

**Manuscript version: Unpublished Version**

The version presented here is the unpublished version that may later be published elsewhere.

**Persistent WRAP URL:**

<http://wrap.warwick.ac.uk/168689>

**How to cite:**

Please refer to the repository item page, detailed above, for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

**Copyright and reuse:**

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

**Publisher's statement:**

Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: [wrap@warwick.ac.uk](mailto:wrap@warwick.ac.uk)

## SOLUTION TO A GENERALIZATION OF THE BUSY BEAVER PROBLEM\*

Let  $\varphi$  be a fixed numerical function. If the  $k$ -state Turing machine  $\mathbb{M}$  with input string  $\varphi(k)$  (that is, started in its initial state scanning the leftmost 1 of a single string of  $\varphi(k)$  1s on an otherwise blank tape) produces the output string  $m$  (that is, halts in its halting state scanning the leftmost 1 of a single string of  $m$  1s on an otherwise blank tape), we shall say that the  $\varphi$ -fecundity of  $\mathbb{M}$  is  $m$ . If  $\mathbb{M}$  halts in any other position or state, or fails to halt, its  $\varphi$ -fecundity is 0.

Since there are only finitely many  $k$ -state Turing machines, there is one that is at least as  $\varphi$ -fecund as any other. Let  $f_\varphi(k)$  be the  $\varphi$ -fecundity of the most  $\varphi$ -fecund  $k$ -state machine.

LEMMA 0: The 2-state machine  $\{A10B, B0RA\}$  deletes an input string of 1s of any length, and halts in state A scanning a 0. For  $k > 0$  the  $(k + 2)$ -state machine  $\mathbb{K}$

$$\begin{array}{cccc} A10B & B0RA & A01C_0 & C_001C_0 \\ C_01LC_1 & C_101C_1 & \dots & C_{k-1}01C_{k-1} \end{array}$$

replaces an input string of 1s of any length (in particular, of length  $\phi(k+2)$ ) by an output string of length  $k$ , and halts in state  $C_{k-1}$  scanning the leftmost 1.

PROOF: Exercise. ■

LEMMA 1: The 8-state machine below replaces an input string of  $k$  1s by an output string of  $2k$  1s, and halts in state A scanning the leftmost 1.

$$\begin{array}{ccccccc} A0RB & A10A & B0RC & B1RB & C01D & C1RC & D01E \\ D1RD & E0LF & E1LE & F0RG & F1LF & G0RJ & G10A \end{array}$$

PROOF: Exercise. ■

LEMMA 2: For all  $k > 10$

$$(1) \qquad f_\varphi(k) \geq 2k - 20.$$

PROOF: Use the two preceding lemmas. ■

\*This note, to be read in conjunction with Chapter 4.2 of G.S. Boolos, J.P. Burgess, & R.C. Jeffrey, *Computability & Logic* (4th edition, CUP 2002), is dedicated to the memory of Richard Jeffrey, who died on November 9, 2002. The proof of its main result, that the busy beaver function eventually dominates every total Turing computable function (THEOREM 3.2), was shown to me in 1989 by Richard Hill, who was following my *Symbolic Logic* course at the University of Warwick. The 8-state doubling machine of LEMMA 1 is due to Richard Schefer, who followed the course in 2000/2001. It is evident that a similar result can be obtained using the 12-state doubling machine (Example 3.2) on p.28 of *Computability & Logic* (p.24 of earlier editions). Note that machine states are here named by upper case roman letters.

**THEOREM 3:** Let  $h$  be any total, strictly increasing, Turing computable function. Then for all sufficiently large  $j$ ,  $f_\varphi(j) > h(j)$ .

**PROOF:** Let  $\mathbb{F}_\varphi$  be a machine with  $q_\varphi$  states that computes  $f_\varphi$ , and  $\mathbb{H}$  be a machine with  $r$  states that computes  $h$ . The composite machine  $\mathbb{K} + \mathbb{F}_\varphi + \mathbb{H}$  has  $j = k + 2 + q_\varphi + r$  states, and with any input (in particular  $\varphi(j)$ ) its output is  $h(f_\varphi(k))$ ; hence

$$(2) \quad f_\varphi(j) \geq h(f_\varphi(k)).$$

Now by elementary algebra, if  $k > q_\varphi + r + 22$  then  $2k - 20 > k + 2 + q_\varphi + r = j$ ; and therefore by (1), since  $k > 10$  whenever  $k > q_\varphi + r + 22$ ,

$$(3) \quad k > q_\varphi + r + 22 \Rightarrow f_\varphi(k) > j.$$

Since  $h$  is a strictly increasing function,

$$(4) \quad k > q_\varphi + r + 22 \Rightarrow h(f_\varphi(k)) > h(j).$$

Combining (2) and (4), we obtain:

$$(5) \quad k > q_\varphi + r + 22 \Rightarrow f_\varphi(j) > h(j).$$

In other words, since  $k = j - 2 - q_\varphi - r$ ,

$$(6) \quad j > 2q_\varphi + 2r + 24 \Rightarrow f_\varphi(j) > h(j),$$

which is what was to be proved. ■

**COROLLARY 0:** Let  $h$  be any total Turing computable function. Then for all sufficiently large  $j$ ,  $f_\varphi(j) > h(j)$ .

**PROOF:** If  $h$  is not strictly increasing, replace it by the strictly increasing function  $h'$  defined by

$$h'(j) = \max\{h(i) \mid i \leq j\} + 1.$$

Then  $h'(j) > h(j)$  for all  $j$ . The **THEOREM** tells us that for sufficiently large  $j$ ,  $f_\varphi(j) > h'(j)$ . It follows that for sufficiently large  $j$ ,  $f_\varphi(j) > h(j)$ .

**COROLLARY 1:** The function  $f_\varphi$  is not Turing computable.

**PROOF:** If  $f_\varphi$  is Turing computable then it is distinct from any Turing computable function. It follows that  $f_\varphi$  is not Turing computable. ■

**COROLLARY 2:** Neither the scoring function  $s$  (*Computability & Logic*, Proposition 4.3) nor the busy beaver function  $p$  (*op.cit.*, Theorem 4.7) is Turing computable.

**PROOF:** For the scoring function  $s$  take  $\varphi$  to be the identity function. For the busy beaver function  $p$  take  $\varphi$  to be the zero function. ■

David Miller  
January 9, 2003