Finite-Dimensional Analog Computers: Flows, Maps, and Recurrent Neural Networks

Cristopher Moore

Santa Fe Institute
1399 Hyde Park Road, Santa Fe, New Mexico 87501 USA
moore@santafe.edu

Abstract. A number of authors have explored the computational power of dynamical systems with a finite number of continuous degrees of freedom. We review this work, from the point of view both of physics and of recurrent neural networks. We state several conjectures limiting the dimensionality, smoothness, and robustness to noise that these systems can have and still be computationally powerful. We also explore resource-bounded and real-time analog computers, review how techniques like VC dimension can be used to prove lower bounds, and suggest several directions for further research.

1 Computational Universality

1.1 Maps in $\mathbb{R}^2$ and Flows in $\mathbb{R}^3$

Recall that a *Turing machine* (TM) [21] consists of a bi-infinite tape, on which symbols in a finite alphabet (say binary) are written. The machine's head has a finite set S of internal states. At each time-step, it examines the symbol at its location on the tape and its own state. Based on these, it updates its own state, writes a new symbol at that place on the tape, and then moves one step left or right. The machine starts in a particular initial state, at the left end of a finite input word written on the tape, with a marker at the right end of the word. If it ever enters a particular state $q_{\text{halt}} \in S$, it halts and is said to *accept* the input.

The question of whether a Turing machine M will ever halt on a given input w is the Halting Problem, and Turing showed that it is unsolvable; unsolvable, that is, by any Turing machine given both w and a description of M as its input. This turns out to be closely related to Gödel's proof that there are unsolvable truths in any axiomatic system sufficient to express number theory.

In 1990 and in several years following, various authors independently showed that finite-dimensional piecewise-linear maps and flows can simulate Turing machines [22, 23, 28, 31, 14, 1]. The Halting Problem corresponds to questions such as whether an initial point will ever fall into a given open set A as the map is iterated. Thus even simple dynamical questions about compact, low-dimensional systems can be undecidable. Statements such as "x will fall into A an infinite number of times" can be true but unprovable!

Note that this is qualitatively different from the unpredictability that comes from "chaos" or sensitive dependence to initial conditions, in which errors in our knowledge of the initial state are amplified over time. Here, the system's behavior as $t \to \infty$ is unpredictable, even if we know the initial conditions exactly, e.g. if x's coordinates are rational.

The construction is so simple that it's surprising no one noticed it earlier. For each point (x, y) in the unit square, associate the binary digits of its x and y coordinates with the left and right halves of a Turing machine's tape: if the tape is $\cdots a_{-2}a_{-1}a_0a_1a_2\cdots$, let $x = 0.a_{-1}a_{-2}\ldots$ and $y = 0.a_0a_1a_2\ldots$. Then shifting the tape head right, say, is equivalent to halving x, doubling y, and shifting y's most significant digit to x. This is simply the Baker's Map on the unit square, well-known in ergodic theory [8] and shown in figure 1.

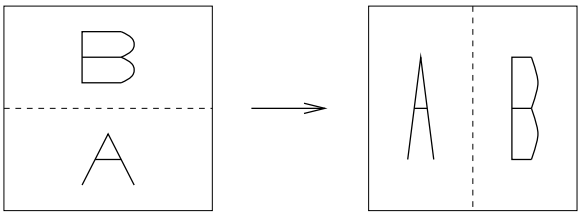

Fig. 1. The Baker's Map, equivalent to shifting the tape head to the right: $y = 0.a_0a_1a_2\ldots$ becomes $0.a_1a_2\ldots$, and $x = 0.a_{-1}a_{-2}\ldots$ becomes $0.a_0a_{-1}a_{-2}\ldots$.

We can write on the tape by adding constants to x and y, and represent the machine's internal state either with a finite number of separate squares in the plane, or by absorbing the machine's state into the tape and reading several sites on the tape at once. Either way, any Turing machine can be converted into a piecewise-linear map of the plane with a finite number of rectangular components, each one of which is an affine transformation.

Since halting corresponds to falling into a rectangle A representing q_{halt}, the question of whether a given initial point x will fall into a given open set A is undecidable. By Rice's Theorem [29], it is also undecidable whether the basin of attraction of A is dense, of measure 1, of measure greater than a given μ, or whether it has a non-zero intersection with another open set. In fact, the measure of A's basin of attraction will be related to Ω, Chaitin's uncomputable number [6]. It is also easy to replace halting with returning to the system's initial state, so it is undecidable whether x is a periodic point; or we can make halting correspond to entering a chaotic region of the plane, so that the Lyapunov exponent of the map is uncomputable, and it is even undecidable whether the map is chaotic or not [23].

We can obtain a smoother map. By using base 3, we can place our points on a Cantor set, like the invariant set of Smale's Horseshoe Map [32]. We can then smooth the map in the gaps between components, making it infinitely differentiable.

Furthermore, using Bennett's proof [2] that any Turing machine can be simulated by a reversible one, we can ensure that this map is one-to-one. We can also make it isotopic to the identity [23]. Then we can embed it in a flow in $\mathbb{R}^3$, as a billiard or optical ray-tracing system [28], or as a point particle moving in an infinitely differentiable energy potential [22]. A similar map can be embedded in the dynamics of a recurrent neural network with piecewise-linear activations [31], or a hybrid system with piecewise-constant derivatives [1].

To what extent can this construction be improved, by reducing the dimensionality or increasing smoothness from infinite differentiability to analyticity? Three dimensions is minimal for a flow, since flows in a compact subset of $\mathbb{R}^2$ must converge either to a fixed point or a limit cycle. However, it turns out that even in one dimension, iterated maps can be computationally universal.

1.2 Maps in $\mathbb{R}$

The simplest way to do this is to simply take a Turing machine's dynamics as a function on the set of bi-infinite sequences. If we conjugate it with a map that folds two-sided sequences into one-sided ones, $\cdots a_{-2}a_{-1}a_0a_1a_2\cdots \mapsto a_0a_1a_{-1}a_2a_{-2}\cdots$, we get a function which is continuous in the standard metric on sequences, $d(x,y) = \sum_{i=0}^{\infty} 2^{-i}|x_i - y_i|$.

Interpreting these sequences as digits in the middle-thirds Cantor set in the unit interval, and smoothing in the gaps, gives us a function which is continuous and *Lipshitz*, i.e. there exists a constant c such that $d(f(x), f(y)) < cd(x,y)$. (It was erroneously stated in [24] that the resulting function is once-differentiable.) Then this function simulates a Turing machine in real time — that is, one time-step per time-step.

Another embedding of universal computation in one dimension, which is analytic but exponentially slow, is given in [15]. Recall the classic "$3x+1$ problem" [18]. If f is the function on the integers

$$f(x) = \begin{cases} x/2 & (x \text{ even}) \\ 3x+1 & (x \text{ odd}) \end{cases}$$

then, for all x, does there exist a t such that $f^t(x) = 1$? In dynamical terms, is all of $\mathbb{N}$ in the basin of attraction of the periodic orbit $\{1, 4, 2\}$? This remains a tantalizing unsolved problem.

More generally, let

$$f(x) = a_i x + b_i \text{ where } x \equiv i \bmod p \tag{1}$$

for some base p and constants a_i, b_i for $0 \leq i < p$. We will call any such f a *Collatz function*. J.H. Conway [7] showed that it is undecidable in general whether $f^t(x) = 1$ for some t. In fact, the record-holding small Turing machines in the Busy Beaver competition calculate Collatz functions [20], another testament to their complexity.

Conway does this by simulating Minsky machines, which are finite-state automata (FSAs) that can increment, decrement, or branch on zero on a finite

number of counters. Minsky showed [21] that two counters suffice to simulate an arbitrary Turing machine; however, this creates a doubly-exponential slowdown, so we use three counters instead. Let L and R be integers representing the left and right halves of the tape and add a work counter W. For a binary alphabet $\{1,2\}$ plus a blank represented by 0, we have

$$L = \sum_{i=1} 3^{i-1} a_{-i} \text{ and } R = \sum_{i=0} 3^i a_i$$

These are finite if the tape is blank beyond a finite number of sites.

To read the symbol a_0 at the head's current location, give the FSA a loop of 3 states that decrement R, the last one of which also increments W. We will get to $R = 0$ in one of these three states. Which one tells us the tape symbol $a_0 = R \bmod 3$, and leaves us with the remainder of the right half of the tape, $\lfloor R/3 \rfloor$, in W.

To shift the head right, we read W back into R by decrementing W and incrementing R until $W = 0$, multiply L by 3 by incrementing W three times each time we decrement L until $L = 0$, and then read the result from W back into L. Finally, we write the new symbol in the least significant digit of L by incrementing L once or twice. Shifting the head left is similar. The FSA's states keep track of the head's internal state as well as where we are in this set of loops; details are given in [15].

Since this counter machine takes $\mathcal{O}(\max(L,R))$ steps to simulate a single step of the Turing machine, and since L and R are $\mathcal{O}(3^l)$ where l is the length of the tape used by the TM, and since a Turing machine uses at most t tape sites in t computation steps, we find that t time-steps of a Turing machine can be simulated by a three-counter in time $\mathcal{O}(t\,3^t)$.

We now show how to simulate a three-counter Minsky machine with a Collatz function. If the FSA has k states and is currently in state s where $0 \le s < k$, define

$$x = 2^L\, 3^R\, 5^W k + s$$

Clearly all of our operations can be carried out on x by adding, multiplying, or dividing by a constant. For instance, to decrement W and update the state from s to s', we write

$$f(x) = (x-s)/5 + s' = x/5 + (s' - s/5)$$

so $a = 1/5$ and $b = s' - s/5$ in equation (1). Branching on zero for each register can be done by checking $x \bmod p$ for various p. For instance, $R = 0$ if and only if $x \bmod 3k \neq s$. Finally, if $s = 1$ is the halt state, we define $a = 0$ and $b = 1$ whenever $x \bmod 30k = 1$, so a halting computation ends with the fixed point $x = 1$.

Now we note that Collatz functions can be embedded in analytic functions. Let

$$h(x) = \left(\frac{\sin \pi x}{p \sin \frac{\pi x}{p}} \right)^2 = \begin{cases} 1 \text{ if } x \bmod p = 0 \\ 0 \text{ if } x \bmod p \neq 0 \end{cases} \text{ for integer } x$$

This is everywhere analytic, and the fraction can be resolved using multiple-angle formulas. Then

$$f(x) = \sum_{i=0}^{p-1} h(x-i)\,(a_i x + b_i)$$

matches equation (1) on the integers.

Thus iterated maps on $\mathbb{R}$, consisting of a sum of a finite number of trigonometric terms, can simulate Turing machines with an exponential slowdown, making all dynamical questions stated above undecidable in one dimension. (Using Rogozhin's 6-state, 4-symbol Turing machine [30] gives a map with at most 2470 trigonometric terms.)

Can this exponential slowdown be removed? This is easily done with an analytic function in $\mathbb{R}^2$, with one integer for each half of the stack [15], but can we do a faster simulation in one dimension?

Clearly analytic functions on $\mathbb{R}$ can simulate Turing machines at any rate we like, simply because any function on the integers that grows more slowly than some analytic function can be interpolated with an analytic function. However, although such a function could be expressed as an infinite sum, it would not have any convenient finite description. If we limit ourselves to *elementary* functions that can be defined in terms of $\sin x$, e^x, addition, multiplication and composition — or even the *differentially algebraic* functions, which are the solutions to polynomial differential equations — then we conjecture that exponential slowdown is the best we can do:

Conjecture 1. *No elementary or differentially algebraic map on $\mathbb{R}$ can simulate a Turing machine with less than an exponential slowdown.*

Any counterexample to this conjecture would have to rely on a very different construction than the one given here.

1.3 Compactness and Analyticity

We have seen two ways to embed universal computation in low-dimensional spaces, one which is compact and infinitely differentiable, but not analytic, and one which is analytic and not compact. Can Turing machines be simulated by an analytic function on a compact space?

I don't believe so. However, to make such a claim well-defined, we have to define how a discrete input is to be encoded into a point in $\mathbb{R}^n$, and how the result of the dynamics is mapped back into "yes" or "no." After all, even the identity function is capable of computation if we do all the work in the encoding: just send all the "yes" inputs to 1, the "no" inputs to 0, iterate the identity map a few times,[1] and ask whether $x > 1/2$ or not! Clearly such an encoding is unreasonable; the problem is defining "reasonable" in a sufficiently general way, which includes the digits of real numbers or integers as we do above, but excludes absurdities like this one. We return to this question in Section 2.

In any case, we state the following:

[1] Three or four should be sufficient.

Conjecture 2. *No analytic function on a compact, finite-dimensional space can simulate a Turing machine through a reasonable input and output encoding.*

Note that we are not saying that all questions about the long-term behavior of compact analytic dynamical systems are decidable! That would be a much stronger claim.

How might this conjecture be proved? Turing machines generally have a countably infinite number of fixed points and periodic points of each period, while compact analytic maps can have only a finite or uncountable number. In one dimension this creates a contradiction [14], since if f has an infinite number of periodic points of period t, then f^t must be the identity. However, no such contradiction exists in more than one dimension, in which we can easily have a continuum of fixed points without fixing the rest of the space.

It seems difficult to answer this question without a better understanding of the *dynamics* of Turing machines: not just the relationship between their input and output, but what happens in between; and not just what happens when we start in a properly initialized state, but their global dynamics on arbitrary infinite sequences. For instance, Petr Kůrka [17] has conjectured the following:

Conjecture 3. *Every Turing machine, as a dynamical system on the space of bi-infinite sequences, has at least one periodic orbit.*

For instance, if we place a Turing machine designed to increment a binary number at the right end of an infinite string of 1's, it will endlessly convert them to 0's, carrying a 1 forever to the left and making $\cdots 111.1000 \cdots$ a fixed point.

Even this minimal conjecture seems very hard to prove. Perhaps the embedding of generalized shifts as maps in the plane can help by re-phrasing it as a purely geometrical question.

1.4 Genericity and structural stability

These embeddings of universal computation in low-dimensional maps and flows are certainly interesting. But a physicist might well dismiss them unless they are likely to actually occur in nature, and an engineer will do the same if they are so sensitive to noise that they are impossible to build. So we should ask if any of these systems are *generic*, i.e. if they have non-zero measure in some space of maps or flows.

The strongest definition of genericity is *structural stability*. A map or flow f is structurally stable if there exists an ϵ such that, for all perturbations g of norm less than ϵ, the system $f + g$ is topologically equivalent to f; that is, there is a homeomorphism ϕ such that $f + g = \phi^{-1} \circ f \circ \phi$. Then there is a one-to-one correspondence between the orbits and periodic points of the original system and the perturbed system, so the system is surrounded by an open set in the space of maps which are all equivalent to it.

Among other things, structural stability requires that the system's Lyapunov exponents be non-zero, so that nearby initial conditions diverge exponentially from one another, with distance $d_t = e^{\lambda t}$. In the Baker's Map, this takes place

by continually stretching in the vertical direction. But this corresponds to a linear bound on the position of the Turing machine's head, moving it leftward (or rightward) at a steady rate.

Turing machines that do a non-trivial computation spend a lot of time revisiting the same place on the tape, so that they move away from their initial positions less than linearly, corresponding to a less-than-exponential divergence of initial conditions. In [23], examples of generalized shifts are given where d_t is proportional to $2^{\sqrt{t}}$ and t^α, both less than exponential. So similar to conjecture 3, we state the following:

Conjecture 4. *Every Turing machine that accepts a non-regular language has trajectories such that the position x_t of the Turing machine moves less than linearly,* $\lim_{t\to\infty} x_t/t = 0$.

This would imply that computationally powerful systems cannot be structurally stable, at least if the encoding between Turing machines and points in $\mathbb{R}^n$ is anything like the Cantor set used above.

Guckenheimer and Holmes [12] point out that structural stability may be too harsh a requirement for a "natural" system. For instance, it would disqualify the Lorenz attractor. I still feel that these systems are genuinely fragile, and must be perfectly tuned, and so are isolated in the space of maps:

Conjecture 5. *No finite-dimensional system capable of universal computation is stable with respect to perturbations, or generic according to any reasonable definition.*

However, even their existence is an important negative result, in that it precludes any possibility of a complete classification or solution of any class of low-dimensional maps or flows that includes them.

2 Real-time and resource-bounded computation

2.1 Dynamical recognizers

Undecidability and uncomputability is still an exciting idea to many physicists and dynamicists. But most computer scientists long ago shifted their attention from universal computation to *feasible* computation, where resources such as time or memory are bounded to some function of the size of the input. Just because a function is computable doesn't mean we can compute it during the lifetime of the universe.

As Papadimitriou points out [26], we have actually become more conservative in our definition of feasibility as computing technology has improved — from computability in the 30's and 40's, to the Grzegorczyk hierarchy of exponentials in the 50's, to polynomial-time computation and NP-completeness in the 60's and 70's, to logarithmic parallel time in the 80's and 90's. Since Turing's work of 1936 was interesting to physicists in 1990, it makes sense to try to import more recent notions as well.

One of the strictest kinds of resource bound is *real time*. Here a computing machine is fed an input from left to right, makes exactly one computation step per symbol, and then is forced to give an answer immediately, without any further computation.

This suggests a simple kind of analog computer, called a *dynamical recognizer* [27]. For each symbol a in the input alphabet, let f_a be a map from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then starting with an initial point $x_0 \in \mathbb{R}^n$, apply the maps $f_{w_1}, f_{w_2}, \ldots$ for each symbol of the input word w in turn. Using the shorthand $f_w = f_{w_{|w|}} \cdots f_{w_2} f_{w_1}$, we end up at a point $x_w = f_w(x_0)$, which we think of as the encoding of w in $\mathbb{R}^n$. Then we accept the word if x_w is in a particular subset $H_{\text{yes}} \subset \mathbb{R}^n$.

We can define various classes of dynamical recognizers by restricting the maps f_a to a given class of functions, and requiring that H_{yes} be defined with inequalities in the same class; for instance, **Lin**, **PieceLin**, **Poly** and **Elem** for linear, piecewise-linear, polynomial, and elementary (exponential and trigonometric) functions. We can also define non-deterministic classes **NLin**, **NPieceLin**, **NPoly** and **NElem** where there are several choices of function f_a for each symbol, and we accept a word if a set of choices exists such that x_w lands in H_{yes}. We explore these classes in [25].

Dynamical recognizers are capable of many simple kinds of encoding. For instance, if $x_0 = 1/2$, $f_0(x) = x/2$ and $f_1(x) = x/2 + 1/2$, then x_w is the point in the center of the gap of the middle-thirds Cantor set corresponding to the binary word w. Alternately, if $x_0 = 0$, $f_1(x) = 3x + 1$ and $f_2(x) = 3x + 2$, then x_w is w written as an integer in base 3, which we write $\overline{w}$. Thus both kinds of encoding used in the computationally universal systems above can be expressed this way. If we take this as our definition of "reasonable" encoding, then different classes of dynamical recognizers represent encodings of varying power.

A dynamical recognizer in $\mathbb{R}^n$ could also represent a synchronous, recurrent neural network with n neurons. Such neural networks are being studied as models of language recognition for regular [10], context-free [34], and context-sensitive [33] languages, as well as fragments of natural language [9]; in [25], we show that piecewise-linear and polynomial dynamical recognizers can have both stack-like (last-in, first-out) and queue-like (first-in, first-out) memories. This suggests that language might actually be understood in a dynamical, rather than grammatical way. In this context, different classes of dynamical recognizers correspond to different kinds of activation functions (piecewise-linear, quadratic, sigmoidal, etc.) in the recurrent neural network.

3 Using VC dimension to prove lower bounds

Upper bounds on a problem's complexity can be gotten simply by showing that it can be solved with an algorithm in a given class. Lower bounds, on the other hand, are few and far between in computer science; we have to somehow prove that no algorithm in a given class will succeed in solving the problem. In this section, we show one technique for proving such lower bounds.

Consider the language

$$L = \{\, w_1 \not\in w_2 \not\in \cdots \not\in w_m \$ v \mid v = w_i \text{ for some } i \}$$

where v and the w_i are words over a binary alphabet $\{0, 1\}$. This represents a simple memory task: given a list of words w_i and a word v, determine if v was in the list. We will show that L is not in **Poly** or **PieceLin**, nor can it be recognized in real time by a recurrent neural network with sigmoid or threshold activations.

Note that L can be "programmed" to recognize any finite language. If $u = w_1 \not\in w_2 \not\in \cdots \not\in w_m \$$ where $w_1, \ldots, w_m$ are all the words in a finite language L_u, then $uv \in L$ if and only if $v \in L_u$. Therefore, any recognizer for L contains recognizers for all possible finite languages in its state space, in that it recognizes L_u if we use x_u instead of x_0 as our initial point.

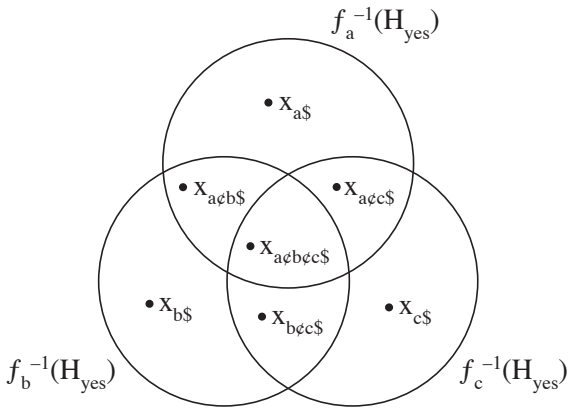

Fig. 2. The family of sets $f_v^{-1}(H_{\text{yes}})$ over all finite words v is independent.

A family of sets $\mathcal{S} = \{S_1, \ldots, S_n\}$ is *independent* if all 2^n possible intersections of the S_i and their complements are non-empty; in other words, if the S_i overlap in a Venn diagram. But since $f_v(x_u) \in H_{\text{yes}}$ if and only if $v \in L_u$, x_u is in the following intersection of sets:

$$x_u \in \left(\bigcap_{v \in L_u} f_v^{-1}(H_{\text{yes}}) \right) \cap \left(\bigcap_{v \notin L_u} \overline{f_v^{-1}(H_{\text{yes}})} \right)$$

For instance, if a, b and c are binary words, then $x_{a \not\in b\$}$ is in $f_a^{-1}(H_{\text{yes}})$ and $f_b^{-1}(H_{\text{yes}})$, but not in $f_c^{-1}(H_{\text{yes}})$ as shown in figure 2. Since any such intersection is therefore non-empty, the family of sets $f_v^{-1}(H_{\text{yes}})$ is independent, where v ranges over any finite set of words.

The *Vapnik-Chervonenkis (VC) dimension* of a family $\mathcal{S}$ of sets is the size of its largest independent sub-family. Alternately, it is the size of the largest set X

of points which are *shattered* by $\mathcal{S}$, meaning that for every subset $Y \subset X$, there is some $S \in \mathcal{S}$ such that $X \cap S = Y$. Thus a dynamical recognizer for L has to achieve a VC dimension of 2^n in n time-steps, since the sets $f_v^{-1}(H_{\text{yes}})$ for the 2^n different v of length n are all independent.

Goldberg [11] has shown that the VC dimension of a dynamical recognizer whose maps are polynomials of any degree can only grow linearly with time, and a similar argument works for piecewise-linear maps. So L is not in **PieceLin** or **Poly**. Koiran and Sontag [16] have shown linear and quadratic upper bounds on the VC dimension of recurrent neural networks with threshold and sigmoidal activations respectively, so such networks also cannot recognize L in real time; in fact, all of these would need exponential time to do so.

On the other hand, L is easily seen to be in **NLin**. As each w_i comes in, we choose whether to encode it into an integer $\overline{w}_i$ as above, or ignore it; we then encode v into an integer $\overline{v}$ and check that $\overline{v} = \overline{w}_i$. As a corollary, **Lin**, **PieceLin** and **Poly** are all properly contained in their non-deterministic counterparts [25].

L is also in the class **Elem**. By encoding words into integers $\overline{w}_i$ and letting $f_\not{c}(x) = x + 2^{\overline{w}}$, we can encode the entire list into one huge integer $x = \sum_i 2^{\overline{w}_i}$ by the time we reach the \$. Then the $2^{\overline{v}}$ digit of x is 1 if $v = w_i$ and 0 otherwise, so we let H_{yes} require that $\sin \pi(x/2^{\overline{v}}) < 0$ or $\cos \pi(x/2^{\overline{v}}) = -1$, i.e. $x/2^{\overline{v}} \in [2k+1, 2k+2)$ for some integer k. Here we're using the fact that all the sets $S_j = \{x \mid \sin 2^j x < 0\}$ for $j = 0, 1, 2, \ldots$ are independent, so the family $\{S_j\}$ has infinite VC dimension.

In [25] we also show that a unary version of L

$$L_{\text{unary}} = \{a^{p_1} \not{c} a^{p_2} \not{c} \cdots \not{c} a^{p_m} \$ a^q \mid q = p_i \text{ for some } i\}$$

is in **PieceLin** and **Poly** but not **Lin**, since its VC dimension grows linearly.

Unfortunately, VC dimension arguments seem rather limited in their scope. They cannot distinguish between polynomial maps of different degree, even though there is strong evidence [25] that the classes $\mathbf{Poly}_k$ of dynamical recognizers of degree k form a distinct hierarchy. Neither do they help us show that a language is not in **NLin** or any other non-deterministic class.

3.1 Robustness to noise

If we push n binary symbols onto a stack represented by the digits of a real number in the unit interval, then any perturbation greater than 2^{-n} in size will destroy them. Thus although dynamical recognizers can in principle store arbitrary stacks or queues of information, these are very fragile. In fact, using different models of noisy recognition, Maass and Orponen [19] and Casey [5] have shown that no compact dynamical recognizer can recognize arbitrarily long strings of a non-regular language in the presence of non-zero noise.

However, we can ask how quickly the computation goes wrong; that is, to what length we can correctly recognize a non-regular language in the presence of a given amount of noise. Let **DIGITS** be the maximum number of digits in a recognizer's variables as a function of the input length n. If our variables are

rational and confined to the unit cube, and the system is exposed to noise of size ϵ, then words in a language in **DIGITS**$(f(n))$ will be correctly recognized up to length $n \sim f^{-1}(\log \epsilon^{-1})$.

The class **LOGDIGITS** = **DIGITS**$(\log n)$ is a good definition of robust computation, analogous to the class **LOGSPACE** of languages recognizable by a Turing machine with $\mathcal{O}(\log n)$ memory. Its relationship between noise and length is a power law, so we can double the length by dividing the noise by a constant. It's easy to show that $n \sim \epsilon^{-d}$ is the best that a d-dimensional recognizer can do, so **DIGITS**$(f(n))$ contains only the regular languages if $\lim_{n\to\infty} f(n)/\log n = 0$. Thus **LOGDIGITS** is the most robust class with interesting behavior.

In analogy to the space hierarchy theorem for Turing machines [13], we suggest the following:

Conjecture 6. **DIGITS**$(f(n))$ *is properly contained in* **DIGITS**$(g(n))$ *if* $\lim_{n\to\infty} f(n)/g(n) = 0$ *and if* $f(n) > c \log n$ *for some* c.

It would also be nice (as always) to be able to prove lower bounds on **DIGITS** for a given language.

3.2 Bounding the number of variables

The main difference between dynamical recognizers and the Blum-Shub-Smale model of analog computation [3] is that their model can access an arbitrary number of real variables, growing linearly with time, while dynamical recognizers and recurrent neural networks have a fixed dimensionality.

This suggests using the number of variables as a resource bound. **LOGVARS**, for example, would be the class of languages recognizable by an analog computer whose dimensionality grows logarithmically with the input length. We could combine this with time bounds as well, or restrict ourselves to the real-time case. I think this is likely to be an interesting class.

4 Acknowledgements

I thank Cristian Calude and John Casti for inviting me to UMC '98, Michael Dinneen for his patience, and Spootie the Cat for her companionship, even though her incessant washing made it difficult to catch much-needed sleep during the writing of this paper. I also thank J. Felix Costa for noticing some mistakes. This work was supported in part by NSF grant ASC-9503162.

References

1. E. Asarin, O. Maler and A. Pnueli, "Reachability analysis of dynamical systems with piecewise-constant derivatives." *Theoretical Computer Science* **138** (1995) 35-66.
2. C. H. Bennett, "Logical Reversibility of Computation." *IBM J. Res. Develop.* **17** (1973).

3. L. Blum, M. Shub, and S. Smale, "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines." *Bull. Amer. Math. Soc.* **21** (1989) 1-46.
4. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth, "Learning and the Vapnik-Chervonenkis dimension." *Journal of the ACM* **36(4)** (1989) 929-965.
5. M. Casey, "The dynamics of discrete-time computation, with application to recurrent neural networks and finite-state machine extraction." *Neural Computation* **8:6** (1996)
6. G. Chaitin, "A theory of program size formally identical to information theory." *J. Assoc. Comput. Mach.* **22** (1975) 329-340; and "Algorithmic information theory." *IBM J. Res. Develop.* **21** (1977) 350-359.
7. J.H. Conway, "Unpredictable iterations." In *Proc. 1972 Number Theory* University of Colorado (1972) 49-52.
8. I.P. Cornfeld, S.V. Fomin, and Ya.G. Sinai. *Ergodic Theory.* Springer-Verlag, 1982.
9. J. Elman, "Language as a dynamical system." In R.F. Port and T. van Gelder (Eds.), *Mind as Motion: Explorations in the Dynamics of Cognition.* MIT Press, 1995.
10. C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee, "Learning and extracting finite-state automata with second-order recurrent networks." *Neural Computation* **2** (1992) 331-349
11. P.W. Goldberg and M.R. Jerrum, "Bounding the Vapnik-Chevonenkis dimension of concept classes parametrized by real numbers." *Machine Learning* **18** (1995) 131-148.
12. John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields.* Springer-Verlag, 1983.
13. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
14. P. Koiran, M. Cosnard, and M. Garzon, "Computability with low-dimensional dynamical systems." *Theoretical Computer Science* **132** (1994) 113-128.
15. P. Koiran and C. Moore, "Closed-form analytic maps in one and two dimensions can simulate universal Turing machines." *Theoretical Computer Science* **210** (1999) 217-223.
16. P. Koiran and E.D. Sontag, "Vapnik-Chervonenkis dimension of recurrent neural networks." To appear in *Discrete Applied Math.*
17. P. Kůrka, "On topological dynamics of Turing machines." *Theoretical Computer Science* **174** (1997) 203-216.
18. J.C. Lagarias, "The $3x+1$ problem and its generalizations." *Amer. Math. Monthly* **92** (1985) 3-23.
19. W. Maass and P. Orponen, "On the effect of analog noise in discrete-time analog computations." To appear in *Proc. Neural Information Processing Systems* (1996).
20. P. Michel, "Busy beaver competition and Collatz-like problems." *Arch. Math. Logic* **32** (1993) 351-367.
21. Marvin Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.
22. C. Moore, "Unpredictability and undecidability in dynamical systems." *Physical Review Letters* **64** (1990) 2354-2357
23. C. Moore, "Generalized shifts: unpredictability and undecidability in dynamical systems." *Nonlinearity* **4** (1991) 199-230.
24. C. Moore, "Smooth one-dimensional maps of the interval and the real line capable of universal computation." Santa Fe Institute Working Paper 93-01-001 (1993).
25. C. Moore, "Dynamical Recognizers: Real-time Language Recognition by Analog Computers." *Theoretical Computer Science* **201** (1998) 99-136.

26. C.H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.
27. J. Pollack, "The Induction of Dynamical Recognizers." *Machine Learning* **7** (1991) 227-252.
28. J.H. Reif, J.D. Tygar, and A. Yoshida, "The computability and complexity of optical beam tracing." In *Proc. 31st Annual Symposium on Foundations of Computer Science* (1990) 106-114.
29. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.
30. Yu.V. Rogozhin, "Seven universal Turing machines" (in Russian), *Systems and Theoretical Programming, Mat. Issled* **69** (1982) 76-90, and "Small universal Turing machines" (in English), *Theoretical Computer Science* **168** (1996) 215-240.
31. H. Siegelmann and E.D. Sontag, "On the Computational Power of Neural Nets." *Journal of Computer and Systems Sciences* **50** (1995) 132-150.
32. S. Smale, "Diffeomorphisms with many periodic points", in *Differential and Combinatorial Topology*, S. S. Cairns, Ed. Princeton University Press, 1963, pp. 63-80.
33. M. Steijvers and P.D.G. Grünwald, "A recurrent network that performs a context-sensitive prediction task." NeuroCOLT Technical Report NC-TR-96-035 (1996), and in *Proceedings of the 18th Annual Conference of the Cognitive Science Society.*
34. J. Wiles and J. Elman, "Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks." *Proceedings of the 17th Annual Conference of the Cognitive Science Society.* MIT Press, Cambridge, Massachusetts, 1995.