LIMITATIONS ON OUR UNDERSTANDING OF THE BEHAVIOR OF SIMPLIFIED PHYSICAL SYSTEMS

by
Harvey M. Friedman*
Distinguished University Professor of Mathematics,
Philosophy, and Computer Science
Ohio State University
October 30, 2008

Abstract. Results going back to Turing and Gödel provide us with limitations on our ability to algorithmically decide the truth or falsity of mathematical assertions in a number of important mathematical contexts. Here we adapt some of this earlier work to very simplified mathematical models of discrete deterministic physical systems involving a few moving bodies (twelve point masses) in potentially infinite one dimensional space.

There are two kinds of such limiting results that must be carefully distinguished. Results of the first kind state the nonexistence of any algorithm for determining whether any statement among a given set of statements is true or false. Results of the second kind are much deeper and present much greater challenges. They point to specific statements A, where we can neither prove nor refute A using accepted principles of mathematical reasoning.

We give a brief survey of these limiting results. These include limiting results of the first kind: from number theory, group theory, and topology, in mathematics, and from idealized computing devices in theoretical computer science. We present a new limiting result of the first kind for simplified physical systems. We conjecture some related limiting results of the second kind, for simplified physical systems.

TABLE OF CONTENTS

1. EXAMPLES OF ALGORITHMS. Arithmetic ops, gcd, primality, factoring, solvability of equations.

We want to provide some significant context by discussing demonstrable successes before I discuss demonstrable limitations.

Long before there were any appropriate models of computation, there were actual interesting algorithms.

Schoolchildren are still taught the standard algorithms for adding, subtracting, and multiplying two integers given in base 10. They are also taught the standard division algorithm with remainder.

Algorithms for these standard arithmetic operations have been revisited in a very powerful way because of the computer revolution. There is a real need for optimizing speed as much as possible, for numerous applications. These modern algorithms take full advantage of actual circuit designs, and exploit their capacity for parallelism. See, for example, [CLR90], [BP00].

It is instructive to consider one the most famous of all algorithms - the Euclidean algorithm from Greek times, which is believed to even predate Euclid.

The purpose of this algorithm is as follows. When given two positive integers, n,m, in base 10, return the greatest common divisor of n,m in base 10. We write this as gcd(n,m).

The most obvious algorithm, when presented with n,m, maintains a number d which is the greatest integer found thus far that divides both n and m, and a number t indicating where we are in the process.

We start with t = 1 and d = 0. We determine whether t = 1 divides n,m. It does, and so we update d = 1, and also update t = 1 to t = 2.

We then determine whether $t = 2$ divides n,m. If it does, then we update $d = 1$ to $d = 2$, and update $t = 2$ to $t = 3$. If it does not, then we leave $d = 1$ alone, and update $t = 2$ to $t = 3$.

We continue in this way, updating d sporadically, until we reach (and finish with) $t = \min(n,m)$. The final value of d is the greatest common divisor of n,m.

Of course, this is about the "worst possible algorithm imaginable" in terms of computing resources. The Greeks did far better, in the following clever way.

If $n = m$, then $\gcd(n,m) = n$. So assume, say, $n < m$. Divide n into m and get a remainder $r_1 < n$. Divide r_1 into n and get a remainder $r_2 < r_1$. Divide r_2 into r_1 and get a remainder $r_3 < r_2$. Keep doing this until the remainder is 0. The divisor r that generated this 0 remainder is $\gcd(n,m)$. This is because every common divisor of n,m divides r (in fact, n,m and all r's), and r is a common divisor of n,m. Both of these facts are proved by induction, up and down the construction.

The number of steps that the Euclidean algorithm takes is known to be at most the total number of base 10 digits in the problem - an enormous improvement over the "worst" algorithm. See [Kn73], [CLR90].

The prime numbers (integers ≥ 2 that cannot be factored) have played an essential role in mathematics, and more recently, in cryptography. E.g., The Diffie-Hellman key exachange protocol, [DH76].

It is useful to know that there are approximately $n/\ln(n)$ primes below any positive integer n. I.e., the n-th prime is approximately $n/\ln(n)$. This is the so called Prime Number Theorem, due independently to Hadamard and de la Vallée Poussin, in 1896.

Here is a good candidate for the "worst" way to test whether or not a positive integer n is prime. Just divide n by all of the numbers from 2 through $n-1$ and see if you find a factor.

Primality testing has gone through a very interesting history, involving evolution in the notion of algorithm. The very best algorithms in practice have an exotic component which represents a modern paradigm - namely, the probabilistic algorithm.

The Miller Rabin primality test is a beautiful example of a probabilistic algorithm. See [Mi76], [Ra80], [CLR90]. A set $S[n] \subseteq [1,n-1]$ is cleverly associated with any odd positive integer n, with the following properties:

i. If n is prime then S[n] is empty.
ii. If n is composite then S[n] has at least (n-1)/2 elements.
iii. It is highly efficient to test for membership in S[n].

We can now "test" whether n is prime, highly efficiently. Generate a lot of numbers in [1,n-1], "randomly". Say $a_1,...,a_{200}$. Check for membership in S[n]. If at least one of them lies in S[n] then we know that n is composite. If none of them lie in S[n], then almost certainly n is prime. Why? If it were composite then you would have a misleading outcome 200 times in a row! Chance of that is 2^{-200}.

You want more "certainty"? Then use a bigger number of trials than 200.

This algorithm raises a number of deep foundational, philosophical, and metaphysical issues surrounding randomness and truth - in a particularly rich and focused way. E.g., what do we mean by random trials? What kind of truth is involved here? How do we actually generate random numbers?

We now come to factoring. It is widely believed that factoring a positive integer is "hard" in the sense that it is going to take an unfeasible amount of time, generally, to factor positive integers randomly chosen with, say, hundreds of digits.

There has been a huge success in building up a theory and practice of cryptographic schemes based on the assumption that factoring positive integers chosen randomly is too difficult for anyone armed with anything like existing computer power. See [DH76], [NOV97], [Go01,04].

Nevertheless, there are methods that are considerably better than the "worst" - which is merely to try everything. There is a family of methods called the sieve method, which originated in ancient times. This continues to be an active area of research. See [MP96].

We now come to the solvability of equations. Let me concentrate on a single polynomial equation with integer coefficients. There are three key parameters: the number of variables, the degree, and the size of the coefficients.

1. There is an algorithm for testing solvability with real or complex number unknowns. [Co75], [CJ98], [Ta51].

2. There is an algorithm for testing solvability for degree 2, with integer or rational (or real or complex number) unknowns. [GS81].

3. There is no algorithm for testing solvability for 9 nonnegative integer unknowns. The same negative result holds with integer unknowns, but apparently the number of unknowns needs to be raised somewhat. http://logic.pdmi.ras.ru/Hilbert10/index.html, [Jo82], [Ma93].

4. There is no algorithm for testing solvability for degree 4, with integer unknowns. [Ma93].

5. Nobody knows if there is an algorithm for testing solvability with rational unknowns. [Maz94].

6. Nobody knows if there is an algorithm for testing solvability for degree 3, with 2 rational unknowns. There is an algorithm for degree 3, with 2 integer unknowns. [Maz94].

There continues to be a lot of work trying to deal with the theoretical and practical issues surrounding the implementation of 1 and fragments. E.g., see [BPR03].

2. NO ALGORITHMS. Robust model of computation. Behavior of abstract machines, solvability of equations.

In order to establish that there is no algorithm for determining whether a property holds of inputs from a class, or no algorithm for returning information related to inputs from a class, one relies on a standard robust model of computation.

The first presentation of such a model was by Alan Turing. It was rather specialized, and considerable effort went into establishing its robustness. See [Tu37], [Da65], [Da00].

Turing's formal computing devices were, on the surface, rather limited and restricted. So it was not entirely clear what would happen if one enlarged them without destroying the general features that they possessed that made them obviously algorithmic.

After much work along these lines, it became generally accepted that an extremely robust analysis had been given. In particular, the notions of partial recursive and recursive functions on nonnegative integers and on finite strings from a finite alphabet, as well as of recursively enumerable and recursive sets of nonnegative integers and finite strings from a finite alphabet, were fundamental, and "correctly analyze" clear informal notions.

In fact, this feeling was codified by Alonzo Church into what is called "Church's Thesis".

I am one of the few people who has some optimism that there is some strikingly simple condition that can be placed on "computable" or "algorithm", which, in some way, "obviously" encompasses all "imaginable" computations and algorithms, and which is then proved to be equivalent to the Turing model. Then we have an indisputable "proof" of Church's Thesis. See [COWJ06], [Si97].

The original algorithmic unsolvability result was the unsolvability of the "halting problem". See, e.g., [Da82]. This is normally stated as follows: determine whether a given Turing machine TM halts at a given input.

This formulation refers to an enumeration of the Turing machines, TM[0], TM[1], ... However, we can avoid this, and give a much cleaner formulation:

For each specific TM, determine whether TM halts at a given input x.

Note that this cleaner formulation asks for something weaker than the usual formulation. However, below we sketch a proof that even this weaker formulation is impossible.

For specificity, we will assume that our TMs take nonnegative integers as inputs and outputs. We use N for the set of all nonnegative integers.

THEOREM 2.1. There is a TM with the following property. There is no algorithm for determining whether TM halts at a given nonnegative integer input x.

Proof: This statement avoids enumerations of TMs. However, the proof uses the existence of a "reasonable" enumeration of TMs by nonnegative integers, TM[0], TM[1], ...

Let TM be a Turing machine that runs as follows. Let x ∈ N be the input. Set up TM[x] and run (i.e., simulate the running of) TM[x] at input x. If TM[x] halts at x, with output y, then the output of TM is y. If TM[x] does not halt at x, then TM does not halt at x.

We claim that there is no algorithm for determining whether TM halts at a given x ∈ N. Suppose there is such an algorithm, A. Then we can construct a new TM' whose behavior at input x differs from the behavior of TM[x] at input x. This is impossible since TM' is itself some TM[x].

Specifically, TM' at input x runs as follows. First use algorithm A to determine if TM[x] halts at input x. If it does, say with output y, then output y+1. If it does not, then output 0. QED

However theoretically satisfying this kind of thing might be to many of us, the down to earth mathematician would like (or dislike!) to see more. I.e., a no algorithm result that is considerably closer to their mathematical interests - one that may impact their work.

There is a shortage of such negative results along these lines. I have already mentioned the no algorithm results concerning solvability of polynomial equations in 9 nonnegative integer unknowns.

Another important example is that there is no algorithm for testing whether a given finitely generated group is trivial. See [Ad55], [No55], [Ra58], [Po08].

A related example in topology is: there is no algorithm for testing whether two closed 4-manifolds are homeomorphic. [Mar58], [Po08].

3. SIMPLIFIED PHYSICAL SYSTEMS. Linear Order Systems. No algorithm for determining Boundedness.

We now describe a very simplified kind of discrete physical system, based on finitely many bodies. We call these Linear Order Systems.

We prove that there is no way to analyze the behavior of such systems over eternity, in the following sense. There is no algorithm that determines, for any given initial configuration of the bodies, whether the evolution is contained in a finite region of space.

Moreover, the result is proved in the following strong form:

There is a specific Linear Order System with 12 bodies such that there is no algorithm that determines, for any given initial configuration of the 12 bodies, whether the evolution is contained in a finite region of space.

There has been considerable work on no algorithm results for simple machines. Below, we indicate how this work builds on existing work, and takes us into exciting new directions. E.g., see [Ba01], [NW07], [Ne08].

There are obvious extensions of Linear Order Systems to Planar Systems and Space Systems. We expect that the number of bodies required for the analogous results will be considerably smaller than 12.

Linear Order Systems involve quite general laws of motion, with few bodies. It will be important to extend the work to cover much more restricted laws of motion, again with a reasonable small number of bodies. The ultimate goal is to incorporate more and more features of real physical systems.

We now present the Linear Order Systems.

Space is identified with the integer number line

$$\ldots,-2,-1,0,1,2,\ldots$$

We will not assume any reference point (e.g., 0).

Time is identified with the nonnegative integer number line

$$0,1,2,...$$

REMARK. An interesting direction of research is to allow arbitrary integers as times. We suspect that there are very interesting no algorithm results concerning what configurations can occur in a given Linear Order System.

In an n body Linear Order System, we have n bodies, $B_1,...,B_n$, where $n \geq 1$. We write

$$B_i[t],\ 0 \leq i \leq n,$$

for the position of body B_i at time t. Each $B_i[t]$ is an integer.

Thus the initial configuration of the n bodies is given by

$$B_1[0],...,B_n[0].$$

Motion is deterministic, and very restricted. At each time > 0, each body will move to the left one unit, to the right one unit, or stay at the same position. I.e., we have

$$B_i[t+1] - B_i[t] \in \{-1,0,1\}.$$

where the value of $B_i[t+1] - B_i[t]$ generally depends on both i and t.

In a Linear Order System, motion is completely specified in terms of the relative order of the n bodies.

The relative order of the n bodies $B_1,...,B_n$ at time t is just the relative order of the n integers $B_1[t],...,B_n[t]$. This is merely a tabulation of which of these integers is less than which others. Obviously, this information also tells us which of these integers is equaled to which others, and which of these integers is greater than which others.

There is a standard, compact way of specifying the relative order of $B_1[t],...,B_n[t]$. It is by a listing

$$C_1[t]\ ?\ C_2[t]\ ?\ ...\ ?\ C_n[t]$$

where each ? is either < or =, and $C_1,...,C_n$ is a permutation of the bodies $B_1,...,B_n$. (In general, the ? vary in the above). A useful convention is that if we see

$$C_i[t] = C_j[t]$$

adjacently in the listing, then we require that i < j.

How can we justify this setup physically?

Think of each body as, e.g., constantly transmitting radiation with its unique signature. So each body is constantly aware of the presence of the other n-1 bodies, and knows which is which. In particular, each body is constantly aware of the presence of the other bodies to the limited extent of: is it to my left, to my right, or right here (at the same position)?

Furthermore, assume that each body can sense, given what it receives, the relative distance of each of the bodies from itself - at least to the extent of knowing which ones are closer or further away than others. E.g., by the relative strength of the received signals.

From this information, each body can obviously infer the relative order of ALL of the n bodies along one dimensional space.

Each body then moves, by 0 or 1 units in each direction, according to this information only.

We are considering arbitrary rules of this kind, so that there is no assumption that different bodies may react in the same way to the same information. This makes sense because we are thinking of the bodies as being of different kinds, each generated signals with its characteristic signature.

ONE BODY LINEAR ORDER SYSTEMS. There are exactly three one body Linear Order Systems. Always moving to the right, always moving to the left, always not moving. For any initial configuration, the first two are unbounded, and the third is bounded.

TWO BODY LINEAR ORDER SYSTEMS. There are three relative orders.

$B_1 = B_2$.
$B_1 < B_2$.
$B_2 < B_s$.

B_1,B_2 have to be assigned a number from -1,0,1 under each of the relative orders. There are 9 possibilities for this, and so we obtain $9^2 = 81$ two body Linear Order Systems. It should be quite manageable - and interesting - to give a complete analysis of which of these laws lead to boundedness under which initial configurations, especially with the help of a computer to dispense with trivial cases, and to provide the relevant decision procedures.

THREE BODY LINEAR ORDER SYSTEMS. There are 13 relative orders.

$B_1 < B_2 < B_3$.
$B_1 < B_3 < B_2$.
$B_2 < B_1 < B_3$.
$B_2 < B_3 < B_1$.
$B_3 < B_1 < B_2$.
$B_3 < B_2 < B_1$.
$B_1 = B_2 < B_3$.
$B_1 = B_3 < B_2$.
$B_2 = B_3 < B_1$.
$B_1 < B_2 = B_3$.
$B_2 < B_1 = B_3$.
$B_3 < B_1 = B_2$.
$B_1 = B_2 = B_3$.

B_1,B_2,B_3 have to be assigned a number from -1,0,1 under each of the relative orders. This consists of 39 numbers drawn from {-1,0,1}. Thus there are 3^{39} three body Linear Order Systems. The challenge is to construct an efficient algorithm that determines boundedness for any three body Linear Order System under any given initial configuration of the three bodies. We conjecture that such an algorithm exists.

This is already quite difficult for three bodies. The challenge gets far more difficult as we go from four on.

TWELVE BODY LINEAR ORDER SYSTEMS. Our theorem asserts that a boundedness test is impossible for at least one twelve body Linear Order System.

Officially, any 12 body Linear Order System involves 12 numbers from {-1,0,1} assigned to each relative order for $B_1,....,B_{12}$. The number of such relative orders is rather large. E.g., considerably more than 12!. So each law of motion for 12 bodies officially requires a table of considerably more than 12(12!) numbers from -1,0,1.

However, there are many opportunities for providing far shorter descriptions of Linear Order Systems.

A partial relative order for n bodies, is a set of conditions of the form

$$B_i = B_j$$
$$B_i < B_j$$
$$B_i \leq B_j$$
$$B_i \neq B_j$$

where $1 \leq i,j \leq n$. We can now write laws of motion in the following form.

Partial relative order #1. B_1 moves ?,...,B_n moves ?
Partial relative order #2. B_1 moves ?,...,B_n moves ?
...
Partial relative order #m. B_1 moves ?,...,B_n moves ?
Otherwise. B_1 moves ?,...,B_n moves ?.

Here ? ∈ {-1,0,1}, and they generally vary on each line above. We demand that these partial relative orders are mutually exclusive.

Let us call the total number of partial relative orders used, the presentation complexity.

Our intractable 12 body law of motion has presentation complexity far lower than the theoretical limit, which is greater than 12!, and probably can be massaged to have still lower presentation complexity - perhaps down to a few dozen.

Before getting into the details, we make some basic remarks on the relation between this work and previous work on abstract machines.

Interpretations of the no algorithms results on Turing machines, in terms of the motion of bodies, involves having arbitrarily large numbers of bodies, with no a priori bound. This is because they involve arbitrarily long strings of bits, or symbols from a finite alphabet.

Linear Order Systems are more closely related to register machines. But in Linear Order Systems, all abstract state and all flow of control must emanate from the configuration of the bodies alone - not from an ordered list of instructions and list of abstract states.

Of course, we rely heavily on previous work on Turing machines.

We anticipate a steady stream of restrictions on Linear Order Systems (such as the one mentioned earlier) and also the introduction of other models which take more and more aspects of actual physical systems into account.

The mathematics will then get increasingly deeper and more involved, getting further away from the earlier no algorithms work, which was not primarily motivated by this kind of direct consideration of (simplified) physical systems.

4. SIMULATION OF TURING MACHINES.

We now show how to simulate the action of any Turing machine TM with two symbols and n states, by a Linear Order System, in the present sense, with considerably fewer than n bodies.

Because we are focused only on the determination of boundedness at arbitrary initializations, we shall see that we need only simulate the action of Turing machines without initial states and without halting states. In particular, all computations will go on forever.

We will work entirely with TMs having two distinct symbols 0,1, and n states $q_1,...,q_n$, $n \geq 1$. Here 0 is the blank symbol. We assume $0,1,q_1,...,q_n$ are distinct. We use a standard quintuple formulation of TMs.

The configurations take the form

$$x\ q\ y$$

where x is an infinite bit sequence indexed by {...,2,1,0}, and y is an infinite bit sequence indexed by {0,1,2,...}, where all but finitely many terms of x,y are 0's. Also $q \in \{q_1,...,q_n\}$.

TM operates by a complete set of 2n quintuples of the two forms

$$q_i\ c\ d\ L\ q_j$$
$$q_i\ c\ d\ R\ q_j$$

where $1 \leq i,j \leq n$, L = left, R = right, and $c,d \in \{0,1\}$. In a TM, it is required that there be exactly one instruction starting with any q_i,c, where $1 \leq i \leq n$ and $c \in \{0,1\}$.

The configuration x q y indicates that we are in state q, the reading head is over the first symbol of y, the tape to the left of the reading head is x, and the tape at and to the right of the reading head is y.

These instructions are executed on configurations x q y in the standard fashion.

q_i c d L q_j (q_i c d R q_j) can be executed on x q y if and only if q = q_i, and the first symbol of y is c. In this case, we overwrite the first symbol of y with d, move to the left (right), and go into state qj.

Inputs for TMs are finitary. They are of the form

$$u\ q\ v$$

where $u,v \in \{0,1\}^*$, and $q \in \{q_1,...,q_n\}$. Computation proceeds on the corresponding configurations. The corresponding configuration for u q v is obtained from uv by adding an infinite series of 0's to the left of u and to the right of v, and going into state q. This defines a configuration. Here $\{0,1\}^*$ is the set of all finite (possibly empty) strings from {0,1}.

We do not have outputs because we are assuming that the TMs have no halting instructions. However, we will have need to consider TMs with halting instructions when we use [Ne08]. There we will only need to consider halting.

There is a crucial pair of nonnegative integer associated with every TM configuration. For any configuration x q y, we let |x| be the base 2 integer represented by x, and |rev(y)| be the base 2 integer represented by rev(y) = the reverse of y. Because x and rev(y) are indexed by {...,2,1,0}, and are almost all 0's, this base 2 integers exist.

We will first simulate the action of TM by what we call an extended linear order system, or ELOS. For our ELOS, we will use three bodies, B_1,B_2,B_3, which are, however, augmented with some additional apparatus that is strictly forbidden in a LOS:

i. 0 as a reference point.
ii. Relative orders will use 0. I.e., they take the form $0 \alpha C_1 \beta C_2 \gamma C_3$, where C_1,C_2,C_3 is a permutation of B_1,B_2,B_3, and $\alpha,\beta \in \{<,=\}$.
iii. The n abstract states $q_1,...,q_n$.
iv. Three special control states S_1,S_2,S_3. Here $S_1 \in \{0,...,5\}$, $S_2,S_3 \in \{0,1\}$.
v. Thus the "global state" of the ELOS will be the relative order (in the sense of ii), together with the q state (one of the n abstract states $q_1,...,q_n$), and the S_1,S_2,S_3 states.
vi. We assign -1,0,1 for movement of bodies B_1,B_2,B_3, as in a LOS.

The simulation will be instruction by instruction, where at any initialization of the TM, the TM will be bounded if and only if the ELOS is bounded at the corresponding initialization.

We will then eliminate the use of q,S_1,S_2,S_3 and the reference point 0, in order to obtain a LOS.

We will simulate the effect of any TM instruction I of the form

$$q_i \text{ c d L } q_j$$
$$q_i \text{ c d R } q_j$$

with a corresponding set SIM(I) of ELOS laws of motion. These SIM(I) will only be partial laws of motion in that they will specify the motion only at certain global states - the ones that can arise. We can obviously extend to the remaining global states arbitrarily. Here "SIM" abbreviates "simulation".

For any TM configuration C = x q_i y, we associate the corresponding ELOS global state GS(C) =

$B_1 = |x|$.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 =$ the first bit of y if y is nonempty; 0 otherwise.
$S_3 = |rev(y)| \bmod 2$.
$q = q_i$.

Note the useful numerical equivalent for S_3, given above.

Suppose TM has input C. Let I be the unique TM instruction that applies to C. Let I(C) be the input resulting from applying I to C.

We require that the ELOS laws of motion SIM(I), starting with the ELOS global state GS(C), stabilizes at global state GS(I(C)).

The above values of S_1,S_2,S_3,q tell us "to start performing the unique instruction that applies to x q_i y". The above values of S_3,q_i tell us that "we are simulating the unique instruction that applies to x q_i y". The simulation will end with $B_3 = S_1 = S_2 = 0$. Then the updated values of S_3,q will trigger the simulation of the next instruction that applies. Note that it is possible that the next instruction is the same as the current instruction.

Below $B_i\uparrow$ means "increment B_i", and $B_i\downarrow$ means "decrement B_i". Here "increment" means add 1, and "decrement" means subtract 1.

case 1. SIM(I) for I = q_i 0 0 L q_j at input x q_i y, where $|rev(y)|$ is even.

We start the simulation with

$B_1 = |x|$.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = 0$.
$q = q_i$.

We must transition to

$B_1 = |x|$ div 2.
$B_2 = 2|rev(y)| + (|x|$ mod 2).
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x|$ mod 2.
$q = q_j$.

where S_3,q get updated only at the last step of the simulation. This guarantees that the simulations of the various instructions in TM do not interfere with each other. This requirement will be met for this and the remaining seven cases.

1.1. $B_3 < B_1$, $S_1 = 0$, $S_3 = 0$, $q = q_i \Rightarrow B_1\downarrow$, $B_3\uparrow$.
1.2. $B_3 = B_1$, $S_1 = 0$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 1$.
1.3. $B_3 > B_1$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 1$, $S_2 = 1$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|$.
$B_3 = |x|$ div 2 + ($|x|$ mod 2).
$S_1 = 1$.
$S_2 = |x|$ mod 2.
$S_3 = 0$.
$q = q_i$.

1.4. $B_3 > 0$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow B_3\downarrow$.
1.5. $B_3 = 0$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 2$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 2$.
$S_2 = |x|$ mod 2.
$S_3 = 0$.
$q = q_i$.

1.6. $B_3 < B_2$, $S_1 = 2$, $S_3 = 0$, $q = q_i \Rightarrow B_3\uparrow$.
1.7. $B_3 = B_2$ $S_1 = 2$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 3$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|$.
$B_3 = |rev(y)|$.
$S_1 = 3$.
$S_2 = |x|$ mod 2.
$S_3 = 0$.
$q = q_i$.

1.8. $B_3 > 0$, $S_1 = 3$, $S_3 = 0$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\downarrow$.
1.9. $B_3 = 0$, $S_1 = 3$, $S_2 = 0$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 0$, $q = q_j$.
1.10. $B_3 = 0$, $S_1 = 3$, $S_2 = 1$, $S_3 = 0$, $q = q_i \Rightarrow$
$B_2\uparrow$, $S_1 = 0$, $S_2 = 0$, $S_3 = 1$, $q = q_j$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = 2|\text{rev}(y)| + (|x| \bmod 2)$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x| \bmod 2$.
$q = q_j$.

case 2. SIM(I) for $I = q_i\ 0\ 1\ L\ q_j$ at input $x\ q_i\ y$, where $|\text{rev}(y)|$ is even.

We start the simulation with

$B_1 = |x|$.
$B_2 = |\text{rev}(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = 0$.
$q = q_i$.

We must transition to

$B_1 = |x|$ div 2.
$B_2 = 2(|\text{rev}(y)|+1) + (|x| \bmod 2)$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x| \bmod 2$.
$q = q_j$.

1.1. $B_3 < B_1$, $S_1 = 0$, $S_3 = 0$, $q = q_i \Rightarrow B_1\downarrow$, $B_3\uparrow$.
1.2. $B_3 = B_1$, $S_1 = 0$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 1$.
1.3. $B_3 > B_1$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 1$, $S_2 = 1$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |\text{rev}(y)|$.
$B_3 = |x|$ div $2 + (|x| \bmod 2)$.
$S_1 = 1$.
$S_2 = |x| \bmod 2$.
$S_3 = 0$.

$q = q_i$.

1.4. $B_3 > 0$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow B_3\downarrow$.
1.5. $B_3 = 0$, $S_1 = 1$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 2$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 2$.
$S_2 = |x|$ mod 2.
$S_3 = 0$.
$q = q_i$.

1.6. $B_3 < B_2$, $S_1 = 2$, $S_3 = 0$, $q = q_i \Rightarrow B_3\uparrow$.
1.7'. $B_3 = B_2$ $S_1 = 2$, $S_3 = 0$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\uparrow$, $S_1 = 3$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|+1$.
$B_3 = |rev(y)|+1$.
$S_1 = 3$.
$S_2 = |x|$ mod 2.
$S_3 = 0$.
$q = q_i$.

1.8. $B_3 > 0$, $S_1 = 3$, $S_3 = 0$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\downarrow$.
1.9. $B_3 = 0$, $S_1 = 3$, $S_2 = 0$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 0$, $q = q_j$.
1.10. $B_3 = 0$, $S_1 = 3$, $S_2 = 1$, $S_3 = 0$, $q = q_i \Rightarrow$
$B_2\uparrow$, $S_1 = 0$, $S_2 = 0$, $S_3 = 1$, $q = q_j$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = 2(|rev(y)|+1) + (|x|$ mod 2).
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x|$ mod 2.
$q = q_j$.

case 3. SIM(I) for $I = q_i\ 1\ 0\ L\ q_j$ at input $x\ q_i\ y$, where $|rev(y)|$ is odd.

We start the simulation with

$B_1 = |x|$.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.

$S_3 = 1.$
$q = q_i.$

We must transition to

$B_1 = |x| \text{ div } 2.$
$B_2 = 2(|\text{rev}(y)|-1) + (|x| \bmod 2).$
$B_3 = 0.$
$S_1 = 0.$
$S_2 = 0.$
$S_3 = |x| \bmod 2.$
$q = q_j.$

3.1. $B_3 < B_1,\ S_1 = 0,\ S_3 = 1,\ q = q_i \Rightarrow B_1\downarrow,\ B_3\uparrow.$
3.2. $B_3 = B_1,\ S_1 = 0,\ S_3 = 1,\ q = q_i \Rightarrow S_1 = 1.$
3.3. $B_3 > B_1,\ S_1 = 1,\ S_3 = 1,\ q = q_i \Rightarrow S_1 = 1,\ S_2 = 1.$
These stabilize with
$B_1 = |x| \text{ div } 2.$
$B_2 = |\text{rev}(y)|.$
$B_3 = |x| \text{ div } 2 + (|x| \bmod 2).$
$S_1 = 1.$
$S_2 = |x| \bmod 2.$
$S_3 = 1.$
$q = q_i.$

3.4. $B_3 > 0,\ S_1 = 1,\ S_3 = 1,\ q = q_i \Rightarrow B_3\downarrow.$
3.5. $B_3 = 0,\ S_1 = 1,\ S_3 = 1,\ q = q_i \Rightarrow S_1 = 2.$
These stabilize with
$B_1 = |x| \text{ div } 2.$
$B_2 = |\text{rev}(y)|.$
$B_3 = 0.$
$S_1 = 2.$
$S_2 = |x| \bmod 2.$
$S_3 = 1.$
$q = q_i.$

3.6. $B_3 < B_2,\ S_1 = 2,\ S_3 = 1,\ q = q_i \Rightarrow B_3\uparrow.$
3.7. $B_3 = B_2\ S_1 = 2,\ S_3 = 1,\ q = q_i \Rightarrow B_2\downarrow,\ B_3\downarrow,\ S_1 = 3.$
These stabilize with
$B_1 = |x| \text{ div } 2.$
$B_2 = |\text{rev}(y)|-1.$
$B_3 = |\text{rev}(y)|-1.$
$S_1 = 3.$
$S_2 = |x| \bmod 2.$
$S_3 = 1.$
$q = q_i.$

3.8. $B_3 > 0$, $S_1 = 3$, $S_3 = 1$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\downarrow$.
3.9. $B_3 = 0$, $S_1 = 3$, $S_2 = 0$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 0$, $S_3 = 0$, $q = q_j$.
3.10. $B_3 = 0$, $S_1 = 3$, $S_2 = 1$, $S_3 = 1$, $q = q_i \Rightarrow$
$B_2\uparrow$, $S_1 = 0$, $S_2 = 0$, $q = q_j$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = 2(|rev(y)|-1) + (|x|$ mod 2).
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x|$ mod 2.
$q = q_j$.

case 4. SIM(I) for I = q_i 1 1 L q_j at input x q_i y, where |rev(y)| is odd.

We start the simulation with

$B_1 = |x|$.
$B_2 = |rev(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = 1$.
$q = q_i$.

We must transition to

$B_1 = |x|$ div 2.
$B_2 = 2|rev(y)| + (|x|$ mod 2).
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x|$ mod 2.
$q = q_j$.

3.1. $B_3 < B_1$, $S_1 = 0$, $S_3 = 1$, $q = q_i \Rightarrow B_1\downarrow$, $B_3\uparrow$.
3.2. $B_3 = B_1$, $S_1 = 0$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 1$.
3.3. $B_3 > B_1$, $S_1 = 1$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 1$, $S_2 = 1$.
These stabilize with
$B_1 = |x|$ div 2.
$B_2 = |rev(y)|$.
$B_3 = |x|$ div 2 + ($|x|$ mod 2).
$S_1 = 1$.
$S_2 = |x|$ mod 2.
$S_3 = 1$.

$q = q_i$.

3.4. $B_3 > 0$, $S_1 = 1$, $S_3 = 1$, $q = q_i \Rightarrow B_3\downarrow$.
3.5. $B_3 = 0$, $S_1 = 1$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 2$.
These stabilize with
$B_1 = |x| \text{ div } 2$.
$B_2 = |\text{rev}(y)|$.
$B_3 = 0$.
$S_1 = 2$.
$S_2 = |x| \bmod 2$.
$S_3 = 1$.
$q = q_i$.

3.6. $B_3 < B_2$, $S_1 = 2$, $S_3 = 1$, $q = q_i \Rightarrow B_3\uparrow$.
3.7'. $B_3 = B_2$ $S_1 = 2$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 3$.
These stabilize with
$B_1 = |x| \text{ div } 2$.
$B_2 = |\text{rev}(y)|$.
$B_3 = |\text{rev}(y)|$.
$S_1 = 3$.
$S_2 = |x| \bmod 2$.
$S_3 = 1$.
$q = q_i$.

3.8. $B_3 > 0$, $S_1 = 3$, $S_3 = 1$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\downarrow$.
3.9. $B_3 = 0$, $S_1 = 3$, $S_2 = 0$, $S_3 = 1$, $q = q_i \Rightarrow S_1 = 0$, $S_3 = 0$, $q = q_j$.
3.10. $B_3 = 0$, $S_1 = 3$, $S_2 = 1$, $S_3 = 1$, $q = q_i \Rightarrow$
$B_2\uparrow$, $S_1 = 0$, $S_2 = 0$, $q = q_j$.
These stabilize with
$B_1 = |x| \text{ div } 2$.
$B_2 = 2|\text{rev}(y)| + (|x| \bmod 2)$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = |x| \bmod 2$.
$q = q_j$.

case 5. SIM(I) for $I = q_i\ 0\ 0\ R\ q_j$ at input $x\ q_i\ y$, where $|\text{rev}(y)|$ is even.

We start the simulation with

$B_1 = |x|$.
$B_2 = |\text{rev}(y)|$.
$B_3 = 0$.
$S_1 = 0$.

$S_2 = 0$.
$S_3 = 0$.
$q = q_i$.

We must transition to

$B_1 = 2|x|$.
$B_2 = rev(y)$ div 2.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = (rev(y)$ div 2) mod 2.
$q = q_j$.

5.1. $B_3 < B_1, S_1 = 0, S_3 = 0, q = q_i \Rightarrow B_3\uparrow$.
5.2. $B_3 = B_1, S_1 = 0, S_3 = 0, q = q_i \Rightarrow S_1 = 1$.
This stabilizes with
$B_1 = |x|$.
$B_2 = |rev(y)|$.
$B_3 = |x|$.
$S_1 = 1$.
$S_2 = 0$.
$S_3 = 0$
$q = q_i$.

5.3. $B_3 > 0, S_1 = 1, S_3 = 0, q = q_i \Rightarrow B_2\uparrow, B_3\downarrow$.
5.4. $B_3 = 0, S_1 = 1, S_3 = 0, q = q_i \Rightarrow S_1 = 2$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = |rev(y)|$.
$B_3 = |x|$.
$S_1 = 2$.
$S_2 = 0$.
$S_3 = 0$
$q = q_i$.

5.5. $B_3 < B_2, S_1 = 2, S_3 = 0, q = q_i \Rightarrow B_2\downarrow, B_3\uparrow$.
5.6. $B_3 \geq B_2, S_1 = 2, S_3 = 0, q = q_i \Rightarrow S_1 = 3$.
5.7. $B_3 > 0, S_1 = 3, S_3 = 0, q = q_i \Rightarrow B_3\downarrow$.
5.8. $B_3 = 0, S_1 = 3, S_3 = 0, q = q_i \Rightarrow S_1 = 4$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = |rev(y)|$ div 2.
$B_3 = 0$.
$S_1 = 4$.
$S_2 = 0$.
$S_3 = 0$

$q = q_i$.

5.9. $B_3 < B_2$, $S_1 = 4$, $S_3 = 0$, $q = q_i \Rightarrow B_2\downarrow$, $B_3\uparrow$.
5.10. $B_3 = B_2$, $S_1 = 4$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 5$.
5.11. $B_3 < B_2$, $S_1 = 4$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 5$, $S_2 = 1$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = (|\text{rev}(y)| \text{ div } 2) \text{ div } 2$.
$B_3 = ((|\text{rev}(y)| \text{ div } 2) \text{ div } 2) + ((|\text{rev}(y)| \text{ div } 2) \text{ mod } 2)$.
$S_1 = 4$.
$S_2 = (\text{rev}(y) \text{ div } 2) \text{ mod } 2$.
$S_3 = 0$.
$q = q_i$.

5.12. $B_3 > 0$, $S_1 = 5$, $S_3 = 0$, $q = q_i \Rightarrow B_2\uparrow$, $B_3\downarrow$.
5.13. $B_3 = 0$, $S_1 = 5$, $S_2 = 0$, $S_3 = 0$, $q = q_i \Rightarrow S_1 = 0$, $q = q_j$.
5.14. $B_3 = 0$, $S_1 = 5$, $S_2 = 1$, $S_3 = 0$, $q = q_i \Rightarrow$
$S_1 = 0$, $S_2 = 0$, $S_3 = 1$, $q = q_j$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = |\text{rev}(y)| \text{ div } 2$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = (\text{rev}(y) \text{ div } 2) \text{ mod } 2$.
$q = q_j$.

case 6. SIM(I) for $I = q_i\ 0\ 1\ R\ q_j$ at input $x\ q_i\ y$, where $|\text{rev}(y)|$ is even.

We start the simulation with

$B_1 = |x|$.
$B_2 = |\text{rev}(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = 0$.
$q = q_i$.

We must transition to

$B_1 = 2|x| + 1$.
$B_2 = |\text{rev}(y)| \text{ div } 2$.
$B_3 = 0$.
$S_1 = 0$.

$S_2 = 0$.
$S_3 = (\mathrm{rev}(y) \text{ div } 2) \bmod 2$.
$q = q_j$.

We need only change 5.4 to 5.4' below, by adding $B_1\uparrow$ on the right side, in order to change $2|x|$ to $2|x|+1$.

5.4'. $S_1 = 1, B_3 = 0 \Rightarrow S_1 = 2, B_1\uparrow$.

case 7. SIM(I) for I = q_i 1 0 R q_j at input x q_i y, where $|\mathrm{rev}(y)|$ is odd.

We start the simulation with

$B_1 = |x|$.
$B_2 = |\mathrm{rev}(y)|$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = 1$.
$q = q_i$.

We must transition to

$B_1 = 2|x|$.
$B_2 = \mathrm{rev}(y) \text{ div } 2$.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = (\mathrm{rev}(y) \text{ div } 2) \bmod 2$.
$q = q_j$.

7.1. $B_3 < B_1, S_1 = 0, S_3 = 1, q = q_i \Rightarrow B_3\uparrow$.
7.2. $B_3 = B_1, S_1 = 0, S_3 = 1, q = q_i \Rightarrow S_1 = 1$.
This stabilizes with
$B_1 = |x|$.
$B_2 = |\mathrm{rev}(y)|$.
$B_3 = |x|$.
$S_1 = 1$.
$S_2 = 0$.
$S_3 = 1$.
$q = q_i$.

7.3. $B_3 > 0, S_1 = 1, S_3 = 1, q = q_i \Rightarrow B_2\uparrow, B_3\downarrow$.
7.4. $B_3 = 0, S_1 = 1, S_3 = 1, q = q_i \Rightarrow S_1 = 2$.
This stabilizes with
$B_1 = 2|x|$.

$B_2 = |\text{rev}(y)|$.
$B_3 = 0$.
$S_1 = 2$.
$S_2 = 0$.
$S_3 = 1$.
$q = q_i$.

7.5. $B_3 < B_2, S_1 = 2, S_3 = 1, q = q_i \Rightarrow B_2\downarrow, B_3\uparrow$.
7.6. $B_3 \geq B_2, S_1 = 2, S_3 = 1, q = q_i \Rightarrow S_1 = 3$.
7.7. $B_3 > 0, S_1 = 3, S_3 = 1, q = q_i \Rightarrow B_3\downarrow$.
7.8. $B_3 = 0, S_1 = 3, S_3 = 1, q = q_i \Rightarrow S_1 = 4$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = |\text{rev}(y)|$ div 2.
$B_3 = 0$.
$S_1 = 4$.
$S_2 = 0$.
$S_3 = 1$.
$q = q_i$.

7.9. $B_3 < B_2, S_1 = 4, S_3 = 1, q = q_i \Rightarrow B_2\downarrow, B_3\uparrow$.
7.10. $B_3 = B_2, S_1 = 4, S_3 = 1, q = q_i \Rightarrow S_1 = 5$.
7.11. $B_3 < B_2, S_1 = 4, S_3 = 1, q = q_i \Rightarrow S_1 = 5, S_2 = 1$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = (|\text{rev}(y)|$ div 2) div 2.
$B_3 = ((|\text{rev}(y)|$ div 2) div 2) + ((|rev(y)| div 2) mod 2).
$S_1 = 4$.
$S_2 = (\text{rev}(y)$ div 2) mod 2.
$S_3 = 1$.
$q = q_i$.

7.12. $B_3 > 0, S_1 = 5, S_3 = 1, q = q_i \Rightarrow B_2\uparrow, B_3\downarrow$.
7.13. $B_3 = 0, S_1 = 5, S_2 = 0, S_3 = 1, q = q_i \Rightarrow S_1 = 0, S_3 = 0, q = q_j$.
7.14. $B_3 = 0, S_1 = 5, S_2 = 1, S_3 = 1, q = q_i \Rightarrow$
$S_1 = 0, S_2 = 0, q = q_j$.
This stabilizes with
$B_1 = 2|x|$.
$B_2 = |\text{rev}(y)|$ div 2.
$B_3 = 0$.
$S_1 = 0$.
$S_2 = 0$.
$S_3 = (|\text{rev}(y)|$ div 2) mod 2.
$q = q_j$.

case 8. SIM(I) for I = q_i 1 1 R q_j at input x q_i y, where |rev(y)| is odd.

We start the simulation with

B_1 = |x|.
B_2 = |rev(y)| div 2.
B_3 = 0.
S_1 = 0.
S_2 = 0.
S_3 = 1.
q = q_i.

We must transition to

B_1 = 2|x| + 1.
B_2 = |rev(y)| div 2.
B_3 = 0.
S_1 = 0.
S_2 = 0.
S_3 = (|rev(y)| div 2) mod 2.
q = q_j.

We need only change 7.4 to 7.4' below, by adding $B_1\uparrow$ on the right side, in order to change 2|x| to 2|x|+1.

7.4. B_3 = 0, S_1 = 1, S_3 = 1, q = q_i ⇒ $B_1\uparrow$, S_1 = 2.

This completes the eight cases.

We can now combine these cases by taking the union of the SIM(I), for instructions I in TM. There will be exactly one instruction for each pair (u,q_i), where u ∈ {0,1}. As indicated earlier, the various simulations SIM(I), I ∈ TM, do not interfere with each other, and so we get a global simulation by an ELOS of the action of TM at any given input x q y, in this way.

The bodies of this ELOS are B_1,B_2,B_3, and the state space is the set of all (S_1,S_2,S_3,q) that occur. S_2,S_3 can only have 2 values each, and q can only have n values (the number of states in TM). However, S_1 generally has 4 values for cases 1-4, and 6 values for cases 5-8.

So the state space for cases 1-4 has at most 4(2)(2)(n) = 16n elements, and the state space for cases 5-8 has at most 6(2)(2)(n) = 24n elements.

Looking more closely, we see that in cases 5-8, S_2 is always 0 when $S_1 \in \{0,1,2,3,4\}$. Hence the state space for cases 5-8 also has at most 7(2)(n) = 14n ≤ 16n elements.

We can obviously reduce our ELOS to a LOS with reference point 0, by starting with the three bodies B_1,B_2,B_3, and then adding at least log(16n) = log(n)+4 additional bodies, each of which are constricted to 0 or 1. Any transition for the additional bodies can be given by a single law of motion. The reference point 0 can be removed at the cost of one additional body. The total number of bodies is log(n)+8.

We have proved the following.

THEOREM 4.1. Suppose there is a TM with 2 symbols and n states, for which the boundedness problem is algorithmically unsolvable. There is a $\lceil \log(n) \rceil$+8 body Linear Order System LOS whose boundedness problem is algorithmically unsolvable. If the former is complete r.e. then the latter is complete r.e.

We now make use of the following universal TM' from [Ne08], p. 102 (also see [NW07]). TM' has 15 states $q_1,...,q_{15}$, and 2 symbols 0,1, where 0 is the blank. It's quintuples are

q_1 0 0 R q_1	q_1 1 1 R q_2
q_2 0 0 R q_1	q_2 1 0 R q_3
q_3 0 1 L q_5	q_3 1 1 L q_7
q_4 0 0 L q_5	q_4 1 1 L q_6
q_5 0 0 L q_4	q_5 1 0 R q_1
q_6 0 0 L q_4	q_6 1 0 L q_4
q_7 0 0 L q_7	q_7 1 1 L q_8
q_8 0 0 L q_7	q_8 1 0 L q_9
q_9 0 0 L q_{10}	q_9 1 1 R q_1
q_{10} 0 0 HALT	q_{10} 1 0 L q_{11}
q_{11} 0 0 R q_{14}	q_{11} 1 1 R q_{12}
q_{12} 0 0 R q_{12}	q_{12} 1 1 R q_{13}
q_{13} 0 0 R q_{12}	q_{13} 1 1 L q_2
q_{14} 0 1 R q_{15}	q_{14} 1 1 L q_3
q_{15} 0 0 R q_{14}	q_{15} 1 1 R q_{14}

We have the following facts.

i. The set of all inputs α at which TM' halts is complete r.e. [Ne08].

ii. At all valid encodings α of a TM and its input, TM' either halts or is unbounded. Email from T. Neary to the author, 10/28/08.

The TM needed for our purposes must be halting free. So we take our TM* to modify TM' as follows. TM* has one more state, q_{16}, with the following instructions:

q_1 0 0 R q_1	q_1 1 1 R q_2
q_2 0 0 R q_1	q_2 1 0 R q_3
q_3 0 1 L q_5	q_3 1 1 L q_7
q_4 0 0 L q_5	q_4 1 1 L q_6
q_5 0 0 L q_4	q_5 1 0 R q_1
q_6 0 0 L q_4	q_6 1 0 L q_4
q_7 0 0 L q_7	q_7 1 1 L q_8
q_8 0 0 L q_7	q_8 1 0 L q_9
q_9 0 0 L q_{10}	q_9 1 1 R q_1
q_{10} 0 0 L q_{16}	q_{10} 1 0 L q_{11}
q_{11} 0 0 R q_{14}	q_{11} 1 1 R q_{12}
q_{12} 0 0 R q_{12}	q_{12} 1 1 R q_{13}
q_{13} 0 0 R q_{12}	q_{13} 1 1 L q_2
q_{14} 0 1 R q_{15}	q_{14} 1 1 L q_3
q_{15} 0 0 R q_{14}	q_{15} 1 1 R q_{14}
q_{16} 0 0 R q_{10}	q_{16} 1 1 R q_{10}

LEMMA 4.2. For valid encodings α of a TM and its input, TM* is bounded at α if and only if TM' halts at α. The set of all inputs at which TM* remains bounded is complete r.e.

Proof: Let α be as given. Suppose TM* is bounded at input α. Then TM' is also bounded at input α, since every non halting instruction of TM' is an instruction of TM*. By ii above, TM' halts at input α.

Suppose TM' halts at input α. Then TM' arrives in state q_{10} reading symbol 0. Hence TM* also arrives in state q_{10} reading symbol 0. Then TM* continues by moving left and going into q_{16}. Then TM* moves right and goes into q_{10}. This toggle must go on forever. Hence TM* is bounded at input α.

For the second claim, use i above. QED

THEOREM 4.3. There is a 12 body Linear Order System for which there is no algorithm for determining whether an arbitrary initial configuration has bounded evolution. In fact, the boundedness problem is complete r.e.

Proof: Use Theorem 4.1 with n = 16. Then 4+8 = 12 bodies suffice. QED

5. ZFC AND INCOMPLETENESS. Two kinds of undecidability.

Already by the early part of the 20th century, the standard axiomatic basis for mathematics had been carefully formulated and reasonably well accepted. This is through the so called ZFC axioms, or Zermelo Frankel set theory with the axiom of choice. See [En77], [Fr08].

The vast preponderance of mathematics can, without difficulty, be proved within ZFC. Thanks to computer technology, full blown documentation of this is being carried out by the actual construction of formal proofs via the Mizar project, and related enterprises. For Mizar, see http://www.mizar.org/.

For some surveys of various systems for formalizing mathematics, see [Sh02], [WW02], [Wi03], [Wi05], [Wi07], [Fr07]. There are some related "digitization of mathematical knowledge" projects:
The Logosphere project. http://www.logosphere.org/.
The Mathscheme project.
http://imps.mcmaster.ca/mathscheme/.
The Mathweb project. http://www.mathweb.org/.

Notable exceptions to the sufficiency of ZFC include mathematical statements with an uncharacteristically heavy set theoretic component, plus some examples of ours that are much more down to earth. These incompleteness issues for ZFC are discussed in [Fr09a] and [Fr09b].

It is crucially important to distinguish between two kinds of "undecidability". They are often conflated by nonexperts.

First, there is undecidability, in the sense of no algorithm. I.e., we have an infinite family of mathematical statements, often nicely indexed by a parameter. We seek an algorithm for determining whether or not any given instance is true. Undecidability simply means that there is no such algorithm. In this Chapter, we have so far discussed only results of this first kind.

There is a generally much more delicate, deeper, and more difficult second kind of "undecidability", that is often confused with algorithmic undecidability.

This second kind of undecidability concerns a **SINGLE STATEMENT**. Not an infinite family of statements.

By way of background, there is no clear way to get at the hardness of a SINGLE QUESTION using algorithms. Obviously, there is an algorithm for deciding whether a single given statement S is true or not.

case 1. S is true. Let ALG be the algorithm that always says true. Then ALG correctly decides the truth value of S.

case 2. S is false. Let ALG be the algorithm that always says false. Then ALG correctly decides the truth value of S.

But this is essentially a joke that does not join the issue.

So how do we talk about "undecidability" in the context of a single statement S?

One simple way is to assert that S is neither provable or refutable in ZFC.

From the very standard work in mathematical logic, we have the following.

THEOREM 5.1. Let S be a set of positive integers defined in ZFC. Assume that there is no algorithm for determining whether any given positive integer is an element of S. Then there exists a positive integer n such that the statement “$n \in S$” is neither provable nor refutable in ZFC. This result also applies, more generally, to finite strings in a finite alphabet.

With a little bit of fiddling, this tells us that there is a 12 body Order System with initial conditions, for which the statement “the system with these initial conditions is bounded” is neither provable nor refutable in ZFC.

But this misses an essential point. If we pass through this classic Theorem, the number of digits needed to present the initial condition is unacceptably large.

On the other hand, if we require that the bodies start out at the same place, then the number of bodies needed also becomes unacceptably large.

6. CONJECTURES

We now formulate two conjectures concerning undecidability of the second kind in the realm of simplified physical systems.

There is a 20 body Linear Order System such that ZFC neither proves nor refutes boundedness with the initial configuration having all bodies at the same point.

As remarked before, an arbitrary 20 body Linear Order System may have so little in the way of symmetry that it may be unfeasible to describe it. This point is addressed by the following much stronger conjecture.

There is a 20 body Linear Order System of presentation complexity at most 50 such that ZFC neither proves nor refutes boundedness with the initial configuration having all bodies at the same point.

Such results cannot be obtained by simply applying Theorem 5.1. The issues are much deeper. For such conjectures, we need to powerfully exploit the intricacies of ZFC.

We view the results obtained here as representing a very primitive beginning. We anticipate that there will be undecidability results of the first and second kind concerning simplified physical systems which are successively closer to real physical systems, incorporating more and more of their essential features.

REFERENCES

[Ad55] S.I. Adian, The algorithmic unsolvability of problems concerning certain properties of groups, Dokl. Akad. Nauk SSSR, 103, 533-535.

[Ba01] C. Baiocchi, Three Small Universal Turing Machines. In: Springer LNCS 2055, Machines, Computation, and Universality, 2001, pp. 1-10.

[BP00] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, 2000.

[BPR03] Saugata Basu, Richard Pollack, and Marie-Fran¸coise Roy. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, Berlin, 2003.

[Co75] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages (Second GI Conf., Kaiserslautern, 1975)*, pages 134–183. Springer-Verlag, Berlin, 1975. Reprinted in [CJ98].

[CJ98] B. F. Caviness and J. R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*.
Springer-Verlag, Vienna, 1998.

[CLR90] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, 1990.

[COWJ06] A. Church, A. Olszewski, J. Woleński, R. Janusz, Church's Thesis after 70 Years, Ontos, 2006.

[Da65] M. Davis, The Undecidable, New York: Raven Press (1965).

[Da82] M. Davis, Computability and Unsolvability, Dover, 1982.

[Da00] The Universal Computer: The Road from Leibniz to Turing, W. W. Norton and Company, 2000.

[DH76]] New Directions in Cryptography, W. Diffie and M.E. Hellman, IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644-654.

[En77] H.B. Enderton, Elements of Set Theory, Academic Press, 1977.

[Fr08] H. Friedman, LOGIC: Interdisciplinary Adventures in Mathematics, Philosophy, Computer Science, and Education, Lecture Notes, http://www.math.ohio-state.edu/%7Efriedman/manuscripts.html #40, 2008.

[Fr09a] H. Friedman, My Forty Years On His Shoulders, Horizons of Truth, 2009, to appear.

http://www.math.ohio-state.edu/%7Efriedman/manuscripts.html

[Fr09b] H. Friedman, Boolean Relation Theory and Incompleteness, Lecture Notes in Logic, ASL, 2009, to appear.
http://www.math.ohio-state.edu/%7Efriedman/manuscripts.html

[Go01,04] O. Goldreich, Foundations of Cryptography: Volume 1, Basic Tools, 2001. Volume 2, Basic Applications, 2004.

[GS81] F.J. Grunewald and D. Segal, How to solve a quadratic equation in integers, *Math. Proc. of the Cambridge Phil. Soc.*, 89 (1981), 1-5.

[Jo82] J.P. Jones, Universal Diophantine equation, Journal of Symbolic Logic, 47(3):549-571, 1982.

[Kn73] D. Knuth, Seminumerical Algorithms, volume 2 of The Art of Computer Programming, Addison-Wesley, 1969. Second edition. 1981.

[Ma93] Y. Matiyasevich, Hilbert's Tenth Problem, MIT Press, 1993.

[Mar58] A.A. Markov, Unsolvability of the problem of homeomorphy, in: Proceedings of the International Congress of Mathematicians (Cambridge University Press, London), pp. 300-306, 1958.

[Maz94] B. Mazur, Questions of decidability and undecidability in number theory. Journal of Symbolic Logic, 59(2):353-371, June 1994.

[Mi76] G.L. Miller, *Riemann's Hypothesis and Tests for Primality*, Journal of Computer and System Sciences 13 (1976), no. 3, pp. 300–317.

[MOV97] Handbook of Applied Cryptography, A. Menezes, P. Oorshot, S. Vanstone, CRC Press, 1997.

[MP96] J. McKee, R. Pinch, Old and New Deterministic Factoring Algorithms, Lecture Notes in Computer Science; Vol. 1122, 1996.

[NW07] T. Neary, D. Woods, Four Small Universal Turing Machines, Machines, Computations, and Universality, LNCS 4664, 242-254, 2007.

[Ne08] T. Neary, Small universal Turing machines, http://www.cs.nuim.ie/~tneary/tneary_Thesis.pdf 2008, 145 pages.

[No55] P.S. Novikov, On the algorithmic unsolvability of the word problem in group theory, Dokl. Akad. Nauk SSSR Mathematiceskii Institut Trudy, 44, Moscow, 1955.

[Po08] B. Poonen, Undecidability everywhere, March, 2008, http://www-math.mit.edu/~poonen/slides/cantrell3.pdf

[Ra58] M.O. Rabin, Recursive unsolvability of group theoretic problems, Ann. of Math., 67, 172-194.

[Ra80] M.O. Rabin, *Probabilistic algorithm for testing primality*, Journal of Number Theory 12 (1980), no. 1, pp. 128–138.

[Sh02] N. Shankar, Little Engines of Proof, Proceedings of FME'02, LNCS, 2002. ftp://ftp.csl.sri.com/pub/users/shankar/floc02.pdf

[Si97] W. Sieg, Step by Recursive Step: Church’s analysis of effective calculability; Bulletin of Symbolic Logic 3, 1997; 154-180.

[Ta51] A. Tarski. *A Decision Procedure for Elementary Algebra and Geometry*. Prepared for publication by J. C. C. McKinsey. University of California Press, second edition edition, 1951. Reprinted in [CJ98].

[Tu37] A. Turing, On computable numbers with an application to the Entscheidungsproblem, P. Lond. Math. Soc. (2) 42 (1936-7) 230-267; A correction, ibid. 43 (1937) 544-546; reprinted in [Da65].

[Wi03] F. Wiedijk, Comparing mathematical provers, In: Andrea Asperti, Bruno Buchberger & James Davenport (eds.), Mathematical Knowledge Management, Proceedings of MKM 2003, Springer LNCS 2594, 188-202, 2003 http://www.cs.ru.nl/~freek/comparison/diffs.pdf

[Wi05] F. Wiedijk, "Formalization of mathematics", TYPES Summer School 2005, Göteborg, 2005-08-23, 11:10 & seminar Algebraic and Numerical Algorithms and Computer-assisted

Proofs, Dagstuhl, 2005.
http://www.cs.ru.nl/%7Efreek/talks/gothenburg.pdf

[Wi07] F. Wiedijk, "Formalization of mathematics", Nederlands Mathematisch Congres, Leiden University, 2007.
http://www.cs.ru.nl/%7Efreek/talks/nmc.pdf

[WW02] M. Wenzel, F. Wiedijk, "A comparison of the mathematical proof languages Mizar and Isar", Journal of Automated Reasoning 29, 389-411, 2002
http://www.cs.ru.nl/~freek/pubs/romantic.pdf

*This research was partially supported by NSF DMS 0245349.