International Journal of Pure and Applied Mathematics
Volume 120 No. 3 2018, 497-510
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v120i3.17

ijpam.eu

THE SOLUTION OF THE 3RD CLAY MILLENNIUM PROBLEM. A SHORT PROOF THAT $P \neq NP = EXPTIME$ IN THE CONTEXT OF ZERMELO-FRANKEL SET THEORY

Konstantinos Kyritsis

Department of Accounting-Finance
University of Applied Sciences (TEI) of Epirus
GREECE

Abstract: In this paper I provide a very short but decisive proof that $P \neq NP$, and $NP = EXPTIME$ in the context of the Zermelo-Frankel set theory and deterministic Turing machines. We discuss also the subtle implications of considering the P versus NP problem, in different axiomatic theories. The results of the current paper definitely solve the 3rd Clay Millennium problem P versus NP, in a simple and transparent away that the general scientific community, but also the experts of the area, can follow, understand and therefore become able to accept.

AMS Subject Classification: 68Q15
Key Words: 3rd Clay millennium Problem, $EXPTIME$-complete problems, NP-complexity, P-complexity

1. Introduction

In the history of mathematics, it is known that difficult problems that have troubled a lot the mathematicians, turned out to have different proofs one simple and one very complex. Such an example is if the general 5th order polynomial equation can be solved with addition, subtraction, multiplication , division and extraction of radicals starting from the coefficients. This was a problem that troubled the mathematicians for centuries! No doubt a very

Received: October 30, 2017
Revised: February 12, 2018
Published: January 14, 2018

difficult problem. Yet it was the famous mathematician Niels Henrik Abel who gave a very simple proof that a general 5^{th} order polynomial equations cannot be solved so, in about 5-6 pages! On the other hand the proof of the same, by the E. Galois theory, is a whole book of dozens of pages! This is not strange, as it depends on the right twist of semantics and symbols in mathematics. The proof in the way of E. Galois is much longer as it shows also how to solve any such equation when it is solvable! The fact that N. H. Abel solved it in only 5-6 pages should not make us think that it was an easy problem!

We may make the metaphor that the proof that a general polynomial equation of the 5th order cannot be solved with radicals, is like a mountain path. It is not long, it is say only one kilometre, but at a particular point in order to cross it, small bridge is required, that no-one has ever constructed for...centuries! The unconstructed bridge is the metaphor of a particular type of abstarct thinking, that non-one was able to compose so far, therefore no-one has been able to walk this mountain path, or in other words non-one has been able to solve the particular problem. It was N.H. Abel who though of the abstraction of the structure of the group of permutations of 5 symbols (the missing bridge) who was able to prove it! The proof of E. Galois may correspond to our metaphor, to another long mountain path, that passes from the start and end points of the N.H. Abel's path, but also climbs up all the mountain and it is of course longer say 10 kilometres.

A famous mathematician once said that **"Once a proof is known to a mathematical problem, then immediately after the publication the problem becomes trivial!"**

It is the same with the solution of the P versus NP problem in this paper. We will utilize in our proofs, the **key abstraction** of the existence of an $EXPTIME$ complete language, (it is known that it exists) without specifying which one, which will simplify much the arguments. Then we synthesize other languages and arguments over it, that will solve the problem.

A second issue that is important to mention, is a statement, that is usually attributed to the famous mathematician Yuri Manin, that "A correct proof in mathematics is considered a proof only if it has passed the social barrier of being accepted and understood by the scientific community and published in accepted Journals"

Passing the obstruction of the social barrier, sometimes is more difficult than solving the mathematical problem itself!

An example in the history of mathematics is the Hyperbolic or Bolyai-Lobachevskian geometry. If it was not that the famous and well established J.C.F. Gauss, assured the mathematic community that he himself had also

discovered this geometry, the scientific community, would not accepted it so easily. Gauss also mentioned that he refrain from publishing it for obvious reasons. It seems that he was afraid that he might be ridiculed, and so although he himself seemed that he had solved the very famous and century old problem of the independence of the 5^{th} axiom of parallel lines in the Euclidean geometry, he did not dare to try to pass also the social barrier.

These two observations seem to apply also to the famous 3^{rd} Clay millennium problem of P versus NP.

We must notice here that the P versus NP problem, is in fact a set of different problems within different axiomatic systems. And in the context of what axiomatic system is the Complexity Theory of Turing machines? Since the complexity theory of Turing machines requires entities like infinite sets of words etc then it is in the context of some axiomatic set theory, together with the axiom of infinite. So we notice that the next are different problems:

1. The P versus NP problem in the Zermelo-Frankel axiomatic system of sets without the axiom of choice and this axiomatic system formulated in the 2^{nd} order formal languages.

2. The P versus NP problem in the Zermelo-Frankel axiomatic system of sets with the axiom of choice and this axiomatic system formulated in the 2^{nd} order formal languages.

3. Etc

We might try to think of the P versus NP problem within the context of the axiomatic system of Peano Arithmetic with or without the axiom of induction and within second order formal languages. But to do so, we must carefully define, what additional axioms or definitions give the existence of infinite subsets of natural numbers that are used in the Computational Complexity Theory.

My main hidden guiding idea in searching for such a simple proof, was that what the "arbitrary human-like free-will" of a non-deterministic Turing machine as human-machine interactive software (e.g. in password setting), can do in polynomial time cannot be done by a purely mechanical deterministic Turing machine in polynomial time. (See also the beginning of paragraph 4). After the Key-abstraction mentioned above I had to find the right simple arguments to make a valid proof of this idea. The proof of the P versus NP problem in the direction $P \neq NP$, is supposed also to mean that the standard practice of encryption in the internet, is safe.

We notice also that the P versus NP:

1. It is a difficult problem, that has troubled the scientific community for some decades.

2. It may have simple proofs of a few paragraphs, hopefully not longer than the proof of the Time Hierarchy theorem, which seems to be a deeper result.

3. But it can also have very lengthily and complex proofs, than may take dozens of pages.

4. There are many researchers that have claimed to have solved it, either as $P = NP$, or as $P \neq NP$, and even suggestions that neither are provable, but only a handful of them seem to have been able to pass the preliminary social barrier and publish their solution in conferences or Journals with referees. It seems to me though that it is not probable that all of them have correct solutions. Especially in the direction P=NP, there is a common confusion and mistake, that has been pointed out by Yannakakis M. 1998 in [12]. Furthermore this confusing situation has contributed so that although there are publications in respectable Journals, the experts and the scientific community do not seem of being able to decide if the P versus NP problem has been solved or not. This is reasonable, as there are proofs of close to 100 pages, and no average reader would feel comfortable to go through them, and decide for himself if there is a flaw or error somewhere. Still it is better to have published results than non-published, and then let the large number of readers to try to find errors or flaws in the solutions if there are any.

So here comes the need of a more challenging problem: Not only to solve the P versus NP problem, but also solve it in such an simple, elegant and short way, so that the researchers will know a decisive proof that can understand and control that $P \neq NP$ or not, so short that anyone familiar with the area, would discover any flaw or error if it existed.

This is I believe the value of the present paper that provides such a proof in the context of the Zermelo-Frankel set theory (we do not use the axiom of choice), formulated within 2^{nd} order formal languages.

What this proof is or is not:

1. It does not introduce new theoretical concepts in computational complexity theory so as to solve the P versus NP.

2. It does not use relativization and oracles.

3. It does not use diagonalization arguments, although the main proof, utilizes results from the time hierarchy theorem.

4. It is not based on improvements of previous bounds of complexity on circuits.

5. It is proved with the method of counter-example. Thus it is transparent short and "simple". It takes any $EXPTIME$-complete DTM decision problem, and from it, it derives in the context of deterministic Turing machines a decision problem language which it is apparent that it belongs in the NP class decision problems while it does not belong the class P of decision problems. So the counter example is abstract but existing.

6. It seems a "simple" proof because it chooses the right context to make the arguments and constructions and the key-abstraction mentioned above. So it helps that the scientific community will accept that this 3rd Clay Millennium problem has already been solved.

In the paragraph 4, we give an advanced, full proof that $P \neq NP$, in the standard context of deterministic Turing machines, solving thus the 3rd Clay Millennium problem. And we prove also that $NP = EXPTIME$.

2. Preliminary concepts, and the formulation of the 3rd Clay millennium problem, P versus NP.

In this paragraph , for the sake of the reader, we will just mention the basics to understand the formulation of the 3rd Clay Millennium problem. The standard formulation is found in [3] (Cook, Stephen (April 2000), *The P versus NP Problem (PDF)*, Clay Mathematics Institute site), together with an appendix where there is concise definition of what are the Determinstic Turing machines, that it is considered that they formulate, in Computational Complexity theory, the notion and ontology of the software computer programs.

In the same paper are also defined the computational complexity classes P, NP.

The elements of the classes P, NP etc strictly speaking are not only sets of words denoted by L, that is not only languages, but also for each such set of words or language L at least one DTM, M that decides it, in the specified complexity so they are pairs (L, M). Two such pairs (L_1, M_1), (L_2, M_2) are called equidecidable if $L_1 = L_2$ although it may happen that $M_1 \neq M_2$. E.g. if the complexity of M_1 is polynomial-time while that of M_2 exponential-time

choosing the first pair instead of the second means that we have turned a high complexity problem to a low complexity problem.

The definition of other computational complexity classes like $EXPTIME$ etc can be found in standard books like [4],[8],[9]. In the standard formulation [3] there is also the definition of the concept of a decision problem language in polynomial time reducible to another decision problem language.

Based on this definition it is defined that an $EXPTIME$-complete decision language of $EXPTIME$ is $EXPTIME$-complete, when all other decision problems languages of $EXPTIME$ have a polynomial time reduction to it. Here is the exact definition

Definition 1. Suppose that L_i is a language over all words Σ_i, $i = 1, 2$. Then $L_1 \leq_p L_2$ (L_1 is p-reducible to L_2) iff there is a polynomial-time computable function $f : \Sigma_1 \to \Sigma_2$ such that $x \in L_1$ if and only if $f(x) \in L_2$, for all $x \in \Sigma_1$.

In the same books [4],[8],[9] can be found the concepts and definitions of NP-complete and $EXPTIME$-compete decision problems. See also [5], [9] where its proved that specific decision problems are $EXPTIME$-complete.

For simplicity we will consider here only binary alphabets $\{0, 1\}$ and binary sets of words Σ.

3. Well known results that will be used.

We will not use too many results of the computational complexity theory for our proof that $P \neq NP$.

A very deep theorem in the Computational Complexity is the **Time Hierarchy Theorem** (see e.g. [4],[8],[9],[7],[10]). This theorem gives the existence of decision problems that cannot be decided by any other deterministic Turing machine in less complexity than a specified.

Based on this theorem, it is proved that:

Proposition 1. *There is at least one $EXPTIME$-complete decision problem, that cannot be decided in polynomial time, thus $P \neq EXPTIME$.*

The next two propositions indicate what is necessary to prove in order to give the solution of the P versus NP problem.

Proposition 2. *If the class NP contains a language L which cannot be decided with a polynomial time algorithm, then $P \neq NP$.*

Proposition 3. *If the class NP contains a language L which is $EXPTIME$-complete, then $NP = EXPTIME$.*

4. The solution: $P \neq NP = EXPTIME$ in the context of deterministic Turing machines.

We will prove in this paragraph that $P \neq NP = EXPTIME$ in the context of second order formal language of the Zermelo-Frankel set theory.

Since we are obliged to take strictly the standard formulation of the problem, rather than text books about it, we make the next clarifications.

We will use the next conditions for a Language to be in the class NP, as stated in the standard formulation of the P versus NP problem (see [3] Cook, Stephen (April 2000), *The P versus NP Problem (PDF)*, Clay Mathematics Institute). We denote by $\Sigma^\star$ all the words of an alphabet Σ.

Definition 2. A language L of binary words is in the class NP if and only if the next conditions hold

1. There is a deterministic Turing machine M that decides L. In other words for any word x in L, when x is given as input to M, then M accepts it and if x does not belong to L then M rejects it.

   In symbols: $\exists$ a deterministic Turing machine M, such that $\forall x \in \Sigma^\star$, x is either accepted or rejected by M and if M accepts $x \to x \in L$, and if M reject $x \to x \notin L$.

2. There is a polynomial-time checkable relation $R(x, y)$, and a natural number k of $\mathbb{N}$, so that for every word x, x belongs to L if and only if there is a word y, with $|y| \leq |x|^k$, and $R(x, y)$ holds.

   In symbols: $\exists$ relation R which is polynomial-time checkable, and $\exists k \in \mathbb{N}$, such that $\forall x \in \Sigma^\star$,

$$x \in L \leftrightarrow (\exists y \in \Sigma^\star,\ |y| \leq |x|^k \text{ and } R(x, y) \text{ holds}).$$

Remark 1. In the standard statement of the P versus NP problem (see [3] Cook, Stephen (April 2000), *The P versus NP Problem (PDF)*, Clay Mathematics Institute) the condition 1. is not mentioned. But anyone that has studied complexity theory, knows that it is required. The condition 2. alone cannot guarantee that there is a deterministic Turing machine that decides the

language, as the polynomial checkable relation works only if we provide it with certificate y, and not with only x as input.

Indeed we shall see below at the end of the proposition in Remark 4, that there is even an undecidable language L, for which nevertheless there is a polynomial checkable relation R, so that condition R is satisfied. The languages of NP cannot be semidecidable (or undecidable). The NP class is also defined as $NP = \cup_{k\in\mathbb{N}} NTIME(n^k)$, but this definition is also in the context of non-deterministic Turing Machines. The situation with P, is more clear, because the mere requirement that a language of P is of polynomial time complexity as it standard to define it, involves already that there exist a deterministic Turing machines that for every input word, it halts within polynomial time steps and either accepts or rejects it, therefore it decides it and not that is simply the language of a deterministic Turing machine, and therefore maybe only semi-decidable.

Remark 2. Notice that in the condition 2. the k depends on the relation R and is not changing as the certificate y changes. In other words k does not depend on y and we did not state the next:

There is a polynomial-time checkable relation $R(x, y)$, so that for every word x, x belongs to L if and only if there is a word y, and k in $\mathbb{N}$, with $|y| \leq |x|^k$, and $R(x, y)$ holds. In symbols: $\exists$ relation R which is polynomial-time checkable, such that $\forall x \in \Sigma^\star$,

$$x \in \leftrightarrow (\exists y \in \Sigma^\star \text{ and } \exists k \in \mathbb{N} \text{ such that } |y| \leq |x|^k \text{ and } R(x, y) \text{ holds}).$$

In the standard statement of the P versus NP problem (see [3] Cook, Stephen (April 2000), *The P versus NP Problem (PDF)*, Clay Mathematics Institute) this is not made clear, in the natural language that the definition is stated. But that k does not depend on the certificate, but on the polynomial checkable relation becomes clear, when we look at the proof in any good textbook about complexity theory, of how a non-deterministic Turing machine which runs in polynomial time, can define a deterministic Turing machine with a polynomial time checkable relation, which is considered that replaces it.

Remark 3 (My main intuition to find a proof that $P \neq NP = EXPTIME$. The password setting)**.** *Let us make the next thought-experiment: Imagine a human Mr H who has available infinite time, and has infinite mental capabilities. Now the world asks Mr H to set passwords on all lenths of words! So Mr H sets a password $p(l)$ for words of length $l = 1, 2, 3, \ldots, n, \ldots$ etc. Next let us imagine the problem of finding the password of length say $l = 153$. Mr H has*

an arbitrary free will and he is honest not to give his passwords, in addition Mr H has provided us with a device $D(l)$ for each length l, that unlocks if we give to it the password $p(l)$, so we will know if w is the password or not. So the only way to discover if a partcular word w of length $|w| = l = 153$ is the password $p(153)$ or not, it is to search all the words of length l in an exhaustive way and try them on the device $D(l)$. This is of course an $EXPTIME$ complexity problem, that cannot be reduced to a polynomial time problem. Therefore, finding the language L_P of passwords $p(l)$ of Mr H, cannot be a problem of polynomial time complexity. If in addition, we assume that the blind exaustive search of all words of length l, is an $EXPTIME$-complete complexity problem on the initial data l, then finding the language of passwords of Mr H is also an $EXPTIME$-complete problem. Nevertheless for each word w of length l, the Device(l) is the polynomial time on the length l, checkable relation (certificate for each word w), that can decide if w is a password or not, therefore the language L_P is in the NP class complexity. But the above then after Propositions 2 and 3, indicate that $P \neq NP = EXPTIME$.

Now this intutive idea, is obviously not a formal proof at all, as we are taking about "human Mr H", "arbitratry human free will" etc. Besides we are talking about the complexity of problems here rather than the complexity of languages. How can we turn this intuition to strict and formal proof, without using oracles, or non-formal arguments? The solution is the key-abstarction that I mentioned in the Introduction, that is to start with the existence of an $EXPTIME$-complete complexity language, that we know it exists, without specifying which one. Then define other languages over it and make simple arguments that solve to P versus NP problem.

The strategy to do so is quite simple: We will start with an $EXPTIME$-complete decision problem and its language L_{exp} and we will derive from it an NP class decision problem than cannot be solved in the polynomial time (it does not belong to the class P).

The next proposition sets the existence of an $EXPTIME$-complete complexity language of the $EXPTIME$ complexity class (Proposition 1) in a conventient form, that can be used for further compositions of other languages over it.

Proposition 4. *There is at least one infinite binary sequence, that can be computed and decided in $EXPTIME$-complete complexity.*

Proof. Let an $EXPTIME$-complete decision problem A, that its existence is guaranteed by Proposition 1, and its language $L_{exp} \in EXPTIME$. We will need for the sake of symbolic convenience this language and decision problem,

in the form of a binary sequence. If $\Sigma^\star$ is the set of all words of the binary alphabet Σ of the language L_{exp}, then we give a linear order to the binary alphabet $\Sigma = \{0, 1\}$, $0 < 1$, and then the inherited linear lexicographic order to the set of words $\Sigma^\star$. Since $\Sigma^\star$ is linearly and well ordered with a first element and after excluding all words with a left sequence of consecutive zeros (which is obviously a polynomial time decision on the length of the words) reducing it to the set denoted by $\Sigma^{\star\star}$, we fix the identity map as an arithmetization with an $1-1$ and onto correspondence $F : \Sigma^{\star\star} \to \mathbb{N}$ to the set of natural numbers in binary representation, so that the language L_{exp} can be considered after this fixed arithmetization identity mapping correspondence F, as a subset of the natural numbers. So let $Char(L_{exp}) : \mathbb{N} \to \{0, 1\}$ be the characteristic function of the set L_{exp} in the Natural numbers encoded thus in a binary base. Then $Char(L_{exp})$ consists of d_i, for $i \in \mathbb{N}$, and d_i is binary digit, that is equal to 0 or 1. A first finite 7-digits segment of it, would seem for example like $(0010110\ldots)$. Since L_{exp} is an $EXPTIME$-complete decision problem $L_{exp} \in EXPTIME$, the characteristic function of $\Sigma^{\star\star}$ is computable with an $EXPTIME$-complexity too on the length of the binary words, and conversely any Turing machine computation of this characteristic function of $\Sigma^{\star\star}$ and also infinite binary sequence $Char(\Sigma^{\star\star}) : \mathbb{N} \to \{0, 1\}$: consisting from d_i for all $i \in \mathbb{N}$, and d_i is binary digit, that is equal to 0 or 1, is also essentially a Turing machine decision computation of the language L_{exp}. Therefore there is no polynomial time complexity computation of this infinite binary sequence, as this would make $EXPTIME = P$ and we know that $P \neq EXPTIME$. For the sake of intuitive understanding of the following arguments we call this binary sequence "An $EXPTIME$-compete binary DNA sequence" and we denote it by DNA_{exp}. This simplification from the original $EXPTIME$-complete decision problem and Language L_{exp} of $\Sigma^\star$ to the DNA_{exp} of $\mathbb{N}$ can be considered also as a polynomial time reduction of decision problem and languages $L_{exp} \leq_p DNA_{exp}$ (L_{exp} is p-reducible to DNA_{exp}) (see Definition 1). □

Proposition 5 (3^{rd} Clay millennium Problem)**.** *There is at least one decision problem language of the class NP which is not also in the class P. Therefore $P \neq NP$.*

1^{st} Proof. In the next we show that there is a language L_{np} belonging to the class NP that cannot also belong to the class P without making, the previous binary sequence in the proof of the Proposition 4 called "$EXPTIME$-complete binary DNA sequence" and denoted by DNA_{exp}, computable in polynomial time complexity! To ensure that a language L_{np} belongs to the class NP it must hold that there is a polynomial-time checkable relation $R(x, y)$ and a

natural number k, so that for every word x, it holds that x belongs to the language L_{np} if and only if there is another word y, called "certificate" with length $|y| \leq |x|^k$, so that $R(x, y)$ holds. Here by $|x|$ we denote the length of the word x, which is a natural number.

Now comes the intuition behind calling the binary sequence DNA_{exp} of the previous proof, a DNA sequence: The trick here is to define this language denoted by L_{np} with the information encoded in the binary sequence DNA_{exp} so that, although a human with deterministic Turing machines and $EXPTIME$-time complexity can compute DNA_{exp} and therefore decide L_{np}, no deterministic Turing machine within polynomial-time complexity can compute and decide the L_{np}. In addition for every word x, if a human will give to such deterministic machines the necessary information in the form of a "certificate" y, then a deterministic Turing machine can decide if x belongs or not to L_{np} within polynomial-time complexity.

We define such a language L_{np} with the previous requirements simply as follows:
For any word $x \in \Sigma^\star$, $x \in L_{np}$ if and only if, the $d_{|x|} = 1$, where $d_{|x|}$ is the $|x|$-order binary digit of the infinite binary sequence DNA_{exp}, and $d_{|x|}$ is equal to 1. And of course x does not belong to L_{np} if and only if the $d_{|x|} = 0$, where $d_{|x|}$ is the $|x|$-order binary digit of the infinite binary sequence DNA_{exp}.

Then we define as "certificate" y of the word x, the finite sequence $y = (d_1, d_2, \ldots, d_{|x|})$, and as polynomial time checkable relation $R(x, y)$, and that $R(x, y)$ holds, the fact that given x, and y, the last digit of y is 1. Notice that here a human gives a lot of information to a Turing machine that will check if x belongs or not to L_{np}, in the form of the $|x|$-length initial segment y of the infinite binary sequence DNA_{exp} that we know that no Turing machine can compute within polynomial-time complexity.

That this relation $R(x, y)$ is checkable in polynomial time relative to the length $|x|$ of x, is obvious as the Turing machine with input x and y, will have only go through $|x|$-many steps to check the last digit of y.

Now no deterministic Turing machine M can decide the language L_{np}, in other words decide given as input only the word x (without its "certificate" y), if $x \in L_{np}$ or not. And this is so, because if it exist such a deterministic Turing machine M, then it could also decide (or compute) the digit $d_{|x|}$ of DNA_{exp} which we know that is not computable in polynomial-time complexity. Thus L_{np} does not belong to P, and therefore $P \neq NP$. □

2nd Proof. We may define, in a simpler way, the language $L_{passwords}$ (the index passwords, is so as to follow the intuition of password setting as in the Remark

3 of §4) as the set of all binary words, that are the successive n-initial segments of the infinite binary sequence DNA_{exp}. Then this language is obviously (after Proposition 4) an $EXPTIME$-complete language. Nevertheless the language $L_{passwords}$ also belongs to the class NP, because for each word w, of length $|w| = n$, a "certificate" y of it is the word w itself $y = w$, and the polynomial time checkable relation $R(w, y)$, $y = w$, is checkable in polynomial time, relative to the length $|w| = n$. Notice that we have here one only word w for each word-length n. But then from the Proposition 3 $NP = EXPTIME$, and thus $P \neq NP$. □

Corollary 3. *It holds that* $NP = EXPTIME$.

Proof. Direct from the Proposition 3, and that the language $L_{passwords}$ in the 2nd proof of the Proposition 5 is also $EXPTIME$-complete language, besides belonging in the class NP. □

Remark 4. Notice that instead of taking, the characteristic function DNA_{exp} of an $EXPTIME$-complete language, we could have taken the characteristic function DNA_{und} of an undecidable language and we know that, there is at least one, and repeat the definition of the Language L_{np}, deriving thus an undecidable language, which still it has a polynomial time checkable relation, that nevertheless works only if a human feeds it with a certificate y and there is not a Turing machine that can decide it by taking as input the word x alone. This confirms that in the definition of NP in, Definition 2, the condition 1) is required. Alternatively we may prove the same thing in a different way. By using the axiom of choice of the ZFC set theory we may define for example an arbitrary infinite sequence L_p of passwords p_n, each one of length exactly n, from the infinite set of the sets of words Σ_n of length n. It is known that the axiom of choice of the ZFC set theory, gives no information at all about what are the elements of such a set, besides that each p_n belongs to Σ_n. We cannot expect that any such infinite choice L_p of n-length passwords p_n can be decided by a deterministic Turing machine. If it was so, as such Turing machines are countable, we order all such languages $L_{p,i}$, $i \in \mathbb{N}$ in a sequence and with the diagonal method we define a new and different such language L_0 of passwords, differing to at least one password from all those $L_{p,i}$, thus this L_0 is undecidable. Still again there is a polynomial-time checkable relation $R(x, p_{|x|})$, which simply is checking if $x = p_{|x|}$, so that for every word x, there is a "certificate", here the password $p_{|x|}$, and x belongs to the language L_0 of passwords iff $R(x, p_{|x|})$ holds.

5. Conclusions

Sometimes great problems have relatively short and elegant solutions provided we find the key-abstractions and convenient context, symbols and semantics to solve them. But even relatively simple paths of reasoning, may be difficult to travel, if there is not, at a certain point of them, the necessary "bridge", that is the necessary key-abstarction or the right conceptual "coins" of symbols and semantics to exchange and convert. Here the key-abstraction was to start from the class $EXPTIME$ and an $EXPTIME$-complete language of it, without specifying which one instead starting from the class NP. If the P versus NP problem is researched without a main strategy, that would require a short proof, it might become a very complex problem to solve. **My main hidden guiding idea in searching for such a simple proof, was that what the "arbitrary human-like free-will" of a non-deterministic Turing machine as human-machine interactive software (e.g. in password setting), can do in polynomial time cannot be done by a purely mechanical deterministic Turing machine in polynomial time.** Since in my opinion the Hierarchy Theorem is a deeper result than the P versus NP problem, in principle there should exist a not much more complicated proof of the P versus NP problem, compared to the proof of the Hierarchy Theorem. The proof of the P versus NP problem in the direction $P \neq NP$, is supposed also to mean that the standard practice of encryption in the internet, is safe.

References

[1] Conway J.H., *On numbers and games*, Academic press (1976). http://dx.doi.org/10.1090/psapm/043/1095538

[2] Cook, Stephen A., A hierarchy for nondeterministic time complexity, *Proceedings of the fourth annual ACM symposium on Theory of computing, STOC '72*, Denver, Colorado, USA, ACM., 187-192. http://dx.doi.org/10.1145/800152.804913

[3] Cook, Stephen, The P versus NP Problem (PDF), *Clay Mathematics Institute site*, (2000).

[4] Harry R. Lewis and Christos H. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, New Jersey (1981)

[5] Hartmanis, J., Stearns, R. E. (1 May 1965), On the computational complexity of algorithms, *Transactions of the American Mathematical Society*, American Mathematical Society, **117**, 285-306. http://dx.doi.org/10.1090/S0002-9947-1965-0170805-7

[6] Kyritsis C., On the solution of the 3rd Clay Millennium problem. A short and elegant proof that $P \neq NP$ in the context of deterministic Turing machines and Zermelo-Frankel set theory, *Proceedings of the first ICQSBEI 2017 conference*, Athens, Greece, 170-181.

[7] Luca Trevisan, *Notes on Hierarchy Theorems*, U.C. Berkeley.

[8] John C. Martin, *Introduction to Languages and the Theory of Computation (2nd ed.)* McGraw-Hill, (1997).

[9] Papadimitriou Christos, *Computational Complexity*, Addison-Wesley, (1994).

[10] Stanislav, Žák, A Turing machine time hierarchy, *Theoretical Computer Science*, Elsevier Science B.V., **26**, 3 (1983), 327-333.

[11] Ivanov Viktor V., A short proof that NP is not P, *International Journal of Pure and Applied Mathematics IJPAM* (2014), 81-88. http://dx.doi.org/10.12732/ijpam.v94i1.9

[12] Yannakakis M., Expressing combinatorial optimization problems by linear programs, *Proceedings of STOC 1988*, (1998), 223-228. http://dx.doi.org/10.1145/62212.62232