Chapter 12
Linear Cellular Automata and Decidability

Klaus Sutner

Abstract. We delineate the boundary between decidability and undecidability in the context of one-dimensional cellular automata. The key tool for decidability results are automata-theoretic methods, and in particular decision algorithms for automatic structures, that are inherently limited to dealing with a bounded number of steps in the evolution of a configuration. Undecidability and hardness, on the other hand, are closely related to the full orbit problem: does a given configuration appear in the orbit of another?

12.1 Linear Cellular Automata

Cellular automata arise as a natural discretization of continuous dynamical systems. In the continuous case, we are dealing with configurations of the form, say, $X : \mathbb{R} \to \Sigma$, where Σ is some set of values whose precise nature is not important here. Write $\mathcal{C}$ for the space of all configurations. The shift operation $\sigma_a(x) = x + a$ acts on $\mathcal{C}$ in the natural way and accounts for a change in coordinates. The dynamics of the system are then given by a family of evolution operators $\tau_t : \mathcal{C} \to \mathcal{C}$, where "time" t is a non-negative real. These operators form a monoid under composition: $\tau_t \circ \tau_s = \tau_{t+s}$ and $\tau_0 = \mathrm{Id}$. By discretizing the underlying space, as well as the operators σ and τ, we naturally arrive at maps $G : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ that are continuous in the standard product topology and invariant under the discrete shift operator $\sigma : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$, $\sigma(X)(i) = X(i+1)$. This is expressed in the classical Curtis-Hedlund-Lyndon theorem, see [18]. Hence, in a cellular automaton the *global map* $G = G_\rho$ can be represented by a finite lookup table, essentially a map $\rho : \Sigma^w \to \Sigma$, and is thus amenable to analysis from the perspective of decidability and computational complexity. It is customary to refer to this finite function as the *local map* or *rule* of

Klaus Sutner
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213, USA
e-mail: sutner@cs.cmu.edu

© Springer International Publishing Switzerland 2015

A. Adamatzky (ed.), *Automata, Universality, Computation,*
Emergence, Complexity and Computation 12, DOI: 10.1007/978-3-319-09039-9_12

the cellular automaton. The operator monoid is generated by G and we are therefore interested in the iterates G^t for all $t \in \mathbb{N}$.

One of the main areas of interest in the study of cellular automata is their *classification* with respect to the evolution of configurations, i.e., the repeated application of the global map. This problem was expressed by Wolfram in [53] as "What overall classification of cellular automata behavior can be given?" While aspects of short-term evolution such as reversibility are not excluded, it is clear from the classification proposed by Wolfram that he had the long-term evolution in mind, see [52, 54]. In a nutshell, the Wolfram classification looks like so. The evolution of a configuration leads to

- Class I: homogeneous fixed points,
- Class II: periodic configurations,
- Class III: chaotic, aperiodic patterns,
- Class IV: persistent, complex patterns of localized structures.

In [27, 28] Li and Packard introduce an alternative version of this hierarchy. Again, their classification is based on the asymptotic behavior of the automaton.

Another well-known attempt at classification is due to Kurka, see [25, 26, 31, 47], and is based on the topological notions of equicontinuity, sensitivity to initial conditions and expansivity. Needless to say, these notions all relate to long-term behavior.

In [30], Margenstern gave a comprehensive description of the "frontier between decidability and undecidability," in a variety of computational settings. In this paper we will focus on one-dimensional cellular automata and discuss an approach to classification that is based on model theory and computability rather than classical dynamics. Our key tool is the use of automata theory to obtain decidability results for the first-order theory of cellular automata, construed as first-order structures. Using first-order logic one can easily formalize assertions such as "the global map is surjective," "the system is reversible", "the global map is k-to-1" where k is fixed, "there exists a 5-cycle" or "there are exactly 2 fixed points." Thus, we will deal exclusively with one-dimensional cellular automata; their higher-dimensional analogues are not amenable to these techniques as can be seen for example from Kari's theorem concerning the undecidability of injectivity and surjectivity of the global map in dimension 2, see [20]. As Douglas Lind pointed out, the study of higher-dimensional cellular automata requires one to step into "the Swamp of Undecidability," see [29]. In fact, even the most basic question of whether a shift of finite type, given by a finite collection of forbidden 2-dimensional patterns, is nontrivial already turns out to be undecidable.

More formally, suppose a one-dimensional cellular automaton has local map $\rho : \Sigma^w \to \Sigma$. We consider the associated first-order relational structure

$$\mathfrak{C}_\rho = \langle \mathcal{C}, \to \rangle$$

where Σ is a finite set, the *alphabet* of the cellular automaton, and w the *width*, $\mathcal{C}$ denotes the space of *configurations*. We always assume that our language includes

equality without further mention. Given a first-order sentence φ, we want to determine whether φ is valid over $\mathfrak{C}_\rho$, in symbols $\mathfrak{C}_\rho \models \varphi$. In computer science this problem is often referred to as *model checking* , see [9, 19]. There are two natural variants that are both relevant in the context of cellular automata: first, in *expression model checking* the structure is fixed and we are interested in establishing the validity of a number of assertions. For example, we may be interested in understanding properties of particularly interesting specific elementary cellular automata such as rule 30 or rule 110, [54]. Second, in *data model checking* the sentence is fixed and one tries to determine validity in as many structures as possible. For example, one might try to examine the existence of a particular finite subgraph of $\mathfrak{C}_\rho$ for all elementary cellular automata ρ. In our case, the machinery for both problems is quite similar, though in particular for data model checking it is important to optimize the decision procedures in order to avoid efficiency problems.

The reasons for representing the global map as a binary relation rather than a function are purely technical and need not concern us here. To avoid trivial cases, we assume that Σ has cardinality at least 2, so configuration spaces are infinite and indeed uncountable, but see section 12.2 below for a natural reduction to countable subspaces. For our purposes, there are three natural choices of the configuration space $\mathcal{C}$: the bi-infinite case Σ^ζ, the one-way infinite case Σ^ω and the finite case Σ^n. In the last case, $n \in \mathbb{N}$ is a parameter and we are mostly concerned with the *spectrum* of a first-order sentence, see section 12.2.1 below. Correspondingly, we write $\mathfrak{C}_\rho^n$ for the finite structure $\langle \Sigma^n, \to \rangle$, $\mathfrak{C}_\rho^\omega$ for the one-way infinite structure $\langle \Sigma^\omega, \to \rangle$ and $\mathfrak{C}_\rho^\zeta$ for the two-way infinite structure $\langle \Sigma^\zeta, \to \rangle$, following the terminology established in [35, 40] for the corresponding automata. In all cases, we think of configurations as *words* over a finite alphabet Σ and use appropriate finite state machines to obtain decision algorithms: ordinary word automata in the finite case, Büchi automata in the infinite case and ζ-automata in the bi-infinite case. Note that special care is needed to deal with boundary conditions both in the finite and one-way infinite case. We point out that the notion of finite configuration here refers exclusively to configurations that are finite in the sense that they involve a finite grid of cells. Unfortunately, it is customary in the literature to use the same terminology with respect to infinite configurations that have finite support: only finitely many cells are in a state different from some specially designate null state. We will avoid this usage here. At any rate, $\mathfrak{C}_\rho$ is an *automatic structure* in the sense of Khoussainov and Nerode [22, 23] and generalizations in [5]. Historically, automaticity was first exploited in work by Gilman, Cannon, Holt, Thurston and others in study of various types of groups, see [12]. More recently, there has been substantial progress in elucidating the structure of automorphism groups of the infinite binary tree that are described by certain types of Mealy automata, see [4, 15, 16, 34, 50]. As we will see in section 12.2, automaticity can be exploited in all three cases to produce decision algorithms for the first-order theory of these structures. Alas, the efficiency of the algorithms varies considerably; in particular in the bi-infinite case the corresponding ζ-automata are algorithmically difficult to handle and it is hard to push the decision methods beyond very basic properties. One interesting technique in this connection

is to enlarge the language by adjoining suitable automatic predicates. For example, the predicate "differ in only finitely many places" is useful to express surjectivity of the global map in the bi-infinite case and easily seen to be automatic.

Needless to say, any full classification of cellular automata will require stronger logics such as monadic second-order logic or transitive closure logic. Alternatively, we can consider augmented structures

$$\mathfrak{T}_\rho = \langle \mathcal{C}, \to, \overset{*}{\to} \rangle.$$

where $\overset{*}{\to}$ denotes the transitive reflexive closure of $\to$, the *reachability* or *orbit relation* of the automaton. Whenever necessary, we use qualifiers n , ω and ζ to indicate the type of cellular automaton under consideration. Pace Lind, this leads back into the undecidability swamp: the first-order theory of $\mathfrak{T}_\rho$ is not decidable in general, though there are interesting automatic relations where the augmented structure surprisingly remains automatic and can thus be handled in very much the same way as the plain structure, see [50].

In the following section 12.2 we will describe the machinery required to produce decision algorithms for the first-order theory of all three kinds of cellular automata. As we will see, there are significant efficiency obstacles to overcome, in particular in the bi-infinite case. Section 12.3 then summarizes corresponding undecidability results in connection with the long-term evolution of configurations. Lastly, we comment on open problems and future work.

12.2 The First-Order Theory

In this section we show how to solve a very limited version of the Entscheidungsproblem for one-dimensional cellular automata: the first-order theory of all these automata is decidable. Of course, the first-order theory of $\mathfrak{C}_\rho$ is too weak to deal with aspects of the long-term behavior of the cellular automaton, but it easily captures elementary properties such as injectivity, surjectivity, k-to-1-ness, the existence of k-cycles and so on. We can think of these properties as being local in the temporal sense, but note that we can obviously construct a first-order sentence that asserts, say, that every configuration has distance at most k from an ℓ-cycle. Or we can express the assertion that a particular finite directed graph has an isomorphic copy somewhere in $\mathfrak{C}_\rho$. As we will see, all these temporally local properties are decidable, at least in principle.

12.2.1 Finite Configurations

In the finite case we are dealing with structures $\mathfrak{C}_\rho^n$ which are clearly automatic, uniformly in n, meaning that the same automata can be used for all these structures.

As a sample decision problem, consider the standard question of testing reversibility of the system, i.e., injectivity of the global map. The corresponding problem for bi-infinite cellular automata was handled by Amoroso and Patt [2] and appears to be the first clear example of a decidability result in the context of cellular automata, following the paradigm established by Rabin and Scott [36]. To see how to tackle injectivity in our setting, consider first an automaton $\mathcal{A}_\rho(x,y)$ that tests whether two finite words $X,Y \in \Sigma^n$ are related by $X \to Y$. The automaton works on words over the alphabet Σ^2 which we may consider as having two tracks:

$$X{:}Y = \begin{array}{|c|c|c|c|c|c|c|}\hline x_1 & x_2 & \dots & x_i & \dots & x_{n-1} & x_n \\ \hline y_1 & y_2 & \dots & y_i & \dots & y_{n-1} & y_n \\ \hline \end{array} \in \Sigma^2$$

This two-track word $X{:}Y \in (\Sigma^2)^n$ is often referred to as the *convolution* of X and Y, unfortunate but well-established terminology. For simplicity assume that the width $w = 2r+1$ of the local map is odd. To test whether word y is obtained from x by uniform application of the local map on all finite blocks of the form $x_{i-r},\dots,x_i,\dots,x_{i+r}$, the appropriate automaton $\mathcal{A}_\rho(x,y)$ has state set $\Sigma^{2r} \times \Sigma^r$ and the transitions are of the form

$$\begin{array}{c} a_1,\dots,a_{2r} \\ b_1,\dots,b_r \end{array} \xrightarrow{a:b} \begin{array}{c} a_2,\dots,a_{2r},a \\ b_2,\dots,b_r,b \end{array}$$

provided that $\rho(a_1,\dots,a_{2r},a) = b_1$. Here $a{:}b$ indicates the 2-track letter composed of $a,b \in \Sigma$ and labels the transition. Thus, we are dealing with a subautomaton of the complete de Bruijn automaton over Σ^2 of order $2r$: we remove all the directed edges from the full de Bruijn automaton that do not conform to ρ. An example is shown in figure 12.1. Note that the underlying CA is the additive rule 150, as a consequence the automaton uses the full de Bruijn graph.

A further complication is caused by the boundary conditions associated with a finite cellular automaton. In the case of fixed boundary conditions, we can augment the de Bruijn automaton with additional initial and final states that represent the phantom cells. More precisely, the initial states have indegree 0 and transitions leading to the main automaton; the final states have outdegree 0 and are reachable via transitions leaving the main automaton. To deal with periodic boundary conditions, these additional states have to be associated appropriately.

Injectivity is now easily expressed in our relational setting as the first-order formula

$$\varphi \equiv \forall x,y,z\,(x \to z \wedge y \to z \Rightarrow x = y)$$

To convert the sentence φ into an automaton $\mathcal{A}_\varphi$ we use the 3-track alphabet $\Gamma = \Sigma^3$. We combine two copies of the $\to$-testing automaton, $\mathcal{A}_\rho(x,z)$ and $\mathcal{A}_\rho(y,z)$, associated with tracks in the obvious manner. A standard product construction is used to express logical conjunction of $x \to z$ and $y \to z$, refer to the result as $\mathcal{A}'$. Lastly, let $\mathcal{A}$ be the conjunction of $\mathcal{A}'$ and an automaton testing inequality $x \neq y$. This last step amounts to creating two copies of $\mathcal{A}'$, with appropriate cross transitions that verify inequality. To check injectivity we need to test the acceptance language of $\mathcal{A}$ for emptiness. Clearly, this last step can be handled in time linear in the size of

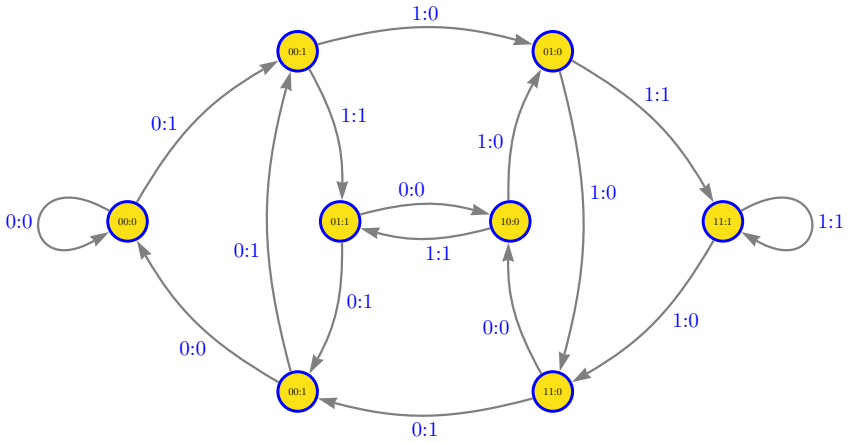

Fig. 12.1 An example of an automaton $\mathcal{A}_\rho$; the underlying cellular automaton here is the additive elementary automaton number 150

$\mathcal{A}$, which size is quadratic in the size of the lookup table for the local map of the cellular automaton and we have a perfectly practical algorithm.

In the general case, the algorithm rests on the fact that regular languages form an effective Boolean algebra. Suppose we are given a first-order formula $\varphi(x_1,\ldots,x_k)$ with k free variables as indicated. The decision algorithm constructs an automaton $\mathcal{A}_\varphi$ that accepts exactly those k-track words that satisfy the formula:

$$\mathcal{L}(\mathcal{A}_\varphi) = \{u_1{:}u_2{:}\ldots{:}u_k \in \Gamma^\star\}\, \mathfrak{C}_\rho \models \varphi(u_1,u_2,\ldots,u_k)$$

where $\Gamma = \Sigma^k$ is the k-track version of Σ. The construction proceeds by induction on the subformulae of φ. For simplicity, assume that the given formula is in prefix normal form, but note that this may not be the most desirable setup for efficiency reasons.

We can handle the Boolean connectives in the matrix of the formula using standard algorithms such as product machines and determinization, see [40] for a recent description of the requisite machinery. Note that determinization here refers to the classical Rabin-Scott algorithm [36] used for word automata; in the following sections significantly more complicated procedures are required.

Universal quantifiers can be translated into existential ones via the standard transformation $\forall x\,\varphi \equiv \neg\exists x \neg\varphi$. Somewhat surprisingly, existential quantifiers are actually easier to handle than logical connectives, at least from a purely algorithmic perspective: we can simply erase the corresponding track from the transition labels. Note, however, that this transformation usually introduces nondeterminism, which can cause problems at later stages if determinization is required to deal with logical

negation. Unfortunately, this typically happens when dealing with universal quantifiers in the manner described above.

A first-order sentence has no free variables. To handle this case properly it is convenient to adopt the convention that $(\Sigma^0)^\star$ is the 2-element Boolean algebra $\{\top, \bot\}$. Projection yields $\bot$ if the set is empty and $\top$ otherwise. To determine which case obtains one has to test the corresponding regular language for emptiness, which can be handled by standard path-existence algorithms in the unlabeled directed graph that results after all quantifiers have been removed. At any rate, we have the following result, see also [48].

Theorem 1. *First-order logic for finite one-dimensional cellular automata is decidable.*

It should be noted that the use of product automata and determinization has the potential effect of exponential blow-up in the size of the finite state machines involved in the decision algorithm. Hence, as a practical matter, only relatively simple formulae can be handled.

While the automaton $\mathcal{A}_\varphi$ can be used to determine validity for specific values of n, it is usually more interesting to exploit it to compute the *spectrum* of φ, defined as the tally language

$$\mathsf{spec}(\varphi) = \{0^n \mid \mathfrak{C}^n_\rho \models \varphi\} \subseteq 0^\star$$

As a tally language, the spectrum can be determined by a finite state machine and hence must be regular.

Lemma 1. *Any sentence in first-order logic has regular spectrum over the structures* $\mathfrak{C}^n_\rho$.

Alternatively we can think of the spectrum as a set of natural numbers, in which case the set must be semi-linear (i.e., a finite union of linear sets). As an example consider the elementary cellular automaton number 90, an additive automaton whose local rule corresponds to the exclusive-or of the left and right argument. The property "every configuration has exactly 4 predecessors" here has spectrum $2\mathbb{N}$. Likewise, elementary cellular automaton number 30 has spectrum $12\mathbb{N}$ for the property "there is a 3-cycle." See Wolfram [54] and references therein for more background on elementary cellular automata. One particularly important case arises when the spectrum of a sentence is universal: all finite grids have the property in question. We can test universality of a first-order sentence φ by testing universality of the associated automaton, at least disregarding the fact that this test is PSPACE-hard in general [14]. In general, efficiency is a non-negligible issue in the finite case, but with some effort one can obtain feasible decision algorithms for interesting properties.

12.2.2 Infinite Configurations

To lift our decision algorithm to infinite configurations of the form $X \in \Sigma^\omega$ we need to generalize ordinary word automata to machines operating on infinite inputs. More precisely, we can retain the finite transition systems that describe word automata, but we have to work with a significantly more complicated acceptance condition. To this end, a *Büchi automaton* $\mathcal{B} = \langle Q, \Sigma, \tau; I, F \rangle$ is said to accept a word $X \in \Sigma^\omega$ if there is a computation $\mathcal{B}$ on X that starts at a state in I and touches F infinitely often, see [35,40]. In other words, for a computation π, let $\mathrm{Inf}(\pi) = \{ p \in Q \mid \overset{\infty}{\exists} i \in \mathbb{N} (p_i = p) \}$ denote the set of recurrent states in π. Then $\mathcal{B}$ accepts X if there exists some computation π on X, starting at I, such that $\mathrm{Inf}(\pi) \cap F \neq \emptyset$. Of course, as a finite data structure a Büchi automaton is indistinguishable from an ordinary nondeterministic automaton.

The addition of automatic predicates to our language makes it possible to describe other properties of interest. For example, consider the left-shift operator L and suppose we adjoin a predicate xLy, also denoted by L, that is interpreted as "x is the left-shift of y." Using the standard sloppy notation to express chains of relations, we can then write in the extended language $\mathcal{L}(\to, L)$

$$\exists x, y, z, u, v \, (x \to y \to z \to u \wedge uLvLx)$$

to formalize the assertion that, for some configuration X, we have $G^3(X) = L^2(X)$. Thus we can state that 3 steps in the temporal evolution correspond to a double left-shift. Similarly we could restrict our attention to spacially periodic configurations of a fixed period. Note that chains such as $x \to y \to z \to u$ in the preceding formula require an adjustment in the basic de Bruijn automaton; except for u both tracks of a state will now contain $2r$ symbols.

Symbolic dynamics is concerned with the Cantor space Σ^ω, but our use of finite state machines to determine first-order properties has the side effect that we are essentially dealing with a smaller space. Define a configuration to be *ultimately periodic* if it is of the form vw^ω where v and w are finite words, w not empty. Thus, after a finite initial segment v the configuration simply repeats the block w forever. We write $\mathscr{C}_{\mathsf{up}}$ of this space of configurations. From the perspective of computability, ultimately periodic configurations may seem ad hoc, but they have a perfectly good justification in terms of model theory:

Theorem 2. *The space $\mathscr{C}_{\mathsf{up}}$ of ultimately periodic configurations is an elementary substructure of the full space.*

This is easy to see using the standard Tarski-Vaught test, see [8, 49]. Thus, ultimately periodic configurations are finitary objects but are indistinguishable from arbitrary configurations from the perspective of first-order logic. In fact, they form the least class of configurations that contain configurations of finite support and form an elementary subspace. Another useful property of this subspace is that its elements afford a natural finite description as pairs of finite words (v, w). Hence the orbits on $\mathscr{C}_{\mathsf{up}}$ are *recursively enumerable*, or r.e. for short, see [41]. As in the finite

case, the collection of ω-regular languages over some alphabet forms an effective Boolean algebra and we can lift the construction from the finite to the infinite case.

Theorem 3. *First-order logic for infinite one-dimensional cellular automata is decidable.*

It was pointed out by O. Finkel that we can extend our language slightly by quantifiers for "there exist infinitely many" and "there exist r mod k many" without affecting decidability, see [13]. As a consequence, we can check, say, whether there are infinitely many fixed points. Alas, efficiency problems now become dominant: it is still straightforward to determine emptiness of a Büchi automaton, but the construction of the machines becomes problematic. As an example, consider the following test related to *nilpotency*: a cellular automaton is nilpotent if there exists a quiescent configuration Y such that all configurations evolve to Y in finitely many steps. A standard compactness argument shows that there has to be a fixed bound n, the nilpotency index, that limits the length of the corresponding transient:

$$\exists y \text{ quiescent } \forall x \exists x_1, \ldots, x_{n-1} \, (x \to x_1 \to x_2 \to \ldots \to x_{n-1} \to y)$$

A priori, quiescence is not first-order definable in $\mathcal{L}(\to)$, but we can easily add an automatic predicate that singles out quiescent configurations. The matrix of the formula has a simple structure, and can easily be converted into a product de Bruijn automaton; alas, the size of this automaton is exponential in n and there is little hope to be able to deal with the universal quantifier, except for exceedingly small values of n.

While Büchi automata can be represented by the exact same data structure as a standard nondeterministic automaton, some of the attendant algorithms are significantly more complicated. In particular determinization based on Safra's algorithm [39] is notoriously difficult to implement well; the upper bound of $O(n^n)$ is known to be tight in general. Indeed, much effort has gone into finding ways around determinization, see [24] and references there. Minimization techniques are also of little help, since one would need to minimize nondeterministic machines: deterministic Büchi automata are strictly weaker than their nondeterministic counterparts, so one has to resort to other devices such as Rabin and Muller automata, see [35]. The data structures associated with these types of automata can be quite large and are difficult to deal with in conversion algorithms between the various types of machines. Overall, and in contrast to the finite case, it is quite difficult to test interesting properties and more effort is needed to find ways to design feasible decision algorithms.

12.2.3 Bi-infinite Configurations

The study of classical symbolic dynamics deals with another, yet more complicated situation: bi-infinite words $X \in \Sigma^{\zeta}$. Acceptance conditions become correspondingly more unwieldy and, informally, need to cover an infinite past as well

as an infinite future. To this end let $\mathrm{Inf}^-(\pi) = \{\, p \in Q \mid \overset{\infty}{\exists} i \in \mathbb{N}(p_{-i} = p) \,\}$ and $\mathrm{Inf}^+(\pi) = \{\, p \in Q \mid \overset{\infty}{\exists} i \in \mathbb{N}(p_i = p) \,\}$ denote the set of negatively and positively recurrent states in π. Then $\mathcal{A}$ accepts a bi-infinite word X if there exists some computation π on X such that $\mathrm{Inf}^-(\pi) \cap I \neq \emptyset$ and $\mathrm{Inf}^+(\pi) \cap F \neq \emptyset$. While these conditions are arguably more complicated, they sometimes are actually easier to deal with than in the plain infinite case. To wit, in the the basic de Bruijn automata all states are initial and final, so that $\mathcal{A}_\rho(x,y)$ accepts its input if there is a bi-infinite computation: no complications arise from boundary conditions as in the finite and infinite case. This was used in [44] to give simple quadratic algorithms to test for injectivity, surjectivity and openness of the global map.

Of course, for more complicated formulae the construction of the associated automaton overall becomes more involved. The only known approach is to split a ζ-word into two ω-words and to use two Büchi automata to handle these one-way infinite input words, see again [35] for a careful discussion. Note, though, that we are actually dealing with a whole family of pairs of Büchi automata, corresponding to a representation of the ζ-regular languages in the form

$$L = \bigcup_{i \in I} U_i^{\mathsf{op}} V_i$$

where U^{op} stands for reversed ω-words obtained from U. The languages U_i and V_i are ω-regular and the index set I is finite. As in the infinite case we can identify a collection of ultimately periodic words, this time words of the form ${}^\omega uvw^\omega$ where u, v, w are finite words. Again we obtain an elementary subspace which we denote by $\mathscr{C}_{\mathsf{up}}$.

Theorem 4. *The space $\mathscr{C}_{\mathsf{up}}$ of ultimately periodic configurations is an elementary substructure of the full space of bi-infinite configurations.*

One can then express all the necessary Boolean operations as well as projections in terms of this union-of-pairs representation. Alas, there are substantial efficiency problems; in particular determinization becomes highly problematic. To see why, note that the first step in the determinization process is to make sure that that the languages U_i are pairwise disjoint, and likewise for the V_i. This can be achieved by exploiting the fact that ω-regular languages form and effective Boolean algebra, but may cause an exponential blow-up in the size of the index set. In conjunction with the complexity of determinization of Büchi automata this makes if typically unfeasible to handle first-order sentences of all but the most limited complexity. Even pure Σ_1 sentences can cause major efficiency problems if the matrix of the formula is sufficiently large. A typical example for this kind of problem is again the existence of "long" $\to$-chains, the automaton for the matrix grows exponentially in k. Still, we have a decision algorithm, at least in principle.

Theorem 5. *First-order logic for bi-infinite one-dimensional cellular automata is decidable.*

In this context it is particularly important to exploit computational shortcuts whenever possible. For example, surjectivity is a priori the Π_2 statement $\forall x \exists y\, (y \to x)$. The automaton corresponding to $\mathcal{A}_{\exists y\,(y \to x)}$ is nondeterministic and thus the test for universality necessitated by the outermost quantifier can cause exponential blow-up. On the other hand, one can exploit Hedlund's classical result that characterizes surjectivity in terms of injectivity on configurations with finite support. More precisely, introduce the obviously automatic predicate "equal except for finitely many places," in symbols $x\, E\, y$. We can now express surjectivity as $\forall x, y, z\, (x \to z \wedge y \to z \wedge x\, E\, y \Rightarrow x = y)$ in the extended language $\mathcal{L}(\to, E)$. Testing this formalization is significantly easier: once the automaton for the matrix of the formula has been constructed we simply have to solve a path-existence problem in a directed graph, a problem easily tackled in linear time and space.

Perhaps more importantly, the choice of suitable additional predicates makes it possible to formalize properties that fail to be first-order in the original language $\mathcal{L}(\to)$. As an example, consider openness of the global map, a property known to be equivalent to the map being k-to-1 for some k. As such, the property cannot be formalized in first-order. However, consider the predicate $x =_L y$ if $\exists n \in \mathbb{Z} \forall i < n\, (x_i = y_i)$ and likewise for $x =_R y$. It is easy to see that both predicates can be tested by a finite state machine. Both are weaker versions of being almost equal: $x\, E\, y \iff x =_L y \wedge x =_R y$. Hence, the global map is open if, and only if,

$$\forall x, y, z \left(x \to z \wedge y \to z \wedge (x =_L y \vee x =_R y) \Rightarrow x = y\right),$$

following the same pattern as injectivity and surjectivity and using the language $\mathcal{L}(\to, =_L, =_R)$. With a little more effort this produces an alternative proof for the quadratic algorithms from [44].

12.3 Undecidability and Hardness

When it comes to undecidability, there is little difference between the one-way infinite situation and the two-way infinite one, and we will focus on the latter. It is clear from the previous sections that the framework appropriate to the study of undecidability and computational hardness is given by the augmented structures $\mathfrak{T}_\rho^\zeta = \langle \mathcal{C}, \to, \overset{*}{\to} \rangle$ and their finite counterparts $\mathfrak{T}_\rho^n = \langle \Sigma^n, \to, \overset{*}{\to} \rangle$. Moreover, since the orbit relation $\overset{*}{\to}$ involves transitive closure one can expect hardness results for the plain Reachability Problem: for two configurations X and Y that have finitary descriptions, is $X \overset{*}{\to} Y$? Technically we need to augment our language by appropriate constants to reflect Reachability, but we will ignore these details. The first observation is

Theorem 6. *Reachability over $\mathfrak{T}_\rho^n$, uniformly in n, is* PSPACE*-hard.*

This follows from the fact that cellular automata are easily capable of simulating linear bounded automata. On the other hand, over $\mathfrak{T}_\rho^n$, validity of any first-order

sentence is trivially decidable in polynomial space, so the validity problem is PSPACE-complete. Thus, the problem is still decidable in principle, though the computation may well fail to be feasible. This changes drastically when we try to classify all finite structures at once by determining the spectrum of a sentence. In particular, consider

$$FP \equiv \forall x \exists y (x \xrightarrow{*} y \wedge y \to y),$$

a Π_2 sentence that expresses the assertion that every orbit ends in a fixed point. We can construe *FP* as a formalization of the first Wolfram class, see [11, 42] for a discussion of more general attempts to formalize Wolfram's heuristics. To test membership in this class we have to solve the data version of the model checking problem. Alas, this problem is undecidable.

Theorem 7. *It is undecidable whether the spectrum of the sentence FP from above is universal. In fact, this property is Π_1^0-complete.*

It should be noted that this result does not follow directly from standard results about the computations of Turing machines: we need to deal with all possible configurations of the cellular automaton, not just the ones that code stages in a computation. As it turns out, similar assertions about the length of the limit cycle of all orbits in $\mathfrak{C}_\rho^n$ for all n are undecidable, see [43].

With a view towards the importance of Reachability, it is desirable to address decidability questions in the augmented structures $\mathfrak{T}_\rho$ not in the full Cantor space of configurations but in a subspace that is amenable to the methods of classical computability theory as presented in, say, [37, 41]. As we have seen in sections 12.2.2 and 12.2.3, arguably the most plausible choice is the set of ultimately periodic configurations. Further evidence for the naturalness of this choice is given by Cook's result [10] that, in the bi-infinite case, there is an elementary cellular automaton capable of universal computation. Cook's proof of computational universality uses ultimately periodic configurations. Interestingly, the construction seems to rest on the ability to have two different periodic blocks, one extending to the left and the other to the right: in the construction, the left block serves to time the computation whereas the right block encodes the cyclic tag-system that is essential for universality. By contrast, Reachability for rule 110 is trivially decidable for configurations of finite support.

It has since been shown by Neary and Woods that the exponential slow-down inherent in the original construction can be avoided entirely, so that the simulation is actually quite efficient, at least from the perspective of complexity theory, see [32, 33]. This has the consequence that it is $\mathbb{P}$-complete to determine the state of a particular cell at time t of the evolution of a finite configuration under rule 110.

From now on, let us assume that the configurations in our structures, infinite or bi-infinite, are always ultimately periodic. It is obvious that in both cases Reachability is undecidable in general. Significantly more effort shows that one can carefully control the evolution of configurations on the cellular automaton in question so as to make sure that the complexity of Reachability is an arbitrarily chosen recursively enumerable degree, see [41]. Of course, at heart the construction rests on the ability

of a one-dimensional cellular automaton to simulate Turing machines. Alas, there are two substantial problems to overcome. First, hardness results for Turing machines are always phrased in terms of instantaneous descriptions, snap-shots of a computation that actually occur. By contrast, we have to deal with all configurations of the cellular automaton, most of which are meaningless from the perspective of the Turing machine. Second, we need to make sure that there are no unintended simulations that could drive the degree of Reachability up to, say, r.e.-completeness. It was shown in [42, 45, 46] and theorem 9 below how to cope with this problems. In summary, we can derive the following result.

Theorem 8. *The Reachability problem of an infinite or bi-infinite cellular automaton over $\mathscr{C}_{\mathsf{up}}$ can be chosen to be any r.e. degree.*

The theorem suggests that, if one is interested in computational properties, as opposed to the more classical dynamical systems aspects, one should consider a more fine-grained hierarchy based on the complexity of Reachability. The resulting classification is again highly undecidable: testing whether a cellular automaton has Reachability problem of degree $\mathbf{d}$ is $\Sigma_3^{\mathbf{d}}$-complete. In particular, testing for computational universality is Σ_4^0-complete. As it turns out, no constraints arise if we were to choose to base our classification on both Reachability and Confluence: two configurations are confluent if their orbits overlap. Thus, confluence is an equivalence relation and corresponds vaguely to basins of attraction. It is clear from the definitions that Confluence also has r.e. degree, but it is quite surprising that there is no connection between the two: given arbitrary r.e. degrees $\mathbf{d}_1$ and $\mathbf{d}_2$ there is a cellular automaton whose Reachability and Confluence problems have exactly those two chosen degrees as their complexity. It remains to be seen how other more complicated properties fit into this pictures.

As is well-understood, the semi-lattice of the r.e. degrees exhibits a rather complicated structure. For example, by a famous theorem of Sacks, the partial order of the r.e. degrees is dense: whenever $A <_T B$ for two r.e. sets A and B there is a third r.e. set such that $A <_T C <_T B$, [38]. Here $<_T$ denotes Turing reducibility. The first order theory of this semi-lattice is highly undecidable [17].

More complicated sentences in the first-order logic of $\mathfrak{T}_\rho$ will, in general, yield undecidable versions of the data model checking problem. For example, the nilpotency statement $\exists y \forall x (x \xrightarrow{*} y \wedge y \to y)$ is undecidable according to [21]. By interchanging quantifiers we obtain the "fixed point" sentence $FP \equiv \forall x \exists y (x \xrightarrow{*} y \wedge y \to y)$ from above. For finite cellular automata deciding FP is Π_1^0-complete, but for infinite and bi-infinite ones it is harder, given a slight restriction of the configuration space. For simplicity, let us only consider the bi-infinite automata, the argument is entirely similar in the one-way infinite case. Choose a particular symbol \$ in the alphabet and define $\mathcal{C}_\$ \subseteq \mathscr{C}_{\mathsf{up}}$ to be the class of all ultimately periodic configurations that contain \$ infinitely often in both directions. In other words, in the representation (u, v, w) both u and w contain \$. This property is easy to check by a pair of Büchi automata, so we are dealing with an automatic subspace.

Theorem 9. *Deciding FP for an infinite cellular automaton is Π_2^0-complete on $\mathcal{C}_\$$.*

For the proof first note that the problem lies in Π^0_2, since we only consider ultimately periodic configurations and these can be coded naturally as triples (u,v,w) of finite words, representing ${}^\omega uvw^\omega$. For hardness, recall that *INF*, the collection of all indices of infinite r.e. sets is well-known to be Π^0_2-complete, see [41]. Now $e \in INF \iff \forall n \exists m (n < m \wedge m \in W_e)$ where n and m range over the naturals. The construction of the cellular automaton ρ_e begins with a Turing machine M_e with binary tape that, on input 1^n dovetails on computations $m \in W_e$ for all $m > n$. More precisely, at stage s of the construction M_e will perform s steps in the computation of $m \in W_e$ for all $n < m < n + s$. If any of these computations converges, M_e halts.

A standard simulation of the Turing machine M_e by a cellular automaton will not produce the desired results since there are configurations of the cellular automaton that do not translate into instantaneous descriptions of M_e. For example, there might be several cells indicating state and head position of M_e; in fact, there could be infinitely many such cells due to the periodicity of the blocks on either side. A significantly more difficult problem is that a configuration may syntactically look like an instantaneous description of the Turing machine, but may actually not appear in any computation of M_e on any input 1^n. For example, it might contain an inaccessible state of the Turing machine M_e, which state then launches a divergent computation despite of the fact that the actual machine on input 1^n would halt. Let us refer to an instantaneous description as *admissible* if it occurs during a computation on some input. Needless to say, accessibility is not decidable in general but one can convert M_e into an equivalent Turing machine M'_e that is *stable* in the sense that it ultimately halts on all inadmissible descriptions, see [45, 49] for a more detailed description. The idea is to modify the machine so that it becomes self-verifying: retraces its steps every so often to try to verify that the current instantaneous description is indeed admissible. To this end, the Turing machine keeps a copy of the alleged original input throughout the computation. Roughly, we consider instantaneous descriptions of the form

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

where x, s and t are numbers written in unary and D and E are instantaneous descriptions of M_e, the original machine. The symbol $\underline{b}$ is the blank symbol for M_e and # is a new separator symbol. The idea is that x is the original input to M_e and that instantaneous description D occurs after s steps in the computation. The remaining fields are used for the admissibility test, t as a counter and E as a corresponding instantaneous description.

As to the actual cellular automaton that simulates the self-verifying Turing machine, consider the alphabet

$$\Gamma = \underset{\rightarrow}{\Sigma} \cup Q \cup \underset{\leftarrow}{\Sigma} \cup \{\$\}.$$

As usual, Q denotes the state set of the stable Turing machine and $\underset{\rightarrow}{\Sigma}$ and $\underset{\leftarrow}{\Sigma}$ are copies of the tape alphabet modified by additional direction bits that indicate whether the symbol is to the right or to the left of the state symbol. The special symbol \$ is the one that is required to appear infinitely often in both directions and serves the purpose of limiting the part of the configuration that can code instantaneous

descriptions. Any local configurations that represent syntactically incorrect descriptions are fixed points; for example, two adjacent symbols from $\underset{\rightarrow}{\Sigma}$ and $\underset{\leftarrow}{\Sigma}$ will stay fixed. Similarly two adjacent symbols from Q do not change. Changes do occur at local configurations of the form $\underset{\rightarrow}{\Sigma} \times Q \times \underset{\leftarrow}{\Sigma}$, except when the state in Q is the halting state of the stable Turing machine. The special symbol \$ can be converted into a symbol in $\underset{\rightarrow}{\Sigma} \cup Q \cup \underset{\leftarrow}{\Sigma}$ in the presence of the state head. As a consequence, the only orbits free of "frozen" components are the ones that correspond to instantaneous descriptions of M'_e, though a priori not necessarily admissible ones and not necessarily single ones either. But since M'_e is stable, inadmissible descriptions will be recognized and lead to the Turing machine halting. At the configuration level we obtain a fixed point. Thus, the only situation where a configuration can not evolve to a fixed point is when we are in fact simulating a divergent computation of M_e. But then $e \in INF$ if, and only if, the sentence FP holds over the structure $\mathfrak{C}_{\rho_e}$.

We note that the argument relies quite heavily on the fact that we consider configurations in $\mathcal{C}_\$$, for general ultimately periodic configurations there appears to be no way to organize the self-testing mechanism for the intermediate Turing machine. To wit, a non-fixed point computation could occur because of an ultimately periodic configuration of the cellular automaton that corresponds to an infinitely instantaneous description of the Turing machine.

12.4 Summary

We have shown that a small part of the classification of one-dimensional cellular automata can be handled automatically by using ideas from model checking. Unfortunately, this approach leads to considerable efficiency problems and it is not clear at present how far the corresponding decision algorithms can be pushed. Efforts are under way to give a local classification for elementary cellular automata on one-way infinite grids, but even in this relatively simply case there are problems dealing with the corresponding Büchi automata. In particular, determinization using Safra's classical algorithm often disrupts the decision algorithm. It is safe to assume that the problems become virtually unsurmountable when one moves on to bi-infinite automata. One could try use compact representations such as binary decision diagrams to try to cope with large state spaces [6, 7], but first experiments in this direction have been less than compelling.

Needless to say, any plausible classification scheme for cellular automata will have to go further and include aspects of the long-term evolution of configurations. Even in the one-dimensional case, assertions about long term evolution tend to be undecidable and fully automatic classification is simply impossible. On the other hand, as shown conclusively by Cook, simulation tools can help to sharpen one's intuition and provide sufficient insights into the behavior of a particular cellular automaton to complete classification "by hand." This is somewhat similar to the current situation in theorem proving and formally verified mathematics: fully automatic tools can cover only so much ground, but with sufficient intervention

from the user, proof assistants can produce quite interesting results, see the recent [3]. Some efforts in this direction can be found in [1, 51, 55]. One wonders whether crowd-sourcing might be helpful; in astronomy, classification of galaxies by large numbers of amateur volunteers has produced spectacular results, see `http://www.galaxyzoo.org/`. For example, the pseudo-random behavior of elementary cellular automaton rule 30 seems to make it a hopeless undertaking to encode any kind of undecidability proof. Perhaps many eyes could find enough structure to support such an argument.

One natural challenge is to determine the degree of the full first-order theory of $\mathfrak{T}_\rho$, perhaps first over some specialized configuration space such as $\mathcal{C}_\$$. One suspects that any level in the arithmetic hierarchy can be represented by a suitable assertion about orbits of a cellular automaton. Note, though, that the preceding construction differs in significant technical details from the result in [49] that shows that Reachability has arbitrary r.e. degree. In particular, a special annihilator symbol $\perp$ is used there to reduce the complexity of orbits of syntactically incorrect configurations to being decidable–the corresponding light cones of $\perp$ cells would break the current argument. Hence, it is not entirely clear how to generalize this kind of argument to longer chains of arithmetic quantifiers. Lastly, there is the question of the degree of the full first-order theory of $\mathfrak{T}_\rho$ over $\mathscr{C}_{\mathsf{up}}$ or the full configuration space.

References

1. Adamatzky, A.: Identification of Cellular Automata. Taylor & Francis, London (1994)
2. Amoroso, S., Patt, Y.N.: Decision procedures for surjectivity and injectivity of parallel maps for tesselation structures. Journal of Computer and Systems Sciences 6, 448–464 (1972)
3. Avigad, J., Harrison, J.: Formally verified mathematics. Comm. ACM 57(4), 66–75 (2014)
4. Bartholdi, L., Silva, P.V.: Groups defined by automata. CoRR, abs/1012.1531 (2010)
5. Blumensath, A., Grädel, E.: Automatic structures. In: Proc. 15th IEEE Symp. on Logic in Computer Science, pp. 51–62. IEEE Computer Society Press (1999)
6. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers C-35(8), 677–691 (1986)
7. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, J.: Symbolic model checking: 10^{20} states and beyond. Information and Computation 98(2), 142–170 (1992)
8. Chang, C.C., Keisler, H.J.: Model Theory. In: Studies in Logic and the Foundations of Mathematics, Elsevier (1990)
9. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
10. Cook, M.: Universality in elementary cellular automata. Complex Systems 15(1), 1–40 (2004)
11. Culik, K., Yu, S.: Undecidability of CA classification schemes. Complex Systems 2(2), 177–190 (1988)
12. Epstein, D.B.A., Cannon, J.W., Holt, D.F., Levy, S.V.F., Patterson, M.S., Thurston, W.P.: Word Processing in Groups. Jones and Bartlett, Burlington (1992)
13. Finkel, O.: On decidability properties of one-dimensional cellular automata. Computing Research Repository, abs/0903.4615 (2009)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman (1979)

15. Grigorchuk, R., Šunić, Z.: Self-Similarity and Branching in Group Theory. In: Groups St. Andrews 2005. London Math. Soc. Lec. Notes, vol. 339. Cambridge University Press (2007)
16. Grigorchuk, R.R., Nekrashevich, V.V., Sushchanski, V.I.: Automata, dynamical systems and groups. Proc. Steklov Institute of Math. 231, 128–203 (2000)
17. Harrington, L., Shelah, S.: The undecidability of the recursively enumerable degrees. Bull. Amer. Math. Soc. 6, 79–80 (1982)
18. Hedlund, G.A.: Endomorphisms and automorphisms of the shift dynamical system. Math. Systems Theory 3, 320–375 (1969)
19. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems, Cambridge, UP (2000)
20. Kari, J.: Reversibility of 2D cellular automata is undecidable. Physica D 45, 379–385 (1990)
21. Kari, J.: The nilpotency problem of one-dimensional cellular automata. SIAM J. Comput. 21(3), 571–586 (1992)
22. Khoussainov, B., Nerode, A.: Automatic presentations of structures. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995)
23. Khoussainov, B., Rubin, S.: Automatic structures: overview and future directions. J. Autom. Lang. Comb. 8(2), 287–301 (2003)
24. Kupferman, O.: Avoiding determinization. In: Proc. 21st IEEE Symp. on Logic in Computer Science (2006)
25. Kurka, P.: Languages, equicontinuity and attractors in cellular automata. Ergodic Th. Dynamical Systems 17, 417–433 (1997)
26. Kurka, P.: Topological and Symbolic Dynamics. Number 11 in Cours Spécialisés. Societe Mathematique de France, Paris (2003)
27. Li, W., Packard, N.: The structure of the elementary cellular automata rule space. Complex Systems 4(3), 281–297 (1990)
28. Li, W., Packard, N., Langton, C.G.: Transition phenomena in CA rule space. Physica D 45(1-3), 77–94 (1990)
29. Lind, D.: Multi-dimensional symbolic dynamics. In: *Symbolic Dynamics and its Applications*. Proc. Sympos. Appl. Math., vol. 60, pp. 61–79. AMS (2004)
30. Margenstern, M.: Frontier between decidability and undecidability: a survey. TCS 231, 217–251 (2000)
31. Meyers, R.A. (ed.): Encyclopedia of Complexity and System Science. Springer, Berlin (2009)
32. Neary, R., Woods, D.: On the time complexity of 2-tag systems and small universal turing machines. In: FOCS, pp. 439–448. IEEE Computer Society, Washington (2006)
33. Neary, T., Woods, D.: P-completeness of cellular automaton rule 110. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 132–143. Springer, Heidelberg (2006)
34. Nekrashevych, V.: Self-Similar Groups. In: Math. Surveys and Monographs, vol. 117. AMS (2005)
35. Perrin, D., Pin, J.-E.: Infinite Words. In: Pure and Applied Math., vol. 141, Elsevier, Amsterdam (2004)
36. Rabin, M.O., Scott, D.S.: Finite automata and their decision problems. IBM Jour. Research 3(2), 114–125 (1959)
37. Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill, New York (1967)
38. Sacks, G.E.: The recursively enumerable degrees are dense. Ann. Math. 80, 300–312 (1964)

39. Safra, S.: On the complexity of ω-automata. In: Proc. 29th FOCS, pp. 319–327. IEEE Computer Soc. Press, Washington (1988)
40. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press (2009)
41. Soare, R.I.: Recursively Enumerable Sets and Degrees. In: Perspectives in Mathematical Logic. Springer, Berlin (1987)
42. Sutner, K.: A note on Culik-Yu classes. Complex Systems 3(1), 107–115 (1989)
43. Sutner, K.: Classifying circular cellular automata. Phys. D 45(1-3), 386–395 (1990)
44. Sutner, K.: De Bruijn graphs and linear cellular automata. Complex Systems 5(1), 19–30 (1991)
45. Sutner, K.: Cellular automata and intermediate degrees. Theoretical Computer Science 296, 365–375 (2003)
46. Sutner, K.: Universality and cellular automata. In: Margenstern, M. (ed.) MCU 2004. LNCS, vol. 3354, pp. 50–59. Springer, Heidelberg (2005)
47. Sutner, K.: Encyclopedia of Complexity and System Science, chapter Classification of Cellular Automata. In: Meyers [31], (2009)
48. Sutner, K.: Model checking one-dimensional cellular automata. J. Cellular Automata 4(3), 213–224 (2009)
49. Sutner, K.: Cellular automata, decidability and phasespace. Fundamenta Informaticae 140, 1–20 (2010)
50. Sutner, K., Lewi, K.: Iterating invertible binary transducers. In: Kutrib, M., Moreira, N., Reis, R. (eds.) DCFS 2012. LNCS, vol. 7386, pp. 294–306. Springer, Heidelberg (2012)
51. Vorhees, B.: Computational Analysis of One-Dimensional Cellular Automata. World Scientific, Singapore (1996)
52. Wolfram, S.: Computation theory of cellular automata. Comm. Math. Physics 96(1), 15–57 (1984)
53. Wolfram, S.: Twenty problems in the theory of cellular automata. Physica Scripta T9, 170–183 (1985)
54. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign (2002)
55. Wuensche, A.: Classifying cellular automata automatically. Complexity 4(3), 47–66 (1999)