Closed-form Analytic Maps in One and Two Dimensions Can Simulate Universal Turing Machines

Pascal Koiran*, Ecole Normale Supérieure de Lyon koiran@lip.ens-lyon.fr
Cristopher Moore†, Santa Fe Institute moore@santafe.edu

April 6, 1999

Abstract

We show closed-form analytic functions consisting of a finite number of trigonometric terms can simulate Turing machines, with exponential slowdown in one dimension or in real time in two or more.

1 Introduction

Various authors have independently shown [9, 12, 4, 14, 1] that finite-dimensional piecewise-linear maps and flows can simulate Turing machines. The construction is simple: associate the digits of the x and y coordinates of a point with the left and right halves of a Turing machine's tape. Then we can shift the tape head by halving or doubling x and y, and write on the tape by adding constants to them. Thus two dimensions suffice for a map, or three for a continuous-time flow. These systems can be thought of as billiards or optical ray tracing in three dimensions, recurrent neural networks, or hybrid systems.

However, piecewise-linear functions are not very realistic from a physical point of view, and although these maps can be smoothed into infinitely differentiable ones [9], most physical dynamical systems are analytic, at least in a perfectly classical world. Analytic functions in general are trivially computationally universal, since we can embed any function from the integers to $\{0, 1\}$ (in fact, any function on the integers that grows more slowly than some primitive recursive function) in an analytic function on the reals. However, it would be more satisfactory to find computationally universal analytic functions with elementary closed forms. Specifically,

*A part of this author's work was done when he was visiting DIMACS at Rutgers University.
†Corresponding author.

Definition. *Let U_n be the smallest class of functions $f : \mathbb{R}^n \to \mathbb{R}$ containing rational constants, π, the n projections $x \mapsto x_i$, and satisfying the following closure properties:*

- *if $f, g \in U_n$ then $f \perp g \in U_n$, where $\perp \in \{+, -, \times\}$.*
- *if $f \in U_n$ then $\sin f \in U_n$.*

We will say that $f : \mathbb{R}^n \to \mathbb{R}^n$ is elementary *if its n components are in U_n.*

In this paper, we construct two elementary functions: one in one dimension based on counter machines that simulates Turing machines with an exponential slowdown, and another in two dimensions that simulates TMs in real time.

Preliminary versions of these results appeared in [5] and [10].

2 One dimension: Minsky machines and Collatz functions

Recall [6] the classic "$3x + 1$ problem." 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 systems terms, is all of $\mathbb{N}$ in the basin of attraction of the periodic orbit $\{1, 4, 2\}$?

We can generalize this as follows. 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 $0 \le i < p$. We will call any such f a *Collatz function*. J.H. Conway [3] showed that it is undecidable in general whether $f^t(x) = 1$ for some t. Based on this fact, Burckel [2] has shown that it is undecidable whether certain functional equations have non-trivial solutions; in addition, the record-holding small Turing machines in the Busy Beaver competition calculate Collatz functions [8], 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 [11] 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 represent the left and right halves of the tape (with R including the tape symbol a_0 at the head's current location) and add a work counter W. Specifically, if our Turing machine has n states and m tape symbols,

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

For each state of the TM, our FSA has a loop of m states that decrement R, the last one of which also increments W. Where in this loop we get $R = 0$ tells us the tape symbol $a_0 = R \bmod m$, and we are left with $W = \lfloor R/m \rfloor$.

If the TM's instructions are to shift the head to the right at this point, we read W back into R by repeating $(\mathrm{dec}_W, \mathrm{inc}_R)$ until $W = 0$, multiply L by m by looping $(\mathrm{dec}_L, m\,\mathrm{inc}_W)$ until $L = 0$ and then $(\mathrm{dec}_W, \mathrm{inc}_L)$ until $W = 0$ (where "$m\,\mathrm{inc}_W$" means "increment W m times"), whereupon we add the new symbol a' to L with $a'\,\mathrm{inc}_L$.

If the TM is to shift left, we first set $R = m^2 W$ with the loop $(\mathrm{dec}_W, m^2\,\mathrm{inc}_R)$ until $W = 0$, and then enter a loop of m states that decrement L. We leave this loop with $W = \lfloor L/m \rfloor$ and knowing $a_{-1} = L \bmod m$, so we add $a_{-1} + ma'$ to R. Finally we loop $(\mathrm{dec}_W, \mathrm{inc}_L)$ until $W = 0$ to read W back into L.

In either case, we then go to the section of the FSA corresponding to the new state s'.

As shown in figure 1, this FSA needs 4 states for each state/symbol pair that shifts right, $m + 3$ for each pair that shifts left, and two for a pair that halts. Using Rogozhin's 4-state, 6-symbol universal Turing machine [13], which has 8 pairs that shift left and 14 that shift right, we get a total of 130 states. We have no idea if this is minimal; it could be reduced either by finding a more compact universal TM (Minsky's 7-state, 4-symbol machine yields 149 states) or by exploiting regularities in the rule table to merge multiple states in the FSA. We leave this to the reader.

Since each Turing machine step takes $\mathcal{O}(\max(L, R))$ steps of the FSA to simulate, and since L and R are $\mathcal{O}(m^l)$ where l is the length of the tape used by the TM, a Turing machine computation that takes time t can be performed by a three-counter Minsky machine in time $\mathcal{O}(tm^t)$.

We now show how to simulate a three-counter Minsky machine with a Collatz function, using a slightly different construction from Conway's. 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. For instance, to decrement L, increment W m times, and update the state from s to s', we write

$$(\mathrm{dec}_L, m\,\mathrm{inc}_W):\;\; f(x) = (5^m/2)(x - s) + s' = (5^m/2)x + (s' - (5^m/2)s)$$

so $a = 5^m/2$ and $b = s' - (5^m/2)s$. Moreover, we can distinguish s and test for zero on all our registers in terms of $x \bmod 30k$:

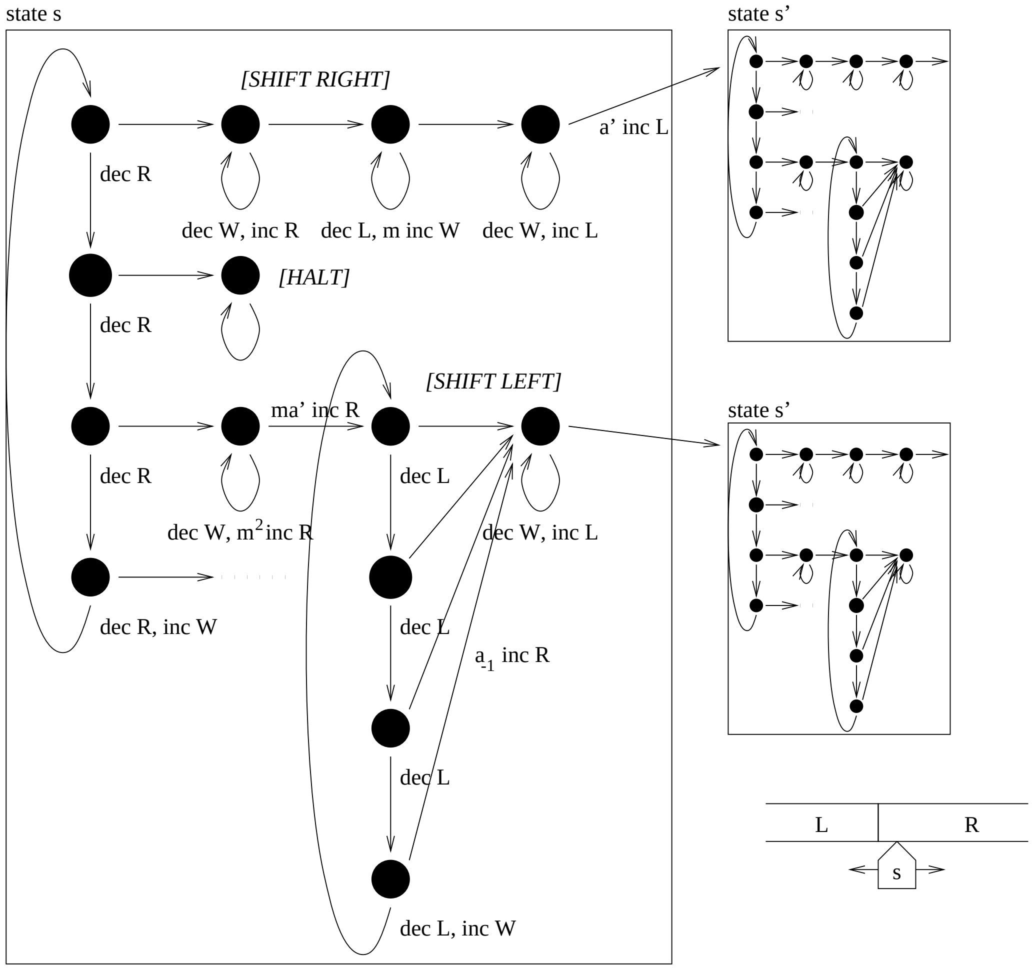

Figure 1: How to simulate a Turing machine with a finite-state machine that can increment and decrement three registers, L, R and W. As an example, we show a state s which takes three possible actions — shifting left, shifting right, and halting — depending on the tape symbol a.

$$x \bmod 30k = \begin{cases} s & L>0, R>0, W>0 \\ s+15k & L=0, R>0, W>0 \\ \left.\begin{array}{l} s+10k \\ s+20k \end{array}\right\} & L>0, R=0, W>0 \\ \left.\begin{array}{l} s+6k \\ s+12k \\ s+18k \\ s+24k \end{array}\right\} & L>0, R>0, W=0 \\ \left.\begin{array}{l} s+2k \\ s+4k \\ s+8k \\ s+16k \end{array}\right\} & L>0, R=0, W=0 \\ \left.\begin{array}{l} s+3k \\ s+9k \\ s+27k \\ s+21k \end{array}\right\} & L=0, R>0, W=0 \\ \left.\begin{array}{l} s+5k \\ s+25k \end{array}\right\} & L=0, R=0, W>0 \\ s+k & L=0, R=0, W=0 \end{cases}$$

(note that $21 = 3^4 \bmod 30$). Finally, if $s = 1$ is the halt state, we define $a_i = 0$ and $b_i = 1$ for all $i = 1 + ck$; then a halting computation ends with the fixed point $x = 1$. (Note that we are throwing away the result of the computation.)

This base of $p = 30k$ is more compact than Conway's construction, which gives $p = \mathcal{O}(2^k)$ [2]. At most $19k$ terms are actually needed, since the other 11 possibilities $s + ck$ for $c \in \{7, 11, 13, 14, 17, 19, 22, 23, 26, 28, 29\}$ never happen. Furthermore, not all combinations of non-zero-ness of L, R and W occur for each state in the FSA.

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

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

This is everywhere analytic, and the fraction can be resolved using multiple-angle formulas. Specifically, setting $y = \pi x / p$, one can write

$$\frac{\sin py}{\sin y} = \frac{e^{ipy} - e^{-ipy}}{e^{iy} - e^{-iy}} = \sum_{k=0}^{p-1} e^{iky} e^{-i(p-1-k)y}$$

(so division is not necessary in our definition of U_n). Then

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

matches equation (1) on the integers. Finally, if our initial state is s_0 and we put the input entirely on the left half of the Turing machine, an input w corresponds to an initial value $x = k2^w + s_0$. So we have shown

Theorem 1. *For any Turing machine M with m tape symbols and any input w, there is an elementary function f of one variable, and constants k and s_0, such that M halts in time t if and only if $f^{t'}(k2^w + s_0) = 1$ for some $t' = \mathcal{O}(tm^t)$.* ■

From this it follows that predicting iterated functions is undecidable, for elementary functions in general and for particular functions f_U that simulate universal Turing machines.

3 Two dimensions

With two integer variables we can simulate a Turing machine directly. Let x encode the current state s and the right half of the tape, including the symbol a_0 at the head's current location, and let y encode the left half of the tape. If the TM has n states and m tape symbols, let

$$\begin{aligned} x &= s + n \sum_{i=0}^{\infty} (m+1)^i \, a_i \\ y &= \sum_{i=1}^{\infty} (m+1)^{i-1} \, a_{-i} \end{aligned}$$

where $0 \le s < n$ and $0 \le a_i < m$ for all i. We represent a blank as $a_i = 0$, which can also serve as an end marker, and we use base $m+1$ to avoid ambiguities like $1 = 0.999$. If we define

$$h_p(x) = \left(\frac{\sin \pi x}{p \, \sin \frac{\pi x}{p}} \right)^2$$

then we have (for integer t and a)

$$\begin{aligned} h_{(m+1)n}\,(x - (t + na)) &= \begin{cases} 1 \text{ if } s = t \text{ and } a_0 = a \\ 0 \text{ otherwise} \end{cases} \\ h_{m+1}(y - a) &= \begin{cases} 1 \text{ if } a_{-1} = a \\ 0 \text{ otherwise} \end{cases} \end{aligned}$$

Then let $s' = S_{s,a_0}$ be the Turing machine's new state, $a' = A_{s,a_0}$ be the symbol it writes on the tape, and Δ_{s,a_0} its movement left or right with the convention that $\Delta = 0$ for halt states and ± 1 for all non-halting states. Then if we define

$$(x_{\text{right}}, y_{\text{right}}) = \left(S_{s,a_0} + \tfrac{x - s - na_0}{m+1},\; (m+1)\, y + A_{s,a_0} \right)$$
$$(x_{\text{left}}, y_{\text{left}}) = \left(S_{s,a_0} + (m+1)\big(x - s + n(A_{s,a_0} - a_0)\big) + na_{-1},\; \tfrac{y - a_{-1}}{m+1} \right)$$

corresponding to shifting the machine to the right or left, we can simulate the TM with the function

$$f(x,y) = \sum_{s=0}^{n-1} \sum_{a_0=0}^{m-1} \Delta_{s,a_0}^2 \cdot h_{(m+1)n}\big(x-(s+na_0)\big) \times$$

$$\left[\left(\frac{1+\Delta_{s,a_0}}{2}\right) \cdot (x_{\text{right}}, y_{\text{right}}) + \left(\frac{1-\Delta_{s,a_0}}{2}\right) \sum_{a_{-1}=0}^{m-1} h_{m+1}(y-a_{-1}) \cdot (x_{\text{left}}, y_{\text{left}}) \right]$$

An initial TM state s_0 with an input w on the right half of the tape corresponds to an initial point $(s_0+nw, 0)$. If the machine erases the tape before halting, and if the halt state is $s=0$, halting is indicated by arriving at $(0,0)$. Of course, we can arrange to replace this with any pair $(x_{\text{halt}}, y_{\text{halt}})$ we like. So we have shown

Theorem 2. *For any Turing machine M and any input w, there is an elementary function f on two variables and constants a and b such that M halts on input w after t time-steps if and only if $f^t(a+bw,0)=(0,0)$.* ■

Finally, we note that this function has essentially one term for each state-symbol pair of the Turing machine. For Rogozhin's 4-state, 6-symbol machine [13] this is 24, although each of these terms has multiple elementary functions in it.

4 Conclusion

We have extended previous results on the computation universality of piecewise-linear and C^∞ functions in one and two dimensions to include analytic functions with elementary closed forms. Several open questions suggest themselves:

1.) Is there an elementary function in one dimension that can simulate Turing machines with a less-than-exponential slowdown? Continuous and Lipschitz functions can simulate Turing machines in real time, since (for instance) the function

$$\sigma(0.a_0a_{-1}a_1a_{-2}a_2\ldots) = 0.a_1a_0a_2a_{-1}a_3\ldots,$$

that shifts x's digits as if it were a two-sided sequence folded up, is continuous and can be embedded in a Lipshitz map on a Cantor set [10], but it is certainly not elementary. Any faster simulation by one-dimensional elementary functions would have to rely on a very different construction than the one given here.

2.) Can Turing machines be simulated by an analytic function on a compact space? TMs 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 [4], since if f has an infinite number of periodic points of period t, then f^t must be the identity; but such a simulation may still be possible in more than one dimension.

3.) What is the minimum number of terms in our two constructions needed to simulate a universal Turing machine? The figures given above (24 in two dimensions and $p \leq 19 \cdot 130 = 2470$ in one) can probably be reduced by noting as in [7] that not all combinations of state and symbol actually occur in the course of a properly initialized computation.

Acknowledgements: C.M. is grateful to Mats Nordahl and Seth Lloyd for helpful discussions, to an anonymous referee for pointing out that the functions in [10] are Lipschitz rather than differentiable, and to J. Felix Costa for a thorough reading of the manuscript.

References

[1] E. Asarin, O. Maler and A. Pnueli, "Reachability analysis of dynamical systems with piecewise-constant derivatives." *Theor. Comp. Sci.* **138** (1995) 35-66.

[2] S. Burckel, "Functional equations associated with congruential functions." *Theor. Comp. Sci.* **123** (1994) 397-406.

[3] J.H. Conway, "Unpredictable iterations." In *Proc. 1972 Number Theory* University of Colorado (1972) 49-52.

[4] P. Koiran, M. Cosnard, and M. Garzon, "Computability with low-dimensional dynamical systems." *Theor. Comp. Sci.* **132** (1994) 113-128.

[5] P. Koiran, "Puissance de calcul des réseaux de neurones artificiels." Ph.D. Thesis, Ecole Normale Supérieure de Lyon, 1993.

[6] J.C. Lagarias, "The $3x + 1$ problem and its generalizations." *Amer. Math. Monthly* **92** (1985) 3-23.

[7] K. Lindgren and M.G. Nordahl, "Universal computation in simple one-dimensional cellular automata." *Complex Systems* **4** (1990) 299-318.

[8] P. Michel, "Busy beaver competition and Collatz-like problems." *Arch. Math. Logic* **32** (1993) 351-367.

[9] C. Moore, "Unpredictability and undecidability in dynamical systems." *Phys. Rev. Lett.* **64** (1990) 2354, and *Nonlinearity* **4** (1991) 199.

[10] 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).

[11] M. Minsky, *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[12] 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.

[13] 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), *Theor. Comp. Sci.* **168** (1996) 215-240.

[14] H. Siegelmann and E.D. Sontag, "On the computational power of neural nets." *Journal of Computer and Systems Sciences* **50** (1995) 132-150.