Theoretical Computer Science 646 (2016) 61–85

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Busy beaver machines and the observant otter heuristic (or how to tame dreadful dragons)

James Harland

School of Computer Science and Information Technology, RMIT University, GPO Box 2476, Melbourne, 3001, Australia

ARTICLE INFO

Article history:
Received 27 February 2015
Received in revised form 19 April 2016
Accepted 11 July 2016
Available online 26 July 2016
Communicated by O.H. Ibarra

Keywords:
Busy beaver
Turing machines
Placid platypus

ABSTRACT

The busy beaver is a well-known specific example of a non-computable function. Whilst many aspects of this problem have been investigated, it is not always easy to find thorough and convincing evidence for the claims made about the maximality of particular machines, and the phenomenal size of some of the numbers involved means that it is not obvious that the problem can be feasibly addressed at all. In this paper we address both of these issues. We discuss a framework in which the busy beaver problem and similar problems may be addressed, and the appropriate processes for providing evidence of claims made. We also show how a simple heuristic, which we call the *observant otter*, can be used to evaluate machines with an extremely large number of execution steps required to terminate. We also show empirical results for an implementation of this heuristic which show how this heuristic is effective for all known 'monster' machines.

1. Introduction

The *busy beaver problem* has been an object of fascination since its introduction by Rado in 1962 as a specific example of a non-computable function [1]. The problem is to find the largest number of non-blank characters that are printed by a terminating Turing machine of no more than a given size on the blank input. Given the simplicity of the problem statement, it is often surprising to discover the extraordinarily large numbers of symbols that can be printed by some machines despite containing only a handful of states (see Table 1 below; the number of non-blank characters printed by a machine is known as its *productivity* [2]). It is counterintuitive, to say the least, to learn that a machine with only six states may terminate on the blank input after $10^{36,534}$ steps! The sheer size of this and similar numbers is not only motivation for an analysis of their behaviour, but also means that naive execution of such machines is hopelessly infeasible.

Whilst analyses of specific machines can reveal surprising and interesting properties [3,4], in order to determine the maximum value for a given size of machine, it is necessary to search through all such machines and record the maximum value found. In order to make such a result scientifically credible, this should not be simply an algorithmic search which produces a conclusion, but also sufficient evidence to allow the result to be checked or reproduced. Unfortunately, this is generally not the case for the known results for the busy beaver. For example, Lin and Rado [5] analyse the case for 3 states and 2 symbols by using a program to reduce the unknown cases (or 'holdouts') to 40, and "... these 40 holdouts were checked by hand". Whilst they provide a specification of the 40 holdout machines and an illustration of their method, the details provided are descriptive rather than comprehensive. Similarly Brady describes the determination of the busy beaver function for the case of 4 states and 2 symbols by a process of search through a total of around 500,000 machines,

E-mail address: james.harland@rmit.edu.au.

http://dx.doi.org/10.1016/j.tcs.2016.07.016
0304-3975/

Table 1
"Dreadful dragon" records for various classes of machines.

States	Symbols	States × Symbols	Non-blank characters	Hops
1	2	2	1	1
2	2	4	4	6
3	2	6	6	21
2	3	6	9	38
2	4	8	$\geq$ 2050	$\geq$ 3,932,964
4	2	8	13	107
3	3	9	$\geq$374,676,383	$\geq 1.12 * 10^{18}$
2	5	10	$\geq 1.7 * 10^{352}$	$\geq 1.9 * 10^{704}$
5	2	10	$\geq$4098	$\geq$47,176,870
2	6	12	$\geq 1.9 * 10^{4933}$	$\geq 2.4 * 0^{9866}$
3	4	12	$\geq 3.7 * 0^{6518}$	$\geq 5.2 * 10^{13036}$
4	3	12	$\geq 1.383 * 10^{7036}$	$\geq 1.025 * 10^{14072}$
6	2	12	$\geq 3.51 * 10^{18267}$	$\geq 7.41 * 10^{36534}$

from which there remained a total of 5,280 remained unclassified. Some specially designed programs further reduced the holdouts to a list of 218. Brady then states "Direct inspection of each of the 218 holdouts from these programs reveals that none will ever halt, …" ([6], p. 649), and uses this and other results to determine the busy beaver value. In other words, the remaining 218 cases were hand checked, but without any evidence provided. Machlin and Stout, who performed a similar analysis of the same class of machines, also use programs to significantly aid their search, until 210 holdouts remain [7]. However, similar comments apply to their statement that "The final 210 holdouts were examined by hand to verify that they were in infinite loops." ([7], p. 95). There also appears to be no record of the code or data files used by Lin and Rado, Brady or Machlin and Stout in these searches.

Perhaps the most comprehensive analysis of the busy beaver problem to date is that by Lafitte and Papazian [3]. They have analysed various instance of the busy beaver problem, including an enumeration of all machines with 2 states and 3 symbols, and those with 3 states and 2 symbols. They have also provided analyses of some of the larger cases (2 states 4 symbols, 4 states 2 symbols, 3 states 3 symbols), but in less detail, and have identified some problematic machines. This level of detail is a good start, but unfortunately falls short in terms of providing reproducable evidence, and as above, there appears to be no code available, nor any data files.

Similar remarks apply to the work of Wolfram [8,9]. His work is wider in scope, in that it takes in various properties of a number of types of automata, but the process he follows is comparable, in that it is based on searching through large numbers of machines looking for properties of interest. However, the ability to reproduce (and perhaps vary slightly) the process that has been undertaken seems difficult based on the information provided, especially as the busy beaver problem for Turing machines forms a small part of his overall work.

Whilst there is no reason whatsoever to doubt the veracity or sincerity of these results and others like them in the literature, it seems unscientific to accept such results as proven in the absence of both mathematical proof and empirical evidence that can be inspected, assessed and checked. This seems particularly true for claims about the maximality of specific machines, and hence particular values of the busy beaver function. In fairness, it should be said that much of the above work was done when computing resources were significantly more limited than today, and over the course of more than seventy years now there have been phenomenal improvements in processing power, storage capacity and network bandwidth which have made tasks that were once unthinkable into ones which are now routine. Those same improvements mean that now, in the era of cloud computing (in which data and computation can be easily distributed), it seems that the provision of such 'computational evidence' should be a minimum requirement for such claims. This evidence should include not only the programs used for the search, but also the list of machines generated as well as the evidence used to draw conclusions about their status. In other words, it is not sufficient just to determine the appropriate values; we must do so in such a way that provides verifiable and reproducable evidence for why this is the case. This seems particularly important now, when the famous proof of the Four Colour Theorem seems like ancient history [10], and the much more recent work of Hales et al. on the Flyspeck project to prove the Kepler conjecture represents the cutting edge of mathematical knowledge incorporating computation [11,12].

The nature of the process for determining busy beaver values has been discussed by de Mol [13]. Her aim is different to ours, in that she is interested in the cultural and philosophical aspects of the nature of computer-assisted proof in mathematics. De Mol discusses the issue of evidence in such cases, and the problem of *unsurveyability* (i.e. that for many computer-assisted proofs, it is impossible for humans to comprehend all details of the proof), and the importance of independent verification of results. From our perspective, the main conclusion is that computer-assisted proofs (which include the busy beaver problem and its variants) should include a *description of the computational process*, its *output* and the *code used* [13].

Hence it seems that there is a need to revisit our current level of knowledge of the busy beaver problem with a view to providing sufficient evidence for the results known. In particular, there is a need for an appropriate methodology in which the results are not only determined, but also justified in sufficient detail to be checked and/or reproduced by an independent researcher.