Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 6 Issue 10, October - 2020

Natural Substantiation of the Inequality between P and NP

Oleg V. German*
State University of Informatics and RadioElectronics
220600, P. Brovki street, 6,
Minsk, Rep. of Belarus
*Corresponding author e-mail:ovgerman@tut.by

***Abstract*—The paper contains a proof for the hypothesis that classes P and NP do not coincide with the help of the two «natural» postulates. The postulates restrict capacity of the Turing machines and state that each independent and necessary condition of the problem should be considered by a solver (Turing machine) individually, not in groups. That is, a solver should spend at least one step to deal with the condition and, therefore, if the amount of independent conditions is exponentially growing with polynomially growing problem sizes then exponential time is needed to find a solution. To be efficient, the postulate needs a definite type of the problem (called here *TSP* – total solutions problem) which requires to find all valid solutions by means of some efficient generating scheme (substitution system). The paper gives specification of this problem, shows that it belongs to class NP and contains exponentially growing number of solutions which can be defined by means of some algorithmically efficient generating scheme which should be found by a solver. With the postulates, it is enough to build a natural (not pure mathematical) proof that P is not equal to NP.**

Keywords—computational complexity; Turing machine; P versus NP hypothesis; Satisfiability problem

I. INTRODUCTION

If one admits that there is no efficient algorithm for the NP-complete problems like SATISFIABILITY [1] (SAT, for short), then the «try-and-test» strategy remains an essential part of each solver for this type of problems. We show that under quite a general supposition no efficient algorithm exists for SAT. The two hypotheses are required for our goals. The first one is the next: when working with problem conditions a solver (Turing machine) «takes into account» each independent and necessary condition separately from the others (that is, individually, spending at least one step for it). It is beyond our interests the solutions previously found for the problem(s). So, we are interested only in algorithms which find (initially unknown) solutions from the problem statement represented by some set of conditions. The solution process should warrant correctness of the result for each individual problem of the given type.

The second hypothesis sounds like this: no Turing machine (TM) exists with throughput exceeding some constant value (this is similar to the well-known restriction on the light speed in relativity theory).

These hypotheses are sufficient to prove the inequality between P and NP.

To prove P≠NP, a number of significant efforts were undertaken. A short review of them can be found in [2]. Thus, the methods of diagonalization and relativization were used by analogy with the same methods applied to prove the undecidability of some well-known algorithmic problems, such as, for example, the HALT problem. However, as stated in [3], there are different relativizations addmitting both P = NP and P ≠ NP. Also, a schematic approach was used, and we point to the results of A. Razborov [4, 5] who showed superpolynomial complexity of functional schemes in the AND, OR basis to realize a solver for CLIQUE problem. There are no encouring results in the AND, OR, NOT basis as well. The best complexity estimation for the 3-SAT problem is 1.5^n [6], where *n* stands for amount of variables. Other complexity estimations and approaches may be found in [7]. They testify, by the way, that resolution-based solving strategies for SAT have exponential computational complexity.

There exists a viewpoint that P ≠?NP problem is not a pure mathematical and, in particaular, does not depend on the axioms of Zermelo-Fraenkel set theory.

A lot of papers were published with the proofs both of P = NP and P≠NP [8-10]. We can recommend web-site [11] for a closer acquaintance. Many proofs, especially for the statement P = NP were afterwards recognized as incorrect. Some proofs were left without attention, what resembles in some extent the situation with the proofs of the famous Ferma theorem until it was successfully solved by Andriew Wiles. Nevertheless, the P=?NP problem cannot be arbitrarily closed from the further attempts to solve it.

One could seek a solution with the help of phisycal concepts. The idea to unite mathematical and physical approaches is presented, for example, in [12]. It is necessary to bear in mind that a Turing machine is not only an abstract mathematical notion, but also an information processing unit. Due to this, physical analogs get obviously important sense. This relates, for instance, to the notions of information quantity, entropy, throughput of the computational device, and

some others. These notions have, first of all, a physical sense. Namely from these positions with application of necessary mathematical means, the given paper is built. The postulates accepted here, have both mathematical and physical nature, that is why we do not claim that our approach contains a pure mathematical proof of the P≠ NP problem. But mathematical substantiation of the approach is a part of the common approach**.**

II. Formalization of the Prerequisites

Definition. A condition is understood as a formula (formal expression) with variables. A condition can be true or false with respect to its variable values.

Definition. A problem is understood as a non-empty set of conditions. (An arbitrary) solution to a problem is a set of values of its variables satisfying all the explicitly stated conditions of the problem.

When solving a problem, a solver (Turing machine) works with the problem conditions. A part of conditions is stated explicitly, while the other part of conditions is hidden in the problem formulation. Each condition links the solution elements.

A condition is taken into consideration, provided the solver makes one or more steps to test if it is true or false, or treats it in the solution pocess.

A condition that does not logically follows from the other conditions is called independent of those ones.

If the falsehood of a condition leads to the loss of all solutions to the problem, then this condition is called necessary.

Definition. Any necessary and sufficient set of conditions of a problem *A*, which should be taken into consideration by its solver, is called infological set of this problem and designated by Inf_A^{set}. The elements of Inf_A^{set} are called the infs. In general, the problem may have more than one infological set.

If a truth value (true or false) of some condition does not influence on the correctness of a problem solution, then this condition is not included at least in one infological set of the problem. If non-fulfillment of a condition leads to loss of all problem solutions, then this condition is included into each infological set or can be inferred from it. Provided that a condition is independent of the other ones, it should be explicitly presented in infological set as we require of infs.

In what follows, we shall deal only with those conditions (infs) which are necessary to find a solution and independent.

We are interested only in the algorithms, which take infs into consideration, that is, not ignore them, since ignoring inf(s) means that a necessary and independent condition is not considered, so its fulfillment remains indefinite in the course of computation. We shall build a problem not permitting ignoring the infs.

In this paper, we deal with the problems formulated with $n>1$ Boolean variables $x_1, x_2, \ldots, x_n$ (or integer-valued variables taking values from the restricted diapazons). A solution to the problem is represented by some feasible (satisfying) interpretation (a set of $x_1, x_2, \ldots, x_n$ values meeting the problem conditions stated in its specification). This somewhat differs such type of problems from YES-NO problems, however, not essentially. A principal point consists in the following: whether it is sufficient to find any one feasible solution (satisfying interpretation), if it exists, or it is required to find all feasible solutions.

Definition. A *generating scheme* represents a rule (rules), pointing to how to compute each solution to the problem.

Definition. The problems requiring to find any one feasible (valid) solution (if exists) will be called *ASP* (*a*ny *s*olution *p*roblems), while the problems requiring to find some generating scheme to find all solutions, will be called *TSP* (*t*otal *s*olution *p*roblems).

It is clear that if *TSP* has exponentially growing set of feasible solutions then even their simple enumeration requires an exponential time expenses. However, the answer to *TSP* may be in some cases represented as a set of generating rules (substitutions) producing values for the problem variables. To be clear, let us consider a problem with integer-valued variables y_i:

$$52y_1 - 15y_2 - 3y_3 = 2,$$

$$y_1 \in [-2, 2],$$

$$y_2 \in [-3, 7],$$

$$y_3 \in [-23, 2], \text{ all } y_i \text{ are integer.}$$

One matter is to seek any one valid solution, satisfying the above specification. Another matter is to find some substitution system for y_i enabling to get all solutions with polynomial time for each one. In the example, such a substitution system has the next possible form:

$$x_1 + x_2 + x_3 = 2,$$

$$y_1 = x_1 + x_2 - 2x_3,$$

$$y_2 = 3x_1 + 4x_2 - 3x_3,$$

$$y_3 = 2x_1 - 3x_2 - 20x_3,$$

$$x_i \in [0,1], \text{ all } x_i \text{ are integer.}$$

Additionally, it is required that:

R1) the least (the greatest) value of L_i (R_i) for each variable y_i obviously is not less (greater) than the sum of all negative (positive) coefficients in the corresponding substitution for y_i or 0, if there are no such coefficients.

R2) substitutions for x_i are being looked for in a strict order from the substitution system accordingly to

Gauss direct method of exclusions [13, vol.1, p.53] as follows: x_1 is expressed from the substitution for y_1, and x_1 should have coefficient +1 or –1 in this expression. Then expression for x_1 is used instead of x_1 in the expressions for y_2, …, y_n. Then one takes expression for y_2 and gets substitution for x_2 from it. Again, x_2 should have coefficient +1 or –1. The obtained substitution for x_2 is then used in expressions for y_3, …, y_n. The process repeats by analogy for the rest variables x_i, $i = 3, 4, \ldots, n$ by replacing x_i in y_{i+1}, …, y_n. Each time x_i should have coefficient either +1 or –1 in the expression for y_i. The obtained substitutions should contain integer coefficients only. Thus, in the example we have

$$x_1 = y_1 - x_2 + 2x_3; \quad x_2 = -3y_1 + y_2 - 3x_3;$$
$$x_3 = 17y_1 - 5y_2 - y_3.$$

Clearly, from the obtained substitutions with the help of the backward Gauss method one can find the integer-valued substitutions for each x_i with variables y_j $(i, j = 1, \ldots, n)$ only:

$$x_1=89y_1 - 26y_2 - 5y_3; \quad x_2= -54y_1 + 16y_2 + 3y_3;$$
$$x_3=17y_1 - 5y_2 - y_3.$$

Because of the crucial importance of this problem, let us denote $A = \{a_1, a_2, \ldots, a_n\}$, $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}, \boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}, \boldsymbol{d} = \{d_i | d_i = [L_i, R_i]\}$ and use the abbreviation SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) to denote its specification with the following formulation. Let there be given

$$a_1y_1 + a_2y_2 + \ldots + a_ny_n = c, \qquad (1)$$
$$y_i \in [L_i, R_i], \quad (y_i, a_i, L_i, R_i, c \text{ – all integer}).$$

with the known a_i, L_i, R_i, c and unknown y_i, $i = 1,\ldots, n$. Denote the length of the specification (1) by LG_3. Then it is asked, if for the given (fixed) polynomial Ω there exists efficiently verifiable condition and the system of substitutions of the type

$$x_1 + x_2 + \ldots + x_n = c,$$
$$x_i \in [0, 1], \qquad (2)$$
$$y_i = b_{i1}x_1 + b_{i2}x_2 + \ldots + b_{in}x_n, \quad i = 1, \ldots, n,$$
all b_{ij} are integer,

and

1) the sizes of the substitution matrix $B = [b_{i,j}]$ and the sizes of the inverse matrix B^{-1} do not exceed $\Omega(LG_3)$;
2) **R1**, **R2** are satisfied.

Definition [13]. The sizes of a rational number $\gamma = p/q$ (p, q – are integer and coprime numbers, $q \neq 0$), rational vector $c = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ and rational matrix $B = [b_{ij}]$ are defined as follows

$$\textbf{size}(\gamma) = 1 + \lceil \log_2(|p|+1) \rceil + \lceil \log_2(|q| + 1) \rceil,$$

$$\textbf{size}(c) = n + \textbf{size}(\gamma_1) + \ldots + \textbf{size}(\gamma_n),$$

$$\textbf{size}(B) = m \cdot n + \Sigma_{i,j}\, \textbf{size}(b_{ij}),$$

where $\lceil x \rceil$ is minimal integer value greater than x.

Theorem 1. If for conditions (1) there exists system (2), then for each integer-valued set, satisfying (1), there exists a unique integer-valued set, satisfying (2), and vice versa.

Proof. Can be simply obtained from linear algebra provided that matrix $B = [b_{i,j}]$ is not singular. The requirement of integrality of the coefficients is fulfilled by **R1**, **R2**.

Note 1. There may be more than one suitable system (2) for (1) in general case.

Note 2. The number of satisfiable problems SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) is infinite. Elementary technique to generate them consists in the following. Take each substitution $y_i = b_{i1}x_1 + b_{i2}x_2 + \ldots + b_{in}x_n$ starting with $i = 1$ and set $b_{11} = 1$ with the other coefficients b_{1j} $(j > 1)$ representing arbitrary integer values. For $i = 2$, the coefficients b_{2j} $(j <> 2)$ represents arbitrary integer values, besides b_{22}. The value of b_{22} is defined in such a way that after performing substitution for x_1 from $y_1 = x_1 + b_{12}x_2 + \ldots + b_{1n}x_n$ to make $b_{22} = 1$ or $b_{22} = -1$ and so on.

Note 3. Requirement **R2** is not peculiar. One can restrict the substitutions for x_i by y_j $(i, j = 1, \ldots, n)$ only by integer-valued ones, since it is possible to show how an arbitrary integer-valued substitution matrix can be reduced to the form meeting **R2**. For this aim, one should apply the known technique on the basis of Euclidean method for seeking the integer-valued solutions of the linear algebraic equalities with multiple variables and integer (rational) coefficients (see, for instancce [14, p.p. 52–53]). However, the technique, outlined above, is further used to estimate the memory expences for representation of the coefficients of the substitution matrix B and its reverse matrix B^{-1}.

It is clear, that from system (2) the values of x_i are defined elementary and deliver the corresponding solutions to the original problem. Nethertheless, it is also clear that a solution may not exists, what depends on the form of polynomial Ω and initial formulation (1).

Theorem 2. SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) is polynomially reducible to SATISFIABILITY problem.

Proof. There exists a polynomial complexity method to test the condition

$$x_1 + x_2 + \ldots + x_n = c,$$
$$x_i \in [0, 1], \; i = 1, \ldots, n.$$

There also exists a polynomial complexity method to generate values $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ for an arbitrary guess for $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, satisfying the above conditions, and using an arbitrary (a priori unknown) substitution matrix B (with the coefficients also defined as a guess). Also, there exists a polynomially efficient method for testing the conditions

$$a_1y_1 + a_2y_2 + \ldots + a_ny_n = c,$$

$y_i \in [L_i, R_i]$, (y_i, a_i, L_i, R_i, c – all integer).

Finally, there is a polynomially efficient algorithm to get the substitutions for x_i (limited only by the direct forward Gaussian elimination method) to test the fulfillment of **R2**.

It is necessary to note that by means of the given polynomial Ω, one is in position to efficiently define the sizes of the Boolean representation of the integer-valued coefficients of substitution matrix B and its reverse matrix B^{-1}. There remains to point to the fact that maximal sizes of the intermediate coefficients obtained by procedure for verifying the requiremeent **R2**, are restricted by the value $4 \cdot \mathbf{size}(B)$ [13, vol. 1, p. 56] where $\mathbf{size}(B)$ defines the sizes of the substitution matrix B.

The values of intermediate coefficients can be found from the relationships pointed to in Gantmaher's book [15, p. 43] and are expressed through the minors of the matrix B. If M is a maximal absolute value of a coefficient in substitution matrix B, then the value of each minor in B is not higher than $n! \cdot M^n$ what requires no more than $n \cdot Log_2(n \cdot M)$ bits for representation in memory, i.e. is estimated as $O(\mathbf{size}(B))$ with $\mathbf{size}(B)$ not exceeding $n^2 \cdot \Omega(LG_3)$.

From the above considerations, one can conclude that the technique used to reduce SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) to SAT is polynomially efficient. It follows then that if there not exists a polynomially efficient algorithm for SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) with some fixed Ω then SAT has no polynomial solution as well.

Next, we formulate two Postulates, which play decisive role for the goals of this paper.

III. The Postulates

POSTULATE 1.

Any *TSP*-problem requires to individually treat (take into consideration) each independent and necessary condition, spending for this at least one step of the solver (Turing machine – TM) work. That is, if the number of all independent and at the same time necessary conditions is q then the number of steps, TM should perform, is not less than q.

A natural explanation of POSTULATE 1 may be given through the system of logical clauses (disjuncts) D_1, D_2, ..., D_p forming a SAT problem. In fact, if some clause depends on the others (that is, logically follows from them) then that clause can be deleted (not taken into consideration) from SAT without loss of every solution what is necessary for *TSP*-problems. On the contrary, if a clause does not depend on the others than it cannot be deleted and should be taken into account individually or in groups. However, POSTULATE 1 does not admit the last opportunity. Indeed, let $r = p \ \& \ q \rightarrow p$, with p and q mutually independent (r is a group, consisting of the conditions p and q). From the logical value (true/false) of q, one cannot establish logical value of p without analyzing p individually. By this, the value of the whole group r cannot be established without knowing logical values of the conditions p and q from r.

Independence has a fundamental nature. If some condition C is necessary and does not depends on the other conditions, then truth or falsity of those last says nothing about truth or falsity of C. Hence, C should be taken into consideration by necessity. In a standard way, it is adopted that formula C does not depend on the formulas φ_1, φ_2,..., φ_k, provided that there exists some interpretation I, such that $\varphi_1(I)$ & $\varphi_2(I)$ &...& $\varphi_k(I)$ = *true*, but $C(I)$ = *false*.

This definition of independence should be somewhat modified for the paper needs. Let I_1, I_2,..., I_n be particular interpretations and $I = I_1 I_2 ... I_n$ stand for their concatenation. Let $\varphi_1(I) = f(I_1)$, $\varphi_2(I) = f(I_2)$,..., $\varphi_{n-1}(I) = f(I_{n-1})$, $\varphi_n(I) = f(I_n)$, and there exists interpretation $I = I_1 I_2 ... I_n$, in which $\varphi_1(I)$, $\varphi_2(I)$,..., $\varphi_{n-1}(I)$ all are true, and $\varphi_n(I)$ is false. Hence, φ_n does not depend on $\varphi_1(I)$, $\varphi_2(I)$, ... , $\varphi_{n-1}(I)$.

This formulation of independence will be further referred to as independence in private interpretations. Evidently, independence in private interpretations is a particular case of the independence defined in standard way.

Finally, a condition C is a necessary one, provided, that its failure (falsity) leads to the loss of all solutions.

Note that an independent condition is not obligatory a necessary one. A necessary condition may be dependent. The conditions explicitly formulated in a problem specification are necessary ones.

We shall use the next

Lemma$^{\clubsuit}$. Let $\varphi \& F$ be a compatible system of logical formulas. Then from $\varphi \& F \rightarrow H$ follows $\varphi \& \neg H \rightarrow \neg F$ ($\neg$ denotes logical negation).

Proof folows from equivalence $x \rightarrow y \equiv \neg x \vee y$.

Definition of reduction of a problem *A* to problem *B* can be found in [1].

Denote by TH the throughput of TM; by N – the number of steps it performs in order to rich the final state; by I – the number of the infs, TM takes into consideration during its work. Let $TH = I / N$.

POSTULATE 2.

For each totally finite TM_A, and each common problem solved by TM_A, the following relationship is true

$$TH_A \leq c_A < \infty, \qquad (3)$$

where c_A stands for some fixed constant value related to this TM_A.

From (3) one concludes that each physical data processing device cannot process an infinitely many infs (bits of information) per one step (cycle/transmission) of its work. This fundamental principle is asserted, in particular, in [16].

Let us briefly dwell on a possible counter-argument through the well-known acceleration theorem of M. Blum [17]. This theorem asserts that there exist general recursive functions f with values from {0,1} and such that for each TM Z_i, calculating $f(n)$ for $\Phi_i(n)$ steps, there exists another TM Z_j, which calculates $f(n)$ significantly faster for $\Phi_j(n)$ steps with $\Phi_i(n) > 2^{\Phi j(n)}$. Moreover, there is an infinite sequence τ of TMs, calculating f, in which for every neighbor pair of Turing machines Z_s и Z_{s+1} one has inequality $\Phi_s(n) > 2^{\Phi s+1(n)}$, correct for almost all n.

From Blum's theorem, however, it does not follow existence of TM with unlimited throughput. Even if one accepts that TMs, computing the same function, process the equal quantity of information, the Blum's theorem only states for each pair Z_s и Z_{s+1} availability of the fixed number m defined for this pair of TMs, such that for almost each $n > m$ one has $\Phi_s(n) > 2^{\Phi s+1(n)}$. This means that starting from the value m, time expences of Z_s grows significantly faster than time expences of Z_{s+1}. The Blum's acceleration theorem does not impose restrictions on the upper boundary of $\Phi_i(n)$ for all indices i from τ. Hence, for each fixed n, the fastest computation of $f(n)$, say on Z_r, may require an arbitrary many number of steps, while for $l > n$ the fastest computation of $f(l)$ is performed by another TM, say, Z_y with very great value of $\Phi_y(l)$ as well.

We have reach the point in our reasoning where it is required to introduce some consistent *TSP* problem A with exponentially growing sizes of Inf_A^{set} for linearly grow of the variables number n. It requires of us to show that minimum Conjunctive Normal Form (CNF) of this problem grows exponentially in sizes provided, that the number of variables grows linearly. Consequently, this problem cannot be solved for polynomial time whichever solver is used, provided, that all problem infs are taken into consideration and each inf is considered individually, not in groups. The reader evidently has guessed that we intend to use SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω).

IV. The Infs

Some NP-complete problems use the condition formally represented as

$$a_1x_1 + a_2x_2 + ... + a_nx_n = c, \qquad (4)$$

where a_i , c, x_i are integer non-negative numbers and $x_i \in \{0, 1\}$. In particular, some private case of (4) is treated in a Minimum-Size Covering Problem (MSCP) of a 0,1-matrix. SAT can be reduced to (4). Let us call (4) a Container Packing Problem (CPP). One can see that CPP is an NP-complete problem. Now ask, if CPP can be polynomially reduced to equivalent SAT provided, that both problems are specified with the same set of variables? The answer is delivered by

Theorem 3. It is impossible to reduce CPP(x_1, x_2, …, x_n) to equivalent SAT(x_1, x_2, …, x_n) with the sizes of SAT(x_1, x_2, …, x_n) restricted by some polynomial of n.

Proof. Consider the equation

$$x_1 + x_2 + ... + x_n = (n+1)/2 \qquad (5)$$

with odd n.

Let us build CNF for (5) in order to demonstrate that its sizes grow exponentially with linear growth of n. It is easy to comppose the disjunctive normal for (DNF) for (5). To make the proof clear, consider the equation

$$x_1 + x_2 + x_3 + x_4 + x_5 = 3 \qquad (6)$$

with DNF

$$x_1x_2\,x_3\neg x_4\neg x_5 \vee x_1x_2\neg x_3x_4\neg x_5 \vee ... \vee \neg x_1\neg x_2x_3x_4x_5. \qquad (7)$$

Let a conjunction $K_i = (x_1)^{\alpha 1}(x_2)^{\alpha 2} ... (x_n)^{\alpha n}$ deliver a solution to (5) ($(x)^{\alpha} = x$, if $\alpha = 1$ and $\neg x$, if $\alpha = 0$). Call the set $K_i^* = (x_1)^{1-\alpha 1}(x_2)^{1-\alpha 2} ... (x_n)^{1-\alpha n}$ conjugate to set K_i. Disjunction (clause) D_i (D_i^*), associated with K_i (K_i^*) is defined as follows

$$D_i = \neg K_i = (x_1)^{1-\alpha 1} \vee (x_2)^{1-\alpha 2} \vee ... \vee (x_n)^{1-\alpha n}, \qquad (8a)$$

$$D_i^* = \neg K_i^* = (x_1)^{\alpha 1} \vee (x_2)^{\alpha 2} \vee ... \vee (x_n)^{\alpha n}. \qquad (8b)$$

They say that D_α is deducible from D_β, if $D_\beta \subseteq D_\alpha$.

A clause D_j is a simple implicit clause with respect to logical formula $F(x_1, x_2 ..., x_n)$ if

(**i**) $F(x_1, x_2 ..., x_n) \to D_j$,

(**ii**) $\neg\exists\, D_k\,((D_k \subset D_j)\ \&\ (F(x_1, x_2 ..., x_n) \to D_k))$

($\to$ denotes implication).

The notion of a resolvent is standard [18]. Notice here that resolution principle by J. Robinson is a universal inference rule.

A set Z of clauses is called closed if each resolvent (logical consequence), deducible from any subset of Z, belongs to Z.

A subset Π of the set of clauses Z is called a covering set of Z provided, that each clause from Z either belongs to Π or can be deduced from Π. A covering set $\Pi^{\min}$ is a minimal if it consists of the minimum number of clauses among all sets, covering Z.

Each subset of the independent clauses of the set R is called a kernel of R. Obviously, each $\Pi^{\min}$ is a kernel. The opposite, however, may be false.

Return to (6). It is clear that conjunction

$$K_\alpha = x_1x_2\,x_3\neg x_4\neg x_5$$

belongs to DNF of (6) and defines the associated clause (8*a*)

$$D_\alpha = \neg x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4 \vee x_5.$$

D_α is not deducible from the set of clauses SAT, equivalent to (6) (thereafter denoted as D-SAT). This means, that there exists an interpretation I in which each clause from D-SAT is true, while D_α is false, and this interpretation is $x_1 = x_2 = x_3 =$ '1'; $x_4 = x_5 =$ '0' delivering true value to K_α.

Further, no one clause D_β, $D_\beta \subset D_\alpha$ can be deduced from D-SAT, e.g. $\neg x_1 \vee \neg x_2$, or $x_4 \vee x_5$, or $\neg x_1$ cannot be deduced from D-SAT. It is also clear that D-SAT is not satisfied with each set conjugate to K_i from DNF representation of (5), (6). For instance, D-SAT is not satisfied with the set $(K_\kappa)^* = \neg x_1 x_2 x_3 \neg x_4 \neg x_5$, determining associated clause $(D_\kappa)^* = x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4 \vee x_5$.

Consider clauses D_α and $(D_\alpha)^*$. From our considerations follows that the clauses

$$\chi_1 = x_4 \vee x_5, \ \chi_2 = \neg x_3 \vee x_5, \ \chi_3 = \neg x_2 \vee \neg x_3 \vee x_4$$

and the others of that kind are not deducible from D-SAT, that is, do not belong to the clauses of D-SAT and cannot be deduced from them by means of the existing inference rules, since they are refuted in some intrepretation satisfying (6). Let us now find such an arbitrary minimum-size subset of the clause $(D_\alpha)^*$ which is deducible from D-SAT by virtue of the inference rules (in particular, by the resolution rules). The sizes of a clause (its subset) define the number of literals in it. So, $d^\wedge = x_1 \vee x_4 \vee x_5$ represents one such a subset ($d^\wedge \subset (D_\alpha)^*$). One can easily convince himself (herself) that each interpretation, satisfying (6), satisfies $d^\wedge$ as well. From this, and well-known K. Gedel's theorem about equivalency of syntactic and semantic deducibility [8], one concludes that $d^\wedge$ either belongs to D-SAT or can be inferred from D-SAT with inference rules. We are dealing with D-SAT with minimum possible sizes, that is, coinciding with a $\Pi^{\min}$ for (6).

It is clear that any proper subset of $d^\wedge$ cannot be deduced from D-SAT. For instance, $x_1 \vee x_4$ is not deducible from D-SAT since it belongs to the clause $D' = x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5$ not deducible from D-SAT as was explained above. Let us notice that no one clause with a single or two literals is included in D-SAT or deduced from D-SAT as it is a proper subset of some broader clause that is not derived from D-SAT. Next, there are no clauses (nor they can be deduced) in D-SAT with 3 letters like those

$$x_i \vee x_j \vee \neg x_k, \quad x_i \vee \neg x_j \vee \neg x_k, \quad \neg x_i \vee \neg x_j \vee \neg x_k$$

by the same reason: they represent a proper part of some underivable clauses. Hence, in D-SAT may be included the clauses of the form $x_i \vee x_j \vee x_k$ or some subset of more broader clauses, from which logically follows $x_i \vee x_j \vee x_k$. From this follows that $d^\wedge$ may be deduced in the best case from a group of clauses $\varphi_{\min}$ with at least two (deducible) clauses, e.g.

$$\varphi_{\min} = \begin{matrix} x_1 \vee \neg x_2 \vee x_4 \vee x_5, \\ x_1 \vee x_2 \vee x_4 \vee x_5. \end{matrix}$$

Each clause from $\varphi_{\min}$ provided, that it belongs to D-SAT, can be replaced by $d^\wedge$, from which it follows. Consequently, since our goal is to refute including $d^\wedge = x_1 \vee x_4 \vee x_5$ in D-SAT, it is necessary to recognize that both clauses $x_1 \vee \neg x_2 \vee x_4 \vee x_5$ and $x_1 \vee x_2 \vee x_4 \vee x_5$ can be deduced without use of $d^\wedge$. Consider the first of them

$$d^{(A)} = x_1 \vee \neg x_2 \vee x_4 \vee x_5.$$

Clearly, any subclause of $d^{(A)}$ with fewer letters does not belong to D-SAT and cannot be deduced from it, excluding $d^\wedge$ (but we have supposed that $d^\wedge \notin$ D-SAT). There is a possibility (among the others!) to deduce $d^{(A)}$ from another two deducible from D-SAT clauses:

$$d^{(B)} = x_1 \vee \neg x_2 \vee x_4 \vee x_5 \vee x_3$$

and

$$d^{(C)} = x_1 \vee \neg x_2 \vee x_4 \vee x_5 \vee \neg x_3.$$

Again, no one subclause of $d^{(C)}$ can be deduced from D-SAT, excluding $x_1 \vee x_4 \vee x_5$. So, consider this second clause $d^{(C)}$. Any proper its subclause is underivable and does not belong to D-SAT, excluding $x_1 \vee x_4 \vee x_5$. Moreover, $d^{(C)}$ is not deduced from any broader clauses as there are no such clauses anymore. So, $d^{(C)}$ cannot be deduced with the help of resolution inference rules. There remains a possibility to deduce $d^{(C)}$ with the help of some clause $d^* \in \Pi^{\min}$ with lesser sizes. Clearly, d^* cannot be a proper part of $d^{(C)}$ as we accepted. Now, represent $\Pi^{\min} = F \& d^*$ where Φ and d^* are mutually independant (d^* does not follow from F). Then $F \& d^* \rightarrow d^{(C)}$. By Lemma$^\clubsuit$, $F \& \neg d^{(C)} \rightarrow \neg d^*$. Replace $\neg d^{(C)}$ by $\neg x_1 x_2 \neg x_4 \neg x_5 x_3$. Clearly,

$$F \& \neg d^{(C)} = \neg x_1 x_2 \neg x_4 \neg x_5 x_3, \qquad (9)$$

and

$$\neg x_1 x_2 \neg x_4 \neg x_5 x_3 \rightarrow \neg d^*. \qquad (10)$$

(The other possibility is $F \& \neg d^{(C)} = \square$ ($\square$ denotes false). From this, $F \rightarrow d^{(C)}$, what contradicts to the fact that to deduce $d^{(C)}$, one needs to use d^*.)

Again, $\neg d^*$ stands for some part of the conjunction $\neg x_1 x_2 \neg x_4 \neg x_5 x_3$. In these terms, d^* stands for some part of the clause $d^{(C)} = x_1 \vee \neg x_2 \vee x_4 \vee x_5 \vee \neg x_3$. As we now know, no proper part of $d^{(C)}$ is valid besides that one containing $d^\wedge = x_1 \vee x_4 \vee x_5$. However, in this case, d^* may be replaced by $d^\wedge$ with preserving the sizes of $\Pi^{\min}$.

This reasoning, remains valid with respect to any clause of the form $x_i \vee x_j \vee x_k$ with the pairwise different i, j, k.

We can generalize our considerations to an arbitrary case (5). Consider a minimum-size positive deducible clause

$$d^{(D)} = x_{i1} \vee x_{i2} \vee \ldots \vee x_{i(n+1)/2}.$$

Then it may be deduced from the next pair of the clauses

$$d^{(E)} = x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee x_k,$$
$$d^{(F)} = x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee \neg x_k.$$

For the second clause $d^{(F)}$, one cannot use fewer number of positive letters since it is refuted in some satisfying interpretation for the SAT (5). Let's denote this SAT as G-SAT. Consider this second clause $d^{(F)}$ and notice that any its proper subclause does not belong to G-SAT nor is deduced from G-SAT, besides $d^{(D)}$. Hence, if this clause is deducible from the other clauses, then it should of course, be deducible from the pair of clauses

$$d^{(E)} = x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee \neg x_k \vee x_r,$$
$$d^{(F)} = x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee \neg x_k \vee \neg x_r.$$

Each of the above clauses is either deducible from G-SAT or belongs to G-SAT. Again, any proper subclause of $d^{(F)}$ is not deducible from G-SAT nor belongs to it, excluding $d^{(D)}$. Hence, as we excluded belonging $x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2}$ to G-SAT, then either $d^{(F)}$ is deduced from the other clauses of the G-SAT, or belongs to the G-SAT. In the last case, this clause can be replaced by $d^{(D)}$. In the first case, one should consider the next pair of the deducible clauses resolving in $d^{(F)}$. One of the clauses of that pair would be the clause $x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee \neg x_k \vee \neg x_r \vee \neg x_t$. This process can be continued by analogy and stops after all G-SAT variables are used. Then, there remains a possibility to replace one of the remaining clauses of the form

$$d^{(G)} = x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2} \vee \neg x_{i(n+1)/2+1} \vee ... \vee \neg x_n$$

by $x_{i1} \vee x_{i2} \vee ... \vee x_{i(n+1)/2}$. Clearly, $d^{(G)}$ cannot be deduced from any pair of clauses provided, that none of them contains the subclause $d^{(D)}$ and, therefore, may be replaced by $d^{(G)}$ from which it follows. More strictly,

$\Pi^{\min} = \Phi\_ \,\&\, d^*\_$ where $\Phi\_$ and $d^*\_$ are mutually independant ($d^*\_$ does not follow from $\Phi\_$ and has sizes smaller than the sizes of $d^{(G)}$) and $\Phi\_ \,\&\, d^*\_ \rightarrow d^{(G)}$. As was shown earlier, in these terms $d^*\_$ should contain $d^{(D)}$ otherwise it fails in some valid interpretation for G-SAT.

There remains to make a final step in the proof. In each conjugate set K_i^* one can define a unique clause $d_i^\wedge$ ($d^{(D)}$) (like that one considered above, and consisting with positive letters only ($d_i^\wedge$ is written with the positive letters from the representation of $D_i^* = \neg K_i^*$). Clearly, all such clauses $d_i^\wedge$ ($d^{(D)}$) form a kernel and belong (as we have shown) to some minimum-size covering set $\Pi^{\min}$ of G-SAT for all possible cases of $d_i^\wedge$ ($d^{(D)}$). The number of such clauses $d_i^\wedge$ ($d^{(D)}$) is $C_n^{(n+1)/2} = O(2^{n/2})$. By this, one concludes that minimum-size CNF for (5) contains exponentially growing number of clauses with respect to the number of variables n.

V. The SUBST($\boldsymbol{Y}, \boldsymbol{X}, A, \boldsymbol{D}, N, C, \Omega$) Problem

Consider a system of substitutions

$$y_i = a_i + b_{i1}x_1 + b_{i2}x_2 + ... + b_{in}x_n, \quad i = 1, n, \qquad (11)$$

with integer a_i, b_{ij}, and binary x_i. The main requirement to the substitutions (11) is to provide the uniquness of the transformation, or in the other words, to ensure that each set $\boldsymbol{x}^* = \langle x^*_1, x^*_2, ..., x^*_n \rangle$ is mapped to unique set $\boldsymbol{y}^* = \langle y^*_1, y^*_2, ... y^*_n \rangle$ (the opposite is obvious). The said requirement is fulfilled by nondegeneracy of the transformation matrix $B = [b_{ij}]$.

From (11),

$$\boldsymbol{y} = B^{-1} \cdot (\boldsymbol{x} - \boldsymbol{a}). \qquad (12)$$

Theorem 4 [13].

1. If the system of rational equations (11) is consistent then it has a solution $\boldsymbol{y}$ with the sizes restricted by some polynomial $\wp_1$ of the sizes of B^{-1} and $(\boldsymbol{x} - \boldsymbol{a})$.
2. The inverse matrix B^{-1} has the sizes, restricted by some polynomial $\wp_2$ of the matrix B sizes.

Notice that we are interested in the values of y_i which define the values of $x_i \in \{0,1\}$. For convinience, let us set $\boldsymbol{b} = 0$. From this, the sizes of $\boldsymbol{y}$ are restricted by some polynomial $\wp_3(\text{size}(B)) = \wp_1(\wp_2(\text{size}(B)))$.

One can see that for $m = n$ and some fixed constant k, $\textbf{size}(B) \le O(k \cdot n^2 \cdot (\max_{i,j} \textbf{size}(b_{ij})))$. So, there remains to provide that the value of $\textbf{size}(b_{ij})$ grows not faster than some fixed polynomial. However, this requirement is trivial as in selecting the coefficients of the matrix B one should preserve only its nodegeneracy ($\det B \neq 0$).

We have reached the final point. Accordingly to SUBST($\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega$), it is necessary to build a function f (generated by the system of substitutions) mapping each satisfying interpretation $I(\boldsymbol{x}^*)$ (x-values) satisfying system (2), to the unique interpretation $I(\boldsymbol{y}^*)$ (y-values) satisfying system (1) or vice versa. It was demonstrated that the number of feasible interpretations for the system (2) grows exponentially with linear growth of the number of variables n. The reverse function f^{-1} can be found (in the form of a reverse matrix B^{-1}) provided, that the matrix B of substitutions is nondegenerate. Therefore, if to consider each pair of interpretations ($\boldsymbol{x}^*, \boldsymbol{y}^*$) satisfying to (1, 2), separately from the other pairs ($\boldsymbol{x}, \boldsymbol{y}$), then it is necessary to consider exponentially growing number of all pairs ($\boldsymbol{x}, \boldsymbol{y}$), and by POSTULATE 2 to spend exponentially growing time to solve SUBST($\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega$) in general. However, let us try to refute this conclusion and suppose that after establishing some part of all pairs ($\boldsymbol{x}, \boldsymbol{y}$), satisfying (1, 2), the remaining pairs of interpretations may be not considered and, therefore, be ignored by the solver. Again, to make our reasoning clearer, consider an illustration. Let the generating rule be as before

$$x_1 + x_2 + x_3 + x_4 + x_5 = 3. \qquad (13)$$

The next table shows all feasible pairs of interpretations, satisfying the systems (1, 2).

Table 1. The pairs of iterpretations deliverig the solutions to the systems (1, 2)

x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5
0	0	1	1	1	y_{11}	y_{21}	y_{31}	y_{41}	y_{51}
0	1	0	1	1	y_{12}	y_{22}	y_{32}	y_{42}	y_{52}
0	1	1	0	1	y_{13}	y_{23}	y_{33}	y_{43}	y_{53}
0	1	1	1	0	y_{14}	y_{24}	y_{34}	y_{44}	y_{54}
1	0	0	1	1	y_{15}	y_{25}	y_{35}	y_{45}	y_{55}
...	...	...	...	...		...	...	...	...
1	1	1	0	0	$y_{1,0}$	$y_{2,10}$	$y_{3,10}$	$y_{4,10}$	$y_{5,10}$

We assume that the rows of the table 1 are arranged in descending strong order of values in the column y_1. This assumption does not violate the strength of the results obtained. With the help of the rule **R2**, it is possible to generate an infinite number of the individual problems SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) satisfying this assumption. For this, one should use the first substitution

$$y_1 = b_{11}x_1 + b_{12}x_2 + \ldots + b_{1n}x_n \qquad (14)$$

for y_1 with the coefficient $b_{11} = 1$ and each subsequent coefficient $b_{1j} > \sum_{k<j} b_{1k}$. It may be proved by induction on the number of variables in the substitution (14) that in this case the values of y_1 would be arranged in desceinding order (leave this to the reader). Consider, for example, the fifth row in the table 1

1	0	0	1	1	y_{15}	y_{25}	y_{35}	y_{45}	y_{55}

Suppose that it depends on the previous rows in the table 1. This means, that the values y_{15}, y_{25}, y_{35}, y_{45}, y_{55} automatically fall into ranges $[L_i, R_i]$ provided, that the previous rows $<y_{1i}, y_{2i}, y_{3i}, y_{4i}, y_{5i}>$ have fallen into ranges $[L_i, R_i]$. However, before solving any individual problem SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) (including the considered one) this fact is unknown, and there exist two types of SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω) such that y_{15} falls into $[L_1, R_1]$ for the first type, and does not fall into $[L_1, R_1]$ for the second type of SUBST($\boldsymbol{y}$, $\boldsymbol{x}$, A, $\boldsymbol{d}$, n, c, Ω). Indeed, in comparison with the previous row $<y_{14}, y_{24}, y_{34}, y_{44}, y_{54}>$ the following relations take place $y_{15} < y_{14}, y_{14} \in [L_1, R_1]$. Then, by increasing the value of L_1 one can provide that $y_{11}, y_{12}, y_{13}, y_{14} \in [L_1, R_1]$, but $y_{15} \notin [L_1, R_1]$. This means that the row $<y_{15}, y_{25}, y_{35}, y_{45}, y_{55}>$ should be taken into consideration by the solver even in the case that the previous rows satisfied the corresponding ranges $[L_i, R_i]$. Note that lexicographical descending order of the rows with interpretations can be obtained by a simple indeces mixing in variables x_1, x_2, x_3, x_4, x_5 as they are mutually independent.

The situation, we have described, is applicable to any part of the table with the interpretations (including the entire table) and means that each row of the table would be considered separately by virtue of POSTULATE 1 as a formula independent in particular interpretations from the other ones. Indeed, one can define the next formulas

$\varphi_1(I) = h(\text{SUBST}(\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega), I_1)$, $\varphi_2(I) = h(\text{SUBST}(\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega), I_2)$, ..., $\varphi_n(I) = h(\text{SUBST}(\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega), I_n)$ where $h(\text{SUBST}(\boldsymbol{y}, \boldsymbol{x}, A, \boldsymbol{d}, n, c, \Omega), I_i)$ is true for the i-th row of the interpretation table 1 with $I_i = (\boldsymbol{x_i}, \boldsymbol{y_i})$ ($\boldsymbol{x_i}$, $\boldsymbol{y_i}$ represent the corresponding x-set and y-set in the i-th row) if and only if $\boldsymbol{A}(\boldsymbol{y_i})^{\mathrm{T}} = c$ and each member of $\boldsymbol{y_i}$ falls into the corresponding range d_i from $\boldsymbol{d}$ we have showed above, $\varphi_1, \ldots, \varphi_n$ are independent in private case. By POSTULATE 1, each of these functions should be considered by the solver separately from the others.

This last remark completes the natural proof for P ≠NP.

VI. Conclusion

The proof of P≠NP given in the article is not purely mathematical, since it uses, in any case, one purely physical postulate about the limited capacity of the Turing machine.

An obvious correspondence to the postulate of the theory of relativity restricting the speed of light can be found if we draw an analogy with the emission of a photon of light and the operation of transition in a Turing machine (in this case, the speed of light is a physical analogue of the speed of information processing by a Turing machine).

The postulates we have introduced characterize the understanding of complexity that is intuitively used by the algorithms developers. Therefore, we are not talking about a "universal" mathematical proof of the P≠NP formula, but about a proof within the framework of the accepted postulates.

References

[1] Garey M., Johnson D. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., New York, 1979. – 340p.

[2] Fortnow L. The Status of the P versus NP Problem. Communications of the ACM. 2009, 52(9). p.p.78–86.

[3] Baker Th., Gill J., Solovay R. Relativization of the P=?NP question. SIAM J. of Computing. 1975. 4(4), p.p. 431– 442.

[4] Razborov A. A. On the method of approximation. In Proceedings of the 21st ACM Symposium on the theory of Computing. New York, 1989, p.p. 167–176.

[5] Razborov A. A., Rudich S. Natural proofs. Journal of Computer and System Sciences. 1997. 55(1), p.p.24 – 35.

[6] Cook S. The P versus NP problem. Clay Mathematics Institute. Retrieved 18 October 2006. 12p.

[7] Melkebeek van D. A survey of lower bounds for satisfiability and related problems. Foundation and trends in theoretical computer science. 2007, vol. 2, p.p. 197– 303.

[8] Diaby M. The Traveling Salesman Problem: A Linear Programming Formulation. WSEAS Transactions on Mathematics. Issue 6. vol. 6. June 2007, p.p. 745 –754.

[9] Craig Alan Feinstein. An elegant argument that P != NP. Progress in Physics. 2011, vol. 2, p.p. 30 – 31

[10] Jorma Jormakka. On the existence of polynomial-time algorithms to the subset sum problem. arXiv e-prints: https://arxiv.org/abs/0809.4935, 2008.

[11] https://www.win.tue.nl/~gwoegi/P-versus-NP.htm.

[12] Aaronson S. NP-complete problems and physical reality / ACM SIGACT News. 2005, vol. 36(1), p.p. 30–52.

[13] Schrijver A. Theory of Linear and Integer Programming. John Wiley and Sons. N.Y., 1986.

[14] Sushkewich A. K. Number Theory. Elementary Course. Kharkov Univ. Press. (Ukraine, USSR). 1954 (In Russian) [Sushkewich A.R. Teoria chisel. Elementarny kurs. Kharkow Universitet, 1954].

[15] Gantmaher F. R. Theory of Matrices. Moscow Science. The 3-rd Edition. 1967 (in Russian) [Gantmaher F. Teoria Matric. Moskwa. Nauka, 1967. Izdanie 3. – 576s]

[16] Harmuth H. F. Information Theory Applied to Space-Time Physics. The Catholic univ. Of America, Washington, DC. 1989. –350p.

[17] Blum M. A machine-independent theory of the complexity of recursive functions. Journal of the Association for Computing Machinery. 14,№ 2 (1967), 322–336.

[18] Chang C-L., Lee R. Symbolic Logic and mechanical Theorem Proving. Academic Press., New York., 1973. –360p.