Volume 5, Issue 6, June 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Analysis of Busy Beaver

Harjot Tiwana, Rahul Kumar Singh
Department of Computer Science & Engineering
Chandigarh University, Gharuan, India

Abstract— ***Since 1962, when Tibor Rado proposed the Busy Beaver function, it has attracted the attention of many researchers and several contests were organized trying to produce good solutions. This paper describes a problem of Rado on Turing machine now known as the Busy Beaver problem. This problem involves the search for the Turing Machine that produces the maximum number of ones when started on a blank tape. The aim of this paper is to undertake the systematic review on the Busy Beaver. The study was carried out using a theoretical hard problem of Busy Beaver.***

Keywords— ***Busy Beaver, Turing machine, non-computable functions, theoretical computer science.***

I. INTRODUCTION

One of the most important results of theoretical computer science deals with the existence of non-computable functions. This fact can be easily established showing that there are functions, which are not Turing computable. Even Turing machine was made for universal computation. It is the well known models for universal computational. It means that Turing machine can compute anything that can be computed by human. Turing machine was inspiration of the computer system that came two decades later [1]. But there are some functions,which are not turing computable. Tibor Rado proposed one such functions in 1962 which is known as "Busy Beaver" in current era [2]. In the Computer Recreations department of a recent issue of Scientific American [20], A. K. Dewdney discusses efforts to calculate the Busy Beaver function ∑. This is a very interesting endeavour for a number of reasons. Busy Beaver is basically a problem of the Turing machine. The original definition proposed by Rado [2] [11], considered deterministic 5-tuple TMs with N+1 states (N states and an anonymous halting state). In each transition, the machine writes a symbol to the tape and moves its head either left or right, i.e., the transition function has the following format:

∂: Q□ □ □ □ □ Q□ □ □ □ {L, R}

Where L denotes move left and R move right. A common variation consists in considering 4-tuple TMs, where the transition function has the following format:

□ : Q□ □ □ □ □ Q□ {□ □ {L, R}}

i.e., a 4-tuple TM either writes a new symbol on the tape or moves its head before entering the new state.

The productivity of a Turing Machine (TM) can be defined as the number of ones present on the (initially blank) tape when the machine halts. Machines that do not halt have productivity zero. The function ∑N is defined to be the maximum productivity that can be achieved by N-state TM. This TM is called a Busy Beaver [2] [11].

In the 4-tuple variant, productivity is usually defined as the length of the sequence of ones produced by a TM when started on a blank tape, and halting when scanning the leftmost one of the string, with the rest of the tape blank [2].

II. TURING MACHINES

Turing machine was invented by Alan Turing in 1936.Turing machine was inspiration of the computer system that came two decades later. Turing machine was made for universal computation. It means that Turing machine can compute anything,which is computable [1]. Turing machine has two-way infinite tape which is divided into number of cells. Cell can either be blank or contain a non-blank symbol. Each cell contains only one symbol. Turing machine has one head, known as R\W head (Read & Write head) that move over the cells of tape. R/W head can examine the one cell at a time. At each step, the machine reads the symbol under the head, and depending upon the present state, it write new symbol in the cell under the head and goes to new state. The R/W head can either move left or right [1] [7].The main steps followed by a Turing machine:

- A new symbol is written on the cell under the R/W head
- A movement of R/W head: either head moves one cell left (L) or one cell right (R)
- Goes to the new state
- Whether to halt or not [18]

Definition [1]: A Turing machine M has 7-tuple namely (Q, ∑, ┌, ∂, q_0, b, F) where

1. Q is a finite non-empty set of states.
2. ┌ is a finite non empty set of tape symbols.
3. b ∈ ┌ is the blank.

4. $\sum$ is a non-empty set of input symbols and is a subset of Γ
5. ∂ is the transition function mapping (q, x) onto (q', y, D) where D is direction of movement of R/W head.
6. $q_0 \in Q$ is the initial state, and
7. $F \subseteq Q$ is the set of final states. [18]

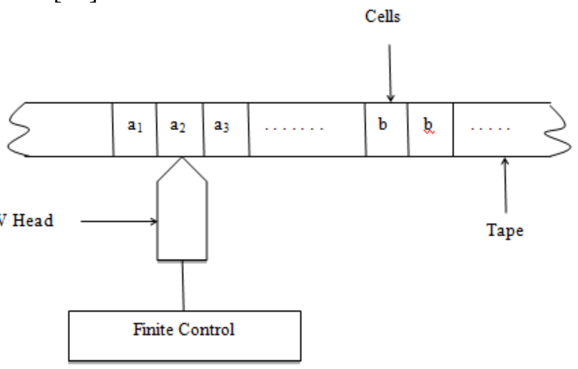

Fig 1 Turing machine

The Church-Turing thesis states that any algorithmic procedure that can be carried out by human beings/computer can be carried out by a Turing machine. It has been universally accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer [7] [18].
Turing machines are quite useful in several ways. As an automaton, the Turing machine is the most general model. It accepts type-0 languages. It can also be used for computing functions. It turns out to be a mathematical model of partial recursive functions. Turing machines are also used for determining the un-decidability of certain languages and measuring the space and time complexity of problems. In Turing machines, the acceptability of a string is decided by the reachability from the initial state to some final state. So the final states are also called the accepting states [18].

III. BUSY BEAVER

Suppose a Turing Machine (TM) with a two-way infinite tape and a tape alphabet = {blank, 1} (the symbol 0 is used as blank symbol) [2]. Also assume that Turing machine initially completely blank and the machine must move either left or right at each step, i.e., it cannot remain stationary. There is single halting state from which no transitions emerge, and this halt state is not counted in total number of states [19]. The question Rado asked was: What is the maximum number of 1's (not necessarily consecutive) that can be written on the tape after such an N-state Turing machine halts, when started on a blank tape [2][19]? This number, which is function of the number of states, is denoted by $\sum N$ (Rado's function). A machine that produces $\sum N$ non-blank cells is called a Busy Beaver (BB) [2]. Rado also defined a function S (N) which counts the maximum number of moves that can be made by N-state halting Turing machine of this form [19]. If a Turing machine writes the maximum possible number of 1's for its number of states then it is called a "Busy Beaver" [19]. For example, consider the 3-state Turing machine. If a machine is started with a tape of all blanks, it halts with 13 moves with 6 consecutive 1's on the tape. This shows that $\sum(3) \geq 6$ and $S(3) \geq 13$ [19].
Busy Beaver are hard to find, even for relatively small n, for two reasons. First, the search space is extremely large – there are $(4(N+1))^{2N}$ different Turing machines with N-states. Second, it is in general not possible to determine whether a particular Turing machine will halt or not. So, it is clear that neither $\sum N$ nor S(N) are computable functions – it means that theres is no halting Turing machine, which on arbitrary input N, will always halt and successfully compute these functions [19]. $\sum N$ grows very faster than any computable function [2] [11]. Nevertheless, it is possible to compute $\sum N$ and S(n) for small values of N [19]. As the number of states increases, the problem become harder. There is no particular theory about the structure of Busy Beaver.The only way for finding such machines is to performing an exhaustive search for all N-state Turing machines[2] [19] .
The following table gives what is known about $\sum N$ and S(N) $1 \leq N \leq 5$ [19].

Table 1: $\sum N$ and S(n) for $1 \leq N \leq 5$

N	$\sum N$	S(N)
1	1	1
2	4	6
3	6	21
4	13	107
5	≥ 4098	≥ 47,276,870

IV. APPLICATIONS OF BUSY BEAVER

The Busy Beaver function is of interest to information theorists, because it measures the capability of computer programs as a function of their size, as a function of the amount of information which they contain [6]. $\sum N$ is defined to be the largest number which can be computed by an n-state Turing machine. So, it is clear that the correct measure is bits, not states. Thus it is more correct to define ∑N as the largest natural number whose program size complexity or algorithmic information content is less than or equal to N. Of course, the use of states has made it easier and a definite. Yet another reason for interest in the Busy Beaver function is that, when properly defined in terms of bits, it immediately provides an information-theoretic proof of an extremely fundamental fact of recursive function theory, namely Turing's theorem that the halting problem is unsolvable [6].

The Busy Beaver function is also of considerable mathematical interest, in principle it would be extremely useful to know larger values of ∑N. Let P be a computable predicate of a natural number, so that for any specific natural number *n* it is possible to compute in a mechanical fashion whether or not P(n), P of n, is true or false, that is, to determine whether or not the natural number n has property P. How could one use the Busy Beaver function to decide if the conjecture that P is true for all natural numbers is correct? An experimental approach is to use a fast computer to check whether or not P is true, say for the 1st billion natural numbers [6].

Now let's prove that niether ∑N not S(N) are computable [19].

Theorem 1 [19]**:** The function ∑N is not computable by Turing machine.

Proof [19]*:* The idea is to show that if f(N) us any computable funtion, then there exist N_0 such that $\sum N > f(N)$ for $N>N_0$. Our model of computable function is that a Turing machine calculating f(N) starts with a tape with a block of N 1's immediately to the right of starting blank, and halts after a finite number of moves with block of f(N) consecutive 1's on a tape [19].

Let f be an arbitrary computable function and define

$$F(x) = \sum_{0\leq i \leq x} (f(i) + i^2)$$

Since f is computable, so is F. In fact, there is a Turing machine M_F that, when started with a tape with x 1's, writes a block of F(x) 1's to its right, separated by at least one blank. Assume M_F has N states [19].

Consider a Turing machine M which, on input ʌ, first write x 1's on an initially blank tape, and then halts with its head reading the rightmost 1. This can be done with x states. Next, this TM mimics M_F, writing F(x) 1's to the right of initial block of x 1's, separated by at least one blank. This machine has x+2n states [19].

Now any Busy Beaver machine of x+2n states will leave at least as many 1's as M does on input of a block of x 1's.

$$\sum(x+2n) \geq x + F(x) + F(F(x))$$

But from its definition, $F(x) \geq x^2$, and there exist a constant c_1 such that $x^2 > x+2n$ for all $x \geq c_1$. It follows that $F(x) > x+2n$ for $x \geq c_1$. Now from its definition $F(x) > F(y)$ if $x > y$, so we have $F(F(x)) > F(x+2n)$ for $x \geq c_1$. It follows that

$$\sum(x+2n) \geq x + F(x) + F(F(x)) > F(F(x)) > F(x+2n) \geq f(x+2n)$$

For $x \geq c_1$; it follows that ∑ is eventually greater than f. since f was arbitrary; ∑ is non-computable [19].

Corollary 2 [19]**:** The function S (N) is also non-computable.

Proof: There exist a TM M with N states that writes ∑N 1's on its tape before halting. Such a TM make at least ∑N moves. Hence S(N) ≥ ∑N; Since ∑N is eventually greater than any computable function, so is S(N). Hence S(N) is also non-computable [19].

V. CONCLUSIONS

Busy Beaver is basically a problem of Turing machine. There are some functions, which are not Turing computable. Since T. Rado in 1962 defined the busy beaver, several approaches have used computer technology to search busy beaver or even compute some values of ∑ [9]. A lot of efforts are undertaken to calculate the values of non-computable Busy Beaver function [6]. In this paper the study was carried out which describes that the Busy Beaver function is of interest to researchers for plethora of reasons. For all these reasons, it is really quite fascinating to contemplate the successful efforts which have been made to calculate some of the initial values of ∑N [6]. Several researchers attempted to find good solutions to the BB problem using a broad variety of techniques [2]. But ∑ (Rado's function) is non-computable function [19].

REFERENCES

[1] J. J. Joosten, F. Toscano and H. Zenil "*Program-size versus Time complexity Slowdown and speed-up phenomena in the micro-cosmos of small Turing machines*," 16 April 2011.

[2] F. B. Pereira, P. Machado, E. Costa, A. Cardoso, "*Busy Beaver – An Evolutionary Approach*"

[3] Q. Gao and X. Xinhe "*The analysis and research on computational complexity*," Control and Decision Conference, The 26th Chinese, IEEE 2014.

[4] C. Diem, "*On the complexity of some computational problems in the Turing model,*" Preprint, November 18, 2013.

[5] C. S. Calude, Michael A. Stay, "*Most programs stop quickly or never halt*," Advances in Applied Mathematics 40, 2008.

[6] G. J. Chaitin, "*Computing the Busy beaver function*," In T. M. Cover and B. Gopinath, Open Problems in Communication and Computation, Springer, pp. 108–112,1987.

[7] T. Neary, D. Wood, "*Small fast universal Turing machines,"* Theoretical Computer Science 362, 1 June 2006.

[8] P. Michel, "*Small Turing machines and generalized busy beaver competition*," Theoretical Computer 326, 18 May 2004.

[9] M. Heiner, S. Gmbh, "*Attacking the busy beaver*," Bull EATCS. 1990.

[10] W. Damien, and T. Neary, "*The complexity of small universal Turing machines: A survey,"* Theoretical Computer Science 410.4, 2009.

[11] F. B Pereira, et al. "*Graph based crossover–a case study with the busy beaver problem,*" Proceedings of the 1999 Genetic and Evolutionary Computation Conference, 1999.

[12] Ed Blakey, "*Computational Complexity in Non-Turing Models of Computation: The What, the Why and the How*," Electronic Notes in Theoretical Computer Science 270.1, 2011.

[13] T. Worsch, "*Parallel Turing machines with one head control units and cellular automata,*" Theoretical computer science 217.1,1999.

[14] J. Terry, and G. JE Rawlins, "*Reverse Hill climbing Genetic Algorithms and the Busy Beaver Problem*," ICGA, 1993.

[15] T. Rado, "*On non-computable functions*," The Bell System Technical Journal, vol. 41, no. 3, pp.877-884, 1962.

[16] A.H. Brady, "*The conjectured highest scoring machines for Rado's S (k) for the value k=4,"* IEEE Transactions on Electronic Computers, vol. EC-15, pp. 802-803, October 1966.

[17] A.H. Brady, "*Solution of the non-computable Busy Beaver game for k=4*," Abstracts for ACM Computer Science Conference Washington, DC, p. 27, ACM, February 18-20, 1975.

[18] K.L.P. Mishra, N. Chandrasekaran, "*Theory of computer science*," Third edition.

[19] University of Waterloo, "*Introduction to the theory of computing – Handout on the Busy Beaver problem*," Winter, 1998.

[20] K. Dewdney, "*A computer trap for the busy beaver, the hardest-working Turing machine*," Computer Recreations Dept., Scientific American 251, No. 2, Aug, 1984.