T. Toffoli (ed.): submitted to *Int. J. Theor. Physics*
for a special issue on the *Digital Perspectives* workshop, Washington DC, July 2001.

Explicit finitism

András Kornai
Metacarta Inc., 126 Prospect St., Cambridge MA 02139
andras@kornai.com

1 Introduction

Recently Lloyd[2] established 10^{120} bits as an upper bound for the storage capacity of the universe. Mathematicians and computer scientists tend to be dismissive about such notions of a computationally closed universe, comparing the issue to one of characterizing the set of functions computable if you are given a roomful of NAND gates. In this paper we show how the problem can be stated in a more interesting form, as being about sequences J_i of subsets of the real numbers tending to a limit $J \subset R$, and pose some open questions about J.

We assume a background storage of 2^{512} bytes (about 10^{155} bits) – the body of this paper is devoted to exploring the range of mathematics that can be performed in this *explicitly finite* realm. Our goal is to collect all real numbers accessible to present and future computers in J, but there are serious difficulties, both philosophical and technical, in developing a theory that takes the stance that numbers not computable with a computer with limited (albeit large) amount of storage are indeed out of reach. On the philosophical side, it is very hard to answer the charge of arbitrariness: what if we used the number of elementary logical operations, wouldn't we obtain a somewhat different constant than the limit based on considerations of space? Actually, the two numbers are highly similar (as noted by Lloyd), but this is more of a statement about our universe than about a mathematical system: surely, if we can conceive of a system of size X we can also conceive of a system of size 2X.

To clarify matters, we use a convenient means of rigorously defining and talking about large numbers, the Ackermann function. As usual, we define $A(n,m)$ by induction: $A(1,m) = 2m, A(n,1) = 2, A(k+1,n+1) = A(k, A(k{+}1,n))$. Therefore, $A(2,m) = 2^m$ and $A(3,m)$ is a tower of stacked exponentials of m 2s (for brevity, we will denote such towers by $E^*(m)$). In a lecture Friedman[1] proposed $A(5,5)$ as a "sort of benchmark" for incomprehensibly large numbers. Here we will describe why $A(4,4) = E^*(65536)$ must already be considered to be outside the arithmetic power of of any civilization restricted to the material resources of our universe. Note that our baseline, $2^{512} = 2^{2^9}$, is well below $E^*(5)$.

Let us first consider the product $143 \cdot 157$. As any third grade student can tell you, this is 22451. A student of high school algebra may notice that $143 = 150 - 7, 157 = 150 + 7$ so that the formula $(a-b)(a+b) = a^2 - b^2$ is applicable, and may take advantage of this and similar facts, such as $150^2 = (15 * 10)^2 = 225 * 100$, to compute the result from $22500 - 49$, without going through the tedious steps of the multiplication algorithm the third grade student used. Yet at the heart of our understanding of arithmetic we find the tedious algorithm used by the third grader, rather than the more sophisticated reasoning used by the high school student, for the simple reason that the basic algorithm remains usable even in those case where we find no special properties to help us. In particular, when the numbers to be multiplied are random, we can only rely on the generic methods. To be sure, there are better (deeper and more effective) methods for multiplying too large integers than our third grader is aware of (e.g. based on the Fast Fourier Transform), but this does not affect the argument inasmuch as these methods are also generic.

Returning to $E^*(65536)$, we can see many ways we can take advantage of its special form to answer arithmetic questions about it. We may not be able to actually divide it by 7 and convert the result to decimal notation, but we can be absolutely certain that if we did so, the thirty third digit of the fractional part would be 5. However, there are many simple questions about this number that we are not in a position to answer. In base 10, how many digits would it have? The answer is given by the integer part of $E^*(65535) \cdot 0.30102999566398...$, but this in not a product we can evaluate to the required degree of precision. In fact, it is not trivial to determine what its first 10 digits are. There may be some clever algorithm to compute this, but the standard methods break down.

Notice that we do not have the same problem with $E^*(5) = 2^{65536}$: we can compute with quite ordinary resources that it would have 19728 decimal digits. For $E^*(6)$ this is a bit more complicated, we would have to compute the base 10 log of 2 to nearly 20k digits, compute $E^*(5)$, and multiply the two. Using fast multiplication, this is still well within the resources of an ordinary PC: the results would run to ten printed pages (assuming 2k characters per page). But for $E^*(7)$, we would have to calculate log(2) to 10^{19728} decimal places, and this is well beyond the material resources of the universe. By the 7th iteration we are already in trouble, and we would have to do 65536 iterations to get to $A(4,4)$.

To summarize, it is not the raw size, but rather the information content of a number that determines its accessibility. The key issue here is randomness: powers of 2, towers of such powers, and in general the values of the Ackermann function, are far from random, both in the technical sense of Kolmogorov complexity, and in the more physical sense of randomness that we will first rely on. Some numbers x are more accessible to our arithmetic than other numbers y in the sense that we can establish certain arithmetic statements $D(x)$ but not $D(y)$. It follows from a cardinality argument that we must have many numbers z about which nothing but trivial arithmetic facts, such as $0 \cdot z = 0$ can be established. The standard mathematical tools of investigating complexity lack the required resolving power: even the lowest Turing degree lumps together problems which are solvable, such as finding the number of digits in $E^*(6)$, and problems which are not, such as finding the number of digits in $E^*(7)$. Since we are expressly including symbolic techniques, we need a setup devoted to the manipulation of strings, trees, and more complex structures of symbols. Once such a setup is defined, there is a finite number of states S it can have, and any fixed interpretation of the states can only have a finite set $J(S) \subset R$ as its image.

2 Defining J by machines

By pointing her browser to `http://www.fourmilab.com/hotbits` the user can download about 240 randomly generated bits per second. Whether these bits, generated by comparing the length of consecutive intervals between ticks of a Geiger counter, are truly random, is of course a deep philosophical question, but one that we need not address here. For our purposes, it is sufficient to note that in order to replicate or predict these bits one would need the resources to simulate our whole universe, and such resources can not be found within the universe itself. For the mathematician, a simple cardinality argument shows the existence of truly random (incompressible) bit sequences, but for the more practical-minded reader, the hotbits setup provides a constructive method, complete with hardware description and wiring diagrams, for obtaining incompressible sequences.

If we impose an explicit memory limitation, Turing machines and finite transducers would become equally good computational models, but we need not take either of these as basic. Our model of computation (see `http://www.kornai.com/Drafts/fathom.html`) takes the radical step of incorporating the full set of the reals, R, in the machine model itself. This means that J itself can only be approximated by real hardware built from finitely many elements, but this is not a serious problem inasmuch as our goal with J-machines is to provide an upper bound on the set J by an abstract model of computation rather than to actually compute things. By adding the reals we accomplish two main goals: enable direct modeling of deterministic physical theories (Montague[3]), and make it possible to deploy Blum-Shub-Smale complexity theory to analyze the symbolic (as opposed to numeric) techniques available on the machine.

In all other respects our computing model is a straightforward extrapolation of what is happening in any modern computer – here we confine ourselves to a few remarks about memory. To simplify matters, we assume that a cache of 2^{32} bytes (4 gigabytes) is available on chip (currently unrealistic, but easy enough to simulate). We can think of this cache as the innermost (0th) layer of memory, composed of registers, whose contents are directly accessible (in a single cycle) to primitive operations such as addition. At the next (1st) layer, we assume that 2^{64} bytes (16 exabytes) memory is directly addressable – this will be called the *core*. To perform (arithmetic or logical) operations on numbers stored in the core requires a few CPU cycles for bringing them to the 0th layer, and some care in programming to make sure still valuable parts of the cache are not overwritten.

Finally there is an outer (2nd) layer of memory in the 2^{512} byte range, called the *disk*. Since this requires the whole universe, we do not follow the usual assumption that fetching data from this layer can be done in a constant number of cycles. Rather, we assume that this is limited by the speed of light, so that if memory is arranged linearly, the time required for reading or writing the nth byte is proportional to n. If memory is arranged in concentric circles, the time required is proportional to $n^{1/2}$, if it is arranged spherically, to $n^{1/3}$, which is the best we can do. These non-random access characteristics call for a whole set of unusual memory management techniques.

First, we wish to be able to seed far parts of the disk with colonies of computing agents, who will use certain parts (local to them, but not to us) as core and cache. Second, we need to make sure that different colonies, who may themselves be engaged in their own secondary or n-th generational colonization efforts, recognize different parts of the disk as being already in use, and don't step on it. Third, we can not simply assume a generally shared system of coordinates, or a master plan that each colony will abide by, for the simple reason that a system the size of the disk can not be kept noise-free. Therefore, allocating a large segment will actually consume some overhead space to mark the segment as being in active use, and possibly some constant drain on time as well, for running a process that defends the segment from corruption by other processes. Finally, note that at this level of abstraction we do not need to consider parallelism, since all computers under the control of our civilization can be thought of as being part of the same large J-machine.

The points on the real line which are directly accessible to this kind of computing apparatus are collected in the set J_{512}. More crude approximations of the set J are provided by J_0 (no external memory) to J_{511}, and finer approximations J_n for $n > 512$ are also logically conceivable. In the limit we would obtain all the Turing-computable numbers, but the key issue is understanding the fractal properties of the sets leading to the limit. It is important to keep in mind that many numbers such as 4.5558062... that are not directly accessible, may still be symbolically accessible, in this case as $\pi + \sqrt{2}$. It is this kind of symbolic accessibility which makes it necessary to incorporate the full set of the reals in J as part of its hardware: as a benign side effect, we no longer have to worry about countable models.

What we are interested in is the set J of explicitly computable numbers, which is already approximated *from above* by J_{512}. Clearly, any rational whose numerator and denominator are both sufficiently small (e.g. less than 512 bits) will be included in J, but rationals requiring a large amount of storage, e.g. a random integer composed of 2^{512} digits, are outside J. The same is true for irrationals: those with a sufficiently compact definition (e.g. as a root of a small polynomial with small coefficients, or the limit of a simple power series) are included, but those with large definitional complexity are excluded. Although the numbers in J are not all rationals, the situation is in many respects analogous to that found in floating point hardware, to which we turn now.

In IEEE-754 each number requires 8 bytes: a sign bit s, 52 bits for mantissa M, and 11 bits for the base 2 exponent e. Triples (s, m, e) are mapped into reals by the following interpretation function: $i(0,0,0) = 0; i(1,m,e) = -i(0,m,e); i(0,m,e) = 1.m \cdot 2^{e-1023}(0 < e < 2047)$. Let us collect the numbers that can be represented in this format in the set I: the largest member of I is $2^{1024} - 2^{972}$. For any real number x with smaller absolute value we define Up(x) as the smallest number in I above x, Down(X) as the largest number in I below it, and Near(x) as Up(x) or Down(x) depending on which is closer to x. Let $\circ$ be one of $+, -, \cdot$, and $/$, `RoundingMode` be one of Up, Down, or Near, and $a, b \in I$.

A chip satisfies the IEEE-754 standard if (i) whenever $a \circ b \in I$ the computation returns this exact value, (ii) when ($a \circ b \notin I$ but lies between the smallest and the largest representable number, the chip returns the number appropriate for the pre-set `RoundingMode`, and (iii) when the result would fall outside the representable range the chip signals the overflow just as it signals division by zero. (We could require separate signals for underflow to zero, but it is easy to see that if overflow signals are guaranteed we can always structure the computation so as to avoid silent underflow.) The existence of different rounding modes enables the chip to perform semantically correct interval analysis: for example in interval addition we select Down when we compute the left, and Up when we compute the right end of the result interval, thereby guaranteeing that the true result is always in the computed interval. It is not hard to see how $I(1, 52, 11)$ would generalize to $I(1, 112, 15)$ (quad) or even higher precision. For reasonably small values of i and j $I(1, i, j)$ approximates J from below. In this context, a megabyte is still reasonably small – while nobody actually needs hardware support for $I(1, 2^{20}, 2^{20})$ "megaprecision" arithmetic, it is clear that this could be emulated on ordinary hardware at the cost of moving from gigahertz to kilohertz speeds.

3 Open problems

To some extent, the set J of numbers computable in this universe is an object of curiosity, or even an object of religious awe, the same way as Chaitin's ω. However, it is an object that is defined by our universe, rather than by some arbitrary choice of encoding or enumeration (see Raatikainen[4]), and as such deserves some attention. In this paper we have provided an upper bound J_{512} and a lower bound $I(1, m, m)$ on J, but left essentially all important problems about it open.

It is clear that J is symmetrical about the origin, but not closed under addition, reciprocal, or multiplication. Functions from J to itself are not necessarily computable, not even linear functions with small coefficients. Elementary statements about "true" addition and "rounded" addition are indistinguishable within the resolution offered by J, so real numbers do *not* have a unique description in a form $j + r$, where $j \in J$ and r is some small remainder term, as was the case in $I(1, i, j)$.

J gives rise to a new form of the Berry paradox: what is the first integer that is excluded from it? Clearly, J was defined by a finite method, and the definition required much less than 2^{512} symbols, so aren't we defining the undefinable here? The correct response seems to be that there is such a number in N, but we would need resources greater than what we can actually muster to compute it. Some science-fictional power, with a great deal more computing muscle, can compute it.

Is there a limit above which J has no members? This is to some extent the mirror image of the Berry paradox, but the answer is less clear-cut: a lot depends on the details of the notation we adopt for large numbers. What is clear from the definition is that random numbers above 2^{512} are excluded, but large numbers like $A(4, 4)$ will only be excluded if we strictly identify knowing a number with being able to compute all its digits (which is, of course, the only viable definition at the moment).

Since J has a fractal-like structure, perhaps the first order of business would be to establish its dimension: we leave the reader with this challenging open problem.

Acknowledgments

The author gratefully acknowledges the help of Tibor Beke (University of Michigan, Ann Arbor), Bruno Caprile (ITC/IRST), Péter Gács (Boston University), Rick McGowan (Unicode Consortium), Doug Merritt (ex-UCB), Gábor Tóth (Loránd Eötvös University, Budapest) and Tom Toffoli (Boston University) in helping him formulating some of the ideas in this paper.

References

[1] Friedman, Harvey, "Enormous integers in real life," 1999, `www.math.ohio-state.edu/foundationslct/EnormousInt.12pt.6_1_00.doc`.

[2] Lloyd, Seth, "Computational capacity of the universe," *Phys. Rev. Lett.* **88** (2002), 237901.

[3] Montague, Richard, "Deterministic theories,"in Richmond H. Thomason (ed.), *Formal Philosophy*, Yale University Press 1974.

[4] Raatikainen, Panu, "On interpreting Chaitin's incompleteness theorem," *J. Philos. Logic* **27** (1998), 569–586.