Theoretical Boundaries and Practical Implications of Computability

Arash Dewan
Final Research Paper
Prompt II | PHILOS C128
University of California, Los Angeles

I. Introduction

The discipline of computation is derived from the boundaries of mathematical knowledge. We face the challenge of how far we can take computing, pushing it to its limits; yet even with these foundations there prevails the nature of problems that are dedicated to algorithmic solutions. This theoretical ideology reveals to us insight on what can and cannot be computed, occupying a critical space of science that shows us what the true limitations of these computable systems are.

This is the basis of non-computable sets. By highlighting the mathematical capabilities within any given formal system, we bring a sense of appreciation for the scopes of algorithmic methods. This discussion regarding the emphasized issue of compatibility, in principle, is to broaden the knowledge of how we classify what we are capable of. Non-computable sets are ultimately a desired field trying to understand what is computable. Incorporating these two sides of the computable and uncomputable will only bring us closer to perceiving what mathematicians can do, and what are the areas of algorithmic building that humans are incapable of solving.

Many previous papers have discussed this theory of computation, debating what exactly is the need to study non-computable sets, and also visualizing if there even demands a need to observe it at all. In a sense, it is puzzling to think of what is actually being done when observing non-computable sets; is it merely just trying to find the degree of difficulty of a problem without solving it yourself. In general, there is a functionality to what non-computable sets are. By bringing in contributions of Tarski's Theorem on Truth as well as Gödel's First Incompleteness Theorem, I am able to offer significant findings that these analyses give an enhanced recognition in the context of computation. Alongside past theorems, current day literature by Jenny (2018) proposes a nuanced reason for counterfactuals regarding relative computability, to help determine the computation as a science.

The premise of this paper will be to break down the computable agents of mathematics, and perceive what the potentials of this in the field of computability is. This will provide us with the essence of studying non-computable and reflect a sense of practicality to the theoretical. Grabbing from various scientific and mathematical domains and papers, I am able to develop and share a novel proposal to what it means when we study non-computable sets.

Influenced by past research, my current proposal for non-computable sets goes as follows.

If provided with a problem space without limits or boundaries, we are tasked to solve all the problems.

Given this, it can almost become infinitely long to finish these problems. We must recognize the complexity of what we are trying to solve, and determine if we can decide, or remain undecided. It does not mean that the problem is unresolved, but likewise, if it is uncomputable, that is a label of being solved as such. We use these limits and boundaries with regard to complexity.

Complexity will be the premise of this proposal. It will be what guides the paper in its description of what non-computable sets are. It will highlight what those are doing when studying these types of sets, and it will also be used to carry the development of problems.

II. Background

To determine the definition of a non-computable set, it requires first understanding what computability means. A computable set is visualized through natural numbers which in turn can be manipulated through computation, recursiveness, or decidability in the form of an algorithm. These numbers are directed as input, and function until there is a provided output derived from them. It works all in a finite amount of time, and allows for the algorithm to correctly decide the answer. On the contrary, non computability is in short the opposite. It is seen as a more theoretical approach, presenting numbers in a possible irrational fashion, and pushes the finite simplicial complexes that most computable functions do. The algorithms that are non-computable are left undecided, with no final result or output.

It is important to again note that just because something is seen as undecided, I would argue this does not mean unresolved. To draw the conclusion that there is no answer to this question, is an answer within itself. I share this ideology in terms of non-computable sets; there is wisdom in learning these, and we can find profit from their study.

The primary historical theorems that provide an excellent portrayal of non-computable sets both within mathematics and in the real world. First let's begin with a simpler understanding of something that reveals indecision. This comes from Tarski's Theorem on Truth, which was dedicated to the notion of truth as clear and precise.

A proposed definition of truth for the language of arithmetic is adequate if it implies all sentences of the form:
(T) $\ulcorner\Phi\urcorner$, is true if and only if Φ,
for Φ a sentence of the language of arithmetic and $\ulcorner\Phi\urcorner$, is Gödel number, and it also implies "($\forall\rightarrow$x)(x is true $\forall\rightarrow$ x is a sentence)."
(Massachusetts Institute of Technology)

In explanation for the above, Tarski focused his work on the non-arithmetical and semantic paradoxes. By doing so, his usage of the liar paradox helped reflect that certain sets reflected by other philosophers is actually not definable in natural numbers, coinciding with how it may be non-computable.

Furthermore, the famous Gödel's First Incompleteness Theorem helped guide formal systems to express basic arithmetic, and even arguing of the differences in finite lists of axioms. Gödel argued that in any given system of mathematics, there will always be true statements that cannot be proven. The need for completeness can never be satisfied, and in a way, paved the view of uncomputability. In other words,

If T is deductively complete, that is, for all formulas φ either T proves φ or T proves $\neg\varphi$,
then is not computable.

III. Non-Computability

Given the information from previous philosophers and mathematicians of what computability is, as well as how it can be shown through algorithmic theorems, I find that the complexity of a computable set gives an indication of if it can or cannot be computed. There is a limit when it comes to what we can solve, and when we are learning formulated solutions, we use non-computable sets to study how complex a function can be.

To build the groundwork for this proposal, I take a look at Jenny's (2018) paper. By drawing similar derivations of what non-computability is and taking a look at how he determines there is a need to study this realm, it will make it easier to bring the objective forward. The entirety of Jenny's paper is focused on the counterfactuals (e.g. counterpossibles) and metaphysicality of mathematical theorems. He builds on orthodoxy that argues on the side of fixing what we visualize as the computability theory. There are flaws in what we determine as indefeasible and reducible. The researchers build an expectation of authority through assertions, which ultimately causes these counterpossibles to be seen as a problem. However, these counterpossibles are exactly the staple of development in the majority of computability theories.

I now want to translate this towards what my current proposal is suggesting. This backing of having problems sometimes remain undecided, and counterpossibles leaving space for the unknown, it does not cause a discrepancy to any theory. If anything, it brings a greater appreciation for how precise these sets are. The determination of complexity within a problem space can guide this. We must distinguish and level the ranks at which when the problem gets so irrational, that it does not have a solution, only to be concluded as non-computable.

To grasp what a problem space can be, we visualize it through the differentiation of P vs NP, introduced in Chapter 9 of this course. Formally speaking, problems in P can be solved using an abstract computational model known as deterministic Turing Machines, and usually taking a polynomial amount of space; whereas problems in NP can be solved using non-deterministic Turing machines, and lies in the complexity space (Landman et al.). The majority of common problems computed fall under one of these categories, and it helps determine how although NP problems almost seem impractical to solve and live in the complexity space, I argue that they are just as prominent to machine modeling as P problems are. This differentiation between the two is reflected in terms of computability versus non-computability. I suggest it is almost impractical to ignore the NP class of a problem space, and that is exactly where the non-computable set is present.

Ultimately, what we are doing when we study non-computable sets is determining the complexity of it as such. It is not to find a solution or leave it unsolved, but draw conclusions of what our limitations are within the mathematical truths we have built the system upon. By finalizing the intricacies of problems that do not have a said output or cannot be used in a practical sense, the observation of its complexity helps build the problem space needed to better advance computability theories as a whole.

IV. Objections

It does seem necessary to observe the objections against the study of non-computable sets, as it is not the kind of inquiry that results in knowledge, especially with the flipside being computable sets, that due provide more immediate utility. There is no fault in this belief, yet, I would argue that findings of Gödel's Theorem throughout mathematical nature is what helps tell us that the need for non-computability studies only helps computable sets, instead of hindering. The theorem helps show existence of:

I. *Computable theories which do not have computable completions*
II. *Computable subtrees without a computable path*
III. *Computation continuous function without a computable supremum*
(more results of this documented in (Simpson, 2009))

These methods above help show just exactly the limit to which computability exists in the world of truths. When presented with an entire problem space, through determining the complexity of what can be decided and what remains undecided, carries over to benefiting the entirety of computability theories.

Likewise, much of the work done in studying non-computable sets is not practical, and only exists to be theoretical. The lack of application in the real-world deems it not useful to study. In response, this argument ties closely with conversion focused on polynomial vs non-polynomial time. Although when building an algorithm that only can be solved with NP time, it does not make it ineffective to draw a conclusion. Mathematicians dedicate time to comprehend what is decidable and it is for good reason. They benefit each other, and complexity is what guides them to do so.

V. Conclusion

In summary, there is substantial need for the study of non-computable sets, and direction from the problem space of complexity is what prospers it into an essential for all forms of the computability theory. In this paper, I highlighted various theorems that helped pave the way for non-computability; Gödel and Tarski made the idea of the incomplete real. Alongside this was work done by Jenny to present the relative computability theory, that seems to drive for refinement.

By shaping what is the non-computable sets, it gives the ability to explore fundamental limits of computation, and find the true depths of mathematical truths. These formal systems subsist for a reason, and it is deeming to keep establishing more upon it.