

Almost Periodic Configurations on Linear Cellular Automata

K. Sutner

Computer Science Department

Carnegie Mellon University

Pittsburgh, PA 15213

sutner@cs.cmu.edu

Abstract. We study computational properties of linear cellular automata on configurations that differ from spatially periodic ones in only finitely many places. It is shown that the degree structure of the orbits of cellular automata is the same on these configurations as on the space of finite configurations. We also show that it is undecidable whether the cellular automaton exhibits complicated behavior on configurations of sufficiently long spatial periods and exhibit cellular automata with undecidable orbits whose orbits on backgrounds of all fixed sizes are decidable.

1. Computation in Cellular Automata

The study of computational properties of cellular automata on infinite grids typically focuses on so-called finite configurations, i.e., configurations where all but finitely many cells are in a special quiescent state, see for example [10, 11]. Hence, disregarding placement information, finite configurations can be construed as finite words and the machinery of classical computation theory can be brought to bear on the analysis of the behavior of cellular automata on this class of configurations. In particular, it is straightforward to express instantaneous descriptions of Turing machines as finite configurations of one-dimensional cellular automata so that the global rule of the cellular automaton corresponds to the one-step function of the Turing machine.

On the other hand, the quest for small, computationally universal systems naturally leads one to consider a slightly more general class of configurations, namely those that differ from a spatially periodic one in only finitely many places, see [13]. The importance of periodic backgrounds was emphasized in recent work by Wolfram, see [26], in particular chapter 11. The reference outlines a proof, due to Cook and Wolfram, of the universality of elementary cellular automaton rule 110. See also [14] for a careful and detailed description of the argument. The proof crucially depends on a periodic background pattern: on ordinary finite configurations it is trivially decidable whether a configuration evolves to another under rule 110. This is true for all elementary cellular automata, see also [3] for a related argument in two

dimensions. While it is intuitively appealing that the existence of a background pattern could cause very simple rules to exhibit universal behavior, it is still surprising that an elementary cellular automaton, a binary cellular automaton of width 3, should suffice. The local map of rule 110 is given by the Boolean function $\rho(x, y, z) = (\bar{x} \wedge y \wedge z) \oplus y \oplus z$, where \oplus denotes exclusive or. Alternatively, the local map can be construed to be the binary polynomial $\rho(x, y, z) = y + z - (1 + x)yz$.

We will consider the space of *almost periodic configurations*, configurations that differ from a two-way spatially periodic one of the form ${}^{\omega}uv^{\omega}$, the *background*, in only finitely many places. As we will argue in section 2, these configurations form the most natural setting for the study of computational properties of one-dimensional cellular automata, at least in the context of classical recursion theory. We denote the space of all almost periodic configurations by \mathcal{C}_{ap} , and write \mathcal{C} and \mathcal{C}_f for the space of all configurations and all finite configurations, respectively. Thus, $\mathcal{C}_f \subsetneq \mathcal{C}_{\text{ap}} \subsetneq \mathcal{C}$ assuming a non-trivial alphabet. Clearly, \mathcal{C}_{ap} is dense but fails to be closed \mathcal{C} in the topological sense. However, \mathcal{C}_{ap} is closed under application of the global map and also the inverse global map. The latter fact makes two-way periodic configurations a more natural choice than strictly spatially periodic ones of the form ${}^{\omega}u^{\omega}$. Two-way periodic backgrounds may also provide a better setting for very small universal machines where asymmetry may be required. Unlike with finite configurations the notion of quiescence is not relevant in this setting; in particular we will not insist that the underlying background configuration be a fixed point of the global map.

There are two basic approaches to measure the complexity of a cellular automaton. The first is inspired by computability theory: use some type of simulation to show that the cellular automaton has the same computational power as some other device such as a universal Turing machine. It is rather difficult to find convincing concepts of simulations in the realm of cellular automata, see [5, 7], in particular with a view towards input/output conventions. For example, depending on what conventions are adopted, small cellular automata may or may not be capable of solving the density problem, see [1] and the references therein. The central issue here is the output function: the global function of a cellular automaton is supposed to be iterated indefinitely whereas Turing machines or other models such as random access machines halt to signal the end of a computation. It is thus not clear where and when one is supposed to read off the result of a computation by a cellular automaton. Another complexity measure, inspired by methods in dynamics, is to categorize the collection of all orbits of the cellular automaton. One should mention that Davis suggested in [4] an orbit-based notion of universality for Turing machines: a Turing machine is Davis-universal if the collection of finite orbits is r.e. complete. A noteworthy feature of Davis' definition is that it bypasses the issue of simulation entirely. A now classic scheme for the classification of cellular automata was suggested by Wolfram in [25]. As is pointed out in [2], the Wolfram classification seems more appropriate for distinguishing behavior on a particular initial configuration, rather than a whole space of configurations.

Wolfram's approach is justified by visual inspection of the orbits of many cellular automata but it is not entirely clear how this classification should be formalized, see [11] for the first attempts in that direction. We will use a much more fine-grained classification depending on the collection of all orbits of the cellular automaton on the class of almost periodic configurations. More precisely, given a one-dimensional cellular automaton ρ define the *complete orbit* on \mathcal{C}_{ap} to be

$$\text{Orb}_{\rho} = \{ (X, \rho^t(X)) \mid X \in \mathcal{C}_{\text{ap}}, t \geq 0 \} \subseteq \mathcal{C}_{\text{ap}} \times \mathcal{C}_{\text{ap}}$$

Here the configurations are expressed in some natural coding, see section 2 for details. We adopt as our measure of the complexity of the cellular automaton the Turing degree of the complete orbit on almost

periodic configurations. Since these orbits are trivially recursively enumerable (r.e.) we can then classify cellular automata as follows. For any r.e. set W define

$$\mathbb{C}_W = \{ \rho \mid \text{Orb}_\rho \equiv_T W \}$$

to be the collection of all cellular automata whose complete orbit has the same Turing degree as W . Thus, \mathbb{C}_\emptyset is none other than the third Culik-Yu class, see [11], a formalization of Wolfram's heuristic class 3. No distinction is made at the lower levels of the hierarchy. It is shown in [24] that all classes \mathbb{C}_W are non-empty when finite configurations are under consideration rather than almost periodic ones. The same holds for the cumulative version of the hierarchy where we require $\text{Orb}_\rho \leq_T W$. Since finite configurations are but a special case of almost periodic ones, lower bound arguments for the complexity of orbits are easily inherited. However, upper bounds require separate arguments since it is a priori possible that the cellular automaton has more complicated behavior on the larger class of configurations. The Cook-Wolfram result implies that rule 110 belongs to \mathbb{C}_\emptyset .

Consider for the moment a strictly periodic configurations of the form ωu^ω . All orbits on such configurations are finite as sets and thus easily decidable in this setting. Indeed, we are essentially dealing with a finite cellular automaton on a circular grid. Unsurprisingly, the occurrence of a target configuration in the orbit of a given source configuration is PSPACE-complete in this case, so the orbits are far from trivial from the point of view of computational complexity. A more interesting question is how the size of the circular grid affects the complexity of the orbits. For example, we could ask if all orbits of strictly periodic configurations evolve to a fixed point. As is shown in [20] this decision problem is Π_1 -complete. Likewise there is no computable bound on the lengths of the limit cycles of all strictly periodic configurations of size n , uniformly for all n . We will show that a somewhat similar problem arises with almost periodic configurations: we cannot effectively determine whether the orbits of a given cellular automaton will be undecidable for almost periodic configurations of sufficiently large spatially periodic parts. Likewise one cannot determine effectively whether these orbits will be r.e. complete. The arguments verifying these claims produce cellular automata that have undecidable orbits for almost periodic configurations of some sufficiently large period. By contrast we construct a family of cellular automaton whose orbits on backgrounds of any particular fixed period are decidable. However, the decision procedures are non-uniform and if we consider periodic backgrounds of arbitrary size the orbits become undecidable.

Another area where background patterns of sorts play an important role is computational mechanics, an approach to the dynamics of cellular automata that is strongly inspired by physics, see [9] and the references therein. The main objective here is to identify so-called *domains*, parts of configurations that are periodic under the global map and that can be described in terms of certain finite state machines. More specifically, domains are the acceptance languages of Fischer automata, see [22] for a general discussion of the relationship between these machines and one-dimensional cellular automata. The interesting dynamics occur at domain walls, or defects, where different domains interact. Hence, a configuration of the form ωuv^ω with simple blocks u and v may be construed as having just a single defect, and we are concerned with the evolution of this defect. Filtering out parts of configurations that belong to domains can make it easier to identify mechanisms that support universal computations such as particles and their interactions. For example, for elementary cellular automaton 54, which is analyzed in great detail in the reference, the primary domain is essentially none other than the background $\omega 0111^\omega$. In this particularly simple situation the temporal period is 2 and the domain can easily be discovered by inspection. The authors present some heuristic methods to construct candidate domains, but in the light

of the results here it seems doubtful that general tools can be developed that would allow one to tackle more general one-dimensional cellular automata. For example, in general it is undecidable whether a defect constitutes a particle (i.e., has bounded spatial extension during its whole evolution), though the cellular automata used in the hardness arguments are quite complicated and artificial. It is conceivable that for elementary cellular automata such as rules 30 and 110 critical domains can be found and their interactions analyzed.

To avoid lengthy expositions of purely recursion theoretic material we refer the reader to the literature; good expositions can be found in [12] or [18]. Our notation is compatible with the last reference, so we will not repeat standard definitions that can be found there. In section 2 we will show how to transfer the simple decision procedures for reversibility and surjectivity to almost periodic configurations. We then establish the undecidability of the existence of sufficiently complicated backgrounds in section 3 and close with a few open problems in section 4.

2. Almost Periodic Backgrounds

For our purposes it suffices to think of a one-dimensional cellular automaton as a *local map* or *local rule* $\rho : \Sigma^w \rightarrow \Sigma$ where $w \geq 2$ is the *width* of the automaton (the case $w = 1$ is of no interest here) and Σ is the underlying alphabet. For simplicity assume that w is odd, so $w = 2r + 1$ where r is the *radius* of the local map. A *configuration* is any map $X : \mathbb{Z} \rightarrow \Sigma$. The local map naturally extends to a *global map* on the space of all configurations by

$$\rho(X)(i) = \rho(X(i-r), X(i-r+1), \dots, X(i+r-1), X(i+r)).$$

Here we abuse notation and denote both the global and the local map by ρ . A *local configuration* associated with X is any block $X(i-r)X(i-r+1) \dots X(i+r-1)X(i+r)$ construed as a word of length w over Σ . We refer to a symbol $\# \in \Sigma$ as a fixed point of the local map if $\rho(x_1, \dots, x_r, \#, x_{r+2}, \dots, x_w) = \#$ for all $x_i \in \Sigma$. It is a classical result by Curtis, Lyndon and Hedlund that precisely all the continuous shift-invariant functions on the space of all configurations arise as global maps of cellular automata.

Configurations suitable for computational studies have to have a finitary description, at least within the context of classical computability theory. The largest such class \mathcal{C}_{rec} comprises all configurations $X : \mathbb{Z} \rightarrow \Sigma$ given by recursive functions. However, it is undecidable in this setting whether a local configuration occurs in a given global configuration, so even one-step evolution produces undecidable problems. A more reasonable approach is to identify a class \mathcal{D} of configurations that has the following properties. First, all finite configurations and all strictly periodic ones belong to \mathcal{D} . Second, \mathcal{D} is closed under the application of all global maps ρ and also closed under inverse maps in the following sense.

Reflection Principle

If configuration $Y \in \mathcal{D}$ has a predecessor under ρ then there exists a predecessor in \mathcal{D} .

Thus, the dynamics of the cellular automaton are reasonably well reflected when we restrict the space of configurations to \mathcal{D} : given a configuration X in \mathcal{D} we can trace its orbit under any global map in both directions. We will show in a moment that the least such class of configurations consists of almost periodic ones.

Let us introduce a bit more notation. A superscript ω indicates an infinite or co-infinite word obtained by infinitely many repetitions of a finite word. Thus $u^\omega = uuu\dots$ and ${}^\omega u = \dots uuu$. We write $X = {}^* Y$

to indicate that configurations X and Y differ in only finitely many places. Clearly, $=^*$ is an equivalence relation and traditional finite configurations are simply the equivalence class of ${}^\omega 0^\omega$ where $0 \in \Sigma$ is assumed to be a particular quiescent state in the alphabet of the cellular automaton. We are interested in the equivalence classes of all two-way periodic backgrounds of the form ${}^\omega uv^\omega$. For the sake of clarity, we will always refer to backgrounds of the form ${}^\omega u^\omega$ as strictly periodic. For our purposes it suffices to consider only the case where $|u| = |v|$ since otherwise we can replace, say, u by $u^{\text{lcm}(|u|,|v|)/|u|}$. Denote $\mathcal{C}_{\text{ap}}^{(n)}$ the space of all configurations that differ from a background ${}^\omega uv^\omega$, $n = |u| = |v|$, in only finitely many places:

$$\mathcal{C}_{\text{ap}}^{(n)} = \{ X \mid X =^* {}^\omega uv^\omega, u, v \in \Sigma^n \}.$$

It is convenient to write $X = {}^\omega uvw^\omega$ for these configurations. Thus, an almost periodic configuration can be represented by a triple $(u, w, v) \in \Sigma^n \times \Sigma^* \times \Sigma^n$. The triple (u, w, v) is not uniquely determined by X but for us there is no need to constrain the definitions in order to insure uniqueness. All relevant operations such as computing the t th generation configuration $\rho^t(X)$ are primitive recursive, uniformly in ρ . Likewise, we can effectively construct a finite state machine that accepts all finite factors of configuration $\rho^t(X)$.

An issue that bears some scrutiny is placement information. Strictly speaking, any notation like ${}^\omega uvw^\omega$ only defines the class of configurations obtained by arbitrary shifts of any particular placement of the finite words and their repetitions. Configurations are functions $\mathbb{Z} \rightarrow \Sigma$ and thus have natural coordinates which can be exploited computationally. For example, the proof in [21] that recursive predecessors cannot be determined effectively uses very simple configurations of the form ${}^\omega aba^\omega$ where $a, b \in \Sigma$. It is the placement of b in a position that cannot be effectively determined that causes the problems in finding a predecessor. In the constructions below, though, these difficulties do not arise. All the relevant configurations contain a fixed point of the local map, and any query about the presence of a target configuration in the orbit of a source configuration is well-formed, even if the configurations are given in our abbreviated notation.

Arguable the most basic questions about the dynamics of a cellular automaton are reversibility and surjectivity of the global map. To this end, it is best to consider the associated *de Bruijn automaton* $\mathcal{B}(\rho)$ defined as follows. Suppose ρ uses alphabet Σ and has width $w \geq 2$. Consider the directed graph with vertex set Σ_k^{w-1} and edges $(ax, \rho(axb), xb)$ where $a, b \in \Sigma$ and $x \in \Sigma_k^{w-1}$. By labeling the edge (ax, xb) by $\rho(axb)$ we obtain a semiautomaton whose acceptance language is the collection of all finite factors of configurations of the form $\rho(X)$, $X \in \Sigma^\infty$. Alternatively one can think of $\mathcal{B}(\rho)$ as a transducer: every configuration X traces a unique path in $\mathcal{B}(\rho)$ and the biinfinite sequence of edge labels along that path is $\rho(X)$. The de Bruijn automaton $\mathcal{B}(30)$ of elementary CA number 30 is shown in figure 1. Our first application of de Bruijn graphs is to show that Reflection holds for almost periodic configuration. This claim follows immediately from lemma 2.1 but it is worth stating it separately.

Proposition 2.1. Reflection holds for \mathcal{C}_{ap} : For any one-dimensional cellular automaton ρ , if an almost periodic configuration has a predecessor under ρ , then it already has an almost periodic predecessor.

However, Reflection fails for strictly periodic backgrounds. As an example, consider elementary cellular automaton number 30 from figure 1. Here $X = {}^\omega 0110^\omega$ has two predecessors ${}^\omega 01^\omega$ and ${}^\omega 1010^\omega$ but no predecessor can be obtained from a strictly periodic configuration by a finite modification. On the other hand, Reflection does hold for \mathcal{C}_{rec} , the class of all configurations that are given by a computable function, see [21]. Note, though, that the existence of predecessors for recursive configurations

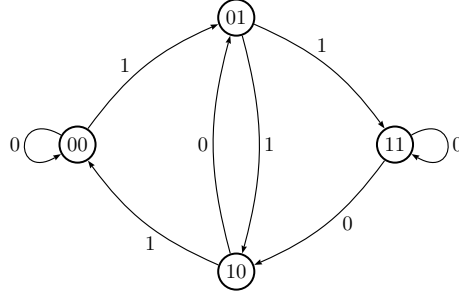


Figure 1. The de Bruijn transducer corresponding to ECA 30.

is undecidable, and even if a predecessor is known to exist a recursive predecessor cannot be constructed effectively. By contrast, testing for predecessors is straightforward for almost periodic target configurations, and one can effectively compute an almost periodic predecessor should one exist.

Lemma 2.1. It is polynomial time decidable whether an almost periodic configuration has a predecessor under a one-dimensional cellular automaton ρ . Moreover, if a predecessor exists then an almost periodic predecessor can be computed in polynomial time from the target configuration.

Proof:

Let $Y = {}^\omega u w v {}^\omega$ be a target configuration. To test whether a predecessor exists, consider the de Bruijn automaton $\mathcal{B} = \langle Q, \Sigma, \delta \rangle$ for ρ . Let \mathcal{A} be an automaton accepting v^+ . Construct the standard product automaton $\mathcal{B} \times \mathcal{A}$ and use it to determine the set Q_v of all states p such that $\langle Q, \Sigma, \delta, p, p \rangle$ accepts a word in v^+ . Likewise, we can determine the set Q_u of all states p such that $\langle Q, \Sigma, \delta, p, p \rangle$ accepts a word in u^+ . We claim that Y has a predecessor if, and only if, the automaton $\mathcal{A} = \langle Q, \Sigma, \delta, Q_u, Q_v \rangle$ accepts a word in $u^* w v^*$.

The direction from right to left is clear: if \mathcal{A} accepts $u^i w v^j$ then by the choice of Q_u and Q_v there is a bi-infinite path in \mathcal{B} labeled Y , see [21]. The fusion of the nodes on this path is a predecessor configuration for Y . For the opposite direction suppose that we have a predecessor X , which traces some bi-infinite path in \mathcal{B} labeled Y . Since \mathcal{B} is finite there must be some state p such that p appears on every final segment of the path. But then $p \in Q_v$. Similarly some state $q \in Q_u$ must appear in every initial segment of the path. One can select two suitable occurrences of q and p on the path for X such that the path segment between them is labeled $u^i w v^j$. But then \mathcal{A} accepts the latter word.

The size of the requisite finite state machines is quadratic in $|u w v| + k^w$ where k is the size of the underlying alphabet. All the needed algorithms such as acceptance testing are polynomial, and we are done. \square

We now show that the decision algorithms for injectivity and surjectivity for one-dimensional cellular automata for the space of all configurations carry over to our restricted setting. A careful description of fast algorithms for injectivity and surjectivity testing that exploit properties of the de Bruijn automaton can be found in [21].

Lemma 2.2. A cellular automaton is injective if, and only if, it is injective on almost periodic configurations. Likewise, a cellular automaton is surjective if, and only if, it is surjective on almost periodic configurations.

Proof:

It is easy to dispense with surjectivity. First, since \mathcal{C}_{ap} is dense in the space \mathcal{C} with the usual product topology, any cellular automaton whose global map is surjective on \mathcal{C}_{ap} must also be surjective on \mathcal{C} . The opposite direction follows from the Reflection Principle above.

As regards injectivity, clearly, any injective rule is also injective on almost periodic configurations. For the opposite direction suppose that ρ fails to be injective on the full space of all configurations. It is shown in [21] that in the product de Bruijn graph of ρ there must be a non-trivial strongly connected component other than the diagonal, either because the component of the diagonal contains points outside of the diagonal, or because there is some other component disjoint from the diagonal. In the second case, choose a loop in one such component. The loop gives rise to two distinct strictly periodic configurations that have the same image under ρ .

In the first case, we can establish a slightly stronger result. Choose any point p in the diagonal, and a loop labeled u that is anchored at p and does not lie entirely within the diagonal. Since the diagonal is isomorphic to a de Bruijn graph there is another loop labeled x inside the diagonal anchored at p . But then the path traced in the product graph by ${}^\omega x u x^\omega$ gives rise to two distinct almost periodic configurations with the same background whose image under ρ is the same. Using the strong transitivity of \mathcal{B} one can even insure that the background is strictly periodic. In either case, ρ is not injective on \mathcal{C}_{ap} . \square

Hence the algorithms in [21] can be applied directly to test surjectivity and injectivity over the restricted configuration space \mathcal{C}_{ap} .

Corollary 2.1. One can test for surjectivity and injectivity on \mathcal{C}_{ap} in quadratic time.

Quadratic time here refers to a uniform cost function and the natural presentation of the local map as a table of size k^w where k is the size of the alphabet and w the width of the rule.

3. Classification and Universality

The heuristic classification of cellular automata by Wolfram in [25] is based on the visual observation of geometric structures in the evolution of a configuration. From the point of view of a Turing degree based hierarchy Wolfram's classification is rather coarse: if one adopts the formalization of Culik and Yu in [11], the only distinction at the high end is between decidable orbits in class three and undecidable ones in class four. Using the Turing degree hierarchy as the basic measure of complexity we obtain a much richer structure. More precisely, if one considers only finite configurations the classes

$$\mathbb{C}_A^f = \{ \rho \mid \text{Orb}_\rho^f \equiv_T A \}$$

of automata whose orbit has a given r.e. degree are all non-empty, see [24]. The third Wolfram-Culik-Yu class is \mathbb{C}_\emptyset in this setting, and computationally complete cellular automata correspond to class $\mathbb{C}_{\emptyset'}$ where \emptyset' denotes the Turing jump of \emptyset , see [18]. While \mathbb{C}_A^f affords no direct comparison between cellular automata in terms of simulations, it does provide an indirect comparison. For example, if a cellular

automaton ρ simulates a universal Turing machine in the standard way, the orbit Orb_ρ^f will have Turing degree \emptyset' . Thus, if we have Orb_ρ^f as an oracle we can answer questions about the behavior of any cellular automaton on \mathcal{C}_{ap} since the complete orbit of any such cellular automaton is at most r.e. complete. Of course we are ignoring here the computational cost of the Turing reduction.

However, due to the complicated structure of \mathcal{R} , the upper semi-lattice of the r.e. degrees, there are many cellular automata with intermediate computational properties. For example, the existence of incomparable r.e. degrees implies that for some pairs of cellular automata complete knowledge of the orbit of the first cannot be used to predict the orbits of the other, and vice versa. By Sack's density theorem, for any two r.e. sets $A <_T B$ there exists a third r.e. set C such that $A <_T C <_T B$. Correspondingly there is a cellular automaton whose complete orbit has complexity strictly between two given ones of strictly increasing computational power. Indeed, any countable partially ordered set can be embedded in \mathcal{R} , so the situation is as complicated as possible: any consistent existential statement in the language of partial orders can be modeled in \mathcal{R} . As a consequence, the existential theory of \mathcal{R} is decidable. However, the full theory is undecidable, see [15, 8, 18].

We will now show that the transition from \mathcal{C}_f to \mathcal{C}_{ap} does not affect the degree structure of the orbits. In the context of hardness arguments finite configurations are somewhat easier to deal with than almost periodic ones since the quiescent state that is guaranteed to surround the central part of the configuration affords an end-marker. This end-marker can be used to make all unwanted configurations evolve to, say, a fixed point in finitely many steps. By compactness, no such mechanism exists in the space of the almost periodic configurations and one has to be somewhat more careful with upper bound arguments.

As in the finite configuration case, our arguments rely on a non-standard simulation of Turing machines by a cellular automaton. The reason for adopting a non-standard simulation is that we are here interested in questions regarding the space of all configurations (of a certain type) whereas classical computability theory is only concerned with the "evolution" of configurations of a very special type: only configurations that are reachable from certain initial instantaneous descriptions (IDs) corresponding to a specific input are relevant. Suppose M is some Turing machine and let I_x be the initial ID corresponding to input x , some string. Write $M^t(I)$ for the ID obtained by applying the one-step function of M t -times to I (assuming that no halting configuration has appeared before that). A plausible general notion of simulation for the Turing machine by some cellular automaton ρ is the following, see [5, 7] for a discussion of related notions. Encode IDs as configurations by means of some, say, injective recursive function $\gamma : \mathcal{I} \rightarrow \mathcal{C}_{\text{ap}}$ where \mathcal{I} denotes the collection of all IDs of M . We have to ensure that for all inputs x and all times t

$$\gamma(M^t(I_x)) = \rho^{f(t,x)}(\gamma(I_x)) \quad (1)$$

where f is some computable function. By observing the orbit of $\gamma(I_x)$ at certain computable moments one can thus determine the orbit of I_x under the next-step function of the Turing machine, and thus the computation of M on x . Clearly, this process selects only a small class of orbits of ρ , and it may well happen that the Turing machine performs only a simple recursive task, but the cellular automaton has very complicated orbits, albeit not on configurations corresponding to computations of M . In general it is Π_1 -complete to test whether an ID appears in an actual computation of the Turing machine, so these faulty IDs cannot simply be eliminated. Detailed descriptions of how to circumvent this problem can be found in [19, 24]. The first reference in particular contains a very careful treatment of coding issues.

We outline the construction and show how it can be adjusted to work with almost periodic configurations rather than just finite ones. In order to pin down the Turing degree of an orbit on \mathcal{C}_{ap} we first

consider configurations of the form

$$\dots 00 \$1 t \$2 x \$3 \hat{I} \$4 S \$5 00 \dots \tag{2}$$

where I is an ID of some Turing machine M to be simulated and \hat{I} its representation as part of a configuration. As usual, the ID is expressed as a string in $\Gamma^*Q\Gamma^*$ where Γ is the tape alphabet of the Turing machine and Q its state set (and we tacitly assume that these sets are disjoint). \hat{I} is then obtained by adding an orientation tag \rightarrow or \leftarrow to all the tape symbols (the symbols to the left of the tape head are labeled \rightarrow , all others \leftarrow). The fields x and t represent the alleged input and number of steps in the computation, respectively. Note that one can test primitively whether the initial ID I_x really leads to I in t steps. The scratch-space S is used for this verification process by the cellular automaton. If verification succeeds, the next ID is generated, t is incremented, and the verification begins anew. We may safely assume that an unbounded computation leads to a sequence of configurations whose $\$1$ and $\$5$ separators move unboundedly far to the left and right, respectively. $\$2$ and $\$3$ are fixed points of the local map, though. The behavior of the Turing machine on some input x is reflected in the orbit of the following configuration C_x under the global map of the cellular automaton:

$$C_x = \dots 00 \$1 \$2 x \$3 \hat{I}_x \$4 \$5 00 \dots \tag{3}$$

If a halting ID is reached, we hide the time-stamp and evolve to a halting configuration

$$H_x = \dots 00 \$1 \$2 x \$3 \$4 \$5 00 \dots \tag{4}$$

As already mentioned, placement information is not an issue here since these configuration contain fixed points of the local map.

Another standard technique already implicit in the direction tags of the coded IDs is the use of multiple tracks, i.e., of alphabets of the form $\Sigma \supseteq \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_k$. In our case, we use two tracks plus a special blocking symbol $\#$, a fixed point of the local map, so that the cellular automaton has alphabet $\Sigma = \Sigma_1 \times \Sigma_2 \cup \{\#\}$. The second alphabet effectively selects one particular cell, the center cell, by imposing an orientation on the symbols in Σ_1 . More precisely, $\Sigma_2 = \{\rightarrow, \leftarrow\} \cup L \cup R$ and the projection of a configuration on its second component is intended to be of the form $\omega \rightarrow (L \mid R) \leftarrow \omega$. Here L and R should be thought of as collections of particles, all moving to the left or right at unit speed, respectively, under the action of the global map. The positions of these particles are constrained to be somewhere between $\$1$ and $\$5$, inclusively. Thus, the particles bounce back and forth in a zigzag movement between the domain walls $\$1$ and $\$5$ where they are reflected. Changes in track one of the configuration can only occur when a particle moves past in track two. For example, in the situation shown below, the state p of the Turing machine and the surrounding symbols in the tape inscriptions can be modified since a right-moving particle r is passing by on its way to separator $\$5$.

...	\$3	a_1	a_2	a_3	p	a_4	a_5	a_6		...
	→	→	→	→	r	←	←	←		

(5)

The need for multiple particles arises from the fact that we have to communicate local events such as the occurrence of a halting state in the ID part to the remainder of the configuration. Multiple particles are also convenient to move the separator symbols other than $\$2$ and $\$3$ whenever more space is needed, as

well as during the contraction phase after a halting ID has appeared. We will refer to the first track as the computation track and the second track as the particle track.

Any configuration that locally looks like the one in (2) will be called *admissible*. In other words, the collection of all factors of length w of the configuration is a subset of the set of such factors in any configuration of type (2). If a configuration violates admissibility the blocking symbol is generated. Also, the blocking symbol destroys active cells that collide with it. If we disregard the top track a short fragment of an orbit might look like so.

$$\begin{array}{|c|c|c|c|c|} \hline \rightarrow & \rightarrow & r & \leftarrow & \# \\ \hline \rightarrow & \rightarrow & \rightarrow & r & \# \\ \hline \rightarrow & \rightarrow & \rightarrow & \rightarrow & \# \\ \hline \end{array} \tag{6}$$

Separator symbols trying to move into a cell occupied by $\#$ will disappear. Note that any admissible configuration has at most one particle. However, an admissible configuration may still differ globally from the intended form (2). For example, the whole configuration might have a direction sequence $\omega \leftarrow \omega$, in which case the configuration is already a fixed point of the global map.

It is clear that a local map $\rho = \rho(M)$ can be generated primitive recursively from M . We can now verify that the degree-based classification carries over from finite configurations to almost periodic ones. In the following, whenever we refer to an orbit $\text{orb}_\rho(X)$ of a cellular automaton as being finite, we mean that it is finite as a set; as a sequence it will, of course, always be infinite. For a Turing machine M , though, the orbit $\text{orb}_M(x)$ obtained by iterating the next-step function on ID I_x may indeed be finite.

Lemma 3.1. Simulation Lemma

For any r.e. set W there is a one-dimensional cellular automaton ρ whose complete orbit on \mathcal{C}_{ap} has the same Turing degree as W .

Proof:

Let M be a Turing machine accepting the r.e. set $W \subseteq \mathbb{N}$ and $\rho = \rho(M)$ the cellular automaton as described in the preceding paragraphs. To determine whether $x \in \mathbb{N}$ is an element of W consider C_x , the configuration representing the initial ID I_x . Ignoring direction tags, C_x has the form $\omega 0 \$1 \$2 x \$3 \hat{I}_x \$4 \$5 0^\omega$. From the definition of ρ we have that for any ID $I = M^t(I_x)$ there is a configuration $X = \omega 0 \$1 t \$2 x \$3 \hat{I} \$4 \$5 0^\omega$ in the orbit of X . But then M accepts x if, and only if, $\text{orb}_M(x)$ is finite and ends in a halting ID, if, and only if, the halting configuration H_x from (4) above appears in the orbit of X . It follows that $W \leq_T \text{Orb}_\rho$.

For the opposite direction consider two almost periodic configurations $X = \omega u w v^\omega$ and Y . Using W as oracle we have to determine whether Y occurs in the orbit of X . First consider the case when X fails to be admissible. Hence, in one step the blocking symbol $\#$ will be generated (and may also be already present in X). But then after finitely many steps all blocks coding computations as in (3) will either have reached form (4) or will have collided with a blocking symbol. In either case, all the active center cells will have disappeared. At this point the configuration becomes a fixed point of the global map. Hence the orbit of X is finite and we can determine if Y appears in the orbit by brute force enumeration.

So suppose that X is admissible but not as in (2). If there is no particle anywhere, X is already a fixed point and we are done. Otherwise, by admissibility, there must be exactly one particle. If this particle fails to appear between separator symbols $\$1$ and $\$5$ there must be an infinitely long block of

non-separator symbols. When the particle enters that block (perhaps after leaving it once) it will move in the same direction indefinitely since at least one domain wall is missing. The orbit of X is infinite in this case, but it is still trivial to check if Y occurs in it.

Lastly, consider the case when X has the intended form (2). We may safely assume that the verification of input x leading to I in t steps succeeds since otherwise a blocking symbol will be generated and we again have a finite orbit. Thus, the evolution of X under ρ indeed simulates the computation of M on input x . If $x \in W$ then the orbit $\text{orb}_\rho(X)$ is finite and we can test for the occurrence of Y . Otherwise the computation is divergent and we are dealing with an infinite orbit. However, for the configuration Y to appear in this orbit it must carry a time-stamp t between the $\$1$ and $\$2$ separators. Hence, given W as oracle, it is easy to check if Y appears in the orbit of X . \square

The notion of simulation used in the proof is fairly weak. First off, the delay between the action of the Turing machine and the cellular automaton is unbounded even for a single orbit. Second, there is no encoding γ as in (1). The problem is that an encoding of the form $\gamma : \mathcal{I} \rightarrow \mathcal{C}_{\text{ap}}$ would have to express an ID $I = M^t(I_x)$ by a time-stamped configuration $X = {}^\omega 0 \$1 t \$2 x \$3 \hat{I} \$4 \$5 0^\omega$, ignoring the particle track. While the appearance of I in $\text{orb}_M(I_x)$ is in general undecidable, the appearance of X is trivially decidable. However, since halting configurations carry no time-stamps we can salvage enough of a classical simulation to express acceptance of M in terms of the orbits of the cellular automaton. At any rate, as an immediate consequence of the lemma one can now establish the analogue of theorem 7 in [24].

Theorem 3.1. For any recursively enumerable set W the class \mathbb{C}_W is Σ_3^W -complete.

It follows in particular that it is Σ_3 -complete to test whether the orbits on \mathcal{C}_{ap} are decidable. To test for r.e. completeness is even Σ_4 -complete, so there is little hope for general methods to establish computational universality. The degree of the complete orbit of a cellular automaton formalizes the difficulty of the reachability problem for configurations of the automaton. In a similar fashion one can consider the confluence problem: given two configurations X_1 and X_2 , do their orbits overlap? In other words, do there exist times t_1 and t_2 such that $\rho^{t_1}(X_1) = \rho^{t_2}(X_2)$? The problem is certainly r.e., and one can show that one can construct cellular automata for which the degree of the confluence problem assumes any given r.e. degree, see [23] for the case of finite configurations. In fact, it is even possible to combine the constructions to obtain cellular automata whose reachability problem has some given r.e. degree, and whose confluence problem has another given r.e. degree.

These results refer to the space \mathcal{C}_{ap} of all almost periodic configurations. As rule 110 documents, it may well happen that the orbits on $\mathcal{C}_{\text{ap}}^{(1)}$ are trivial, at least in the sense of being decidable, but for some sufficiently large n the background of a configuration in $\mathcal{C}_{\text{ap}}^{(n)}$ can be used to encode enough support structure for the same cellular automaton to have undecidable orbits. As we will show, there is no algorithm to check whether such sufficiently complicated configurations exist for a given rule. It is instructive to see why orbits of rule 110 on $\mathcal{C}_{\text{ap}}^{(n)}$ are decidable for small n . For example, finite configurations grow at a steady rate of 1 to the left and have a stationary right end. In the terminology of computational mechanics, there are two main defects, one stationary and the other traveling at constant speed. Hence every target configuration in \mathcal{C}_f carries its implicit time-stamp and one can easily check if it appears after the appropriate number of steps. The same holds true for $\mathcal{C}_{\text{ap}}^{(n)}$ for $n = 1, 2, 3$ since all possible backgrounds here evolve to all-zeros in a few steps. But already for $n = 4$ backgrounds with

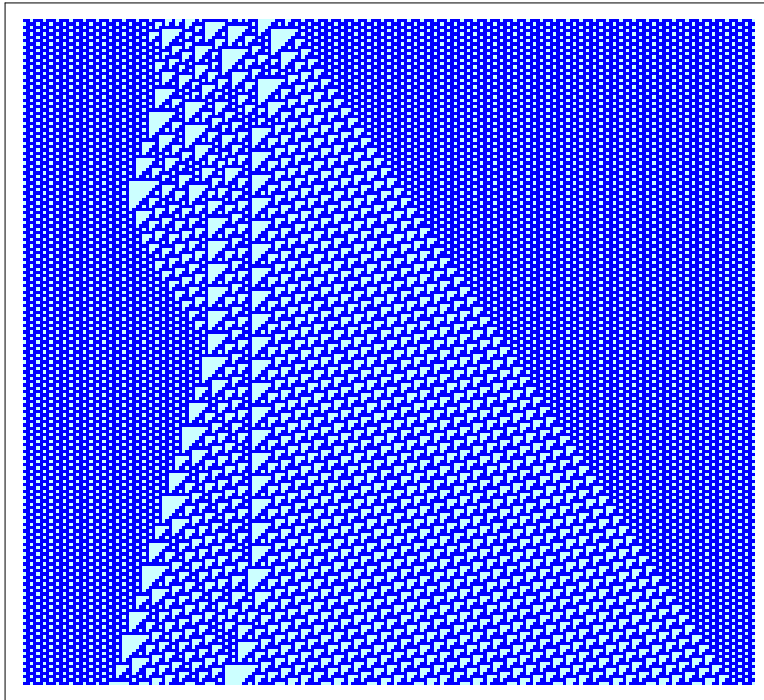


Figure 2. The orbit of an almost periodic configuration whose background is of the form ${}^{\omega}0111^{\omega}$. A block of 40 bits was altered at random.

non-trivial temporal period exist, and changing a few bits in such a background produces surprisingly complicated orbits. Figure 2 shows an example of the initial part of such an orbit. Note there still is an implicit time-stamp since the defect on the right hand side of the non-background pattern moves at uniform speed $2/3$. It is not hard to check that this is true for all backgrounds in $\mathcal{C}_{\text{ap}}^{(4)}$, so again we have a simple decision procedure. The propagation of the defect on the right hand side also shows that we are not dealing with a particle (which is supposed to be a bounded change in the background).

Of course, there is no reason why this type of ad hoc argument should carry over to larger values of n , and indeed the Cook-Wolfram result demonstrates that, for n on the order of several thousand, orbits in $\mathcal{C}_{\text{ap}}^{(n)}$ are no longer decidable. The next theorem shows that in general it is undecidable whether orbits on $\mathcal{C}_{\text{ap}}^{(n)}$ become complicated for some n .

Theorem 3.2. Given a one-dimensional cellular automaton ρ it is undecidable whether there exists an n such that ρ has non-recursive orbits on $\mathcal{C}_{\text{ap}}^{(n)}$.

Proof:

The construction uses four tracks in the configuration of the cellular automaton. There are two main tracks, each subdivided into a computation and a particle sub-track, as in the proof of lemma 3.1. We begin with a description of the second main track whose computation sub-track holds intended configurations of the form

$$\omega(\$_1 t \$_2 x \$_3 I \$_4 S \$_5)^\omega. \quad (7)$$

Thus, the second track is intended to perform computations of M in the sense of the Simulation Lemma. The admissible patterns in the particle track are now $\omega(\rightarrow^*(L \mid R) \leftarrow^*)^\omega$. Unlike with the previous construction, all the separation markers $\$_i$ are now fixed and cannot be moved. Thus, all the computations expressible by such configurations are space and thus time constrained. If the simulated machine accepts, the configuration is reset to $\$_1 \$_2 x \$_3 \widehat{I}_x \$_4 S \$_5$ where \widehat{I}_x is the ID corresponding to input x , and the computation starts anew. If, on the other hand, the simulation runs out of space a blocking cell $\#$ is generated (for all tracks, the cellular automaton uses an alphabet of the form $\Sigma_1 \times \Sigma_2 \times \Sigma_3 \times \Sigma_4 \cup \{\#\}$).

As before, admissibility in the sense of being locally indistinguishable from an intended configuration is but a local property and it may happen that, say, one of the time-stamps extends infinitely far to the left in a configuration in \mathcal{C}_{ap} . No blocking cell will be generated in this case. However, any left-moving particle entering this region will wander off to the left. As we will see, this will inhibit any useful computation in main track one.

The first track performs a simulation of a Turing machine in exactly the same fashion as described in the Simulation Lemma. The particle sub-track makes sure that there is at most one center cell except in orbits that lead to a blocking symbol. However, the movement of the particle in track one is constrained in the following way: a particle cell in track one can change position only when a particle in track two passes by. The change in position will be only one cell, and the the direction will be the same as that of the passing particle. A snapshot of such a configuration is shown below.

	x_1	x_2	x_3	x_4	p	x_5	x_6	x_7	
	→	→	→	→	r	←	←	←	
...	$\$1$	1	0	0	1	1	0	$\$2$...
	→	→	→	→	→	→	l	←	

Since the l particle in sub-track 4 is moving left it will pass by the r particle in sub-track 2 without interaction. Upon reflection at domain wall $\$1$ it will turn into an r' particle, and then interact with the one in sub-track 2 on the way back. Given two Turing machines M and M' we can thus construct a cellular automaton $\rho = \rho(M, M')$ and the construction is primitive recursive, uniformly in M and M' . Now suppose that M' is a machine accepting a non-recursive set W' , and M accepts W . We claim that W is not empty if, and only if, ρ has non-recursive orbits in $\mathcal{C}_{\text{ap}}^{(n)}$ for some n .

To see this, first suppose that $W \neq \emptyset$. Pick some element $a \in W_e$ and consider a block that codes the initial ID for the computation of M_e on input a , say $x = \$1 00 \dots 00 \$2 a \$3 \hat{I}_a \$4 S \$5$. Note that there has to be enough space between $\$1$ and $\$2$ and enough scratch-space between $\$4$ and $\$5$ to allow for a halting computation to be fully simulated. Now consider the configuration that has a strictly periodic background ${}^\omega x {}^\omega$ on track two. Hence we have infinitely many particles bouncing back and forth along the whole length of the configuration. Moreover, the particles will continue to move indefinitely. On the first track, the configuration will simulate computations of M' on some input b , so disregarding the particle track the configuration there will be of the form ${}^\omega 0B0 {}^\omega$. Together they form a configuration $X_{a,b}$ in $\mathcal{C}_{\text{ap}}^{(n)}$. Since the orbit of $X_{a,b}$ on the second track is periodic, any computation on track one will continue indefinitely until a halting ID is encountered, if indeed $b \in W'$. But then the orbits of ρ on $\mathcal{C}_{\text{ap}}^{(n)}$ fail to be decidable since otherwise we could check for the existence of a halting ID on track 1 in the orbit of $X_{a,b}$.

For the opposite direction suppose $W = \emptyset$ and consider an arbitrary configuration X in \mathcal{C}_{ap} . If the second track of X is inadmissible a blocking cell $\#$ will be generated. That blocking cell constrains the computation, if any, on track one and makes the orbits decidable. On the other hand, suppose the second track is admissible. Since X is almost periodic, it can code a finite number of computations of M_e . But since none of these computations is accepting, they will all lead to a blocking cell being generated. Hence the orbit of X is decidable. If the second track is admissible but does not contain both domain walls $\$1$ and $\$5$ no blocking cell will be created, but an infinite block of cells will have at most two occurrences of a particle in track two. Thus, the computation in track one will simply be frozen after finitely many steps, and the whole orbit will be decidable. \square

It is not difficult to modify the argument from the theorem to show that it remains undecidable whether orbits on $\mathcal{C}_{\text{ap}}^{(n)}$ are undecidable for some n , even given the fact that $\mathcal{C}_{\text{ap}}^{(m)}$ is decidable for all $m \leq n_0$. Furthermore, one can see that the construction in the last proof yields a slightly stronger result: it is undecidable whether the orbits of ρ on $\mathcal{C}_{\text{ap}}^{(n)}$ have any particular r.e. degree other than \emptyset . The argument also shows that one cannot decide whether a given local rule will be capable of generating orbits of maximum degree, given a background of sufficiently long spatial period.

Theorem 3.3. Given a one-dimensional cellular automaton it is undecidable whether there exists an n such that $C_{\text{ap}}^{(n)}$ has Σ_1 -complete orbits.

Proof:

The argument uses the same 4-track cellular automaton $\rho = \rho(e, e')$ as in the last theorem. Select $W_{e'}$ to be r.e.-complete. Then W_e fails to be empty if, and only if, the orbits of ρ on $C_{\text{ap}}^{(n)}$ are r.e. complete for some n . \square

Needless to say, a similar result holds for arbitrary r.e. degrees. The cellular automata constructed in the last two theorems have the property that the orbit of $C_{\text{ap}}^{(n)}$ are already complicated for some sufficiently large n . The question arises whether all these orbits might be decidable, but the complete orbit Orb_ρ still undecidable. In other words, can it happen that there is a decision procedure for $\text{Orb}_\rho \cap C_{\text{ap}}^{(n)} \times C_{\text{ap}}^{(n)}$ for all n , but there is no such procedure that is uniform in n .

Theorem 3.4. There is a cellular automaton whose complete orbit C_{ap} on almost periodic configurations is undecidable, but all the orbits on backgrounds of fixed size are decidable.

Proof:

Consider an arbitrary Turing machine M accepting $W \subseteq \mathbb{N}$. The construction of $\rho = \rho(M)$ is best described in terms of configurations using three tracks. The first two are computation and particle tracks, as in the previous arguments. It is convenient, though, to reorganize the configurations slightly, so that the computation track looks like so:

$$\dots 00 \$_1 S \$_2 \hat{I} \$_3 t \$_4 x \$ 00 \dots \quad (8)$$

The separator $\$$ here has a special role and will be referred to as the gate. Neither the gate nor $\$_4$ can change their position. The whole alphabet of the cellular automaton has the form $\Sigma_1 \times \Sigma_2 \times \Sigma_3 \cup \{\#\}$ where $\#$ is a fixed point of the local map. The third track is a signal track containing symbols from the input alphabet of the Turing machine plus a special synchronization symbol \diamond . On the third track, the global map simply acts like a left-shift. The intended configurations on the third track have the form ${}^\omega(\diamond y_1 y_2 \dots y_m)^\omega$ where the y_i are symbols from the tape alphabet and contain the input x as a scattered subsequence.

The two top tracks operate in the same way as in the previous arguments until a verification phase testing whether $I = M^t(I_x)$ is successfully completed. At this point, a special particle μ is sent to the gate and we enter a matching phase. As μ passes by the original input $x = x_1 x_2 \dots x_k$ of the Turing machine between $\$_4$ and $\$$ on the first track, it turns each symbol x_i into an active version x'_i , so alphabet Σ_1 contains two copies of the tape alphabet. The particle μ attaches itself to the gate, and remains there until a signal \diamond arrives from the right. When that happens, the first symbol y after \diamond passing through the gate is accompanied by a particle in track two. Symbol and particle keep moving at unit speed until it they arrive at the last cell before $\$_4$. At this point, y is compared to x_1 . If the symbols match, x'_1 is deactivated and turns back into x_1 and the particle in track two is reflected and returns to the gate. If a mismatch occurs the blocking symbol $\#$ is generated. When the return particle reaches the gate, the next signal y' passing through the gate is matched with x'_2 , and so on. The following scenario shows a step during the matching phase where $x = ababa$. The first symbol has already been matched, and the

second will be matched in two more steps. We have masked out the symbols in track three other than the one accompanied by the particle in track two.

	\$ ₄	<i>a</i>	<i>b</i> '	<i>a</i> '	<i>b</i> '	<i>a</i> '	\$	0	
...	→	→	→	→	<i>l</i>	←	<i>μ</i>	←	...
	—	—	—	—	<i>b</i>	—	—	—	

Should another \diamond arrive at any time during this process a blocking symbol will be generated. Likewise, no blank symbol can arrive from the right without generating $\#$. Otherwise, if all the signals match up with the corresponding x_i 's, the special particle μ is released and the action returns to computing the next ID, updating the time-stamp and so on in the computation track, all changes controlled by the particle track. During this phase, track three only performs left-shift operations (where all symbols at the gate turn into blanks). As usual, if a halting ID is reached we erase all information except for the input part x and the separators in track one and reach a halting configuration. Note that this halting configuration is a fixed point only in the two top tracks, in the signal track the configuration keeps being shifted to the left.

We claim that for any fixed n the orbits of any configuration $X = \omega_{wwwv\omega}$ in $\mathcal{C}_{\text{ap}}^{(n)}$ are decidable. As in the previous arguments we can focus on the case where no blocking symbol $\#$ appears in any configuration in $\text{orb}_\rho(X)$, otherwise decidability is straightforward. For a finite number of steps, depending on the size of the non-periodic part w , the number k of symbols in the input to the Turing machine can be arbitrarily large. However, after at most $|w|$ steps the signals in track three reaching the gate from the right are all from the right background pattern v . If v does not contain any \diamond 's then the gate will never open in the sense of accompanying passing signals with particles that allow for the matching process to proceed. Hence we are dealing with a finite and therefore decidable orbit. So suppose that v does contain at least one \diamond . Since we assume that no blocking symbol appears the distance between any \diamond that opens the gate and the next \diamond after that must be at least k and, in fact, quadratic in k . Thus, unless the computation finishes before the periodic background in track three reaches the gate, we must have $k < n$. It follows that the number of possible inputs for the Turing machine causing computations of unbounded length is finite. Hence knowledge of some initial segment σ of W suffices to answer all membership queries concerning orbits in $\mathcal{C}_{\text{ap}}^{(n)}$ for fixed n : configuration that involve computations on inputs whose membership in W cannot be determined from σ are easily dispensed with, and for computations dealing with critical inputs we can consult σ as an oracle. Since σ is finite, all orbits in $\mathcal{C}_{\text{ap}}^{(n)}$ are decidable.

On the other hand, σ cannot be computed effectively from W and one would expect Orb_ρ to be undecidable whenever W is so undecidable. To see this, note that for any x there is a background pattern in track three that will insure that all the matching phases succeed. The length of the pattern will be quadratic in $|x|$ and can certainly be computed primitive recursively from x . Hence we can effectively construct an initial configuration C_x and a halting configuration H_x , both in \mathcal{C}_{ap} , such that $H_x \in \text{orb}_\rho(C_x)$ if, and only if, $x \in W$. \square

One can think of the last construction as requiring the input x to the Turing machine as being contained infinitely often in the initial configuration, albeit in somewhat obfuscated form. Thus, within $\mathcal{C}_{\text{ap}}^{(n)}$, essentially only finitely many inputs can be properly represented, leading to decidability of these restricted orbits. It would be interesting to see if the last theorem can also be established using backgrounds that constrain the computations rather than the inputs.

4. Conclusion and Problems

The argument by Cook and Wolfram suggests that rule 110 is currently one of the simplest known computationally universal system, though a careful check of all the proof details is still required. The construction uses rather large blocks of cells to produce the appropriate background for the simulation of a tag system; it would be interesting to know which is the least value of n for which rule 110 has undecidable orbits on $\mathcal{C}_{\text{ap}}^{(n)}$. No doubt the implicit time-stamp argument from section 3 can be pushed further, but it is not clear how far.

A natural question at this point is whether other elementary cellular automata could also turn out to be universal, or, at the very least, have undecidable orbits in $\mathcal{C}_{\text{ap}}^{(n)}$ for sufficiently large n . A natural candidate is rule 30, whose behavior is sufficiently irregular to make it a source of pseudo-random bits. One should note that randomness is the source of a natural intermediate degree, albeit in the setting of Muchnik degrees, see [16]. It seems that there is little hope to tackle the question of the Turing degree of rule 30 on almost periodic configurations at this point. But perhaps a close analysis of the Cook-Wolfram argument will help to pinpoint the degree of rule 30 if it should turn out to be complete after all. Since defects and particles seem to play a fundamental role in the argument a careful study of rule 30 using the ideas in computational mechanics would probably be helpful in this regard.

Chapter 12 in Wolfram's [26], is dedicated to a proposed "principle of computational equivalence". The principle may be described as "almost all processes which are not obviously simple can be viewed as computations of equivalent sophistication," see pages 716–717. While this statement does not amount to a formal definition, it would seem to suggest that natural processes are constrained to just two types of complexity: decidable and r.e. complete. The absence of natural examples of intermediate degrees is well-known, and has led to work on alternative notions of reducibility, notions that do admit concrete examples of intermediate complexity, see [16]. But even in the classical setting of Turing degrees the principle of computational equivalence coexists uneasily with many results concerning the structure of \mathcal{R} , the upper semi-lattice of r.e. degrees. It remains to be seen how the fact that cellular automata, the standard example of simple systems exhibiting complicated behavior in Wolfram's book, generate orbits of arbitrary degree can be reconciled with the principle of computational equivalence. As a case in point, consider the standard construction of two incomparable r.e. sets A and B . The construction uses a finite injury priority argument to insure that $A \neq \{e\}^B$ and conversely $B \neq \{e\}^A$ for any index e . As a consequence, both sets A and B are necessarily of intermediate degree. However, as was shown by Soare [17], the join of A and B is complete, though both sets are low. One could thus argue that a cellular automaton (or any other device) performing the priority construction is actually computationally universal: the orbit of the computation would indeed allow to read off both sets rather than just one. Incidentally, an implementation of such a cellular automaton can be found in [26]. However, it is not clear how to associate universal computation with the construction of arbitrary intermediate degrees.

References

- [1] Baldwin, J. T.: Computation versus Simulation, www.math.uic.edu/~jbaldwin/pub/cafom.ps, July 2002.
- [2] Baldwin, J. T., Shelah, S.: On the Classifiability of Cellular Automata, *Theoretical Computer Science*, **230**(1-2), 2000, 117–129.

- [3] Codd, E. F.: *Cellular Automata*, ACM Monograph Series, Academic Press, 1968.
- [4] Davis, M.: *A note on universal Turing machines*, Princeton University Press, 1956, 167–175.
- [5] Delorme, M.: *An Introduction to Cellular Automata*, Vol. 460 of *Mathematics and Its Applications* [6], 1999, 5–49.
- [6] Delorme, M., Mazoyer, J.: *Cellular Automata: A Parallel Model*, vol. 460 of *Mathematics and Its Applications*, Kluwer Academic Publishers, 1999.
- [7] Durand, B., Róka, Z.: *The Game of Life: Universality Revisited*, Vol. 460 of *Mathematics and Its Applications* [6], 1999, 51–74.
- [8] Griffor, E. R., Ed.: *Handbook of Computability Theory*, vol. 140 of *Studies in Logic*, North-Holland, 1999.
- [9] Hanson, J. E., Cruchfield, J. P.: Computational Mechanics of Cellular Automata, An example, *Physica D*, **103**, 1997, 169–189.
- [10] II, K. C., Hurd, L. P., Yu, S.: Computation Theoretic Aspects of Cellular Automata, *Physica D*, **45**, 1990, 357–378.
- [11] II, K. C., Yu, S.: Undecidability of CA Classification Schemes, *Complex Systems*, **2**(2), 1988, 177–190.
- [12] Lerman, M.: *Degrees of Unsolvability*, Perspectives in Mathematical Logic, Springer Verlag, 1983.
- [13] Lindgren, C., Nordahl, M.: Universal Computation in Simple One Dimensional Cellular Automata, *Complex Systems*, **4**, 1990, 299–318.
- [14] Rahn, M.: Uuniversalität in Regel 110, <http://www.stud.uni-karlsruhe.de/~uyp0>, March 2003.
- [15] Shore, R. A.: *The recursively enumerable degrees*, chapter 6, Vol. 140 of Griffor [8], 1999, 169–197.
- [16] Simpson, S. G.: An extension of the recursively enumerable Turing degrees, <http://www.math.psu.edu/simpson/talks/ams0301>, 2003.
- [17] Soare, R. I.: The Friedberg-Muchnik theorem re-examined, *Canad. J. Math.*, **24**, 1972, 1070–1078.
- [18] Soare, R. I.: *Recursively Enumerable Sets and Degrees*, Perspectives in Mathematical Logic, Springer Verlag, 1987.
- [19] Sutner, K.: A note on Culik-Yu classes, *Complex Systems*, **3**(1), 1989, 107–115.
- [20] Sutner, K.: Classifying Circular Cellular Automata, *Physica D*, **45**(1–3), 1990, 386–395.
- [21] Sutner, K.: De Bruijn Graphs and linear cellular automata, *Complex Systems*, **5**(1), 1991, 19–30.
- [22] Sutner, K.: Linear cellular automata and Fischer automata, *Parallel Computing*, **23**(11), 1997, 1613–1634.
- [23] Sutner, K.: Cellular Automata and Intermediate Reachability Problems, *Fundamentae Informaticae*, **52**(1-3), 2002, 249–256.
- [24] Sutner, K.: Cellular Automata and Intermediate Degrees, *Theoretical Computer Science*, **296**, 2003, 365–375.
- [25] Wolfram, S.: Computation Theory of Cellular Automata, *Comm. Math. Physics*, **96**(1), 1984, 15–57.
- [26] Wolfram, S.: *A New Kind of Science*, Wolfram Media, 2002.