Journal Academica

VOLUME 6, NO. 3, September 15 2016

N. Abdelwahab

JOURNAL ACADEMICA • NEW YORK

VOLUME 6, NO. 3, SEPTEMBER 15 2016

EDITOR IN CHIEF

S. Feigenbaum

TABLE OF CONTENT

Theoretical Computer Science

Elnaserledinellah Mahmoud Elsayed Abdelwahab

On the Dual Nature of Logical Variables and Clause-Sets

pp. 202-239

Copyright © 2016 JOURNAL ACADEMICA FOUNDATION • All rights reserved

ISSN 2161-3338 • www.journalacademica.org

Journal Academica Vol. 6(3), pp. 202-239, September 15 2016 - Theoretical Computer Science - ISSN 2161-3338
online edition www.journalacademica.org - Copyright © 2016 Journal Academica Foundation - All rights reserved

Full Length Research Paper

On the Dual Nature of Logical Variables and Clause-Sets

Elnaserledinellah Mahmood Abdelwahab*
Senior Project Manager, makmad.org e.V., Hanover (Germany)

Received August 4, 2016; Accepted September 12 2016

ABSTRACT

This paper describes the conceptual approach behind the proposed solution of the 3SAT problem recently published in [Abdelwahab 2016]. It is intended for interested readers providing a step-by-step, mostly informal explanation of the new paradigm proposed there completing the picture from an epistemological point of view with the concept of duality on center-stage. After a brief introduction discussing the importance of duality in both, physics and mathematics as well as past efforts to solve the P vs. NP problem, a theorem is proven showing that true randomness of input-variables is a property of algorithms which has to be given up when discrete, finite domains are considered. This insight has an already known side effect on computation paradigms, namely: The ability to de-randomize probabilistic algorithms. The theorem uses a canonical type of de-randomization which reveals dual properties of *logical variables* and *Clause-Sets*. A distinction is made between what we call the *syntactical Container Expression* (CE) and the *semantic Pattern Expression* (PE). A single sided approach is presumed to be insufficient to solve anyone of the dual problems of efficiently finding an assignment validating a 3CNF Clause-Set and finding a 3CNF-representation for a given semantic pattern. The deeply rooted reason, hereafter referred to as *The Inefficiency Principle,* is conjectured to be the inherent difficulty of translating one expression into the other based on a single-sided perspective. It expresses our inability to perceive and efficiently calculate complementary properties of a logical formula applying one view only. It is proposed as an alternative to the commonly accepted P≠NP conjecture. On the other hand, the idea of algorithmically using information deduced from PE to guide the instantiation of variables in a resolution procedure applied on a CE is as per [Abdelwahab 2016] able to provide an efficient solution to the 3SAT-problem. Finally, linking de-randomization to this positive solution has various well-established and important consequences for probabilistic complexity classes which are shown to hold.

Keywords: Algorithm, Arabic, Logic, Syntax, Semantics, Complexity, Dualities, 3SAT, Inefficiency Principle, P=NP, P=BPP

INTRODUCTION

Regarding the 3SAT-problem and its solution published in [Abdelwahab 2016] we find it appropriate to further detail the conceptual approach in a mostly informal way and follow up with its respective epistemological framework.
This paper is organized as follows: After a brief introduction about duality, its importance, application, and benefits for both, physics and mathematics, a short survey based on [Fortnow 2009] discusses different approaches to solve the NP problem, their

*Corresponding author: elnaser@makmad.org

weaknesses and epistemological grounds (if any). Furthermore, a theorem is proven showing the necessity to drop the hope for true randomness of input-variables in discrete, finite domain applications. This theorem shows that such an idealistic notion can lead to inconsistency in our perception of what a computable function is.
The notion of "true randomness" used here (*T-randomness*) is related to the fundamental inability of predicting patterns. This notion is different from the formal one used in mathematics, probability, and statistics where a random variable is an assignment of a numerical value to each possible outcome of an event space. The latter defines randomness to exist wherever probability distributions can be assigned to expected outcomes. The reason we needed to distinguish between those two notions is that they express fundamentally different phenomena: When a mathematical probability distribution exists, there is *only uncertainty* for a specific measurement, given in terms of the probability for that outcome, while T-randomness is related to a *cognitive boundary*, i.e., the non-existence of an algorithmic procedure for predicting pattern values. The respective theorem shown here uses canonical de-randomization techniques which reveal pattern natures of variables. In Section III, properties of those patterns are explored via concrete, straightforward examples. Section IV reformulates the intuitive intentions behind the belief that P≠NP in a more profound way. A conjecture, hereafter referred to as *Inefficiency Principle*, expresses our inability to efficiently translate complementary Pattern Expression (PE) and Container Expression (CE) properties of a logical formula into each other using a single view only. Section V describes the formal approach of [Abdelwahab 2016] in a more detailed manner. It is shown that pattern-oriented algorithms which always use relevant information from the PE side, succeed in forming small resolution-trees for the CE side. This section is intended as a guide for any reader to facilitate understanding proofs and conclusions of [Abdelwahab 2016]. In Section VI, known exponential lower bounds are re-interpreted according to the new paradigm, more specifically: The exponential lower bound obtained for FBDD construction in the *blocking sets problem of finite projective planes* case. The intrinsic duality between lines and points retained in the *k*CNF representation is shown to dissolve when the 3CNF-expression is used by the proposed Solver in [Abdelwahab 2016]. Eventually, Theorem 3 shows that all lower bounds obtained for FBDDs (using a common technique), fail the 3CNF-expression test when pattern oriented algorithms of [Abdelwahab 2016] are used. Finally, Section VII establishes with Theorem 4 some already expected consequences of P=NP on probabilistic complexity classes.

I DUALITY IN PHYSICS AND MATHEMATICS

Although the definitions of *duality* are not agreed upon, two types are usually referred to in modern scientific literature: Physical and mathematical duality. In the mathematical context, duality has numerous meanings, but is a very pervasive and important concept in modern mathematics and a fundamental general theme that has manifestations in almost every area of mathematics. It is commonly understood as follows:
"A duality, generally speaking, translates concepts, theorems or mathematical structures into other concepts, theorems or structures, in a one-to-one fashion, often (but not always) by means of an involution operation: if the dual of A is B, then the dual of B is A. {….} Many mathematical dualities between objects of two types correspond to pairings, bilinear functions from an object of one type and another object of the second type to some family of scalars. For instance, *linear algebra duality* corresponds in this

way to bilinear maps from pairs of vector spaces to scalars, the *duality between distributions and the associated test functions* corresponds to the pairing in which one integrates a distribution against a test function, and *Poincaré duality* corresponds similarly to intersection number, viewed as a pairing between submanifolds of a given manifold. From a category theory viewpoint, duality can also be seen as a functor, at least in the realm of vector spaces. This functor assigns to each space its dual space, and the pullback construction assigns to each arrow f: $V \rightarrow W$ its dual f^*: $W^* \rightarrow V^*$."[1]
In physics, duality describes the concept expressing that every phenomenon (e.g., elementary particle) exhibits facets of profoundly different views in the same time. It addresses the inability of classical concepts like "particle" or "wave" for example to quantum-mechanically describe the physically perceived world or, as *Albert Einstein* wrote with regard to physical light:

"It seems as though we must use sometimes the one theory and sometimes the other, while at times we may use either. We are faced with a new kind of difficulty. We have two contradictory pictures of reality; separately neither of them fully explains the phenomena of light, but together they do".

Its epistemological interpretation seems to be rooted in the reciprocal nature of human cognition and the nature of the physically perceived world [Daghbouche 2012]. How important is the interaction between physical and mathematical duality in modern science? The following citation from [Atiyah 2007] describes that:

"In the plane we have points and lines. Two different points can be joined by a unique line. Two different lines meet in one point unless they are parallel. People did not like this exception and so they worked hard and realized that if they added some points at infinity then they got what is called the projective plane in which the duality is perfect: the relationship between points and lines is perfectly symmetrical. This led to the classical principle of projective duality, which says that the "dual statement" of a theorem is also a theorem, so that we can talk about the dual theorem {….} One of the most exciting things that physicists have discovered is that in QFT there are many dualities which are not at all easy to understand geometrically. These theories are non-linear, and so they are not trivial. An important observation here is that if we have a given classical geometrical picture, then there is some kind of procedure, called 'quantization', by which we replace classical variables by operators to produce, with a bit of guesswork, a 'quantization' of the original theory. In this quantized theory there is a parameter that plays the role of Planck's constant and which allows us to recover the classical theory by letting it go to 0. Given a quantum theory, however, it may turn out that there are several ways in which it can be realized as the quantization of a classical geometry picture. For example, for a single classical particle we can use the position observable or the momentum observable (they are actually symmetrical), and so there are two ways of reaching the same theory from two geometrical points of view. Then these two points of view are called a duality. This is, of course, a very simple example, but it turns out that physicists have found many much deeper dualities which exist in complicated QFTs which link two very different-looking geometric pictures {…} Using this fantastic duality the physicists were able to calculate the numbers of rational curves of any degree on very simple examples of algebraic varieties of dimension 3. The

[1] Duality (mathematics). (2016, July 16). In Wikipedia, The Free Encyclopedia. Retrieved 09:52, July 17, 2016, from https://en.wikipedia.org/w/index.php?title=Duality_(mathematics)&oldid=730131312

formulae they got were so spectacular to algebraic geometers that at first they did not believe them, but eventually they were converted. Then they began a big industry that has produced many books on mirror symmetry. It is a whole new area in algebraic geometry that arises out of this particular simple example, just one example of duality in quantum theory."

The above insights have the following important relevance in the context of the P vs. NP problem: While the container nature of logical variables, the building blocks of Clause-Sets, comes (per definition) from their use as means to represent data not revealing any generically recognizable *value structure* of this data, *the enumeration of their possible truth-values in a truth table (de-randomization) lends itself to seeing them exhibiting patterns*[2]. The interplay between those dual sides of the nature of a variable (hence a Clause-Set) is the basis for the ideas presented here as well as in [Abdelwahab 2016]. Since it is conjectured below that efficient switching between those two sides is generally not possible without additional information, we call this conjecture *The Inefficiency Principle*.

II EFFORTS TO SOLVE THE P VS. NP PROBLEM

Following [Fortnow 2009], the latest serious trial to settle the P vs. NP question is a long-term algebraic geometry research plan (called Geometric Complexity Theory, GCT) which may be implemented targeting the separation of the two classes P and NP. In brief, this plan consists of the following steps[3]:

- Define a family of high-dimension polygons P_n based on group-representations of certain algebraic varieties. Roughly speaking, for each n, if P_n contains an integral point, then any circuit family for the Hamiltonian path problem must at least have size $n^{\log n}$ on inputs of size n, which implies P≠NP. Thus, to show that P≠NP it suffices to show that P_n contains an integral point for all n.
- Although it suffices to show that P_n contains an integral point for all n, it is argued that this direct approach would be difficult and it is suggested instead to first show that the integer-programming problem for the family P_n is in fact in P. Under this approach, there are three significant steps remaining: (1) Prove that the LP-relaxation solves the integer programming problem for P_n in polynomial time; (2) Find an efficient, simple combinatorial algorithm for the integer programming problem for P_n, and; (3) Prove that this simple algorithm always answers "yes."
- Since the polygons P_n are algebro-geometric in nature, solving (1) is thought to require algebraic geometry, representation theory, and the theory of quantum groups.
- Although step (1) is difficult, definite conjectures based on reasonable mathematical analogies that would solve (1) are provided. In contrast, the path to completing steps (2) and (3) is less clear.
- This approach has reduced a question about the nonexistence of polynomial-time algorithms for all NP-complete problems to a question about the existence of a polynomial-time algorithm (with certain properties) for a specific problem.

[2] Combinations in a combinatory space of discrete variables lose their non-predictive order the moment they get enumerated in a truth table. This has the effect that involved variables lose their T-randomness property as well as shall be seen further below.

[3] All efforts which are classified today under what is called "Quantum Computing" may be also understood as serious trials to use known physical processes/tools/intuitions to solve the NP problem.

This plan is one in a long series of attempts to prove P≠NP of which many failed for the following reasons (c.f. [Fortnow 2009]):

Diagonalization. Diagonalization requires simulation and we don't know how a fixed NP machine can simulate an arbitrary P machine. Also, a diagonalization proof would likely relativize, that is, work, even if all machines involved have access to the same additional information. It has been shown that no relativizable proof can settle the P vs. NP problem in either direction: Any diagonalization argument for P≠NP could trivially be modified to show that P≠NP relative to any oracle. But we already know of specific oracles for which P=NP and of other specific oracles for which P≠NP. Accordingly, any diagonalization argument would have to contradict one or the other of these. Complexity theorists have used diagonalization techniques to show that SAT cannot have algorithms that use both, a small amount of time and memory, but this is a long way from P≠NP.

Circuit Complexity. To show P≠NP it is sufficient to show some -complete problem cannot be solved by relatively small circuits of AND, OR, and NOT gates (the number of gates bounded by a fixed polynomial in the input size). In 1984, it has been shown that small circuits cannot solve the parity function if the circuits have a fixed number of layers of gates. In 1985, it was also shown that the NP-complete problem of finding a large clique does not have small circuits if one only allows AND and OR gates (no NOT gates). Later it was shown that those techniques would fail if one allows NOT gates. A notion of "natural" proofs was developed and gave evidence that our limited techniques in circuit complexity cannot be pushed much further. This can be shown assuming a widely believed conjecture on the existence of pseudorandom functions.

Proof Complexity. Resolution is a standard approach to proving tautologies of DNFs by finding two clauses of the form (ψ_1 AND x) and (ψ_2 AND NOT x) and adding the clause (ψ_1 AND ψ_2). A formula is a tautology exactly when one can produce an empty clause in this manner. In 1985, it was shown that tautologies that encode the pigeonhole principle do not have short resolution proofs. Since then complexity theorists have shown similar weaknesses in a number of other proof systems including cutting planes, algebraic proof systems based on polynomials, and restricted versions of proofs using the *Frege* axioms. But to prove P≠NP it has to be shown that tautologies cannot have short proofs in an arbitrary proof system. Even a breakthrough result showing tautologies don't have short general *Frege* proofs would not suffice in separating NP from P.

The latest proof making headlines claiming that P≠NP is attributed to [Deolalikar 2010]. One of the objections to the proof is the following:

"I have posted what I think is a definitive objection to any proof separating P from NP that relies on the structure of the solution spaces of SAT, and tries to argue that no problem in P has this kind of solution space. The basic idea is that every satisfiable formula’s solution space can be mapped to the solution space of a formula satisfied by the all-zeroes assignment, which is always trivially solvable. This map exactly preserves the number of variables and all distances between satisfying assignments. It is of course non-uniform, but that doesn’t matter: for every infinite collection of satisfiable formulas there’s an infinite collection of formulas satisfied by all-zeroes which has exactly the same solution space structure." [Williams 2010]

As per [Aaronson 2010], there are eight signs which give the impression that a P≠NP proof is wrong. Of those eight signs we have selected the following five for citation:

1. The author can't immediately explain why the proof fails for 2SAT, XOR-SAT, or other slight variants of NP-complete problems that are known to be in P. Historically, this has probably been the single most important "sanity check" for claimed proofs of P≠NP.
2. The proof doesn't "know about" all known techniques for polynomial-time algorithms, including dynamic programming, linear and semi definite programming, and holographic algorithms.
3. The paper doesn't prove any weaker results along the way: for example, P≠PSPACE, NEXP⊄P/poly, NP⊄TC^0, SAT requires super linear time. P vs. NP is a staggeringly hard problem, which one should think of as being dozens of steps beyond anything that we know how to prove today. So then the question arises: forget steps 30 and 40, what about steps 1, 2, and 3?
4. Related to the previous sign, the proof doesn't encompass the *known* lower bound results as special cases. For example: where inside this proof are the known lower bounds against constant-depth circuits? Where's Razborov's lower bound against monotone circuits? Where's Raz's lower bound against multi-linear formulas? All these things (at least the uniform versions of them) are implied by P≠NP, so any proof of P≠NP should imply them as well. Can we see more-or-less explicitly why it does so?
5. The paper lacks a coherent overview, clearly explaining how and why it overcomes the barriers that foiled previous attempts.

What about P=NP proof claims? To our knowledge, there are no serious comparative studies relating them, i.e., linking approaches and listing positive and negative sides of each one, lessons learned, etc.[4] As the consensus stands today: Many researchers refuse even to read such attempts. Most of those approaches present constructive algorithms[5] which have the following flaws[6]:

1. Focusing on a known NP complete problem: They mistakenly present solutions to special cases of P vs. NP or algorithms working only under certain assumptions which cannot be generalized.
2. Assuming some constructions used in their algorithms to be trivially solvable which doesn't turn out to be the case after thorough investigation. In fact: Many new solution approaches tend to be very tricky in "moving" the exponential behavior from one part of a known algorithm to an unintentionally hidden part of a new one.
3. And most importantly in the context of this paper: Positive solution attempts, similar to their negative counterparts (c.f. 5. above), don't give an intuitive explanation[7] for either the generic exponential behavior noticed using "usual"

[4] except for mere listings of selected papers and their refutations if existent as presented in the Web-page, e.g.: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm

[5] To our knowledge only few are non-constructive in nature and don't exceed a couple of pages.

[6] We are expressing drawbacks without providing examples to avoid *firstly* going into unnecessary length and *secondly* criticizing the work of specific known or unknown colleagues who surely deserve for their courage, time, and effort invested in such publications a lot of respect and acknowledgment for integrity and dedication.

[7] By intuitive we mean an explanation which can be rooted epistemologically.

algorithms or the special efficient behavior of the newly proposed algorithm. Known exponential lower bounds for special cases are not challenged and/or re-interpreted according to the new approach. *A scientific theory should always be able to answer the question: "Why?" in the most direct and clear terms, especially when it is confronted by a consensus against its fundamental assertions.*

Getting back to the mainstream opinion, we think that it is valid to question the following assertion:

Conjecture 1: No universal, uniform algorithm[8] or non-uniform family of algorithms can be found which efficiently solves all instances of an NP-complete problem, therefore: P≠NP.

Independent of any epistemological debate, the here exposed critics concern our perception of what a *Universal, Uniform Algorithm* (short: AU) really is[9] . Historically, an algorithm is a construction perceived for mathematics which paved its way into the physical world in terms of computing machines interacting with- and constituting algorithmically processed aspects of the physically real world (Fig. 1):

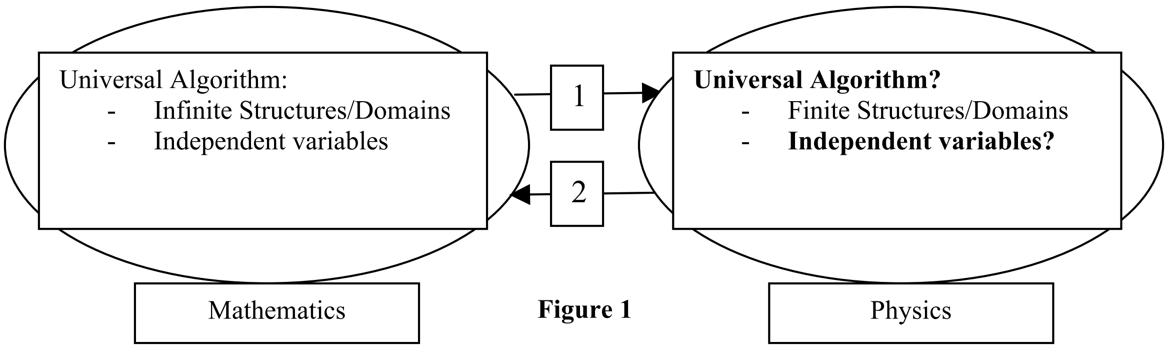

Figure 1

What happened to the concept of an AU during the transition from mathematics to physics? In our opinion, one aspect deserves attention: Infinite, countable or uncountable domains over which variables can range became finite, because real devices needed measurable approximations. In finite model theory finite structures (rather than finite domains) are the focus of investigation. It has been shown there that well-known properties of *First Order Logic* such as compactness for example, are lost when finite models are drawn [Vaeaenaenen 1993]. *We go here a level deeper and show that even for Propositional Logic (PL) it suffices to use discrete finite domains to lose T-randomness of involved input-variables and render extreme notions of universality, as the ones usually used to solve NP-complete problems, inappropriate.* The positive side of this discovery shall be the clue for the solution presented in [Abdelwahab 2016]. First some important definitions and a principle which shall be used further down:

[8] Universal, Uniform Algorithm: A generic, recursive way to solve all instances of a problem which is independent of the choice of a particular domain and/or its representation (i.e., universality not in the sense "simulating other arbitrary algorithms"). It is safe to talk about computable functions as well, because of the Church-Turing thesis (c.f. [Church–Turing thesis. (2016, August 25). In Wikipedia, The Free Encyclopedia. Retrieved 10:31, August 26, 2016, from https://en.wikipedia.org/w/index.php?title=Church%E2%80%93Turing_thesis&oldid=736126870]).

[9] Non-uniform algorithms are not subject to any investigation of this paper.

Definition 1: Consider two discrete random variables X and Y defined over a sample space S. We say that X and Y are *stochastically independent* if their probabilities fulfill: $P(X\in A, Y\in B)=P(X\in A)P(Y\in B)$, for all sets A and B.

Definition 2: A discrete variable X is called *computationally undetermined by variable set* $\mathcal{Y}$, if there is no algorithm/computable function f such that $X=f(Y_1,Y_2,Y_3,..Y_n)$, where $Y_i \in \mathcal{Y}$ is a discrete input-variable and any $Y_i<>X$[10]. Otherwise X is called *computationally determined by* $\mathcal{Y}$**.** If there is no variable set $\mathcal{Z}$ such that X is computationally determined by $\mathcal{Z}$, then X is said to be *computationally indeterminate.*

Turning our attention to what classic[11] AUs are in the extremist's point of view (universal view)[12]:

1- The unrestricted use of input/output variables in the following ways:
   a. We allow input-variables to range over ANY domains including those defined by the function itself.
   b. We allow input-variables to range over ANY values in those domains.
   c. Any finite computation using input-variables and yielding output variables is repeatable[13].
   d. Input-variables may be computationally dependent, independent or indeterminate[14]. We call this condition: *Container Property.*

[10] The condition $Y_i<>X$ is necessary to avoid circular definitions.

[11] Classical AUs: deterministic or non-deterministic, but always neither randomized nor probabilistic. Here are informal definitions of the latter two types: "A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behavior, in the hope of achieving good performance in the "average case" over all possible choices of random bits. Formally, the algorithm's performance will be a random variable determined by the random bits; thus either the running time, or the output (or both) is random variables. One has to distinguish between algorithms that use the random input to reduce the expected running time or memory usage, but always terminate with a correct result (Las Vegas algorithms) in a bounded amount of time, and probabilistic algorithms, which, depending on the random input, have a chance of producing an incorrect result (Monte Carlo algorithms) or fail to produce a result either by signaling a failure or failing to terminate. In the second case, random performance and random output, the term "algorithm" for a procedure is somewhat questionable. In the case of random output, it is no longer formally effective. However, in some cases, probabilistic algorithms are the only practical means of solving a problem In common practice, randomized algorithms are approximated using a pseudorandom number generator in place of a true source of random bits; such an implementation may deviate from the expected theoretical behavior." [Randomized algorithm. (2016, August 30). In Wikipedia, The Free Encyclopedia. Retrieved 10:42, September 1, 2016, from https://en.wikipedia.org/w/index.php?title=Randomized_algorithm&oldid=736945906]

[12] Extremist because it can be argued that one or more of the postulated conditions a.-d. may be relaxed as shall be seen further below.

[13] In probabilistic machines for example, it is easy to design algorithms/functions that do not always give the same output regardless of the random branch of computation that is followed.

[14] To explain why this last property is, ideally, allowed for input-variables of a Universal Algorithm, in the classical sense: Suppose $f(X=a,Y=b)$ calculates the value $Z=c$ in the best possible way. We want to make sure that this solution is also the most generic, data-independent one so that it can be used for all other possible combinations of values of X, Y. If a, b are functionally dependent (i.e., we can calculate them from each other), then f could use this dependency implicitly (as a sub-routine of f for example) to create an algorithmic "short-cut" to the solution $Z=c$ which might not work for other values of X, Y. Although this is allowed: We tend to enforce writing f in such a way to discard all solutions using special properties of values a, b (including any structural information which may be found either in a or in b) or

2- Studies in the foundations of mathematics have shown that point 1-a. above is the cause for intrinsic inconsistency and known anomalies. They lead to Russell's theory of types [Whitehead 1963] as well as to the incompleteness results of *Goedel*. Therefore, this condition has been relaxed to prohibit the use of domains/types which are defined by the function in question.
3- In practical terms we call, e.g., the function

```
Fakt(N): -
    If N=1 return 1
    Else return N*Fakt(N-1)
```

an universal calculator of factorial numbers, because the input-variable N can (ideally) span all values of the set N (condition 1-b.), the infinite set of natural numbers, and any value of N when given to *Fakt* results always in the same computed output (condition 1-c.). On the other hand, when we talk about the Boolean function XOR(X, Y), we mean by its universality something more: X,Y are preferably computationally independent, i.e., values of X,Y may be undetermined by other variables (ideally T-random) and this guarantees that our definition of XOR, at least in the input-part, is computationally the most abstract one possible. The variable whose values we can (and must) determine, per definition, is the one holding the value for the XOR function itself.

Let us more formally consider the question of computational independence of input-variables in an AU. The following principle is trivially valid[15]:

Principle 1: Discrete T-random variables are computationally indeterminate, but not necessarily stochastically independent of other variables.

Proof: We show that discrete, computationally determinate variables cannot be T-random. We also show that the concept of computational independence is different from that of stochastic independence[16]. First: Suppose X is a computationally determinate variable, i.e., there is a function f and a set of discrete variables $Y=\{Y_1,Y_2,\dots Y_n\}$ such that $X=f(Y_1,Y_2,Y_3,..)$ and $X<>Y_i$ for any i, meaning the variable set Y determines X. T-randomness is the lack of pattern or predictability. A T-random sequence of events, symbols or steps has no order and does not follow an intelligible pattern or combination [Randomness. (2016, August 30)]. Individual, T-random events represented by T-random variables are therefore by nature unpredictable and thus contradicting the essential property of outputs of what we classically call AU, namely: Predict- and/or Repeatability (function f). Moreover, functions assign per definition one output value with probability 1 to any input value. Accordingly, X cannot be T-random. Second: Suppose X is not determined through Y by any f. It still can be the case that there is a relation R such that X=R(Y). Below Fig. 2 shows such a typical case. If two

assumptions about their computational interdependency. This practically means that we are assuming that no subset Z of input-variables necessarily determines any single input-variable. Even better: In the ideal case we prefer to consider any input-variable to be computationally undetermined by any possible variable set (i.e., to be indeterminate), since this would underline the "universal" nature of our code. We treat X, Y therefore as *mere containers of values of the respective domains* which is the most abstract notion of computational independence possible and is sufficiently achieved, for the case of discrete domains, using the concept of T-randomness as seen in Principle 1.

[15] It relates to the intuitive notion of unpredictability.

[16] Obviously in classical AUs we are concerned in the first place with computational rather than stochastic properties of their components.

consecutive experiments are conducted with outcomes $r\in Y, s\in X$, forming a tuple (r,s) running over the sample space S={{1,*a*}{1,*b*}{1,*c*}{1,*d*}{2,*a*}{2,*b*}{2,*c*}{2,*d*}{3,*a*}{3,*b*}{3,*c*}{3,*d*}}, where R={{1,*a*}{1,*c*}{2,*b*}{3,*d*}}, then random variables *Y*='***r*** is the first element of a tuple in R' and *X*='***s*** is the second element of a tuple in R', are stochastically dependent. To see this it is sufficient to realize that for $1\in Y$ P(1)=1/2*1/3 and for $a\in X$ P(*a*)=1/4*1/3, but P(1,*a*)=1/12 which is not equal to P(1)P(*a*) as required by Definition 1. In fact, one condition which makes *X*,*Y* stochastically independent is R being equivalent to S, i.e., R not expressing any specific relational intention. On the other hand, if *f* exists and is bijective, then it is easily seen using a similar argument that *X* and *Y* are stochastically independent. Accordingly, indeterminate, discrete T-random variables can be stochastically dependent while determinate ones may be stochastically independent.
(Q.E.D.)

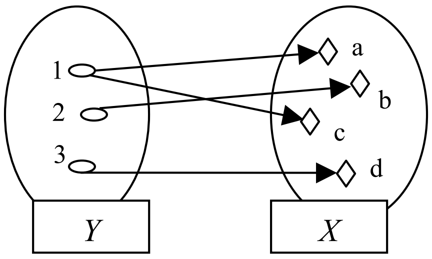

Figure 2

Linking randomness to computation is not new. Since *Kolmogorov complexity* was introduced in 1965, the amount of randomness in a value series is attributed to how small a program generating this series can be written. A series is deemed random (K-random or *computationally random*) if the smallest algorithm capable of specifying it to a computer has about the same number of bits of information as the series itself. This is an indication of the fact, that no short, algorithmically expressible rule can be found which facilitates the generation of such a series [Chaitin 1975]. Comparing this notion to what is presented here we realize that Principle 1 is more idealistic and calls a value series random (T-random), if its digits cannot be determined by any computable function *whether efficient or not*. If *f* in Definition 2 is required to be also efficient, this notion is similar to the one defined using *Kolmogorov complexity*. *We will see below that both notions of randomness can be shown not to hold when finite domain applications are considered.*

Let us formalize the above more precisely with the following definitions:

Definition 3: Let Vars={$X_0,X_1,X_2,\ldots$} be the set of all discrete variables, Domains={$D_1,D_2,\ldots$} the set of all possible finite or infinite domains of variables, Elements= {$e_1,e_2,e_3,\ldots$} the set of all elements of all such domains, function Val:Vars=> Elements mapping variables to their values in their respective domains, then:

> $X\in$Vars is *computationally indeterminate* or just *indeterminate* if there are no subsets S ⊆ Vars and no classical, i.e., non probabilistic, computable functions *f*:Elements^n => Elements such that for any possible values of variables: Val(*X*)=*f*(Val(Y_1), Val(Y_2), Val(Y_3),…,Val(Y_i).., Val(Y_n)), where $Y_1,Y_2,Y_3,..Y_i,..,Y_n\in$S and any Y_i<>*X*, otherwise it is said to be *computationally determined by S.*

Definition 4: An n-ary classical, computable function $f(X_0, X_1, X_2, \ldots. X_{n-1})$, $X_i \in$Vars, is called *universal* if:

1. any X_i maybe K-random, hence not efficiently determined or T-random, hence indeterminate (per Principle 1)
2. any X_i may assume any value in its domain
3. its computation is repeatable yielding always the same result

If a classical, computable function doesn't satisfy Condition 1, i.e., no T- or K-random variables are admitted as inputs, then it is called *Weakly Universal* (WU).

Theorem 1 is then straightforward[17]:

Theorem 1: Any finite set S=$\{X_1, X_2, X_3, \ldots, X_i, \ldots\} \subseteq$ Vars whose members span over finite domains can contain neither T-random nor K-random variables.

Proof: A canonical, classical, computable function f fulfilling Definition 3 can be shown to always exist. Let $D1 \times D2 \times \ldots \times D_n$ be the natural combinatory space **S** of the set S. As any D_i is a discrete, finite domain, we can always build **S** in finitely many steps enumerating in the same time each combination. **S** $=\{[e_{10}, e_{20}, \ldots e_{n0}][e_{11}, e_{21}, \ldots e_{n1}], \ldots. [e_{1a}, e_{2b}, \ldots e_{nz}]\}$, where e_{ij} is the j's element of D_i. For example if D1={A,B},D2={X,Y,Z},D3={I,J,K,L} then **S** is given in the following Table 1:

X_1	X_2	X_3	**Enum$_1$**	**Enum$_2$**
A	X	I	10	1
A	X	J	22	2
A	X	K	5	3
A	X	L	7	4
A	Y	I	13	5
A	Y	J	6	6
A	Y	K	17	7
A	Y	L	18	8
A	Z	I	19	9
A	Z	J	20	10
A	Z	K	11	11
A	Z	L	21	12
B	X	I	14	13
B	X	J	1	14
B	X	K	2	15
B	X	L	3	16
B	Y	I	8	17
B	Y	J	12	18
B	Y	K	15	19
B	Y	L	24	20
B	Z	I	9	21
B	Z	J	23	22
B	Z	K	16	23
B	Z	L	4	24

Table 1

[17] It is straightforward because it uses well-known "de-randomization by enumeration" or "truth table" techniques [Vadhan 2012] to show validity of the trivial statement: *One can only de-randomize objects which are neither fundamentally- nor computationally random.*

The last two columns of this table represent two different ways of enumeration showing two different bijective functions F_1, F_2: $\text{Enum}_1=F_1(X_1,X_2,X_3)$, $\text{Enum}_2=F_2(X_1,X_2,X_3)$. Both of them make all X_is, $i<=3$, determined by sets $\{\text{Enum}_1\}$ and $\{\text{Enum}_2\}$ respectively, since bijections can always be reversed. For example: $X_1=F_1^{-1}(\text{Enum}_1)\downarrow_1$ where $(\ldots)\downarrow_i$ stands for the projection of the resulting row on position *i*. What is the difference between F_1 and F_2? F_1 seems not to follow any clear pattern of choice in the selection of the next row while F_2 represents the canonical, linear way of enumeration of combinations. Note that combinations are already sorted in ascending order according to the precedence of elements in domains D_1, D_2 and D_3. Different element precedence choices in the domains may lead to different enumeration possibilities. Fig. 3 shows the distribution of combination values for both enumerations.

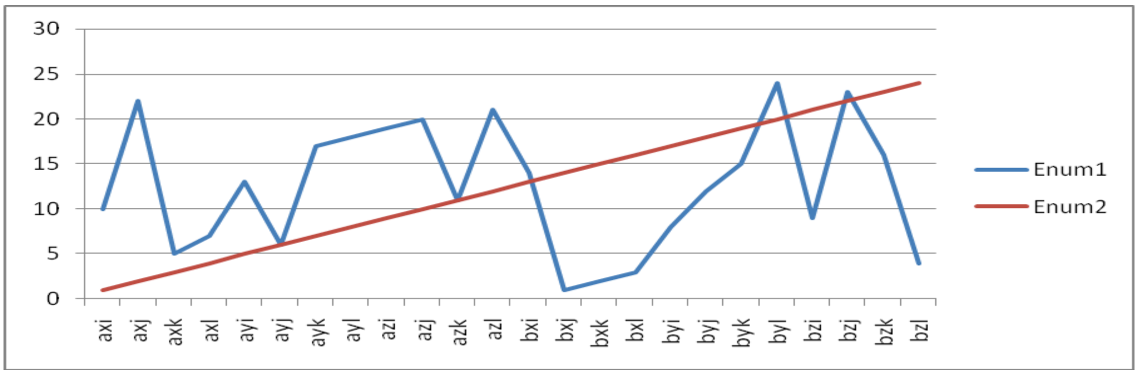

Figure 3

Projecting those two enumerations on variable X_2 yields the following Figures 4 and 5 which reveal what we call the *Pattern Property* of a variable[18].

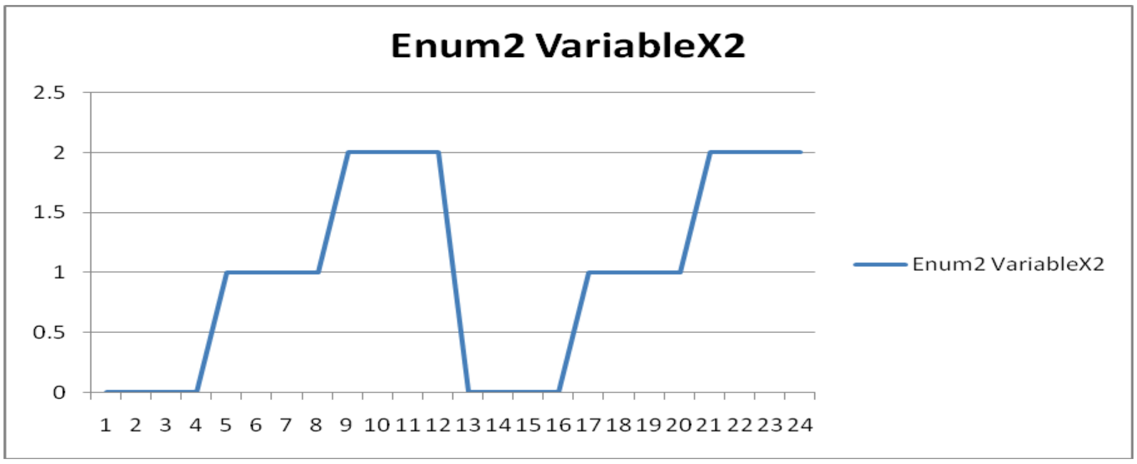

Figure 4

[18] Substituting for the values *X,Y,Z*: 0,1,2 respectively.

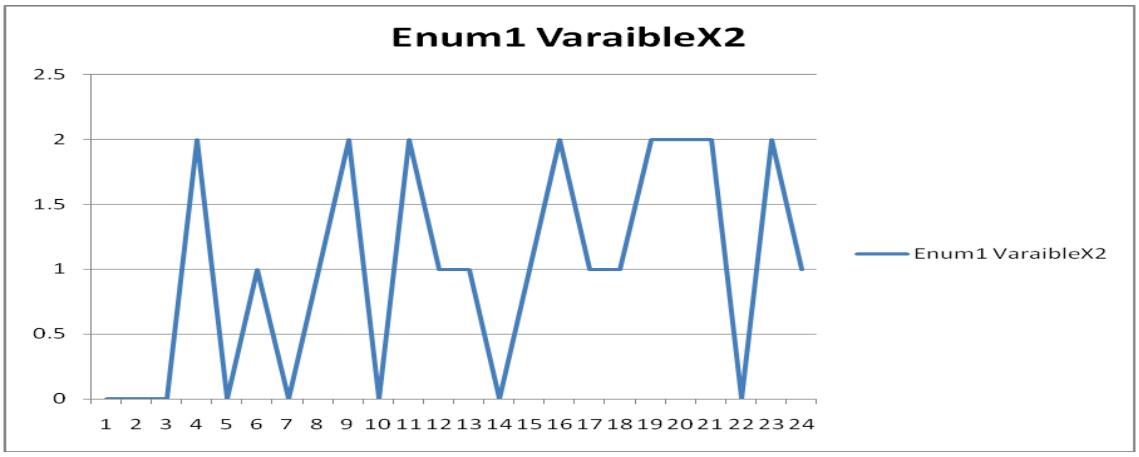

Figure 5

The above figures show a clear pattern for X_2 in F_2. We could similarly draw patterns for X_1, X_3 in F_2. On the other hand: F_1 doesn't lend itself to the same perception. We call enumerations which generate pattern properties for variables Harmonizing Enumerations (HE). Can this insight have any effect on the way we compute F_1 and F_2? The answer is: Yes. While we need to store the whole table to be able to determine the value of any variable from the row number in the case of F_1, we can avoid this by implementing F_2 in the following efficient way: Let E_{Index} be a variable holding the row number given by $Enum_2$, V_{Index} be holding indices of variables, $CardD_i$ stands for the cardinality of the discrete domain D_i. The n+2ary function F_2: N*x*N*x*…*x*N=>Elements can be defined as follows:

```
F_2(E_Index, V_Index, CardD_1, CardD_2,CardD_3,…. CardD_n) =
        if (V_Index==LastVarIndex)[19]
        {
        If (Mod(E_Index, TotalElements(V_Index))<>0)
                        F_2=ValueInD(V_Index,Mod(E_Index,TotalElements(V_Index))
                        Else
                        F_2=LastValueInD(V_Index)
        }
        Else
        {
                Quotient = Div(E_Index, PacketLength(V_Index)/ TotalElements(V_Index))
                Rest = Mod(E_Index, PacketLength(V_Index)/ TotalElements(V_Index))
                If (Quotient== 0)
                        F_2=FirstValueInD(V_Index)
                Else
                If (Rest == 0)
                        F_2=ValueInD(V_Index, CountValues(V_Index, Quotient))
                        Else
                        F_2=ValueInD(V_Index, CountValues(V_Index, Quotient)+1)
        }
```

Where $LastValueInD_i$ is a function N => Elements which gives back the last discrete value in a domain D_i for a variable V_i. For example $LastValueInD_2$=Z. Similarly for $FirstValueInD_i$. $ValueInD_i$ (<count>) is also an N=> Elements function which gives back the value number <count> in any D_i. $ValueInD_3(2)$ = J for example. TotalElements

[19] Variables are aligned from left to right in ascending order of the number of discrete values in their relative domains, the last variable domain to the right being the one with the most values.

for V_{Index} has the obvious meaning. PacketLength returns an integer indicating the number of rows which need to be filled with all values of a given variable according to its position. Here a recursive definition of PacketLength

```
PacketLength($V_{Index}$):-
        if (Index = LastVarIndex)
                PacketLength= TotalElements($V_{Index}$)
        Else
                PacketLength= TotalElements($V_{Index}$) * PacketLength($V_{Index}$+1)
```

E.g.: PacketLength(X_3)=4 while PacketLength(X_1) = 2*PacketLength(X_2) = 2*3*PacketLength(X_3)=2*3*4=24. Let us see if we can indeed calculate the value for any Variable V_i just from its index i and the E_{Index}: F_2(13,2)=ValueInD_2(1)=X which is right (compare with table above). F_2(10,2) will yield Z, the last value of D_2 which is also correct. F_2(1,3)=I, the first value of D_3[20]. It is easily seen, that F_2^{-1} can be efficiently calculated as well. Note that while F_1 is sufficient to show contradiction with T-randomness of Principle 1, F_2 and/or its components can be used to show the same for K-randomness based on *Kolmogorov complexity*, since any value series of a discrete variable X_i may be generated (after application of HE) using, for example, the following algorithm whose size is much smaller than any solution requiring storing the whole truth table:

```
Generate($V_{Index}$, $CardD_1$, $CardD_2$,$CardD_3$,.... $CardD_n$):-
        TotalNumberOfCombinations= $CardD_1$ x $CardD_2$ x $CardD_3$,.... $CardD_n$
        i= 1, j=1
        CurrentValue= FirstValueInDomain($D_{Index}$)
        RepetitionCount= PacketLength($V_{Index}$)/$CardD_{Index}$
        While i<= TotalNumberOfCombinations
                If j<=RepetitionCount
                        Result=Result+Curren-tValue
                        Else
                        CurrentValue= NextValueInDomain($D_{Index}$)
                        j=1
                        Result=Result+CurrentValue
                j=j+1, i=i+1
        Return Result
```

(Q.E.D.)

The main assertion of Theorem 1 is therefore: *Discrete variables in finite domains can neither be truly, non-predictively nor computationally random because their natural combinatory space can always be enumerated using an HE defining an order between possible combination values. This order, when known, efficiently generates a value pattern for each variable which becomes then a characteristic second nature.*

The following story is used as an intuitive, informal illustration:

> *"Leila and Qu'ais were lovers whose fate was sealed when Qu'ais had to travel to a far place, away from his beloved princess. In this far place time didn't play any role. They agreed before he left to send each other encrypted poetry expressing their feelings. Leila was not free to send her beloved one ordinary messages which might be intercepted by her family. So they had to think of something else. As poetic words of love are finite in number, they ordered them*

[20] The reader is encouraged to go through all combinations in the table to verify that F_2 is indeed calculating the indicated values of the variables V_i.

> *in categories, then made a long list of phrases and enumerated those phrases. Leila was always under observation by her family, so they needed to agree also on an adequate way to express them. They chose flowers whose colors could easily be interpreted as words. Leila started to acquire all and every type of flowers she could get and place them every day in a specific place in front of her window. Each flower representing a poetic word of one category, the whole bouquet would then form a phrase. She would choose flowers having harmonic sizes with pure, explicit colors as if she herself was making rhymes. For an external observer she was placing colored flowers conveying random feelings. For her and her beloved Qu'ais it was much more!"*

Take now the position of M, a smart observer and member of her family, who was suspecting that his/her relative didn't get over her love story. When could he/she discard the hypothesis that *Leila* was only sending random messages*?* If he/she could prove that whatever *Leila* was sending was from an *enumerated list of combinations*, i.e., single flower colors in specific positions were repeated in the same order after a certain cycle length for example, then he/she would know that it probably reveals some sort of encoding. Of course this would only work if *Leila* and *Qu'ais* kept sending messages in the enumeration order. On the other hand: What if *Leila* and *Qu'ais* wanted to avoid sending phrases revealing content and agreed to communicate only row numbers of desired phrases, which function would they have preferred: F_1 or F_2? As *Qu'ais* didn't care in his world about time, any choice would have been valid for him. This couldn't have been appropriate for *Leila*, because of the needed combinatory space in case of F_1. They probably would have used F_2, the one exploiting pattern properties of variables.

As anticipated it follows from Theorem 1:

Corollary 1: Algorithms whose input-variables span over discrete, finite domains are only weakly universal (WUA).

Proof: Follows immediately from Theorem 1 and Definition 4.
(Q.E.D.)

Although conceived for classical algorithms, Theorem 1 and Corollary 1 reproduce questions and facts related to probabilistic computation:

1. If we know that we can always de-randomize, i.e., predict and determine mathematically input-variables of probabilistic algorithms (c.f. fn. 11 for their informal definitions) when they are used in finite domains, what exactly do we mean by their *randomness*? The commonly accepted answer to this question is: We mean what is called *pseudo-randomness*, i.e., processes which only appear to an observer to be random, but which are not[21]. Theorem 1 shows that we can mean neither T- nor K-randomness. *Recall that what is shown here is not related to the controversy about whether T-randomness exists or not[22], but to the fact, that if/when it exists and is allowed to be included in our definition of* algorithm*s in finite domain applications, it leads to an inappropriate notion of computability allowing us to calculate/predict/determine an indeterminate quantity.*
2. In the same line: Theorem 1 suggests that probabilistic definitions of BPP and related complexity classes *only reflect statistical randomness*, *i.e., neither T- nor K-randomness.* In that case, what shall be the added value of this randomness over non-determinism or determinism? An intuitive answer to this question is: Maybe none. In fact, as shall be seen again below, one known way to prove that P=BPP is by de-randomizing and showing that the number of random bits *n* used by any BPP algorithm can be reduced to become O(log (*n*)) [c.f. Vadhan 2012][23].
3. *Randomness vs. Hardness* efforts use the presumable intractability of some functions to generate good pseudo random bits. The idea is that any one-way permutation[24] can be used to construct pseudo random bit generators producing strings which can fool every polynomial time computation. This assumes that those functions are (for small circuits) not only intractable, but also not approximately attainable. Given such a "hard" function it is easy to generate bits which look like random from the perspective of any small circuit [Nisan 1994]. Those bits are then neither T- nor K-random.

[21] c.f. Pseudorandomness. (2015, December 20). In Wikipedia, The Free Encyclopedia. Retrieved 11:08, July 4, 2016, from https://en.wikipedia.org/w/index.php?title=Pseudorandomness&oldid=696096676]

[22] In physics: If outcomes can be determined (by hidden variables or whatever), then any experiment will have a result. More importantly, any experiment will have a result whether or not you choose to do that experiment, because the result is written into the hidden variables before the experiment is even done. Like the dice, if you know all the variables in advance, then you don't need to do the experiment (roll the dice, turn on the accelerator, etc.). The idea that every experiment has an outcome, regardless of whether or not you choose to do that experiment is called "the reality assumption", and it should make a lot of sense. If you flip a coin, but don't look at it, then it'll land either heads or tails (this is an unobserved result) and it doesn't make any difference if you look at it or not. In this case the hidden variable is "heads" or "tails", and it's only hidden because you haven't looked at it. It took a while, but hidden variable theory was eventually disproved by John Bell, who showed that there are lots of experiments that cannot have unmeasured results. Thus the results cannot be determined ahead of time, so there are no hidden variables, and the results are *truly* random. *That is, if it is physically and mathematically impossible to predict the results, then the results are truly, fundamentally random.* [Physicist 2009]

[23] This hasn't been successfully done until this day although it is widely believed that P=BPP. The technique of reducing random seed bits seems to only work for some problems (like the MAX-CUT), where pair wise independence of variables is sufficient to produce fast approximation algorithms [Rubinfeld 2012].

[24] i.e., permutations which are presumed to be calculated much easier than their inverse

4. Although it is known that if P=NP then P=BPP [Goldreich 2011], little effort seems to have been spent in understanding the effect of intelligent enumerations/de-randomizations (such as HE) on the P vs. NP question itself. More specifically the following may be asked: *Can an intelligent, canonical enumeration (such as HE) help solve the P vs. NP question by revealing generic structure in an NP complete problem showing indirectly that P=BPP?*

In the following, we will answer this question informally while a formal implementation of our answer can be found in [Abdelwahab 2016]. Noting that Theorem 1 as well as Corollary 1 have validity for all attempted solutions of NP-complete problems, we find that:

1. An HE's side effect is the creation of patterns for all input-variables of any NP complete problem. *Thus in any such problem we can safely assume that variables represent truth patterns like the ones depicted in Fig. 4.*
2. Assuming truth patterns in input-variables of NP complete problems has the following important consequence: *It defines a canonical order between variables* (as shall be seen in the examples below). This order is determined by what [Abdelwahab 2016] calls Pattern Length. Furthermore: The order enables sorting conditions on Clause-Sets, the most relevant one being the linear order (l.o.) condition.
3. On the other hand: Finding an optimal order between variables facilitating the construction of minimal BDDs is (in itself) a known NP complete problem [Tani 1993].
4. It is then sufficient to show that *any optimal order of variables sought for the construction of minimal BDDs necessitates l.o. conditions on all Clause-Sets* of a resolution tree (Lemma 9 in [Abdelwahab 2016]) and that such resolution trees have a polynomial number of unique nodes in the worst case (Lemma 13). This is the core of proofs and theorems which use w.l.o.g. FBDDs[25].

[25] More precisely, it is shown that FGSPRA$^+$, the pattern oriented 3SAT core-Solver-algorithm constructing FBDDs: 1- always generates a polynomial number of nodes in a polynomial amount of steps (in M, the number of clauses). All of them possessing linearly ordered Clause-Sets 2- always approximates MinFBDD, the problem of constructing the minimal FBDD for a Boolean function, to a constant factor of 2.

III EXPLORING PATTERN PROPERTIES

The majority of algorithmic attempts to solve NP complete problems were actually, as per Corollary 1, weakly universal. Moreover: Although only weakly universal, they were viewing variables as mere containers. *In this section, an attempt to solve 3SAT using pattern properties of variables only is shown to fail.* Its failure shall be the basis for insights leading to the inefficiency principle formulated as a conjecture in the next section. But first, Fig. 6 shows a complete, simple example of a 3CNF Clause-Set[26] and its dual nature. The function depicted shows XOR(X_0,X_1,X_2) enumerated using Truth Table 2. The reader may verify the correctness of the patterns for all clauses given below[27]:

X_0	X_1	X_2	X_1 OR X_2	X_1! OR X_2	X_1! OR X_2!
0	0	0	0	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	1	0

Table 2

[26] X_i! stands for not(x_i)

[27] Clause Patterns are simply bit-or results of single variable patterns. Note that not all patterns depicted in Fig. 6 are shown in the table.

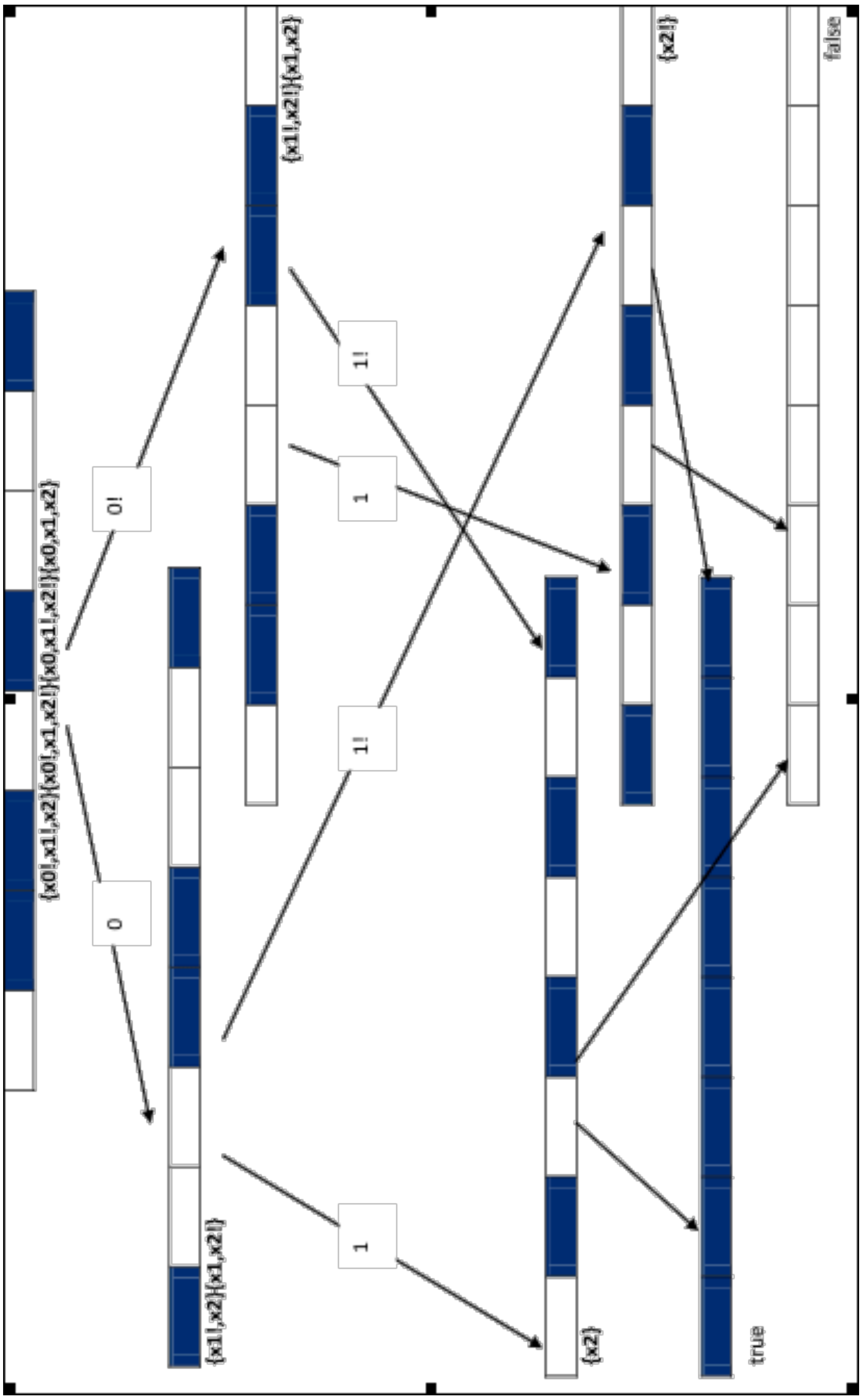

Figure 6: dual natures of 3CNF Clause-Sets

Important remarks related to Fig. 6:

1. Variables X_i differ in their pattern lengths by power-of-two factors. They can be thought to represent different harmonic occurrences of one base pattern (e.g., PatternLength(X_1)=2*PatternLength(X_2) etc.).
2. One Clause-Set can be assigned only one exact pattern. One pattern may be shared by many Clause-Sets which are logically equivalent.
3. Clause-Sets/Patterns may be organized in trees whose nodes are countable when resolution procedures are used to gradually instantiate literals.
4. All variables in all nodes are ordered as per Table 2, i.e., from left to right in decreasing pattern length. This is an important premise of the l.o. condition. General 3CNF-formulas are usually not l.o.. Such property can always be attributed to *n*-ary parity functions as shall be discussed in below explanations of Section VI.

Now suppose S={ {x_0,x_1!,x_3} {x_0!,x_2} {x_0} {x_2!,x_3!} {x_3} }. We will attempt to solve S using continuous functions only. If X_i is assumed to possess a truth pattern, then it can be modeled by a square-wave[28]. Fig. 7 shows patterns reflecting all variables and clauses. S is unsatisfiable.

x_0	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□
x_1	■	■	■	■	□	□	□	□	■	■	■	■	□	□	□	□
x_2	■	■	□	□	■	■	□	□	■	■	□	□	■	■	□	□
x_3	■	□	■	□	■	□	■	□	■	□	■	□	■	□	■	□
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
value =3	□	□	□	□	■	□	■	□	□	□	□	□	□	□	□	□
value =2	■	□	■	□	■	■	■	■	□	□	□	□	■	□	■	□
C_1=x_0 OR x_1! OR x_3	■	■	■	■	■	■	■	■	■	□	■	□	■	■	■	■
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
value =3	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
value =2	□	□	□	□	□	□	□	□	■	■	□	□	■	■	□	□
C_2=x_0! OR x_2	■	■	□	□	■	■	□	□	■	■	■	■	■	■	■	■
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
value =3	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
value =2	□	□	□	■	□	□	□	■	□	□	□	■	□	□	□	■
C_3=x_2! OR x_3!	□	■	■	■	□	■	■	■	□	■	■	■	□	■	■	■
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
value =3	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
value =2	□	□	□	□	□	■	□	□	□	□	□	□	□	■	■	■
C_1 & C_2 & C_3	□	■	□	□	□	■	□	□	□	□	■	□	□	■	■	■

Figure 7

[28] Using Fourier expansion with cycle frequency *f* over time *t*, we can represent an ideal square-wave with an amplitude of 1 as an infinite series of the form:

$$x_{\text{square}}(t) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi(2k-1)ft)}{(2k-1)}$$
$$= \frac{4}{\pi}\left(\sin(2\pi ft) + \frac{1}{3}\sin(6\pi ft) + \frac{1}{5}\sin(10\pi ft) + \cdots\right)$$

c.f. Square wave. (2016, July 31). In Wikipedia, The Free Encyclopedia. Retrieved 09:29, August 1, 2016, from https://en.wikipedia.org/w/index.php?title=Square_wave&oldid=732329468

There may be many ways to express the pattern nature of variables and try using this nature alone to solve the satisfiability problem. One could for example use simple trigonometric functions. Here, we have chosen w.l.o.g. to express any variable X_i using the formula: SquareWave[c*x]*0.5+0.5 and X_i! using: SquareWave[c*x]*-0.5+0.5, where factors 0.5 are necessary to keep the square-wave positive, c is the frequency of the wave. Then, solving the above S is equivalent to solving the inequality:

```
((SquareWave[x]*0.5+0.5)+(SquareWave[2*x]*-0.5+0.5)+(SquareWave[8*x]*0.5+0.5))
* ((SquareWave[x]*-0.5+0.5) + (SquareWave[4*x]*0.5+0.5))
* (SquareWave[x]*0.5+0.5)
* ((SquareWave[4x]*-0.5+0.5) + (SquareWave[8*x]*-0.5+0.5))
* (SquareWave[8x]*0.5+0.5) > 0
```

In [Ahmadi 2012] it is shown that the related problem of deciding local asymptotic stability of trigonometric vector fields is strongly NP hard. This result means that even in the continuous spectrum we seem not to be able to efficiently recognize positive values of a truth pattern without checking exponentially many points. In reality, there might be much less to check if sub-patterns are always repeated like in the case of: C_3={x_2! OR x_3!} in Fig. 7 for example, where sub-pattern: "0112" is constantly repeating. How do we count unique sub-patterns in a pattern? This doesn't seem to work as well, using the pattern view only, without recurring through the whole pattern length.

IV THE INEFFICIENCY PRINCIPLE

Definition 5: Let S be a Clause-Set. Its 3CNF form is called: Container Expression of S (CE_S), while its pattern form is called Pattern Expression of S (PE_S). PE_S is a string representing all possible assignments which satisfy S and may be formed using the following canonical brute force WUA (cWUA):

```
Inputs: $CE_S$
Outputs: $PE_S$
1- Create the truth table for all variables in $CE_S$,
2- Form $PE_S$ from left to right
    a. For all combinations c in the truth table:
        i. Substitute values of c for literals in S
        ii. If S is satisfied write "1" in the $PE_S$ string else write "0"
3- Return $PE_S$
```

For any S: PE_S exists iff CE_S exists (per definition). They are called *Dual Images* of S[29]. $\forall C_i \in$S: The Pattern Expression of C_i shall be called PE_{C_i} and may be obtained using the above algorithm putting S=C_i.[30] A sub-pattern $sp \subseteq PE_S$ is said to satisfy a sub-formula $sf \subseteq CE_S$ (written: $sp \vdash sf$) if sp contains at least one assignment represented by a "1" which satisfies sf.

[29] Classically, duality of logical formulas is based upon duality of logical operators used to construct those formulas. In such dualities the dual of an object is of the same kind as the object itself. Another example of this type is the dual of a vector space, which is again a vector space. Some duality statements, like the one presented here, are not of this kind. Instead, they reveal a close relation between objects of a priori fundamentally different nature (i.e., here the dual of a logical formula (CE_S) is a semantic pattern (PE_S)).

[30] Note that any PE_{C_i} is *not* a sub-pattern of PE_S

Definition 6: A Weakly Universal Algorithm (WUA) processing S by using *only one* of the dual expressions CE_S or PE_S or any information deduced from it in its inputs or definition is called a *Single Viewed Weakly Universal Algorithm (WUA_{SV})*. A WUA processing S by using *both* expressions or any information deduced from them in its inputs or definition is called a *Pattern Oriented Algorithm (POA)*[31]. The cWUA of Definition 5 is therefore a WUA_{SV}.

Definition 7: Let CE_S and PE_S be dual images of Clause-Set S, then*: $3SAT_S$* is *any of the following* problems expressed using different viewpoints:

1- Deciding whether consequent instantiations of literals in clauses $C_i \in S$ forming a resolution tree for CE_S result in TRUE. (*Container View*)
2- Deciding whether the sub-pattern representing the intersection of all PE_{C_i}s where $C_i \in S$ contains "1". (*Pattern View*)
3- Deciding whether there is any sub-pattern $sp \subseteq PE_S$: $sp \vdash \bigwedge C_i \in S$ ($\equiv CE_S$). (*Combined View*)

Using the *Combined View* we can define *Dual-$3SAT_S$* to be the problem of: Deciding whether there is any sub-formula $sf \subseteq CE_S$: $\forall PE_{C_i}$ where $C_i \in S$: $\cap PE_{C_i}$ ($\equiv PE_S$) $\vdash$ sf[32]. $Conv_S$ is the problem of converting CE_S to PE_S and vice versa.

We are ready now to propose (without proof) the following conjecture:

Conjecture 2 (Inefficiency Principle)[33]**:** *For any 3CNF Clause-Set S: Efficient WUA_{SV}s solving $3SAT_S$ and Dual-$3SAT_S$*[34] *exist iff efficient WUA_{SV}s solving $Conv_s$ exist*[35]. *Moreover: No efficient WUA_{SV}s solving Conv can exist, therefore no efficient WUA_{SV}s solving $3SAT_S$ and Dual-$3SAT_S$ can exist as well.*

In this paper, we are suggesting the inefficiency principle to replace the commonly accepted Conjecture 1 for the following reasons:

---

[31] Formal definitions and properties of POAs are thoroughly presented in [Abdelwahab 2016].

[32] In other words, Dual-$3SAT_S$ is the question: Are there any sub-formulas in CE_S satisfied by common sub patterns of clauses of S*?* This is not to be confused with MAX-SAT which is the problem of determining the maximum number of clauses which can be satisfied (nor with MIN-SAT having a similar definition).

[33] We are borrowing this nomenclature from physics to express the fact that: *Both expressions of a Clause-Set cannot be efficiently measured/constructed/observed/attained using one perspective only.* Analogous to the Wave-Particle duality case: The uncertainty principle states that specific pairs of quantities cannot be simultaneously measured to arbitrary accuracy.

[34] Because of the duality it is clear that a solution for $3SAT_S$ exists iff a solution for Dual-$3SAT_S$ exists. To see this suppose that $3SAT_S$ is solved. This means that there is a sub-pattern $sp \subseteq PE_S$: $sp \vdash CE_S$. In that case both sp and CE_S help satisfying $PE_S \vdash sf$ where $sf = CE_S$ and thus solving Dual-$3SAT_S$ as well. Suppose now that there is $sf \subseteq CE_S$, $PE_S \vdash sf$. This means that there is a sub-pattern $sp \subseteq PE_S$: $sp \vdash sf$. Therefore, there is an assignment represented by a "1" in sp which satisfies sf. This assignment must also satisfy CE_S, because $sp \subseteq PE_S$ Therefore it must be the case that $sp \vdash CE_S$ which means that $3SAT_S$ is solved.

[35] Notions used here could have been expressed in terms of Turing-reductions. This is omitted, to avoid complication and because proving the inefficiency principle is not attempted.

1. Linking the existence/non-existence of a positive, single perspective solution of $3SAT_S$ and Dual-$3SAT_S$ to the existence/non-existence of a positive, single perspective solution of the seemingly deeper $Conv_S$ may be thought of as an intrinsic "natural reason" for inefficiency of single views as opposed to the "no apparent reason" understood from Conjecture 1.
2. $Conv_S$ seems to be "deeper", but as hard (for WUA_{SV}s) as both $3SAT_S$ and Dual-$3SAT_S$. It is a problem whose solution seems to entail the solution of those latter two and vice versa. Note that if we started only with CE_S and got PE_S somehow efficiently formed using a WUA_{SV} for $Conv_S$, then $3SAT_S$ (in form 3- of Definition 7) is solved because we can easily identify peaks in PE_S during the process of its formation. On the other hand, if we started only with PEs and managed to create CE_S efficiently using such a WUA_{SV}, then sub-formulas of CE_S can be checked for satisfiability via PEs during this process as well. The other way round is less obvious, but seems nevertheless possible, i.e., an answer to the question: If we had efficient WUA_{SV}s for both $3SAT_S$ and Dual-$3SAT_S$, how can we create (using one perspective only*)* PE_S or CE_S*?*[36]
3. Conjecture 1 generalizes its assumption to AUs along with non uniform algorithms while the inefficiency principle generalizes only to WUA_{SV}s. As per Corollary 1, this is more reasonable since known algorithms failing to solve NP-complete problems were/are actually WUA_{SV}s[37]. Therefore, the inefficiency principle is detached from the more general P vs. NP question and is thus not contradicting the P=NP result of [Abdelwahab 2016] which only shows that there are POAs efficiently solving $3SAT_S$.
4. And most important in the epistemological context: As HE producing pattern properties of logical variables represent side effects of realizations in the finite, physical world of concepts developed for mathematics, it seems only plausible that dualities inherent in nature are reflected back causing problems. *While Conjecture 1 just states the fact that no efficient* algorithm*s can be found for $3SAT_S$, Conjecture 2 attributes the phenomenon to the inherent difficulty of switching (without additional information) between dual pattern and container views, a familiar inability which is known to exist in some form in other contexts as well.*

[36] One possible answer to this question might be: If $3SAT_S$ and Dual-$3SAT_S$ could be efficiently solved using WUA_{SV}s, it probably would be the case that the number of small sub-formulas/sub-patterns in CE_S/PE_S, which are relevant for deciding the question in either perspective, is always polynomial for a given "magic" transformation of S. We could then efficiently transform each such small sub-formula to the corresponding sub-pattern or vice versa, if this transformation could be shown to be better than brute force.

[37] Dual views of a logical variable/formula were first introduced in [Abdelwahab 2016]. Current state-of-the-art research related to the P vs. NP problem still assumes the single container view.

V THOUSAND AND ONE NIGHT

And the love story continues this way:

"Leila's family decided to lock her up in a palace behind one of thousand doors as a measure to prevent her from communicating with her beloved one. She could only send Qu'ais one single message, before she was taken away, telling him that the door behind which she was imprisoned shall be marked with her special sign. When Qu'ais arrived he realized that all the doors were exactly similar in color and size. No special marks could be found on any of them. However, on each doorstep there was a different flower-bouquet. All bouquets contained flowers of all colors and sizes. Obviously, trying all doors one by one was not a practical solution, since this would definitely draw the guard's attention to him. What could Qu'ais do to recognize the correct bouquet and thus the right door? While searching for a clue, he thought for a while of all love conversations he had with his princess who always used rhymed poetry to convey her feelings. It suddenly occurred to him that this must be her sign: One of the bouquets must be containing harmonically organized flowers. Indeed there it was: A neatly organized one with flowers whose sizes and colors were so beautifully synchronized that it definitely couldn't have been created by accident or mistake."

If a bouquet was a Clause-Set and individual flowers were literals, then the harmonic order of flowers solving the problem of *Qu'ais* would be the l.o. condition. This condition is enabled by the HE materialized in the truth table as mentioned in previous sections. The problem of *Qu'ais* was solved by recognizing signs on similarly looking doors, then differentiating between harmonic and non-harmonic order conditions of those signs. This is very similar to what is done in the *I'rāb* procedure[38] used to clarify the meaning of a sentence in the classical Arabic language. Inspired by that[39] both, a *conjecture* and an *objective* were formulated and proven in [Abdelwahab 2016]. Fig. 8 shows an overview of all core ideas including those presented in the current paper. The remarks below are intended to clarify and help critical readers find flaws in [Abdelwahab 2016]. Arrows point at necessary conditions shown to exist through definitions/theorems/lemmas whose numbers are noted in smaller fonts on top of each arrow:

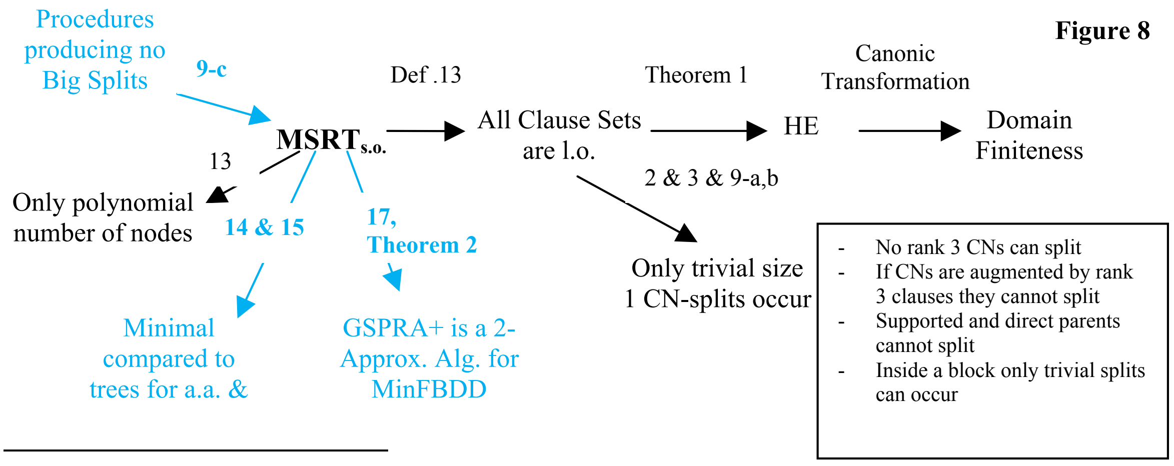
Procedures producing no Big Splits
9-c
MSRT s.o.
Def .13
All Clause Sets are l.o.
Theorem 1
HE
Canonic Transformation
Domain Finiteness
13
Only polynomial number of nodes
14 & 15
Minimal compared to trees for a.a. &
17, Theorem 2
GSPRA+ is a 2-Approx. Alg. for MinFBDD
2 & 3 & 9-a,b
Only trivial size 1 CN-splits occur
- No rank 3 CNs can split
- If CNs are augmented by rank 3 clauses they cannot split
- Supported and direct parents cannot split
- Inside a block only trivial splits can occur

Figure 8

[38] c.f. Introduction of [Abdelwahab 2016]

[39] Ancient linguistic theories concerned with the relation between syntax and semantics of classical Arabic inspired other modern, seemingly not related disciplines as well. In [Al-Harithy 1987] an application of *Al-Jurjanis* theory of meaning on ancient and modern architectural artifacts is used to reveal a new understanding of architecture as a mode of communication.

The above figure shows the following:

1) Special FBDDs (called also $\mathrm{MSRT}_{\mathrm{s.o.}}$s) generated by GSPRA^+ and FGPRA^+ (the POAs presented in [Abdelwahab 2016]) are the center of all investigations which go into two directions:
   a) To show that $\mathrm{MSRT}_{\mathrm{s.o.}}$s possess always a polynomial number of unique nodes (in the length of the 3CNF formula, more particularly: in M, the number of clauses). This is the formulated *objective*.
   b) To show that they are near to optimal. More precisely: GSPRA^+ and FGPRA^+ are 2-Approximation algorithms for the MinFBDD NP complete problem. This is the formulated *conjecture*. Arrows and comments related to it are marked blue (grey) in the above figure.
   c) While a) solves the P vs. NP problem, b) is a practical result leading to *Theorem 4* which states that all Boolean functions expressed in a compact form, i.e., M is a polynomial function of N, the number of variables, can have small FBDDs.
   d) For both directions: Points of reference and tools, making comparisons between resolution procedures possible, are:
      i) The idea of splits of resolution trees which are copies of sub-trees performed during sequential resolution steps causing a blow up of the number of newly created nodes. (Section I and II-C).
      ii) The concept of linking 3CNF resolution procedures to FBDD construction while using always the least literal as the current choice of instantiation in any step (least literal rule), i.e., not "trying" different choices of literals (Section II-A Definition 2). This rule goes hand-in-hand with the concept of a canonical ordering of variables shown to be the most generic notion possible (Property 9).
      iii) 10 properties which are shown to be valid for POAs independent of their different flavors (Section II-B).
      iv) The idea of renaming Clause-Sets to re-align literals in case the l.o. condition is not met in any sub-tree. Consequence of this idea is the possibility to use one single canonical form for all Clause-Sets and make therefore procedures comparing them (using the concept of syntactical equivalence of nodes instead of the more elaborate algorithmic equivalence (Section II-D)) sufficiently efficient.
2) To show that *objective* is attained, [Abdelwahab 2016] needed to show that:
   a) L.o. conditions imposed on Clause-Sets, which are the marking property of $\mathrm{MSRT}_{\mathrm{s.o.}}$s, don't permit any non-trivial splits of Common Nodes (CNs), a trivial split being a copy of a size 1 CN. This required Lemma 2 and 3 for the GSPRA version (the one without renaming) and then Lemma 9[40] for GSPRA^+. The basic ideas behind those proofs can be summarized in the following points:
      i) If a CN of rank 3 (i.e., having clauses containing three literals) is formed in step *k*, it cannot be spilt in steps >*k*. Same for CNs augmented to sizes >1 by a clause of rank 3.
      ii) Supporting and direct parents of CNs cannot be used for splitting (see Definition 5, Section II).

[40] Lemma 9 is essentially showing that properties of s.o. Sets/SRTs proven in Lemma 3 remain untouched by renaming operations.

   iii) Splits can only occur within blocks. While all trials to augment sizes of CNs to become >1 before splitting fail in ordinary cases, Symmetric Blocks (SBs) allow for so called tCNs (trivial CNs) to have sizes >1 before they are even formed. Fortunately splitting such tCNs is not harmful since they can either be avoided altogether using additional sorting conditions (called: l.o.s.) or ignored altogether, because l.o.s. and l.o. conditions are equivalent with respect to the total number of generated nodes (Lemma 2-f).

  b) Using the fact that there are no big splits in $\text{MSRT}_{\text{s.o.}}$s, GSPRA^{+} is shown in Lemma 13 to produce a worst case of $O(M^4)$ unique nodes
  c) The practical, parallel algorithm FGPRA^{+} is then shown to be:

   i) Equivalent to GSPRA^{+} (Lemma 18)
   ii) Possessing a worst case complexity of $O(M^9)$ (Lemma 19)
   iii) Efficiently solving the 3SAT problem (Theorem 1)

3) To show that *conjecture* is valid:
  a) Procedures producing only trivial splits are shown to be actually producing $\text{MSRT}_{\text{s.o.}}$s (Lemma 9-c).
  b) Procedures using a.a. or l.o.u. 3CNF Clause-Sets and producing big splits are shown to generate resolution trees whose number of nodes is larger than or equal to $\text{MSRT}_{\text{s.o.}}$s produced by GSPRA^{+} (Lemma 14 and 15)
  c) GSPRA^{+} is near to optimal, FGPRA^{+} is a 2-Approximation algorithm for MinFBDD (Lemma 17, Theorem 2 respectively)

4) All this is possible because of HE and the underlying finite domains assumption (Theorem 1 in this paper).

A critical reader who is tempted to find flaws in [Abdelwahab 2016] may therefore try the following ways[41]:

1) *Without investing time reading the lengthy proofs of the paper:*
Show the existence of an exponential lower bound on the construction of FBDDs for compactly expressed 3CNF Boolean formulas. This lower bound should be able to survive the 3SAT Solver test as per [Abdelwahab 2016], i.e., in case it is proven for *k*SAT, it should stand even after the conversion from *k*SAT to 3SAT and passing it to FGPRA^{+}[42].

2) *After going through the proofs,* show for GSPRA^{+}, the main SPR-like algorithm described there one of the following:
  a) A technical mistake in one or more proofs of lemmas and theorems describing behavior or properties.
  b) Splits are not the only cause of exponential behavior, i.e., number of nodes can experience a blow-up between sequential resolution steps even without N- or CN-splits.

[41] Such a reader is not only encouraged, but also helped clarify arguments against what is proposed here or in [Abdelwahab 2016] if required.

[42] This is *not* the case for example with the "blocking-sets-in-finite-projective-planes" problem which is discussed in Theorem 2 in [Abdelwahab 2016] and which shall be dealt with explicitly in the next section in the context of this paper as well.

c) Big Splits may occur even when l.o. conditions are enforced on all Clause-Sets. This would defeat Theorem 1 and the *objective***.** It is sufficient to show that a non-trivial CN augmented in step *k* to a size >1 can be split in any step>*k*. For trivial CNs: Show that they cannot be avoided and/or that l.o. and l.o.s. conditions are not equivalent in terms of the final number of nodes they generate (defeating Lemma 2-f).
d) Show that renaming algorithms are either incorrect or not terminating for any given special case of 3CNF formulas.
e) Show that using a.a. or l.o.u. Clause-Sets can, for any given SPR-like or unlike procedure generating Big Splits, produce less nodes (defeating Lemma 14, 15, and 17). For the case of procedures not producing Big Splits: Show that they don't produce $\text{MSRT}_{\text{s.o.}}$s (defeating Lemma 9-c and underlying 3-d). This amounts to showing that GSPRA^{+} and FGPRA^{+} are not generating near to optimal FBDDs (defeating Theorem 2 and the *conjecture*)

VI EXPONENTIAL LOWER BOUNDS AND OTHER PHENOMENA

In this section known exponential lower bounds are shown *not to contradict* the main results of [Abdelwahab 2016], which can be summarized in the following points:

1. POAs using information from both sides of the dual nature(s) of logical variables and Clause-Sets (especially information related to pattern lengths of variables and/or their alignments) can *always enforce* safety conditions and construct FBDDs which are small with respect to the input size. In technical terms: POAs produce FBDDs which always possess (in the worst case) a polynomial number of nodes with respect to M, the number of clauses in a 3CNF representation of a Boolean function.
2. This doesn't contradict the fact that for some Boolean functions, there are no known ways to reach a compact 3CNF representation. For such cases FBDDs produced by POAs may[43] be large with respect to N (but not to M).
3. Hence, there are no compactly described Boolean functions which fail to produce small FBDDs when resolved using POAs.

Known lower bounds mentioned in Section II:

- *Diagonalization: SAT cannot have algorithms that use both, a small amount of time and memory.* This is not contradicted by [Abdelwahab 2016] since both, time and space used by POAs are in $O(M^9)$ and $O(M^4)$ respectively.
- *Proof Complexity*: POAs don't use resolution-based or -like procedures to solve 3CNF formulas in the first place.
- *Circuit Complexity and natural proofs: It has been shown that small circuits cannot solve the parity function if the circuits have a fixed number of layers of gates. It was also shown that the NP-complete problem of finding a large clique does not have small circuits if one only allows AND and OR gates (no NOT gates).* Firstly, BDDs in general and FBDDs in particular are not built using logical AND, OR or NOT gates. Accordingly, there is no contradiction to the here-discussed results. Secondly, circuit lower bounds, especially related to the parity function, seem to underline the case described in 2. above of a Boolean function which cannot be expressed in a compact

[43] Example of such FBDDs are those constructed for Integer Multiplication.

way. In fact, using those lower bounds as well as the results here, we can explain the following behavior of parity functions: *They cannot be expressed in a compact form, but always possess small FBDDs. We do this because of known lower bounds:*

*k*CNF representations of a parity function of arity *k* have:

- always: M=*f*(N=*k*), where *f* is exponential *and*
- each clause in the *k*CNF-form containing a literal for each variable
- *because of l.o. conditions:* If in a *k*CNF representation of a parity function the least literal is chosen for instantiation by a resolution procedure, then FBDDs constructed always have (without the need for renaming) nodes with l.o. Sets and are thus free of big splits even if the *k*CNF is not converted to 3CNF.

This result can be formally expressed through the following theorem:

Theorem 2: Any parity function $f(x_1, x_2, x_3, \dots x_N)$ of *arity* N and a *k*CNF representation (*k*=N) with N variables has an FBDD with a number of nodes linear in N.

Proof (using induction on N):

Base Cases N=2: C.f. Fig. 9 below. Number of unique nodes=3 (not counting TRUE and FALSE leafs), **N=3**: See Fig. 6 above. Number of unique nodes = 5.

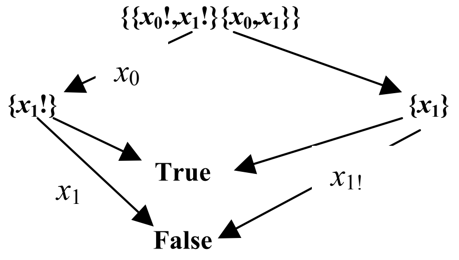

Figure 9

Induction Hypothesis: Suppose parity $f(x_1, x_2, x_3, \dots x_N)$ has an FBDD for the Clause-Set S which is a *k*CNF form of *f* (*k*=N) and this FBDD has a linear number of nodes in N.

Induction Step: For $f(x_1, x_2, x_3, \dots x_N, x_{N+1})$ it suffices to do the following to construct the corresponding FBDD:

a) Extend all Clauses in S with a positive literal representing the new variable and build for the obtained S_1 an FBDD1 using exactly the same procedure you used for S before, i.e., instantiating the same least literals as for $f(x_0, x_1, x_2, \dots x_N)$. This creates a linear number of nodes per induction hypothesis. Then create for the unitarian clauses $\{X_{N+1}\}$ or $\{X_{N+1}!\}$ new sub-trees and connect them if you can. In the example of the base cases of Figures 9, 6: For N=2, $S=\{\{x_0!,x_1!\}\{x_0,x_1\}\}$ shall become $S_1=\{\{x_0!,x_1!,x_2\}\{x_0,x_1,x_2\}\}$ for which FBDD1 looks like:

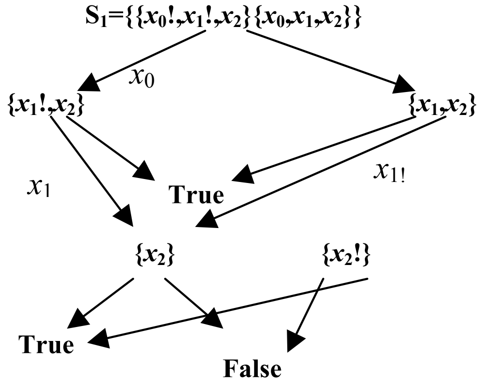

Figure 10: Note that node $\{x_2!\}$ is not connected to the main tree, because no instantiation of S_1 can reach it yet. The number of newly added nodes is only 2.

b) New clauses completing the *k*CNF form of $f(x_1, x_2, x_3, \ldots x_N, x_{N+1})$, must have literals for all variables per definition of the parity function. This implies that they cannot breach the l.o. condition and shall form an s.o. SRT as defined in [Abdelwahab 2016, Section II]. It means also that any new clause C resolved in a sequential resolution step with FBDD1 cannot add any new node, because all its variables will be instantiated using nodes and edges already in FBDD1. Therefore: The total number of added nodes in any step remains only 2. In the example above the following clauses are added to S1 in Fig. 10: $\{\{x_0!,x_1,x_2!\}\{x_0,x_1!,x_2!\}\}$. Both of them don't add any new sub-trees and produce the final s.o. SRT shown in Fig. 6.

(Q.E.D.)

On the other hand: Exponential lower bounds for BDDs are mainly known for Ordered Binary Decision Diagrams (OBDDs), which are the best studied forms of BDDs and which only need one variable order to govern instantiations of Clause-Sets. Alternatively: An FBDD allows the flexibility to choose a different order for each branch. There are many BDD variable ordering heuristics in literature, but the most common way to deal with ordering is to start with something "reasonable" and then swap variables around to improve BDD size. This dynamic variable reordering is usually called sifting [Rudell 1993]. The overall idea is: Based on a primitive "swap" operation that interchanges x_i and x_{i+1}, pick a variable x_i and move it up and down the order using swaps until the process no longer improves the size. *[Abdelwahab 2016] shows a generic way of finding a near-to-optimal variable order without the need for sifting.* It uses, guided by information related to literal alignments and pattern lengths of variables, renaming procedures which enable efficient l.o. transformations of Clause-Sets achieving near-to-optimal node counts.

The first exponential lower bounds on the size of FBDDs have been proven in 1984 by [Zak 1984] and [Wegener 1988]. In his seminal paper *Bryant* also showed [Bryant 1986] that integer multiplication is a function which cannot have a small OBDD irrespective of the variable ordering used. Later, this result was also extended for the FBDD case. A long list of papers followed or were published in the same period which reported similar results for Boolean functions such as: Hamiltonian-Circuit, Perfect Matching, Clique-Only, Triangle-Parity, Blocking Sets in finite projective planes etc. [Gal 1997]. Those results, except the last one, relate to Boolean functions which cannot be expressed in a compact way and pose thus no challenge to the ideas presented in [Abdelwahab 2016]. In [Wegener 2000] a lower bound technique which is influenced by the algorithmic point of view due to [Sieling 1995] is used to explain the methodology behind the majority of results. It turns out that variants of the following observation were constantly used:

"Lemma: Let f be a Boolean function of n variables. Assume that m is an integer, $1<m<n$, if for m any m-element subset Y of the variables $N(f, Y) = 2^m$ holds[44], then the size of any read-once branching program computing f is at least 2^{m-1}."

In [Gal 1997] this *Lemma* is proven and utilized to exhibit a compactly expressed Boolean function which has an exponential lower bound on the number of nodes in its FBDD. *In Section III of [Abdelwahab 2016] this result is taken up as a challenge and the following observations were made:*

[44] N(*f,Y*) denoting the number of different sub-functions obtained under all possible assignments to *Y*.

1. The proof of the above *Lemma* (given in [Gal 1997]) includes showing that the first m-1 levels of any FBDD implementing *f must* form a complete binary tree. This means, intuitively, that for any n there can be no way to find the overall truth value of such an $f(n)$, when expressed in FBDD form, except after trying first all assignment possibilities of m variables for a given $m<n$.
2. *Lemma* could only be applied to the blocking Sets problem, because of the following combinatory property shown to hold for projective planes [Gal 1997]:

   "Fact: Let $J=\{p_1,...,p_t\}$ be a set of $t<=m$ distinct points of the projective plane P, then there exist distinct lines $\{l_1,...l_t\}$ such that for $i>=1$, $j<=t$ we have $p_i \in l_j$ iff $i=j$."
3. This property is used in the lower bound proof as follows:

   "Proof of the theorem. We show that for every q-element subset A of the variables, $N(f_\prod, A) = 2^q$ holds, i.e., each truth assignment to the variables in A yields a different sub-function on the remaining variables. Since each line defines a clause $\bigvee_{i\in\lambda} x_i$ of the function $f_\prod$[45], it follows from the Fact that for an arbitrary q-element subset A of the variables there exist q clauses such that each variable from A appears in exactly one of them, and each variable appears in a different clause."[46]
4. The proof assumes that each line defines a clause, i.e., the projective planes function f_Π is expressed in kCNF form where $k=m=n$, m number of lines and n number of points per line. *This representation preserves the duality between lines and points* which is the characteristic property of projective planes (as seen in Section I) making *Fact* possible in the first place[47]. By successfully expressing in the kCNF description a line by a clause and a point by a literal/variable, duality between clauses and literals/variables is created as well. We call this type of CNF description preserving *all* properties of decision structures of a problem as well as interrelationships between those structures, a *reserved description.*
5. When this kCNF representation is converted into a 3CNF one using an equisatisfiable translation, as needed for algorithms GSPRA$^+$ and/or FGPRA$^+$ described in [Abdelwahab 2016], *this reserved description is broken* and each line is scattered between many clauses. Property 8 of SPR-like algorithms guarantees then that the first m-1 levels of any constructed FBDD, where m>3, *never* form a complete binary tree, since one clause is always picked up for full instantiation.
6. The effect of translating reserved kCNF descriptions to *unreserved* ones, like equisatisfiable 3CNF representations, seems to be overseen in lower bound results published in literature (obtained using variations of this Lemma) including those not related to compactly expressed Boolean functions. This is a reason to formulate the following theorem.

[45] The projective plane function.

[46] (c.f. [Gal 1997] p.15)

[47] Proof of *Fact* is given in the same above reference as follows: "Recall that there are exactly m lines that contain any given point. Let us consider any arbitrary point $p_i \in$J, and the m lines that contain it. Since any two lines intersect in at most one point, each of the other t-1<=m-1 points of the set J belong to at most one of these lines. Thus at least one of the m lines containing *pi* will contain no other point from the set J."

Theorem 3: Lower bounds related to the construction of FBDDs obtained using the above *Lemma* are overthrown by SPR-like algorithms using 3CNF representations.

Proof: As seen above: Proof of *Lemma* includes requiring that the first m-1 levels of any FBDD constructed for such a function to constitute a complete binary tree. On the other hand and whether such a function is compactly expressed or not: SPR-like algorithms using 3CNF representations form *always* trees in which one single clause is chosen for full instantiation (Property 8 in [Abdelwahab 2016, Section II). Therefore, no such tree can have its first m-1 level complete if m>3. (Q.E.D.)

As many of the lower bound results are obtained using a reserved description as well, Theorem 3 encourages us to go a step further and conjecture here without proof: All such lower bound results fail also what we call "the unreserved-description-test", i.e., converting a reserved CNF representation to an unreserved one, not necessarily the equisatisfiable 3CNF representation, and passing the resulting CNF to an SPR-like procedure.

VII REVISITING RANDOMNESS

An attentive reader would have realized by now that we didn't discuss all possible consequences of our work on probabilistic computation. Although Theorem 1 basically states that probabilistic or randomized algorithms (using T- or K-random input bits) cannot exist in finite domain applications, this is still not sufficient to collapse BPP into P or NP. On the other hand: Theorem 1 links de-randomization techniques (like HE) with the P vs. NP question as mentioned in Section II. In literature, P=NP is *not* a de-randomization hypothesis, although many believe that P=BPP [Fortnow 2001][48]. Before we state the main consequences of [Abdelwahab 2016] related to randomized algorithms, it is important to review the current consensus of the scientific community regarding the relation between the P vs. NP question and randomized or probabilistic computation.

1) It is known that P=NP => P=BPP, i.e., if non-determinism is not more powerful than determinism then neither is randomness [Goldreich 2011]
2) Many also believe that P=PBB⊆NP, i.e., determinism is as powerful as randomness and they are both less powerful than non-determinism
3) It is also known that if for any A ∈ BPP it can be shown that:
   a) The number of random bits n can be reduced to become O(log(n)) (cf. fn. 23 in Section II) *and*
   b) A can be de-randomized

   Then P=BPP. In other words: If for every A ∈ BPP there exists a Pseudo Random Generator (PRG) reducing the amount of n random bits needed to O(log (n)), then P=BPP.[49]

[48] This belief has hardened since the discovery that primality testing is in P using a de-randomization argument [Agrawal 2004]

[49] The existence of PRGs can be non-uniformly proven. This is not sufficient to prove P=BPP, though, because to do that one needs to show also that such PRGs succeed against all small circuits.

4) So-called Cryptographic PRGs (CPRG) are PRGs where security is required even against distinguishers with greater running time than the generator. The most common condition in this respect is that the generator should run in a fixed polynomial time, but the adversary can run in an arbitrary polynomial time. CPRGs are built upon the assumption of the existence of one-way functions[50]. One-way functions themselves can only exist if P≠NP [Selman 1992].
5) In [Fortnow 2001], a very useful summary of all known de-randomization hypothesis (as well as P=NP) are drawn as in Fig. 11 and their interdependence discussed in short proofs referring to seminal papers establishing the corresponding results and their interrelations.

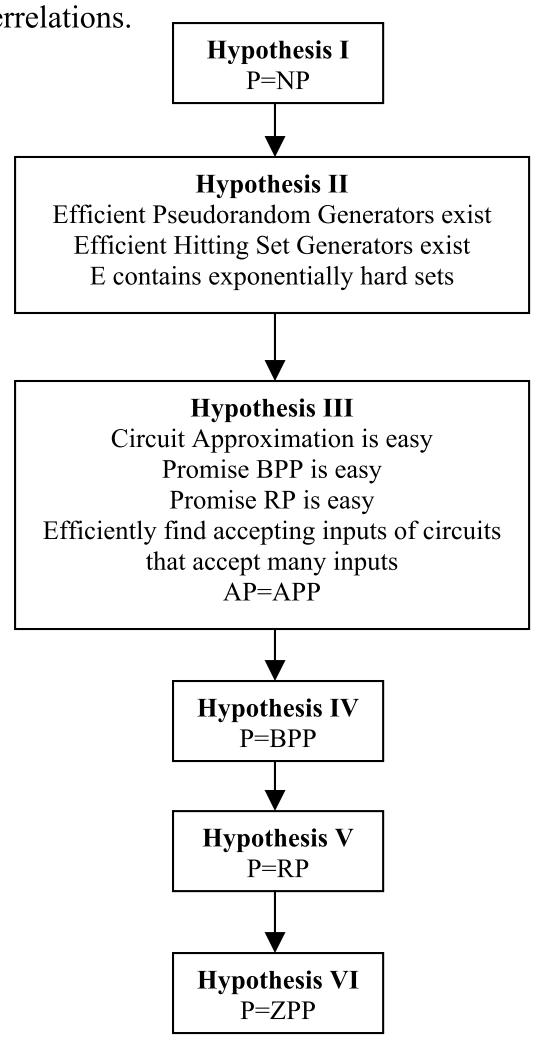
Hypothesis I
P=NP
Hypothesis II
Efficient Pseudorandom Generators exist
Efficient Hitting Set Generators exist
E contains exponentially hard sets
Hypothesis III
Circuit Approximation is easy
Promise BPP is easy
Promise RP is easy
Efficiently find accepting inputs of circuits that accept many inputs
AP=APP
Hypothesis IV
P=BPP
Hypothesis V
P=RP
Hypothesis VI
P=ZPP

Figure 11
Taken from [Fortnow 2001] (Figure 1)

While de-randomization techniques are known and exhaustively studied with respect to the big picture of complexity classes and their relationships as well as properties of algorithms belonging to them, *no effort until [Abdelwhab 2016] has been done to understand and explore the effect of de-randomizations (especially HE) on the nature of logical variables.* This paper (as well as [Abdelwahab 2016]) shows that dual natures of logical variables are side effects of HE and can be used to uniformly define efficient POAs solving 3SAT and thus the P vs. NP question.

[50] One-way function. (2016, August 1). In Wikipedia, The Free Encyclopedia. Retrieved 10:42, August 1, 2016, from https://en.wikipedia.org/w/index.php?title=One-way_function&oldid=732477569

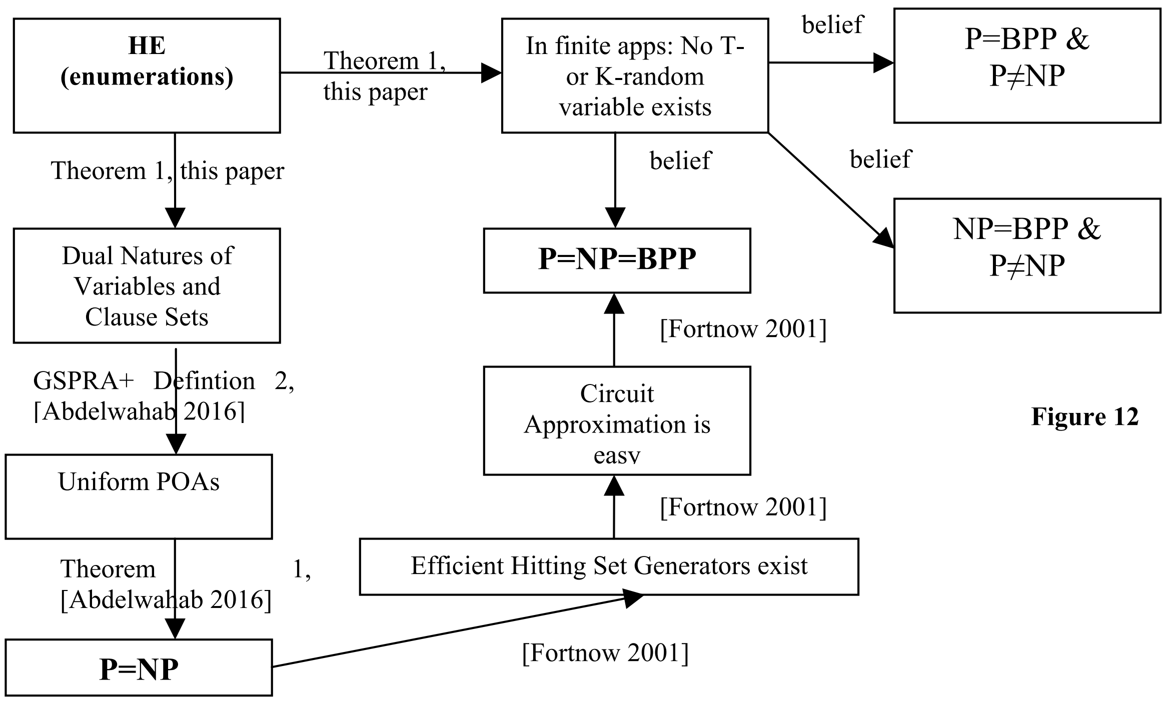
HE (enumerations)
Theorem 1, this paper
In finite apps: No T- or K-random variable exists
belief
P=BPP & P≠NP
Theorem 1, this paper
belief
belief
Dual Natures of Variables and Clause Sets
P=NP=BPP
NP=BPP & P≠NP
[Fortnow 2001]
GSPRA+ Defintion 2, [Abdelwahab 2016]
Circuit Approximation is easy
Uniform POAs
[Fortnow 2001]
Theorem 1, [Abdelwahab 2016]
Efficient Hitting Set Generators exist
P=NP
[Fortnow 2001]

Figure 12

Fig. 12 relates our findings to P=BPP. From the figure it is clear that HE and Theorem 1 of this paper can be thought of as anchor insights in two equally important directions revealing that:

a. Variables possess pattern natures which can be used in constructing efficient POAs leading to a positive solution of the NP question and thus to P=BPP.
b. Current theoretical definitions of AUs (Corollary 1) as well as probabilistic and/or randomized algorithms have to take into account that there *can be neither T- nor K-random variables in finite applications.* The only randomness notion admissible is a statistical one.

The last theorem of this paper presents then known consequences of P=NP in this context:

Theorem 4: The following is true for probabilistic and/or randomized algorithms as well as related complexity classes:

1. Efficient Pseudorandom Generators exist[51]
2. Efficient Hitting Set Generators exist
3. E contains exponentially hard sets
4. Circuit Approximation is easy
5. Promise BPP is easy
6. Promise RP is easy
7. There exists a polynomial time computable function *f* such that for all circuits C that accept at least half of their inputs, C accepts *f*(C) as well.
8. AP=APP
9. P=BPP
10. P=RP
11. P=ZPP
12. One-way functions do not exist

[51] Not to be confused with crypto logical (secure) PRGs.

Proof: Results hold as per Theorem 1 of [Abdelwahab 2016] and from 1-11, because of [Fortnow 2001] (in particular Definitions 2.1, 2.2. and Theorems 2.3, 2.7). The 12th result holds, in particular because of Proposition 1 in [Selman 1992].
(Q.E.D.)

VIII DISCUSSION OF RESULTS

Are logical variables physical objects existing in reality or merely tools constructed by our imagination to facilitate description and solution of logical and mathematical problems? As it turned out: *This is not only a philosophical or epistemological question.* The present paper as well as [Abdelwahab 2016] show that logical variables possess dual natures when they are used in finite applications. On one side, they are mere containers of information and on the other side, they are exhibiting truth patterns giving them concrete positions and roles in any perceivable interpretation of their world. *The fact that variables reveal truth patterns in finite applications contradicts the idealistic assumption that they can be either truly- or computationally random (Theorem 1).* This insight has many effects on our ideas about algorithms. Firstly, it renders the extreme notion of "universality", aiming at true computational independence of input-variables, unrealistic. Secondly, it poses serious questions related to what we want to call *probabilistic computations* restricting their notion of randomness to a notion related only to uncertainty but not fundamental inability as it might be thought. As the pattern view of a logical variable was only recently introduced in [Abdelwahab 2016], it was important to try using it (exclusively) to solve 3SAT efficiently. It turns out that this task is as difficult as attempting to solve 3SAT using the container view only. *This is not a surprise in the context of this paper, because we conjecture that this inability is as deep as other similar inabilities encountered in physics and related to the dual nature of certain pairs of object properties.* The proposed inefficiency principle states that any single sided algorithm using either pattern- or container-views only cannot efficiently solve 3SAT or Dual-3SAT unless there is an efficient, single sided algorithm converting the pattern expression of a logical formula to the container expression and vice versa. *This latter condition is deemed impossible and represents therefore a principal cognition boundary.* The inefficiency principle, whose formulation is detached from the P vs. NP questions, is thought to be a replacement for the famous, commonly accepted conjecture that P≠NP. It doesn't contradict the main P=NP result of [Abdelwahab 2016] in that it concerns only single-sided algorithms while [Abdelwahab 2016] is mainly focused on POAs, which are algorithms using information from both sides of the dual nature. Of course: The P=NP result must stand against all critical attempts as well as known lower bounds. In this paper, Sections V and VI respectively are dedicated to clarify core technical concepts and proofs leading to the P=NP conclusion (facilitating for critical readers the task of finding flaws) and to explain known lower bound results using the new theory. Side effects of the second task are Theorems 2 and 3. Theorem 2 shows that any parity function of arity *k*, although only expressed in an exponential number of clauses (as per known lower bounds) has an FBDD with a linear amount of nodes in *k*, because all nodes in a resolution tree of a *kCNF* parity expression possess automatically l.o. Clause-Sets. Theorem 3 shows that lower bound results obtained using a commonly used technique fail when CNF-representations are converted to 3CNF ones and fed to SPR-like algorithms of [Abdelwahab 2016]. It is also conjectured that results obtained using what we call *reserved CNF expressions* will fail any unreserved-description-test using SPR-like algorithms. A reader who didn't find any flaws in our arguments until Section VI must accept known consequences of P=NP on probabilistic algorithms, beautifully worked out in literature and exposed very shortly in Section VII.

This paper is the last one in a series of three papers completing an endeavor which started by examining the feasibility of the one single application still missing from our world (although desperately needed) namely: The *Fiqh*-Application[52]. Ironically, it is this missing application which gave the whole algorithmic computer science its modern name in the first place. In anticipation, the author as well as some of his interested colleagues who were mainly concerned with Islamic sciences and epistemology, proceeded in the following way:

1) In [Abdelwahab et al. 2014] formal *Fiqh*-systems, described in ancient Muslim scholar texts, were studied and a decision problem for *Fiqh* was formulated linking the answer of queries to a *Fiqh*-machine with the concept of consistent and complete *Fiqh*-legalizations. Those are mostly deductive, logically expressible sets of rules which have to be found in the respective chapters of *Fiqh*, according to established Islamic schools of thought, before mechanical theorem proving is attempted.
2) This study revealed, among other findings, that ancient Muslim scholars successfully used Aristotelian syllogistic proof systems adopting them for expressing and validating their logical *Fiqh*-theories. While doing so they were:
    a) *Mostly concerned with logical form or syntax and how it relates to the intended meaning or semantics.*
    b) Inspired by the almost mathematical structure of classical Arabic Grammar, they were keen to find similar structures in almost all disciplines studied or created for *Fiqh*:
        i) Syllogisms were classified according to pure *syntactical modifications* made to suit an intended purpose
        ii) *Linear Algebra* was invented taking into account *forms* in which equations had to appear to fulfill criteria of efficiency and correctness of computation
        iii) Arabic language recognition rules were extensively formulated to facilitate disambiguating source texts
    c) Following a more deeper characteristic methodology guided by the perception that *the world is full of signs revealing otherwise ambiguous directions and meanings*: "And He has set up on the earth mountains standing firm, lest it should shake with you; and rivers and roads; that ye may guide yourselves; (15) And marks and sign-posts; and by the stars (men) guide themselves. (16)" [Quran, Nahl].
3) In [Abdelwahab 2016], the decisive step of adopting the same methodology of *searching for residual signs of meanings in the syntax of a CNF formula* was made. The result is the discovery that variables are not only names of data containers, but if seen in the appropriate light, also possessing pattern lengths *bridging the gap between the syntactic and the semantic expression of a formula materialized in its truth table*. All other algorithmic consequences of this step were elaborated in [Abdelwahab 2016] which reached in the most direct way the conclusion that P=NP.

[52] *Fiqh* = Islamic Jurisprudence.

4) In this paper, the discovered dual nature of variables is traced back to its creation moment, when a combinatory space of discrete variables is enumerated (de-randomized) and thus given its intended meaning and order. *It is in this moment when variables lose their randomness and render otherwise established views of what algorithms really are inconsistent*. Dual natures of logical variables and Clause-Sets give us yet another important advantage: *We can explain the fundamental inability to efficiently solve the P vs. NP problem using single viewed* algorithm*s in a new, but familiar way.* We conjecture that the deep source of inability is the fact that no single sided algorithms exist which can efficiently switch between pattern and container views. This claim is very similar to inability claims encountered in physics and is therefore called: The Inefficiency Principle. Our methodology is epistemologically well founded realizing that we not only can explain our own positive result, but also why negative results have always been reported until this day.

On a final note the reader is reminded that the most important discoveries of modern ages were those related to cognitive boundaries. All facts and realities of modern science and technology are consequences of understanding cognitive limits and coping with them [Daghbouche 2012]. Most relevant here is the beautifully expressed epistemological statement: "*of knowledge it is only a little that is communicated to you*" [53]. Amazingly enough, this famous inability condition is linked to the mysterious relation between body and soul (form and meaning) as everything else in this paper.

ACKNOWLEDGEMENT

This work is dedicated to all those who respect and admire the humanitarian values and scientific achievements of the great Islamic civilization which made our modern world possible in the first place. Also to those who still cherish hopes for a better future for humanity in spite of all the atrocities committed recently in her name.

REFERENCES

[Aaronson 2010] Scott Aaronson, *Eight signs a claimed P≠NP proof is wrong*, Shtetl-Optimized the Blog of Scott Aaronson, Link: http://www.scottaaronson.com/blog/?p=458

[Abdelwahab 2016] Elnaserledinellah Mahmood Abdelwahab*, Constructive Patterns Of Logical Truth*, J.Acad.(N.Y.)6,1:3-96 2016 and [v2] J.Acad.(N.Y.)6,2:99-199 2016

[Abdelwahab et al. 2014] Elnaserledinellah Mahmood Abdelwahab, Karim Daghbouche, Nadra Ahmad Shannan, *The Algorithm of Islamic Jurisprudence (Fiqh) with Validation of an Entscheidungsproblem*, J.Acad.(N.Y.)4,2:52-87 2014

[Agrawal 2004] Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004), *PRIMES is in P*, Annals of Mathematics. 160 (2): 781–793
doi:10.4007/annals.2004.160.781.JSTOR 3597229

[53] Quran: Al-Israa, 85

[Ahmadi 2012] Amir Ali Ahmadi, Anirudha Majumdar, and Russ Tedrake, *Complexity of Ten Decision Problems in Continuous Time Dynamical Systems*, Proceedings of the American Control Conference October 2012 DOI: 10.1109/ACC.2013.6580838

[Al-Harithy 1987] Howayda Al-Harithy, *Architectural Form and Meaning in Light of AlJurjani's Literary Theories*, MIT Master of Science Thesis, 1987

[Atiyah 2007] Sir Michael Atiyah, *Duality in Mathematics and Physics*, Lecture delivered at the Institut de Matematica de la Universitat de Barcelona (IMUB), Link: https://www.fme.upc.edu/ca/arxius/butlleti-digital/riemann/071218_conferencia_atiyah-d_article.pdf/@@download/file/071218_conferencia_atiyah-d_article.pdf

[Bryant 1986] Randal Bryant, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on Computers, C-35-8, pp. 677-691, August, 1986

[Chaitin 1975] Gregory J. Chaitin, *Randomness and Mathematical proof*, Scientific American 232, No. 5 (May 1975), pp. 47-52

[Daghbouche 2012] Karim Daghbouche, *The Ontological Principle*, J.Acad.(N.Y.)2,4: 160-163 2012

[Deolalikar 2010] Vinay Deolalikar, *P≠NP,* unpublished paper, HP research labs, Palo Alto, Link: https://www.win.tue.nl/~gwoegi/P-versus-NP/Deolalikar.pdf

[Fortnow 2001] Lance Fortnow, *Comparing Notions of Full Derandomization*, Proceedings of Computaional Complexity, Sixteenth Annual IEEE conference, DOIBookmark: http://doi.ieeecomputersociety.org/10.1109/CCC.2001.933869

[Fortnow 2009] Lance Fortnow, *The Status of the P Versus NP Problem*, Communications of the ACM, Vol. 52 No. 9, Pages 78-86,
10.1145/1562164.1562186

[Gal 1996]: Anna Gal, *A simple function that requires exponential size read-once branching programs*, Information Processing Letters 62 (1997) 13-16

[Goldreich 2011] Oded Goldreich, *In a world of P=BPP*, Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation Volume 6650 of the series: Lecture Notes in Computer Science pp 191-232

[Nisan 1994] Noam Nisan, Avi Wigderson, *Hardness vs. Randomess*, Journal of Computer and System Sciences, Volume 49, Issue 2, October 1994, pages 149-167, Elsevier

[Physicist 2009] *Do physicists really believe in true randomness?*, Ask a mathematician, ask a physicist Blog, Link: http://www.askamathematician.com/2009/12/q-do-physicists-really-believe-in-true-randomness/

[Randomness. (2016, August 30)]. In Wikipedia, The Free Encyclopedia. Retrieved 08:00, September 10, 2016, from
https://en.wikipedia.org/w/index.php?title=Randomness&oldid=736945819

[Rubinfeld 2012] Ronitt Rubinfeld, *Randomness and Computation*, lecture notes, Lecture 4, MIT,
permalink: http://people.csail.mit.edu/ronitt/COURSE/S12/handouts/lec4.pdf

[Rudell 1993] R. Rudell, *Dynamic Variable ordering for ordered binary decision diagrams*, ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on Computer-Aided Design, 1993

[Selman 1992] Alan L. Selman, *A survey of one-way functions in complexity theory.* Mathematical systems theory, 25(3):203–221, 1992

[Sieling 1995] D. Sieling and I. Wegener, *Graph driven BDDs - a new data structure for Boolean functions*, Theoretical Computer Science, 141, 1995, 283-310

[Tani 1993] Tani, S., Hamaguchi, *K.*, *The Complexity of the Optimal Variable Ordering Problems of Shared Binary Decision Diagrams*, Proceedings of the 4th International Symposium on Algorithms and Computation, 1993

[Vadhan 2012] Salil P. Vadhan, *Pseudorandomness*; Foundations and Trends in Theoretical Computer Science: Vol. 7: No. 1–3, pp 1-336 (2012),
http://dx.doi.org/10.1561/0400000010

[Vaeaenaenen 1993] A short course on finite model theory, Juko Vaeaenaenen, Department of Math, University of Helsinki, Finland

[Wegener 1988] I. Wegener*, On the complexity of branching programs and decision trees for clique functions*, Journal of the ACM, 35, 1988,461-471

[Wegener 2000] Ingo Wegener, I. Althafer et al. (eds.), *Communication complexity and BDD Lower bound techniques*; Numbers, Information and Complexity, 615-628. Kluwer Academic Publishers 2000

[Whitehead 1963] Whitehead, Alfred North, Russell, Bertrand, *Principia Mathematica*, Cambridge University Press, 2nd edition, 1963

[Williams 2010] Ryan Williams, *Comment on the Blog of rjliption: Goedel's Lost Letter and P=NP*, Subject: *Fatal Flows in Deolaliker's Proofs?* permalink:
https://rjlipton.wordpress.com/2010/08/12/fatal-flaws-in-deolalikars-proof/#comment-5368

[Zak 1984] S. Zak, *An exponential lower bound for one-time-only branching programs*, MFCS'84, LNCS 176, 1984, 562-566

Copyright © 2016 JOURNAL ACADEMICA FOUNDATION • All rights reserved
ISSN 2161-3338 • www.journalacademica.org