**Consistency Problem and “Unexpected Hanging Paradox” (An answering to P=NP Problem)**

**Farzad Didehvar**

**didehvar@aut.ac.ir**

**Abstract**

The Theory of Computation in its existed form is based on Church –Turing Thesis. Throughout this paper, we show that the Turing computation model of this theory leads us to a contradiction. In brief, by applying a well-known paradox (Unexpected hanging paradox) we show a contradiction in the Theory when we consider the Turing model as our Computation model.

**Introduction**

In [1] we present a situation based on a paradox in which any axiomatization system to show the situation leads us to contradiction.

We quote this restated “Unexpected hanging paradox” as follows:

‘*’ A Computer Scientist invents a Computer which argues logically, and also any information and conclusion on this Computer is announced to public openly.*

*The People of the city do not like this Computer and they appoint a jury which decides about the destiny of this Computer.*

*The jury decision is declared to Computer Scientist and Computer as follows:*

*The Computer will be destroyed in the next week on a day that wasn’t concluded by the computer itself before that day. Definitely we know everything that the computer knows or everything it will conclude. (The computer is invented in a way that it declares anything it concludes; we can imagine that it reveals all its conclusions in a printing form).*

*After announcing it, the computer starts to argue and it finds the following arguments (the argument of ‘hanging paradox’). By the way, because of the computer's Intelligence, all people know that destroying the computer is equal to “executing the Computer”.*

***Demonstration of a Computer***

*I will not be executed on Saturday, since if I was executed on Saturday, I would be alive till Friday, and on Friday I conclude that I will be executed on Saturday ( since based on what jury said I will be executed in the next week). So in case if I remain alive (!) till Friday, I will conclude on that day, I will not be executed on Saturday! (in view of the proposed claim of the jury).This argument exists in my data base right now (after concluding by me, right now).In a more exact form, the Computer argues as follows: I have a timer, by using my storage and database I conclude that I will conclude on Friday “I will be executed on the next day, so based on the claims of juries, I will not be executed on Saturday”.*

*2. I will not be executed on Friday, since if I was executed on Friday I wouldremain alive till Thursday, and on Thursday and our claim in point 1 and based on what jury said(“The Computer will be executed in the next week”) I conclude that I will be executed on Friday. So by the proposed claim of the jury I will not be executed on Friday!(This argument right now exists in my data base after concluding by me, so in case if I remain alive(!) till Thursday, I will conclude on that day, that I will not be executed on Friday). In a more exact form, the Computer argues as follows: I have a timer, by using my storage and database I conclude that I will conclude “on Thursday that I will be executed on the next day, so by the proposed claim of juries, I will not be executed on Friday”.*

*…*

*6. I will not be executed on Monday, since if I will be executed on Monday I will be alive till Sunday,*

*And on Sunday and the claims made in point numbers 1&2&…&5, I conclude that I will be executed on Monday, since by what jury has said I will be executed in the next week. So, in case if I remain alive (!) till Sunday, I will conclude on such a day that I will not be executed on Monday! (Based on the proposed claims of jury), this argument exists right now in my data base after concluding by me. In a more exact form, the Computer argues as follows: I have a timer, based on my storage and database I conclude that I will conclude on Sunday “I will be executed on the next day, so by the claim of jury, I will not be executed on Monday!”*

*7. I will not be executed on Sunday, since based on points 1-6 I will not be executed on Saturday, Friday …Monday.*

*So I should be executed on Sunday! My conclusion (to be executed on Sunday) concludes that I will not be executed on Sunday!*

*So, I will not be destroyed in the next week.*

*The computer didn’t conclude more, except it's every day conclusion; it concluded that it will not be executed on the next day based on the similar above argument.*

*The computer was destroyed on Tuesday (it was executed), and the Computer Scientist complained about this injustice, and he transferred this argument of Computer to journals and the court. In this message the computer scientist mentions that the computer presented a logical argument, based on that he will not be executed in the next week, nevertheless he was executed).*

*The court said:*

*The Computer proved that he will not be executed. This is a true proof. We executed him on Tuesday, and as he claimed he didn’t conclude that he will be executed on Tuesday. On the contrary, he announced that “he will not be executed on that day”. In other words, he proved in his message that by accepting what the judge said as a true claim, he would not be executed in that week. More formally, here we state as follows:*

*P: we consider what the judge said as a true claim*

*Q: He would not be executed in this week*

*“His proof (p|---q) is a true proof, but it doesn’t show the Truth”. (\*)*

*In this paper, we defend the above claim of the juries (\*) and we try to show how we could develop this idea.* “

We show that the above paradox and phrases demonstrate a contradiction in “Theory of Computation". More exactly by considering the above claims as true words, we are able to design a Turing machine demonstrates the above situation of paradox.

**The contradiction and introducing the Model**

Firstly we should define a special type of Turing machine. This Turing machine is able to smash itself (From now on instead of “executing” and “smashing” we use "cracking”).

To do this, we add symbols of alphabet “s” and “\*” to $∑$ to have $∑\_{1}$.The machine is fed by a listing of a c.e set A, when it reaches the symbol “s’’ it halts and it never works.

Here we change slightly the above description for some technical purposes and before “it halts and it never works” it emits “\*” as the final output (\* is the last word).

**Definition:** Let M be a Turing Machine and A$⊆∑\_{1}^{\*}$a is a c.e set. Suppose $L\_{A}$a list of A. (M,$L\_{A}$) is a Turing machine that is fed by strings in $L\_{A}$by the same order in $L\_{A}$and when it reaches the symbol “s” first it emits a symbol“\*”,then it halts and does not work forever.

**Theorem 1:** For any Turing machine M and c.e. set A, (M,$L\_{A}$) is equivalent to a Turing machine.

**Proof:** First we consider the following cases:

1. The strings of A contain no “s’’ it is easy to see that the range and domain of this machine is c.e and the machine is equivalent to a Turing machine.

2. At least one of the strings in A contains s, so the range and domain of the machine would be finite. Hence it will be equivalent to a Turing machine.$∎$

Now we are able to design a (M,$L\_{A}$) machine to demonstrate the situation in the paradox. In designing, the computer accepts seven number of inputs as the number of days of week.(1,0,0,0,0,0,0) for Sunday, and respectively we have

(0,1,0,0,0,0,0),…,(0,0,0,0,0,0,1).

Here, our inputs are (0,0,0,0,0,0,0) represent no day in the week the computer would be cracked on it.

Here the computer has a proof that it will not be cracked and this proof is in its storage. It is easy to see that, this machine is a (M,$L\_{A}$) machine for a finite set A, and hence it is equivalent to a Turing machine by Theorem1.

So, any input from the above set of inputs is acceptable either as the day of cracking (executing day) or the computer is not cracked. This claim adapts to the jury claim in restated paradox.

But by demonstration and proof of computer simply (0,0,0,0,0,0,0) is considered as acceptable.

So 7=1, consequently $∀n 2^{n}$=1.We have two immediate results:

1. The Theory of Computation is self –contradictory,
2. P=PSPACE =NP.

The second claim seems interesting and exciting, but it is based on a contradiction in the system.

We should mention that, here the proof contradicts the computation.

So, we have a contradiction in the “Theory of computation” when we apply Turing model as our model of computation.

**Conclusion:**

Here, the Theory of Computation leads us to a contradiction (More exactly by above description, we are able to put a contradiction in it).It is a proof that we should concentrate more on some subjects and models like Quantum Computation and Randomized algorithm. In brief, it might be better to assume the models whose bits and states are not considered definitely separated. In these cases, the classes of complexity and problems are drastically changed, and the problems there, are not parallel to P=NP problem;so problems are totally in a different type of form. In practice, any Algorithm and Computation is either a Randomized algorithm or Quantum Computation, based on existed acceptable theory of Physics besides the possibility of errors in Computation. The other possible solution is considering non-classical logics, like Para consistent Logic.

Anyhow, we are not able to reply this Algorithmic problem(P=NP) by proof approach, since Theory of Computation is not a consistent Theory in classical sense.

*References:*

*1. FarzadDidehvar, A Semantic Situation without Syntax (Non- axiomatizibility of some Theories), represented in Unilog2015).”It could be finding in arxive.org”*

*2. FarzadDidehvar,when proof doesn’t show the Truth, (Accepted in “The World Philosophy Day”, 2010).*