Devansh Shah

1914078

B1

Group 6

**Approximations to the Halting Problem**

**Introduction**

* In 1931, the logician KURT GODEL constructed a mathematical predicate which could neither be proven nor falsified. In 1936, ALAN TURING introduced and showed the Halting Problem H to be undecidable by a Turing machine. This was considered a strengthening of Godel’s result regarding that, at this time and preceding AIKEN’s Mark I and ZUSE’s Z3, the Turing machine was meant as an idealization of an average mathematician.
* Nowadays the Halting Problem is usually seen from a quite different perspective. Indeed with the advent of and increasing reliance on high speed digital computers and huge pieces of software running on them, source code verification or at least the detection of stalling behaviour becomes ever more important.
* In fact, by RICE’s Theorem, this is equivalent to many other real-world problems arising from goals like automatized software engineering, optimizing compilers, formal proof systems and so on. Thus, the Halting problem is a very practical one which has to be dealt with some way or another. One direction of research considered and investigated the capabilities of extended Turing machines equipped with some kind of external device solving the Halting problem.
* While the physical realizability of such or other kinds of super-Turing computers is questionable and in fact denied by the Church-Turing Hypothesis, the current field of Hypercomputation puts this hypothesis into question. On the theoretical side, these considerations led to the notion of relativized computability and the Arithmetical Hierarchy which have become standard topics in Recursion Theory.
* Since this problem cannot be solved, we find approximate solutions for the halting problem.

**Concept of the research paper**

* In decision problems, a notion of approximate solution has been established in Property Testing. Here for input x ∈ Σ n , the answer “x ∈ L” is considered acceptable even for x not belonging to L provided that y ∈ L holds for some y ∈ Σ n with (edit or Hamming) distance d(x,y) ≤ εn. Observe that this notion of approximation strictly speaking refers to the arguments x to the problem rather than the problem L itself. Also, any program source x is within constant distance from the terminating one y obtained by changing the first command(s) in x by a halt instruction.
* Average case analysis is an approach based on the observation that the hard instances which make a certain problem difficult might occur only rarely in practice whereas most ‘typical’ instances might turn out as easy.
* So, although for example N Pcomplete, an algorithm would be able to correctly and efficiently solve this problem in, say, 99.9% of all cases while possibly failing on some few and unimportant others. In this example, ε = 1/1000 is called the error rate of the problem under consideration with respect to a certain probability distribution or encoding of its instances.
* Such weakenings have previously been mainly applied in order to deal with important problems where the practitioner cannot be silenced by simply remarking that they are NP-complete, that is, within complexity theory. However the same makes sense, too, for important undecidable problems such as Halting: even when possibly erring on, say, every 10th instance, detecting the other 90% of stalling programs would have prevented many buggy versions of a certain operating system from being released prematurely.

There have been several propositions and proofs for the same trying to sharpen the approximation where using the results from the proofs, some propositions are termed false and others can not find the exact solution and reach an approximation for the same.

**Problem Discussed**

Halting means that the program on certain input will accept it and halt or reject it and halt and it would never go into an infinite loop. Basically halting means terminating. So can we have an algorithm that will tell us whether the given program will halt or not. In terms of a Turing machine, will it terminate when run on some machine with some particular given input string.

We cannot design a generalized algorithm which can appropriately say that given a program will ever halt or not.The only way is to run the program and check whether it halts or not.

One way to define Halting problem with respect to programming is:

* A G¨odelization ϕ is a sequence of all partial recursive functions s.t.

– there exists a partial universal program u with ϕu(hi,xi) = ϕi(x) (UTM)

– and a total program s with ϕs(hi,xi) (y) = ϕi(hx,yi) (SMN)

– for a bijective computable function h<·,·>i : Σ ∗ ×Σ ∗ → Σ ∗ or h<·,·>i : N×N → N. called pairing function. The Halting problem for ϕ is Hϕ = {hi,xi: x ∈dom(ϕi)}

* The Halting problem is sometimes alternatively defined as the task H˜ ϕ of deciding whether a given program i terminates on the empty input, that is, whether λ ∈ dom(ϕi); or the question whether i ∈ dom(ϕi).
* Halting ratio states that neither nearly all programs halt nor do nearly all of them stall.

**Conclusion**

* Since, the Halting problem is of practical importance yet cannot be solved in the strict sense, we considered the possibility of approximating it. Similarly to the average-case theory of complexity, this depends crucially on the encoding of the problem, that is here, the programming system under consideration.
* Many practical programming languages lacking density in fact do admit such an approximation with asymptotically vanishing relative error for the simple reason that the fraction of syntactically incorrect instances tends to 1. This was exemplified by a combinatorial analysis of the Turing-complete formal language BF.
* Here and in similar cases, the question for approximation the Halting problem is equivalent to a mere syntax check and thus becomes trivial and vain. On the other hand, considering only syntactically correct sources was established to yield an efficient and dense programming system in the case of BF. For any such system, we proved a universal constant lower bound on relative approximations to the Halting problem even in the weak io-sense.

Hence from this paper, we can conclude that there is no algorithm to solve the halting problem but using the proposed definitions and their respective proofs, we can approximate it to an extent.