IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁑ Peer-reviewed / Refereed journal ⁑ Vol. 11, Issue 11, November 2024

DOI: 10.17148/IARJSET.2024.111103

The Existence of the Halting Function in a Purely Mathematical Context and the Solution of Halting paradox with termination checking of programs over definite Topological spaces. (AMS MSC2020 03-08) Category: Mathematics and computation

Dr.SOHAM DASGUPTA[1], DIPANJAN ROUT[2],SOURJYA GUPTA[3], ARCHISMAN MUKHRJEE[4]

PM SHRI KENDRIYA VIDYALAYA NO.2 SALTLAKE[1]

Department of Data Science and AI, IIT Madras[2]

Techno India University, Dept. of Computer science[3]

UG Basic Science NISER Bhubaneswar & Ex student of PM Shri Kendriya Vidyalaya No.2 Saltlake[4]

Abstract: Here we define, mathematically, a program $f_i : w \longrightarrow \{0,1\}_{\aleph_0}$. Where w is a set of all programmable words, we consider as the domain, and $\{0,1\}_{\aleph_0}$is the co-domain is the set of all finite or infinite strings of 0 & 1. (**Ref.1*). In this paper, we propose a function $f_i *$, which we call the stop function and we propose another function h, which we call the halt function. Our objective of the paper is to show their existence in a completely mathematical form of the, well known halting problem and its solution using simple functional compositions. Our next approach is to study the structure of the domain of programmable functions i.e. w and its topology with respect to the topology of $\{0,1\}_{\aleph_0}$. Followed by defining the finite string topology and the product topology on $\{0,1\}_{\aleph_0}$and study the continuous functions from w to $\{0,1\}_{\aleph_0}$. Our main intention is to show that a programmable function will terminate for a specific input if and only if the function is continuous at that specific input (point on w).

The main deference between the classical halting problem and our method is instead of mechanical switch program we utilize two-point functions. Which leads us to generate some interesting results through a purely mathematical context.

Keywords: Halting Problem, Turing machine, Undecidability, Stop function, Halting function,Product topology

INTRODUCTION

Alan Turing's 1936 introduction of the halting problem is regarded as one of the most significant advances in theoretical computer science. It is a classic illustration of an undecidable problem one which cannot be solved always by any algorithm. The halting problem specifically focuses on figuring out if a given program will continue indefinitely or finally end for a given input. This issue, which shows inherent limitations in what can be calculated algorithmically, sits at the intersection of computability theory and mathematical philosophy.

In this paper, we revisit the halting problem by developing a mathematical foundation for the existence of halting functions, which can indicate whether a program terminates for certain inputs. We explore the structure of the set of all programs, modeling them binary strings, and introduce the concept of a stop function to describe the behavior of programs for different inputs. Our approach allows us to construct a halting function within this framework, providing insight into the conditions under which a program halts or non-terminating.

DOI: 10.17148/IARJSET.2024.111103

Furthermore, using topological techniques, we demonstrate which are the programable words and how particular programs can be designed to fail under the assumptions of a general halting algorithm, reinforcing the undecidability of the problem. In the prospects machine the existence of halting function leads to a contradiction, but in a completely in a theoretical and mathematical approach its existence is definable. Thus, roughly speaking there exists some problem which computer cannot solve but it is solvable in a completely theoretical and mathematical way.

The Halting Problem and its classical proof for undecidability.

The main concept of the halting problem is somewhat straightforward: given an input and a program, is it possible to predict whether the program will terminate or run forever? Undecidability became a fundamental aspect of computational theory when Turing's groundbreaking work demonstrated that no generic algorithm may determine this for all programs and inputs. Although there are some specific situations in which termination can be anticipated, this basic restriction applies to the wide range of programs.

Let $h(i, x)$ be a function that decides whether the program P_i halts on input x. We define $h(i, x)$ as:

$$h(i,x) = \begin{cases} 1 & \text{if } P_i \text{ halts on input } x, \\ 0 & \text{otherwise.} \end{cases}$$

The goal is to prove that no total computable function can always correctly compute $h(i, x)$, i.e., the function h is undecidable.

Proof by Contradiction

Assume, for contradiction, that such a total computable function $h(i, x)$ exists. We now define a new function $g(i)$ based on an arbitrary total computable function $f(i, i)$:

$$g(i) = \begin{cases} 0 & \text{if } f(i,i) = 0, \\ \text{undefined} & \text{if } f(i,i) \neq 0. \end{cases}$$

Here, $g(i)$ is a partial computable function, depending on the value of $f(i, i)$.

We now define a program e that computes $g(i)$:

We now define a program e that computes $g(i)$:

```
procedure $e(i)$:
    $if\ f(i,i) == 0$ :
        $then\ return\ 0$
    $else$:
        Non terminating
```

This program halts and returns 0 if $f(i, i) == 0$, and non-terminating if $f(i, i) \neq 0$.

This program halts and returns 0 if $f(i, i) = 0$, and non-terminating if $f(i, i) \neq 0$.

Behavior of e on Input e

Now, we consider the behavior of e when given the input e:

- **Case 1**: If $f(e, e) = 0$, then $g(e) = 0$, meaning e halts on input e, so $h(e, e) = 1$.
- **Case 2**: If $f(e, e) \neq 0$, then $g(e)$ is undefined, meaning e does not halt on input e, so $h(e, e) = 0$.

Contradiction

This leads to a contradiction:

- If $f(e, e) = 0$, then $h(e, e) = 1$, implying e halts.
- If $f(e, e) \neq 0$, then $h(e, e) = 0$, implying e does not halt.

Thus, $h(e, e)$ cannot simultaneously be both 1 and 0, which contradicts the assumption that $h(i, x)$ is a total computable function.

Mathematical approach to redefine the problem and to investigate its solution.

Definition : Let w be the set of all programable sentences and $\{0,1\}_{\aleph_0}$ is the collection of any finite or infinite countable strings of 0 and 1. Now, $f_i : w \longrightarrow \{0,1\}_{\aleph_0}$ is called a program.
$\rho = \{f_i | f_i : w \longrightarrow \{0,1\}_{\aleph_0}\}$ is the set of all programs.

Definition : Now we define f_i^*as the stop function for every f_i that is if f_i(x) is finite for some input x belongs to w then f_i^*(x) = 0, else 1. i.e. $\begin{cases} f_i^*(x) = 0 \ \ if \ f_i(x) \ is \ finite \\ f_i^*(x) = 1 \ if \ f_i(x) \ infinite \end{cases}$ $x \in w$

The set of all such stop functions is $\rho^* = \{f_i^* | f_i^* : w \longrightarrow \{0,1\}\}$.

Now we defining the $h: \{0,1\} \longrightarrow \{0,1\}$ named as the halting function. Observe that there exist only one such non constant function out of 4. Also, that is self-invertible.

Now $h \circ f_i^* : w \longrightarrow \{0,1\}$. Obviously $h \circ f_i^* \epsilon \rho$

Hence $\begin{cases} (h \circ f_i^*)(x) = h[f_i^*(x)] = 1\,; If \ \ f_i^*(x) = 0 \\ \ (h \circ f_i^*)(x) = h[f_i^*(x)] = 0; If \ f_i^*(x) = 1 \end{cases}$ $x \in w$

Now, observe the following composite function $h \circ [h \circ f_i^*]$

that is $h \circ [h(f_i^*(\mathrm{x}))] = 0$ if $f_i^*(x) = 0$

and $h \circ [h(f_i^*(\mathrm{x}))] = 1$ if $f_i^*(x) = 1$

Thus, there exists a halting function h theoretically. So, we can conclude that for every program f_i and any input $x \in w$ there is a halting function h along with the program $h \circ f_i^*$, which can able to say that f_i terminates or not for the certain x.

Topological structures of $\{0,1\}_{\aleph_0}$and *w*

Now our second intension is to study that the structure of w, such that the finite space and product space structure of $\{0,1\}_{\aleph_0}$ and with respect to them the topological structures of w.

Firstly, we are defining a simple topology on $\{0,1\}_{\aleph_0}$ named as the *finite space topology* τ .

Definition: The base of τ consists of all the subsets B_s of $\{0,1\}_{\aleph_0}$ which contains only finite strings of 0 & 1 as the open sets.

Obviously $\bigcup_{s \in S} B_s$ is also containing only finite strings of 0 & 1 similarly for $\bigcap_{s=1}^{n} Bs$. So τ is closed under arbitrary union and finite intersections. Thus, along with Ø and $\{0,1\}_{\aleph_0}$ itself. So τ is a topology on $\{0,1\}_{\aleph_0}$

Theorem: *A program terminates for a certain input if and only if it is continuous*.

From the previously defined topology τ on $\{0,1\}\aleph_0$ for every program i.e. $f_i: w \longrightarrow \{0,1\}_{\aleph_0}$if we consider the sets $f_i^{-1}(any \ set \ of \ finite \ strings \ of \ 0,1) \subseteq w$,we can generate different topological bases on w depending upon each f_i.
Since $f_i^{-1}(\bigcup_{s \in S} B_s) = \bigcup_{s \in S} f_i^{-1}(B_s)$ and $f_i^{-1}(\bigcap_{s \in S} B_s) = \bigcap_{s \in S} f_i^{-1}(B_s)$.
We denote them as $w\ [\tau(f_i^{-1})]$={ $f_i^{-1}(B_s) \mid B_s \in \tau - \{\{0,1\}_{\aleph_0}\} \cup \{w\}$. Therefore, on this context we can conclude that for such topological spaces a program f_i will be continuous for a $x \in w$ if and only if $f_i(x)$ is an output in finite string of 0&1 that is $f_i^*(x)$= 0 if and only if f_i is continuous at x.
Thus, for here we can say roughly that a program terminates for a certain input if and only if it is continuous.

Cartesian product structure and the product topology on $\{0,1\}_{\aleph_0}$

Secondly, we can generalize it more by introducing product topology on $\{0,1\}_{\aleph_0}$.Before that we need to redefine the structure of $\{0,1\}_{\aleph_0}$as ∪{X_i ǀ X_i= $\{0,1\}^i$; $0 \leq i \leq \aleph_0$ }. That is from now we will consider the elements of $\{0,1\}_{\aleph_0}$as the elements of some cartesian product of {0,1}. Now lets consider the following set X{X_i ǀ X_i = $\{0,1\}^i$; $0 \leq i \leq \aleph_0$ } , in short we write it X[X_i] as the cartesian product of all the X_i 's.

Thus X[X_i]= {f I f⟶ ∪{{X_i ǀ X_i = $\{0,1\}^i$; $0 \leq i \leq \aleph_0$ } with f(i) ∈ X_i }

⇒ X[X_i]= {f I f:I⟶$\{0,1\}_{\aleph_0}$& f(i) ∈ X_i } where $I = \mathbb{N} \cup \{ \aleph_0\}$ as the index set.
Now let for every X_i there is a topology $\mathfrak{I}_i$ assigned to it. i.e. (X_i, $\mathfrak{I}_i$) are pre-defined topological spaces.
Now the functions p_j: X[X_i] ⟶ X_i defined as p_j(f)=f(j) with j∈I , are our required projection maps.
The product topology X$\mathfrak{I}_i$ on X[X_i] will be defined as the weakest topology where all the projection maps p_j are continuous.

Theorem: *The projection of a product space into each of its coordinate spaces is open.*

Let P_c be the projection of X { X_a: a ∈ A} into X_c. In order to show that P_c is open it is sufficient to show that the image of a neighborhood of a point x in the product is a neighborhood of P_c(x), and it may be assumed that the neighborhood in the product space is a member of the defining base for the product topology. Suppose that $x \in V = \{y{:}\, y_a \in U_a\, for\ a\ in\ F\}$, where F is a finite subset of A and U_a is open in X_a for each a in F. We construct a copy of Xe which contains point x.
For z ∈Xe let $f(z)_e = z$, and for a ≠c let $f(z)_a = x_a$ .Then $P_c \circ f(z) = z$
If $c \notin F$, then clearly f[X_c] ⊂V and P_c[V] = X_c which is open. If c ∈ F, then f (z) ∈ V iff z ∈ U_c and P_c[V]= U_c.
The theorem follows. (As a matter of fact, the function f defined in this proof is a homeomorphism, a fact that is occasionally useful.)

Theorem: A *function f on a topological space to a product X {X_a:a ∈ A} is continuous if and only if the composition $P_a \circ f$ is continuous for each projection P_a.*

If f is continuous, then $P_a \circ f$ is always continuous because Pa is continuous. If $P_a \circ f$ is continuous for each a, then for each open subset U of X_a the set $(P_a \circ f)^{-1}[U] = f^{-1}[{P_a}^{-1}[U]]$ is open. It follows that the inverse under f of each member of the defining subbase for the product topology is open, and hence f is continuous.

Thus, we are using this result for every program f_i where $f_i{:}\, w \longrightarrow \{0,1\}_{\aleph_0}$to check that they are continuous or not. On otherward to check a program terminates or not for a specific input.

CONCLUSION

The mathematical fundamentals of halting functions were the primary topics of our discussion. We begin by defining the stop function for a class of programs that will terminate for certain inputs and intends to run indefinitely for others. Analyzing these functions within the framework of programmatic sentences and countable binary strings, we are able to establish the existence of a halting function. We explore the zero sets of continuous functions using their topology and product space structure, offering fresh perspectives on the connection between input structures and program termination.

Additionally, we examine programs that do not halt for specific inputs, extending the classical proof by contradiction that no algorithm can universally solve the halting problem. We construct programs where recursive loops ensure non-termination under particular input conditions, using these results to show the inherent undecidability of certain computational problems. This study provides a deeper understanding of program behavior in relation to the halting problem and offers a theoretical framework for future computational complexity research.

REFERENCES

[1]. https://www.rapidtables.com/convert/number/ascii-to-binary.html
[2]. ***Code:*** ***https://github.com/DR2K02/Halting Problem***
[3]. Topological Structures by Wolfang J. Thron (Holt Rinehart and Winston)
[4]. https://encyclopediaofmath.org/index.php?title=Turing_machine
[5]. CS208: Automata Theory and Logic Part II: Turing Machines and Undecidability (IIT BOMBAY)
[6]. General Topology by N.Bourbaki (Vol.I&II) Addison Wesley
[7]. Introduction to the Theory of Computation by Michael Sipser
[8]. Introduction to Algorithms by Thomas H. Cormen

BIOGRAPHY

1. Dr.Soham Dasgupta (PGT Math PM Shri Kendriya Vidyalaya No.2 Saltlake,Kolkata) t1273.soham.dasgupta@kvsrokolkata.co.in , mathsoham@gmail.com
2. Dipanjan Rout (Department of Data Science and AI, IIT Madras) 21f3002560@ds.study.iitm.ac.in
3. Sourjo Gupta (Techno India University, Dept. of Computer science) sourjya231001202002@technoindiaeducation.com
4. Archisman Mukherjee (UG Basic Science NISER Bhubaneswar & Ex student of PM Shri Kendriya Vidyalaya No.2 Saltlake) archisman.mukherjee@niser.ac.in