# Who Can Name the Bigger Number?

Scott Aaronson[*]

1999

In an old joke, two noblemen vie to name the bigger number. The first, after ruminating for hours, triumphantly announces "Eighty-three!" The second, mightily impressed, replies "You win."

A biggest number contest is clearly pointless when the contestants take turns. But what if the contestants write down their numbers simultaneously, neither aware of the other's? To introduce a talk on "Big Numbers," I invite two audience volunteers to try exactly this. I tell them the rules:

> *You have fifteen seconds. Using standard math notation, English words, or both, name a single whole number–not an infinity–on a blank index card. Be precise enough for any reasonable modern mathematician to determine exactly what number you've named, by consulting only your card and, if necessary, the published literature.*

So contestants can't say "the number of sand grains in the Sahara," because sand drifts in and out of the Sahara regularly. Nor can they say "my opponent's number plus one," or "the biggest number anyone's ever thought of plus one"–again, these are ill-defined, given what our reasonable mathematician has available. Within the rules, the contestant who names the bigger number wins.

*Are you ready? Get set. Go.*

The contest's results are never quite what I'd hope. Once, a seventh-grade boy filled his card with a string of successive 9's. Like many other big-number tyros, he sought to maximize his number by stuffing a 9 into every place value. Had he chosen easy-to-write 1's rather than curvaceous 9's, his number could have been millions of times bigger. He still would been decimated, though, by the girl he was up against, who wrote a string of 9's followed by the superscript 999. Aha! An exponential: a number multiplied by itself 999 times. Noticing this innovation, I declared the girl's victory without bothering to count the 9's on the cards.

And yet the girl's number could have been much bigger still, had she stacked the mighty exponential more than once. Take $9^{9^9}$, for example. This behemoth, equal to $9^{387,420,489}$, has 369,693,100 digits. By comparison, the number of elementary particles in the observable universe has a meager 85 digits, give or take. Three 9's, when stacked exponentially, already lift us incomprehensibly

[*]Revised by Florian Mayer

beyond all the matter we can observe–by a factor of about 10369,693,015. And we've said nothing of $9^{9^{9^{9}}}$ or $9^{9^{9^{9^{9}}}}$.

Place value, exponentials, stacked exponentials: each can express boundlessly big numbers, and in this sense they're all equivalent. But the notational systems differ dramatically in the numbers they can express concisely. That's what the fifteen-second time limit illustrates. It takes the same amount of time to write 9999, $99^{99}$, and $9^{9^{9^{9}}}$ –yet the first number is quotidian, the second astronomical, and the third hyper-mega astronomical. The key to the biggest number contest is not swift penmanship, but rather a potent paradigm for concisely capturing the gargantuan.

Such paradigms are historical rarities. We find a flurry in antiquity, another flurry in the twentieth century, and nothing much in between. But when a new way to express big numbers concisely does emerge, it's often a byproduct of a major scientific revolution: systematized mathematics, formal logic, computer science. Revolutions this momentous, as any Kuhnian could tell you, only happen under the right social conditions. Thus is the story of big numbers a story of human progress.

And herein lies a parallel with another mathematical story. In his remarkable and underappreciated book *A History of p*, Petr Beckmann argues that the ratio of circumference to diameter is "a quaint little mirror of the history of man." In the rare societies where science and reason found refuge–the early Athens of Anaxagoras and Hippias, the Alexandria of Eratosthenes and Euclid, the seventeenth-century England of Newton and Wallis–mathematicians made tremendous strides in calculating p. In Rome and medieval Europe, by contrast, knowledge of p stagnated. Crude approximations such as the Babylonians' 25/8 and the Bible's 3 held sway.

This same pattern holds, I think, for big numbers. Curiosity and openness lead to fascination with big numbers, and to the buoyant view that no quantity, whether of the number of stars in the galaxy or the number of possible bridge hands, is too immense for the mind to enumerate. Conversely, ignorance and irrationality lead to fatalism concerning big numbers. The Bible, for example, refers twenty-one times to the supposed uncountability of sand. Take Genesis 32:12: "And thou saidst, I will surely do thee good, and make thy seed as the sand of the sea, which cannot be counted for multitude." Or Hebrews 11:12: "So many as the stars of the sky in multitude, and as the sand which is by the seashore innumerable." This notion that the multitude of sand grains might as well be infinite, that it's fit for dumbfounded stupefaction but not for quantification, is an ancient one. Historian Ilan Vardi cites the ancient Greek word 'yammkosioi,' or *sand-hundred*, colloquially meaning *zillion*; as well as a passage from Pindar's *Olympic Ode II* asserting that *"sand escapes counting."* But sand doesn't escape counting, as Archimedes recognized in the third century B.C. Here's how he began The Sand-Reckoner, a sort of pop-science article addressed to the King of Syracuse:

> There are some ... who think that the number of the sand is infinite in multitude ... again there are some who, without regarding it as infinite, yet think that no number has been named which is great enough to exceed its multitude ... But I will try to show you

> [numbers that] exceed not only the number of the mass of sand equal in magnitude to the earth ... but also that of a mass equal in magnitude to the universe.

This Archimedes proceeded to do, essentially by using the ancient Greek term myriad, meaning ten thousand, as a base for exponentials. Adopting a prescient cosmological model of Aristarchus, in which the "sphere of the fixed stars" is vastly greater than the sphere in which the Earth revolves around the sun, Archimedes obtained an upper bound of $10^{63}$ on the number of sand grains needed to fill the universe. (Supposedly $10^{63}$ is the biggest number with a lexicographically standard American name: *vigintillion*. But the staid vigintillion had better keep vigil lest it be encroached upon by the more whimsically-named *googol*, or $10^{100}$, and *googolplex*, or $10^{10^{100}}$.) Vast though it was, of course, $10^{63}$ wasn't to be enshrined as the all-time biggest number. Six centuries later, Diophantus developed a simpler notation for exponentials, allowing him to surpass. Then, in the Middle Ages, the rise of Arabic numerals and place value made it easy to stack exponentials higher still. But Archimedes' paradigm for expressing big numbers wasn't fundamentally surpassed until the twentieth century. And even today, exponentials dominate popular discussion of the immense.

Consider, for example, the oft-repeated legend of the Grand Vizier in Persia who invented chess. The King, so the legend goes, was delighted with the new game, and invited the Vizier to name his own reward. The Vizier replied that, being a modest man, he desired only one grain of wheat on the first square of a chessboard, two grains on the second, four on the third, and so on, with twice as many grains on each square as on the last. The innumerate King agreed, not realizing that the total number of grains on all 64 squares would be $2^{64} - 1$, or 18.6 quintillion–equivalent to the world's present wheat production for 150 years.

Fittingly, this same exponential growth is what makes chess itself so difficult. There are only about 35 legal choices for each chess move, but the choices multiply exponentially to yield over $10^{50}$ possible board positions–too many for even a computer to search exhaustively. That's why it took until 1997 for a computer, Deep Blue, to defeat the human world chess champion. And in Go, which has a 19-by-19 board and over $10^{150}$ possible positions, even an amateur human can still rout the world's top-ranked computer programs. Exponential growth plagues computers in other guises as well. The traveling salesman problem asks for the shortest route connecting a set of cities, given the distances between each pair of cities. The rub is that the number of possible routes grows exponentially with the number of cities. When there are, say, a hundred cities, there are about $10^{158}$ possible routes, and, although various shortcuts are possible, no known computer algorithm is fundamentally better than checking each route one by one. The traveling salesman problem belongs to a class called NP-complete, which includes hundreds of other problems of practical interest. (NP stands for the technical term 'Nondeterministic Polynomial-Time.') It's known that if there's an efficient algorithm for any NP-complete problem, then there are efficient algorithms for all of them. Here 'efficient' means using an amount of time proportional to at most the problem size raised to some fixed power–for example, the number of cities cubed. It's conjectured, however, that no efficient algorithm for NP-complete problems exists. Proving this conjecture, called $P^1$

NP, has been a great unsolved problem of computer science for thirty years.

Although computers will probably never solve NP-complete problems efficiently, there's more hope for another grail of computer science: replicating human intelligence. The human brain has roughly a hundred billion neurons linked by a hundred trillion synapses. And though the function of an individual neuron is only partially understood, it's thought that each neuron fires electrical impulses according to relatively simple rules up to a thousand times each second. So what we have is a highly interconnected computer capable of maybe $10^{14}$ operations per second; by comparison, the world's fastest parallel supercomputer, the 9200-Pentium Pro teraflops machine at Sandia National Labs, can perform $10^{12}$ operations per second. Contrary to popular belief, gray mush is not only hard-wired for intelligence: it surpasses silicon even in raw computational power. But this is unlikely to remain true for long. The reason is Moore's Law, which, in its 1990's formulation, states that the amount of information storable on a silicon chip grows exponentially, doubling roughly once every two years. Moore's Law will eventually play out, as microchip components reach the atomic scale and conventional lithography falters. But radical new technologies, such as optical computers, DNA computers, or even quantum computers, could conceivably usurp silicon's place. Exponential growth in computing power can't continue forever, but it may continue long enough for computers–at least in processing power–to surpass human brains.

To prognosticators of artificial intelligence, Moore's Law is a glorious herald of exponential growth. But exponentials have a drearier side as well. The human population recently passed six billion and is doubling about once every forty years. At this exponential rate, if an average person weighs seventy kilograms, then by the year 3750 the entire Earth will be composed of human flesh. But before you invest in deodorant, realize that the population will stop increasing long before this–either because of famine, epidemic disease, global warming, mass species extinctions, unbreathable air, or, entering the speculative realm, birth control. It's not hard to fathom why physicist Albert Bartlett asserted "the greatest shortcoming of the human race" to be "our inability to understand the exponential function." Or why Carl Sagan advised us to "never underestimate an exponential." In his book *Billions & Billions*, Sagan gave some other depressing consequences of exponential growth. At an inflation rate of five percent a year, a dollar is worth only thirty-seven cents after twenty years. If a uranium nucleus emits two neutrons, both of which collide with other uranium nuclei, causing *them* to emit two neutrons, and so forth–well, did I mention nuclear holocaust as a possible end to population growth?

Exponentials are familiar, relevant, intimately connected to the physical world and to human hopes and fears. Using the notational systems I'll discuss next, we can concisely name numbers that make exponentials picayune by comparison, that subjectively speaking exceed $9^{9^{9}}$ as much as the latter exceeds 9. But these new systems may seem more abstruse than exponentials. In his essay "On Number Numbness," Douglas Hofstadter leads his readers to the precipice of these systems, but then avers:

> If we were to continue our discussion just one zillisecond longer, we would find ourselves smack-dab in the middle of the theory of

recursive functions and algorithmic complexity, and that would be too abstract. So let's drop the topic right here.

But to drop the topic is to forfeit, not only the biggest number contest, but any hope of understanding how stronger paradigms lead to vaster numbers. And so we arrive in the early twentieth century, when a school of mathematicians called the formalists sought to place all of mathematics on a rigorous axiomatic basis. A key question for the formalists was what the word 'computable' means. That is, how do we tell whether a sequence of numbers can be listed by a definite, mechanical procedure? Some mathematicians thought that 'computable' coincided with a technical notion called 'primitive recursive.' But in 1928 Wilhelm Ackermann disproved them by constructing a sequence of numbers that's clearly computable, yet grows too quickly to be primitive recursive.

Ackermann's idea was to create an endless procession of arithmetic operations, each more powerful than the last. First comes addition. Second comes multiplication, which we can think of as repeated addition: for example, $5 * 3$ means 5 added to itself 3 times, or $5 + 5 + 5 = 15$. Third comes exponentiation, which we can think of as repeated multiplication. Fourth comes ... what? Well, we have to invent a weird new operation, for repeated exponentiation. The mathematician Rudy Rucker calls it 'tetration.' For example, '5 tetrated to the 3' means 5 raised to its own power 3 times, or $5^{5^5}$, a number with 2,185 digits. We can go on. Fifth comes repeated tetration: shall we call it 'pentation'? Sixth comes repeated pentation: 'hexation'? The operations continue infinitely, with each one standing on its predecessor to peer even higher into the firmament of big numbers.

If each operation were a candy flavor, then the Ackermann sequence would be the sampler pack, mixing one number of each flavor. First in the sequence is $1 + 1$, or (don't hold your breath) 2. Second is $2 * 2$, or 4. Third is 3 raised to the 3rd power, or 27. Hey, these numbers aren't so big!

*Fee. Fi. Fo. Fum.*

Fourth is 4 tetrated to the 4, or $4^{4^{4^4}}$, which has $10^{154}$ digits. If you're planning to write this number out, better start now. Fifth is 5 pentated to the 5, or with '5 pentated to the 4' numerals in the stack. This number is too colossal to describe in any ordinary terms. And the numbers just get bigger from there.

Wielding the Ackermann sequence, we can clobber unschooled opponents in the biggest-number contest. But we need to be careful, since there are several definitions of the Ackermann sequence, not all identical. Under the fifteen-second time limit, here's what I might write to avoid ambiguity:

*A(111)–Ackermann seq–*$A(1) = 1 + 1, A(2) = 2 * 2, A(3) = 33$*, etc*

Recondite as it seems, the Ackermann sequence does have some applications. A problem in an area called Ramsey theory asks for the minimum dimension of a hypercube satisfying a certain property. The true dimension is thought to be 6, but the lowest dimension anyone's been able is prove is so huge that it can only be expressed using the same 'weird arithmetic' that underlies the Ackermann sequence. Indeed, the Guinness Book of World Records once listed this dimension as the biggest number ever used in a mathematical proof. (Another contender for the title once was Skewes' number, about $10^{10^{10^{10^{3}}}}$, which arises in the study of how prime numbers are distributed. The famous mathematician G. H. Hardy quipped that Skewes' was "the largest number which has ever served any definite purpose in mathematics.") What's more, Ackermann's briskly-rising cavalcade performs an occasional cameo in computer science. For example, in the analysis of a data structure called 'Union-Find,' a term gets multiplied by the inverse of the Ackermann sequence–meaning, for each whole number X, the first number N such that the N$^{\text{th}}$ Ackermann number is bigger than X. The inverse grows as slowly as Ackermann's original sequence grows quickly; for all practical purposes, the inverse is at most 4.

Ackermann numbers are pretty big, but they're not yet big enough. The quest for still bigger numbers takes us back to the formalists. After Ackermann demonstrated that 'primitive recursive' isn't what we mean by 'computable,' the question still stood: what *do* we mean by 'computable'? In 1936, Alonzo Church and Alan Turing independently answered this question. While Church answered using a logical formalism called the lambda calculus, Turing answered using an idealized computing machine–the Turing machine–that, in essence, is equivalent to every Compaq, Dell, Macintosh, and Cray in the modern world. Turing's paper describing his machine, "On Computable Numbers," is rightly celebrated as the founding document of computer science.

"Computing," said Turing,

> is normally done by writing certain symbols on paper. We may suppose this paper to be divided into squares like a child's arithmetic book. In elementary arithmetic the 2-dimensional character of the paper is sometimes used. But such use is always avoidable, and I think it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, on a tape divided into squares.

Turing continued to explicate his machine using ingenious reasoning from first principles. The tape, said Turing, extends infinitely in both directions, since a theoretical machine ought not be constrained by physical limits on resources. Furthermore, there's a symbol written on each square of the tape, like the '1's and '0's in a modern computer's memory. But how are the symbols manipulated? Well, there's a 'tape head' moving back and forth along the tape, examining one square at a time, writing and erasing symbols according to definite rules. The rules are the tape head's program: change them, and you change what the tape head does.

Turing's august insight was that we can program the tape head to carry out any computation. Turing machines can add, multiply, extract cube roots, sort, search, spell-check, parse, play Tic-Tac-Toe, list the Ackermann sequence. If we represented keyboard input, monitor output, and so forth as symbols on the tape, we could even run Windows on a Turing machine. But there's a problem. Set a tape head loose on a sequence of symbols, and it might stop eventually, or it might run forever–like the fabled programmer who gets stuck in the shower because the instructions on the shampoo bottle read "lather, rinse, repeat." If the machine's going to run forever, it'd be nice to know this in advance, so that we don't spend an eternity waiting for it to finish. But how can we determine, in a finite amount of time, whether something will go on endlessly? If you bet a friend that your watch will never stop ticking, when could you declare victory? But maybe there's some ingenious program that can examine other programs and tell us, infallibly, whether they'll ever stop running. We just haven't thought of it yet.

Nope. Turing proved that this problem, called the Halting Problem, is unsolvable by Turing machines. The proof is a beautiful example of self-reference. It formalizes an old argument about why you can never have perfect introspection: because if you could, then you could determine what you were going to do ten seconds from now, and then do something else. Turing imagined that there was a special machine that could solve the Halting Problem. Then he showed how we could have this machine analyze itself, in such a way that it has to halt if it runs forever, and run forever if it halts. Like a hound that finally catches its tail and devours itself, the mythical machine vanishes in a fury of contradiction. (That's the sort of thing you don't say in a research paper.)

"Very nice," you say (or perhaps you say, "not nice at all"). "But what does all this have to do with big numbers?" Aha! The connection wasn't published until May of 1962. Then, in the *Bell System Technical Journal*, nestled between pragmatically-minded papers on "Multiport Structures" and "Waveguide Pressure Seals," appeared the modestly titled "On Non-Computable Functions" by Tibor Rado. In this paper, Rado introduced the biggest numbers anyone had ever imagined.

His idea was simple. Just as we can classify words by how many letters they contain, we can classify Turing machines by how many rules they have in the tape head. Some machines have only one rule, others have two rules, still others have three rules, and so on. But for each fixed whole number N, just as there are only finitely many words with N letters, so too are there only finitely many machines with N rules. Among these machines, some halt and others run forever when started on a blank tape. Of the ones that halt, asked Rado, what's the maximum number of steps that any machine takes *before* it halts? (Actually, Rado asked mainly about the maximum number of symbols any machine can write on the tape before halting. But the maximum number of steps, which Rado called S(n), has the same basic properties and is easier to reason about.)

Rado called this maximum the N$^{\text{th}}$ "Busy Beaver" number. (Ah yes, the early 1960's were a more innocent age.) He visualized each Turing machine as a beaver bustling busily along the tape, writing and erasing symbols. The challenge, then, is to find the busiest beaver with exactly N rules, albeit not an infinitely busy one. We can interpret this challenge as one of finding the "most complicated" computer program N bits long: the one that does the most amount of stuff, but not an infinite amount.

Now, suppose we knew the N$^{\text{th}}$ Busy Beaver number, which we'll call BB(N). Then we could decide whether any Turing machine with N rules halts on a blank tape. We'd just have to run the machine: if it halts, fine; but if it doesn't halt within BB(N) steps, then we know it never *will* halt, since BB(N) is the maximum number of steps it could make before halting. Similarly, if you knew that all mortals died before age 200, then if Sally lived to be 200, you could conclude that Sally was immortal. So no Turing machine can list the Busy Beaver numbers–for if it could, it could solve the Halting Problem, which we already know is impossible.

But here's a curious fact. Suppose we could name a number *greater* than the N$^{\text{th}}$ Busy Beaver number $BB(N)$. Call this number D for dam, since like a beaver dam, it's a roof for the Busy Beaver below. With D in hand, computing $BB(N)$ itself becomes easy: we just need to simulate all the Turing machines with N rules. The ones that haven't halted within D steps–the ones that bash through the dam's roof–never will halt. So we can list exactly which machines halt, and among these, the maximum number of steps that any machine takes before it halts is $BB(N)$.

Conclusion? The sequence of Busy Beaver numbers, $BB(1)$, $BB(2)$, and so on, grows faster than any computable sequence. Faster than exponentials, stacked exponentials, the Ackermann sequence, you name it. Because if a Turing machine could compute a sequence that grows faster than Busy Beaver, then it could use that sequence to obtain the D's–the beaver dams. And with those D's, it could list the Busy Beaver numbers, which (sound familiar?) we already know is impossible. The Busy Beaver sequence is non-computable, solely because it grows stupendously fast–too fast for any computer to keep up with it, even in principle.

This means that no computer program could list all the Busy Beavers one by one. It doesn't mean that specific Busy Beavers need remain eternally unknowable. And in fact, pinning them down has been a computer science pastime ever since Rado published his article. It's easy to verify that BB(1), the first Busy Beaver number, is 1. That's because if a one-rule Turing machine doesn't halt after the very first step, it'll just keep moving along the tape endlessly. There's no room for any more complex behavior. With two rules we can do more, and a little grunt work will ascertain that BB(2) is 6. Six steps. What about the third Busy Beaver? In 1965 Rado, together with Shen Lin, proved that BB(3) is 21. The task was an arduous one, requiring human analysis of many machines to prove that they don't halt–since, remember, there's no algorithm for listing the Busy Beaver numbers. Next, in 1983, Allan Brady proved that BB(4) is 107. Unimpressed so far? Well, as with the Ackermann sequence, don't be fooled by the first few numbers.

In 1984, A.K. Dewdney devoted a *Scientific American* column to Busy Beavers, which inspired amateur mathematician George Uhing to build a special-purpose device for simulating Turing machines. The device, which cost Uhing less than $100, found a five-rule machine that runs for 2,133,492 steps before halting–establishing that BB(5) must be at least as high. Then, in 1989, Heiner Marxen and Jürgen Buntrock discovered that $BB(5)$ is at least 47,176,870. To this day, BB(5) hasn't been pinned down precisely, and it could turn out to be much higher still. As for $BB(6)$, Marxen and Buntrock set another record in 1997 by proving that it's at least 8,690,333,381,690,951. A formidable accomplishment, yet Marxen, Buntrock, and the other Busy Beaver hunters are merely wading along the shores of the unknowable. Humanity may never know the value of $BB(6)$ for certain, let alone that of BB(7) or any higher number in the sequence.

Indeed, already the top five and six-rule contenders elude us: we can't explain how they 'work' in human terms. If creativity imbues their design, it's not because humans put it there. One way to understand this is that even small Turing machines can encode profound mathematical problems. Take Goldbach's conjecture, that every even number 4 or higher is a sum of two prime numbers: $10 = 7 + 3$, $18 = 13 + 5$. The conjecture has resisted proof since 1742. Yet we could design a Turing machine with, oh, let's say 100 rules, that tests each even number to see whether it's a sum of two primes, and halts when and if it finds a counterexample to the conjecture. Then knowing $BB(100)$, we could in principle run this machine for $BB(100)$ steps, decide whether it halts, and thereby resolve Goldbach's conjecture. We need not venture far in the sequence to enter the lair of basilisks.

But as Rado stressed, even if we can't list the Busy Beaver numbers, they're perfectly well-defined mathematically. If you ever challenge a friend to the biggest number contest, I suggest you write something like this:

*BB(11111)–Busy Beaver shift #–1, 6, 21, etc*

If your friend doesn't know about Turing machines or anything similar, but only about, say, Ackermann numbers, then you'll win the contest. You'll still win even if you grant your friend a handicap, and allow him the entire lifetime of the universe to write his number. The key to the biggest number contest is a potent paradigm, and Turing's theory of computation is potent indeed.

But what if your friend knows about Turing machines as well? Is there a notational system for big numbers more powerful than even Busy Beavers?

Suppose we could endow a Turing machine with a magical ability to solve the Halting Problem. What would we get? We'd get a 'super Turing machine': one with abilities beyond those of any ordinary machine. But now, how hard is it to decide whether a *super* machine halts? Hmm. It turns out that not even super machines can solve this 'super Halting Problem', for the same reason that ordinary machines can't solve the ordinary Halting Problem. To solve the Halting Problem for super machines, we'd need an even *more* powerful machine: a 'super duper machine.' And to solve the Halting Problem for super duper machines, we'd need a 'super duper pooper machine.' And so on endlessly. This infinite hierarchy of ever more powerful machines was formalized by the logician Stephen Kleene in 1943 (although he didn't use the term 'super duper pooper').

Imagine a novel, which is imbedded in a longer novel, which itself is imbedded in an even *longer* novel, and so on ad infinitum. Within each novel, the characters can debate the literary merits of any of the sub-novels. But, by analogy with classes of machines that can't analyze themselves, the characters can never critique the novel that they *themselves* are in. (This, I think, jibes with our ordinary experience of novels.) To fully understand some reality, we need to go outside of that reality. This is the essence of Kleene's hierarchy: that to solve the Halting Problem for some class of machines, we need a yet more powerful class of machines.

And there's no escape. Suppose a Turing machine had a magical ability to solve the Halting Problem, *and* the super Halting Problem, *and* the super duper Halting Problem, *and* the super duper pooper Halting Problem, and so on endlessly. Surely this would be the Queen of Turing machines? Not quite.

As soon as we want to decide whether a 'Queen of Turing machines' halts, we need a still more powerful machine: an 'Empress of Turing machines.' And Kleene's hierarchy continues.

But how's this relevant to big numbers? Well, each level of Kleene's hierarchy generates a faster-growing Busy Beaver sequence than do all the previous levels. Indeed, each level's sequence grows so rapidly that it can only be computed by a higher level. For example, define $BB_2(N)$ to be the maximum number of steps a super machine with N rules can make before halting. If this super Busy Beaver sequence were computable by super machines, then those machines could solve the super Halting Problem, which we know is impossible. So the super Busy Beaver numbers grow too rapidly to be computed, even if we could compute the ordinary Busy Beaver numbers.

You might think that now, in the biggest-number contest, you could obliterate even an opponent who uses the Busy Beaver sequence by writing something like this:

*$BB_2(11111)$.*

But not quite. The problem is that I've never seen these "higher-level Busy Beavers" defined anywhere, probably because, to people who know computability theory, they're a fairly obvious extension of the ordinary Busy Beaver numbers. So our reasonable modern mathematician wouldn't know what number you were naming. If you want to use higher-level Busy Beavers in the biggest number contest, here's what I suggest. First, publish a paper formalizing the concept in some obscure, low-prestige journal. Then, during the contest, cite the paper on your index card.

To exceed higher-level Busy Beavers, we'd presumably need some new computational model surpassing even Turing machines. I can't imagine what such a model would look like. Yet somehow I doubt that the story of notational systems for big numbers is over. Perhaps someday humans will be able concisely to name numbers that make Busy Beaver 100 seem as puerile and amusingly small as our nobleman's eighty-three. Or if we'll never name such numbers, perhaps other civilizations will. Is a biggest number contest afoot throughout the galaxy?

You might wonder why we can't transcend the whole parade of paradigms, and name numbers by a system that encompasses and surpasses them all. Suppose you wrote the following in the biggest number contest:

*The biggest whole number nameable with 1,000 characters of English text*

Surely this number exists. Using 1,000 characters, we can name only finitely many numbers, and among these numbers there has to be a biggest. And yet we've made no reference to how the number's named. The English text could invoke Ackermann numbers, or Busy Beavers, or higher-level Busy Beavers, or even some yet more sweeping concept that nobody's thought of yet. So unless our opponent uses the same ploy, we've got him licked. What a brilliant idea! Why didn't we think of this earlier?

Unfortunately it doesn't work. We might as well have written

*One plus the biggest whole number nameable with 1,000 characters of English text*

This number takes at least 1,001 characters to name. Yet we've just named it with only 80 characters! Like a snake that swallows itself whole, our colossal number dissolves in a tumult of contradiction. What gives?

The paradox I've just described was first published by Bertrand Russell, who attributed it to a librarian named G. G. Berry. The Berry Paradox arises not from mathematics, but from the ambiguity inherent in the English language. There's no surefire way to convert an English phrase into the number it names (or to decide whether it names a number at all), which is why I invoked a "reasonable modern mathematician" in the rules for the biggest number contest. To circumvent the Berry Paradox, we need to name numbers using a precise, mathematical notational system, such as Turing machines–which is exactly the idea behind the Busy Beaver sequence. So in short, there's no wily language trick by which to surpass Archimedes, Ackermann, Turing, and Rado, no royal road to big numbers.

You might also wonder why we can't use infinity in the contest. The answer is, for the same reason why we can't use a rocket car in a bike race. Infinity is fascinating and elegant, but it's not a whole number. Nor can we 'subtract from infinity' to yield a whole number. Infinity minus 17 is still infinity, whereas infinity minus infinity is undefined: it could be 0, 38, or even infinity again. Actually I should speak of infinities, plural. For in the late nineteenth century, Georg Cantor proved that there are different levels of infinity: for example, the infinity of points on a line is greater than the infinity of whole numbers. What's more, just as there's no biggest number, so too is there no biggest infinity. But the quest for big infinities is more abstruse than the quest for big numbers. And it involves, not a succession of paradigms, but essentially one: Cantor's.

So here we are, at the frontier of big number knowledge. As Euclid's disciple supposedly asked, "what is the *use* of all this?" We've seen that progress in notational systems for big numbers mirrors progress in broader realms: mathematics, logic, computer science. And yet, though a mirror reflects reality, it doesn't necessarily influence it. Even within mathematics, big numbers are often considered trivialities, their study an idle amusement with no broader implications. I want to argue a contrary view: that understanding big numbers is a key to understanding the world.

Imagine trying to explain the Turing machine to Archimedes. The genius of Syracuse listens patiently as you discuss the papyrus tape extending infinitely in both directions, the time steps, states, input and output sequences. At last he explodes.

"Foolishness!" he declares (or the ancient Greek equivalent). "All you've given me is an elaborate definition, with no value outside of itself."

How do you respond? Archimedes has never heard of computers, those cantankerous devices that, twenty-three centuries from his time, will transact the world's affairs. So you can't claim practical application. Nor can you appeal to Hilbert and the formalist program, since Archimedes hasn't heard of those either. But then it hits you: the Busy Beaver sequence. You define the sequence for Archimedes, convince him that $BB(1000)$ is more than his $10^{63}$ grains of sand filling the universe, more even than $10^{63}$ raised to its own power $10^{63}$ times. You defy him to name a bigger number without invoking Turing machines or some equivalent. And as he ponders this challenge, the power of the Turing machine concept dawns on him. Though his intuition may never apprehend the Busy Beaver numbers, his reason compels him to acknowledge their immensity. Big numbers have a way of imbuing abstract notions with reality.

Indeed, one could define science as reason's attempt to compensate for our inability to perceive big numbers. If we could run at 280,000,000 meters per second, there'd be no need for a special theory of relativity: it'd be obvious to everyone that the faster we go, the heavier and squatter we get, and the faster time elapses in the rest of the world. If we could live for 70,000,000 years, there'd be no theory of evolution, and *certainly* no creationism: we could watch speciation and adaptation with our eyes, instead of painstakingly reconstructing events from fossils and DNA. If we could bake bread at 20,000,000 degrees Kelvin, nuclear fusion would be not the esoteric domain of physicists but ordinary household knowledge. But we can't do any of these things, and so we have science, to deduce about the gargantuan what we, with our infinitesimal faculties, will never sense. If people fear big numbers, is it any wonder that they fear science as well and turn for solace to the comforting smallness of mysticism?

But *do* people fear big numbers? Certainly they do. I've met people who don't know the difference between a million and a billion, and don't care. We play a lottery with 'six ways to win!,' overlooking the twenty million ways to lose. We yawn at six billion tons of carbon dioxide released into the atmosphere each year, and speak of 'sustainable development' in the jaws of exponential growth. Such cases, it seems to me, transcend arithmetical ignorance and represent a basic unwillingness to grapple with the immense.

Whence the cowering before big numbers, then? Does it have a biological origin? In 1999, a group led by neuropsychologist Stanislas Dehaene reported evidence in *Science* that two separate brain systems contribute to mathematical thinking. The group trained Russian-English bilinguals to solve a set of problems, including two-digit addition, base-eight addition, cube roots, and logarithms. Some subjects were trained in Russian, others in English. When the subjects were then asked to solve problems approximately–to choose the closer of two estimates–they performed equally well in both languages. But when asked to solve problems exactly, they performed better in the language of their training. What's more, brain-imaging evidence showed that the subjects' parietal lobes, involved in spatial reasoning, were more active during approximation problems; while the left inferior frontal lobes, involved in verbal reasoning, were more active during exact calculation problems. Studies of patients with brain lesions paint the same picture: those with parietal lesions sometimes can't decide whether 9 is closer to 10 or to 5, but remember the multiplication table; whereas those with left-hemispheric lesions sometimes can't decide whether $2 + 2$ is 3 or 4, but know that the answer is closer to 3 than to 9. Dehaene et al. conjecture that humans represent numbers in two ways. For approximate reckoning we use a 'mental number line,' which evolved long ago and which we likely share with other animals. But for exact computation we use numerical symbols, which evolved recently and which, being language-dependent, are unique to humans. This hypothesis neatly explains the experiment's findings: the reason subjects performed better in the language of their training for exact computation but not for approximation problems is that the former call upon the verbally-oriented left inferior frontal lobes, and the latter upon the spatially-oriented parietal lobes.

If Dehaene et al.'s hypothesis is correct, then which representation do we use for big numbers? Surely the symbolic one–for nobody's mental number line could be long enough to contain $9^{9^9}$, 5 pentated to the 5, or $BB(1000)$. And here, I suspect, is the problem. When thinking about 3, 4, or 7, we're guided by our spatial intuition, honed over millions of years of perceiving 3 gazelles, 4 mates, 7 members of a hostile clan. But when thinking about $BB(1000)$, we have only language, that evolutionary neophyte, to rely upon. The usual neural pathways for representing numbers lead to dead ends. And this, perhaps, is why people are afraid of big numbers.

Could early intervention mitigate our big number phobia? What if second-grade math teachers took an hour-long hiatus from stultifying busywork to ask their students, "How do you name really, *really* big numbers?" And then told them about exponentials and stacked exponentials, tetration and the Ackermann sequence, maybe even Busy Beavers: a cornucopia of numbers vaster than any they'd ever conceived, and ideas stretching the bounds of their imaginations.

Who can name the bigger number? Whoever has the deeper paradigm. Are you ready? Get set. Go.

## References

Petr Beckmann, A History of Pi, Golem Press, 1971.
Allan H. Brady, "The Determination of the Value of Rado's Noncomputable Function Sigma(k) for Four-State Turing Machines," Mathematics of Computation, vol. 40, no. 162, April 1983, pp 647- 665.
Gregory J. Chaitin, "The Berry Paradox," Complexity, vol. 1, no. 1, 1995, pp. 26- 30. At `http://www.umcs.maine.edu/~chaitin/unm2.html`.
A.K. Dewdney, The New Turing Omnibus: 66 Excursions in Computer Science, W.H. Freeman, 1993.
S. Dehaene and E. Spelke and P. Pinel and R. Stanescu and S. Tsivkin, "Sources of Mathematical Thinking: Behavioral and Brain-Imaging Evidence," Science, vol. 284, no. 5416, May 7, 1999, pp. 970- 974.
Douglas Hofstadter, Metamagical Themas: Questing for the Essence of Mind and Pattern, Basic Books, 1985. Chapter 6, "On Number Numbness," pp. 115-135.
Robert Kanigel, The Man Who Knew Infinity: A Life of the Genius Ramanujan, Washington Square Press, 1991.
Stephen C. Kleene, "Recursive predicates and quantifiers," Transactions of the American Mathematical Society, vol. 53, 1943, pp. 41- 74.
Donald E. Knuth, Selected Papers on Computer Science, CSLI Publications, 1996. Chapter 2, "Mathematics and Computer Science: Coping with Finiteness," pp. 31- 57.
Dexter C. Kozen, Automata and Computability, Springer-Verlag, 1997.
——, The Design and Analysis of Algorithms, Springer-Verlag, 1991.
Shen Lin and Tibor Rado, "Computer studies of Turing machine problems," Journal of the Association for Computing Machinery, vol. 12, no. 2, April 1965, pp. 196- 212.
Heiner Marxen, Busy Beaver, at `http://www.drb.insel.de/~heiner/BB/`.
—— and Jürgen Buntrock, "Attacking the Busy Beaver 5," Bulletin of the European Association for Theoretical Computer Science, no. 40, February 1990, pp. 247- 251.
Tibor Rado, "On Non-Computable Functions," Bell System Technical Journal, vol. XLI, no. 2, May 1962, pp. 877- 884.
Rudy Rucker, Infinity and the Mind, Princeton University Press, 1995.
Carl Sagan, Billions & Billions, Random House, 1997.
Michael Somos, "Busy Beaver Turing Machine." At
`http://grail.cba.csuohio.edu/~somos/bb.html`.
Alan Turing, "On computable numbers, with an application to the Entscheidungsproblem," Proceedings of the London Mathematical Society, Series 2, vol. 42, pp. 230- 265, 1936. Reprinted in Martin Davis (ed.), The Undecidable, Raven, 1965.
Ilan Vardi, "Archimedes, the Sand Reckoner," at
`http://www.ihes.fr/~ilan/sand_reckoner.ps`.
Eric W. Weisstein, CRC Concise Encyclopedia of Mathematics, CRC Press, 1999. Entry on "Large Number" at
`http://www.treasure-troves.com/math/LargeNumber.html`.