# COMPUTING THE BUSY BEAVER FUNCTION

**In T. M. Cover and B. Gopinath, Open Problems in Communication and Computation, Springer, 1987, pp. 108–112**

**Gregory J. Chaitin**
*IBM Research Division, P.O. Box 218*
*Yorktown Heights, NY 10598, U.S.A.*

## Abstract

*Efforts to calculate values of the noncomputable Busy Beaver function are discussed in the light of algorithmic information theory.*

I would like to talk about some impossible problems that arise when one combines information theory with recursive function or computability theory. That is to say, I'd like to look at some unsolvable problems which arise when one examines computation unlimited by any practical

bound on running time, from the point of view of information theory. The result is what I like to call "algorithmic information theory" [5].

In the Computer Recreations department of a recent issue of *Scientific American* [7], A. K. Dewdney discusses efforts to calculate the Busy Beaver function $\Sigma$. This is a very interesting endeavor for a number of reasons.

First of all, 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. $\Sigma(n)$ is defined to be the largest number which can be computed by an $n$-state Turing machine; to information theorists it is clear that the correct measure is bits, not states. Thus it is more correct to define $\Sigma(n)$ 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 and fun problem to calculate values of $\Sigma$(number of states); to deal with $\Sigma$(number of bits) one would need a model of a binary computer as simple and compelling as the Turing machine model, and no obvious natural choice is at hand.

Perhaps the most fascinating aspect of Dewdney's discussion is that it describes successful attempts to calculate the initial values $\Sigma(1)$, $\Sigma(2)$, $\Sigma(3), \ldots$ of an uncomputable function $\Sigma$. Not only is $\Sigma$ uncomputable, but it grows faster than any computable function can. In fact, it is not difficult to see that $\Sigma(n)$ is greater than the computable function $f(n)$ as soon as $n$ is greater than (the program-size complexity or algorithmic information content of $f$) $+ O(1)$. Indeed, to compute $f(n) + 1$ it is sufficient to know (a minimum-size program for $f$), and the value of the integer ($n -$ the program-size complexity of $f$). Thus the program-size complexity of $f(n) + 1$ is $\leq$ (the program-size complexity of $f$) $+ O(\log |n -$ the program-size complexity of $f|)$, which is $< n$ if $n$ is greater than $O(1) +$the program-size complexity of $f$. Hence $f(n) + 1$ is included in $\Sigma(n)$, that is, $\Sigma(n) \geq f(n) + 1$, if $n$ is greater than $O(1) +$ the program-size complexity of $f$.

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 [2]. Turing's original proof involves the notion of a computable real number, and the observation that it cannot be decided whether or not the $n$th computer program ever outputs an $n$th digit, because otherwise one could carry out Cantor's diagonal construction and calculate a paradoxical real number whose $n$th digit is chosen to differ from the $n$th digit output by the $n$th program, and which therefore cannot actually be a computable real number after all. To use the noncomputability of $\Sigma$ to demonstrate the unsolvability of the halting problem, it suffices to note that in principle, if one were very patient, one could calculate $\Sigma(n)$ by checking each program of size less than or equal to $n$ to determine whether or not it halts, and then running each of the programs which halt to determine what their output is, and then taking the largest output. Contrariwise, if $\Sigma$ were computable, then it would provide a solution to the halting problem, for an $n$-bit program either halts in time less than $\Sigma(n + O(1))$, or else it never halts.

The Busy Beaver function is also of considerable metamathematical interest; in principle it would be extremely useful to know larger values of $\Sigma(n)$. For example, this would enable one to settle the Goldbach conjecture and the Riemann hypothesis, and in fact any conjecture such as Fermat's which can be refuted by a numerical counterexample. 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 first billion natural numbers. To convert this empirical approach into a proof, it would suffice to have a bound on how far it is necessary to test $P$ before settling the conjecture in the affirmative if no counterexample has been found, and of course rejecting it if one was discovered. $\Sigma$ provides this bound, for if $P$ has program-size complexity or algorithmic information content $k$, then it suffices to examine the first $\Sigma(k + O(1))$ natural numbers to decide whether or not $P$ is always true. Note that the program-size complexity or algorithmic information content of a famous conjecture $P$ is usually quite small; it is hard to get excited about a conjecture that takes a hundred pages to state.

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 $\Sigma(n)$. In a sense these efforts simultaneously penetrate to "mathematical bedrock" and are "storming the heavens," to use images of E. T. Bell. They amount to a systematic effort to settle all finitely refutable mathematical conjectures, that is, to determine all constructive mathematical truth. And these efforts fly in the face of fundamental information-theoretic limitations on the axiomatic method [1,2,6], which amount to an information-theoretic version of Gödel's famous incompleteness theorem [3].

Here is the Busy Beaver version of Gödel's incompleteness theorem: $n$ bits of axioms and rules of inference cannot enable one to prove what is the value of $\Sigma(k)$ for any $k$ greater than $n + O(1)$. The proof of this fact is along the lines of the Berry paradox. Contrariwise, there is an $n$-bit axiom which does enable one to demonstrate what is the value of $\Sigma(k)$ for any $k$ less than $n - O(1)$. To get such an axiom, one either asks God for the number of programs less than $n$ bits in size which halt, or one asks God for a specific $n$-bit program which halts and has the maximum possible running time or the maximum possible output before halting. Equivalently, the divine revelation is a conjecture $\forall k\, P(k)$ (with $P$ of program-size complexity or algorithmic information content $\leq n$) which is false and for which (the smallest counterexample $i$ with $\neg P(i)$) is as large as possible. Such an axiom would pack quite a wallop, but only in principle, because it would take about $\Sigma(n)$ steps to deduce from it whether or not a specific program halts and whether or not a specific mathematical conjecture is true for all natural numbers.

These considerations involving the Busy Beaver function are closely related to another fascinating noncomputable object, the halting probability of a universal Turing machine on random input, which I like to call $\Omega$, and which is the subject of an essay by my colleague Charles Bennett that was published in the Mathematical Games department of *Scientific American* some years ago [4].

# References

[1] G. J. Chaitin, "Randomness and mathematical proof," *Scientific American* **232**, No. 5 (May 1975), 47–52.

[2] M. Davis, "What is a computation?" in *Mathematics Today: Twelve Informal Essays,* L. A. Steen (ed.), Springer-Verlag, New York, 1978, 241–267.

[3] D. R. Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid,* Basic Books, New York, 1979.

[4] M. Gardner, "The random number Ω bids fair to hold the mysteries of the universe," Mathematical Games Dept., *Scientific American* **241**, No. 5 (Nov. 1979), 20–34.

[5] G. J. Chaitin, "Algorithmic information theory," in *Encyclopedia of Statistical Sciences,* Volume 1, Wiley, New York, 1982, 38–41.

[6] G. J. Chaitin, "Gödel's theorem and information," *International Journal of Theoretical Physics* **22** (1982), 941–954.

[7] A. K. Dewdney, "A computer trap for the busy beaver, the hardest-working Turing machine," Computer Recreations Dept., *Scientific American* **251**, No. 2 (Aug. 1984), 19–23.