A Note on Busy Beavers and Other Creatures

Amir M. Ben-Amram
Department of Computer Science
Tel Aviv University
Tel Aviv, Israel
amirben@math.tau.ac.il

Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
St. Cloud, Minnesota 56301 USA
julstrom@eeyore.stcloud.msus.edu

Uri Zwick
Department of Computer Science
Tel Aviv University
Tel Aviv, Israel
zwick@math.tau.ac.il

May 29, 2002

"The Beaver has counted with scrupulous care..."
L. Carrol, *The Hunting of the Snark*

Abstract

Consider Turing machines that read and write the symbols 1 and 0 on a one-dimensional tape that is infinite in both directions, and halt when started on a tape containing all 0s. Rado's *busy beaver* function $\text{ones}(n)$ is the maximum number of 1s such a machine, with n states, may leave on its tape when it halts. The function $\text{ones}(n)$ is noncomputable; in fact, it grows faster than any computable function.

Other functions with a similar nature can also be defined. The function $\text{time}(n)$ is the maximum number of moves such a machine may make before halting. The function $\text{num}(n)$ is the largest number of 1s such a machine may leave on its tape in the form of a single run; and the function $\text{space}(n)$ is the maximum number of tape squares such a machine may scan before it halts.

This paper establishes a variety of bounds on these functions in terms of each other; for example, $\text{time}(n) \leq (2n-1) \times \text{ones}(3n+3)$. In general, we aim at comparing the growth rates of such functions, and also discuss the problem of characterizing their growth behavior in a more precise way than that given by Rado.

1 Introduction

In 1962, Tibor Rado described a problem concerning Turing machines of n states (not counting the halt state) that conform to these restrictions:

- The tape is infinite in both directions.

- The tape alphabet is $\{0, 1\}$.
- The machine both writes and shifts on each move.
- The machine writes on the transition to the halt state.
- When started on a tape containing all 0s (called **blank**), the machine halts.

He asked the following question: what is the largest number of 1s such a machine may leave on its tape when it halts?

This is the **Busy Beaver problem**, and the maximum number of 1s a halting n-state Turing machine may leave when started on a blank tape, as a function of n, is the **Busy Beaver function** ones(n), sometimes called Rado's Sigma function (in the literature, this function has been denoted by $\Sigma(n)$; we find ones(n) a more meaningful notation). A machine of n states that, when started on a blank tape, halts with ones(n) 1s on its tape is an n-state **Busy Beaver**.

Similarly, we may define other functions based on Turing machines of this kind.

- The function time(n) is the maximum number of *moves* an n-state Turing machine, with the above restrictions, may make before it halts (the move to the halt state counts). In other words, it is the maximal time complexity of a halting machine. In the literature this function is frequently denoted by $S(n)$ and called the shift function. A machine of n states that executes time(n) moves and then halts is called an n-state time (or shift) champion.
- The function num(n) is defined as follows. Consider all Turing machines of n states that, when started on blank tapes, halt with a single run of consecutive 1s on the tape. Let num(n) be the length of the longest such run for machines of n states; that is, the largest unary number that can be created by an n-state Turing machine.
- The function space(n) is the maximal number of tape squares an n-state Turing machine can read before halting when started on a blank tape. This count includes the square on which a machine starts, but not a square reached only on the halt transition. This is the maximal space complexity of an n-state halting Turing machine. We call a machine of n states that reads a maximal number of squares an n-state space champion.

Rado showed that ones(n) grows faster than any computable function, and is therefore noncomputable [Rado 1962; see also Julstrom 1993]. Though we cannot evaluate any of these functions in general, methodical examinations of Turing machines of n states have established ones(n) and time(n) for small values of n. In particular, $\mathrm{ones}(1) = 1$, $\mathrm{time}(1) = 1$, $\mathrm{ones}(2) = 4$, and $\mathrm{time}(2) = 6$ [Rado 1962]; $\mathrm{ones}(3) = 6$ and $\mathrm{time}(3) = 21$ [Lin and Rado 1965]; $\mathrm{ones}(4) = 13$ and $\mathrm{time}(4) = 107$ [Brady 1983]. Marxen and Buntrock reported a 5-state Turing machine that takes 47,176,870 moves to leave 4,098 1s on an initially blank tape [Marxen and Buntrock 1990]. This machine is currently a candidate for both the busy beaver and time function 5-state championships; it establishes that $\mathrm{ones}(5) \geq 4,098$ and $\mathrm{time}(5) \geq 47,176,870$. Marxen and Buntrock also described a 6-state Turing machine that takes 13,122,572,797 moves to leave 136,612 1s on an initially blank tape. Thus, $\mathrm{ones}(6) \geq 136,612$ and $\mathrm{time}(6) \geq 13,122,572,797$. We have found that $\mathrm{num}(1) = 1$, $\mathrm{space}(1) = 1$; $\mathrm{num}(2) = 4$, $\mathrm{space}(2) = 4$; $\mathrm{num}(3) = 6$, $\mathrm{space}(3) = 7$; and that $\mathrm{num}(4) = 12$, $\mathrm{space}(4) = 16$. The last two results were obtained using a computer search that relied on the value of time(4) mentioned above.

It is straightforward to demonstrate that the functions num(n), ones(n), and time(n) are strictly increasing, that the function space(n) is non-decreasing, and that

$$\mathrm{num}(n) \leq \mathrm{ones}(n) \leq \mathrm{space}(n) \leq \mathrm{time}(n). \tag{1}$$

An immediate consequence of the relationships in (1) is that space(n) and time(n) also grow faster than any computable function and are therefore noncomputable (this also follows immediately from the undecidability of the halting problem). A consequence of several theorems in Section 3 is that the same is true of num(n).

More interesting and more difficult to establish are bounds on these functions in terms of each other in opposite direction to the relations in (1). For example, Rado [1962] observed that $\mathrm{time}(n) \leq (n+1) \times \mathrm{ones}(5n) \times 2^{\mathrm{ones}(5n)}$, and Julstrom [1992] showed that $\mathrm{time}(n) \leq \mathrm{ones}(20n)$. In Section 3 we prove the following results:

$$\begin{array}{lrcl}
\text{(i)} & \mathrm{space}(n) & \leq & \mathrm{ones}(3n-1) \\
\text{(ii)} & \mathrm{space}(n) & < & \frac{1}{2} \times \mathrm{num}(3n+3) \\
\text{(iii)} & \mathrm{space}(n) & \leq & \log_3 \mathrm{num}(3n+6) \\
\text{(iv)} & \mathrm{time}(n) & \leq & n \times \mathrm{space}(n) \times 2^{\mathrm{space}(n)} \\
\text{(v)} & \mathrm{time}(n) & < & (2n-1) \times \mathrm{ones}(3n+3) \\
\text{(vi)} & \mathrm{time}(n) & < & \mathrm{num}(3n+6) \\
\text{(vii)} & \mathrm{ones}(n) & < & \frac{1}{2} \times \mathrm{num}(3n+3)
\end{array} \tag{2}$$

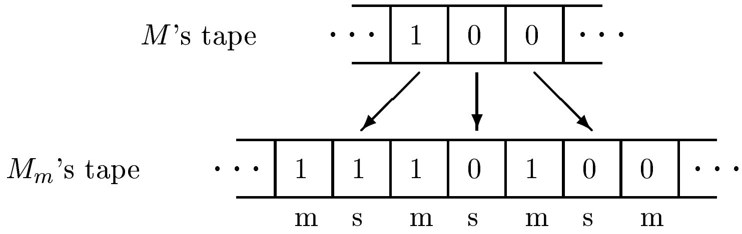
Note that (vi) immediately gives the bound $\mathrm{time}(n) < \mathrm{ones}(3n+6)$. Both this and (v) are bounds on time() in terms of ones(). Which one of these is better? It seems that the bound in (v) is better as it involves a smaller argument of the ones() function. However, we have not been able to prove that this is the case. Following the same reasoning, we expect that $\mathrm{ones}(n+1) > 2 \times \mathrm{ones}(n)$ for every sufficiently large n. Section 4 discusses this problem.

2 The Basic Construction

Almost all the bounds in (2) are based on simulating a given machine M while marking the extent of M's travels on its tape. We begin by describing this construction.

Let M be an n-state machine that halts when started on a blank tape. The machine M has $2n$ transitions (two emanate from every state), among which at least one is a halting transition. Recall that we do not include a halting state in our machine, but rather allow transitions whose "new state" component is replaced by "halt". Let $T(M)$ be the set of non-halting transitions in M; then $|T(M)| \leq 2n-1$. In fact, we can assume that $|T(M)| = 2n-1$, since there is no use in having more than a single halt transition (as there is no input, we deal only with a single computation path, which cannot include two halting transitions).

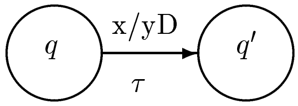
The machine that simulates M will be called M_m. It simulates M on alternate tape squares and uses the intervening squares as *marking squares*. The following figure illustrates the relationship between the squares of M_m's tape and the squares of M's tape. The letters s and m indicate *simulating* and *marking* squares respectively.



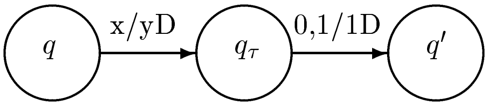
Whenever M_m's head moves from one simulating square to another, M_m writes a 1 in the intervening marking square. When the simulation terminates, there will be a 1 in every marking square that both of its neighbors were visited by M_m during the simulation.

Both machines start on blank tapes, and M_m starts with its head over a simulating square. M_m uses an additional state q_τ to simulate each transition $\tau \in T(M)$, as the following figure illustrates. The label "x/yD" indicates that on the transition the machine reads x, writes y, and moves one square in direction D.

in M:



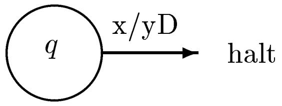
in M_m:



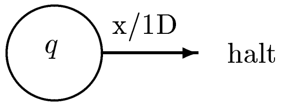
The machine M_m reads the symbol under its head, rewrites it as does the corresponding transition in M, and moves two squares in the direction that M moves, to the adjacent simulating square. *En passant*, it leaves a 1 in the intervening marking square.

No additional state is required to simulate M's halting transition, as the next diagram shows. We modify the halting transition of M_m, if necessary, to write a 1.

in M:



in M_m:



When M and M_m halt, M_m's simulating squares hold a copy of M's tape contents, except perhaps for the 1 M_m's halt transition left. All marking squares between these simulating squares contain 1s (and the rest are zero). The number of marking squares that contain 1s is therefore space$(M) - 1$, where space(M) represents the number of tape squares that M scanned.

The number of states in M_m is $n + |T(M)| = 3n - 1$.

3 Results

We begin by establishing a bound on space(n) in terms of the busy beaver function ones(n); this is a direct use of the above construction.

Theorem 1 space$(n) \leq$ ones$(3n - 1)$

Proof. Let M be an n-state space champion; that is, when started on a blank tape M scans space(n) tape squares and then halts. Construct the simulating machine M_m as above. M_m leaves at least space$(n) - 1$ 1s in the marking squares and at least a single 1 in the simulating squares, written by its halt transition. The machine M_m therefore leaves at least space(n) 1s on its tape. This number, however, cannot exceed the number of 1s left by a $(3n - 1)$-state busy beaver, so

$$\text{space}(n) \leq \text{ones}(3n - 1).$$

□

The next theorem establishes a bound on the function time(n) in terms of the busy beaver function. As in the proof of Theorem 1, we simulate a given Turing machine, but here we also count how many times a chosen transition is performed.

Theorem 2 time$(n) \leq (2n - 1)$ones$(3n + 3)$

Proof. Let M be an n-state time champion; that is, M is a Turing machine with n states that, when started on a blank tape, executes time(n) moves and then halts. For each non-halting transition $\tau \in T(M)$ of M, let $C_M(\tau)$ be the number of times this transition is performed when M starts on a blank tape; clearly

$$\text{time}(n) = 1 + \sum_{\tau \in T(M)} C_M(\tau)$$

where the 1 counts the execution of the halting transition. Since $|T(M)| = 2n - 1$, there must be a transition τ such that

$$C_M(\tau) \geq \frac{\text{time}(n) - 1}{2n - 1}.$$

We call such a transition *popular.*

For a given transition τ, we define a machine M_τ that simulates M in the way M_m did, but makes use of its marking squares both to mark the scanned portion of the tape and to count the number of times M executes the transition τ. More precisely, consider the *marked zone* on the tape; i.e., the range of marking squares set to 1. M_τ will add a 1-mark on the end of this zone each time M executes τ. Thus when M_τ halts, the number of marks will be greater than $C_M(\tau)$ (greater, because there are also "genuine" marks that indicate the scanned portion of the tape; e.g.,the mark M_τ sets when it simulates M's first transition.)

M_τ is defined exactly as M_m, with a new state simulating each transition, except for the transition τ, which we want to count. Assume that τ moves M's head to the right; that is, that τ is labeled x/yR in M. In M_τ, the following steps are taken to simulate and count this transition:

1. M_τ rewrites the symbol under its head according to the transition τ (that is, it overwrites x with y) and moves right so that its head is over a marking square.

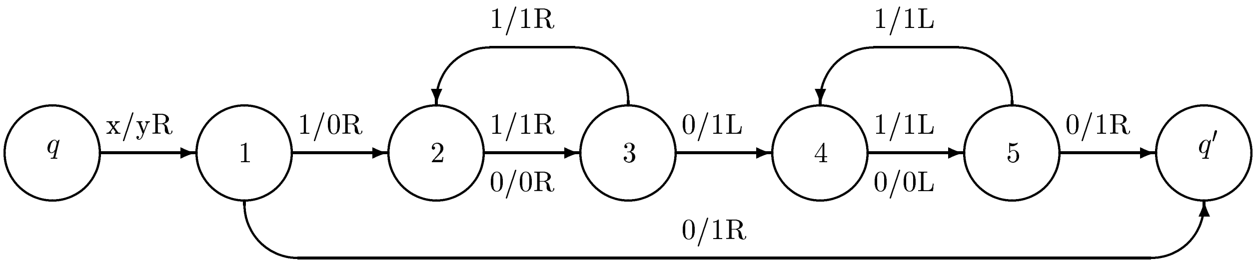
2. If the marking square is 0, M_τ sets it to 1 and shifts right, completing the simulation of the transition in M. If the marking square is already 1, M_τ sets it to 0; the head will be able to return to this position after updating the count by using the fact that this mark bit is 0 instead of 1.

3. M_τ shifts right without changing any symbols until it encounters a marking square that holds a 0.

4. M_τ overwrites this 0 with a 1, thus counting the simulated move, and shifts left.

5. M_τ shifts left to the marking square in which it wrote the 0, overwrites the 0 with a 1, and shifts right to a simulating square. This completes its simulation of τ.

This procedure requires five new states. The following figure shows the new states and transitions in M_τ.



If M shifts its head *left* on τ, M_τ counts the executions of this transition by adding 1s to the *left* in its marking squares; this too requires five new states, illustrated by the above figure with the move directions (L's and R's) reversed.

The construction of M_τ from M introduces one new state for each of the $2n-2$ transitions in $T(M) - \{\tau\}$. To represent τ and count its executions, M_τ uses five new states. Thus the total number of states in M_τ is:

$$n + (2n-2) + 5 = 3n + 3.$$

When M_τ halts, which it will because it simulates M and M halts, its simulating squares contain a copy of the contents M leaves on its tape; its marking squares contain 1s whose number N exceeds the number of executions of transition τ in M. Choose τ to be a popular transition; then

$$N \geq 1 + \frac{\mathrm{time}(n) - 1}{2n-1} > \frac{\mathrm{time}(n)}{2n-1}.$$

Since by definition $\mathrm{ones}(3n+3)$ bounds N, we have

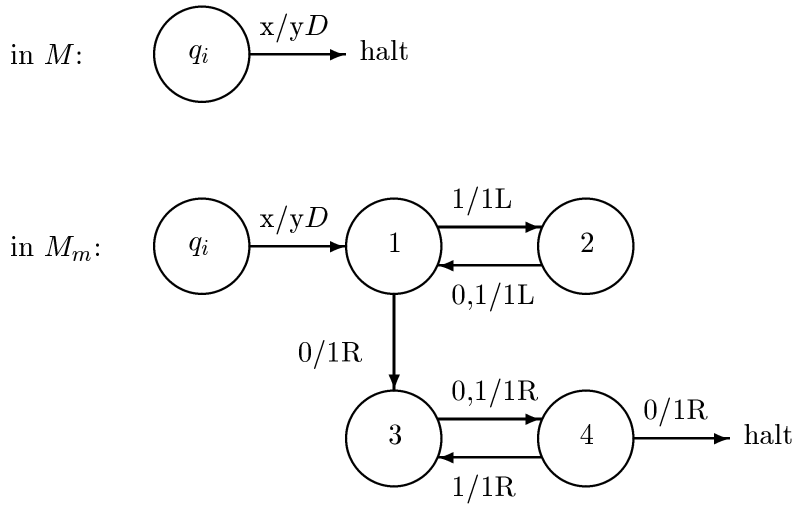
$$\mathrm{time}(n) < (2n-1) \times \mathrm{ones}(3n+3).$$

□

The next theorem extends the marking technique to establish a bound on $\mathrm{space}(n)$ in terms of $\mathrm{num}(n)$.

Theorem 3 $\quad \text{space}(n) < \frac{1}{2} \times \text{num}(3n+3)$

Proof. Let M be an n-state space champion. Build M_m in the standard way: marking squares alternate with simulating squares on which M_m simulates M. Augment this construction with four new states, as the following figure illustrates. Note that the resulting machine has $(3n+3)$ states.



On simulating M's halt transition, new states 1 and 2 scan to the left and write 1s in all squares until they encounter a 0 in a marking square. States 3 and 4 then scan to the right filling all squares with 1s until they too encounter a 0 in a marking square.

The 1s left on M_m's tape thus form an unbroken run, whose length is $2 \times \text{space}(n) + 1$. The number of 1s in this block cannot exceed $\text{num}(3n+3)$, so

$$2 \times \text{space}(n) + 1 \leq \text{num}(3n+3),$$

and the theorem follows. □

Since $\text{ones}(n) \leq \text{space}(n)$, a corollary of this result is

Corollary $\quad \text{ones}(n) < \text{num}(3n+3)/2$

The function $\text{time}(n)$ can be bounded in terms of $\text{num}(n)$ using a construction similar to that of Theorem 2. This is not our best bound, but we give it first for its simplicity.

Theorem 4 $\quad \text{time}(n) \leq n \times \text{num}(3n+7)$

Proof. Let M be an n-state time champion; that is, M executes $\text{time}(n)$ moves before it halts. As in Theorem 2, let τ be a popular transition of M and let M_τ be a machine that simulates M while counting the number of times the transition τ is made. Recall that M_τ leaves in its marking squares a block of 1s whose number exceeds the number of times the transition τ has been made. We add to M_τ four more states as in the proof of Theorem 3. These states fill the entire marked section of M_τ's tape with 1s. This creates a unary number greater or equal to $2C_M(\tau) + 3$. The number of states in M_τ is $(3n+7)$, so that this block of 1s cannot be longer than $\text{num}(3n+7)$. As τ is popular we have $C_M(\tau) \geq \frac{\text{time}(n)-1}{2n-1}$. Thus

$$\text{num}(3n+7) \geq 2 \times \frac{\text{time}(n)-1}{2n-1} + 3$$

and the result follows. □

We proceed by establishing two intermediate results that will serve the better bound on time(n) in terms of num(n). The first one gives a bound on time(n) in terms of space(n), using a standard configuration-counting argument and the pigeon-hole principle.

Theorem 5 $\quad \text{time}(n) \leq n \times \text{space}(n) \times 2^{\text{space}(n)}$

Proof. Let M be an n-state time champion. Before it halts, M scans a number of squares on its tape that is less than or equal to space(n). Therefore the number of distinct instantaneous descriptions of M's computation is bounded above by $n \times \text{space}(n) \times 2^{\text{space}(n)}$: M may be in one of n states with its head over one of space(n) tape squares, and $2^{\text{space}(n)}$ patterns of 0s and 1s may appear on those squares. If M repeats an instantaneous description it cannot halt; since M does halt,

$$\text{time}(n) \leq n \times \text{space}(n) \times 2^{\text{space}(n)}.$$

□

The next result gives a bound on $3^{\text{space}(n)}$ in terms of num(n). Besides yielding a new relation between space(n) and num(n), a bound on $3^{\text{space}(n)}$ will be useful in combination with the last theorem.

Theorem 6 $\quad 3^{\text{space}(n)} \leq \text{num}(3n+6)$

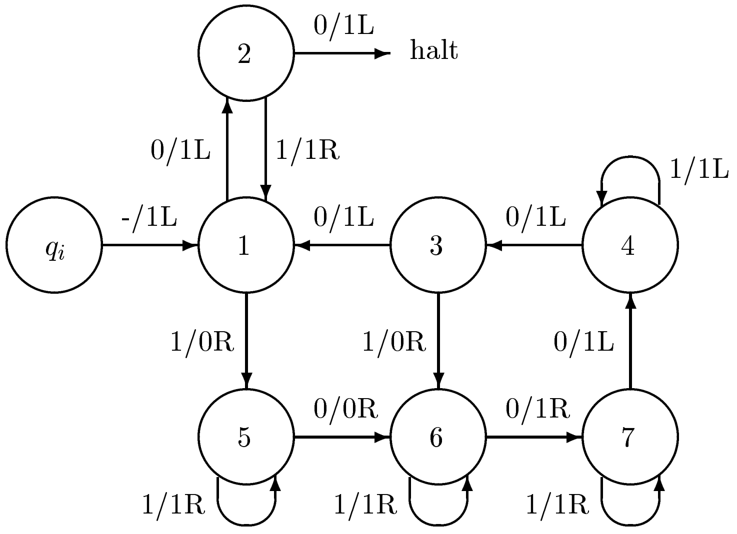
Proof. Let M be an n-state space champion. Build M_m as usual; it has $3n-1$ states.

Note that when M halts, the size of the entire marked zone (simulating squares and marking squares between them) is $2 \times \text{space}(n) - 1$. Thus on at least one side of the head's position there will be at least $\text{space}(n) - 1$ squares of the marked zone (not counting the square the head is on). Without loss of generality, we assume that side to be the left.

We augment M_m with seven additional states, in place of M's halt transition, for a total of $3n+6$ states. These states treat the part of the marked zone to the left of the head, at the moment M halts, as a unary value m (in fact, this area may include some 0s in simulating squares. Our machine will overcome this, and behave as though we really had a run of 1s). The new states, which the figure below illustrates, compute the value 3^{m+1}. The original halt transition from a state q_i is replaced by writing a 1, shifting left, and moving into the first of the new states.

To understand the operation of these states, assume at first that on the right of the head position, the tape is blank; we remove this assumption later. The unary value m to the head's left serves as a counter. To its right is written a value r, set initially to 1 by the transition that enters the new states.

The new states apply the function $r \to 3r+4$ m times, decrementing the value m by one each time. Starting with $r=1$, this yields a value of $3^{m+1}-2$ (this can be easily proved by induction). When the end of m is detected, the machine adds two more 1s for a final value of 3^{m+1}.



Some comments on the machine: the main loop ($r \to 3r + 4$) starts at state 1, with the head scanning the rightmost square of m. States 1 and 2 test if m has become zero; two states are needed to take care of 0s in simulating squares. If $m > 0$, we decrement it by changing its rightmost 1 to a 0 (transition to state 5), thus creating a separation between m and r. We now move right over r (self-loop at state 5) and enter the loop 6–7–4–3 that exits to state 1 after replacing r with 3(r+1)+1 = 3r+4 (the last 1 is added in the transition to state 1). The main loop then starts again, with the head correctly positioned on the rightmost square of m, left of the current representation of r.

To remove the assumption that the tape contains only 0s to the right of the head, note that the presence of some 1s on that part of the tape can only increase the final result, as these 1s are absorbed into r. Thus the final result will be *at least* 3^{m+1}.

Since the initial value of m is at least $\mathrm{space}(n) - 1$, the number of 1s the augmented machine leaves on the tape is at least $3^{\mathrm{space}(n)-1+1} = 3^{\mathrm{space}(n)}$, and these form a single contiguous run. The machine has $3n + 6$ states; therefore, by the definition of $\mathrm{num}(n)$,

$$3^{\mathrm{space}(n)} \leq \mathrm{num}(3n + 6).$$

□

Theorem 7 $\mathrm{time}(n) < \mathrm{num}(3n + 6)$

Proof. By Theorem 5,

$$\mathrm{time}(n) \leq n \times \mathrm{space}(n) \times 2^{\mathrm{space}(n)}.$$

For $n \geq 4$, we know that $m = \mathrm{space}(n) \geq 16$. Since $m^2 2^m < 3^m$ for every $m \geq 13$, for such n,

$$n \times \mathrm{space}(n) \times 2^{\mathrm{space}(n)} < 3^{\mathrm{space}(n)}.$$

By Theorem 6,

$$3^{\mathrm{space}(n)} < \mathrm{num}(3n + 6).$$

Consequently, for $n \geq 4$

$$\text{time}(n) < \text{num}(3n + 6).$$

For $n \leq 3$ the result is easily established directly. □

4 Growth Properties

The relationships established among the functions ones(n), num(n), time(n), and space(n) raise the question of comparing expressions involving those functions. For example, $(2n - 1) \times \text{ones}(3n + 3)$ and ones$(3n + 6)$ are both upper bounds on time(n). Can it be established that the first is lower for all (sufficiently large) n? Since ones(n) grows faster than any computable function [Rado 1962], we expect ones$(3n + 6)$ to be generally much larger than the former bound. In fact, we expect ones$(m + 1)$ to be much larger than ones(m) for all sufficiently large m. Rado's result implies, however, only a weaker property that appears below. The stronger property will be formulated as a conjecture.

Theorem 8 *For any computable function f,*

$$\text{ones}(n + 1) > f(\text{ones}(n))$$

for an infinite number of values of n.

Proof. Let f be a computable function. Let $F(n) = \max_{1 \leq i \leq n} f(i)$. Clearly F is non-decreasing and also computable. If there existed an N such that

$$\text{ones}(n + 1) \leq f(\text{ones}(n))$$

for all $n \geq N$, then for all $n > N$ we would have

$$\text{ones}(n) \leq F^{(n-N)}(\text{ones}(N)),$$

where $F^{(i)}$ denotes F iterated i times.

But ones(N) is a constant so $F^{(n-N)}(\text{ones}(N))$ is a computable function of n. This contradicts Rado's result: ones(n) cannot be bounded by a computable function. □

The property of ones(n) used in the proof is that of growing faster than any computable function. The same is true for the functions num(n), space(n), and time(n); so the result of Theorem 8 holds for them as well.

Conjecture *For any computable function f, there is a constant N_f such that*

$$\text{ones}(n + 1) > f(\text{ones}(n))$$

for all $n > N_f$.

We further conjecture that the same property also holds for the functions num(n), time(n) and space(n). Note that not every function that grows faster than any computable function has the property of the conjecture. Consider, for example, the function $\text{num}'(n) = \text{num}(\lfloor n/2 \rfloor)$. This function grows faster than any computable function, but for every even n, $\mathrm{num}'(n+1) = \mathrm{num}'(n)$. However, we do not expect any of ones(), num(), space(), and time() to be so ill-behaved.

A stronger conjecture would be that the relationship also holds across the four functions; e.g., that for every computable function f there exists a constant N_f such that for every $n > N_f$ we have

$$\mathrm{num}(n+1) > f(\mathrm{time}(n)).$$

Until the conjecture is proved, we content ourselves with a partial result.

Theorem 9 *For any computable function f there exists a constant c_f such that*

$$\mathrm{num}(n + c_f) \geq f(\mathrm{num}(n))$$

for all n.

Proof. Let f be a computable function. There exists a Turing machine M_f that computes f. When M_f is started on a tape that contains the unary number n, with its head positioned either to the left or inside this number, M_f will create the unary number $f(n)$ and halt. Let c_f be the number of states of M_f. Let n be given, and let M_n be an n-state Turing machine that, when started on a blank tape, halts leaving a block of $\mathrm{num}(n)$ 1s on the tape. Without loss of generality we may assume that M_n halts either inside or to the left of the number it left on the tape.

Let $M = M_m \rightarrow M_f$, where the arrow represents sequential composition of Turing machines. Then M has $n + c_f$ states and, started on a blank tape, it halts leaving a block of $f(\mathrm{num}(n))$ 1s. Of course, the unary number created by an $(n + c_f)$-state machine cannot exceed $\mathrm{num}(n + c_f)$. $\Box$

5 Conclusion

We have described four noncomputable functions related to the Busy Beaver problem and have established bounds on them in terms of each other. The proofs of these theorems illustrate a variety of constructive techniques. Other results motivate conjectures concerning relationships between expressions involving the four functions. These conjectures suggest further research into noncomputable functions and the Busy Beaver problem.

References

Brady, Allen H. (1983). The determination of the value of Rado's noncomputable function $\Sigma(k)$ for four-state Turing machines. *Mathematics of Computation*, Vol.40, No.162, pp.647-665.

Julstrom, Bryant A. (1992). A bound on the shift function in terms of the Busy Beaver function. *SIGACT News*, Vol.23, No.3 (Summer), 1992, pp.100-106.

Julstrom, Bryant A. (1993). Noncomputability and the Busy Beaver problem (UMAP Unit 728). *The UMAP Journal*, Vol.14, No.1, pp.39-74.

Lin, Shen and Tibor Rado (1965). Computer studies of Turing machine problems. *Journal of the Association for Computing Machinery*, Vol.12, pp.196-212.

Marxen, Heiner and Jürgen Buntrock (1990). Attacking the Busy Beaver Problem 5. *Bulletin of the European Association for Theoretical Computer Science*, Vol.40, pp.247-251.

Rado, Tibor (1962). On non-computable functions. *The Bell System Technical Journal.* Vol.41, pp.877-884.