Journal of Computer and System Sciences 132 (2023) 16–44

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

A physically universal Turing machine

Ville Salo *,[1], Ilkka Törmä [2]

Department of Mathematics and Statistics, University of Turku, Turku, Finland

ARTICLE INFO

Article history:
Received 8 December 2021
Received in revised form 12 July 2022
Accepted 30 August 2022
Available online 3 October 2022

Keywords:
Physical universality
Turing machine
Turmite
Dynamical system
Topological mixing

ABSTRACT

We construct a two-dimensional Turing machine that is physically universal in both the moving tape and moving head model. In particular, it is mixing of all finite orders in both models. We also provide a variant that is physically universal in the moving tape model, but not in the moving head model.

1. Introduction

In the theory of dynamical systems, one typically models states of the system as points in a space, and the time evolution as a self-map on this space. However, in many dynamical systems the state is more naturally seen as specifying the "contents" of many "regions" of an "underlying space". If a metric is needed on the space of all states, one postulates that two states resemble each other if the contents are similar in large regions of the underlying space.[3] A basic example are cellular automata, where the underlying space is discrete, and consists of a (possibly infinite) number of cells, each of which holds a symbol from a finite alphabet. One can also think of Newtonian dynamics as such a system: the underlying space is the three-dimensional Euclidean space, and the dynamics describes the interaction of objects occupying the space.

In the context of dynamical systems supporting a notion of underlying space, one can imagine reformulations of classical dynamical properties in terms of the space and its contents. Here, we mainly restrict to notions from topological dynamics. For example, topological transitivity intuitively means that given any bounded region D of the space and two sets of contents A, B for it, there is a state resembling A at D which eventually evolves into another state that resembles B at D. Topological transitivity implies some form of control over the evolution of the system: if one has the power to specify the contents of the space everywhere but in the domain D, then one can force the system to transform from A to B via its own dynamical rules. Weak and strong mixing, total transitivity and specification are examples of stronger notions of controllability considered in topological dynamics, where it is required that the dynamics is able to perform arbitrary transformations (possibly many successive ones) in a finite region, with constraints on the time the transformations take.

* Corresponding author.
E-mail addresses: vosalo@utu.fi (V. Salo), iatorm@utu.fi (I. Törmä).

[1] First author supported by Academy of Finland grant 2608073211.
[2] Second author supported by Academy of Finland grant 295095.

[3] If the state space happens to be a subspace of a function space S^T, it is natural to formalize this with the compact-open topology. However, the reader should keep a more informal mindset.

https://doi.org/10.1016/j.jcss.2022.08.003

The notion of physical universality was defined by Janzing in [9] as a notion of perfect controllability, for (classical and quantum) cellular automata and for systems defined by a Hamiltonian operator. In terms of dynamical systems with an underlying space in the sense of the previous paragraph, the intuition is that a system is physically universal if it is topologically transitive, and in the spatial definition of topological transitivity explained above, the contents of the space outside D and the time required for the transformation do not depend on A and B. Rather, one can implement any (suitably regular) function on the set of all contents of D by specifying the contents of its complement and an amount of time for which to allow the system to evolve. One should imagine a "machine" that can analyze the contents of D with arbitrary precision and then perform a predefined manipulation on the contents in finite time.

In the context of cellular automata, physical universality is given a formal definition in [9] (which we repeat in Section 3). Here, the considered regions are finite and the "machine" is the content of the cells outside the finite region. The existence of a physically universal cellular automaton was left open in [9], and a two-dimensional physically universal cellular automaton was later constructed in [21]. Since then, constructions of a quantum version [22] and a one-dimensional version [20] have also been published. All of these have the additional property that the minimum time required to implement a particular function depends polynomially on its descriptional complexity (when implemented as a Boolean circuit). Such automata were called efficiently physically universal in [21].

It would be interesting to be able to study physical universality in other dynamical systems. Following the informal discussion above, one can imagine what physical universality should roughly mean for a system supporting a notion of underlying space. For example, in the context of Newtonian mechanics, one might say that Newtonian dynamics itself is physically universal if we can perform the following kinds of experiments: specify a cubical region of space and place such a collection of objects with specific momenta outside it, that when the cube is filled with stationary spherical objects of some fixed size, after exactly three minutes it will contain the same number of objects in approximately the same positions if their number was odd, and no objects at all if their number was even.

Formalizing this idea, however, is not easy, even in the context of topological dynamics. To us, of particular interest are the following three questions:

1. How to formalize the notion of underlying space of a topological dynamical system?
2. How to formalize physical universality for a class of topological dynamical systems supporting a notion of an underlying space?
3. Once this has been defined for a particular class of topological dynamical systems, can we find a PU system in the class?

We do not solve the general problem, but we present a solution to the latter two questions for a new class of systems for which the notion of space is intuitively clear: Turing machines, which have been studied as topological dynamical systems by several authors [17,12,2,5,10]. There are two standard ways of seeing a Turing machine as a topological dynamical system, which were introduced in [12], namely the moving head and moving tape models. In this paper, we define a notion of physical universality for Turing machines in both models, and present a two-dimensional Turing machine that is efficiently physically universal in both models. Of course, whether our formalization of PU is "correct" is up to debate (and we point out some deficiencies of the definition ourselves), but we believe it is in the correct spirit.

Our PU Turing machine may also be of independent interest. It resembles the famous Langton's ant [13] in two ways. First, its internal state stores a cardinal direction in which it moves on the two-dimensional tape, and its local rule is (almost) invariant under rotations. We see it as a generalized turmite. Turmites have very simple local dynamics, yet all nontrivial turmites are capable of universal computation in a certain sense [15], so they are natural candidates for physical universality.[4] Second, our machine has no periodic orbits in the moving head model, meaning that the Turing machine head eventually escapes every finite region. This was proved for Langton's ant in [3], and it is a necessary condition for physical universality in both models. Langton's ant was also proved to be topologically transitive (and mixing up to a parity condition) in the moving tape model in [4], which is implied for our machine by physical universality. The dynamics of Langton's ant has been studied further in e.g. [7].

Some other properties of interest that we prove are that our machine in fact escapes all finite rectangular regions in polynomial time in the size of the region, and is topologically mixing in the moving head model. To our knowledge it is the first proven example of the latter behavior in Turing machines. In addition to our main construction, we present a Turing machine which is physically universal in the moving tape model but not the moving head model (it is not even topologically transitive in the latter). We also discuss the invariance properties of PU on Turing machines and possible strengthenings of our definition, and include many questions.

We constructed the proof on the blackboard, except that the trick shown in Fig. 13 was found by trial and error in a computer simulation. With the exception of Example 2 (which is not crucial for the results), computer simulation is not necessary for following the proof. Nevertheless, for readers interested in exploring our rule $\bar{M}$, we have included a @RULE file used by the Golly cellular automaton simulator.

[4] At present, we cannot determine the physical universality status of any nontrivial turmites in the sense of [15].

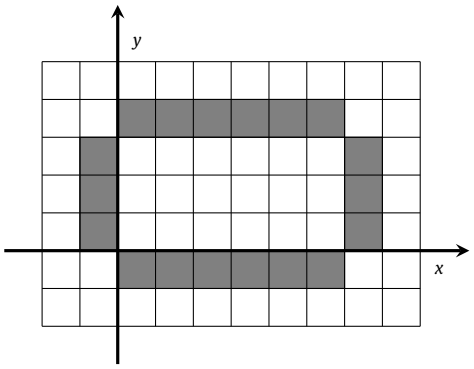


Fig. 1. The outer $(6, 3)$-border $B_{6,3}$.

2. Preliminaries

Partial functions from a set A to another set B are denoted $f : A \dashrightarrow B$. The set of finite words over an alphabet A is denoted A^*. The empty word is denoted by λ. The disjoint union of A and B is denoted by $A \dot\cup B$. Our $\mathbb{N}$ includes $\mathbb{N} \ni 0$, and $\mathbb{Z}_+ = \mathbb{N} \setminus \{0\}$.

Fix a dimension d. The *full shift* over a finite alphabet Σ is the set $\Sigma^{\mathbb{Z}^d}$, whose elements are called *configurations*. For any subset $N \subset \mathbb{Z}^d$, a *shape-N pattern* is an element $P \in \Sigma^N$. The pattern *occurs* in $x \in \Sigma^{\mathbb{Z}^d}$ if for some $\vec{v} \in \mathbb{Z}^d$ we have $x_{\vec{v}+\vec{w}} = P_{\vec{w}}$ for all $\vec{w} \in N$. A *subshift* is a subset of $\Sigma^{\mathbb{Z}^d}$ defined by a set F of finite patterns as the set of those configurations where no $P \in F$ occurs. Denote by $\tau : \mathbb{Z}^d \times \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ the translation action defined by $\tau_{\vec{v}}(x)_{\vec{w}} = x_{\vec{w}+\vec{v}}$. Every subshift X clearly satisfies $\tau_{\vec{v}}(X) = X$. Writing $x|_N$ for the restriction of x to the set N, a pattern P occurs in x if and only if $\exists \vec{v} : \tau_{\vec{v}}(x)|_N = P$.

For a finite set Q and $\# \notin Q$, let $X_Q \subset (Q \cup \{\#\})^{\mathbb{Z}^d}$ be the subshift of those configurations that contain at most one occurrence of an element of Q. For $q \in Q$ and $\vec{v} \in \mathbb{Z}^d$, denote by $q[\vec{v}]$ the unique configuration $x \in X_Q$ with $x_{\vec{v}} = q$. For $N \subset \mathbb{Z}^d$, we denote by $\mathcal{P}_{Q,\Sigma}(N)$ the shape-N patterns occurring in $X_Q \times \Sigma^{\mathbb{Z}^d}$, by $\mathcal{P}^*_{Q,\Sigma}(N)$ those where an element of Q occurs, and by $\mathcal{P}^0_{Q,\Sigma}(N)$ where one does not. We refer to patterns in $\mathcal{P}^*_{Q,\Sigma}(N)$ as *headful* (or more explicitly as *patterns contain a head*), and to patterns in $\mathcal{P}^0_{Q,\Sigma}(N)$ as *headless* (or more explicitly as *patterns that do not contain a head*). For $n \in \mathbb{N}$, we denote $\mathcal{P}_{Q,\Sigma}(n) = \mathcal{P}_{Q,\Sigma}([0, n-1]^d)$, and similarly for $\mathcal{P}^*_{Q,\Sigma}(n), \mathcal{P}^0_{Q,\Sigma}(n)$.

A d-dimensional Turing machine is a 4-tuple $M = (Q, \Sigma, N, \Delta)$, where Q is a finite *state set*, Σ is a finite *alphabet*, $N \subset \mathbb{Z}^d$ is a finite *neighborhood* and $\Delta : (Q \times \Sigma^N) \to (Q \times \Sigma^N \times N)$ is a *transition function*. A number $r \in \mathbb{N}$ is a *radius* of M if $N \subset [-r, r]^d$, and *the radius* refers to its minimal radius.

The machine is associated with two *topological dynamical systems*, meaning continuous self-maps of a topological space. In the *moving head model*, the state space is $X_Q \times \Sigma^{\mathbb{Z}^d}$, and the dynamics is given by $M(\#^{\mathbb{Z}^d}, x) = (\#^{\mathbb{Z}^d}, x)$ and $M(q[\vec{v}], x) = (p[\vec{v} + \vec{w}], y)$, where $\Delta(q, \tau_{\vec{v}}(x)|_N) = (p, P, \vec{w})$ and y is obtained from x by replacing the contents of the translated domain $\vec{v} + N$ with P. In the *moving tape model*, the state space is $Q \times \Sigma^{\mathbb{Z}^d}$, and the dynamics is given by $M(q, x) = (p, \tau_{\vec{w}}(y))$, where $\Delta(q, x|_N) = (p, P, \vec{w})$ and y is obtained from x by replacing the contents of N by P. In each case, the topology on $\Sigma^{\mathbb{Z}^d}$ is the product topology, and on $\mathbb{Z}^d$ and finite sets the discrete topology.

A *(d-dimensional) cellular automaton* is a function $f : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ which for some finite *neighborhood* $N \subset \mathbb{Z}^d$ is defined by a *local rule* $f_{\mathrm{loc}} : \Sigma^N \to \Sigma$ by $f(x)_{\vec{v}} = f_{\mathrm{loc}}(\tau^{\vec{v}}(x)|_N)$. One often refers to Σ as the *alphabet*. All Turing machines in the moving head model can be obtained as (domain and codomain) restrictions of cellular automata on the alphabet $(Q \cup \{\#\}) \times \Sigma$ to self-maps of $X_Q \times \Sigma^{\mathbb{Z}^d}$, though we note that there is typically no unique natural extension of a Turing machine to a cellular automaton, and not every cellular automaton admitting a well-defined restriction to $X_Q \times \Sigma^{\mathbb{Z}^d}$ is a Turing machine.

An important subset of $\mathbb{Z}^2$ is the following: For $m, n > 0$, the *outer* (m, n)*-border* is the set $B_{m,n} = \{-1, m\} \times [0, n-1] \cup [0, m-1] \times \{-1, n\} \subset \mathbb{Z}^2$, see Fig. 1. Seeing $\mathbb{Z}^2$ as an undirected graph with edges $\{\{(m, n), (m + a, n + b)\} \mid |a| + |b| = 1\}$, the outer (m, n)-border consists of the coordinates that do not belong to the set $[0, m-1] \times [0, n-1]$ but have a neighbor that does. We denote $B_n = B_{n,n}$.

3. Physical universality in Turing machines

The *physical universality* (or *PU*) of a d-dimensional cellular automaton f on alphabet Σ is defined as follows: For all finite domains $D \subset \mathbb{Z}^d$ and all functions $g : \Sigma^D \to \Sigma^D$, there exists a partial configuration $y \in \Sigma^{\mathbb{Z}^d \setminus D}$ and a time $t \in \mathbb{N}$ such that for all patterns $P \in \Sigma^D$ we have $f^t(P \sqcup y)|_D = g(P)$. We can assume that the domain D is square-shaped. Since a d-dimensional Turing machine $M = (Q, \Sigma, N, \Delta)$ in the moving head model can be seen as a restriction of a certain cellular

automaton f on $((Q \cup \{\#\}) \times \Sigma)^{\mathbb{Z}^d}$ to the set $X_Q \times \Sigma^{\mathbb{Z}^d}$, it would make sense to define physical universality of M by the analogous condition.

However, there are two complications that make the definition vacuous. First, some patterns in $\mathcal{P}_{Q,\Sigma}(D)$ contain the head of M, so any partial configuration $y \in ((Q \cup \{\#\}) \times \Sigma)^{\mathbb{Z}^d \setminus D}$ that satisfies $y \sqcup P \in X_Q \times \Sigma^{\mathbb{Z}^d}$ for all $P \in \mathcal{P}_{Q,\Sigma}(D)$ cannot contain the head of M. But if $P \in \mathcal{P}_{Q,\Sigma}(D)$ does not contain the head either, $y \sqcup P$ is then a fixed point of M. Hence we restrict the universal quantification to the set of headful patterns $\mathcal{P}^*_{Q,\Sigma}(D)$, and require $y \in \Sigma^{\mathbb{Z}^d \setminus D}$ to not contain the head of M. One could also restrict to patterns that do *not* contain the head, but this in fact gives a weaker definition.

The second and more subtle complication is that apart from trivial cases, there always exist distinct patterns $P, P' \in \mathcal{P}^*_{Q,\Sigma}(D)$ that satisfy $P \sqcup y = M(P' \sqcup y)$ for all $y \in \Sigma^{\mathbb{Z}^d \setminus D}$. This implies $M^t(P \sqcup y) = M^{t+1}(P' \sqcup y)$ for all $t \in \mathbb{N}$, which makes it impossible to implement arbitrary functions $g : \mathcal{P}^*_{Q,\Sigma}(D) \to \mathcal{P}^*_{Q,\Sigma}(D)$. To get around this problem, we restrict our attention to subsets of $\mathcal{P}^*_{Q,\Sigma}(D)$ that are in disjoint M-orbits when completed into full configurations by filling $\mathbb{Z}^d \setminus D$ by a fixed symbol $0 \in \Sigma$.

More precisely, we say a tuple of patterns $(P_0, \dots, P_{k-1}) \in (\mathcal{P}^*_{Q,\Sigma}(m))^k$ has *disjoint 0-orbits* if the M-orbits of $P_i \sqcup 0^{\mathbb{Z}^d \setminus [0,m-1]^d}$ (in the moving head model) are pairwise disjoint for distinct i. Similarly, we say that a tuple $((q_0, P_0), \dots, (q_{k-1}, P_{k-1})) \in Q \times \Sigma^{[-m,m]^d}$ has *disjoint 0-orbits* if the configurations $(q_j, P_j \sqcup 0^{\mathbb{Z}^d \setminus [-m,m]^d})$ have pairwise disjoint M-orbits (in the moving tape model).

Definition 1. Let $M = (Q, \Sigma, N, \Delta)$ be a d-dimensional Turing machine with $0 \in \Sigma$. Let $P = (P_0, \dots, P_{k-1}) \in (\mathcal{P}^*_{Q,\Sigma}(m))^k$ and $(R_0, \dots, R_{k-1}) \in (\mathcal{P}_{Q,\Sigma}(m))^k$ be tuples of patterns. We say *P is physically transformable to R in the moving head model (in time $t \in \mathbb{N}$)*, if there exists a partial configuration $x \in \mathcal{P}_{Q,\Sigma}(\mathbb{Z}^d \setminus [0, m-1]^d)$ such that $M^t(P_j \sqcup x)|_{[0,m-1]^d} = R_j$ for all $j \in [0, k-1]$. Similarly, if $P = ((q_0, P_0), \dots, (q_{k-1}, P_{k-1}))$, $R = ((p_0, R_0), \dots, (p_{k-1}, R_{k-1})) \in (Q \times \Sigma^{[-m,m]^d})^k$, we say *$P$ is physically transformable to R in the moving tape model (in time $t \in \mathbb{N}$)* if there exists a partial configuration $x \in \Sigma^{\mathbb{Z}^d \setminus [-m,m]^d}$ such that $M^t(q_j, P_j \sqcup x) = (p_j, x_j)$ with $x_j|_{[-m,m]^d} = R_j$ for all $j \in [0, k-1]$.

We say M is *physically universal in the moving head model* if for all $k, m \in \mathbb{N}$, for any $P, R \in (\mathcal{P}^*_{Q,\Sigma}(m))^k$ such that the patterns P_i have disjoint 0-orbits, P is physically transformable to R. The machine M is *efficiently physically universal in the moving head model*, if the time t can be bounded by a polynomial in the circuit complexities of the set of patterns $\{P_0, \dots, P_{k-1}\}$ and the function $P_i \mapsto R_i$. *(Efficient) physical universality in the moving tape model* is defined analogously.

The concept of "physically transformable" allows many different definitions of physical universality, and it is hard to say which one is the best. If $k = 1$, P is physically transformable to R for all patterns (more precisely 1-tuples of patterns) P, R if and only if M is topologically transitive. This is true in either model; for the moving tape model it is true almost by definition, while for the moving head model it requires a short proof (which we omit). The precise definitions of PU above are optimized for machine M, see Section 13 for some variants.

Our main result is that efficiently physically universal Turing machines exist.

Theorem 1. *There exists a two-dimensional Turing machine which is efficiently physically universal in both the moving tape and the moving head model.*

This machine M acts on the binary tape. It has 5 internal states and radius 1. The machine does not have particularly good symmetry properties, as we have to break its symmetry to avoid a parity issue. However, M is a trivial modification of a very natural Turing machine which we call $\bar{M}$. The machine $\bar{M}$ has 4 states which carry only its orientation. It is time-symmetric,[5] symbol-conserving (Definition 3) and has rotational symmetry (by 90-degree rotations). The machine also has a natural escape property crucial in our proofs, see Lemma 5.

In Section 12 we sketch some additional constructions obtained by modifying the Turing machine. The following is proved by adding an invariant dynamical factor in the moving head model.

Theorem 2. *There exists a Turing machine that is physically universal in the moving tape model, but not the moving head model.*

Our definition of a Turing machine is the general one from [1], and readers may wonder if the ability to see several cells at once is essential. It is not: we show that any Turing machine in our sense can be simulated by a *classical Turing machine*, namely a Turing machine that performs a permutation of the tape, and then moves in a cardinal direction or stays put as a function of the current state. In particular, we obtain the following.

[5] This means it is conjugate to its inverse by an involution. In the moving head model, the involution lives in the extended symmetry group [16] of the subshift $X_Q \times \Sigma^{\mathbb{Z}^d}$. See Lemma 3 and Section 13.4 for precise statements.

Theorem 3. *There exists a classical Turing machine that is efficiently physically universal in both the moving tape and the moving head model.*

4. Simulation of Turing machines by circuits

We begin by dealing with the simple computational issues that arise from the fact our set of initial conditions A is a (typically proper) subset of the headful patterns $\mathcal{P}^*_{Q,\Sigma}(m)$. The main observation is that if A is efficiently computable, then we can recover the original pattern and the time elapsed so far from the state after the head exits the region $[0, m-1]^d$, assuming the region is exited quickly.

Definition 2. Let $D \subset \mathbb{Z}^d$, and let $M = (Q, \Sigma, N, \Delta)$ be a d-dimensional reversible Turing machine. For a configuration $x = (q[\vec{v}], y) \in X_Q \times \Sigma^{\mathbb{Z}^d}$ and $D \subset \mathbb{Z}^2$ with $\vec{v} \in D$, we denote by $\epsilon^M_D(x)$ the smallest number $t \geq 0$ with $M^t(x) = (p[\vec{w}], z)$ where $\vec{w} \notin D$. It is called the *D-escape time* of x. We also denote by $\rho^M_D(x) = M^{\epsilon^M_D(x)}(x)$ the configuration right after the escape.

Fix a special alphabet symbol $0 \in \Sigma$, and let $r \geq 0$ be the minimal radius of M. For a headful pattern $P \in \mathcal{P}^*_{Q,\Sigma}(D)$, we denote $\epsilon^M(P) = \epsilon^M_D(x)$ and $\rho^M(P) = M^{\epsilon^M(P)}(x)|_E$, where $x = P \sqcup 0^{\mathbb{Z}^d \setminus D}$ is the configuration containing P in a sea of 0-symbols and $E = D + [-r, r]^d$ is the domain D plus a padding of thickness r.

We can simulate a Turing machine with a cellular automaton, and a cellular automaton with a circuit, which gives rise to the following result.

Lemma 1. *Let $r \geq 0$, and let $M = (Q, \Sigma, N, \Delta)$ be a d-dimensional reversible Turing machine with $0 \in \Sigma$ and radius r. Let $A \subset \mathcal{P}^*_{Q,\Sigma}(m)$ be a set of headful m^d-patterns having disjoint 0-orbits and such that ρ^M is defined on each pattern of A. Then ρ^M is injective on A, and the circuit complexity of the function $\rho^M(P) \mapsto (P, \epsilon^M(P))$ is $O(T(C + (m+2r)^d))$, where C is the circuit complexity of A and $T = \max\{\epsilon^M(P) \mid P \in A\}$.*

Proof. Denote $D = [0, m-1]^d$ and $E = [-r, m-1+r]^d$. The injectivity of ρ^M follows from the reversibility of M, and the fact that it cannot affect any cells in the region $\mathbb{Z}^d \setminus E$ before leaving D.

Let f be a cellular automaton on $((Q \cup \{\#\}) \times \Sigma)^{\mathbb{Z}^d}$ whose restriction to $X_Q \times \Sigma^{\mathbb{Z}^d}$ implements the reverse machine M^{-1} in the moving head model. Then r is a radius for f. We construct a Boolean circuit arranged in T 'layers', numbered from 0 to $T-1$. The kth layer contains $(m+2r)^d$ copies of a circuit S that computes the local rule $F : ((Q \cup \{\#\}) \times \Sigma)^{[-r,r]^d} \to (Q \cup \{\#\}) \times \Sigma$ of f, which can be visualized as cells in a d-dimensional grid. On layer 0 the $(2r+1)^d$ input symbols of each copy of S are taken from the input values, and on layer $k > 0$ from the outputs of the copies of S of layer $k-1$. Those copies of S that would take their inputs from outside the grid receive 0-symbols instead.

In addition to the next layer, the outputs of layer k are passed to a copy of a size-$O(C + m^d)$ circuit that computes membership in the set $A' = \{P \sqcup 0^{E \setminus D} \mid P \in A\}$. With $O(T + m^d)$ additional gates, we can decide whether k is the first layer whose pattern is in A', and if so, copy the pattern into the output wires of the circuit. We also copy the information about k to the output.

For any $P \in A$, we have $M^{-\epsilon^M(P)}(\rho^M(P) \sqcup 0^{\mathbb{Z}^d \setminus E}) = P \sqcup 0^{\mathbb{Z}^d \setminus D}$, and $\epsilon^M(P)$ is the smallest number for which the central pattern of the resulting configuration is in A', since the patterns in A have disjoint 0-orbits and the head does not leave the region D before $\epsilon^M(P)$ steps when started from P. If we feed $\rho^M(P)$ to the circuit constructed above, then it will compute the smallest number $t \in \mathbb{N}$ with $M^{-t}(\rho^M(P) \sqcup 0^{\mathbb{Z}^d \setminus E})|_E \in A'$, and this number is exactly $\epsilon^M(P)$. If the head leaves the region D during the simulation, then it will do so after $\epsilon^M(P)$ steps, and any computation results made by the circuit after that are irrelevant (and probably incorrect). In particular, the circuit will correctly simulate M^{-1} for at least $\epsilon^M(P)$ steps. Thus the circuit implements the desired function. □

5. The machine

In this section we present a physically universal two-dimensional Turing machine. It has five states and uses a binary alphabet. It is a *generalized turmite*, meaning that the states correspond to the four cardinal directions plus some auxiliary data, and the machine operates by scanning and manipulating its surroundings, possibly changing its direction, and possibly taking one step forward.

The formal definition of the machine M is as follows. The tape alphabet is $\Sigma = \{0, 1\}$ and the state set is $Q = \{\uparrow, \uparrow^*, \rightarrow, \leftarrow, \downarrow\}$. One step of M is divided into three phases. In the first phase we perform one of the (non-overlapping) local transformations shown in (1), rotated by any multiple of 90 degrees (rotating also the arrows), where $a \in \{0, 1\}$ is arbitrary:

$$\begin{array}{|c|c|}\hline a & 1 \\ \hline 0\uparrow & a \\ \hline\end{array} \leftrightarrow \begin{array}{|c|c|}\hline a & 0 \\ \hline 1\leftarrow & a \\ \hline\end{array} \tag{1}$$

Note that the transition can be applied in both directions: the pattern on the right hand side is replaced by the pattern on the left hand side. In all cases, a single 1 is moved diagonally toward the inner side of a turn on the path that the head of M traces. If no transformation is applicable, the machine maintains its position and state. Note that $\uparrow^*$ does not occur in these patterns.

In the second phase, the head moves one step in the direction indicated by its state, unless the state is $\uparrow^*$, in which case it retains its state and position. In the third phase, the head changes its state from $\uparrow$ to $\uparrow^*$ or vice versa if applicable, and retains its state if not. See Fig. 2 for an example of the phases.

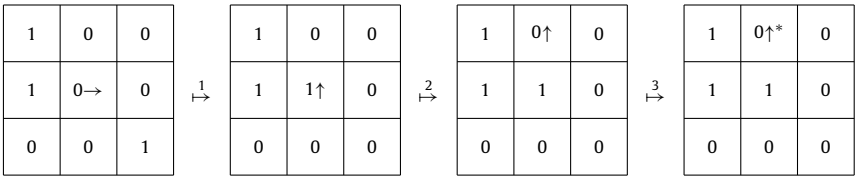


Fig. 2. A step of M broken into phases.

Since each phase is specified by a bijection (the first and third phases are actually involutions), M is reversible. The machine also conserves the number of 1s in the configuration, i.e. it is symbol-conserving in the following sense.

Definition 3. A Turing machine $M = (Q, \Sigma, N, \Delta)$ is *symbol-conserving* if for all $(q[\vec{v}], x) \in X_Q \times \Sigma^{\mathbb{Z}^2}$, if $M(q[\vec{v}], x) = (p[\vec{w}], y)$ then there exists a permutation $\pi \in \mathrm{Sym}(\mathbb{Z}^2)$ with finite support such that $y = \pi(x)$, where $\mathrm{Sym}(\mathbb{Z}^2)$ acts on configurations by $\pi(x)_{\vec{v}} = x_{\pi^{-1}(\vec{v})}$.

We also define a simpler (and nicer) auxiliary machine $\bar{M}$ with state set $\bar{Q} = \{\uparrow, \rightarrow, \leftarrow, \downarrow\}$ and tape alphabet Σ, and which behaves by first applying one of the transformations in (1) if possible, and then advancing for one step in the direction of the head. Like M, it is reversible and symbol-conserving. The following lemma formalizes the correspondence between the machines.

Lemma 2. *For each $z \in X_{\bar{Q}} \times \Sigma^{\mathbb{Z}^2}$ we have $\bar{M}(z) = M^k(z)$ for some $k \in \{1, 2\}$, and for each $y \in X_Q \times \Sigma^{\mathbb{Z}^2}$ we have $M^k(y) \in X_{\bar{Q}} \times \Sigma^{\mathbb{Z}^2}$ for some $k \in \{0, 1\}$.*

Remark 1. To readers familiar with the notion of *suspension* of a dynamical system (which turns it from a discrete system to one with continuous-time by "artificially" adding a continuum of intermediate states between the discrete steps [14]), we note that one can sharpen this lemma and show a *flow equivalence* between the machines, i.e. the suspensions of the two machines M and $\bar{M}$ are equivalent up to time-dilation. ○

We now show a connection between $\bar{M}$ and its inverse. For this, we define a few auxiliary functions. Denote the *mirror map* by $\mu(a, b) = (-a, b)$ for $(a, b) \in \mathbb{Z}^2$, and extend it to $\Sigma^{\mathbb{Z}^2}$ by $\mu(x)_{\vec{w}} = x_{\mu(\vec{w})}$. Then extend it to $X_{\bar{Q}} \times \Sigma^{\mathbb{Z}^2}$ by $\mu(q[\vec{v}], x) = (q'[\mu(\vec{v})], \mu(x))$, where q' points in the direction opposite to q if q is horizontal, and $q' = q$ otherwise. Define the *bit flip map* by $\beta(x)_{\vec{v}} = 1 - x_{\vec{v}}$ for $x \in \Sigma^{\mathbb{Z}^2}$ and the *opposite map* by $\omega(q[\vec{v}]) = \hat{q}[\vec{v} - \vec{w}]$, where $\hat{q}$ points in the direction opposite to q, and $\vec{w}$ is the neighbor of $\vec{0}$ in the direction of q (in other words, we flip the arrow over its tail). Finally, define $\sigma(q[\vec{v}], x) = \mu(\omega(q[\vec{v}]), \beta(x))$.

Lemma 3. *The inverse of $\bar{M}$ satisfies $\sigma \circ \bar{M}^{-1} \circ \sigma = \bar{M}$.*

Proof. This follows from a simple case analysis. □

The idea behind defining both M and $\bar{M}$ is that while the latter has a simpler structure, it suffers from a parity issue. Since the head of $\bar{M}$ always moves to an adjacent coordinate, $\bar{M}^n(q[(a, b)], x) = (p[(c, d)], y)$ implies that the number $n + a + b + c + d$ is even. The machine M uses a modulo-2 counter to simulate one step of $\bar{M}$ in two steps in those configurations where the head travels to the north, which allows breaking any period by having the head take suitable detours.

We now prove that M has no periodic (headful) points in the moving head model, similarly to Langton's ant, and also that it can only escape a finite island of 1s by stepping out of it and walking to infinity along a straight line. Lemma 2 allows us to use $\bar{M}$ in place of M in the proof.

Lemma 4. *The machine $\bar{M}$ cannot make two right turns in a row.*

Proof. After $\bar{M}$ has made a right turn, it has a 1 on its right and a 0 behind it, so it cannot make another right turn. □

Lemma 5. *Let $\vec{u} \in \mathbb{Z}^2$ and $(q[\vec{v}], x) \in X_Q \times \Sigma^{\mathbb{Z}^2}$ be such that $\vec{v} \in \vec{u} + [0, n-1]^2$. Then there exists $t = O(n^4)$ with $M^t(q[\vec{v}], x) = (p[\vec{w}], y)$ and $\vec{w} \notin \vec{u} + [0, n-1]^2$.*

Proof. We prove the analogous result for $\bar{M}$, from which the claim follows by Lemma 2.

We assume without loss of generality that $\vec{u} = (1, 1)$, and denote $R = [1, n]^2$ and $R' = [0, n+1]^2$. Denote the Manhattan norm of a vector $\vec{v} = (x, y)$ by $|\vec{v}|_1 = |x| + |y|$. We define a potential function $p : X_{\bar{Q}} \times \Sigma^{\mathbb{Z}^2} \to \mathbb{Z}$ by $p(q[\vec{v}], x) = p_1(q, \vec{v}) + p_2(x)$, where

$$\begin{aligned} p_1(\rightarrow, \vec{v}) &= -2|\vec{v}|_1 + 2 \\ p_1(\uparrow, \vec{v}) &= -2|\vec{v}|_1 \\ p_1(\leftarrow, \vec{v}) &= 2|\vec{v}|_1 + 2 \\ p_1(\downarrow, \vec{v}) &= 2|\vec{v}|_1 \end{aligned}$$

and $p_2(x) = \sum_{\vec{v} \in R'} x_{\vec{v}} \cdot |\vec{v}|_1^2$. Concretely, p_2 is the sum of squares of the Manhattan distances of each 1 in the extended square R' to the southwest corner of R'. The intuition is that when the head of $\bar{M}$ makes a loop, it pulls 1s closer to the center of R', which decreases the value of p_2. We use p_1 as a balancing term to ensure that p decreases at least every two steps. More explicitly, we claim that $p(y) > p(M^2(y))$ holds for all $y \in X_{\bar{Q}} \times \Sigma^{\mathbb{Z}^2}$ where the head lies in R. The maximum value of p is $O(n^4)$, and its minimum value on configurations whose head lies in R is $-4n$, so the result follows from this.

Suppose that the head of $\bar{M}$ is at a coordinate $\vec{v} \in R$. If $\bar{M}$ does not make a turn on this step, then the value of p_1 decreases by 2 regardless of the state and position of the head, and p_2 retains its value, so that p also decreases by 2.

Suppose that $\bar{M}$ makes a right turn. If the head was facing east, then the value of p_1 increases by $4|\vec{v}|_1 - 4$ and p_2 decreases by $4|\vec{v}|_1 - 4$, which means that p retains its value. If the head was facing west, then p_1 decreases by $4|\vec{v}|_1 + 4$ and p_2 increases by $4|\vec{v}|_1 + 4$, so p again retains its value. If the head was facing north or south, then both p_1 and p_2 retain their value. Thus the value of p stays the same on a right turn.

Suppose then that $\bar{M}$ makes a left turn. If the head was facing north, then p_1 increases by $4|\vec{v}|_1$ and p_2 decreases by $4|\vec{v}|_1 + 4$, so p decreases by 4. If the head was facing south, then p_1 decreases by $4|\vec{v}|_1$ and p_2 increases by $4|\vec{v}|_1 - 4$, so p decreases by 4. If the head was facing east or west, then p_1 decreases by 4 and p_2 retains its value. Thus the value of p always decreases by 4 on a left turn.

All in all, the value of p never increases, and only stays the same on a right turn. By Lemma 4, $\bar{M}$ cannot make two consecutive right turns, so the claim follows. □

Corollary 1. *The machine M has no nontrivial periodic points in the moving head model.*

Lemma 6. *Let $x = (q[\vec{v}], y) \in X_Q \times \Sigma^{\mathbb{Z}^2}$ be a configuration with a finite number of 1s, and let $P = [a, b] \times [c, d]$ be a finite rectangular region such that $\vec{v} \in P$ and $y_{\vec{w}} = 0$ for all $\vec{w} \notin P$. For all $k \geq 0$, the configuration $M^k(x)$ does not contain a 1 outside $P' = [a-1, b+1] \times [c-1, d+1]$.*

Proof. We again prove the result for $\bar{M}$ and apply Lemma 2.

Let $n \geq 0$ be the lowest integer such that $\bar{M}^{n+1}(x)$ contains a 1 outside of P, if one exists. We assume without loss of generality that the head is pointing north in $\bar{M}^n(x)$ at some coordinate $(i, j) \in \mathbb{Z}^2$.

We first observe that for all $m < n$, the head of $\bar{M}^m(x)$ is inside P. Namely, if the head were outside P, then it could not make a turn, since that would involve moving a 1 out of P or standing on a 1 that is already outside P, neither of which happens before n steps. Since P is rectangular, the head can never return near it without turning once it has been exited, and hence could not possibly move a 1 outside it in the transition from $\bar{M}^n(x)$ to $\bar{M}^{n+1}(x)$.

The transition from $\bar{M}^n(x)$ to $\bar{M}^{n+1}(x)$ is either a left or right turn, and in either case, a 1 is moved from P to its complement. Suppose first that the turn is to the left, so that $\bar{M}^n(x)_{(i,j)} = 0$ and $(i, j) \notin P$. Since the head is inside P in $\bar{M}^{n-1}(x)$, we have $(i, j-1) \in P$. The rectangular shape of P implies $(i+1, j+1) \notin P$, so this coordinate contains a 0 in $\bar{M}^n(x)$. Then the head cannot make a left turn from $\bar{M}^n(x)$, a contradiction.

Suppose then that $\bar{M}$ turns to the right, so that $\bar{M}^n(x)_{(i,j)} = 1$. Then we have $(i, j) \in P$ and $(i+1, j-1) \notin P$. We split into two cases depending on whether P contains $(i, j-1)$. Suppose first that $(i, j-1) \in P$ and denote $R = [i+1, \infty) \times [j-1, j+1] \subset \mathbb{Z}^2$. Since P is rectangular, we have $R \cap P = \emptyset$, so that $\bar{M}^{n+1}(x)_{\vec{v}} = 0$ for all $\vec{v} \in R \setminus \{(i+1, j-1)\}$. In $\bar{M}^{n+1}(x)$, the head is at $(i+1, j)$ facing east, after which it keeps traveling east inside the region R, never making a turn again.

Suppose then that $(i, j-1) \notin P$. Since P is rectangular, we have $(i', j-1) \notin P$ and hence $\bar{M}^n(x)_{(i', j-1)} = 0$ for all $i' \in \mathbb{Z}$. Hence the head travels east along the south border of P until it either exits P or makes another right turn, after which it can never make a turn again. In all cases, no 1s are moved outside P'. □

The proof also shows that once the head leaves the region $[a-1, b+1] \times [c-1, d+1]$, it never makes a turn again, traveling in a straight line forever. We will later use this fact to construct a gadget that catches the head and redirects it into a system of circuitry. Using Lemma 2 and Lemma 3, we obtain the analogues of Lemma 5 and Lemma 6 for M^{-1}, but with the roles of 0 and 1 reversed.

Remark 2. It is a well-known conjecture that Langton's ant has a similar property as the one proved in Lemma 5: that started from a configuration of finite support, the head eventually leaves the pattern, and begins traveling in a rational direction in an eventually periodic pattern (in particular the spacetime diagram is semilinear). Since Langton's ant and $\bar{M}$ have the same state set and tape alphabet, one may wonder how similar they are dynamically. They are quite different: The distinct limit behavior on configurations with a finite number of nonzero symbols shows that they are not conjugate by linear automorphisms of $\mathbb{Z}^2$ and automorphisms of $X_Q \times \Sigma^{\mathbb{Z}^2}$. In fact, since Langton's ant leaves "garbage" behind it on its diagonal journeys, the two machines cannot even admit spatiotemporal blockings that are conjugate by linear transformations of the tape and conjugacies between the configuration spaces. ○

6. Sequential automata

In this section, we define a class of token-based circuits that will later be simulated by M. We do this in the framework of sequential automata; see [19] for a classical overview on this subject. We extend this formalism slightly by introducing a time parameter that tracks the progress of the token inside the circuit. The times we deal with will always be nonnegative integers.

Definition 4. A *(deterministic timed) sequential automaton* or *s-automaton* is a 5-tuple $A = (Q, I, O, \delta, \mu)$, where Q is a finite set of *internal states*, I is a finite set of *input terminals*, O is a finite set of *output terminals* disjoint from I, $\delta : Q \times I \dashrightarrow Q \times O$ is the *partial transition function*, and $\mu : Q \times I \dashrightarrow \mathbb{Z}_+$ is the *partial time function* that assigns a time to each transition, and is defined on the same set as δ is. If μ is a constant function, we may replace it with said constant value.

We extend δ into a partial function from $Q \times I^*$ to $Q \times O^*$ by defining $\delta(q, \lambda) = (q, \lambda)$ and $\delta(q, wi) = (p, vo)$ if $\delta(q, w) = (p', v)$ and $\delta(p', i) = (p, o)$. Likewise, we extend μ by $\mu(q, \lambda) = 0$ and $\mu(q, wi) = \mu(q, w) + \mu(p', i)$.

The intuition is that a single token is inserted into the component from an input terminal, and the component ejects it from an output terminal, possibly updating its internal state in the process.

Definition 5. An s-automaton $A = (Q, I, O, \delta, \mu)$ *simulates* another s-automaton $B = (Q', I, O, \delta', \mu')$ from state $q' \in Q'$, if there exists a state $q \in Q$ such that for all words $w \in I^*$ such that $\delta'(q', w)$ is defined, we have $\delta(q, w) = (p, v)$ and $\delta'(q', w) = (p', v)$ for some $v \in O^*$, and $\mu(q, w) = \mu'(q', w)$.

Note that in this definition we do not require the transition functions of A and B to be defined on exactly the same set of input words: A is allowed to process strictly more inputs than B.

Definition 6. If $A = (Q, I, O, \delta, \mu)$ and $B = (Q', I', O', \delta', \mu')$ are two s-automata, the *product* $A \otimes B$ is defined as $(Q \times Q', I \dot\cup I', O \dot\cup O', \delta'', \mu'')$ where $\delta''((q, q'), i) = ((p, q'), o)$ and $\mu''((q, q'), i) = \mu(q, i)$ for $i \in I$ where $\delta(q, i) = (p, o)$; and $\delta''((q, q'), i') = ((q, p'), o')$ and $\mu''((q, q'), i') = \mu'(q', i')$ for $i' \in I'$ where $\delta(q', i') = (p', o')$.

Definition 7. For an s-automaton $A = (Q, I, O, \delta, \mu)$, $i \in I$ and $o \in O$, the *(i, o)-feedback* A^i_o is defined as $(Q, I \setminus \{i\}, O \setminus \{o\}, \delta', \mu')$, where $\delta'(q, i') = (p, o')$ if either $\delta(q, i') = (p, o')$ or we have $\delta(q, i') = (q_0, o)$, $\delta(q_k, i) = (q_{k+1}, o)$ for $k < n$, and $\delta(q_n, i) = (p, o')$, for $i' \in I \setminus \{i\}$ and $o' \in O \setminus \{o\}$. We also define $\mu'(q, i') = \mu(q, i') + \sum_{k \le n} \mu(q_k, i)$.

Thinking of an s-automaton as having input and output terminals for receiving and ejecting tokens, the feedback construction connects the terminals o and i, and $q_0, q_1, \ldots, q_n$ describes the path taken by the token as it loops through the automaton, when the original automaton uses the output terminal o. It may of course enter an infinite loop, in which case the transition is not defined.

These operations allow us to construct new s-automata by combining existing ones into circuits.

Definition 8. Let $\mathcal{A}$ be a set of s-automata and $n \ge 1$. A *timed network of size n* over $\mathcal{A}$ is an s-automaton obtained by taking n copies of automata in $\mathcal{A}$ (with replacement), taking their product, and then performing any number of feedback operations on it (with any choices of i, o).

We define two classes of s-automata that are relevant to our construction. Both of them contain only constant-time automata whose terminals can be used at most once. For this reason, we call them *disposable*.

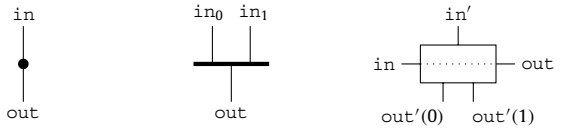


Fig. 3. The trivial merge `triv`, the disposable merge `merge` and the disposable switch `switch`.

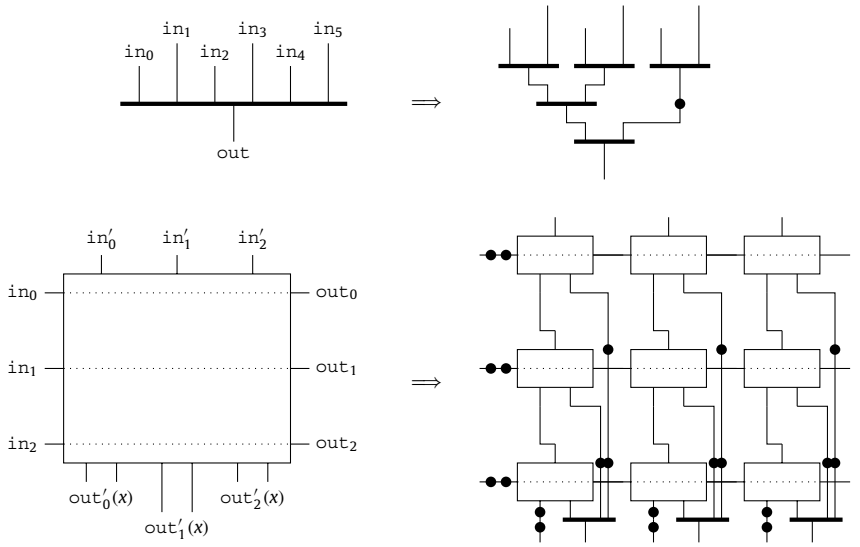


Fig. 4. Constructing the disposable n-merge $\texttt{merge}^n$ (above, with $n = 6$) and the disposable (n, k)-switch $\texttt{switch}^{(n,k)}$ (below, with $n = k = 3$) from primitive components. The trivial merges guarantee that every path through the timed network goes through an equal number of primitive components, all of which have time 1.

Definition 9. For $n \geq 1$, the *disposable n-merge* is the s-automaton

$$\texttt{merge}^n = (\{a, b\}, \{\texttt{in}_0, \texttt{in}_1, \ldots, \texttt{in}_{n-1}\}, \{\texttt{out}\}, \delta, \max(1, n-1))$$

where $\delta(a, \texttt{in}_k) = (b, \texttt{out})$ for $k \in [0, n-1]$. If $n = 1$, we call it the trivial merge, and denote $\texttt{triv} = \texttt{merge}^1$. If $n = 2$, we call it the disposable merge, and denote $\texttt{merge} = \texttt{merge}^2$.

For $n, k \geq 1$, the *disposable (n, k)-switch* is defined as the s-automaton $\texttt{switch}^{(n,k)} = (Q, I, O, \delta, \max(n, k(k-1)-1))$ where

$$Q = \{a_x^N \mid N \subset [0, k-1], x \in \{0, 1\}\},$$
$$I = \{\texttt{in}_0, \ldots, \texttt{in}_{n-1}, \texttt{in}'_0, \ldots, \texttt{in}'_{k-1}\},$$
$$O = \{\texttt{out}_0, \ldots, \texttt{out}_{n-1}, \texttt{out}'_0(0), \texttt{out}'_0(1), \ldots, \texttt{out}'_{k-1}(0), \texttt{out}'_{k-1}(1)\}$$

and $\delta(a_0^{[0,k-1]}, \texttt{in}_j) = (a_1^{[0,k-1]}, \texttt{out}_j)$ for $j \in [0, n-1]$ and $\delta(a_x^N, \texttt{in}'_j) = (a_x^{N \setminus \{j\}}, \texttt{out}'_j(x))$ for $N \subset [0, k-1]$, $j \in N$ and $x \in \{0, 1\}$. If $n = k = 1$, we call it the disposable switch, and denote $\texttt{switch} = \texttt{switch}^{(1,1)}$. When $n = 1$ or $k = 1$, we may also suppress the respective indices from I and O, and in the case of $k = 1$, denote $a_x^{\{0\}} = a_x$ and $a_x^{\emptyset} = b_x$.

The trivial merge, disposable merge and disposable switch are called *primitive components*.

The disposable n-merge simply combines n inputs into one, and the purpose of the trivial merge is to control the times of timed networks in our formalism (i.e. it is a disposable delay component). The purpose of the disposable (n, k)-switch is to store one bit of information to be read later. It has n input terminals, any of which can be used to switch an internal bit from 0 to 1 before the other inputs are used. Each of the k additional input terminals is linked to one of two outputs depending on the bit, and they can be used in any order.

The disposable n-merge can be simulated by $n-1$ disposable 2-merges and $O(n)$ trivial merges, and the disposable (n, k)-switch can be simulated by nk disposable $(1, 1)$-switches, k disposable k-merges and $O(\max(nk, k^2))$ trivial merges. The smallest versions of these s-automata are shown in Fig. 3, and the constructions of the larger versions are shown in Fig. 4.

Lemma 7. *If N is a timed network over* `triv`, `merge` *and* `switch` *of size n, then the time of any transition of N is at most $2n$.*

Proof. Each component of N has time 1, and the token can traverse them at most twice. □

We show that any Boolean circuit can be simulated by a timed network over the merge and switch in a certain sense. This result will be used as a black box later in the construction.

Recall that a *Boolean circuit* is a directed acyclic graph, where the nodes are labeled with operations, and some nodes are labeled as input or output nodes. We refer to in- and out-degrees of nodes as their *fanin* and *fanout*, respectively. A Boolean circuit gives a function in an obvious way, by setting the values at the input nodes, and recursively computing values (based on the operations at the nodes) down the graph (the incoming edges give the inputs, and the output of the operation is copied into all the outgoing edges).

Lemma 8. *Let $k \in \mathbb{N}$, and let $C : \{0,1\}^k \to \{0,1\}^k$ be a Boolean circuit composed of $n \geq k$ gates (unlimited-fanin AND and unary NOT, each with unlimited fanout). Then there exists a constant-time network $A_C = (Q_C, I, O, \delta_C, T)$ over the components* `triv`, `merge` *and* `switch` *with the following properties.*

1. *The size of C is $O(kn^2)$ and its time is $T = O(kn^2)$.*
2. *The set I contains input terminals $\mathtt{in}_i(x)$ and $\mathtt{in}^i$ for each $i \in [0, k-1]$ and $x \in \{0,1\}$, plus one terminal named #.*
3. *The set O contains output terminals $\mathtt{out}_i$ and $\mathtt{out}^i(x)$ for each $i \in [0, k-1]$ and $x \in \{0,1\}$, plus one terminal named \$.*
4. *There exists a state $q_0 \in Q_C$ such that for all $w \in \{0,1\}^k$ we have*

$$\delta_C(q_0, \mathtt{in}_0(w_0)\mathtt{in}_1(w_1)\cdots\mathtt{in}_{k-1}(w_{k-1})\#\mathtt{in}^0\cdots\mathtt{in}^{k-1}) =$$
$$(p, \mathtt{out}_0\mathtt{out}_1\cdots\mathtt{out}_{k-1}\$\mathtt{out}^0(v_0)\mathtt{out}^1(v_1)\cdots\mathtt{out}^{k-1}(v_{k-1}))$$

for some $p \in Q$ and $v = C(w)$.

Proof. First, we replace each AND-gate of C with fanin greater than 2 by a series of binary AND-gates, and each gate with fanout greater than 1 by a gate of the same type with fanout 1 and a collection of *splitters*, which are gates with fanin 1 and fanout 2 that copy their input bit into the two output wires. This causes an at most quadratic blowup in the size of C.

In the construction, each wire W of C corresponds to a switch $\mathtt{switch}_W$ in A_C, whose state is initially a_0. Denote the input wires of C by $W^0, \ldots, W^{k-1}$ and the output wires by $W_0, \ldots, W_{k-1}$. For each $j \in [0, k-1]$, the input terminal $\mathtt{in}_j(1)$ of A_C is connected to the input terminal `in` of $\mathtt{switch}_{W^j}$, the output terminal `out` of $\mathtt{switch}_{W^j}$ is connected to a merge $\mathtt{merge}_{W^j}$. The terminal $\mathtt{in}_j(0)$ is connected to the other input of $\mathtt{merge}_{W^j}$ via a trivial merge, and the output terminal of $\mathtt{merge}_{W^j}$ is connected to $\mathtt{out}_j$. In this way, if terminal $\mathtt{in}_j(x)$ receives a token before $\mathtt{in}_j(1-x)$, then the switch $\mathtt{switch}_{W^j}$ is set to state a_x, and the token travels through the merge $\mathtt{merge}_{W^j}$ and exits via $\mathtt{out}_j$. In particular, $\delta(q_0, \mathtt{in}_0(w_0)\cdots\mathtt{in}_{k-1}(w_{k-1})) = (p', \mathtt{out}_0\cdots\mathtt{out}_{k-1})$ for a state p' where each $\mathtt{switch}_{W^j}$ has internal state b_{w_j} and otherwise agrees with q_0.

Let G be a gate of C. Suppose first that G is a NOT-gate that connects a wire W to another wire V. We connect terminal $\mathtt{out}'(0)$ of $\mathtt{switch}_W$ to terminal `in` of $\mathtt{switch}_V$, and connect terminal `out` of $\mathtt{switch}_V$ to an input terminal of a merge $\mathtt{merge}_G$. We also connect terminal $\mathtt{out}'(1)$ of $\mathtt{switch}_W$ to the other input terminal of $\mathtt{merge}_G$.

Suppose then that G is a splitter that connects a wire W to two wires V and U. We connect terminal $\mathtt{out}'(1)$ of $\mathtt{switch}_W$ to terminal `in` of $\mathtt{switch}_V$, connect terminal `out` of $\mathtt{switch}_V$ to terminal $\in$ of $\mathtt{switch}_U$, and terminal `out` of $\mathtt{switch}_U$ into an input terminal of a merge $\mathtt{merge}_G$. Terminal $\mathtt{out}'(0)$ of $\mathtt{switch}_W$ is connected directly to the other input terminal of $\mathtt{merge}_G$.

Finally, suppose that G is an AND-gate that connects two wires W and V into a wire U. We connect terminal $\mathtt{out}'(0)$ of $\mathtt{switch}_W$ and terminal $\mathtt{out}'(0)$ of $\mathtt{switch}_V$ to two input terminals of a 3-merge $\mathtt{merge}_G$, connect terminal $\mathtt{out}'(1)$ of $\mathtt{switch}_W$ to terminal $\mathtt{in}'$ of $\mathtt{switch}_V$, connect terminal $\mathtt{out}'(1)$ of $\mathtt{switch}_V$ to terminal `in` of $\mathtt{switch}_U$, and connect terminal `out` of $\mathtt{switch}_U$ into the third input terminal of $\mathtt{merge}_G$.

In each case, the input terminal $\mathtt{in}'$ of $\mathtt{switch}_W$ is not connected to anything, and we introduced a merge $\mathtt{merge}_G$ whose output terminal is likewise not yet connected to anything. We denote these terminals by $\mathtt{in}_G$ and $\mathtt{out}_G$, respectively. Finally, we add enough trivial merges to the gadgets that all paths from $\mathtt{in}_G$ to $\mathtt{out}_G$ pass through an equal number of primitive components. See Fig. 5 for a visualization of this construction.

Next, enumerate the gates $G_1, \ldots, G_n$ of C in some order that is consistent with their evaluation (i.e. perform a topological sort of the circuit as a directed acyclic graph). Connect the input terminal # of A_C to $\mathtt{in}_{G_1}$. For each $j = 1, \ldots, n-1$, connect $\mathtt{out}_{G_j}$ to $\mathtt{in}_{G_{j+1}}$, and finally, connect $\mathtt{out}_{G_n}$ to the output terminal \$ of A_C. When we now add a token to the terminal # in the state p' where the values of the input wires have been determined, it travels to the terminal $\mathtt{in}_{G_1}$, sets the values of the switches corresponding to the output wires of G, and exits through $\mathtt{out}_{G_1}$. Then it proceeds to $\mathtt{in}_{G_2}$, repeating this for each gate before exiting through $\mathtt{out}_{G_{k-1}}$ and then \$. By the construction of the gates, each $\mathtt{switch}_W$ in A_C is correctly assigned to the state a_x, where x is the value of the wire W in C, until it evolves to state b_x as the gate that it leads to is evaluated. Furthermore, the number of primitive components that the token passes through does not depend on the states of the switches. In the end, each $\mathtt{switch}_{W_j}$ corresponding to the output wire W_j holds the state $a_{C(w)_j}$.

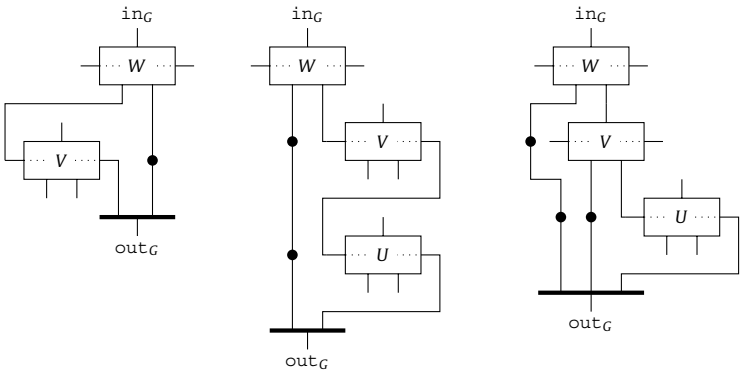


Fig. 5. Implementations of the NOT-gate (left), splitter (middle) and AND-gate (right) as timed networks over the primitive components.

Finally, we connect each terminal $\texttt{in}^j$ of A_C to terminal $\texttt{in}'$ of $\texttt{switch}_{W^j}$, and for each $x \in \{0,1\}$, connect terminal $\texttt{out}'(x)$ of $\texttt{switch}_{W_j}$ to terminal $\texttt{out}^j(x)$ of A_C. We also add a chain of $O(n)$ new trivial merges right after each input terminal of the network other than # in order to balance its time. It is now easy to see that the claim holds. □

7. Simulation of timed networks

In this section, we show how timed networks can be simulated by the Turing machine M. We use such simulations in the proof of physical universality of M, and this property is very time-sensitive, in the sense that we must be able to precisely control the number of steps that M takes to construct the output pattern. Recall that $B_{m,n}$ is the outer border of $[0, m-1] \times [0, n-1]$.

Definition 10. Let $A = (S, I, O, \delta, \mu)$ be an s-automaton and $s_0 \in S$ one of its states, let $m, n > 0$, let $F : \mathbb{N} \to \mathbb{N}$ be a function and denote $D = [0, m-1] \times [0, n-1]$. An (m, n, F)-*simulation* of A by M from (initial state) s_0 is a pair (P, d) with $P \in \Sigma^D$ a pattern and $d : I \cup O \to B_{m,n}$ an injective function with the following properties.

- $d(e)$ is on the leftmost column of $B_{m,n}$ for $e \in I$, and on the rightmost column for $e \in O$.
- For each $R \in \Sigma^D$ and $e \in I \cup O$, let $x(R, e) = (\to [d(e)], R \sqcup 0^{\mathbb{Z}^2 \setminus D})$. Then for each word $i = i_0 \cdots i_{k-1} \in I^*$ such that $\delta(s, i) = (t, o)$ is defined, we have a sequence of patterns $P = P_0, P_1, \ldots P_k$ in Σ^D with $\rho_D^M(x(P_j, i_j)) = x(P_{j+1}, o_j)$ for each $j < k$ and $\sum_{j=0}^{k-1} \epsilon_D^M(x(P_j, i_j)) = F(\mu(s_0, i))$.

If F is multiplication by a constant c, as in $F(n) = cn$, we may replace it by c.

The intuition behind Definition 10 is that we have a single pattern corresponding to a state s_0 of A, and fixed positions around these patterns that correspond to input and output terminals. When the head of M enters the domain of the pattern through an input position, then it will eventually leave the domain through an exit position and manipulate the pattern into one that corresponds to another state, in a way that is consistent with the transition function δ. The correspondence between patterns and states need not be exact, as long as the systems are behaviorally equivalent. Furthermore, the number of steps required to simulate a transition of A is determined by its time given by μ, through the function F. In our construction, F will be a polynomial.

Lemma 9. *There exist simulations by M with equal time function $n \mapsto cn$ of the trivial merge, the disposable merge, and the disposable switch from states a, a and a_0 respectively. In these simulations, no two terminals have adjacent positions on the border of the domain.*

Proof. See Fig. 6 for a $(6, 8, 11)$-simulation of `merge` and an $(8, 13, 22)$-simulation of `switch`. The figures depict states that correspond to a and a_0, respectively, and the other states are obtained by inserting the head at one of the positions corresponding to the input terminals and applying the escape map ρ_D^M until no more inputs can be processed. It is easy to verify that these processes exactly simulate the behavior of `merge` and `switch` on valid input sequences.

The trivial merge can be simulated by a single cell holding the symbol 0. Finally, we can pad the simulations with extra columns of 0-cells to guarantee that the time factor c is the same in all of them. □

Our goal is to simulate any constant-time network over these components.

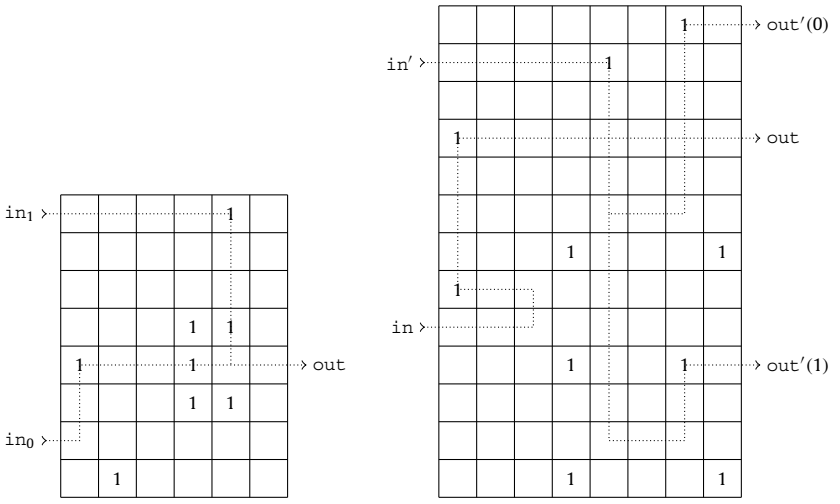


Fig. 6. Simulations of `merge` (left) and `switch` (right) with the paths of the head of M highlighted.

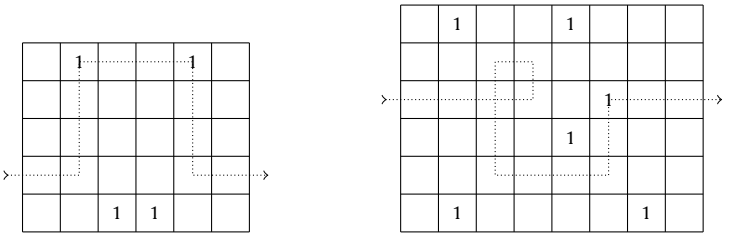


Fig. 7. Gadgets that implement a delay of 9 steps (left) and 11 steps (right) in a west-to-east wire. Note that traversing one step to the north takes two time steps.

Lemma 10. *Let $N = (S, I, O, \delta, \mu)$ be a timed network of size n over the primitive components. Let $s_0 \in S$ be the state where each component is in state a or a_0. Then there are numbers $m = O(n^3)$ and $k = O(n^2)$, a function $F(i) = ci + d$ with $c, d = O(n^2)$, and an (m, k, F)-simulation of N by M from s_0.*

Proof. We construct the simulation by placing the primitive components on a vertical column with $O(n)$ empty cells between them. We then construct the wires so that the head of M traverses each of them in an equal number of steps.

In this construction, a wire is represented by an empty region of 0-symbols, and occasional 1-symbols that cause the head of M to turn right or left. More explicitly, if M encounters a single 1-symbol directly on its path, then it will make a right turn as it moves on that symbol. If M encounters a 1-symbol on the right hand side of its path, then it will make a left turn when that symbol can be moved onto the cell occupied by the head. Since the 1-symbols are moved when the head traverses a wire, it cannot be safely traversed again, but this is not an issue since the primitive components of Definition 9, and thus all timed networks over them, have the property that each input and output terminal can be used at most once. A crossing of two wires is trivial to implement.

We now introduce a way for adding arbitrary delays in horizontal wires. Fig. 7 shows two gadgets, one for delaying the head of M for 9 steps and one for 11 steps. By inserting copies of these gadgets into a wire of length n, we can implement any delay which is $O(n)$ and larger than a fixed constant.

Label the components of N as $A_1, \dots, A_n$ and the wires as $W_1, \dots, W_\ell$. We choose $h = O(n)$, $k = h(n+1)$ and $m = h(\ell+1)$, and place a pattern P_i that simulates A_i in an all-0 configuration so that its southwest corner is at $(m - C, hi)$ for a constant C. For each wire W_i starting from an output terminal of some P_j, we extend a simulated wire to the east and make two left turns that redirect it to the west so that it passes above P_j. This simulated wire is extended to the x-coordinate hi. Within the rectangle spanned by the coordinates $(hi, 0)$ and $(h(i+1) - 1, k - 1)$, we have enough space to implement an arbitrary delay of $O(n^2)$ steps using turns and the two delay components of Fig. 7, so that each wire is traversed in exactly $e = O(n^2)$ steps. The simulated wire is then extended east, either to an input terminal of some other $P_{j'}$ or the east border of the simulation region. If W_i originates from an input terminal of N, we use a similar construction, except that the simulated wire originates from the west border of the simulation region. See Fig. 8 for a diagrammatic representation of the construction.

It is clear from the construction that the resulting $m \times k$-pattern P realizes a simulation of N by M from s_0. It remains to be shown that it is an (m, k, F)-simulation with $F(x) = e(x+1) + cx$, with e being the number of steps in which M traverses any wire in the simulation and c the common time factor of the simulated primitive components given by Lemma 9. But

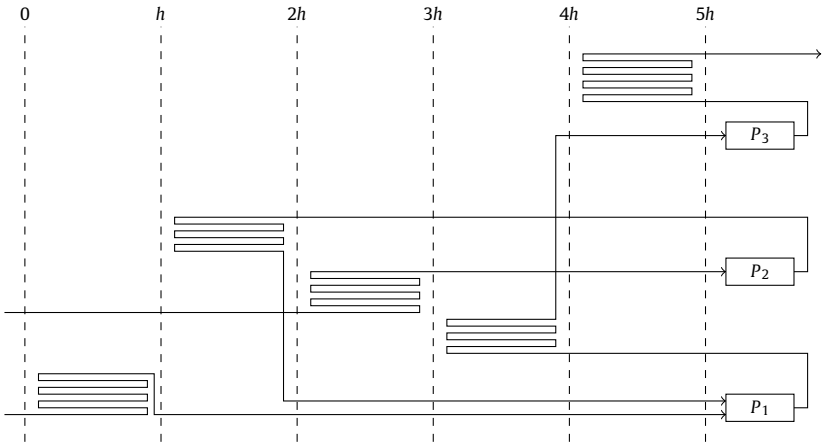


Fig. 8. A diagram of a simulation of a timed network by M.

for any $i \in I^*$, the time $\mu(s_0, i)$ is exactly the number of primitive components of N that the token travels through in the computation of $\delta(s_0, i)$, or equivalently one less than the number of wires it travels through. Hence the number of steps taken by M in the simulation of $\delta(s_0, i)$ is $e(\mu(s_0, i) + 1) + c\mu(s_0, i)$, as claimed. □

8. Border processes

In our construction, the head of M repeatedly enters and exits the rectangular input region in order to extract information from it and manipulate its contents. We now introduce tools that allow us to consider the border of a rectangle as an interface between its inside and outside and consider the two parts separately.

We first define what happens on the outside as an abstract process that satisfies certain properties, and then construct an implementation for its behavior as a concrete configuration of M. We show that the behavior of the machine outside a finite rectangle can be arbitrary, in the sense that we can have the sequence of entrance positions follow an arbitrary function f, depending on exit positions.

Definition 11. Denote $D = [0, m-1] \times [0, n-1]$. A *border process* of size $m \times n$ and depth k is a pair $(\mathcal{P}, f)$ where

- $\mathcal{P}$ is a set of pairs $(\vec{b}, P)$, where $\vec{b} \in B_{m,n}$ is a position on the outer border of D and $P \in \mathcal{P}^0_{Q,\Sigma}(D)$ is a headless pattern where all 1s are contained in $[k, m-1-k] \times [k, n-1-k]$, and
- $f : B^{\leq k}_{m,n} \to B_{m,n}$ is any function.

For $\vec{b}_1, \dots, \vec{b}_i \in B_{m,n}$, we denote by $f[\vec{b}_1, \dots, \vec{b}_i] = \vec{c}_1, \dots, \vec{c}_i$ the sequence of vectors defined by $\vec{c}_j = f(\vec{b}_1, \dots, \vec{b}_j)$ for each $j \leq i$.

The border process represents the head of the Turing machine M entering and exiting an $m \times n$ region exactly k times, with the location of each entrance being a function of the locations of the previous exits. (For a general machine one should also record the states in which we exit and enter, but our machine M effectively has a unique state for each direction, so this is not necessary.)

For a pair $(\vec{b}, P) \in \mathcal{P}$, the vector $\vec{b}$ indicates a "first exit" that happened just before the start of the border process, and P gives the initial contents of the region. The interpretation of $f(\vec{b}_1, \dots, \vec{b}_i) = \vec{c}$ is that if the head has exited the region at successive coordinates $\vec{b}_1, \dots, \vec{b}_i$, then we send it in though $\vec{c}$ on the next round.

Consider a border process $(\mathcal{P}, f)$ of size $m \times n$ and depth k. Denote $D = [0, m-1] \times [0, n-1]$. For any pair $(\vec{b}_1, P) \in \mathcal{P}$, we define $f(\vec{b}_1, P) \in B_{m,n} \times \mathcal{P}^0_{Q,\Sigma}(D)$ as follows. Denote $\vec{c}_1 = f(\vec{b}_1)$, and let $y_1 = (q[\vec{c}_1], P \sqcup 0^{\mathbb{Z}^2 \setminus D} \in \Sigma^{\mathbb{Z}^2})$ be the configuration where the complement of D contains only 0-symbols and $q \in \bar{Q}$ is such that the head faces toward the region D at position $\vec{c}_1$. As the configuration y_1 evolves under M, by Lemma 6, the head eventually leaves the rectangle D, and the 1s stay contained in the region $[k-1, m-1-(k-1)] \times [k-1, n-1-(k-1)]$. Let $x_1 = \rho^M_D(x)$ be the configuration right after the exit, let $\vec{b}_2 \in B_{m,n}$ be the location of the head, and let $\vec{c}_2 = f(\vec{b}_1, \vec{b}_2)$. Let y_2 be the configuration obtained from x_2 by moving the head onto $\vec{c}_2$, facing toward the region D.

In general, we keep defining configurations y_i where the head faces toward the region D and the 1s are contained in $[k-i, m-1-(k-i)] \times [k-i, n-1-(k-i)]$, let it evolve into a configuration $\rho^M_D(y_i) = x_{i+1}$ where the head has just left D onto a coordinate $\vec{b}_{i+1} \in B_{m,n}$, and construct y_{i+1} from x_{i+1} by moving the head to $\vec{c}_{i+1} = f(\vec{b}_1, \dots, \vec{b}_{i+1})$ facing toward D. The process stops at $i = k$, and we define $f(\vec{b}_1, P) = (\vec{c}_k, y_k|_D)$.

Definition 12. A *concrete realization* of a border process $(\mathcal{P}, f)$ of size m, n and depth k is a partial configuration $y \in \Sigma^{\mathbb{Z}^2 \setminus D}$ with a 'hole' of shape $D = [0, m-1] \times [0, n-1]$ with the following property. Take any pair $(\vec{b}_1, P) \in \mathcal{P}$, and consider the configuration $x = (q[\vec{b}_1], y \sqcup P)$ where $q \in Q$ points away from D (in the north direction, we use the starred state). Let $s_1 < t_2 < s_2 < t_3 < s_3 < \cdots$ be the (potentially infinite) sequence of time steps at which the head enters and leaves the region D in the evolution of x by M, and let $\vec{c}_1, \vec{b}_2, \vec{c}_2, \vec{b}_3, \vec{c}_3, \ldots$ be the elements of $B_{m,n}$ at which this happens. Then we have $f[\vec{b}_1, \ldots, \vec{b}_k] = \vec{c}_1, \ldots, \vec{c}_k$.

The intuition is that the partial configuration y implements the process of moving the head to the next coordinate of $B_{m,n}$ given by f after it has left the region $[0, m-1] \times [0, n-1]$.

We also formalize a way of representing a border process as an s-automaton. Our goal is to design border processes that can be represented as small constant-time networks, which in turn can be simulated by M in a small space due to Lemma 10.

Definition 13. An *abstract realization* of a border process $(\mathcal{P}, f)$ of size m, n and depth k consists of a constant-time network $A_f = (Q_f, I_f, O_f, \delta_f, c_f)$ over the primitive components with the following properties.

- $I_f = \{\mathtt{in}_j(\vec{b}) \mid 1 \le j \le k, \vec{b} \in B_{m,n}\}$
- $O_f = \{\mathtt{out}_j(\vec{b}) \mid 1 \le j \le k, \vec{b} \in B_{m,n}\}$
- There exists an initial state $q_0 \in Q_f$ such that for all border coordinates $\vec{b}_1, \ldots, \vec{b}_k \in B_{m,n}$ we have

$$\delta_f(q_0, \mathtt{in}_1(\vec{b}_1) \cdots \mathtt{in}_k(\vec{b}_k)) = (p, \mathtt{out}_1(\vec{c}_1) \cdots \mathtt{out}_{k-1}(\vec{c}_{k-1}) \mathtt{out}_k(\vec{c}_k))$$

  for some state $p \in Q_f$ with $f[\vec{b}_1, \ldots, \vec{b}_k] = \vec{c}_1, \ldots, \vec{c}_k$.

The *abstract complexity* of $(\mathcal{P}, f)$ is the size of its smallest abstract realization.

We now show that every border process $(\mathcal{P}, f)$ has a concrete realization whose size is polynomial in its abstract complexity. Recall from Lemma 10 that the machine M can simulate an arbitrary timed network – in particular an abstract realization A_f of $(\mathcal{P}, f)$ – using a gadget of polynomial size. We construct a system of gadgets that acts as an 'interface' between the rectangular region that f acts on and the gadget that simulates A_f, guiding the head of M to the correct terminals and border cells.

Definition 14. Let $a, n \in \mathbb{N}$. A *west* (a, n)*-catcher* is the pattern of shape $[0, 2a+9] \times [-a-3, n+1]$ with a 1 at $(0, -a-2)$, $(3, -a-3)$, $(3, -a-2)$, $(3, n+1)$, $(2a+6, -a-3)$, $(2a+6, n+1)$ and $(2a+8, -a)$, and a 0 in every other cell. The pattern is divided into three parts: the middle part consists of rows $0, 1, \ldots, n-1$, and the top and bottom parts are formed by the remaining rows. By rotating the pattern by 90, 180 and 270 degrees, we obtain the *north, east and south* (a, n)*-catchers*.

The catcher is initially *inactive*, and can be *activated* by sending the head into the pattern from its south border on the westmost column. The head travels on a spiraling path, pulling four 1s closer to the center, and exits from the eastmost column of the south border. Once the catcher is activated, we can use it to intercept the head of M if it arrives from the east on row $n - a$. See Fig. 9 for a visualization.

Definition 15. Let $n \in \mathbb{N}$. A *west catcher array of width* n is a pattern consisting of one west (a, n)-catcher for each $a = 1, 2, \ldots, n$ positioned side by side in this order. North, east and south arrays are defined as rotated versions of these patterns.

A *catcher system of width* n consists of west, north, east and south catcher arrays of width n positioned so that their middle parts align with the square $[0, n-1]^2$, and each array lies in the respective direction from this square.

We say that a catcher array or catcher system is inactive or activated if each individual catcher in it has this status. Once a west catcher array is activated, it can intercept the head of M as it arrives from the east on any of the rows on its middle part. The (a, n)-catcher is responsible for row $n - a$: if the head arrives on row $n - a$, it passes through the (n, n)-catcher, then the $(n-1, n)$-catcher, all the way up to the (a, n)-catcher which finally intercepts it and sends it north. In other words, the eastmost catcher (i.e. the one closest to $[0, n-1]^2$) is responsible for row 0, the next one for row 1, and so on, with the westmost catcher being responsible for row $n - 1$. The order is important, since it prevents the head from interacting with a catcher that is not responsible for its row. On the other hand, if the array is inactive and the head enters it from the east or west along its middle part, the array does not interact with it, letting it pass directly through. The south, east and north catcher arrays behave analogously, so an activated catcher system intercepts the head as it leaves the square $[0, n-1]^2$ from any border coordinate, while an inactive catcher system does not interact with it.

In our construction, the idea is to have a sequence of nested catcher systems which are activated one by one, starting with the outermost system. The activated system intercepts the head as it exits $[0, n-1]^2$, so that we have a record of the

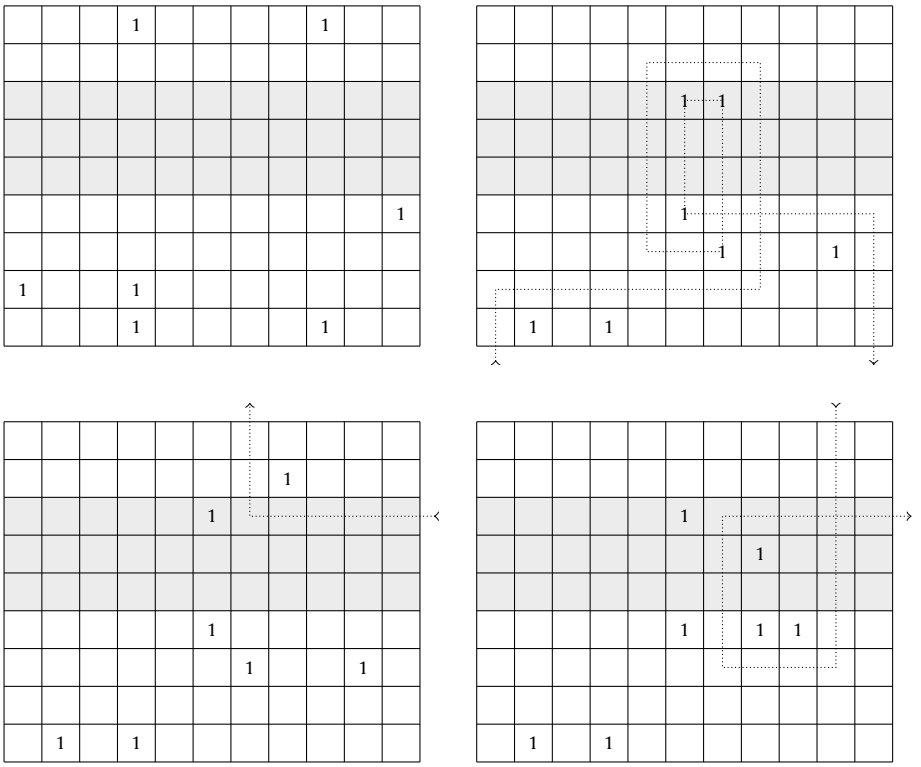


Fig. 9. Top left: an inactive west $(1, 3)$-catcher. Top right: activating the catcher. Bottom left: intercepting the head on row 2 and redirecting it to the north. Bottom right: additional processing performed during partial activation of a catcher array. The middle part of the catcher, corresponding to rows 0, 1 and 2, is shaded in each figure.

exit coordinate. Then the head activates the next system to prepare it for another exit. However, this scheme does not give us a way to actually send the head into $[0, n-1]^2$ as required by a border process, so we modify it slightly.

Suppose we want to send the head into $[0, n-1]^2$ from the west on row $i \in [0, n-1]$ and intercept it when it later exits the square. To achieve this, we will activate the south, east and north catcher arrays of the catcher system and *partially activate* the west catcher array: For each $a = 1, \dots, n$ except for $i+1$, we activate the (a, n)-catcher of the array. The west $(i+1, n)$-catcher, which is closer to $[0, n-1]^2$ than the (i, n)-catcher of the same array, is left inactive. Then we send the head in from the north edge of the activated (i, n)-catcher as in the bottom right part of Fig. 9, so that it is guided to the west on row i and eventually enters $[0, n-1]^2$ on this row. Note that since the west $(i+1, n)$-catcher was left inactive, the head will reach $[0, n-1]^2$ unobstructed. The (i, n)-catcher is left with two 1s that can intercept the head either on row $n-i$ or $n-(i+1)$. The partial activation of south, east and north catcher arrays is defined similarly.

In addition to the catchers, we need some wiring to guide the head during the activation of the catcher arrays, and to feed the head into our computational gadget after it has been intercepted.

Lemma 11. *Let $(\mathcal{P}, f)$ be a border process of size n, depth k and abstract complexity C. Then $(\mathcal{P}, f)$ has a concrete realization $y \in \Sigma^{\mathbb{Z}^2 \setminus [0,n-1]^2}$ in which all 1s are contained in $[-p, p]^2$ for some $p = \mathrm{poly}(n, k, C)$.*

Proof. By Lemma 10, there exist numbers $m = O(C^3)$ and $m' = O(C^2)$, a function $F(i) = ci + d$ with $c, d = O(C^2)$, and an (m, m', F)-simulation of A_f by M. We denote by P_f the $m \times m'$-pattern realizing this simulation, and place it in a partial configuration $y \in \Sigma^{\mathbb{Z}^2 \setminus [0,n-1]^2}$ at an arbitrary location that is sufficiently far away from the origin.

We place around the square $[0, n-1]^2$ one activated catcher system C_0 of width n called the *initial catcher system*. For each $j = 2, \dots, k$ and $\vec{b} \in B_n$, we also place one inactive catcher system $C_{j,\vec{b}}$ of width n, ordered in such a way that C_0 is the outermost system and the systems $C_{j,\vec{b}}$ are placed successively closer to the origin in ascending order of j. This arrangement (with three catcher systems) is illustrated in Fig. 10.

We add some wires to y that connect the catcher systems to the pattern P_f. Namely, for each $\vec{b} \in B_n$, from the position of the initial catcher array C_0 where the head of M is redirected if it exits $[0, n-1]^2$ via $\vec{b}$, we add 1-cells to y that redirect it to the input terminal $\mathtt{in}_1(\vec{b})$ of P_f. For each output terminal $\mathtt{out}_j(\vec{c})$ of P_f with $2 \le j \le k$, we add 1-cells that redirect the head to the inactive catcher system $C_{j,\vec{c}}$, then guide it to partially activate the system so that the head can be sent into $[0, n-1]^2$ through the coordinate $\vec{c}$, and finally redirect it into the system to be sent in through that coordinate. If $j < k$,

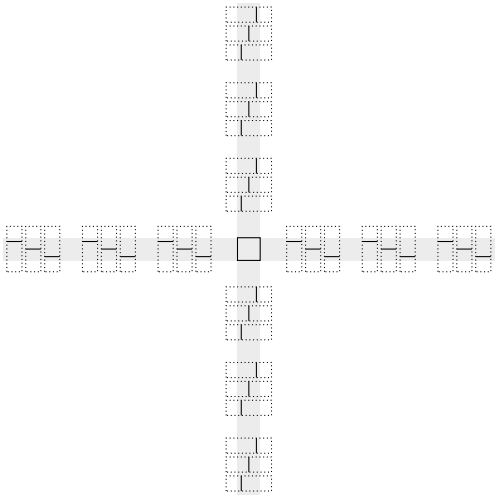


Fig. 10. The collection of catcher systems constructed in the proof of Lemma 11, not drawn to scale. The solid lines mark the row or column each catcher is responsible for. The outermost catcher system is C_0.

then for each $\vec{b} \in B_n$ we also add 1-cells that, once the catcher system $C_{j,\vec{c}}$ is activated and catches the head as it exits $[0, n-1]^2$ via $\vec{b}$, redirect it to the input terminal $\mathtt{in}_j(\vec{b})$ of P_f.

We claim that y is a concrete realization of $(\mathcal{P}, f)$. Take any $(\vec{b}_1, P) \in \mathcal{P}$ and consider the configuration $x = (q[\vec{b}_1], P \sqcup y)$ where the head faces away from $[0, n-1]^2$. The activated catcher system C_0 catches the head as it travels away from $[0, n-1]^2$ and redirects it to terminal $\mathtt{in}_1(\vec{b}_1)$ of P_f. For $j = 2, \dots, k-1$, the head exits P_f through some output terminal $\mathtt{out}_j(\vec{c}_j)$, partially activates the system $C_{j,\vec{c}_j}$, enters its partially activated catcher and is sent into $[0, n-1]^2$ through the border coordinate $\vec{c}_j$. If $j < k$, the head later exits $[0, n-1]^2$ through some border coordinate $\vec{b}_j$, is caught by the catcher of the same system $C_{j,\vec{c}_j}$ that is responsible for that coordinate, and is redirected to input terminal $\mathtt{in}_j(\vec{b}_j)$ of P_f.

It remains to bound the number p. The pattern P_f has size $O(C^3) \times O(C^2)$, and we have a total of $O(kn^2)$ catcher systems and "wires" guiding the head of M from P_f to these systems and back. All this can be fit into a square pattern with side length $p = O(C^3 + kn^4)$. □

9. Implementation of transformations by border processes

In this section, we prove that a border process can implement an arbitrary transformation on the set of local patterns of shape $m \times m$, if they are embedded inside a larger $(2n+m) \times (2n+m)$-pattern in a suitable way. For the statement of the lemma it is useful say that a pattern P whose domain is contained in $[0, 2n+m-1]^2$ has *unobstructed center* if $P_{\vec{v}} = 0$ for all $\vec{v} \in ([0, 2n+m-1] \times [n-3, n+m+2] \cup [n-3, n+m+2] \times [0, 2n+m-1]) \setminus [n, n+m-1]^2$. We also define a correspondence between central border vectors of $[0, 2n+m-1]^2$ and $[0, m-1]^2$: for $\vec{b} = (i, j) \in B_m$, define $s(\vec{b}) \in B_{2n+m}$ by

$$s(\vec{b}) = \begin{cases} (-1, j+n) & \text{if } i = 0, \\ (2n+m, j+n) & \text{if } i = m-1, \\ (i+n, -1) & \text{if } j = 0, \\ (i+n, 2n+m) & \text{if } j = m-1. \end{cases}$$

Lemma 12. *Let $m \in \mathbb{N}$ and $A \subset B_m \times \Sigma^{[0,m-1]^2}$ a set of coordinate-pattern pairs. Let $g : A \to B_m \times \Sigma^{[0,m-1]^2}$ be a function with circuit complexity C. Then there exists $n = \mathrm{poly}(m)$, a pattern $P \in \Sigma^{[0,2n+m-1]^2 \setminus [n,n+m-1]^2}$ with unobstructed center, and a border process $(\mathcal{P}, f)$ of size $2n+m$, depth $\mathrm{poly}(m)$ and abstract complexity $\mathrm{poly}(m, C)$ with the following properties.*

- *For each $(\vec{b}, R) \in A$, we have $(s(\vec{b}), \tau_{(n,n)}(R) \sqcup P) \in \mathcal{P}$.*
- *Denoting $g(\vec{b}, R) = (\vec{c}, S)$, we have $f(s(\vec{b}), \tau_{(n,n)}(R) \sqcup P) = (s(\vec{c}), Q)$, where Q has unobstructed center and $Q|_{[n,n+m-1]^2} = \tau_{(n,n)}(S)$.*

Note that we essentially just state that f simulates g in the central area $[n, n+m-1]^2$. The correspondence is visualized in Fig. 11. Note that the entire pattern Q, not just its center, may depend on R.

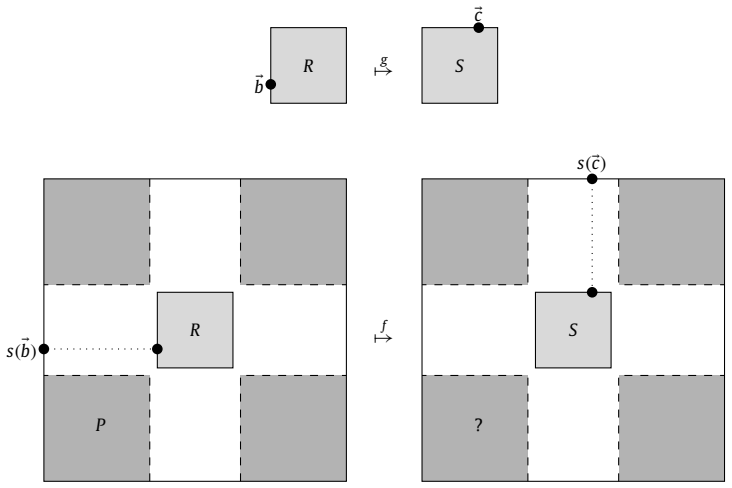


Fig. 11. A diagram of Lemma 12, not drawn to scale. The inner and outer squares have size $m \times m$ and $(2n+m) \times (2n+m)$, respectively.

Proof. We explicitly construct the pattern P and border process $(\mathcal{P}, f)$ that have the required properties, and then estimate its abstract complexity. The construction proceeds in four stages, which we call the *probe stage*, *transport stage*, *sculpt stage* and *cleanup stage*, each of which is further divided into substages. Each (sub)stage consists of a number of *rounds*, and a round corresponds to the head leaving the square $[0, 2n+m-1]^2$ and returning on some border coordinate. In order to reduce the abstract complexity of the process and simplify timing issues later on, each (sub)stage will consist of a number of rounds that is independent of the choice of $(\vec{b}, R) \in A$.

In the probe stage, the head is repeatedly sent in from the west to probe the x-coordinate of a single 1 occurring on a specific horizontal row of the region $D = [n, n+m-1]^2$, after which it is moved far away from the region by repeatedly colliding the head with it. After the probe stage, the region D contains no more 1s, and the transport stage consists of moving a large solid block of 1s onto it. This is again achieved by repeatedly colliding the head with a single 1, moving it toward the target by one diagonal step. In the sculpt stage, some of the 1s in the solid block are 'carved out' and moved out of the region, in a process that mirrors the probe stage. In this way, the desired output pattern can be constructed on the region D at the completion of the border process.

We now give more details on these stages, beginning with the probe stage. For this, we define the pattern P to be filled with 0s for now; we will add some 1s to it later. The probe stage is divided into several substages, one for each $i = 0, \dots, m-1$, and the purpose of substage i is to determine the positions of 1s on row $n+i$. On substage i, we place the head on the west border of P at y-coordinate $n+i$.

If row $n+i$ contains at least one 1, then the head collides with the westmost one and turns right, exiting through the south border of P at some coordinate $\vec{v}$. If the 1 was at coordinate $(a, n+i)$, then $\vec{v} = (a, 0)$, and the border process can use this information on later rounds. We then choose a large number $j = O(m^4)$, and for each $k = 1, \dots, j$, place the head on the west border of P at y-coordinate $n+i-k$. This causes the newly discovered 1-symbol to travel j steps to the southwest, where it will not interfere with the rest of the process. We informally say that M transports it to the *landfill*. This process is visualized in Fig. 12.

If there were no 1s on row $n+i$, then the head exits through the east border of P. In this case, the next j rounds of the process are "do-nothing" rounds, in which the head simply travels through the southmost row of P.

We continue in this manner, repeatedly probing for a 1-symbol on the lowest nonempty row $n+i$ and moving it out of the way, for a total of m times before proceeding to the next row. Whenever the head encounters a 1 at some coordinate $(a, n+i)$, it exits P at coordinate $(a, 0)$. If n is large enough ($n = \Omega(m^4)$ suffices for this), there is enough room to transport each 1 to the landfill following the first collision with it, after which it does not affect the remaining rounds. Once the top row of D has been completely probed, the head exits through the east border of P and we have complete information of the contents of the pattern R. This concludes the probe stage.

We now describe the transport stage. At this point, the region D contains only 0s, and there are also 0s to the east, north, west and south of it. There is a collection of 1s to the southwest of D (the landfill), which are the remnants of the probe stage. We introduce new 1s into the pattern P, to the northeast of the region D (these 1s of course should be there from the beginning of the process). For each $j \in [0, 5m^2+1]$ and $i \in [0, 10m^2+1-j]$, we add a 1 at the coordinate $(n+3+i, n+3+j)$. We call this pattern of 1s the *clay*, as it will be used in the sculpt stage. Of course, the clay is also present during the escape and probe stages, but does not affect them, as its southmost row lies four steps to the north of the row $n+m-1$, which is the northmost row of D. Note that the bottom row of 1s in this pattern is the longest one, and the lengths of consecutive rows differ by 1.

We describe a way to translate the entire clay a single step to the southwest. This is done one row at a time, starting from the bottom row, and each row is translated one cell at a time, from west to east. Suppose the southwest corner of the

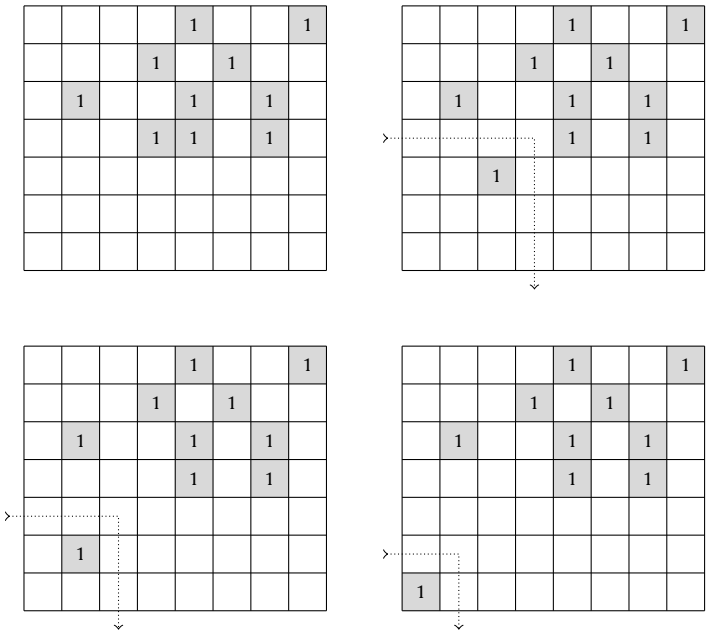


Fig. 12. The head of M extracts a 1 from the bottom row of the region R.

clay is at $(a, b) \in \mathbb{Z}^2$. We send the head of M in from the south border of P on column $a-1$. The head travels north until it is diagonally adjacent to the southwest corner of the clay, where it makes a left turn to the west and moves the 1 at the corner from (a, b) to $(a-1, b-1)$, and then travels to the west border of P. Next, the head is placed on the column a, one step to the east. It again travels north, moves a 1 from $(a+1, b)$ to $(a, b-1)$ by a left turn, and travels to the west border of P. We continue in this manner, placing the head on column $a+i-1$ for each $i \in [0, 5m^2+1]$ and moving a single 1 on the row b. We then repeat this procedure for the other rows: for each j from 0 to $5m^2+1$, and for each i from 0 to $5m^2+1+j$, we place the head on the south border of P on column $a-1+i$, from where it travels north and moves a 1 from $(a+i, b+j)$ to $(a+i-1, b+j-1)$ by a left turn. The distinct widths of the rows guarantee that M does not make any additional turns when moving the eastmost 1 of each row. When this operation is complete, we have moved the entire clay one step to the southeast.

We repeat the operation to move the entire clay for a total of $5m^2+m+3$ steps, after which we have a 1 at coordinate $(n+i, n+j)$ for all $j \in [m-5m^2, m-1]$ and $i \in [m-5m^2, m-1+j]$. This concludes the transport stage.

We describe the sculpt stage. Intuitively, it resembles the probe stage, but in reverse and with the roles of 0 and 1 inverted. Our goal is to remove some 1s from the clay so that the output pattern $g(R)$ is formed on the region D. At this point, the process f has full information of $g(R)$. Let $E_0 \subset [0, m-1]^2$ be the set of coordinates $\vec{v}$ with $g(R)_{\vec{v}} = 0$. The idea is that for each $\vec{v} = (i, j) \in [0, m-1]^2$ in decreasing order of $i+j$, we do the following:

1. Remove rows from the south border of the clay or columns from its west border until its southwest corner is on the same diagonal as $\vec{v} + (n, n)$.
2. If $\vec{v} \in E_0$, move each 1 on the diagonal path from $(n, n) + \vec{v}$ to the southwest corner of the clay one step to the southwest. Then move the 1 that was previously at the corner of the clay out of the way.

We describe the sculpt process in more detail. The process is divided into m^2 substages, one for each $(i, j) \in [0, m-1]^2$, and substage (i, j) consists of $O(m^6)$ rounds. It begins with the transportation of some number of southmost rows or westmost columns from the clay to the landfill. As in the probe stage, this is done by moving each 1 a distance of $O(m^4)$, taking $O(m^4)$ rounds; this is enough as there are a total of $O(m^4)$ cells in the clay. Moving each 1 of a row or column of length $O(m^2)$ for a distance of $O(m^4)$ steps takes $O(m^6)$ rounds. If we choose the order of $(i, j) \in [0, m-1]^2$ to be increasing in j when $j+i$ is even and decreasing in j otherwise, on average it suffices to remove at most two rows or columns from the clay on each substage.

Next, we check whether $(i, j) \in E_0$. If this is the case, we send the head from the south, then east and east again, as illustrated in Fig. 13, in order to move a 1 out of the southwest corner of the clay. For the next $O(m^2)$ rounds, we send the head from the west border of the clay in order to move this 0 one step to the northeast within the clay, until it reaches $(n+i, n+j)$. In the case $(i, j) \notin E_0$, the above process is replaced by the head repeatedly entering P on some row that contains only 0s. All in all, the sculpt stage takes $O(m^8)$ rounds. The sculpting process is illustrated in Fig. 14.

In the cleanup stage, all remaining 1s outside the square D are transported out of the region $[0, 2n+m-1] \times [n-3, n+m+2] \cup [n-3, n+m+2] \times [0, 2n+m-1]$ in order to produce a pattern with unobstructed center. This takes $O(m^8)$

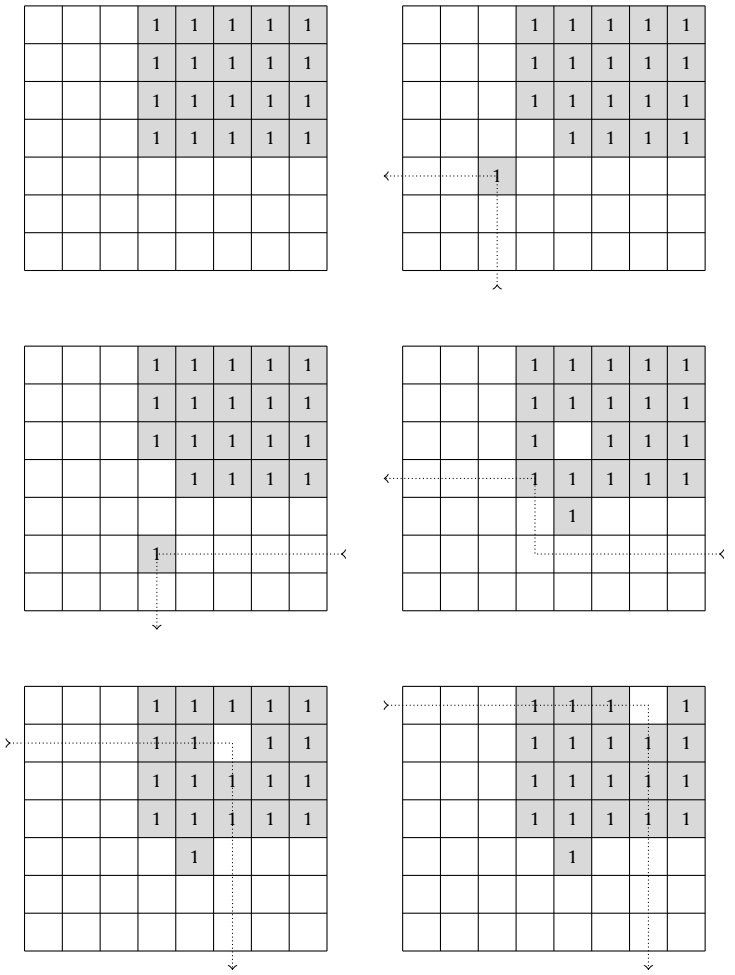


Fig. 13. The head of M extracts a 1 from the corner of the clay, creating a movable "hole".

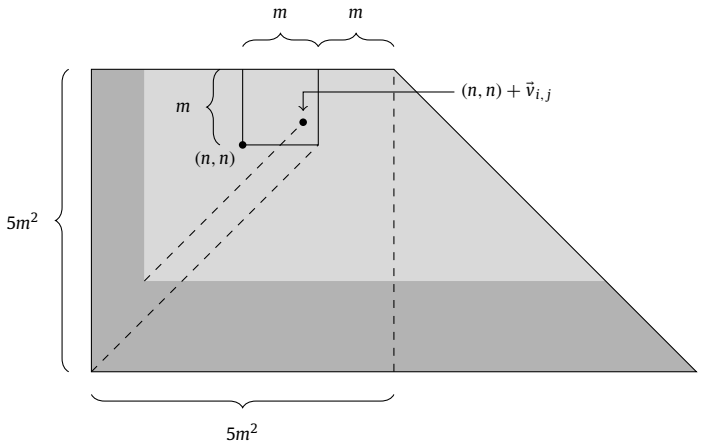


Fig. 14. The sculpting process, not drawn to scale. The dark gray part of the clay is removed before the coordinate $\vec{v}_{i,j}$ is considered.

rounds and is implemented similarly to the probe stage, moving each 1 diagonally for $O(m^4)$ steps (we can assume that the remaining part of the clay is still surrounded by an annulus of 0 of thickness $O(m^4)$).

After these four stages, the output region D contains a translated version of the pattern $g(R)$. A choice of $n = O(m^4)$ is enough to guarantee that the construction can be carried out, since then we have enough room to transport any 1s out of the way during the probe, sculpt and cleanup stages. We can guarantee that the total number of rounds is independent of

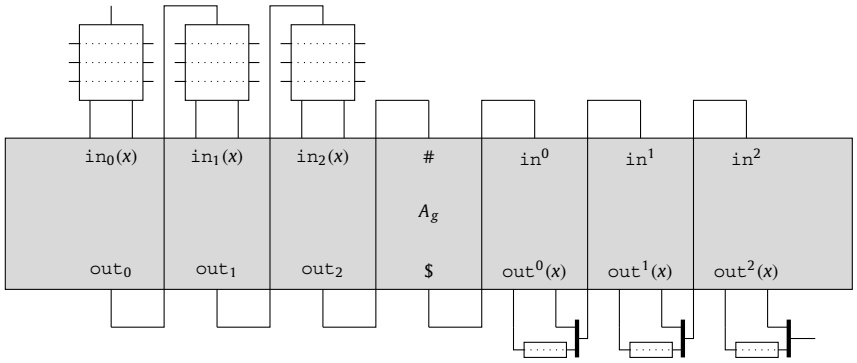


Fig. 15. Rewiring the timed network A_g to transform stored data into stored data.

$(\vec{b}, R)$ by adding some number of do-nothing rounds in which the head is sent through the bottom-most row of P, which we always keep empty. The depth of f is then $O(m^8)$.

Finally, we estimate the abstract complexity of $(\mathcal{P}, f)$ by describing an abstract realization, that is, a timed network that implements it. Any information that f needs to remember on later rounds, such as the position of the leftmost 1 on a certain row, is stored in a disposable switch. A single bit of information that could be stored in any of a rounds and accessed in any of b later rounds can be stored in an (a, b)-switch, requiring $O(\max(ab, b^2))$ primitive components.

By assumption, there is a circuit of size C that computes the function g. We apply Lemma 8 to produce a timed network A_g of size $O(m^2C^2)$. By connecting the input and output terminals of A_g in a suitable way (see Fig. 15), we can use it to read the data stored in disposable switches, compute g on it, and store the result (the pair $g(\vec{b}, R) = (\vec{c}, S)$) in another set of disposable switches, where it can be accessed during the sculpt stage. It suffices to store $2(m^2 + 4m)$ bits of information for at most $O(m^8)$ rounds: the contents of R and S, and the vectors $\vec{b}$ and $\vec{c}$, each of which is stored by flipping exactly one of $4m = |B_m|$ switches. The number of components required for this is polynomial in m.

We construct the abstract realization by connecting each input terminal to the appropriate output terminal, possibly via a disposable switch to store a bit of data, and possibly via other disposable switches to read bits of data and connect to different output terminals depending on the result. Recall that the number of substages in each stage, the number of rounds in each substage, as well as the position of the southwest corner of the clay on each round of the sculpt stage, are independent of $(\vec{b}, R)$. Then the only data the process needs to access on any given round is whether a fixed coordinate belongs to E_0 (which amounts to reading a single switch), and on the last round, the vector $\vec{c}$ (which amounts to reading $4m$ switches and branching off as soon as a flipped one is found). Hence, each round can be implemented using poly(m) primitive components in addition to the ones already listed. □

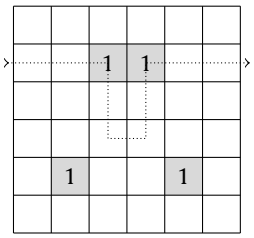
10. Timing

Physical universality says that a particular transformation should happen at a particular moment in time, which means we must have perfect control over all the time taken by our calculations and rewritings. While our automata in Section 6 carry a time component, and our simulations of it in Section 7 have an exact multiplicative delay, our two previous sections mostly ignored timing issues. In this section we revisit these constructions (specifically Lemma 12 and Lemma 11), and explain the simple tricks needed to make them take constant, or in fact essentially any desired, number of steps. We take a somewhat informal approach: the timing details are formally tricky, as *everything* has to do with timing, but the ideas are simple.

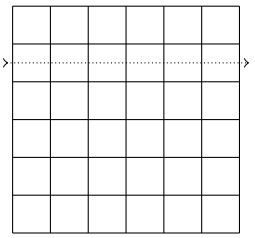
The main result of this section is that the concrete realization of Lemma 12 in Lemma 11 can be performed so that the transformation takes exactly time $C + d$, where C is a sufficiently large constant, and d a *delay function* computable by a polynomial circuit from the initial pattern $(s(\vec{b}), \tau_{(n,n)}(R) \sqcup P) \in \mathcal{P}$. Note that this claim is specific to the border process constructed in Lemma 12, and does not apply to every border process (some border processes inherently cannot be implemented with arbitrary delays). We also note that implementing arbitrary delays is basically equivalent to implementing a process in *constant-time*, meaning with constant 0 delay function.

The main notion we need is *deterministic timing*. A border process has deterministic timing, if, from only looking at the sequence of entrances and exits during the border process, we can tell (at the end) how long the machine has spent *inside* the pattern (without knowing the initial pattern, only that it belongs to the set of patterns the process is defined on). This property is crucial for being able to control the total time of the process, namely the sequence of entrance and exit positions is the only information we can extract from a particular pattern, so if there are two patterns where the sequences are the same but the head spends a different time inside the rectangle, then there is no way for us to know how much we should compensate for delays on the outside.

Example 1. Consider the border process $(\mathcal{P}, f)$ of size 6×6 and depth 1 where $\mathcal{P}$ contains those pairs $(\vec{b}, P)$ such that all 1s in P lie in $[1, 4]^2$, and which always inserts the head on row 4 (second row from the top). This process does not have deterministic timing:

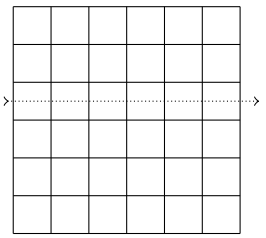


Takes 13 steps.

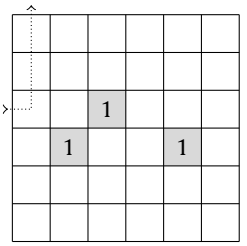


Takes 7 steps.

Consider then the border process $(\mathcal{P}, f)$ of size 6×6 and depth 1, where $\mathcal{P}$ contains those pairs $(\vec{b}, P)$ where the 1s are all contained in the rectangle $[1, 4] \times [2, 3]$, and the head is always inserted on row 3. For example:



Takes 7 steps.



Takes 7 steps.

This process has deterministic timing: either the head travels straight through the pattern (in exactly 7 steps), or it turns exactly once, and we can deduce its path from the exit coordinate. ○

The process in Lemma 12 not only has deterministic timing, but *stepwise deterministic timing*. This means that we can deduce the time that was spent inside $[0, 2n+m-1]^2$ between two given rounds from only the entrance and exit coordinates and the number of the round (without knowing the initial pattern, and without knowing even the history of the process). We also talk about individual rounds having deterministic timing, and stepwise deterministic timing means precisely that every step has deterministic timing. For example, the processes described in Example 1 also have this stronger property.

Lemma 13. *The border process described in Lemma 12 has stepwise deterministic timing.*

Proof. In the probe stage, we repeatedly send the head of M into the pattern from the west on the bottommost row $n \leq i < n+m$ that contains a 1 (recall that the process is defined on patterns of the form $P \sqcup \tau_{(n,n)}(R)$, where P is a fixed pattern that contains the clay and large empty areas). Since P has unobstructed center, the head of M turns south upon encountering the westmost 1 on its row, which lies in $[n, n+m-1]^2$. The time taken by the head depends only on the position of this 1, which in turn can be deduced from the exit position, so this round has deterministic timing. If a 1 was encountered, it is moved into the landfill in a polynomial number of rounds, and if not, we take an equal number of "do-nothing" rounds by sending the head through the bottom row of the pattern. All of these rounds have deterministic timing.

The remaining stages have deterministic timing: we have moved all 1s of the original pattern $\tau_{(n,n)}(R)$ to the landfill and will never interact with them again, so one simple needs to check that the specific moves used in transport and sculpting can be completely deduced from the entrance and exit coordinates. □

We note that seeing deterministic timing is even easier – after the probe stage, already have complete information of the pair $(\vec{b}, R)$, so after this stage we know exactly what the pattern in $[0, 2n+m-1]^2$ looks like.

The next important thing to ensure is that the computation always takes the same time *outside* the pattern. This is easy: by definition our abstract realizations are constant time networks, and Lemma 10 shows that we can simulate a constant time network with M in constant time. "Wiring" the implementation to the catcher system introduces its own delay, but one can easily ensure that all these delays take exactly the same time.

Now, we should precisely correct for the delays inside the pattern to ensure that the entire process takes constant time (apart from the desired delay). Recall from Definition 12 that for a given concrete realization of a border process $(\mathcal{P}, f)$ and

pair $(\vec{b}_1, P) \in \mathcal{P}$, we denoted by $s_1 < t_2 < s_2 < t_3 < s_3 < \cdots$ the time steps at which the head enters and leaves the region D at border positions $\vec{c}_1, \vec{b}_2, \vec{c}_2, \vec{b}_3, \vec{c}_3, \ldots$.

Lemma 14. *Suppose that a depth-k border process $(\mathcal{P}, f)$ has deterministic timing, and the total number of steps can be computed from the final sequence of entrance and exit coordinates $\vec{b}_1, \vec{c}_1, \vec{b}_2, \ldots, \vec{b}_k$ by a circuit of size* poly(m, C). *Then the concrete realization of $(\mathcal{P}, f)$ in Lemma 11 can be constructed so that the time step s_k of the final entrance is independent of $(\vec{b}_1, P)$.*

Proof. We already showed in Lemma 8 that our machine can compute any Boolean circuit in constant time (the input and output being stored in simulated switches). At the end of the process, simply calculate the time spent inside the pattern (with a constant-time circuit), and use the output to send the head off to a delay circuit, which compensates accordingly. □

Of course in the situation of stepwise deterministic timing, there is trivially a polynomial size circuit computing the time spent inside from the entrance-exit pairs, simply because there are only polynomially many cases to cover on each step. In this case, one could also correct for the time delays after each step, by "mechanically" redirecting the head to a delay circuit depending on the output coordinate, as the exits happen in different physical locations (the input coordinate needs to be stored in a switch of course).

We arrive at the desired conclusion that the concrete realization of the border process constructed in Lemma 12 can be performed in constant time, and adding an arbitrary delay is then trivial.

Lemma 15. *The concrete realization constructed in Lemma 11 for the border process described in Lemma 12 can be performed with an arbitrary delay function d, as long as it is computable with a polynomial size circuit and takes polynomial values.*

Proof. As explained above the lemma, combining Lemma 13 and Lemma 14 gives a constant-time solution. In the process described in Lemma 12, we can always easily determine the initial pattern from the entrance-exit sequence, indeed this is known after the probe stage (the whole point of that stage, of course, is to calculate the initial pattern, as we need it for the sculpt stage). We simply have the head of M go through an artificial delay circuit depending on the value of the delay function d. □

11. Physical universality of *M*

We are now ready to prove our main result, the physical universality of M.

Theorem 4. *The Turing machine M is efficiently physically universal in the moving head model.*

Proof. Let $m \in \mathbb{N}$, let $A \subset \mathcal{P}^*_{Q,\Sigma}(m)$ be a set of headful patterns with disjoint 0-orbits, and let $g : A \to \mathcal{P}_{Q,\Sigma}(m)$ be a function. Let $A' \subset A$ be the set of those patterns $R \in A$ for which $g(R)$ contains the head of M. For each $R \in A'$, Lemma 5 and its analogue for the inverse machine M^{-1} states that the escape time of M from R when it is surrounded by 0s, and the escape time of M^{-1} from $g(R)$ when it is surrounded by 1s, are both $O(m^4)$. Denote their sum by $\epsilon'(R)$, and for $R \in A \setminus A'$, define $\epsilon'(R)$ as just the escape time of M from R. For $R \in A'$, let $b(R) \in B_m$ be the coordinate where M^{-1} escapes from $g(R)$, let $g'(R)$ be the contents of $[-1, m]^2$ as this happens. For $R \in A \setminus A'$, we (somewhat arbitrarily) define $b(R) = (-1, 0)$ and $g'(R) = g(R)$. Lemma 1 lets us compute ϵ' and b by a circuit of polynomial size in m and the circuit complexity of g.

For $R \in A$, let $p(R) \in \mathcal{P}^*_{Q,\Sigma}(m+2)$ be R padded with a thickness-1 border of 0s on all sides, and let $p'(R) \in \mathcal{P}_{Q,\Sigma}(m+2)$ be $g'(R)$ padded similarly by 1s. By Lemma 12, there exists a number n, a pattern $P \in \Sigma^{[0,2n+m+1]^2 \setminus [n,n+m+1]^2}$, and a border process $(\mathcal{P}, f)$ of size $2n+m+2$, polynomial depth and abstract complexity, which that transforms a pattern $P \sqcup p(R) \in \mathcal{P}$ for $R \in A$ into another pattern containing $p'(R)$ at $[n, n+m+1]^2$ and the head pointed toward $b(R) + (n+1, n+1)$ on the border of P. Lemma 11 shows that there is a concrete realization $y \in \Sigma^{\mathbb{Z}^2 \setminus [0,2n+m+1]^2}$ for $(\mathcal{P}, f)$ where all 1s are within polynomial distance from the origin, and by Lemma 15 we can add an arbitrary (polynomial) delay (computable by a polynomial size circuit) on top. We take the delay to be $C - \epsilon'(R)$ for some C of polynomial size.

In this concrete realization, the number of steps M takes between leaving P for the first time and entering it for the final time is now $C' + C - \epsilon'(R)$ for some constant C'. Since the pattern R is surrounded by 0-cells in P on all four sides, as is $g'(R)$ when the process ends, the number of steps taken by the head between leaving $[n, n+m-1]^2$ for the first time and re-entering for the last time is $C' + C - \epsilon'(R) + 2n + 2$.

Note that the padding provided by p and p' does not affect the movement of the head as it enters or leaves the square $[n, n+m-1]^2$. If $R \in A'$, then we have $M^t(y \sqcup P \sqcup R)|_{[n,n+m-1]} = g(R)$ since the additional $\epsilon'(R)$ steps account exactly for the head leaving $[n, n+m-1]$ for the first time, and transforming $g'(R)$ into $g(R)$ after re-entering for the last time. If $R \in A \setminus A'$, then the same holds because when the head is sent into P for the last time, it is directed toward the coordinate $b(R) + (n+1, n+1)$ but will not enter $g'(R)$ at time t. Since C', C and n are all polynomial in m and the circuit complexities of A and g, the claim follows. □

It is easy to show that PU in the moving head model is stronger than PU in the moving tape model:

Lemma 16. *Every Turing machine that is (efficiently) physically universal in the moving head model is (efficiently) physically universal in the moving tape model.*

Proof. Pairs $(q, P) \in Q \times \Sigma^{[-m,m]^d}$ in the moving tape model correspond in a natural way to patterns in $\mathcal{P}^*_{Q,\Sigma}(2m+1)$ with heads at $(m, m, \dots, m)$ in the moving head model, and this transformation preserves the disjoint 0-orbits property. This directly translates instances of PU in the moving tape model to instances of PU in the moving head model. □

Corollary 2. *There exists a two-dimensional reversible Turing machine which is efficiently physically universal in both the moving head model and the moving tape model.*

A Turing machine with one state that always moves to the right is obviously topologically mixing in the moving tape model. In the moving head model, it is to our knowledge previously unknown whether Turing machines can be topologically mixing. The one-dimensional case is the most interesting one, but our machine provides at least a two-dimensional example, since it is easy to see from the construction that we can freely choose the time parameter t as long as it is large enough (topological transitivity follows directly as a special case of physical universality).

We show a somewhat stronger fact, which also follows from the proof of Theorem 4 with minor modifications.

Definition 16. A dynamical system (X, f) is *topologically mixing of order k* if given nonempty open sets $U_0, U_1, \dots, U_k$ in X, there exists m such that for all $n_1, n_2, \dots, n_k \geq m$ there exists a point $x \in X$ such that $x \in U_0$ and $f^{n_1+\cdots+n_i}(x) \in U_i$ for all $i \geq 1$.

Theorem 5. *There exists a reversible Turing machine which is mixing of all finite orders in the moving head model.*

Proof sketch. We show that M is such a machine. We may assume the open sets $U_0, U_1, ..., U_k$ are defined by individual patterns in $P_i \in \mathcal{P}^*_{Q,\Sigma}(N)$ for the same N. It is easy to construct a border process that successively goes through these patterns, by repeating the transport and sculpt stages k times. Using Lemma 11, we obtain a configuration that realizes this border process.

It does not immediately follow from the lemma that we can to add arbitrary delays in this process: indeed, in the definition of a concrete realization, we do not make any guarantees about the timing of the intermediate steps (only the last one). However, in the proof, it is easy to add a delay at some arbitrary point of the computation, by sending the head to a delay area away from the rest of the computation zone.

More precisely, allocate for each $i = 1, \dots, k$ an area $[a_i, b_i] \times [\ell, \infty)$ (for large enough ℓ and $b_i - a_i$) on the plane, where we can realize any long enough delay by making the head jump through hoops. The delays are independent as long as the areas are separated from each other and from the rest of the computation. Now simply send the head to the ith delay area at some arbitrary point of the process of sculpting P_i. □

Constructing a one-dimensional physically universal Turing machine seems more difficult. If one is to mimic our construction, information from the initial pattern needs to be fetched by some type of gadget rather than by simply shooting the head in the correct cell. We conjecture that such Turing machines exist, but we do not have a candidate.

12. Modifications

We can make slight modifications to our physically universal machine M and obtain machines with different universality properties.

Theorem 6. *There exists a two-dimensional reversible Turing machine that is physically universal in the moving tape model, but not the moving head model.*

Proof sketch. Define a Turing machine M' as follows. The tape alphabet is $\Sigma = \{0, 1\}^2$, and the state set is $Q' = Q \times \{1, 2\}$, where Q is the state set of M. The idea is that M' has two binary tapes, one of which is analogous to that of M and the other is immutable but affects the movement of the head. The machine behaves as follows:

1. If the machine is in state (q, b), and there are 1-symbols on both layers of the tape under the head, and 1-symbols on the first layer in all eight neighbors, then the state becomes $(q, 3 - b)$.
2. After this, if the machine is in state $(q, 1)$, then it takes one step forward, retains its state and does not modify the tapes. Intuitively, in a state $(q, 1)$ the machine is partially deactivated.
3. If the machine is in state $(q, 2)$ instead, then it behaves as M would on the first layer of the tape, reading and modifying it, then moving into a new position and assuming a new state $(q', 2)$.

It is immediately clear that M' is not physically universal in the moving head model, since it cannot modify the second layer of the tape. In dynamical systems terms, the second tape is a nontrivial invariant factor, which prevents even transitivity. Lemmas 5 and 6 apply to M' with only minor modifications.

We sketch the proof of physical universality in the moving tape model. Consider a set of patterns $A \subset \mathcal{P}^*_{Q',\Sigma}(m)$ and a function $g : A \to \mathcal{P}_{Q',\Sigma}(m)$. As with M, we can construct a catcher system around the square $[0, m-1]$ that redirects the head after its exit. In case the head exits in some state $(q, 1)$, it cannot be redirected, so we add width-3 stripes of 1-cells on both layers behind the four catcher arrays that change the state to $(q, 2)$, and add a second catcher system that is then able to redirect the head. At the same time we obtain full information about its state.

After catching the head, we repeatedly probe the square $[0, m-1]^2$ to obtain the positions of 1s on the first layer, as in the probe stage of the proof of Lemma 12. Since the head will always see a 0 in some neighboring coordinate, it will not change its state into any $(q, 2)$ during this process, ignoring the second layer and behaving like M instead. Next, on each coordinate of $[0, m-1]^2$ in turn we transport a 3×3 pattern of 1-cells on the first layer and send the head through it to test whether there is a 1 on the second layer, then move these 1s out of the way again.

Once we have obtained full information about the input pattern, the head is redirected into one of $|A|$ different patterns, one for each $R \in A$, which evolves into $g(R)$ after a certain number of steps when the head is sent into it through a certain border coordinate. We can control the number of steps taken by the head of M' so that the total number of steps required for this does not depend on the input pattern. This construction shows that M' is physically universal in the moving tape model. □

Note that the above proof does not show M' to be efficiently physically universal, since the number of patterns we insert into the gadget may be exponential in m. It remains an open problem whether there exists a Turing machine that is efficiently physically universal in the moving tape model, but not in the moving head model.

Our machine M is allowed to inspect several tape cells simultaneously, which is not the case for a classical Turing machine. It is possible to simulate general reversible Turing machines by reversible classical Turing machines. We sketch the proof that this can be done in a way that preserves physical universality.

Definition 17. A *classical reversible Turing machine* first applies a *state-symbol permutation* $\chi \in \mathrm{Sym}(Q \times \Sigma)$ to the current state and the current tape symbol, and then performs a *state-dependent shift*, i.e. moves the head as a function of the current state only.

This terminology is from [1]. This is equivalent to the quadruple model of reversible Turing machines defined in [18], used also in [11]. Another model that is often considered is that of a classical non-reversible Turing machine (which can rewrite and move at the same time), which is simply postulated to have a bijective global transition rule. This is the model used in e.g. [12], and corresponds to the quintuple model in [18]. Definition 17 is the more restrictive out of these two, as shown in [18, Section 5.1.3].

Theorem 7. *There exists a two-dimensional classical reversible Turing machine that is physically universal.*

Proof sketch. Our machine M has the property that its behavior is composed of a joint permutation of the state and a local neighborhood on the tape (a *local permutation* in the terminology of [1]) and then a state-dependent shift. We explain how we can modify M so that the local permutation is replaced by a sequence of state-symbol permutations.

The basic idea is the following. For any machine that admits such a decomposition, with state set Q and alphabet Σ, we can construct a reversible classical Turing machine which is an "mth root" for it, for any large enough m. For this, let $Q' = Q \times \{0,1\}^2 \times \{0, 1, ..., m-1\}$ for large m, and $\Sigma' = \Sigma$. On every time step, we increment the final component modulo m. This allows us to think of the dynamics as being composed of m distinct steps. On these steps, the head walks around deterministically, performing even permutations on the current tape symbol and the $Q \times \{0,1\}^2$-component of its state. Since we can swap tape bits with the two bits memorized in the state, we can effectively perform arbitrary joint permutations on Q and two symbols of the tape. By Lemma 5 of [1], this allows us to perform any even permutation of $Q \times \{0,1\}^2 \times \{0,1\}^N$ where $N \subset \mathbb{Z}^2$ is finite. Every permutation that fixes the two bits in the $\{0,1\}^2$-component of the state is even, so this allows us to perfectly simulate the behavior of an arbitrary Turing machine composed of a local permutation followed by a state-dependent shift.

This does not quite give physical universality, as the simulation happens every m steps rather than on every step, leading to a parity (or rather modulo m) issue, and also because the state contains two bits whose values are never modified. To combat these issues for our specific machine M, we return to $\bar{M}$ and first observe that its state set $\bar{Q}$ is in bijection with $\{0,1\}^2$. Thus, there is no need to add the helper bits $\{0,1\}^2$ in the construction of the above paragraph (again since permutations touching three bits at once generate all even permutations). Next, instead of using a fixed Cartesian product $\bar{Q} \times \{0, 1, ..., m-1\}$, we use $\{(d, i) \mid d \in \bar{Q}, i \in N_d\}$, where $N_d = \{0, 1, ..., m-1\}$ for $d \neq \uparrow$ and $N_\uparrow = \{0, 1, ..., m\}$.

Since we have again slowed down upward movement, we do not have any parity issues. Furthermore, the counter value never needs to be "read" to know what we need to write on the tape when executing a physical transformation, since other things being equal, two distinct counter values are put in the same orbit, and thus in the definition of physical universality

we cannot have both in the same initial pattern. Thus the proof of PU works the same as that of the main theorem, up to timing details. □

13. Additional information and open questions

In this section, we collect questions from previous sections under one heading, and ask several more questions. We also sketch the proofs of two additional results: we show that physical universality of Turing machines is closed under a natural group of transformations, and show that the maximal number of steps the head of $\bar{M}$ can stay inside $[0, n-1]^2$ is $\Theta(n^3)$ when restricted to configurations with $O(n)$ many 1s.

13.1. Questions asked in previous sections

Conjecture 1. *There exists a one-dimensional reversible Turing machine which is physically universal in both the moving head model and the moving tape model, and is topologically mixing.*

Question 1. Does there exist a Turing machine that is efficiently physically universal in the moving tape model, but not in the moving head model?

13.2. Alternative definitions

13.2.1. PU from headless patterns

A natural alternative to requiring that *all* the patterns P_i in the definition of PU contain the head is to require that none of them do. Say a machine is *physically universal from headless patterns*, if, whenever P, R are k-tuples of patterns and the patterns in P contain no heads, then P is physically transformable to R in the moving tape model. Physical universality from headless patterns holds for our machine under this definition as a simple consequence[6] of Theorem 4.

13.2.2. PU from context-separable patterns

Question 2. Does there exist a Turing machine M such that every configuration with finite support is in the same orbit in the moving tape model?

If such a Turing machine exists, it is vacuously physically universal in the moving tape model, under our definition.

A possible alternative definition of physical universality that avoids this problem would be that the patterns $P_i \in \mathcal{P}^*_{Q,\Sigma}(m)$ do not necessarily have distinct 0-orbits, but simply admit some context $x \in \mathcal{P}_{Q,\Sigma}(\mathbb{Z}^d \setminus [0, m-1]^d)$ such that the resulting orbits are disjoint. Say a Turing machine is physically universal from *context-separable patterns* if any tuple P admitting such a context is physically transformable to any tuple R in the moving head model. One can state an analogous definition in the moving tape model.

Question 3. Is M physically universal from context-separable patterns, in either model? Is there any Turing machine with this property?

13.2.3. Combinatorial entanglement

Our machine M is symbol-conserving, so one can find a continuous function $P : Q \times \Sigma^{\mathbb{Z}^d} \to \mathrm{Sym}_0(\mathbb{Z}^d)$ where $\mathrm{Sym}_0(\mathbb{Z}^d)$ denotes the discrete group of permutations with finite support, such that in the moving tape model we have $M(q, x) = (p, \tau_{\vec{w}}(P(q, x)(x)))$ for all $(q, x) \in Q \times \Sigma^{\mathbb{Z}^d}$ (compare to the definition of the moving tape model in Section 2). The choice of permutation is not unique, but in our informal description of M and $\bar{M}$ we essentially specified P already: the permutation used is the one that swaps the 0 and 1 visible in (1).

Fixing this P, one can ask for a stronger physical universality allowing "entanglement": in addition to initial patterns $P_1, \dots, P_k$, whose 0-padded orbits are disjoint and final patterns $R_1, \dots, R_k$, we may assume a partially defined bijection between positions of P_j and R_j sharing the same symbol, and require that not only $M^t(P_j \sqcup x)_{[0,n-1]^2} = R_j$ but the cocycle $P(M^t)$ respects the specified partial bijection. For example, if the transformation $P_j \mapsto R_j$ being implemented only flips the central bit, it is natural to require that the other bits are literally not moved, i.e. the partially defined bijection fixes them. In our proof of physical universality, these bits would be sculpted from the clay, and the original bits would become garbage. It seems very difficult to undo this damage.

Let us say a machine is *physically universal on combinatorially entangled* 0*-finite configurations* if it is physically universal in the sense of the previous paragraph.

[6] This implication holds for all machines that admit a semilinear spacetime diagram in the moving head model, from at least one finite initial configuration. Our machine has this property for *all* finite initial configurations. Langton's ant has it from some initial finite patterns and conjecturally all of them.

Question 4. Is M physically universal on combinatorially entangled 0-finite configurations? Does there exist a Turing machine that is?

13.2.4. Physical universality in quantum Turing machines

One may wonder if there are quantum analogues of our results. We are not aware of formal frameworks for TMH and TMT dynamics of quantum Turing machines, so we leave open no precise question. However, given the existence of PU quantum CA proved in [22], it is of interest to try to prove quantum analogs of our results.

Question 5. Is there a natural definition of PU for quantum Turing machines? If so, is there a PU quantum Turing machine?

13.3. Turmites

13.3.1. Langton's ant

It seems natural to ask questions about Langton's ant, since it is perhaps the best-known individual Turing machine. To each pattern $P \in \mathcal{P}^*_{Q,\Sigma}(m)$, where $|Q| = 4, |\Sigma| = 2$ are the states and tape symbols of Langton's ant, we can associate a *parity* $\pi(P) \in \{0, 1, ?\}$ by adding modulo 2 the coordinates of the head and the indicator bit of whether it is traveling vertically, and defining $\pi(P) = ?$ if a head is not visible.

Question 6. Can Langton's ant physically transform a tuple $P \in (\mathcal{P}^*_{Q,\Sigma}(m))^k$ to $R \in (\mathcal{P}_{Q,\Sigma}(m))^k$ whenever the patterns in P are context-separable and all patterns in P (respectively R) with a non-? parity have pairwise the same parity?

It seems difficult to resolve this question without first resolving topological transitivity and the question of whether Langton's ant eventually enters one of the standard cycles from all finite initial patterns.

One can state many variants, e.g. replace context-separability by disjoint 0-orbits as we did with M. Another, perhaps somewhat more tractable type of PU one could investigate for Langton's ant is the following: Say a machine is *headless-to-headless physically universal* if $(P_0, P_1, ..., P_{k-1})$ is physically transformable to $(R_0, R_1, ..., R_{k-1})$ whenever none of the P_i or R_i contain a head. This is of course a weakening of PU from headless configurations, so our machine M has this property.

Question 7. Is Langton's ant headless-to-headless physically universal in the moving head model?

It is easy to verify experimentally that this is true with pattern size $m = 1$.

Question 8. Is Langton's ant headless-to-headless physically universal when restricted to patterns $P, R \in (\mathcal{P}_{Q,\Sigma}(m) \setminus \mathcal{P}^*_{Q,\Sigma}(m))^k$ with $m = 2$?

We also wonder if Langton's ant can physically transform any individual pattern to the all-zero pattern (a very special case of topological transitivity).

13.3.2. Other turmites

In [15], it is shown that every turmite in the sense of [6] is either Turing universal in a certain sense or 4-periodic in the moving head model. Like Langton's ant, turmites that are not 4-periodic have no periodic points in the moving head model [8]. It seems difficult to perform controlled tape manipulation with any of them, or to disprove the possibility of such control.

Question 9. Is every aperiodic turmite PU up to parity caveats (see the previous section)? Is any?

13.4. Invariance and the inverse rule

We show that physical universality on finite configurations has good invariance properties. The group of invertible d-by-d matrices $\mathrm{GL}(d, \mathbb{Z})$ acts on $X_Q \times \Sigma^{\mathbb{Z}^d}$ in the obvious way: $Ax_{\vec{v}} = x_{A^{-1}\vec{v}}$. For X a subshift, $\mathrm{Aut}(X)$ is its *automorphism group*, the group of shift-commuting homeomorphisms $f : X \to X$. Write $\mathrm{Homeo}(X_Q \times \Sigma^{\mathbb{Z}^d})$ for the group of all homeomorphisms. In the following statement, for notational convenience we consider $\Sigma^{\mathbb{Z}^d}$ in a natural way as a subset of $X_Q \times \Sigma^{\mathbb{Z}^d}$. The proof is simply a matter of opening up the definitions, but we go through the motions. For groups G, H, by $G \leq H$ we denote that G is a subgroup of H.

Proposition 1. *Let $G \leq \mathrm{Homeo}(X_Q \times \Sigma^{\mathbb{Z}^d})$ be the group generated by $\mathrm{GL}(d, \mathbb{Z})$ and $\mathrm{Aut}(X_Q \times \Sigma^{\mathbb{Z}^d})$. Let $M = (Q, \Sigma, N, \Delta)$ be a Turing machine, considered in the moving head model. If M is physically universal on a-finite configurations and $g \in G$, then $g \circ M \circ g^{-1}$ is physically universal on b-finite configurations where $g(a^{\mathbb{Z}^d}) = b^{\mathbb{Z}^d}$.*

Proof. In this proof, the notation $[P]_{\vec{v}}$ for a pattern $P \in \mathcal{P}_{Q,\Sigma}(m)$ means the *cylinder set* $\{x \in X_Q \times \Sigma^{\mathbb{Z}^d} \mid (\tau^{\vec{v}}x)|_N = P\}$ of those configurations that contain the pattern at position $\vec{v}$.

Suppose $m, k \in \mathbb{N}$, and let $P_0, \dots, P_{k-1} \in \mathcal{P}^*_{Q,\Sigma}(m)$ and $R_0, \dots, R_{k-1} \in \mathcal{P}_{Q,\Sigma}(m)$ be patterns in $X_Q \times \Sigma^{\mathbb{Z}^d}$ such that the configurations $P_j \sqcup b^{\mathbb{Z}^d \setminus [0,m-1]^d}$ have disjoint $(g \circ M \circ g^{-1})$-orbits. We need to show that there exists a partial configuration $x \in \mathcal{P}_{Q,\Sigma}(\mathbb{Z}^d \setminus [0, m-1]^d)$ and $t \in \mathbb{N}$ such that $(g \circ M \circ g^{-1})^t (P_j \sqcup x)|_{[0,m-1]^d} = R_j$ for all $j \in [0, k-1]$.

For this, consider the configurations $y_j = g^{-1}(P_j \sqcup b^{\mathbb{Z}^d \setminus [0,m-1]^d})$. By basic properties of $\mathrm{Aut}(X_Q \times \Sigma^{\mathbb{Z}^d})$, the y_j are configurations of finite a-support containing a head, and since the configurations $P_j \sqcup b^{\mathbb{Z}^d \setminus [0,m-1]^d}$ have disjoint $(g \circ M \circ g^{-1})$-orbits, the configurations y_j have disjoint M-orbits. Let $P'_0, \dots, P'_{k-1} \in \mathcal{P}^*_{Q,\Sigma}(m')$ for a large even m' be central patterns (i.e. extract the contents of $[-m'/2+1, m'/2]^d$) of the configurations y_j containing the support and the head.

For $R_0, \dots, R_{k-1} \in \mathcal{P}_{Q,\Sigma}(m)$, extend them by bs (or anything else), apply g^{-1}, and in these configurations apply continuity of g to obtain central patterns $R'_0, \dots, R'_{k-1} \in \mathcal{P}_{Q,\Sigma}(m')$ (increasing m' if needed) such that all configurations contained in the cylinder $[R'_i]_{(-m'/2+1,-m'/2+1,\dots,-m'/2+1)}$ contain R_0 in $[0, m-1]^d$.

By physical universality of M (applied through conjugation by translation by $(m'/2-1, m'/2-1, \dots, m'/2-1)$), there exists a partial configuration $x' \in \mathcal{P}_{Q,\Sigma}(\mathbb{Z}^d \setminus [-m'/2+1, m'/2]^d)$ and $t \in \mathbb{N}$ such that $M^t(P'_j \sqcup x')|_{[-m'/2+1,m'/2]^d} = R'_j$ for all $j \in [0, k-1]$.

It follows that for each j, by conjugating back we get that in $(g \circ M \circ g^{-1})^t (g(P'_j \sqcup x'))$ the contents of $[0, m-1]^d$ is R_j. To complete the proof, we need to show that $g(P'_j \sqcup x') = P_j \sqcup x$ for some partial configuration $x \in \mathcal{P}_{Q,\Sigma}(\mathbb{Z}^d \setminus [0, m-1]^d)$, assuming m' was picked large enough.

Since P'_j was extracted from $g^{-1}(P_j \sqcup b^{\mathbb{Z}^d \setminus [0,m-1]^d})$, just like in the choice of R'_i we have by continuity that $g([P'_i]_{(-m'/2+1,-m'/2+1,\dots,-m'/2+1)})$ contains P_j in $[0, m-1]^d$ if m' was taken large enough. In fact, again by continuity, by picking larger m', we may assume every configuration $g([P'_i]_{(-m'/2+1,-m'/2+1,\dots,-m'/2+1)})$ contains $P_j \sqcup b^{[-\ell,\ell]^d \setminus [0,m-1]^d}$ in $[-\ell, \ell]$, for a large ℓ.

Observe now that since g has a local rule (up to conjugation by a linear transformation), if ℓ is large enough, for any $\vec{v} \notin [-\ell, \ell]^d$ we have that $g(P'_j \sqcup x')_{\vec{v}}$ depends only on the values of $P'_j \sqcup x'$ outside the (finite) a-support of y_j, equivalently outside the non-a support of P'_j. Since P'_j contains a outside its non-a support and x' is independent of j, we have that indeed there exists some x such that for all j, $g(P'_j \sqcup x') = P_j \sqcup x$. □

The group G discussed in Proposition 1 is just the normalizer of the group of shift maps (isomorphic to $\mathbb{Z}^d$) in $\mathrm{Homeo}(X_Q \times \Sigma^{\mathbb{Z}^d})$, and is also called the *extended symmetry group* [16].

Corollary 3. *The machine M^{-1} is efficiently physically universal in the moving head model and the moving tape model with zero symbol* 1.

Proof. Let again G be the group of extended symmetries of $X_Q \times \Sigma^{\mathbb{Z}^d}$. By Lemma 3 we have $M^{-1} = \sigma \circ M \circ \sigma$ where $\sigma(q[\vec{v}], x) = \mu(\omega(q[\vec{v}])\beta(x)) = \mu \circ (\omega \times \beta)$. It is easy to verify that $\mu \in G$, and $\omega \in \mathrm{Aut}(X_Q), \beta \in \mathrm{Aut}(\Sigma^{\mathbb{Z}^d})$ so $\omega \times \beta \in \mathrm{Aut}(X_Q \times \Sigma^{\mathbb{Z}^d}) \leq G$. PU in the moving head model follows from Proposition 1. □

A similar proof works in the moving tape model. One can directly consider the group obtained from $\mathrm{Aut}(X_Q \times \Sigma^{\mathbb{Z}^d})$ by fixing the position of the head, but as it is easy to see that this is just the group generated by reversible cellular automata and reversible Turing machines in the moving tape model in the sense of [1], we state the proposition directly in this way. In the following proposition, $\mathrm{Aut}(\Sigma^{\mathbb{Z}^d})$ acts in a natural way on $Q \times \Sigma^{\mathbb{Z}^d}$.

Proposition 2. *Let $G \leq \mathrm{Homeo}(Q \times \Sigma^{\mathbb{Z}^d})$ be the group generated by $G_1 = \mathrm{GL}(d, \mathbb{Z})$, the group of Turing machines $G_2 = \mathrm{RTM}(Q, \Sigma)$ from [1], and $G_3 = \mathrm{Aut}(\Sigma^{\mathbb{Z}^d})$. Let $M = (Q, \Sigma, N, \Delta)$ be a Turing machine, considered in the moving tape model. If M is physically universal on a-finite configurations and $g \in G_i$, then $g \circ M \circ g^{-1}$ is physically universal on b-finite configurations, where*

- *$b = a$ if $g \in G_1 \cup G_2$, and*
- *b is defined by $g(a^{\mathbb{Z}^d}) = b^{\mathbb{Z}^d}$ if $g \in G_3$.*

While the inverse machine is PU on 1-finite configurations, we do not know if it is PU on 0-finite configurations, equivalently whether M is PU on 1-finite configurations. Indeed, while Lemma 5 applies to the inverse, Lemma 6 does not, and indeed on a 0-finite configuration, M^{-1} does not always eventually escape it by running to infinity along an arithmetic progression, and the spacetime diagram of a finite-support configuration need not be a semilinear set.

Question 10. Is M physically universal on 1-finite configurations? Equivalently, is M^{-1} physically universal on 0-finite configurations?

13.5. Escape time

We show in Lemma 5 that the head of the machine escapes a square of side length n in $O(n^4)$ time steps. We do not know if this is optimal.

Question 11. For the machine M, do there exist, for infinitely many n, configurations with all 1-symbols and the machine head contained in $[0, n-1]^2$ such that the head stays in $[0, n-1]^2$ for $\Omega(n^4)$ steps?

Obviously in such configurations, $[0, n-1]^2$ must contain a positive density of 1s since the potential function must have a value of order $\Omega(n^4)$.

The potential function gives the correct polynomial rate of escape for configurations where there are $O(n)$ 1-symbols in $[0, n-1]^2$. Namely, the potential of such a configuration is obviously $O(n^3)$, so the head will escape in time $O(n^3)$, and indeed we can find configurations where the escape time is $\Omega(n^3)$ and the support consists of 7 partial arithmetic progressions.

Example 2. Let $a \geq 1, b \geq 0$. Consider the configuration $(\rightarrow[(1,0)], x)$ where the coordinates of 1-symbols in $x \in \{0,1\}^{\mathbb{Z}^2}$ form the set

$$A = A_{nw} \cup A_n \cup A_{ne} \cup A_{se} \cup A_{se2} \cup A_s \cup A_{sw}$$

where

$$A_{nw} = \{(-3,1) + i(-1,1) \mid i \in [0, a-1]\}$$
$$A_n = \{(0,-1) + i(0,1) \mid i \in [0, a]\}$$
$$A_{ne} = \{(2,0) + i(3,1) + b(2,0) \mid i \in [0, a-1]\}$$
$$A_{se} = \{(2,-2) + i(3,-1) + b(2,-2) \mid i \in [0, a-1]\}$$
$$A_{se2} = \{(2,-4) + i(1,-1) + b(1,-2) + a(0,-1) \mid i \in [0, a-1]\}$$
$$A_s = \{(0,-2) + i(0,-1) + b(0,-2) \mid i \in [0, a-1]\}$$
$$A_{sw} = \{(-3,-4) + i(-1,-1) + b(0,-2) + a(0,-1) \mid i \in [0, a-1]\}$$

We only give a "proof-by-simulation", and give an informal description of what happens.[7] To better follow the description below, we suggest running a computer simulation. Appendix A contains a description of $\bar{M}$ in the @RULE format used by the Golly cellular automaton simulator, and Appendix B contains a Lua script for drawing the configuration $(\rightarrow[(1,0)], x)$ in Golly.

Inspecting the trajectories of these configurations for various a and b, we see that the head "rebuilds a blown-up K-shape", in the sense that the configuration contains four separate segments of 1-symbols, A_n, A_{ne}, A_{se} and A_s, which the head slowly transports together to form a shape that resembles the letter K. This happens in a stages, where on the ith stage for $i \in [1, a]$, it transports one 1-symbol to each of the four tips of the K-shape. This process is similar to the activation of catchers in Section 8: the head travels in a rectangular spiral, bringing four 1-symbols one step closer to the spiral's center on each loop. When the 1-symbols reach the K-shape, the head travels south, hits A_{se2}, moves clockwise around the blown-up-K and begins the next stage.

Now we analyze the amount of time spent on this configuration inside the smallest rectangle R containing A. On the first stage, the machine visits all but five cells[8] of the rectangle $[0, 2b+2] \times [-2b-2, 0]$. For b large enough, this is at least b^2 cells. On each stage, the annulus visited grows strictly, and there are a stages, so in total the machine takes at least ab^2 steps before exiting R. Setting $a = b = n$, the region R is contained in a square of side length $\max(1+4a+2b, a+2+2a+2b) \leq 7n$ and the machine takes at least n^3 steps on it. This concludes the proof. ○

CRediT authorship contribution statement

The authors contributed equally.

[7] It should be a straightforward (but long) exercise to translate this into an explicit inductive proof by simply writing the formulas for the (finitely many types of) intermediate situations. We feel that it is not worth the trouble to torture ourselves and the readers with this, as this construction is only a partial result towards Question 11.

[8] It misses the northwest corner $(0,0)$ by the choice of the initial position of the head, and also misses 4 cells in the center.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcss.2022.08.003.

References

[1] Sebastián Barbieri, Jarkko Kari, Ville Salo, The group of reversible Turing machines, in: Matthew Cook, Turlough Neary (Eds.), Cellular Automata and Discrete Complex Systems, in: Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 49–62.
[2] Vincent D. Blondel, Julien Cassaigne, Codrin Nichitiu, On the presence of periodic configurations in Turing machines and in counter machines, Theor. Comput. Sci. 289 (1) (October 2002) 573–590.
[3] L.A. Bunimovich, S.E. Troubetzkoy, Recurrence properties of Lorentz lattice gas cellular automata, J. Stat. Phys. 67 (1–2) (1992) 289–302.
[4] L.A. Bunimovich, S.E. Troubetzkoy, Topological dynamics of flipping Lorentz lattice gas models, J. Stat. Phys. 72 (1–2) (1993) 297–307.
[5] Jean-Charles Delvenne, Vincent D. Blondel, Quasi-periodic configurations and undecidable dynamics for tilings, infinite words and Turing machines, Theor. Comput. Sci. 319 (1) (June 2004) 127–143.
[6] A.K. Dewdney, Two-dimensional Turing machines and turmites make tracks on a plane, Sci. Am. 261 (1989) 180–183.
[7] Anahi Gajardo, A symbolic projection of Langton's Ant, in: Discrete Mathematics & Theoretical Computer Science, DMTCS Proceedings vol. AB, Discrete Models for Complex Systems (DMCS'03), January 2003.
[8] David Gale, Jim Propp, Scott Sutherland, Serge Troubetzkoy, Further travels with my ant, in: David Gale (Ed.), Tracking the Automatic ANT: And Other Mathematical Explorations, Springer New York, New York, NY, 1998, pp. 137–149.
[9] Dominik Janzing, Is there a physically universal cellular automaton or Hamiltonian? arXiv e-prints, Sep 2010.
[10] Emmanuel Jeandel, Computability of the entropy of one-tape Turing machines, CoRR, arXiv:1302.1170 [abs], 2013.
[11] Jarkko Kari, Nicolas Ollinger, Periodicity and immortality in reversible computing, in: Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science, MFCS '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 419–430.
[12] Petr Kůrka, On topological dynamics of Turing machines, Theor. Comput. Sci. 174 (1–2) (1997) 203–216.
[13] Christopher G. Langton, Studying artificial life with cellular automata, in: Evolution, Games and Learning, Los Alamos, N.M., 1985, Physica D 22 (1–3) (1986) 120–149.
[14] Douglas Lind, Brian Marcus, An Introduction to Symbolic Dynamics and Coding, Cambridge University Press, Cambridge, 1995.
[15] D. Maldonado, A. Gajardo, B.H. De Menibus, A. Moreira, Nontrivial turmites are Turing-universal, J. Cell. Autom. 13 (5–6) (2018) 373–392.
[16] Reem Yassawi, Michael Baake, John A.G. Roberts, Reversing and extended symmetries of shift spaces, Discrete Contin. Dyn. Syst. 38 (2) (2018) 835–866.
[17] Cristopher Moore, Unpredictability and undecidability in dynamical systems, Phys. Rev. Lett. 64 (20) (May 1990) 2354–2357.
[18] Kenichi Morita, Theory of Reversible Computing, Springer, 2017.
[19] Lutz Priese, Automata and concurrency, Theor. Comput. Sci. 25 (3) (1983) 221–265.
[20] Ville Salo, Ilkka Törmä, A one-dimensional physically universal cellular automaton, in: Unveiling Dynamics and Complexity, in: Lecture Notes in Comput. Sci., vol. 10307, Springer, Cham, 2017, pp. 375–386.
[21] Luke Schaeffer, A physically universal cellular automaton, in: ITCS'15—Proceedings of the 6th Innovations in Theoretical Computer Science, ACM, New York, 2015, pp. 237–246.
[22] Luke Schaeffer, A physically universal quantum cellular automaton, in: Cellular Automata and Discrete Complex Systems, in: Lecture Notes in Comput. Sci., vol. 9099, Springer, Heidelberg, 2015, pp. 46–58.