

Cellular Automata, Decidability and Phasespace

K. Sutner*

*Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
sutner@cs.cmu.edu*

Abstract. Cellular automata have rich computational properties and, at the same time, provide plausible models of physics-like computation. We study decidability issues in the phasespace of these automata, construed as automatic structures over infinite words. In dimension one, slightly more than the first order theory is decidable but the addition of an orbit predicate results in undecidability. We comment on connections between this “what you see is what you get” model and the lack of natural intermediate degrees.

1. Cellular Automata, Computation and Physics

The following observation concerning the relationship between the abstract theory of computation and its physical manifestations is attributed to David Deutsch:

The theory of computation has traditionally been studied almost entirely in the abstract, as a topic in pure mathematics. This is to miss the point of it. Computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.

Deutsch, of course, is a physicist and one might suspect a degree of professional bias in this statement. Still, he makes an important point: the kinds of computations that can be physically realized, at least in the context of some idealized model of physics, are not well represented by the purely mathematical theory of computation. The situation is made more complicated by the fact that Hilbert’s challenge

*Address for correspondence: Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213

to find an axiomatization of physics is still unanswered after more than a century. As early as 1920, Max Born pointed out a culture clash between physicists and mathematicians, see [6]. About the former he writes

Conscious of the infinite complexities of the phenomena with which he is confronted in every experiment, he resists the idea of considering a theory as something definitive. He therefore abhors the word “Axiom,” which in its usual usage evokes the idea of definitive truth.

Mathematicians, according to Born, are rather differently inclined:

The mathematician, on the contrary, has no business with factual phenomena, but rather with logical interrelations. In Hilbert’s language the axiomatic treatment of a discipline implies in no sense a definitive formulation of specific axioms as eternal truths, but rather the following methodological demand: specify the assumptions at the beginning of your deliberation, then stop for a moment and investigate whether or not these assumptions are partly superfluous or contradict each other.

This lack of a direct and strong connection has recently engendered an astonishing confusion about the possibility of “hypercomputation” or the ability to break through the “Turing barrier.” From the perspective of Deutsch’s abstract, pure mathematics, there is no question about the existence of hypercomputation: we only need to consider Kleene’s theory of hyperarithmetic sets to step outside the bounds of classical computability, see for example [32]. Yet no one would suggest that ω_1^{CK} , the least admissible ordinal larger than ω , has any relevance for realizable computation, see Martin Davis’ excellent article [13]. We are here also concerned with breaking through the Turing barrier, but in the opposite direction: are there computations that give rise to undecidability, yet are less complicated than the Halting Problem? There are well-known methods in recursion theory, so-called priority arguments, that allow one to construct recursively enumerable (r.e. for short) sets that fail to be recursive but at the same time also fail to be r.e.-complete, see [28, 32, 26]. Yet from the perspective of physically realized computation these constructions all seem to be universal: a universal Turing machine is required and, given a sufficiently encompassing view of the computation, r.e.-complete sets are produced. This problem is discussed in more detail by Wolfram in [45], see also section 4 below where we introduce a model where an observer watches a computational process and interprets certain aspects as the outcome.

Using an observer rather than the traditional input/output behavior of a computational system is inspired by physics-like computation as suggested by Deutsch. In the absence of an axiomatization of physics the question arises which models of computation bear sufficient resemblance to physics so as to be clearly realizable (at least if one disregards questions of resource constraints in terms of mass, time and energy) but, at the same time, have interesting computational properties. Following in the footsteps of Konrad Zuse’s idea of a “Rechnender Raum,” we suggest cellular automata as a good candidate for such models, see [46]. One might suspect that physical reality turns out to be quite a bit more complicated than a cellular automaton with a fixed underlying topological structure, but let us ignore this for the time being. We will also avoid discussions of reversibility and conservation properties that would be expected from more realistic models.

To further simplify matters, consider the one model of computation that settled the question of what constitutes computability, at least in the eyes of Kurt Gödel: Turing machines, introduced in the seminal

1936 paper [42]. By design, these devices are realizable in the physical sense, see for example [8] for a discussion of the relationship between physics and computability. One helpful feature of Turing machines is that they can easily be modeled by one-dimensional cellular automata, which devices are significantly less complicated from a computational perspective than their higher-dimensional analogues. Again, no claim is made that a one-dimensional cellular automaton is a plausible model of physics.

Even if one agrees that one-dimensional cellular automata are a plausible setting for the study of physics-like computation, there is a pronounced tension between the topological and dynamical systems approach to the study of these systems and the computability theory approach: the former focuses on uncountable spaces such as 2^ω whereas the latter employs small subspaces such as the set of configurations of finite support (*i.e.*, those configurations that differ in only finitely many places from a fixed symbol, usually taken to be 0). In a sense, computability theory on infinitary objects is a thankless subject. If one adheres to a finitistic perspective one winds up with approximation mechanisms such as Weihrauch's type-2 Turing machines. Alas, equality then becomes undecidable, a result that is at the very least counterintuitive, see [43]. On the other hand, adopting the approach of computation over algebraic structures as in [4] leads to a serious lack of realizability.

Another important issue is that computational universality should not require inordinately complicated systems. For example, Marvin Minsky found a universal 7-state 4-symbol Turing machine in the early 1960's, see [25]. Perhaps the most compelling recent result along these lines is Cook's theorem about the universality of elementary cellular automaton number 110, see [9]. This cellular automaton is completely specified by the ternary Boolean function

$$\rho(x, y, z) = \neg(x \wedge y \wedge z) \wedge (y \vee z)$$

which acts on a bi-infinite configuration $X = (X_i)_{i \in \mathbb{Z}}$ by $\rho(X)(i) = \rho(X_{i-1}, X_i, X_{i+1})$. For customary "finite configurations," meaning bi-infinite configurations with finite support, the behavior of this cellular automaton is trivial in the sense that it is easily decidable whether one configuration occurs in the orbit of another. However, even this simple, Boolean radius 1 cellular automaton can perform universal computation provided one admits more general configurations. The configurations needed in Cook's argument have a very simple finitary description in terms of three finite words u , w and v and have the form ${}^\omega u w v {}^\omega$: there are infinitely many copies of u to the left, a finite central part w , followed by infinitely many copies of v to the right. We refer to these configurations as *almost periodic* or as *backgrounds*. Note that the commonly studied finite configurations are a special case of almost periodic configurations: they are of the form ${}^\omega 0 w 0 {}^\omega$ where 0 is some special element of the underlying alphabet (which is often assumed to be quiescent: local configurations of all 0's evolve to 0). It is our contention that the space of almost periodic configurations is a natural environment in which to study the computational properties of one-dimensional cellular automata as a first step towards exploring the recursion theoretic consequences of a physics-like model of computation.

Specifically, we will show in section 2 that almost periodic configurations have good model theoretic properties compared to the uncountable space of all bi-infinite configurations. In section 3 we discuss a few of the computational aspects of cellular automata operating on this class of configurations. Section 4 contains some comments on how this model might be helpful in getting a better understanding of the lack of natural intermediate degrees, a phenomenon most radically expressed in Wolfram's Principle of Computational Equivalence. Lastly, section 5 contains a few open problems.

2. Backgrounds and ζ -Automaticity

Since early work by Gilman, Cannon, Holt, Thurston and others on automatic groups, see [14], structures that can be described by finite state machines have received considerable attention, see in particular the work by Khossainov and Nerode [21, 22] and generalizations in [5]. Automatic structures over (finite) words are relational first order structures of the form

$$\mathfrak{A} = \langle A, R_1, \dots, R_n \rangle$$

where $A \subseteq \Sigma^*$ is a regular language and all the relations R_i are rational: there are finite state machines that decide these relations. For example, for a binary relation R the input to the corresponding finite state machine \mathcal{A}_R is the *convolution* $x:y$ of two words over Σ . Formally, $x:y$ is a word over the 2-track alphabet $\Sigma' \times \Sigma'$ where $\Sigma' = \Sigma \cup \{\#\}$ and $\#$ is a special padding symbol. The padding symbol is necessary to deal with words of different lengths.

The central property of automatic structures is that their first order theories are decidable. Indeed, a stronger property holds: for any first order formula $\varphi(x_1, \dots, x_k)$ with k free variables as indicated, one can effectively construct a finite state machine \mathcal{A}_φ that accepts precisely the convolutions of those words that satisfy the formula:

$$\mathcal{L}(\mathcal{A}_\varphi) = \{ u_1:u_2:\dots:u_k \in \Gamma^* \mid \mathfrak{A} \models \varphi(u_1, u_2, \dots, u_k) \}$$

where $\Gamma = \Sigma'^k$.

Decidability carries over to automatically presented structures that are obtained by a mediating surjective map $\nu : D \rightarrow A$ where D is a regular set of words and A an arbitrary set. However, the construction of witnesses becomes more complicated. Another generalization is the use of regular sets of ω -words as pioneered by Büchi in the 1960's.

For our purposes we need to consider yet another generalization: machines operating on the set of bi-infinite words Σ^ζ . To this end, define a ζ -Büchi automaton \mathcal{A} to be a modification of a standard Büchi automaton as follows. \mathcal{A} consists of a finite, nondeterministic transition system $\langle Q, \Sigma, \tau \rangle$ and an acceptance condition (I, F) where $I, F \subseteq Q$. Given a bi-infinite word $x \in \Sigma^\zeta$, a run of \mathcal{A} on x is a bi-infinite sequence $\pi = (p_i)_{i \in \mathbb{Z}}$ of states in Q such that $\tau(p_i, x_i, p_{i+1})$ for all $i \in \mathbb{Z}$. Let $\text{Inf}^-(\pi) = \{ p \in Q \mid \exists i \in \mathbb{N} (p_{-i} = p) \}$ and $\text{Inf}^+(\pi) = \{ p \in Q \mid \exists i \in \mathbb{N} (p_i = p) \}$ denote the set of negatively and positively recurrent states in π . Then \mathcal{A} accepts x if there exists some run π on x such that $\text{Inf}^-(\pi) \cap I \neq \emptyset$ and $\text{Inf}^+(\pi) \cap F \neq \emptyset$. A set of bi-infinite words is *regular* if there is some ζ -Büchi automaton that accepts it. If the underlying alphabet has the form Σ^k we obtain recognizable k -ary relations on Σ^ζ . For example, for a binary relation we use a 2-track alphabet so that the automaton reads words of the form

$$x:y = \begin{array}{|c|c|c|c|c|c|c|} \hline \dots & x_{-2} & x_{-1} & x_0 & x_1 & x_2 & \dots \\ \hline \dots & y_{-2} & y_{-1} & y_0 & y_1 & y_2 & \dots \\ \hline \end{array}$$

the *convolution* of $x, y \in \Sigma^\zeta$. This is somewhat easier than the finite word case since no padding is required. On the other hand, unlike with finite or one-way infinite words, the choice of coordinates here is somewhat arbitrary. In fact it is easy to see that two words $u, v \in \Sigma^\zeta$ that are shifts of each other cannot be distinguished by any ζ -Büchi automaton. Indistinguishability also arises if the the two

words are recurrent (every finite subword that appears anywhere already appears bi-infinitely often) and have the same cover (*i.e.*, the same set of finite factors). These are the only circumstances under which indistinguishability can arise, see [27]. The reference also has additional background on ζ -Büchi automata and contains alternative characterizations of recognizable languages of bi-infinite words.

A ζ -automatic structure is a first order structure

$$\mathfrak{A} = \langle A, R_1, \dots, R_n \rangle$$

where $A \subseteq \Sigma^\zeta$ is recognizable and all the relations R_i are also recognizable. For example, A might be the whole space Σ^ζ or some subspace such as configurations of finite support or almost periodic configurations. We will only deal with relational structures here, functions can always be represented by their graphs. Again, we are not interested in automatically presented structures: our main concern are structures that are obtained from one-dimensional cellular automata. More precisely, given by a local map ρ we consider

$$\mathfrak{C}_\rho = \langle \Sigma^\zeta, \rightarrow \rangle$$

where Σ^ζ is the space of configurations of the system and \rightarrow is a binary predicate that expresses the application of the global map G_ρ . This relational representation is slightly more convenient for our purposes. Note that it also easily accommodates nondeterministic rules. Without loss of generality we consider only standard local rules of the form $\rho : \Sigma^{2r+1} \rightarrow \Sigma$.

One can obtain a representation of a recognizable ζ -language L in terms of pairs of recognizable ω -languages:

$$L = \bigcup U_i^{\text{op}} V_i$$

where U^{op} stands for the co- ω -words obtained from U by reversing all the words. More precisely, for any ω -word $x : \mathbb{N} \rightarrow \Sigma$ define the co- ω -word $x^{\text{op}} : \mathbb{Z}^- \rightarrow \Sigma$, where \mathbb{Z}^- denotes the negative integers, by $x^{\text{op}}(-(i+1)) = x(i)$ for $i \geq 0$. Clearly one can join a co- ω -word x^{op} obtained by reversal with an ω -word y to produce a ζ -word $x^{\text{op}}y$. In the decomposition of L the constituent parts U_i and V_i can be handled by suitable Büchi automata. The reader should consult [27] and [10] for a more detailed discussion of this decomposition. As a consequence, one can exploit classical algorithms for ω -languages to deal with recognizable ζ -languages. In particular the determinization algorithm for ω -languages due to Safra, see [29] can be adapted to determinize ζ -automata. A rather detailed description of the automata theoretic machinery involved in the decision algorithm for first order logic on structures over bi-infinite words can be found in [34]. The basic construction relies on variants of de Bruijn automata to express the next step relation. For example, the two-track automaton that checks \rightarrow for elementary cellular automaton 150 is shown in figure 1. The states contain three bits $ab:d$ and there is a transition $ab:d \xrightarrow{c:e} bc:e$ iff $d = a + b + c \pmod{2} = \rho_{150}(a, b, c)$. Thus there is a bi-infinite run of this automaton on a 2-track input word $X:Y$ iff for all $i \in \mathbb{Z}$ we have $\rho_{150}(X_{i-1}, X_i, X_{i+1}) = Y_i$.

At any rate, as in the classical case of word or ω -automatic structures, we have the following theorem.

Theorem 2.1. The first order theory of ζ -automatic structures is decidable.

As O. Finkel points out decidability can be pushed a bit further; one can extend the language by quantifiers for “there exist infinitely many” and “there exist $r \pmod{k}$ many,” see [15]. Another extension

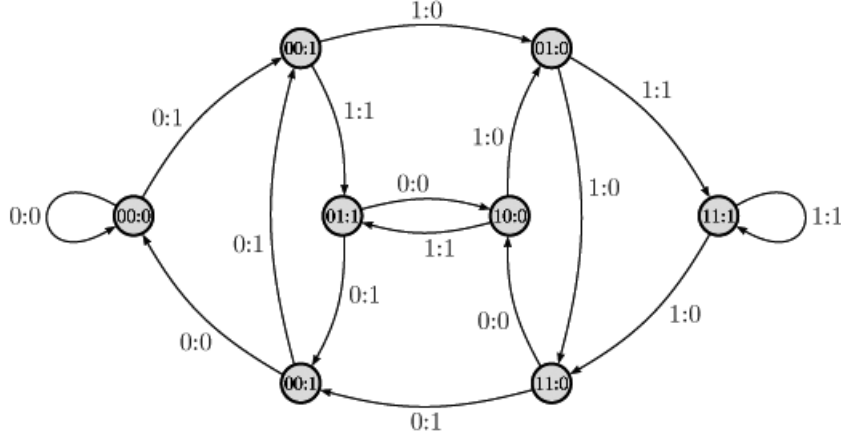


Figure 1. A two-track Büchi automaton checking one step in the evolution of elementary cellular automaton number 150.

comes from the use of additional predicates to the structures:

$$\mathfrak{C}_\rho = \langle \Sigma^\zeta, \rightarrow, \dots \rangle$$

As long as the additional predicates are all rational decidability is preserved. If the predicates are themselves not first order definable we obtain a stronger language. However, even definable predicates may be of interest since they can be used to simplify the assertions one wishes to verify and thus improve the efficiency of the decision algorithm.

As an example for the use of enlarged structures consider the basic properties of injectivity, openness and surjectivity of the global map of a cellular automaton. It is well known that injectivity implies openness which in turn implies surjectivity; the opposite implications are both false, see [18]. Injectivity and surjectivity of the global map are easily definable in \mathfrak{C}_ρ and are thus decidable, a fact first established by Amoroso and Patt [2] using combinatorial methods. The canonical definition for injectivity is the Π_1 sentence

$$\forall x, y, z (x \rightarrow z \wedge y \rightarrow z \Rightarrow x = y)$$

Negating this formula the decision procedure involves three projections (*i.e.*, the removal of one of the tracks and corresponding transition labels) on the automaton representing the negation of the matrix of the formula, followed by a simple path existence test. For surjectivity the situation is slightly more complicated: the natural definition

$$\forall y \exists x (x \rightarrow y)$$

is Π_2 . The two-track automaton that represents the predicate $x \rightarrow y$ is easily constructed, see figure 1 and the existential quantifier is dealt with by projection. Alas, the resulting machine is nondeterministic and checking for universality, while easily decidable, is computationally difficult. By compactness it suffices to deal with finite words only in this case, but universality testing is PSPACE-complete even in this situation. Note that it was shown in [38] that the de Bruijn automata associated with cellular automata do exhibit exponential blow-up during determinization, so any head-on approach is unlikely to succeed.

However, in the presence of a predicate “almost equal” we can express surjectivity of the global map by a Π_1 statement:

$$\forall x, y, z (x \rightarrow z \wedge y \rightarrow z \wedge x \stackrel{*}{=} y \Rightarrow x = y)$$

Here $x \stackrel{*}{=} y$ means that x and y differ in only finitely many places. The last formula can be dealt with in essentially the same way as the injectivity formula from above.

As an example of a property that fails to be definable in the original structure \mathfrak{C}_ρ , but can be defined in an automatic extension, consider openness of the global map in the topological sense. It is well-known that the global map is open if, and only if, it is k -to-one for some fixed k , see [18]. Of course, for a specific value of k we can express the assertion that the global map is k -to-one by a suitable Π_3 sentence but our language is too weak to quantify over k . Now consider the predicate $x =_L y$ if $\exists n \in \mathbb{Z} \forall i < n (x_i = y_i)$ and define $x =_R y$ in a symmetric way. Thus $x \stackrel{*}{=} y \iff x =_L y \wedge x =_R y$. Clearly both relations are automatic. But then the global map is open if, and only if,

$$\forall x, y, z (x \rightarrow z \wedge y \rightarrow z \wedge (x =_L y \vee x =_R y) \Rightarrow x = y)$$

Thus openness is definable over the extended structure $\mathfrak{C}_\rho = \langle \Sigma^\zeta, \rightarrow, =_L, =_R \rangle$ and thus decidable. As before, careful inspection of the automaton will yield a quadratic time algorithm.

In fact, there is a uniform quadratic time algorithm that classifies a one-dimensional cellular automaton according to injectivity, openness and surjectivity that requires no more than a product machine construction followed by the computation of strongly connected components. To see this note that the negation of the formulae for the three properties can be written

$$\exists x, y (\exists z (x \rightarrow z \wedge y \rightarrow z) \wedge P(x, y) \wedge x \neq y)$$

The part quantified by z can easily be represented by a 2-track de Bruijn automaton \mathcal{A} similar to the one in Figure 1 with states $u:v$ where $u, v \in \Sigma^{w-1}$ and transitions $u:v \xrightarrow{a:b} u'a:v'b$ iff $\rho(ua) = \rho(vb)$. Here u' denotes the word obtained by deleting the first letter in u and likewise for v' . Thus, we are looking for the existence of a bi-infinite path in \mathcal{A} . The last condition, $x \neq y$, means that this path must not lie entirely within the diagonal $\Delta = \{u:u \mid u \in \Sigma^{w-1}\}$. Thus, the global map of a cellular automaton is injective iff the only non-trivial strongly connected component in \mathcal{A} is Δ . It is open iff the strongly connected component containing Δ is Δ and there are no paths from Δ to any other non-trivial strongly connected component. Lastly, it is surjective iff the strongly connected component containing Δ is Δ . Of course, the standard decision algorithm for automatic structures fails to take advantage of these computational shortcuts. A direct derivation of the quadratic time algorithm for injectivity and surjectivity can be found in [37].

Write Σ_{ap}^ζ for the collection of almost periodic words over alphabet Σ . For any ζ -automatic structure \mathfrak{A} with carrier set Σ^ζ define \mathfrak{A}_{ap} to be the substructure obtained by restriction to Σ_{ap}^ζ . Note that \mathfrak{A}_{ap} is countable. It was noted in [39] that a cellular automaton on Σ^ζ is injective respectively surjective if and only if it has the same properties on Σ_{ap}^ζ . As it turns out, this is just the tip of an iceberg. Recall that an elementary substructure \mathfrak{A} of \mathfrak{B} is a substructure that has the property that for any first order formula $\varphi(\vec{x})$ and any tuple \vec{a} of elements in \mathfrak{A} we have $\mathfrak{A} \models \varphi(\vec{a})$ iff $\mathfrak{B} \models \varphi(\vec{a})$. For example, for $\varphi(x) = \exists z (z \rightarrow x)$ this would mean that any point in \mathfrak{A} that has a predecessor in \mathfrak{B} will also have a predecessor in \mathfrak{A} .

Theorem 2.2. For any ζ -automatic structure \mathfrak{A} the restriction \mathfrak{A}_{ap} is an elementary substructure.

Proof:

We apply the standard Tarski-Vaught test, see [7]. To this end consider a first order formula $\exists x \varphi(x, a_1, \dots, a_n)$ where all the a_i are in $\Sigma_{\text{ap}}^{\zeta}$ and $\mathfrak{A} \models \exists x \varphi(x, a_1, \dots, a_n)$. For simplicity assume $n = 1$, so we need to show that $\mathfrak{A}_{\text{ap}} \models \exists x \varphi(x, a)$.

Suppose π is an accepting computation of the 2-track automaton \mathcal{A}_φ on $x:a$. Then for some initial state p and some final state q we have $p \in \text{Inf}^-(\pi)$ and $q \in \text{Inf}^+(\pi)$. We focus on the argument for q , the other case is entirely similar. Some final segment of π must lie in a strongly connected component of the diagram of \mathcal{A} , and it touches q infinitely often. Since the component is finite and a is ultimately periodic there must be a loop on π that is anchored at q and such that the input on the second track on the loop is a repetition of a conjugate of the periodic part of a . But then we can modify π to have a final segment of ω -many copies of this loop without affecting acceptance. Performing the same surgery also at p we obtain a background x' on the first track and we are done. \square

Write $\mathfrak{C}_{\rho, \text{ap}}$ for the associated structure where the carrier set is \mathcal{C}_{ap} . As an immediate corollary to the preceding results we have

Theorem 2.3. The first order theory of \mathfrak{C}_ρ is decidable and coincides with the first order theory of $\mathfrak{C}_{\rho, \text{ap}}$.

Thus local properties of the full phasespace of a one-dimensional cellular automaton can be studied by restriction to almost periodic configurations; moreover all these properties are decidable in principle. Alas, it should be noted that the decision algorithm obtained by exploiting automaticity fails to be elementary recursive. There are two major obstructions to obtaining computational results: one is the exponential growth of the basic automata when dealing with, say, chains $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{n-1} \rightarrow x_n$. These chains correspond to conjunctions of basic predicates and are handled via computationally expensive repeated product constructions. The other major problem is the complexity of negation: determinization of ζ -automata is based on Safra's notorious algorithm and thus super-exponential in general. Even in very limited circumstances it is challenging to obtain computational results. As a case in point consider *nilpotency*: a cellular automaton is nilpotent if there exists a quiescent configuration y such that all configurations evolve to y in finitely many steps. By compactness, there has to be a fixed bound n , the nilpotency index, such that

$$\exists y \text{ quiesc. } \forall x \exists x_1, \dots, x_{n-1} (x \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{n-1} \rightarrow y)$$

The first quantifier can easily be replaced by a disjunction over all possible quiescent configurations. While quiescence is not first order definable over our structures it is straightforward to add a corresponding unary predicate. After projecting away the existentially quantified tracks we are left with a universality problem on a subautomaton of the standard de Bruijn automaton for the iterated cellular automaton. Alas, the automaton has exponential size and even ordinary Rabin-Scott determinization becomes problematic. Efforts are under way to employ anti-chain methods for universality testing but efficiency remains a substantial problem.

3. The Orbit Relation

While the first order theory of \mathfrak{C}_ρ is decidable, stronger logics such as monadic second order logic can be expected to be undecidable. For simplicity, let us stay with first order logic and enlarge the structure

by the critical *reachability* (or *orbit*) relation $x \xrightarrow{*} y$ meaning that y lies in the orbit of x . We write \mathfrak{C}_ρ^* for the structure $\langle \Sigma^\zeta, \rightarrow, \xrightarrow{*} \rangle$.

Theorem 3.1. The first order theory of \mathfrak{C}_ρ^* is undecidable in general.

Note that our language lacks constants, so we cannot simply argue by expressing the Halting Problem for Turing machine in terms of a simulation by cellular automata. However, the existence of a universal attracting fixed point is expressible over this structure by a Σ_2 sentence:

$$\exists y \forall x (x \xrightarrow{*} y \wedge y \rightarrow y).$$

This corresponds to nilpotency: the limit set of the cellular automaton consists of just a single configuration. Since nilpotency was shown to be undecidable by Kari [20] theorem 3.1 follows. Undecidability also appears in effective subspaces such as the space of almost periodic configurations and the space of finite configurations. More precisely, the Σ_2 sentence:

$$\forall x \exists y (x \xrightarrow{*} y \wedge y \rightarrow y).$$

which asserts that all configurations evolve to a fixed point is undecidable. This follows from the results in [36] using the methods detailed below in the proof of theorem 3.2 to deal with arbitrary almost periodic configurations rather than just finite ones. In fact, it is Π_2 -complete to test this property for a given cellular automaton. A similar result also holds for the space of spatially periodic configurations ${}^\omega w^\omega$: in this case the complexity drops to Σ_1 -complete since the existential quantifier is effectively bounded, but the problem remains undecidable.

For a more careful analysis of decidability properties of reachability let us consider the effective subspace of almost periodic configurations rather than the whole uncountable space. Thus we are interested in structures $\mathfrak{C}_{\rho, \text{ap}}^* = \langle \Sigma_{\text{ap}}^\zeta, \rightarrow, \xrightarrow{*} \rangle$. It is easy to see that reachability in this structure is r.e. We claim that the Turing degree of the reachability relation on Σ_{ap}^ζ can be any r.e. degree. The proof hinges on a tightly controlled simulation of Turing machines by a cellular automaton. However, there are several technical challenges to overcome. First, instantaneous descriptions of some Turing machine M are naturally translated into almost periodic configurations but the execution of the Turing machine only makes sense for a rather narrow subclass. For example, we need to contend with configurations corresponding Turing machine instantaneous descriptions that involve multiple heads operating on the same tape. Or a configuration may correspond to multiple instantaneous descriptions, separated by blank symbols. Second, even if a configuration is well-formed it may not correspond to an instantaneous description of the Turing machine that could actually occur during any computation of the machine: all computations of M start at special initial configurations I_x corresponding to some particular input $x \in \mathbb{N}$ to the Turing machine. But then the simulating system \mathfrak{C}_M may contain very complicated orbits even though the computations performed by the Turing machine are entirely trivial. For example, M may simply erase its input and accept, but it could contain a copy of a universal Turing machine whose states are not reachable from the initial state of M . The halting set of M is then trivial, but the degree of the orbits of \mathfrak{C}_M is \emptyset' , the degree of the Halting Problem.

To deal with this issue, let us refer to an instantaneous description D as *accessible* if it occurs during the computation of M on some input x . Accessibility is undecidable in general, but one can modify the Turing machine so that inaccessible instantaneous description do not artificially inflate the Turing degree

of the orbits of \mathfrak{C}_M . Call a Turing machine *stable* if it halts on all its inaccessible configurations. The following lemma can be found in [35, 39, 40]. Similar problems in connection with register machines and Turing machines and an application to Thue systems are discussed in [11] and [30].

Lemma 3.1. For every Turing machine M there is a stable Turing machine M_S that accepts the same set of inputs as M . As a consequence, every r.e. degree appears as the acceptance language of a stable Turing machine.

There is considerable freedom in organizing the structure of a stable Turing machine. For the sake of definiteness, let us assume that M_S operates on tape inscriptions of the form

$${}^\omega \underline{b} \# x \# s \# D \# t \# E \# \underline{b} {}^\omega$$

when written as bi-infinite words. Here x , s and t are numbers written in unary and D and E are instantaneous descriptions of M , the original machine. The symbol \underline{b} is the blank symbol for M_S and $\#$ is a separator symbol; neither symbol occurs in the alphabet used for the remainder of the inscription.

Let us call a tape inscription of this form *correct* if indeed x is the input of a computation of M such that the corresponding instantaneous description obtained after s steps is D . Correctness in this sense is decidable by a brute-force simulation: Using x one resets the field t to 0 and E to the initial ID of M with input x . One then increments s and updates E correspondingly until $s = t$. If correctness has been successfully verified, s is incremented, D is updated and the verification process is repeated. Thus M_S alternates between simulating a computation of M and verifying that it is indeed performing such a simulation. For technical reasons it is convenient that the separators $\#$ can only move to the right, the scratch space for t and E must be filled with an auxiliary blank symbol when it is not needed.

If the verification of an instantaneous description of M_S fails the tape is erased except for a single symbol $\#$ and the machine halts. If M performs a divergent computation on x this verify-compute process will continue forever. Note that in this case the tape inscription must increase unboundedly in size, growing to the right (the portion containing x remains static). If, after finitely many steps, M is found to accept input x the simulating machine M' erases its tape and halts. Thus, the ultimate configuration has the form ${}^\omega \underline{b} {}^\omega$.

The second and final step in the construction of \mathfrak{C}_M is to simulate a stable Turing machine by a cellular automaton operating over the space of almost periodic configurations. As usual, the cellular automaton will have radius 1 and its alphabet has the form

$$\Gamma = \underline{\Sigma} \cup Q \cup \overline{\Sigma} \cup \{\perp\}$$

where Σ is the tape alphabet of the stable Turing machine and Q its state set. The variants $\underline{\Sigma}$ and $\overline{\Sigma}$ are obtained by adding an extra direction bit to Σ (and are assumed to be disjoint from Q). Symbol \perp is a special annihilator state of the cellular automaton: for any local configuration that does not appear as a length 3 factor of an instantaneous description of M_S evolves to \perp at the next step. In particular local configurations containing \perp evolve to \perp . The direction bits indicate where a cell containing a symbol from Q might be found, so in orbits corresponding to actual computations we are dealing with configurations in ${}^\omega \underline{\Sigma} Q \overline{\Sigma} {}^\omega$.

In view of the lack of an intrinsic coordinate system for bi-infinite words let us agree that an almost periodic word $X = {}^\omega u w v {}^\omega$ is given more precisely by a finite presentation $(u, w, v, s) \in (\Gamma^*)^3 \times \mathbb{Z}$ to

be interpreted as follows. Define the configuration $X_{u,w,v} : \mathbb{Z} \rightarrow \Sigma$ by

$$X_{u,w,v}(i) = \begin{cases} u_{i \bmod |u|} & \text{if } i < 0, \\ w_i & \text{if } 0 \leq i < |w|, \\ v_{(i-|w|) \bmod |v|} & \text{if } i \geq |w| \end{cases}$$

and let $X = \sigma^s(X_{u,w,v})$ where σ denotes the shift operation. Here we assume that the finite words u, w, v are given in 0-indexed notation so that, for example, $u = u_0u_1 \dots u_{|u|-1}$. Note that this representation of an almost periodic configuration is by no means unique but arguably fairly natural. For example, it is decidable whether the configuration represented by (u, w, v, s) is the same as the configuration represented by (u', w', v', s') . Likewise, we can determine the semi-linear set of all positions of a particular symbol in X and so forth. Ignoring the shift factor we can define a canonical representation of almost periodic bi-infinite words as follows. First, the center part w is chosen to be of minimal length such that for some u and v we have $X = {}^\omega u w v {}^\omega$. Second, u and v are also chosen to be of minimal length. More precisely, any ω -word x^ω can be written uniquely in the form x_0^ω where x_0 is the root of x ; the same holds for the periodic co- ω -word on the left. The canonical representation $\text{rep}(X)$ of X is then the word $u\$w\v where $\$$ is separator symbol that does not occur in the alphabet of X . We will ignore coding details from now on and simply assert that certain properties of these configurations are decidable.

There is no notion of halting in the context of the system \mathfrak{C}_M simulating the stable Turing machine that in turn simulates a given machine M ; instead we only have the whole orbit of a given configuration under the global map. This makes it difficult in general to model terminating computations and the corresponding input/output behavior. However, for our purposes it suffices to pinpoint the complexity of the reachability relation as an r.e. set, termination and output are not directly relevant in the context.

Define ρ to be the cellular automaton simulating M_S as described in the preceding paragraph and write Orb_ρ for the orbit relation of ρ on almost periodic configurations. It is clear that these orbits include in particular all actual computations of the original machine M in coded form, so we should expect the complexity of Orb_ρ to be at least that of the acceptance language of M . However, there are several other types of orbits that need to be dealt with separately in order to pin down the complexity at the level of M .

For simplicity let us say that symbol \perp occurs in the orbit of a configuration X if there is some configuration Y in the orbit of X such that $Y(i) = \perp$ for some position $i \in \mathbb{Z}$. The orbit of X is \perp -free otherwise.

Proposition 3.1. It is decidable whether the orbit of an almost periodic configuration $X = {}^\omega u w v {}^\omega$ contains \perp . Moreover, if the orbit of X contains \perp then it is decidable whether another almost periodic configuration Y appears in the orbit of X .

Proof:

We may safely assume that \perp does not appear in $X = {}^\omega u w v {}^\omega$. First consider the case where X contains no symbol from Q . There will be at least one occurrence of \perp after one step unless $X \in \underline{\Sigma}^\zeta$ or $X \in \underline{\Sigma}^\zeta$; in which case X is a fixed point. On the other hand suppose X contains at least two symbols from Q , say $X = \dots p \dots q \dots$. Unless p is adjacent to symbols from $\underline{\Sigma}$ on the right and q is adjacent to symbols from $\underline{\Sigma}$ on the left, \perp will appear after one step. Thus there have to be adjacent symbols from $\underline{\Sigma}$ and $\underline{\Sigma}$ between p and q , again leading to the occurrence of \perp after one step. Lastly, consider the case where X

contains exactly one occurrence of a symbol from Q ; by necessity this occurrence must be in the central part w of X . If \perp does not occur in one step we must have $u \in \Sigma^*$, $v \in \Sigma^*$ and $w \in \Sigma^* Q \Sigma^*$ and maintains this form for the remainder of the orbit. In this critical case the orbit of X corresponds to a computation of M_S , albeit not necessarily on an admissible tape inscription. Nonetheless, the orbit is \perp -free in this case. Clearly all these conditions can easily be checked given a presentation of X in terms of three finite words u , w and v .

As to decidability, let Y be an arbitrary almost periodic configuration and assume that the orbit of X contains \perp . Then we can enumerate an initial segment of the orbit until we obtain a configuration X' whose set of positions of occurrences of \perp has the form \mathbb{Z} , $\{i \in \mathbb{Z} \mid i \geq n_0\}$, $\{i \in \mathbb{Z} \mid i \leq n_1\}$ or $\{i \in \mathbb{Z} \mid n_0 \leq i \leq n_1\}$. This follows from the fact that the central part of X cannot grow as quickly to the right as a block of \perp symbols. If Y does not occur in this initial segment but still occurs in the orbit of X it must be of the same type, in which case it is easy to determine occurrence. \square

We can now show that the orbit relations on almost periodic backgrounds are as complicated as they can possibly be: every r.e. degree appears as the degree of some cellular automaton.

Theorem 3.2. (Degree Theorem)

For any r.e. degree \mathbf{d} there is a one-dimensional cellular automaton operating on almost periodic configurations whose orbit relation has Turing degree \mathbf{d} .

Proof:

Let M be a Turing machine accepting an r.e. set $W \subseteq \mathbb{N}$ of degree \mathbf{d} and let M_S be the stable version of M as above. Write Orb for the orbit relation of ρ on almost periodic configurations where ρ is the simulating cellular automaton. We claim that $W \leq_T \text{Orb}$.

To see this let $n \in W$ and consider the M_S inscription

$$\omega \underline{b} \# n \# \# I_n \# \# \# \underline{b} \omega$$

where I_n is the initial tape inscription of M on input n . In this case M_S will successfully verify that I_n indeed corresponds to the inscription reached after 0 steps of the computation of M on input n . It will then simulate the computation of M on n and ultimately erase its tape and halt, say, in state q_H . Now consider the cellular automaton on the almost periodic configuration $X = \omega \underline{b} x \underline{b} \omega$ where x is the encoding of the central part of the instantaneous description of M_S as above. Iterating the global map we obtain an orbit of X that ends in the fixed point $\omega \underline{b} q_H \underline{b} \omega$. Clearly the orbit of X will not reach this fixed point unless M accepts n , so W is indeed Turing reducible to Orb .

It remains to show that $\text{Orb} \leq_T W$. To this end consider two almost periodic configurations X and Y . In general these configurations will not correspond to meaningful tape inscription of M_S , but as we have seen in proposition 3.1, it is decidable whether the orbit of X contains \perp . Moreover, in this case we can decide whether Y appears in the orbit. Thus we can safely assume that the orbit of X and Y are \perp -free. The cases where $X \in \Sigma^\zeta$ or $X \in \Sigma^\zeta$ are easily handled since X is a fixed point under the global map G_ρ . As in the proposition, X must contain exactly one symbol p from Q . But since X does not evolve to \perp the only way this can happen is when $X \in \omega \Sigma q \Sigma \omega$. Now consider the almost periodic decomposition of X as $X = \omega u w v \omega$ where q must be contained in the center part w .

By construction, the cellular automaton now simulates M_S on the corresponding tape inscription. If M_S ever enters and successfully completes a verification phase we can associate the orbit with an actual

computation of M on some input n . But then we can use W as an oracle to determine whether M accepts n and thus whether M_S ultimately halts on an empty tape. If so, the orbit of X is finite as a set and we can search for Y by brute force. Otherwise the orbit is infinite but the center block carries a time stamp: recall that the tape inscriptions of M_S explicitly contain the number of steps in the computation of M on n . For Y to appear in the orbit of X it would need to carry a similar time stamp which makes it easy to decide whether Y does indeed appear in the orbit of X . The argument in the case where the verification fails is somewhat easier: we do not need to consult the oracle W .

Since we are dealing with almost periodic rather than finite configurations it may happen that M_S never enters a verification phase, or enters but never completes it. For example, v might be the symbol 1 used in the unary representations of numbers, and the head of M_S may move to the right indefinitely, searching for the next separator symbol. In this case no simulation of a computation of M is performed, but it is easy to decide whether Y appears in the orbit of X .

Overall, the orbit relation is decidable in W for the orbits corresponding to actual computations of M and decidable otherwise. This shows that $\text{Orb} \leq_T W$ and we are done. \square

Using similar techniques and the method from [33] in the context of finite configurations one can establish the following result that combines restrictions on the complexity of the Reachability Problem with similar restrictions on the complexity of the closely related Confluence Problem. The Confluence Problem is to determine whether the orbits of two given configurations overlap.

Theorem 3.3. (Two Degrees Theorem)

Given any two recursively enumerable degrees \mathbf{d}_1 and \mathbf{d}_2 there is a one-dimensional cellular automaton ρ on the space of almost periodic configurations such that the Reachability Problem for ρ has degree \mathbf{d}_1 , and the Confluence Problem for ρ has degree \mathbf{d}_2 .

It follows that cellular automata operating on almost periodic configurations have the same rich computational structure as on configurations with finite support. We can now classify cellular automata according to the complexity of their orbit relation on almost periodic configurations: Given an r.e. degree \mathbf{d} , define

$$\mathbb{C}_{\mathbf{d}} = \{ \rho \mid \text{Orb}_{\rho} \equiv_T \mathbf{d} \}.$$

As we have just seen, all these classes are non-empty. For example, \mathbb{C}_{\emptyset} is comprised of all automata whose orbits are decidable. This hierarchy is somewhat coarse at the bottom since it fails to distinguish between cellular automaton where all configurations evolve to a fixed point or to a limit cycle and more complicated automata in class \mathbb{C}_{\emptyset} such as additive ones. However, it seems to reflect the complexity of these discrete dynamical systems better at the high end: there are cellular automata whose orbits are undecidable yet not as complicated as the Halting Problem. Indeed, if we compare the complexity of cellular automata by the degrees of their orbit relations we obtain a structure of similar complexity as the upper semi-lattice of r.e. sets, see [32] and [23]. For example, every countable partial order can be embedded into this semi-lattice and, as an algebraic structure, its theory is highly undecidable [17].

The question arises how difficult it is to determine membership in a particular class of the degree hierarchy for a given cellular automaton. Yates' index theorem was used in [40] to study this question in the context of finite configurations. The argument relies on the fact that the construction of the simulating cellular automaton is primitive recursive; it is not hard to check that the arguments carry over to almost periodic configurations with minor modifications as in the proof of theorem 3.2 above.

Theorem 3.4. Let \mathbf{d} be an arbitrary recursively enumerable degree. Then class $\mathbb{C}_{\mathbf{d}}$ is $\Sigma_3^{\mathbf{d}}$ -complete.

At the low end of the hierarchy, membership in class \mathbb{C}_{\emptyset} is thus already Σ_3 -complete and equivalent to testing membership in $\text{REC} = \{e \mid W_e \equiv_T \emptyset\}$, the class of all decidable r.e. sets. Here (W_e) denotes a standard enumeration of r.e. sets and \equiv_T denotes Turing equivalence. At the high end things are worse: it is Σ_4 -complete to test membership in $\mathbb{C}_{\emptyset'}$ which is as hard as testing membership in $\text{COMP} = \{e \mid W_e \equiv_T \emptyset'\}$, the class of all complete r.e. sets. As elementary cellular automaton number 110 demonstrates even specific instances of this decision problem can be very hard to deal with. As a case in point consider rule number 30 which provides a reasonable source for pseudo-random bits, see [44]. It is entirely unclear at this point whether rule 30 is computationally universal on almost periodic configurations or whether it belongs to \mathbb{C}_{\emptyset} . The lack of persistent structures in the evolution of configurations under rule 30 would seem to make both upper and lower bound arguments exceedingly difficult.

4. Computational Processes and Almost Periodic Configurations

In his book [45], Wolfram suggests that "... all processes, whether they are produced by human effort or occur spontaneously in nature, can be viewed as computations." This assertion is hardly controversial if one can agree on a rather loose definition of what constitutes a computation. However, Wolfram goes on to make a much more radical claim, his so-called *Principle of Computational Equivalence* (PCE, for short): "... almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication." Of course, on the face of it this would seem to contradict the existence of intermediate degrees and, more generally, ignore the complexity of the whole upper semi-lattice of r.e. degrees. Indeed, in the reference there is mention of an "abstruse" theorem of mathematical logic, the Friedberg-Muchnik theorem that establishes the existence of intermediate degrees. Wolfram's position is essentially this: the construction used in the proof of the Friedberg-Muchnik theorem, a so-called finite injury priority method, relies quite heavily on the use of a universal Turing machine. Thus, if one considers the whole construction rather than the designated output one would indeed conclude that the "computational process" behind Friedberg-Muchnik is complete. One can make this suggestion technically precise by noting that the standard construction produces two r.e. sets A and B that are both of intermediate degree. Alas, as shown by Soare [31] their disjoint union $A \oplus B$ is indeed complete. Similar arguments apply to other known constructions, see [26]. One has to willfully ignore a major part of the generated data to avoid completeness.

Indeed, no "natural" examples of intermediate degrees are known. For example, it is not known what happens to the Turing degree of the intermediate set if the underlying universal Turing machine is replaced by another one. As M. Davis [12] points out: "But one can be quite precise in stating that no one has produced an intermediate r.e. degree about which it can be said that it is the degree of a decision problem that had been previously studied and named." Or consider the following comment by Ambos-Spies and Fejer in [1]: "The sets constructed by the priority method to solve Post's Problem have as their only purpose to be a solution. ... Thus it can be said that the great complexity in the structure of the computably enumerable degrees arises solely from studying unnatural problems."

We suggest that one-dimensional cellular automata on almost periodic configurations provide a good environment to explore a more formal version of PCE. First, they are sufficiently close to physics to produce a credible model of a computational process: computation is realized as the evolution of a con-

figuration. The update operation is local and homogeneous and operates on a simple, discrete structure. Specifically, we are dealing with almost periodic configurations that can be represented by three finite words plus a shift factor. We can think of the periodic part as some kind of background that provides additional mass and energy whenever the computation needs to expand but this interpretation is independent of the computational properties of the system. While higher-dimensional cellular automata are arguably closer to actual physics our key concern is the ability to simulate Turing machines, and for those dimension one suffices. Indeed, as Cook [9] has shown, even exceedingly simple Boolean cellular automata can already give rise to computational universality in this setting.

Other than the ability to simulate universal Turing machines, the critical property of a cellular automaton operating on almost periodic configurations is that no information can be hidden from an observer who is given access to the representations of all the configurations in a particular orbit. Most importantly, the observer is not constrained to drawing conclusions solely for the input/output behavior of the system (however output may be defined in the context of a non-terminating system).

With this in mind, let us define a *computational process* to be a pair $P = \langle \rho, X \rangle$ where ρ is a one-dimensional cellular automaton operating on almost periodic configurations over some alphabet Σ and $X = {}^\omega w w v {}^\omega$ is such a configuration. We refer to ρ as the *computer* and to X as the initial configuration of P . The orbit $(X_t)_{t \geq 0}$ of X under ρ is the associated computation. An *observer* is now defined to be a word function τ on $\Sigma \cup \{\$ \}$ that is computable in constant space. We associate the *observation language*

$$O_\tau = \bigcup \{ \tau(\text{rep}(X_t)) \mid t \geq 0 \}$$

with the observer. Recall that $\text{rep}(X) = u\$w\v is the canonical representation of X , ignoring the shift factor.

It is critical here that the observer is constrained in its computational power so as to make sure that it cannot perform an independent computation producing results that have no connection to the given orbit. In fact, τ needs to be memoryless, otherwise a trivial computational process that simply converts ${}^\omega 01^k 0 {}^\omega$ to ${}^\omega 01^{k+1} 0 {}^\omega$ in one step would already allow for r.e.-complete observation languages when the observer is, say, a linear bounded automaton.

The complexity of a computational process can now be classified according to the existence of complicated observation languages or the lack thereof. Specifically, a computational process is *undecidable* if there exists an observer whose observation language is undecidable; it is *complete* if there exists an observer whose observation language is r.e.-complete. Of course, there always exist observers whose languages are trivial. With a view towards PCE, define a process to be *intermediate* if it is undecidable but fails to be complete. Thus, an intermediate computational process admits at least one observer that finds the process undecidable but prohibits any observer from extracting a complete r.e. set from it.

We can now rephrase Wolfram's Principle of Computational Equivalence as the assertion that intermediate processes do not exist. To obtain an intermediate process the obvious approach is to rephrase one of the standard priority constructions of intermediate r.e. sets in recursion theory [28, 32, 26] in the context of computational processes. These constructions all produce an r.e. set that can be shown to be undecidable while, at the same, not containing enough information to serve as an oracle for the Halting Problem. Alas, from the perspective of computational processes the desired set corresponds to just one observation and it seems that currently available constructions are far too weak to be translated into an intermediate computational process: other observations do in fact produce complete sets.

Here is a brief description of the arguably most basic existence argument for an intermediate r.e. set.

The construction produces two sets S and A such that the following conditions are satisfied

$$\begin{aligned} \mathcal{P}_e : \quad & W_e \text{ infinite} \Rightarrow S \cap W_e \neq \emptyset \\ \mathcal{N}_e : \quad & A \not\equiv \{e\}^S \end{aligned}$$

for all $e \in \mathbb{N}$. Here sets are identified with their characteristic functions so that $A \not\equiv \{e\}^S$ means that the set A differs from the set computed by the e th partial recursive function with oracle S . In other words, A is not Turing reducible to S via function number e . Condition \mathcal{P}_e demands that if the e th r.e. set W_e is infinite it must have non-empty intersection with S . These conditions are usually referred to as *requirements*, the purpose of the construction is make sure that all the requirements are met. $S \subseteq \mathbb{N}$ is the actual goal of the construction and A is an auxiliary r.e. set that witnesses the fact that S is incomplete. The two sets are constructed in stages $s < \omega$ and the actions taken at each stage are easily seen to be primitive recursive uniformly in s , so the sets are indeed r.e., see [32, 26] for details.

Note that the requirements are not effective in the sense that we cannot decide whether, say, W_e is infinite; we have to make do with approximations and make sure that the conditions are ultimately met. Another key problem with our family of requirements is that they can be mutually contradictory: one requirement may suggest to place an element into S whereas another makes it necessary to keep the element out of S . Since we need to build r.e. sets the construction is strictly monotonic; no elements can be removed at a later stage to correct an error. The central idea to deal with clashing requirements is to order them into an ω -sequence $\mathcal{P}_0 < \mathcal{N}_0 < \mathcal{P}_1 < \mathcal{N}_1 < \mathcal{P}_2 < \dots$ where lower rank means higher priority. At any stage during the construction, we work only on the requirement of highest priority that currently fails to be satisfied and that can be addressed at this stage. The details are somewhat complicated and require a great deal of care. However, at each level the construction is primitive recursive and a simple induction argument shows that in the end all requirements are ultimately satisfied.

It remains to translate these priority arguments into a computational process. Of course, one can apply all the standard primitive recursive coding tricks to translate the construction into arithmetic which can then be performed by a cellular automaton. Alternatively we can think of the construction as taking place in a simple model of set theory (without infinity). Perhaps the most natural environment of this type is the structure $\langle \mathbb{HFF}, \in \rangle$ of hereditarily finite sets, see [3]. It is well-known that a subset of \mathbb{N} is r.e. if, and only if, it is Σ_1 -definable over \mathbb{HFF} . The priority construction itself can be expressed by a suitable Σ_1 formula of set theory so that

$$n \in S \iff \mathbb{HFF} \models \exists x \Phi(x, \underline{n}).$$

Here \underline{n} stands for the von Neumann ordinal representing $n \in \mathbb{N}$. At stage s of the construction we truncate \mathbb{HFF} to \mathbb{HFF}_s , the collection of all hereditarily finite sets of rank at most s and obtain a finite approximation to S : all n such that $\mathbb{HFF}_s \models \exists x \Phi(x, \underline{n})$.

Once the priority argument has been cast in terms of a Σ_1 predicate over \mathbb{HFF} we can translate the structure of hereditarily finite sets into, say, non-negative integers that can be handled by a one-dimensional cellular automaton. For example, we can use Ackermann's classical coding function from \mathbb{HFF} to the naturals. More precisely, let $\alpha : \mathbb{HFF} \rightarrow \mathbb{N}$ where $\alpha(\emptyset) = 0$ and $\alpha(y) = \sum_{x \in y} 2^{\alpha(x)}$. Then the basic predicate $x \in y$ is easily decidable by calculating the binary expansion of $\alpha(y)$ and testing for the presence of $2^{\alpha(x)}$. No further coding machinery is required. As a consequence, testing whether $\mathbb{HFF}_s \models \exists x \Phi(x, \underline{n})$ can be handled by a one-dimensional cellular automaton that performs the necessary arithmetic and the logical operations required to reduce validity testing of the whole formula to its

atomic components. The required operations are not particularly complicated; for an implementation of executable finite set theory for example in Haskell see [41].

Finite configurations are sufficient for this simulation, but in the light of Cook's theorem one may expect that the necessary computations can be performed more easily by a cellular automaton operating on almost periodic configurations. Now consider an observer having access to the canonical representations of the almost periodic configurations appearing in the orbit of a suitable initial configuration. It is easy to construct an observer that extracts either the set A or the set S , yielding undecidable observation languages. Alas, it is just as easy to build an observer that extracts the disjoint union $S \oplus A$, coded in some natural fashion, from the process, and the latter set is r.e.-complete. Thus the construction fails to produce an intermediate process.

It appears that similar problems arise in all priority constructions. Jockusch and Soare suggested in [19] that priority constructions obey a kind of "maximum degree principle" that forces constructed sets to be complete provided the requirements are sufficiently weak. If the requirements are strong enough to force incompleteness of the constructed set computational completeness appears in some other part of the construction. This is irrelevant for the existence proofs established by priority arguments but appears to derail any effort to construct an intermediate process when observers have full access to all parts of the computation.

5. Open Problems

We have proposed almost periodic configurations as a natural setting for the study of computation in a physics-like setting. In light of Cook's result it seems that smaller spaces of configurations such as the often invoked configurations of finite support are inappropriate. The question remains whether there are larger configuration spaces that should be considered. As an upper bound, recursive configurations are a step too far: while every recursive configuration that has a predecessor also has a recursive predecessor the latter cannot be calculated effectively from the former [37]. In contrast, first order properties of one-dimensional cellular automata are decidable by theorem 2.3 and corresponding witnesses can be computed effectively.

As already mentioned, the decidability of the first order theory of the phasespace of a one-dimensional cellular automaton does not yield efficient algorithms per se. Due to exponential growth of the automata required to test properties of even modest complexity any attempt at implementation meets with considerable difficulties. We are currently experimenting with fast implementations of Safra's determinization algorithm as well as determinization avoidance. Another possible line of approach is the use of binary decision diagrams for succinct representations of the ζ -automata encountered in the decision algorithm. It is unclear at this point how far this approach can be pushed before state explosion makes practical computation unfeasible.

There are two central questions regarding the logical theories associated with cellular automata. First, there is the problem of classifying the first order theories. While these theories are all decidable it is not clear whether there, say, the theory of rule 110 is significantly different from the theories of other elementary automata? How about rule 30? More generally, are there interesting classes of first order theories associated with cellular automata? Second, are there interesting restricted classes of cellular automata where stronger logics such as monadic second order logic are decidable? Alternatively, when is the first order theory of the structures augmented by a reachability predicate decidable? Nilpotent

cellular automaton fall into this class, are there any other interesting cases? Note that the automata-based decision algorithm for section 2 fails completely for higher-dimensional cellular automata. Are there interesting classes of, say, 2-dimensional cellular automata that have decidable first order theory?

Returning to the Principle of Computational Equivalence, the question arises whether one can demonstrate that certain classes of, say, finite injury priority arguments never give rise to intermediate computational process, either in the sense of our definitions or in some similar scenario. More generally, is there a meaningful definition of computational process and intermediate computational process where the non-existence of the latter can be established? As pointed out above, the hereditarily finite sets seem to provide a natural environment in which to express constructions in recursion theory. It would be interesting to establish the notion of a computational process with attendant observer directly in this setting, without the detour to one-dimensional cellular automata. On the other hand, one might consider fairly concrete models such as Fredkin's SALT automata [24] that are reversible and enjoy various conservation properties. Considering the complexity of all known constructions of intermediate sets it seems difficult to imagine that any of them could work without reference to a universal Turing machine whose presence could not be detected by a suitable observer.

References

- [1] K. Ambos-Spies and P. A. Fejer. Degrees of unsolvability. <http://www.cs.umb.edu/~fejer/articles>, 2006.
- [2] S. Amoroso and Y. N. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *Journal of Computer and Systems Sciences*, 6:448–464, 1972.
- [3] J. Barwise. *Admissible Sets and Structures*. Perspectives in Mathematical Logic. Springer, 1975.
- [4] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer Verlag, 1997.
- [5] A. Blumensath and E. Grädel. Automatic structures. In *Proc. 15th IEEE Symp. on Logic in Computer Science*, pages 51–62. IEEE Computer Society Press, 1999.
- [6] M. Born. *Die Relativitätstheorie Einsteins und ihre physikalischen Grundlagen*. Springer Verlag, 1920.
- [7] C. C. Chang and H. J. Keisler. *Model Theory*. Studies in Logic and the Foundations of Mathematics. Elsevier, 1990.
- [8] P. Cockshott, L. Mackenzie, and G. Michaelson. Physical constraints on hypercomputation. *Theoretical Computer Science*, 394:159–174, 2008.
- [9] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.
- [10] K. Culik and Sheng Yu. Cellular automata, $\omega\omega$ -regular sets, and sofic systems. *Discrete Applied Mathematics*, 32:85–101, 1991.
- [11] M. Davis. *A note on universal Turing machines*, volume 34 of *Annals of Mathematics Studies*, pages 167–175. Princeton University Press, 1956.
- [12] M. Davis. Foundations of mathematics. <http://www.cs.nyu.edu/mailman/listinfo/fom/>, 2003.
- [13] M. Davis. The myth of hypercomputation. In C. Teuscher, editor, *Alan Turing: the life and legacy of a great thinker*, pages 195–212. Springer Verlag, 2006.
- [14] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Patterson, and W. P. Thurston. *Word Processing in Groups*. Jones and Bartlett, 1992.

- [15] O. Finkel. On decidability properties of one-dimensional cellular automata. *Computing Research Repository*, abs/0903.4615, 2009.
- [16] E. R. Griffor, editor. *Handbook of Computability Theory*, volume 140 of *Studies in Logic*. North-Holland, 1999.
- [17] L. Harrington and S. Shelah. The undecidability of the recursively enumerable degrees. *Bull. Amer. Math. Soc.*, 6:79–80, 1982.
- [18] G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Math. Systems Theory*, 3:320–375, 1969.
- [19] C. Jockusch and R. I. Soare. Degrees of members of Π_1^0 classes. *Pacific J. Math.*, 40:605–616, 1972.
- [20] J. Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM J. Comput.*, 21(3):571–586, 1992.
- [21] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC '94: Int. Workshop on Logical and Computational Complexity*, pages 367–392, London, UK, 1995. Springer-Verlag.
- [22] B. Khoussainov and S. Rubin. Automatic structures: overview and future directions. *J. Autom. Lang. Comb.*, 8(2):287–301, 2003.
- [23] M. Lerman. *Degrees of Unsolvability*. Perspectives in Mathematical Logic. Springer, Berlin, 1983.
- [24] D. B. Miller and E. Fredkin. Two-state, reversible, universal cellular automata in three dimensions. In *CF '05: Proceedings of the 2nd conference on Computing Frontiers*, pages 45–51, New York, NY, USA, 2005. ACM.
- [25] M. Minsky. Size and structure of universal Turing machines using tag systems. In *Recursive Function Theory*, number 5 in Proc. Symp. Pure Mathematics, pages 229–238. American Mathematical Society, 1962.
- [26] P. G. Odifreddi. *Classical Recursion Theory, Volume II*, volume 143 of *Studies in Logic*. Elsevier, 1999.
- [27] D. Perrin and J.-E. Pin. *Infinite Words*, volume 141 of *Pure and Applied Math*. Elsevier, 2004.
- [28] G. E. Sacks. *Degrees of Unsolvability*. Princeton University Press, 1963.
- [29] S. Safra. On the complexity of ω -automata. In *Proc. 29th FOCS*, pages 319–327. IEEE Computer Soc. Press, 1988.
- [30] J. C. Shepherdson. Machine configuration and word problems of given degree of unsolvability. *Z. Math. Logik u. Grundlagen d. Mathematik*, 11:149–175, 1965.
- [31] R. I. Soare. The Friedberg-Muchnik theorem re-examined. *Canad. J. Math.*, 24:1070–1078, 1972.
- [32] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer, 1987.
- [33] K. Sutner. Cellular automata and intermediate reachability problems. To appear in *Fundamenta Informaticae*.
- [34] K. Sutner. Model checking one-dimensional cellular automata. Automata 07, Fields Institute, Toronto, to appear in *JCA*.
- [35] K. Sutner. A note on Culik-Yu classes. *Complex Systems*, 3(1):107–115, 1989.
- [36] K. Sutner. Classifying circular cellular automata. *Physica D*, 45(1–3):386–395, 1990.
- [37] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.
- [38] K. Sutner. Linear cellular automata and Fischer automata. *Parallel Computing*, 23(11):1613–1634, 1997.

- [39] K. Sutner. Almost periodic configurations on linear cellular automata. *Fundamenta Informaticae*, 58(3,4):223–240, 2003.
- [40] K. Sutner. Cellular automata and intermediate degrees. *Theoretical Computer Science*, 296:365–375, 2003.
- [41] P. Tarau. A functional hitchhiker’s guide to hereditarily finite sets, Ackermann encodings and pairing functions. *Computing Research Repository*, abs/0808.0754, 2008.
- [42] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *P. Lond. Math. Soc.*, 42:230–65, 1936.
- [43] K. Weihrauch. *Computable Analysis*. EATCS Monographs. Springer, Berlin, 2000.
- [44] S. Wolfram. Random sequence generation by cellular automata. *Adv. in Appl. Mathematics*, 7(2):123–169, 1986.
- [45] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, IL, 2002.
- [46] K. Zuse. *Rechnender Raum*. F. Vieweg & Sohn, Braunschweig, 1967.