Is Classical Mathematics Appropriate for Theory of Computation?

Farzad Didehvar

Amir Kabir University (Tehran Polytechnic), Department of Mathematics &
Computer Science Tehran 1591634311, Iran

Abstract. Throughout this paper, we are trying to show how and why our Mathematical frame-work seems inappropriate to solve problems in Theory of Computation. More exactly, the concept of turning back in time in paradoxes causes inconsistency in modeling of the concept of Time in some semantic situations. As we see in the first chapter, by introducing a version of "Unexpected Hanging Paradox", first we attempt to open a new explanation for some paradoxes.

In the second step, by applying this paradox, it is demonstrated that any formalized system for the Theory of Computation based on Classical Logic and Turing Model of Computation leads us to a contradiction. We conclude that our mathematical frame work is inappropriate for Theory of Computation.

Furthermore, the result provides us a reason that many problems in Complexity Theory resist to be solved.

Keywords: Unexpected Hanging Paradox, Liar Paradox, Turing machine, P=NP, P=PSAPCE

1 Introduction

The surprise exam paradox (Unexpected Hanging Paradox) is a well-known paradox that has a sufficiently large literature in Philosophy and Mathematics. Nowadays, this paradox is known as "Surprise test paradox", as well.

A simple formulation of this paradox can be observed in some of the references that we have introduced, namely references [14], [19].

Historically, this paradox was exposed in academic circles by O'Conner's article [14] for the first time; however, it seems to have been mentioned before 1940 anonymously.

Some people think that basically, it is not a paradox. Quine's work was in this line [15]. Others believe that the paradox has been resolved by Quine and only some of its aspects remain to be discussed [17].

In fact, there is no consensus on this issue, so we have no final resolution yet.

Historically, there are different approaches to solve this problem:

1 Logical approach

Some researchers put more stress on its logical aspects and try to tackle this problem with a logical approach [4], [16], [10]. For instance, they assume that this problem is self-contradictory and self-referencing and try to apply this assumption in order to solve it.

Parallel to this approach, this paradox is applied to give a solution to the First and also the Second Gödel Theorem. In [12], inspired by surprise examination argument and by the application of some other Theorems, the authors give a new proof for the second completeness theorem. Before this work, Chaitin has done a similar work by applying the Berry paradox [2]. It is considered as an attempt to show the self-referential features of this paradox. There are similar earlier articles [5].

The explanations of [7], [11] are in this line.

2 Epistemic approach

Some researchers think about this paradox as a paradox of knowledge (Epistemic Paradox). Some approaches try to know the ability or inability of some statements [3]. In recent years, we have had some attempts to solve this problem in dynamic epistemic logic: [6], [19].

There are attempts to solve this problem once and for ever. While numerous solutions have been proposed for this paradox during the last six decades, it continuously surprises us. A number of articles have made endeavors to put an end to it once and forever [13].

In [8], [18], we have found the relationship among this paradox and the other types of paradoxes. In [9] a game theory approach is presented to solve the paradox.

Solving this paradox is not the main aim of this article. First, in section 1, we introduce and apply a new version of this paradox to show that there is a semantic situation which no formal consistent system in the classical sense supports, and this is one of our major claims. Therefore, we know that this version of paradox is a logical paradox and, consequently, the first approach is appropriate for it. Then, a definite solution for that is given, which is also a possible solution for the "Unexpected Hanging Paradox". Nevertheless, the article is silent about a definite solution to the original form of paradox. In section 1, by applying a version of this paradox, we try to modify and formalize this problem and show the above claim.

In section 2, applying the aforementioned assertions shows us a general claim about a contradiction in the "Theory of Computation" or the deficiency of this theory in modeling some situations. It demonstrates the reason that some problems remain un-answered in the Theory of Computation.

2 A Modified Version of Unexpected Hanging Paradox and the Logical Consequence of It

In this section, first a modified version of the paradox is presented; then we conclude that there is a proof and a semantic situation associated with this proof, but the proof doesn't show the truth. Later, we make the proof more precise in a formal form.

We express this restated "Unexpected Hanging Paradox" as follows:

'' A Computer Scientist invents a Computer which argues logically; in addition, all the information and conclusions on this Computer are announced to the public openly. From now on, in order to avoid ambiguity we call this computer, Computer1.

The inhabitants of the city do not like this computer, and they appoint a jury which decides about the destiny of Computer1.

The jury's decision is declared to the Computer Scientist and the 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. No doubt, we know everything that the computer knows or everything it will conclude. (The computer is invented in a way that it declares everything it concludes and every new piece of information; 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'). It should be noted that because of the computer's intelligence, everyone knows that destroying the computer is equal to "executing the computer".

Demonstration of the computer:

I will not be executed on Saturday, since if I were 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 the jury said, I will be executed in the next week). Thus, if I remain alive until Friday, I will conclude on that day that I will not be executed on Saturday (in view of the proposed claim of the jury).This argument exists in my data base at the moment (by my forthcoming conclusion, 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 jury, I will not be executed on Saturday".

2. I will not be executed on Friday, since if I were executed on Friday, I would remain alive till Thursday, and on Thursday and our claim in point 1 and based on what the jury said ("The computer will be executed in the next week"), I conclude that I will be executed on Friday. Therefore, by the proposed claim of the jury, I will not be executed on Friday (this argument exists in my database right now by my forthcoming conclusion. So in case I remain alive until 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 the jury, I will not be executed on Friday".

...

6. I will not be executed on Monday, since if I am executed on Monday, I will be alive till Sunday, and on Sunday based on the claims made in sections 1&2&...&5, I conclude that I will be executed on Monday, since by what the jury has said, I will be executed in the next week. So, in case I remain alive until Sunday, I will conclude on that day that I will not be executed on Monday (based on the proposed claims of jury). This argument exists in my database right now by my forthcoming conclusion. 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 that "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. However, 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 than its everyday conclusions; it concluded that it will not be executed on the next day based on arguments similar to the ones mentioned above.

The computer was crashed on Tuesday and the Computer Scientist complained about this injustice, and he reported the argument of the computer to journals and the court. In this message the computer scientist mentions that the computer presented a logical argument, proving that he will not be destroyed in the next week. Nevertheless, he was crashed.

The court said:

The computer proved that it will not be destroyed. This is a true proof. We crashed him on Tuesday, and as he claimed he didn't conclude that he will be destroyed on Tuesday. On the contrary, he announced that "he will not be executed (destroyed) 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, it may be stated as follows:

P: The judge's sentence is considered a true claim

Q: He would not be destroyed in this week

"His proof ($p \vdash q$) is a true proof, but it doesn't show the Truth". (*)

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

It is noteworthy to mention that we don't claim the above result proposes a solution to "Unexpected Hanging Paradox" in any of its versions. It is simply considered as the only way to explain this version of "Unexpected hanging paradox". Later on, we will know the above result as a possible explanation for the other versions of this paradox and also some other paradoxes like the liar paradox. In [1] we can find a more conclusive explanation of the liar paradox.

To formalize the above proof, we call our computer "A" and whenever we say "A concludes", in our paradox, we replace it by A:$[\varphi]$. Also, A:$_i[\varphi]$ stands for "A concludes or utters φ in the ith day".

In the following formalism, $\varphi(i)$ stands for "A will be executed in the ith day". We slightly change the scenario. We suppose the ceremony of smashing the computer take places from 11/00 -12/00; this gives us a better understanding for the second principle. By $\varphi(i)$, we mean the computer is being smashed in the ith day.

A: $[\psi]$ means there exists $1 \le i \le 7$ in which A:$_i[\psi]$.

Now in any formalization of the problem and proof, we will have the following assertions:

1. $\bigvee_{i=1}^{i=7} \varphi(i)$
2. $\sim\varphi(n) \rightarrow [A: [\sim\varphi(n)]]$ (In the evening of each day, A understands he is not destroyed and he declares it).
3. $\varphi(k) \rightarrow \bigwedge_{i=1, i\neq k}^{7} \sim\varphi(i)$ (If he is destroyed in a day, we conclude that in the other days he wasn't destroyed).
4. $(\bigwedge_{i=1}^{k-1} A: [\sim\varphi(i)]) \wedge (\bigwedge_{i=k+1}^{7} A: [\sim\varphi(i)]) \rightarrow A:_{k-1} [\varphi(k)]$
5. $A:_k [\varphi(k+1)] \rightarrow \sim\varphi(k+1)$ (If A utters in the ith day that he will be executed in the i+1th day, he will not be executed in i+1th day, i=0,1,...,6,7).

$\sim\varphi(7) \equiv \top$

Suppose $\varphi(7) \equiv \top$

$\bigwedge_{i=1}^{6} \sim\varphi(i)$ (Principle 3)

$\bigwedge_{i=1}^{6} A: [\sim\varphi(i)]$ (Principle 2)

$A:_6 [\varphi(7)]$ (Principle 4)

$\sim\varphi(7) \equiv \top$ (Principle 5)

$\sim\varphi(6) \equiv \top$

Suppose $\varphi(6) \equiv \top$

$\bigwedge_{i=1}^{5} \sim\varphi(i) \wedge \varphi(6) \wedge \sim(7)$ (Principal 3)

$\Lambda_{i=1}^{5}$ A: $[\sim\varphi(i)] \wedge A$: $[\varphi(6)] \wedge A$: $[\sim\varphi(7)]$ (Principal 2)

$A :_5 [\varphi(6)]$ (Principle 4)

$\sim\varphi(6) \equiv \top$ (Principle 5)

3. $\sim\varphi(5) \equiv \top$. Similar to the above proof.

……

……

7. $\sim\varphi(1)$. In a similar way.

So we have a contradiction here, since $\varphi(3)$. ■

So, we have a model whose associated formal system is contradictory. It is notable that any syntactical system that we attribute to the above paradox contains the above statements, whether directly or as a conclusion. In other words, the proof shows that this formalism and any other formalism are essentially contradictory, but we have a semantic situation for this contradictory formalism.

As we see, considering turning back in time which applies in the paradox, causes inconsistency in modeling of the concept of Time in some semantic situations.

2.1 An Explanation

In the above system, there is a contradiction. In brief, the judges claim that the computer will be destroyed the next week, but the computer proves that he will not be destroyed. Thus, this system is an inconsistent system, but at the same time we have a semantic situation for this system. In other words, we have a semantic situation which no consistent syntactical system supports. As a result, the above proof does not show any truth, so there are some incorrigible flaws in the modeling and "formalizing the proof". In other words, formal systems are not able to support such semantic situations. Clearly, this opens a new way to explain some paradoxes similar to the liar paradox or the "Unexpected Hanging Paradox" in its usual form, as stated below:

In such paradoxes the proofs don't show the truth, since there is no consistent syntactical system to support the related semantic situation. Although this explanation seems odd and not essential for many paradoxes, as we explained for our version of the "Unexpected Hanging Paradox", this is the unique and essential way to explain the contradiction when we consider classical logic as back ground. Therefore, we may conclude that this explanation is a plausible and possible way in the other cases, as well.

This is the first central result and thesis of this paper.

3 The Conclusions in the Theory of Computation

In this chapter, by employing the results in the previous chapters we present some conclusions about the consistency of the "Theory of Computation". 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 "destroying", we use "cracking").

To do this, we add letters of the alphabet "s" and "*" to $\sum$ to have $\sum_1$. The machine is fed by a listing of a c.e set A. When it reaches the symbol "s'', first it emits "*" as the final output, then it halts and never works (it is cracked).

Definition 1. *Let M be a Turing Machine and $A \subseteq \sum_1{}^*$ is a c.e set. Suppose that L_A, a list of A. (M,L_A), is a 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 anymore.*

Theorem 1. *For any Turing machine, M, and c.e. set $A \subseteq \sum_1{}^*$, there is set $B \subseteq \sum{}^*$ such that (M,L_A) is equivalent to (M,L_B).*

Proof (of Theorem) We consider the following cases:

1. The strings of A contain no "s". Since it is easy to see that the range and domain of this machine is c.e, it is sufficient to consider B=A and $L_A = L_B$.

2. At least one of the strings in A contains "s", so the range and domain of the machine would be finite. It is sufficient to consider B as that finite subset of A which is the domain of (M,L_A) and the listing which is derived from the listing of L_A.■

Thus, this new machine has no power more than the Turing machine, and it doesn't violate Church Thesis. Now we design a (M,L_A) machine to demonstrate the situation in the paradox.

Here, our possible input is:

(0,0,0,0,0,0,0) represents no day in the week on which the computer would be cracked.

(1) represents it will be cracked on Sunday,

(0,1) represents it will be cracked on Monday, (0,0,1) represents it will be cracked on Tuesday, (0,0,0,1) represents it will be cracked on Wednesday,

….,

(0,0,0,0,0,0,1) represents it will be cracked on Saturday.

We have two different types of states:

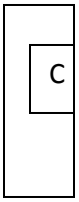
1. First Type:

$Q_0,\ldots,Q_6,Q_F$ are usual states (no emission). They represent days of the week and Q_F is the final state. We call the other states "emission states".

2 Second Type

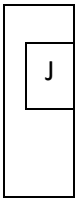
2.1 Q_c When 1 enters this state as input, it emits "s" (the state of cracking).

2.2



represents the computer in the paradox. It emits 0 when 1 enters as input. It emits "*" when "s" enters as input.

2.3



represents the jury in our scenario. It is able to emit 1. (Here, it emits 1 when C (computer) emits 0 and the associated Q_i emits 1 ($1 \leq i \leq 6$), as we see in Fig1).

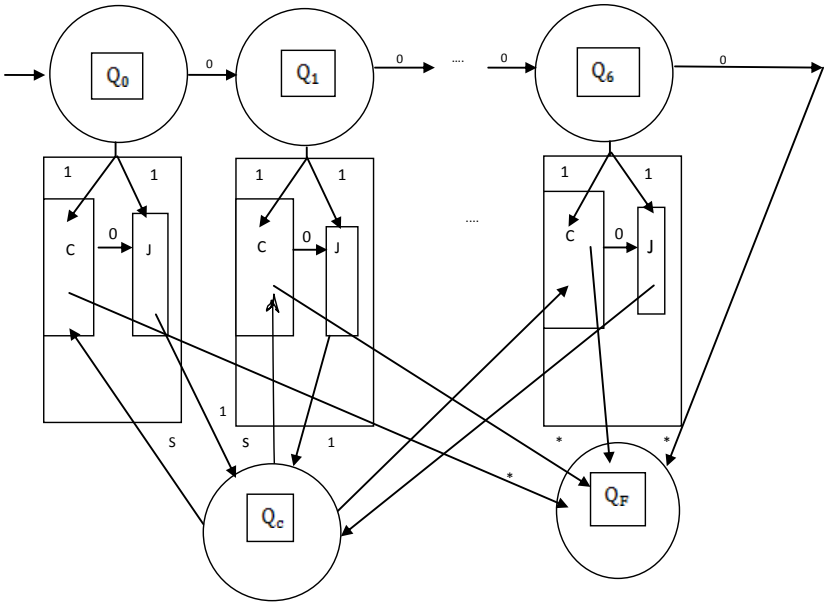


Fig 1

Adopting the scenario of paradox, we have a proof that the computer in the paradox (Computer 1) will not be cracked and this proof is in its storage.

The second Computer (Fig 1) accepts the possible days of execution. We call it Computer2. C represents Computer1 introduced in the paradox. When 1 enters C and J, C emits 0, (it represents "it knows that it will not be cracked"). J emits 1 to Q_c and it emits "s" to C, so it cracks C. (Since Computer1 does not know that it will be cracked. More exactly, the computer claims it will not be cracked and J emits 1, so the machine would be cracked, repeating the argument of the paradox). In the final step, C emits * to go to the final state.

It is easy to see that this machine is a (M, L_A) machine type for a finite set A, and hence it is equivalent to a Turing machine by Theorem1.

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

But by demonstration and proof of computer 1, simply (0,0,0,0,0,0,0) is considered as acceptable input.

So 7=1, consequently $\forall n$ 2^n=1.

By this contradiction, 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 presented in "Unexpected Hanging Paradox" and depicted by Computer2 contradicts the result of the computation.

Remark So, we have a contradiction in the "Theory of computation" when we apply the Turing model as our model of computation in order to model this situation. Here it provides us a reason why so many problems in Complexity Theory is not solved, such as P=NP and P=PSPACE.

4 Conclusion

Most of the proofs and demonstrations in the Theory of Computation (especially in Complexity Theory) employ three elements: first, a situation similar to the above (for instance, discrete objects like graphs and moving from one node to the other and passing edges,…) in which the problem is presented, a model of computation (like Turing machines) as the second element, and Mathematical proofs and reasoning as the third.

In this regard, there is no specific issue or exception in our example in the previous section. Moreover, this should be considered a simple example, since neither infinite Mathematical objects nor large numbers are involved in this problem.

In other word, the situation here is a common situation in proofs and demonstrations in Theory of Computation and simple in philosophical sense. It seems worryingly ad hoc to consider this example as an exception.

In any case, we face a contradiction in the Computational Modeling of this situation. It is notable that the bases of our arguments are Classical Logic and the Turing Model of Computation.

In brief, by applying this paradox, it is demonstrated that *any formalized system for the Theory of Computation based on Classical Logic and Turing Model of Computation leads us to a contradiction*. As a result, it illustrates the reason many problems in the Theory of Computation resist to be solved.

Ultimately, we face the proposed question again: Is classical Mathematics appropriate as a framework for Theory of Computation?

References

1. Barwise. J and Etchmendy. J The Liar, Oxford University Press, (1987)

2. Chaitin, G.J. Computational Complexity and Goedel Incompletness Theorem. ACM SIGACT News 9, (1971), 11-12

3. Chow,T.Y. The Surprise Examination or Unexpected Hanging Paradox, Amer.Math.Monthly, (1998), 10541-51

4. Douven, I . A New Solution to the Paradoxes of Rational Acceptability, British Journal for the Philosophy of Science 53 (3):, (2002), 391-410

5. Fitch F. A. Goedelized Formulation of the Prediction Paradox, Amer.Phil.Quart1, (1964), 161-164

6. Gerbrandy, J. The Surprise Examination in Dynamic Epistemic Logic, synthese 155, (2007), 21-33

7. Holtzman, J.M. An Undecidable Aspect of the Unexpected Hanging Problem, Philosophia 17(2), (1987), 195-198

8. Holtzman, J. M. A Note on Schrodinger's Cat and the Unexpected Hanging Paradox, British Journal for the Philosophy of Science, 39 (3) (1988), 397-401

9. Jose, J.L The Surprise Exam Paradox, Rationality, and Pragmatics: A Simple Game Theoretic Analysis, Journal of Economic Methodology 15(3) (2008), 285-299

10. Kearns, J. An Illocutionary Logical Explanation of the Surprise Execution, History and Philosophy of Logic 20(3-4), (1999), 195-213

11. Kramer, M.H. Another Look at the Problem of the Unexpected Examination, 38 (03), (1999), 491-502

12. Krichmann, S & Raz, R. The Surprise Examination Paradox and the Second Incompleteness Theorem, Notices of the AMS, vol 57, Number 11, (2010), 1454-1458

13. Levy, K. The Solution to the Surprise Exam Paradox, Southern Journal of Philosophy, 47 (2), (2009), 131-158

14. O'Conner, D. Pragmatic Paradoxes, Mind, 57, (1948), 358-359

15. Quine, W. V. On a so Called Paradox, Mind, 62, (1953), 65-66

16. Shapiro, SC. A Procedural Solution to the Unexpected Hanging and Sorties Paradoxes, Mind, 107 (428) (1998), 751-762

17. Smullyan, R. What is the Name of this Book? , (1978)

18. Veber, M. On a so Called Solution to a Paradox, Pacific Philosophical Quarterly, 96(3), (2015)

19. Williams, J.N. The Surprise Exam Paradox: Disentangling Two Reductions, Journal of Philosophical Research, 32, (2007), 67-94