University of Waterloo
CS 360 — Introduction to the Theory of Computing
Winter 1998
Handout on The Busy Beaver Problem

1 The Busy Beaver Problem

In this handout we describe a problem of Rado [5] on Turing machines now known as the *busy beaver problem*. We assume that our Turing machines are deterministic and have tape alphabet consisting of a single symbol $\{1\}$, and the usual blank symbol, Δ. (In Rado's original description, the symbol 0 was used as a blank.) We also assume our Turing machine has a single "doubly-infinite" tape, initially completely blank, and that the machine must move either right or left at each step – it cannot remain stationary. There is a single halting state from which no transitions emerge, and this halt state is not counted in the total number of states.

Rado's function $\Sigma(n)$ is defined to be the maximum number of (not necessarily consecutive) ones left on the tape after such an n-state Turing machine halts. He also defined a function $S(n)$ which counts the maximum number of moves that can be made by an n-state halting Turing machine of this form.

For example, consider the 3-state Turing machine below:

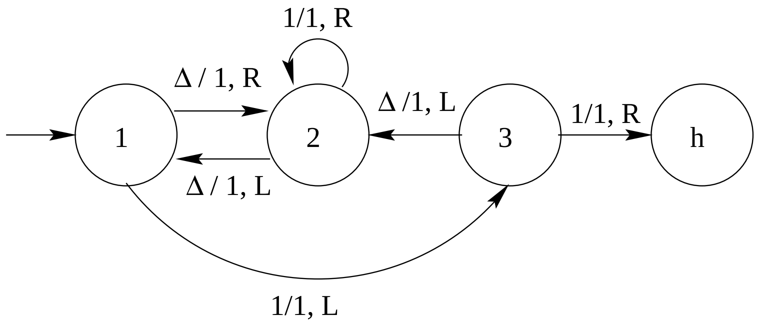

Figure 1: A 3-state busy beaver

If this machine is started with a tape of all blanks, it halts after 13 moves with six consecutive 1's on the tape – check this! This shows that $\Sigma(3) \geq 6$ and $S(3) \geq 13$. In fact, it can be shown [3] that $\Sigma(3) = 6$ and $S(3) = 21$.

If a Turing machine writes the maximum possible number of 1's for its number of states – i.e., $\Sigma(n)$ 1's – then it is called a "busy beaver". Busy beavers 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. (For each nonhalting state, there are two transitions out, so there are $2n$ total transitions, and each transition has 2 possibilities for the symbol being written, 2 possibilities for the direction to move – left or right, and $(n+1)$ possibilities for what state to go to – including the halting state.) Second, it is in general not possible to determine whether a particular Turing machine will halt, so it may not be easy to distinguish between a machine that goes into an infinite loop from one that goes for thousands, millions, or billions of steps before halting. For example, it is known (by explicitly producing an example) that $S(6) \geq 8 \cdot 10^{15}$. It is also known [2] that $\Sigma(8) \geq 10^{44}$.

In fact, we will show in a moment that neither $\Sigma(n)$ nor $S(n)$ are computable functions – this means that there is no halting Turing machine which, on arbitrary input n, will always halt and successfully compute these functions. Nevertheless, it is possible to compute $\Sigma(n)$ and $S(n)$ for some small values of n by a brute-force approach, and this has been done by many investigators.

The following table gives what is known about $\Sigma(n)$ and $S(n)$ for $1 \leq n \leq 6$:

n	$\Sigma(n)$	$S(n)$	Source
1	1	1	Lin and Rado [3]
2	4	6	Lin and Rado [3]
3	6	21	Lin and Rado [3]
4	13	107	Brady [1]
5	≥ 4098	$\geq 47,176,870$	Marxen and Buntrock [4]
6	$\geq 95,524,079$	$\geq 8,690,333,381,690,951$	Marxen [6]

Table 1: $\Sigma(n)$ and $S(n)$ for $1 \leq n \leq 6$

Here, courtesy of Heiner Marxen, is the 6-state Turing machine that makes 8,690,333,381,690,951 moves before halting with 95,524,079 1's on its tape.

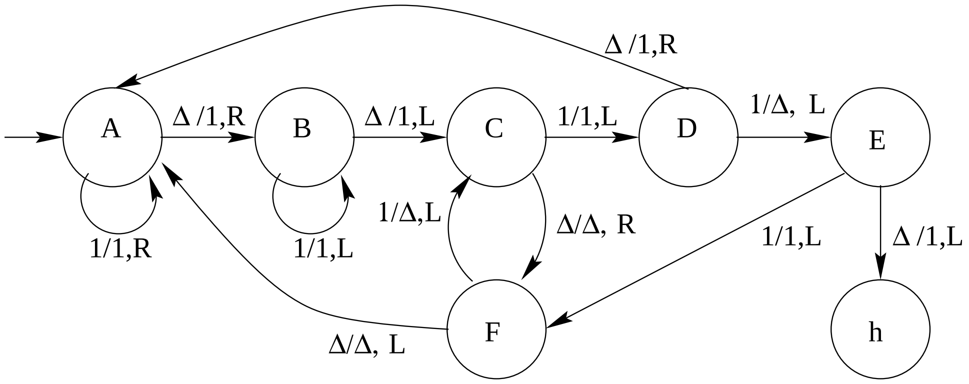

Figure 2: A 6-state busy beaver candidate

Now let's prove that neither $\Sigma(n)$ nor $S(n)$ are computable.

Theorem 1. The function $\Sigma(n)$ is not computable by a Turing machine.

Proof. The idea is to show that if $f(n)$ is any computable function, then there exists n_0 such that $\Sigma(n) > f(n)$ for $n \geq n_0$.

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 the starting blank, and halts after a finite number of moves with a block of $f(n)$ consecutive 1's on the tape.

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.

Consider a Turing machine M which, on input Λ, first writes 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 the initial block of x 1's, separated by at least one blank. Finally, this TM writes $F(F(x))$ 1's to the right of this last block of $F(x)$ 1's, separated by at least one blank. This machine has $x + 2n$ states.

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. Hence we have

$$\Sigma(x + 2n) \geq x + F(x) + F(F(x)).$$

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

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

for $x \geq c_1$. It follows that Σ is eventually greater than f. Since f was arbitrary, Σ is noncomputable. ■

Corollary 2. The function $S(n)$ is also noncomputable.

Proof. There exists a TM M with n states that writes $\Sigma(n)$ 1's on its tape before halting. Such a TM must make at least $\Sigma(n)$ moves. Hence $S(n) \geq \Sigma(n)$. Since $\Sigma(n)$ is eventually greater than any computable function, so is $S(n)$. Hence S is also noncomputable. ■

2 References

1. A. H. Brady, The determination of the value of Rado's noncomputable function $\Sigma(k)$ for four-state Turing machines, *Math. Comp.* **40** (1983), 647–665.

2. M. W. Green, A lower bound on Rado's Sigma function for binary Turing machines, *Proc. 5th Symposium on Switching Circuit Theory and Logical Design*, 1964, pp. 91-94.

3. S. Lin and T. Rado, Computer studies of Turing machine problems, *J. ACM* **12** (1965), 196–212.

4. H. Marxen and J. Buntrock, Attacking the busy beaver 5, *Bull. EATCS* **40** (February 1990), 247–251.

5. T. Rado, On non-computable functions, *Bell System Technical J.* **41** (1962), 877–884.

6. H. Marxen, personal communication, March 6, 1996. Correction, posting to Usenet newsgroup `comp.theory`, October 5, 1997.