

January 15, 1996

**The Outer Limits: In Search of
the “Unknowable” in Science**

JOHN L. CASTI

Technical University of Vienna
A-1040 Vienna, Austria

AND

Santa Fe Institute
Santa Fe, NM 87501

Chapter 2

THE OUTER LIMITS: IN SEARCH OF THE “UNKNOWABLE” IN SCIENCE

JOHN L. CASTI

I. Logical Barriers in Science

To anyone infected with the idea that the human mind is unlimited in its capacity to answer questions about natural and human affairs, a tour of 20th-century science must be quite a depressing experience. Many of the deepest and most well-chronicled results of science in this century have been statements about what *cannot* be done and what *cannot* be known. Probably the most famous limitative result of this kind is Gödel's Incompleteness Theorem, which tells us that no system of deductive inference is capable of answering all questions about numbers that can be stated using the language of the system. In short, every sufficiently powerful, consistent logical system is incomplete. A few years later, Alan Turing proved an equivalent assertion about computer programs, which states that there is no systematic way of testing a program and its data to say whether or not the program will ever halt when processing that data. More recently, Gregory Chaitin has looked at Gödel's notion of provability from an information-theoretic perspective, finding explicit examples of simple arithmetic propositions whose truth or falsity will never be known by following the deductive rules of any system of logical inference. Essentially, what Chaitin's results show is that such mathematical questions are simply too complex for us.

The theorems of Gödel, Turing and Chaitin are limitations on our ability to know in the world of mathematics. The same limitation applies to statements such as the celebrated Heisenberg Uncertainty Principle in quantum theory, which at first glance appears to refer to an inherent limitation on our ability to measure certain quantities in the physical world. But a more careful examination shows that Heisenberg uncertainty

is actually a limitation imposed by certain *mathematical* formulations of quantum theory, and may or may not be a limitation in the real world itself. Similar remarks apply to limitations like Arrow’s Paradox in social choice theory and the Central Dogma of Molecular Biology, which are both limitations imposed by *models* of the real world rather than provable limitations about what can be known and/or done in the real world itself. So it’s reasonable to wonder if there exist questions about the worlds of natural and human phenomena whose answers science is forever powerless to uncover.

The first, and perhaps most vexing, task in confronting this issue is to settle on what we mean by “scientific” knowledge. Philosophers have grappled with the problem of what constitutes knowledge for ages, with no end to their struggle yet in sight. So to cut through this philosophical Gordian knot, let me advance the perhaps moderately controversial position that the *scientific* way to answer a question makes use of a set of rules, essentially a computer program. We simply feed the question into the rules, turn the crank of logical deduction, and wait for the answer to appear as the output of program.

But rules come in many flavors, and certainly not all of them qualify as being scientific. For example, the Ten Commandments are certainly a set of rules. Moreover, they help explain the empirical fact that the majority of people are not regularly engaged in robbery, murder or other extreme types of antisocial behavior. But hardly anyone would consider these rules to be in any way “scientific.” Similarly, the astrologer’s rule “Saturn in the Second House predisposes one to financial misfortunes” is also considered unscientific today, although in an earlier era it might well have been regarded as the height of scientific respectability. This shows, incidentally, that what is and is not scientific is a time-dependent phenomenon, and that scientific rules—or “laws”—are not as absolute as a lot of scientists would like to believe. But if the Ten Commandments and “Saturn in the Second House” are not scientific rules for predicting and/or explaining observed phenomena, what is?

Basically, there are two quite different sets of criteria that must both be satisfied for a rule to have even a chance of being scientific. The first pertains to properties of the rule itself, while the second has to do with the way the rule is arrived at. In regard to the first type of criterion, here is a checklist of characteristics that tend to separate the scientific rules from the pretenders.

- *Explicit*—Scientific rules are explicit, in the sense that there is no ambiguity in the statement of the rule and it requires no private

interpretation to employ the rule for prediction or explanation. For example, Newton's laws of motion state an explicit relationship linking the positions, masses and velocities of a collection of material particles. And as long as you understand what is meant by the terms *position*, *velocity* and *mass*, there is no question about either what the rule says or how it is to be applied.

- *Public*—Scientific rules are open to public scrutiny. They are presented in the open literature and can be tested by anyone who has the time, money, equipment and desire to do so. The contrast here with other types of rules is clear, especially those arising from many religions, where rules accessible only to the “divinely inspired” often play a central role in forming the tenets of a particular system of beliefs.

- *Reliable*—Scientific rules have stood the test of time. Before they are accepted as legitimate laws of nature by the scientific community, the rules must have succeeded in predicting and/or explaining a variety of phenomena over a substantial period of time. Of course, this doesn't mean the rules are infallible and cannot be overthrown in the light of new evidence. But generally speaking, the weight of evidence in favor of the rule must be quite overwhelming before we dignify the rule by labeling it *scientific*.

- *Objective*—Scientific rules are objective in that they are relatively free of investigator bias. In other words, the rule is independent of the social position, financial status or cultural background of the investigator. For example, the exponent in Newton's inverse-square law of gravitation is 2 and not 2.315 or $\sqrt{7}$ or any other number besides 2. And this remains the case for any investigator, regardless of that investigator's professional situation, political leanings or bank balance. In short, the rule is observer-invariant.

I hasten to point out that this does *not* mean that different investigators might not formulate the rule in different terms. But all these formulations must eventually turn out to be equivalent if the rule is to be taken seriously as a scientific rule. For example, in the early days of quantum mechanics there were three seemingly different formulations of quantum phenomena by Heisenberg, Schrödinger and Dirac. Yet upon further investigation it turned out that all three were essentially the same formulation and could be transformed, one to the other, via routine mathematical operations. So what on the surface looked like different rules ended up being the same rule dressed up in different mathematical clothing. This is the kind of objectivity that's characteristic of scientific rules.

So explicitness, public availability, reliability and objectivity are four key properties of rules that tend to separate scientific prescriptions from those that aren't. But earlier we said that these properties constitute just one of two sets of criteria that a rule must satisfy before we label it *scientific*. The other involves the procedure, or method, by which we generate the rule.

If there's even one thing that students in courses on the philosophy of science remember years later, it's the idea of the *scientific method*. This is the process by which many philosophers of science claim that science distinguishes itself from other reality-generation schemes. And it is this process that gives rise to the kinds of rules that serve as candidates for the coveted accolade *scientific*. The principal steps in this process are shown schematically in Figure 1. Here we see three stages: Observation, Hypothesis and Experiment. Traditionally, it's argued that the process starts with Observation, and strictly speaking, I suppose this is indeed always the case. But in a field that's already well developed—particle physics, for example—the diagram may just as often be entered at the Hypothesis stage as at the level of Observation. In any case, after a few tours around the diagram we can hope that the process will converge to something. And that something is what serves as a candidate for a scientific rule. Furthermore, if our use of the scientific method does not converge to something, we generally give up trying to fit the observed phenomena into the framework of science.

Once such filters have been applied to separate scientific rules from the *poseurs*, what remains is pretty much an algorithmic notion of what we usually consider to be a scientific theory. And it is theories we use to answer questions. The problem at this point is that the notion of a set of rules, or an algorithm, is an informal one. Yet to make precise statements about whether or not an algorithm exists for producing the answer to any particular question, we have to formalize the idea of “a set of rules.”

– Models of Computation –

In 1936, Alan Turing published the first truly satisfactory formalization of what we mean by an ‘algorithm.’ Turing's idea was what we now call a *Turing machine*. This is a kind of paper computer, consisting of an infinite tape and a scanning head that runs along the squares of the tape, writing/erasing 0s and 1s depending on what it reads on the current square and the “state” (i.e., configuration) the head happens to be in at the current moment. Thus, the Turing machine is specified completely

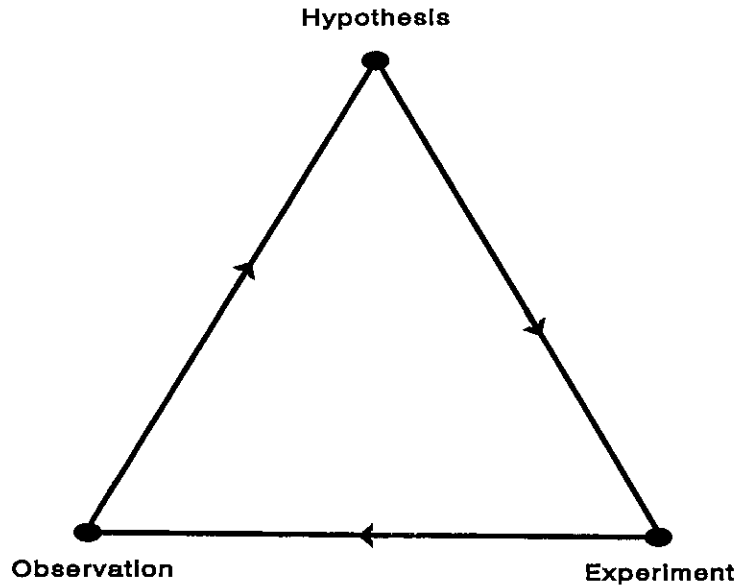


Figure 1. The scientific method.

by its *program*, which is simply a set of instructions telling the scanning head what to do at each moment as a function of what symbol is read on the tape and what the head's current state happens to be. A typical such instruction might be (1, A; R, 0, B), which is interpreted as: "If the tape square currently being scanned contains a 1 and the head is in state A, move Right one square, print a 0 on the new square and enter the state B." Every Turing machine and, hence, Turing machine program, is defined by a finite set of such statements. More details on Turing machines can be found in [1, 8].

The Turing machine is what's called a *model* of computation, since it is a formal mathematical structure defining what we mean by a "computation." But contrary to what some seem to believe, it is not the only model. In fact, a few years ago a very different model was constructed by Blum, Shub and Smale [9]. This BSS model of computation was based on the (informal) idea of a flowchart for a computer program, and gives a different answer than the Turing machine to what is and isn't computable by following a set of rules. So the notion of computability is not an absolute one, but rather hinges upon which model of computation one chooses to employ. The Turing machine model is the de facto standard, and will suffice for the discussion that follows. But it's important to keep in mind that what is scientifically unknowable by the nonexistence of

algorithm of the Turing-type may all of a sudden become knowable in the BSS model.

II. The Intractable

Thinking of knowledge, scientific-style, as tantamount to what can be generated by a computer program opens up the issue of computational intractability. We know that there exist puzzles like the celebrated Traveling Salesman Problem whose computational difficulty is widely believed to increase exponentially with the size of the problem. Thus, to calculate the minimal-cost tour of the 180 or so world capital cities by a brute-force examination of each of the $180!$ possible routings would require a time much greater than the age of the universe with even the fastest of computers. But such a computation is possible—at least, in principle. Let's look just a bit deeper into these types of computational questions.

A famous problem of recreational mathematics is the so-called *Tower of Hanoi*. In this problem, there are three pegs A , B , and C , with N rings of decreasing radii piled on the first peg A . The other two pegs are initially empty. The task is to transfer the rings from A to B , perhaps using peg C in the process. The rules stipulate that the rings are to be moved one at a time, and that a ring can never be placed upon one smaller than itself. Figure 2 shows the problem for the case of $N = 3$ rings.

In this case of three rings, it's not too hard to see that the sequence of seven moves

$$\begin{array}{cccc} A \rightarrow B, & A \rightarrow C, & B \rightarrow C, & A \rightarrow B, \\ C \rightarrow A, & C \rightarrow B, & A \rightarrow B & \end{array}$$

achieves the desired transfer of rings. And, in fact, it can be shown that there is a general algorithm, that is, a program, solving the game for

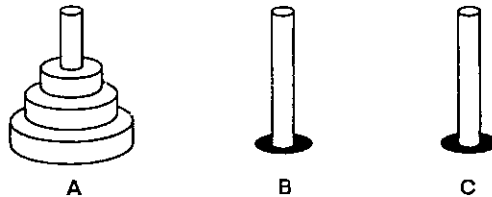


Figure 2. The Tower of Hanoi Game.

any number of rings n . This program shows that the minimal number of transfers required is $2^n - 1$. Amusingly, the original version of this puzzle, dating back to ancient Tibet, involves $n = 64$ rings. So it's not hard to see why the Tibetan monks who originated the game claim that the world will end when all 64 rings are correctly piled on peg B . To carry out the required $2^{64} - 1$ steps, even performing one ring transfer every 10 seconds, would take well over 5 trillion years! Thus, the number of steps needed for the solution of the Tower of Hanoi problem grows exponentially with the number of rings n . This is an example of a "hard" computational problem—one in which the number of computational steps needed to obtain a solution increases exponentially with the "size" of the problem.

By way of contrast, a computationally "easy" problem is the sorting of a deck of playing cards into the four suits in ascending order. First go through the deck until you find the ace of spades. Set it aside and then go through the remaining cards until you find the two of spades, which you also set aside. As one continues in this fashion, the deck is fully sorted. The worst that can happen with this sorting scheme is that the ace of spades is the last card in the unsorted deck, the two of spades is the next-to-last card, and so on. So starting with n cards, you would have to examine at most n^2 cards. Thus, the number of steps needed to completely sort the deck is a quadratic function of the size of the problem, that is, the number of cards in the deck.

In the mid-1960s, J. Edmonds and A. Cobham introduced the idea of classifying the computational difficulty of a problem according to whether or not there exists an algorithm for solving the problem that requires at most a polynomial number of steps in the size of the problem. "Easy" problems can be solved in polynomial time; "hard" problems, on the other hand, require an exponentially increasing number of steps as the problem size grows. The difference in these two rates of growth is shown in Figure 3, where we see that a polynomial function may actually exceed an exponentially increasing one for small values of the size n . But as n continues to grow, the exponential always wins out. To be definite about things, an algorithm is said to run in *polynomial time* if there are fixed integers A and k such that for a problem of size n , the computation will be completed in at most An^k steps. For future reference, we let P be the class of all problems that run in polynomial time. Algorithms that do not run in polynomial time are said to run in *exponential time*. Therefore an algorithm that requires 2^n or $n!$ steps to solve a problem of size n is an exponential-time algorithm.

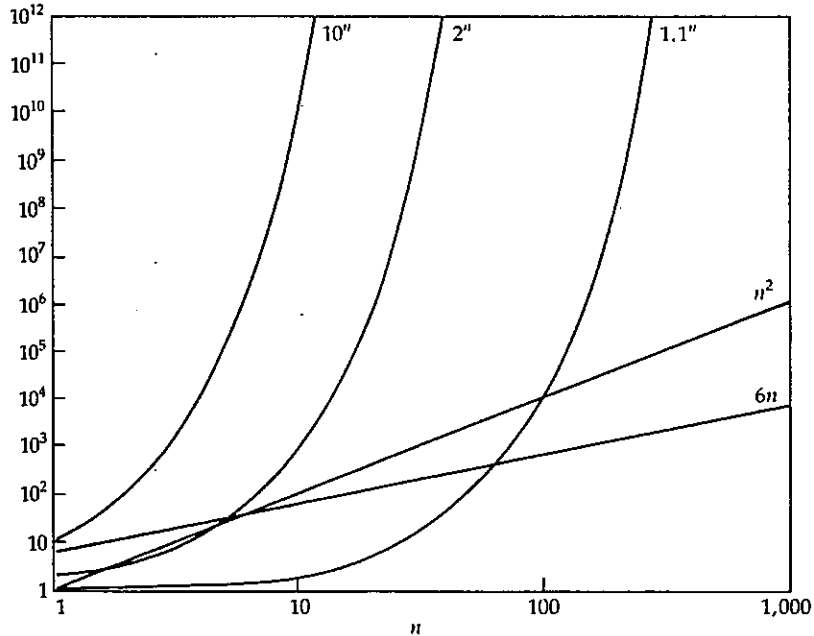


Figure 3. Polynomial and exponential growth of functions.

As an aside, it's worth noting here that this classification into hard and easy problems may be a bit misleading when it comes to hands-on computation. For example, an algorithm that has $A = 10^{50}$ and $k = 500$ will still run in polynomial time. But such an algorithm would hardly be "efficient" in any practical sense. On the other hand, an algorithm for the same problem that ran in a number of steps on the order of 2^n might well be preferable to the polynomial-time procedure for small to moderate values of n —even though it is theoretically "inefficient." So the reader should keep in mind that what we are talking about here is the *theory* of computation, not its practice. However, in practice it seems to be the case that problems are solvable either only by exponential-time algorithms or by polynomial-time algorithms that run on the order of something like $10n^2$ or $50n^3$ steps—or less.

– P and NP –

There is a very important class of problems known as *NP*, which stands for "nondeterministic polynomial time." Let me hasten to point out that this does *not* mean that there is something fuzzy, random, or indeterminate about such problems. Rather, a problem is classified as

NP if it's possible to verify a proposed solution of the problem in a number of steps that grows polynomially in the problem size. So if you happen to stumble across what you think is a solution to a problem in *NP*, you can verify or refute that it is indeed a solution in polynomial time. A good example is the problem of solving a jigsaw puzzle. If the puzzle has a large number of pieces, putting it together correctly is hard. But to check that any particular assembly is really a correct solution to the puzzle is easy: just look at it. Clearly, the polynomial-time problems are a subset of those in *NP*. Here are some examples of other problems that are also in *NP*:

- *Routing problem*: Suppose you are a salesperson who has to visit clients in a number of cities, and you want to make your round of visits without visiting any city more than once. Given the network of cities and the roads linking them, is there a route that starts and finishes at the same city and visits every other city exactly once?
- *Assignment problem*: Given information on lecture times, students, and courses, does there exist a timetable for each student so that no student has a conflict?
- *Map-coloring problem*: Does there exist a way to color a given map using just three colors, so that no two countries with a common border (greater than a single point) have the same color?
- *Bin-packing problem*: Given a collection of packages, together with their sizes, and a collection of identical bins into which the packages are to be placed, does there exist an assignment of packages to bins such that every package can be placed in a bin without any bin overflowing?

There are many problems in *NP* for which it's unknown whether or not they are also in *P*. For example, the question of whether or not n linear constraints on n variables have a solution in integers is of great importance in optimization theory, as we'll see in the next chapter. Yet there is no known polynomial-time procedure for deciding this question. While we have as yet no ironclad proof showing that $P \neq NP$, most computer scientists would be shocked into a state of total catatonia if that turned out not to be the case. One of the main reasons is that a very large number of *NP* problems have all been shown to be equivalent in the sense that if one of them turns out to be in *P*, then they are all in *P*.

This fact, which was proved by Stephen Cook in 1971, is probably the central result of combinatorial computational complexity theory. It serves to motivate work in the field, since it says that in order to refute the $P = NP$ assertion, all we need do is produce a single instance of an NP problem for which there cannot exist a polynomial-time algorithm for its solution. But so far no one has managed to find this elusive counterexample. Who knows, maybe the assertion $P = NP$ will turn out to be an undecidable proposition, thus independent of the usual axiomatic framework of mathematics, just as the famous Continuum Hypothesis was shown to be neither provable nor unprovable in the 1960s. Perhaps.

But do we really want to wait till the Big Crunch to find the most efficient way to visit the world capitals? And do we really care about the absolutely optimal way to schedule students, courses and classrooms? Wouldn't a quick-and-dirty good approximation serve just as well? It's when we begin asking questions like this about time and money that we reach the point at which science and mathematics part company. Enter the notion of *information-based complexity (IBC)*, in which we try to measure the cost in computational resources to obtain the answer to a question to a predetermined accuracy. Let's look at an example of this sort of unknowability.

– Information-Based Complexity –

Consider the problem of approximating the definite integral

$$J(f) = \int_0^1 f(t) dt.$$

To make things interesting, assume the function f has no closed-form anti-derivative, so that all we have at our disposal with which to compute an approximation to $J(f)$ is n values of the integrand f . Let

$$N(f) = [f(t_1), f(t_2), \dots, f(t_n)]$$

denote the *information pattern* available about the function f . To sprinkle a little analytic structure onto the situation, assume that $f \in C^r[0, 1]$ and that the derivatives of f are uniformly bounded by, say, 1. Thus, the class of functions we consider is

$$\mathcal{F} = \left\{ f \in C^r[0, 1] : |f^{(k)}(t)| \leq 1, k = 0, 1, \dots, r \right\}.$$

Our goal is to compute an approximation $U(f)$ to $J(f)$ such that

$$|J(f) - U(f)| \leq \epsilon,$$

for all $f \in F$, where ϵ is some given positive number. Of course, the approximation $U(f)$ is generated by performing various mathematical operations on the information pattern $N(f)$, i.e., $U(f) = \phi(N(f))$, where ϕ is a function defining whatever arithmetic operations, comparison of real numbers, and evaluation of elementary functions needed to compute U for a given information pattern $N(f)$. We define the cost of computing $U(f)$ to be the cost of making the n function evaluations $N(f)$, together with the cost of the operations employed in computing ϕ .

Following in the footsteps of Joseph Traub and Henryk Woźniakowski [6–7], we can now define the computational complexity of the problem of definite integration, $c(\epsilon)$, to be the minimal cost of computing an ϵ -approximation to $J(f)$ for all $f \in \mathcal{F}$. Note that the quantity $c(\epsilon)$ is an *intrinsic* property of computing an ϵ -approximation to a definite integral for the function class \mathcal{F} . In particular, it does not depend on any specific method. It turns out that for definite integration, $c(\epsilon) \sim (1/\epsilon)^{1/r}$.

The three pillars underlying IBC are that information is incomplete, noisy and doesn't come for free. So, for instance, in the integration example, the information pattern is the set of function values $N(f)$, together with the knowledge that $f \in \mathcal{F}$. Generally speaking, this is not enough information to pin down uniquely the function f . So the information at hand is only partial. Moreover, the numbers forming the set $N(f)$ are not computed exactly. Usually they come from some kind of numerical operations that contain round-off error or from measurements that are themselves imprecise. So the information is noisy. Finally, information, like everything else worth having in life, costs something to get. Depending on the situation, the cost may come from making the function values or from performing the combinatory operations represented by the function ϕ . But in all cases there is some cost associated with obtaining the information pattern. The primary question studied in IBC is how to obtain an approximate solution to a problem at minimal cost. Let's see how IBC theorists go about addressing this question of basic practical concern in the case of numerical integration.

Recall, the problem is to calculate an ϵ -approximation to the integral

$$J(f) = \int_0^1 f(t) dt,$$

where the class of functions is $\mathcal{F} = \{f : f \in C^r[0, 1], \|f^{(r)}\|_\infty \leq 1\}$. As for the algorithm ϕ , one possibility is ordinary Riemann approximation

$$\phi(N(f)) = \frac{1}{n} \sum_{i=1}^n f(t_i).$$

In the average-case setting, we seek an approximation $U(f)$ such that

$$\left(\int_{\mathcal{F}} |J(f) - U(f)|^2 d\mu(f) \right)^{1/2} \leq \epsilon,$$

where we could choose the probability measure μ to be a truncated Wiener measure on the r th derivatives of f .

In order to define what we mean by computational complexity, we have to have some model of the computational process. A reasonable set of postulates upon which to base such a model are:

1. For every $f \in \mathcal{F}$, the computation of $U(f)$ costs an amount $c > 0$;
2. Each combinatory operation associated with carrying out the algorithm ϕ can be performed without error and costs a unit amount. In other words, we can perform operations on real numbers without error, and each such operation costs us a unit amount.

So if we let $\text{cost}(U, f)$ denote the total cost of computing the approximation $U(f)$, then we have

$$\text{cost}(U, f) = \text{cost}(N, f) + \text{cost}(\phi, N(f)),$$

where the first term on the right-hand side is the cost of obtaining the information $N(f)$, while the second term measures the cost of combining this information to form the approximation U .

We can now define the computational complexity of a problem to be

$$c(\epsilon) = \inf\{\text{cost}(U) : e(U) \leq \epsilon\},$$

where $e(U)$ is the error of the approximation U . Note that here we adopt the convention that the cost is infinite if there are no ϵ -approximations. In the two settings we have been discussing, worst-case and average-case, the costs and error functions are given by

– *Worst-Case Setting* –

$$\begin{aligned} e(U) &= \sup_{f \in \mathcal{F}} \|J(f) - U(f)\|, \\ \text{cost}(U) &= \sup_{f \in \mathcal{F}} \text{cost}(U, f). \end{aligned}$$

– *Average-Case Setting* –

$$\begin{aligned} e(U) &= \left(\int_{\mathcal{F}} \|J(f) - U(f)\|^2 d\mu(f) \right)^{1/2}, \\ \text{cost}(U) &= \int_{\mathcal{F}} \text{cost}(U, f) d\mu(f). \end{aligned}$$

From these expressions we see that the complexity of a problem involves the intrinsic difficulty in solving that problem, and has nothing to do with the particular algorithm we employ. Rather, it depends on J and \mathcal{F} , as well as upon the setting and the set of allowable information construction operations. Finally, it depends on the model of computation we use and, in the average-case setting, on the probability measure μ . So what do these complexities look like for integration?

It turns out that for functions f of smoothness class r , we have

$$c(\epsilon) \sim \left(\frac{1}{\epsilon} \right)^{1/r}.$$

From this expression, we can get a good handle on the effect of changing the smoothness and/or the error tolerance on the difficulty of calculating the integral of f . For instance, if we go from a once-differentiable function ($r = 1$) to twice-differentiable, then the complexity decreases by a factor of $\epsilon^{1/2}$. It's interesting to note that if we demand only that f be continuous ($r = 0$), then the complexity is infinite! This reflects the well-known fact that continuity doesn't really impose much by way of restricting how the function f can wiggle around. And it's the "wiggles" that cause difficulties in determining anything interesting about a function's nature (like its integral) from a finite amount of numerical information. In short, the less structure there is to exploit, the harder will be the computation.

Challenging as such computational problems are, they are not our primary concern here. Rather, our concerns will center on those questions for which there exists *no program at all* for producing an answer. Our interest, then, is in questions that are *logically*, rather than practically, unanswerable. So let me present some specific examples of questions in the realm of natural and human affairs that serve to motivate what's involved in identifying such logical barriers to science.

III. Limits in Nature and in Life

In order to bring the issues of limits to scientific knowledge into sharper focus, let's look at three well-known questions from the areas of physics, biology and economics.

- *Stability of the Solar System:* Certainly the most famous question of classical celestial mechanics is the N -Body Problem, which comes in many forms. One version involves N point masses moving in accordance with Newton's laws of gravitational attraction. Mathematically, the trajectories of the particles are given by the solution of the set of differential equations

$$\begin{aligned} m_j \ddot{\mathbf{r}}_j &= \sum_{i \neq j} \frac{m_i m_j (\mathbf{r}_i - \mathbf{r}_j)}{r_{ij}^3}, \\ &= \frac{\partial U}{\partial \mathbf{r}_j}, \quad i, j = 1, 2, \dots, N, \end{aligned}$$

Here m_i is the mass and \mathbf{r}_i is the position vector of the i th particle, while $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ is the euclidean distance between the i th and j th particles. The quantity

$$U = \sum_{i < j} \frac{m_i m_j}{r_{ij}},$$

is the self-potential (the negative of the potential energy of the particle system).

If we let Δ be the set of all points in R^{3n} where the above set of equations is not defined, then one way of expressing the N -Body Problem is to ask if there is some set of initial positions and velocities of the particles, such that $\mathbf{r}(t) \rightarrow \Delta$ as $t \rightarrow \infty$? One way for this to happen would be for two particles to collide, in which case $\mathbf{r}_i = \mathbf{r}_j$ for some i and j . Another way would be to have a *noncollision* singularity, in which $\mathbf{r}(t)$ approaches Δ without actually *approaching* any point in this set. In