The Riemann Hypothesis

Cristian Dumitrescu

Abstract. In this article I describe a proof of the fact that ZFC cannot say much about a Turing machine that takes a very long time to halt (if it eventually halts). The consequences of this fact in relation to the Riemann Hypothesis are presented.

Keywords. Riemann Hypothesis, Robin's reformulation, Lagarias' reformulation, Littlewood's reformulation, algorithmic complexity.

Mathematical Subject Classification Number: 11M99.

Introduction.

The Riemann Hypothesis has been an open problem for a long time. The reason why this problem remained not solved for over 150 years is that ZFC cannot settle this problem. In this article I present the main ideas in support of this view.

Section 1. Reformulations of the Riemann Hypothesis.

The following reformulations have been known for some time and the proofs of the statements in this section can be found in many references.

Lagarias' reformulation of RH [1]**.** The Riemann Hypothesis is true if and only if $\sigma(n) \leq H_n + \exp(H_n) \cdot \log(H_n)$, for all n (here $\sigma(n)$ is the sum of divisors function and H_n represent the harmonic sums).

Robin's reformulation of RH [1, 6]**.** The Riemann Hypothesis is true if and only if there is an n_0 (and in fact $n_0 = 5040$) such that $\sigma(n)/n < e^{\gamma} \cdot \log(\log(n))$, for all $n > n_0$ (here $\sigma(n)$ is the sum of divisors function).

Littlewood's reformulation of RH [4]**.** The Riemann Hypothesis is equivalent to the statement that for every $\varepsilon > 0$, we have $M(x) = O(x^{(1/2 + \varepsilon)})$, when $x \to \infty$ (here M(x) is the Mertens' function).

Section 2. Algorithmic Complexity.

I assume that the reader is familiar with the fundamentals of Algorithmic Information Theory, in particular with Kolmogorov complexity, prefix free codes and Chaitin complexity. In [7] it is stated the following theorem.

Theorem [2, 7]. Let T be a 1 - consistent theory and U a universal Chaitin computer. Then there is a positive constant C (depending only on U) such that T can determine at most H(T) + C bits of Ω.

In the theorem above, H(T) represents the number of bits it takes to describe the arithmetical theorems of T. In other words, H(T) represents the algorithmic complexity of the program (Turing machine) that generates all the theorems of T (from axioms and rules of inference).

We consider now Lagarias' reformulation of RH above. We consider the Turing machine Z (that starts from an empty tape) that takes the positive integers n in sequence, calculates the sum of divisors $\sigma(n)$ and the harmonic sum H_n (for each n), and then checks whether the inequality $\sigma(n) \leq H_n + \exp(H_n) \cdot \log(H_n)$ is satisfied. If it is satisfied, then it follows the same procedure for n + 1, and so on. This Turing machine (Z) only stops if it finds an n such that $\sigma(n) > H_n + \exp(H_n) \cdot \log(H_n)$, otherwise it runs forever (it never stops).

We consider a list of all possible programs P_1 , P_2 , P_3 ,…….P_n , ….., encoded as binary strings in lexicographic order.

Define $\tau = \sum_{P_n \text{ halts}} 2^{-n}$. This number is an encoding of the answers to every instance of the halting problem in a single real number. The n-th bit of τ is 0 if the program P_n runs forever, and it has the value 1 if the program P_n halts in finite time. As far as I know, Turing first considered this oracle.

The real number τ is not a random number. The complexity of τ(n), an initial segment of length n is around log(n) [5].

We know how to compute an infinite number of its bits, but an infinity of others are unknowable.

We consider two columns, the left column contains the binary codes of all the programs in lexicographic order. In the right column we have the number τ, with its bits corresponding to each program in the left column.

We write l(P) for the length of the binary encoding of the corresponding program P, and we write t(P) for the number of state transitions, or time steps, before the program P halts. If P does not halt in finite time, then $t(P) = \infty$.

We define a **n - counter program** C(n) as a program that counts till n and then halts. The length l(C(n)) of the binary encoding of a n - counter program satisfies $l(C(n)) < n + C$, where the constant C can be determined. In other words, the length l(C(n)) of the binary encoding of a n - counter program is at most n + C.

We define a **late program** as a program P for which $t(P) > l(P) - (C + 1)$ and $t(P) < \infty$. The reason behind this definition is that we want to make sure that all the n - counter programs are also late programs (according to our definitions). In the following, when we write $t(P) > l(P)$, we understand that we consider the inequality $t(P) > l(P) - (C + 1)$, where the constant C can be effectively computed. For simplicity in the presentation, we just write $t(P) > l(P)$.

We reorder **only** the late programs P (for which $t(P) > l(P)$ and $t(P) < \infty$), in such a manner that $P_i < P_j$ (in the new order), iff $t(P_i) < t(P_j)$. Only the late programs P for which $t(P) > l(P)$ and $t(P) < \infty$ might change their order rank in the left column. The number τ remains unchanged in the right column.

If we know that a late program P halts in time t(P), and that $t(P) > l(P)$ and $t(P) < \infty$, then we can compute all the bits corresponding to all the programs before P in the order relation defined above. Note that if this program halts, but halts early ($t(P) < l(P)$), then the statement above is not true.

We note that the programs that do not halt, or the programs that halt too early (for which $t(P) < l(P)$) keep their order rank (inherited from the original lexicographic order) in the defined new order. Only the programs P that halt late (for which $t(P) > l(P)$ and $t(P) < \infty$) might change their order rank in the left column.

We can find the values for an infinity of bits of τ, but we can only know that a finite number of programs halt late. In other words, if we know that an infinity of bits of τ have value 1, and that they are all related to programs that halt late (for which $t(P) > l(P)$ and $t(P) < \infty$), then we can reconstruct all the bits of the number τ. This would be a contradiction, since the halting problem is unsolvable. We write P_{late} for the program that we can prove that halts late ($t(P_{late}) > l(P_{late})$ and $t(P_{late}) < \infty$), and has the highest order rank in the order relation defined above. We proved that this program exists. We want to find the conditions under which our Turing machine Z has an order rank that is greater that the order rank of P_{late}.

We consider all the bits of τ, up to the bit corresponding to the program P_{late}, and we write N for its length. This initial segment of τ of length N has complexity around log(N). That means that no program much smaller log(N) can generate this initial segment of length N of τ. If a formal system like ZFC can generate an initial system of τ of length N, then this N has a maximum value around $C \cdot 2^{(H(ZFC))}$ (where the constant C can be effectively determined, and does not necessarily have the same value as the constant mentioned before).

As defined before, whether the Turing machine Z (related to the Lagarias‘ reformulation

of RH) halts or runs forever is equivalent to the Riemann Hypothesis being false or true. It is easy to prove that if Z ever halts, then Z is a Turing machine that halts late ($t(Z) > l(Z)$).

If we can verify that Z also satisfies $t(Z) > C \cdot 2^{(H(ZFC))}$, then this means that the order rank of Z (for the order relation defined above) is greater than the length of the initial segment of τ for which ZFC can determine all the bits (we note that $rank(Z) > t(Z)$, for machines that halt late, at least up to a constant, because for any n, there is a n - counter program that halts late). Actually, since it is known that statistically, most programs halt early or run forever, that means that we have to check whether $t(Z) > K$, where K is much smaller than $C \cdot 2^{(H(ZFC))}$.

What we are trying to prove is that the order rank of the program Z is greater than $C \cdot 2^{(H(ZFC))}$. Once this is proved, then we can conclude that ZFC cannot decide whether Z halts in finite time or runs forever, because the order rank of Z is greater than the order rank of P_{late}. ZFC cannot decide whether RH is true or false.

We can then formulate the following theorem.

Theorem. If the Turing machine Z (defined above, as related to RH) satisfies the relation $t(Z) > C \cdot 2^{(H(ZFC))}$ (where the constant C can be determined), then ZFC cannot determine whether the Turing machine Z eventually halts in finite time or runs forever. ZFC cannot determine the value of the bit of τ corresponding to program Z (in other words, Z has a very large order rank in the order relation defined above).

Observation. We used here Lagarias' reformulation of the Riemann Hypothesis, but we can use Robin's reformulation, or other reformulations [3]. The essential idea is to link the Riemann Hypothesis to a Turing machine in such a manner that RH is false iff the Turing machine under consideration eventually halts in finite time (and RH is true iff the Turing machine under consideration does not halt in finite time).

We also note that verifying that the Turing machine Z does not halt in less that $C \cdot 2^{(H(ZFC))}$ computation steps is not an easy task (this may be a task for a quantum computer).

The conclusion is that ZFC cannot say much about a Turing machine that takes a very long time to halt (even if it eventually halts in finite time). The very essence of how a formal system is defined must be modified, in order to deal with these problems, but this is a different challenge altogether. Another conclusion is that, since if t(Z) is finite, then this finite computation would represent by itself a proof that RH is false, the only conclusion that we can reach is that the Riemann Hypothesis is true (as a metatheoretical argument). Refinements of this method can probably lower the value of K. In problems of this type, a very large, but finite computation, would be sufficient, in order to settle the Riemann Hypothesis. Very large, but raw computations can settle the Riemann Hypothesis, and other problems of a similar nature. We can estimate an upper bound for H(ZFC), and the constant C.

References:

1. K. Briggs, *"Notes on the Riemann Hypothesis and abundant numbers"*, 2005 more.btexact.com/people/briggsk2/

2. C. S. Calude, *"Chaitin Ω Numbers, Solovay Machines, and Incompleteness"*, CDMTCS Research Report Series - 114, October 1999.

3. C. S. Calude, *"A New measure of the Difficulty of Problems"*, CDMTCS Research Report Series - 277, February 2006.

4. H. M. Edwards, *"Riemann's Zeta Function"*, Dover Publications, Inc., 2001.

5. T. Ord, T. D. Kieu, *"On the Existence of a New Family of Diophantine Equations for Ω"*, arXiv:math/0301274v3 [math.NT] 12 Oct. 2003.

6. G. Robin, *"Sur L'Ordre Maximum de la Fonction Somme des Diviseurs", Seminaire Delange-Pisot-Poitou, Theorie des nombres (1981 - 1982), Progress in Mathematics 38 (1983), 233 - 244.*

7. R. M. Solovay, *"A version of Ω for which ZFC can not predict a single bit"*, in C. S. Calude, G. Paun (eds.), Finite versus Infinite. Contributions to an Eternal Dilemma, Springer - Verlag, London, 2000, 323 - 334.

Cristian Dumitrescu,
119 Young St., Ap. 11,
Kitchener, Ontario N2H 4Z3,
Canada.

Email: cristiand43@gmail.com
cristiand41@hotmail.com

Tel : (519) 574-7026