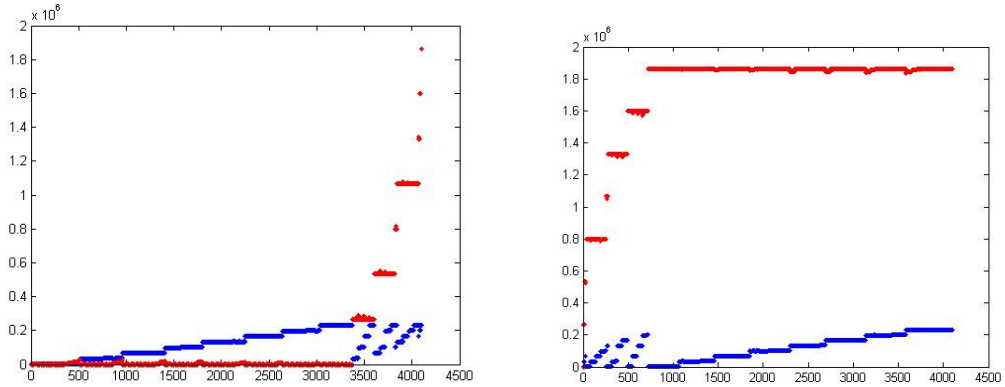


Fig 8. The 2-integer compositions of $\Pi\left(S_j^{(b)}\right)$ for the CNF with j = 4 and 3 clauses of maximal length of 3 bits in the extended base b = 8.

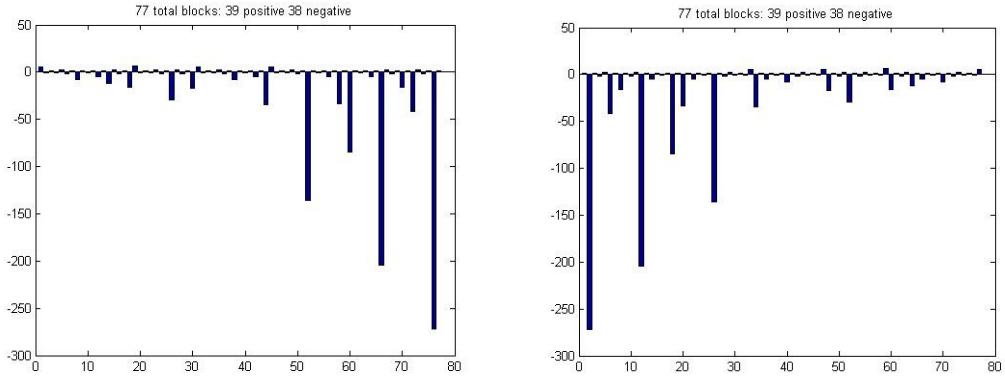


Fig 9. Complementary block analysis of the $\Pi\left(S_j^{(b)}\right)$indicators for the CNF and DNF problems respectively for 3 clauses of maximal length of 2 bits in the higher base b = 4.