Complex Systems **5** (1991) 19–30

De Bruijn Graphs and Linear Cellular Automata

Klaus Sutner
Stevens Institute of Technology, Hoboken, NJ 07030 USA

Abstract. De Bruijn graphs provide a convenient way to describe configurations of linear cellular automata (CAs). Using these graphs, we give a simple quadratic time algorithm to determine whether a linear CA is reversible. Similarly, one can decide in quadratic time whether the global map of the automaton is surjective. We also show that every recursive configuration that has a predecessor on a linear CA already has a recursive predecessor. By contradistinction, it is in general impossible to compute such a predecessor effectively.

1. Introduction

Cellular automata (CAs) are dynamic systems: the space of all configurations is compact under the usual product topology and the global CA map is a continuous shift-invariant operator on this space. On the other hand, configurations of a one-dimensional or linear CA may also be construed as a biinfinite word over the state set of the automaton. Correspondingly, the global rule of the CA may be interpreted as a finite-state transducer that acts on the space biinfinite words. With this interpretation one can then apply ideas from the theory of formal languages to the study of linear CAs.

Our work was motivated by the search for those properties of CAs that are effectively decidable. It was shown by Amoroso and Patt that it is decidable whether the global map of a one-dimensional CA is injective [1]. It is well known that CAs are reversible—in the sense that there exists another CA over the same alphabet that reverses the evolution of configurations on the first—if and only if the global function is injective, that is, reversible in the set-theoretic sense. This was shown in [16]. Hence reversibility of one-dimensional CAs is decidable. By contradistinction reversibility of higher-dimensional CAs fails to be decidable as was recently shown by Kari [9]. The proof is based on the existence of a combinatorial structure in the plane, namely a special directed tiling, which has no counterpart in one dimension. Similarly one can show that surjectivity of the global map is decidable for linear CAs but fails to be decidable for all dimensions higher than one.

The argument by Amoroso and Patt is purely combinatorial. Alternative proofs were given by Culik [8] and Head [6] using automata theoretic

methods. Culik uses the fact that a CA in dimension one is a special type of generalized sequential mapping and brings to bear results about these mappings. In Head's work, the de Bruijn graphs introduced in [20] are used to reduce injectivity and surjectivity to an ambiguity problem for finite automata.

Decidability by itself is still a rather weak condition: the existing algorithm may demand unreasonable computational resources. Thus a property of a CA my be decidable but intractable. For example, to test for surjectivity one can determine whether a certain nondeterministic finite automaton (NFA) accepts all strings over its alphabet. The NFA can be described conveniently in terms of de Bruijn graphs, as shown in section 2 below. The standard algorithm to determine whether an NFA accepts all inputs involves the construction of a deterministic simulation automaton and in general requires exponential time as well as exponential space. Indeed, for arbitrary NFAs it is **PSPACE**-hard to test whether the automaton accepts all inputs (see [3]).

However, we will show that for the NFAs arising from linear CAs the problem can be solved in quadratic time. Hence surjectivity of a linear CA map can be tested in quadratic time. Our algorithm uses only standard graph theoretic methods and is straightforward to implement. Complete programs written in *Mathematica* are available from the author. A slight modification of our algorithm allows one to determine reversibility of a linear CA quadratic time.

One of the historically first problems to be studied in connection with CAs is the existence of configurations without predecessors. These configurations are often referred to as Gardens-of-Eden [2, 12, 14]. Clearly a Garden-of-Eden exists if and only if the global map of the automaton fails to be surjective. The de Bruijn graphs in our decision algorithms can also be applied to the study of predecessor configurations in linear CAs.

The existence of a predecessor configuration of a given target configuration Y is tantamount to the existence of a path in the de Bruijn graph whose label sequence is Y. For spatially periodic configurations on linear CAs, one can easily see that such a path existence problem can be solved in nondeterministic logarithmic space. It is shown in [18] that the problem is indeed **NLOG**-complete.

Returning to infinite CAs, Golze [4] studied restrictions of the global map to various classes of configurations, namely finite configurations, spacially periodic configurations, and recursive configurations. It is shown there that for linear CAs any finite configuration that has a predecessor already has a periodic predecessor.

In this paper we will demonstrate that in any linear CA a recursive configuration with a predecessor must already have a recursive predecessor. Our proof is based on a slightly more general result about the existence of recursive paths in time-dependent graphs (see section 3). Note that there is no analogous result for dimensions larger than one. Indeed, it was shown by Hanf and Myers [5, 13] that one can construct sets of tiles that permit a tiling

of the whole plane, but any such tiling must be nonrecursive. It follows that for some two-dimensional CA the homogeneous target configuration consisting of 0's only has no recursive predecessor, but fails to be a Garden-of-Eden (see also [4]).

At any rate, even in one dimension the recursive predecessor in general cannot be effectively constructed from the target configuration. Even if it is known that there are only finite possible predecessors, all of which are recursive, it is impossible to choose the proper one algorithmically.

Note that a rough upper bound for the complexity of a predecessor of a recursive configuration can be derived from the Kreisel Basis Theorem: the class of all predecessors of a recursive configuration Y is Π_1^0. Hence, by the theorem, Y must have a Δ_2^0 predecessor (see [17]). Our results improve this upper bound to Δ_1^0.

Apart from the introduction this paper contains three sections. In section 2 we present quadratic time algorithms for injectivity and surjectivity. Section 3 contains the results for recursive predecessors. Lastly, in section 4 we summarize our results and pose some open problems. To keep this paper reasonably short we will not review the basic definitions from language theory and recursion theory. We refer the reader to references [7] and [17].

2. Definitions

We will mostly consider one-dimensional cellular automata. Thus a configuration of the automaton is a map $X : \mathbf{Z} \to \Sigma$ from the set of all cells to the alphabet. Here $\mathbf{Z}$ denotes the integers and Σ is the collection of states. For our purposes it is convenient to think of configurations as biinfinite words. For any alphabet Σ denotes by Σ^*, Σ^ω, ${}^\omega\Sigma$ and ${}^\omega\Sigma^\omega$ the collection of finite words, infinite words, coinfinite words, and biinfinite words over Σ, respectively (our ω notation is taken from [15]). For a word X let X_i be the ith symbol in X and for $-\infty \leq n \leq m \leq \infty$ let $X[n{:}\,m]$ be the subword $X_n X_{n+1} \ldots X_m$ of X. Define the s-fusion operation on words in Σ^* as follows:

$$u \odot v = w \iff \exists x \in \Sigma^s, u_0, v_0 \in \Sigma^* \Big(u = u_0 x, v = x v_0, w = u_0 x v_0 \Big)$$

Note that $\odot$ is a partial operation. The s-blocking operation on Σ^* is defined by $\beta(x) := (x[1{:}\,s], \ldots, x[n-s+1{:}\,n])$, where $n := |x|$. Similarly one defines fusion and blocking on infinite words. Since the appropriate s will always be clear from context, we have chosen not to burden our notation by displaying it, say as a subscript.

A linear CA may now be described as follows. A *local rule* is a map $\rho : \Sigma^{2r+1} \to \Sigma$. Here $r \geq 1$ is the radius of the rule. For $X \in {}^\omega\Sigma^\omega$ define the *global rule* (also denoted by ρ) by

$$\rho(X)(i) := \rho(\beta(X)_i)$$

An elegant representation of linear CAs uses labeled de Bruijn graphs (see [20]). To this end let Σ be an alphabet and $s \geq 1$ a number, and define

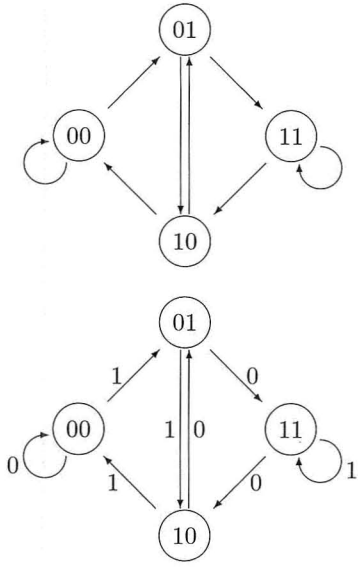

Figure 1: The de Bruijn graphs $B(2, \{0,1\})$ and B_{150} for rule 150.

the *de Bruijn graph* $B(s, \Sigma)$ as follows: $B(s, \Sigma)$ has vertex set Σ^s and edges (ax, xb) for all $a, b \in \Sigma$, $x \in \Sigma^{s-1}$. Figure 1 shows $B(2, \{0,1\})$.

Now consider a rule ρ of radius $r \geq 1$ over alphabet Σ. Define B_ρ to be the de Bruijn graph $B(2r, \Sigma)$, where edge (ax, xb) is labeled by $\rho(axb) \in \Sigma$. Note that this is equivalent to labeling edge (u, v) by $\rho(u \odot v)$. Here $\odot$ denotes $(2r-1)$-fusion. A biinfinite path U in B_ρ is but a biinfinite word over Σ^{2r}. We associate two biinfinite words over Σ, the *trace* of U and the *labeling* of U, with U as follows:

$$tr(U) = \bigodot U_i$$

$$lab(U) = \rho(U_i \odot U_{i+1})$$

It is clear from the definitions that $\rho(tr(U)) = lab(U)$. Thus B_ρ may be construed as a Fischer automaton (a nondeterministic finite automaton where every state is initial as well as final) that recognizes biinfinite words in $\rho({}^\omega\Sigma^\omega)$.

For the purpose of our injectivity testing algorithm, recall that a subset V_0 of a directed graph $G = \langle V, E \rangle$ is *strongly connected* if and only if for any two vertices x and y in V_0 there exists a path from x to y. A *strongly connected component* (SCC) is a maximal strongly connected subset. It is well known that the SCCs of a graph can be computed in time linear in the size of the graph (see [11]). After computing the SCCs, one can form a new directed graph $\mathcal{D}$, the collapse of G. $\mathcal{D}$ contains one node for each SCC of G. There is an edge in $\mathcal{D}$ from C to C' if and only if there is a path in G from some vertex in C to some vertex in C'. Plainly, $\mathcal{D}$ is acyclic. Hence $\mathcal{D}$

can be topologically sorte; that is, one can enumerate the vertices of $\mathcal{D}$ as $\{C_1, \ldots, C_k\}$ in such a way that whenever there is an edge (C_i, C_j) in $\mathcal{D}$ then $i < j$. Indeed, the sorting can again be accomplished in linear time (see [11]).

The topological sorting can be used to eliminate all *transient points* of G, that is, vertices that fail to lie on at least one biinfinite path in G. Call an SCC trivial if and only if its induced subgraph contains no edges. Observe that a vertex is nontransient if and only if it lies in a nontrivial SCC or it lies on a path from a nontrivial SCC to another. This gives rise to the following algorithm to delete all the nontransient points. First mark all the nontrivial SCCs in the collapse $\mathcal{D}$. Then sweep across the graph in order of the topological sort. Whenever a marked vertex is encountered, mark all its immediate successors. Upon completion of the sweep, delete all unmarked nodes. Then reverse all arcs in $\mathcal{D}$ and repeat the process. It is easy to see that upon completion of the second step exactly those SCCs survive that contain nontransient vertices of G. Both sweeps are linear in the size of G. For future reference, denote by $\overline{G}$ the graph obtained from G by deleting all transient points.

An interesting property of CAs is that they are reversible—in the sense that there exists another CA over the same alphabet that reverses the evolution of configurations on the first—if and only if the global function ρ is injective. This was shown in [16]. Since injectivity can be tested in quadratic time, we have the following theorem.

Theorem 2.1. *For one-dimensional cellular automata, reversibility is decidable in quadratic time.*

Proof. It follows from the previous remarks that rule ρ fails to be injective if and only if there are two distinct biinfinite paths V and U in the de Bruijn graph B_ρ with the same sequence of labels: $lab(V) = lab(U)$.

The latter condition can be tested as follows. Define a new graph B_ρ^2 that has vertices $\Sigma^{2r} \times \Sigma^{2r}$. Furthermore, $[(ax, by), (xa', yb')]$ is an edge in B_ρ^2 if and only if $\rho(axa') = \rho(byb')$. Also, let $\Delta := \{ (w, w) \mid w \in \Sigma^{2r} \}$ denote the diagonal in B_ρ^2. It is well known that de Bruijn graphs are Hamiltonian. Since Δ is an isomorphic copy of B_ρ, it follows that Δ is contained in one SCC C_Δ of B_ρ^2. As described in the remarks preceding the theorem, let $\overline{B_\rho^2}$ be the graph obtained from B_ρ^2 by removing all transient points.

Claim: Rule ρ is injective if and only if $\Delta = \overline{B_\rho^2}$.

Note that any two biinfinite paths U and V in B_ρ with the same labeling trace a biinfinite path in B_ρ^2 and hence in $\overline{B_\rho^2}$. The converse also holds.

To verify the claim, first suppose ρ fails to be injective. Then we can choose $U \neq V$ but with $\rho\big(tr(U)\big) = \rho\big(tr(V)\big)$, that is, with the same labeling in B_ρ. Then, however, the corresponding path in $\overline{B_\rho^2}$ cannot lie entirely within Δ, and it follows that $\Delta \neq \overline{B_\rho^2}$.

For the opposite direction suppose we have a biinfinite path $W = \big((V_i, U_i) \big)$ in $\overline{B^2_\rho}$, not entirely in Δ. Letting $V = (V_i)$ and $U = (U_i)$ it is clear that U and V are distinct, but $\rho\big(tr(V) \big) = \rho\big(tr(U) \big)$. Hence ρ is not injective and we are through. ∎

As pointed out above we can construct $\overline{B^2_\rho}$ in time linear in the size of B^2_ρ and therefore quadratic in the size of B_ρ. However, the size of the de Bruijn graph is precisely the same as the size of the local rule ρ, specified as a table. Thus, a one-dimensional rule ρ can be tested for injectivity in quadratic time.

Reversibility in the last lemma refers to injectivity of the global function for arbitrary configurations. One frequently studies the subclass of finite configurations, namely, biinfinite words of the form ${}^\omega 0 Z 0^\omega$, where $Z \in \Sigma^*$ and 0 is a special symbol. Let us call ρ *locally injective* if and only if, for all finite configurations X and Y, $\rho(X) = \rho(Y)$ implies $X = Y$. Let $X =^* Y$ iff X and Y disagree only in a finite number of places. It is straightforward to show that ρ is locally injective if and only if, for all configurations $X =^* Y$, $\rho(X) = \rho(Y)$ implies $X = Y$. It can be shown that local injectivity of ρ is equivalent to surjectivity of ρ (see [16] and [10]). The latter characterization is the basis for the next theorem.

Theorem 2.2. *For one-dimensional cellular automata, surjectivity of the global map is decidable in quadratic time.*

Proof. Let $\overline{B^2_\rho}$ be the nontransient part of the product de Bruijn graph as in the last proof. Recall that C_Δ denotes the SCC containing Δ.

Claim: Rule ρ is locally injective if and only if $\Delta = C_\Delta$.

To see this, first assume that ρ fails to be locally injective. Then by the remark preceding the theorem, there are two configurations $X \neq Y$ that disagree only in a finite number of places and map to the same configuration under ρ. Let W be the corresponding biinfinite path in $\overline{B^2_\rho}$. Plainly, W has an infinite initial segment and an infinite final segment in Δ. However, since $X \neq Y$ the path W cannot be entirely in Δ; consequently, $\Delta \neq C_\Delta$.

For the opposite direction suppose that $\Delta \neq C_\Delta$. Then there exists a biinfinite path W in $\overline{B^2_\rho}$ such that W has an infinite initial segment and an infinite final segment in Δ, but W does not lie entirely in Δ. W readily translates into two configurations that witness ρ's failure to be locally injective.

As in the last theorem it can be seen that the necessary computations take only quadratic time. ∎

Example. Consider the elementary rule ρ with code number 150 (addition modulo 2). The SCC of B^2_{150} containing Δ consists only of Δ itself. Indeed, there are exactly two SCCs: Δ, and all other vertices. Hence, rule 150 is locally injective but not injective. Similarly for rule 90 one can see that B^2_{90}

is partitioned into exactly three SCCs, one of them being Δ. Hence rule 90 is also locally injective but not injective.

Rule 30 produces a product graph with two nontrivial SCCs, one of them Δ and the other containing 8 vertices. Two vertices in B^2_{30} are transient, and the collapse of $\overline{B^2_{30}}$ has diamond shape. Again, rule 30 is locally injective but not injective.

As an example of an injective elementary CA, consider rule 15. B^2_{15} has 13 SCCs, but only one nontrivial SCC, namely Δ.

Lastly, B^2_{37} is strongly connected. Thus rule 37 is not even locally injective.

3. Recursive predecessors

In answer to a question posed by Golze [4] we now show that, in any linear cellular automata, a recursive configuration with a predecessor must already have a recursive predecessor. To demonstrate the existence of a recursive predecessor, we first establish a slightly more general result about the existence of recursive infinite paths in certain recursive graphs. The graphs under consideration are given by a finite static graph and a recursive function that describes the evolution of the static graph in time. A similar type of time-dependent graph was used in [19] to establish **PSPACE**-hardness of motion planning problems in the presence of moving obstacles.

A *time-dependent graph* $G(W)$ consists of a finite directed graph $G = \langle V, E\rangle$ and an alphabet $\Sigma \subset \mathcal{P}(E)$. For any word $W \in \Sigma^\omega$ the time-dependent graph $G(W)$ is defined as follows. The vertices of $G(W)$ are $V \times \mathbf{N}$, and the edges are given by $E(W) := \{ \big((p,t),(q,t+1) \big) \mid (p,q) \in W(t) \}$.

G is the underlying *static graph*. For later use let us define the reachable set of a vertex (p,t) at time s as

$$R((p,t),s) \;:=\; \{q \in V \mid \text{there exists a path from } (p,t) \text{ to } (q,s) \text{ in } G(W)\}$$

For a set P of points in $V \times \mathbf{N}$ let $R(P,s) = \bigcup_{x\in P} R(x,s)$. We will be interested mostly in the case where W construed as a map from $\mathbf{N}$ to Σ is a recursive function. $G(W)$ is then called a recursive time-dependent graph. Note that, in a recursive time-dependent graph, $R((p,t),s)$ is computable uniformly in p, s, and t.

Example. Let $G = \langle V, E\rangle$ be the complete directed graph on two points with self-loops; thus $V = \{0,1\}$ and $E = \{(0,0),(0,1),(1,0),(1,1)\}$. Define alphabet $\Sigma = \{S, B, L, R\}$ by $S := E$, $B := \{(0,0),(1,1)\}$, $L = \{(0,0)\}$, and $R = \{(1,1)\}$. Here S stands for switch, B for both, L for left, and R for right (see figure 2).

Now let $W \in \Sigma^\omega$ be an infinite word. A moment's thought reveals that $G(W)$ has an infinite path if and only if W does not contain a subword of the form LB^kR or RB^kL. Thus the collection of all time-dependent graphs

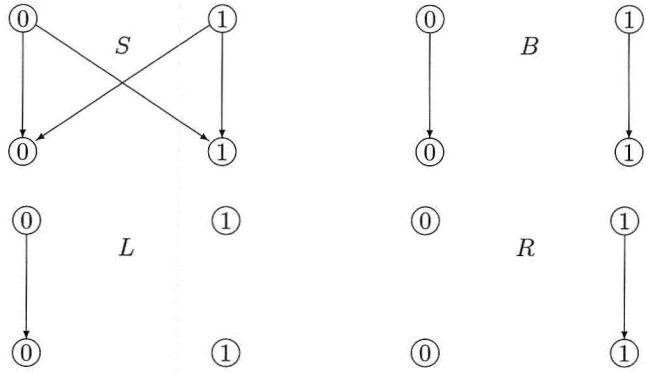

Figure 2: The four components of the time-dependent graph G.

over G with an infinite path forms a regular set in the sense of [15]. This is true for all time-dependent graphs.

We claim that for the static graph G defined as above any recursive time-dependent graph $G(W)$ with an infinite path has a recursive infinite path. To see this, note that we can distinguish the following two cases.

Case 1. *W contains only finitely many symbols S.* Then there is a recursive path of the form $P = P_0 i^\omega$ where P_0 is finite and $i = 0$ or $i = 1$.

Case 2. *W contains symbol S infinitely often.* In this situation we can obtain a recursive path by searching for the next occurrence of S and choosing, say, the lexicographically first finite path to that next occurrence of S. More precisely, suppose we have already constructed an initial segment of the recursive path P up to some time t such that $W(t) = S$. Then we can compute $t' > t$ minimal such that $W(t') = S$. Since $W[t\colon t'] \in S(B+L)^*S + S(B+R)^*S$, we can extend P up to time t'.

It should be noted, however, that the recursive path cannot be generated effectively from W; see also lemma 3.1 below.

Theorem 3.1. *Every recursive time-dependent graph with an infinite path has a recursive infinite path.*

Proof. Let $G(W)$ be a recursive time-dependent graph with an infinite path. For $t \geq 0$ define the *support* of t by $spt(t) := \{ P(t) \mid P \text{ an infinite path} \}$. A set $Q \subset V$ is *persistent* if and only if there are infinitely many t such that

$spt(t) = Q$. Since $G(W)$ contains an infinite path, we may pick a nonempty persistent set Q. Note that $\{ t \mid spt(t) \text{ not persistent} \}$ is necessarily finite. Hence there exists a $t_0 \geq 0$ such that $spt(t_0) = Q$, and for all $t \geq t_0$ the support of t is persistent. Let us define a predicate $P(r, s)$ on $\mathbf{N} \times \mathbf{N}$ as follows:

$$P(r,s) \iff \forall q \in R(Q \times \{r\}, s) - Q \exists t > s \left(R((q,s),t) = \emptyset \right)$$

Thus $P(r, s)$ holds if and only if no path in $G(W)$ starting at some node in Q at time r and passing through a node not in Q at time t has an infinite extension. Clearly P is recursively enumerable. Also note that the domain of P contains all times t such that $spt(t) = Q$. We have the following claim.

Claim: For all $t \geq t_0$, $P(t_0, t)$ if and only if $spt(t) = Q$.

First suppose that $P(t_0, t)$ holds. Since every infinite path must pass through Q at time t_0, no infinite path can fail to pass through Q at time t. All paths through $Q \times \{t_0\}$ and $(V - Q) \times \{t\}$ are finite by our assumption. Hence $spt(t) \subset Q$, and therefore $spt(t) = Q$ by the minimality of Q.

For the opposite direction let $spt(t) = Q$. Consider a point q not in Q, but with (q, t) reachable from $Q \times \{t_0\}$. If we had $\forall t' > t(R((q,t),t') \neq \emptyset)$ then, by compactness, there would be an infinite path starting at (q, t). However, q would then be in the support of t and hence in Q, which is a contradiction. Thus $P(t_0, t)$ holds as required. ∎

Using Σ_1-uniformization we may choose a partial recursive function f that uniformizes P. Since the domain of f is the same as the domain of P, it must include all times t such that $spt(t) = Q$. Define $t_n := f^n(t_0)$ for all $n \geq 0$. It follows from the claim that the sequence $(t_i)_{i \geq 0}$ is well defined. Also note that $(t_i)_{i \geq 0}$ is strictly increasing.

We can now construct a recursive infinite path through $G(W)$ as follows. First choose an initial segment from time $t = 0$ to a point p_0 in Q at time t_0. Suppose a part of the path up to (p_i, t_i) has been constructed, so $p_i \in Q$. Since $spt(t_i) = spt(t_{i+1}) = Q$, it follows that there exists a path from (p_i, t_i) to (p_{i+1}, t_{i+1}) for some $p_{i+1} \in Q$. Choose one such path, say the lexicographically first. Since the sequence $(t_i)_{i \geq 0}$ is computable, we obtain a computable infinite path through $G(W)$.

Theorem 3.2. *In any linear cellular automata the existence of a predecessor of a recursive configuration implies the existence of a recursive predecessor.*

Proof. In light of the last theorem it suffices to show how the existence of recursive predecessors can be translated into the existence of infinite recursive paths in a suitable time-dependent graph. As we saw in the last section, the existence of a predecessor of some configuration $X \in {}^{\omega}\Sigma^{\omega}$ is equivalent to the existence of an biinfinite path in the de Bruijn graph of the automaton. Now let B_ρ^r be the graph obtained from B_ρ by reversing all the edges. Clearly there is a biinfinite path labeled X in B_ρ if and only if for every decomposition

$X = X_0 \odot Z \odot X_1$, $X_0 \in {}^\omega\Sigma$, $Z \in \Sigma^{2r-1}$, $X_1 \in \Sigma^\omega$, there is an infinite path (one-way!) in B_ρ starting at Z labeled X_0, and an infinite path in B_ρ^r starting at Z labeled X_1.

Now fix some decomposition $X = X_0 \odot Z \odot X_1$. For any symbol $\sigma \in \Sigma$ let $E_\sigma := \{\, e \mid e \text{ an edge in } B_\rho \text{ labeled } \sigma \,\}$. Thus, for any word $Z \odot V \in \Sigma^\omega$, we have a time-dependent graph $B_\rho(Z \odot V)$ that contains an infinite path starting at Z if and only if there exists a biinfinite word X such that $\rho(X) = U \odot Z \odot V$. If X is recursive, so is $B_\rho(Z \odot V)$. Hence, by the last theorem there exists a recursive infinite path in $B_\rho(Z \odot V)$ if there exists any path at all.

The argument for the reverse graph B_ρ^r is entirely similar. Thus we have two recursive paths in the two time-dependent graphs that can be combined to produce one recursive biinfinite path in the original de Bruijn graph B_ρ. As we have seen, this path corresponds to a recursive predecessor of X. ∎

The proof of theorem 3.1 suggests that the recursive path cannot in general be obtained effectively from the target configuration. For example, it is not hard to show that persistence is not a recursive property. Thus one cannot hope to compute the parameter Q used in the definition of the sequence $(t_i)_{i\geq 0}$. The same holds for parameter t_0. The following lemma shows that indeed recursive predecessors in one-dimensional CAs cannot in general be constructed effectively.

Lemma 3.1. *Let ρ be a linear cellular automaton. Then in general there is no effective method to determine a recursive predecessor for a recursive configuration with a predecessor.*

Proof. It is routine to construct a linear CA with alphabet $\Sigma = \{a, b, c, d\}$ such that

$$\begin{aligned} \rho({}^\omega aba^\omega) &= {}^\omega aba^\omega \\ \rho({}^\omega cdc^\omega) &= {}^\omega aca^\omega \\ \rho({}^\omega a^\omega) &= \rho({}^\omega c^\omega) = {}^\omega a^\omega \end{aligned}$$

All other blocks are mapped to d. For $e \geq 0$, define a word $Y_e \in {}^\omega\Sigma^\omega$ as follows.

$$Y_e(n) := \begin{cases} b & \text{if } n \leq 0 \text{ is minimal such that } \{e\}_n(e) \simeq 0, \\ c & \text{if } n \leq 0 \text{ is minimal such that } \{e\}_n(e) \simeq 1, \\ a & \text{otherwise} \end{cases}$$

Here $\{e\}_n(x) \simeq y$ means the eth partial recursive function on input x converges after n steps with output y. Similarly $\{e\}(x) \simeq y$ means the eth partial recursive function on input x converges on output y. To determine whether $\{e\}(e) \simeq y$ for some y is the Halting problem and well known to be undecidable (see [17]). Note that $Y_e = {}^\omega a^\omega$, $Y_e = {}^\omega aba^\omega$, or $Y_e = {}^\omega aca^\omega$, depending on whether a certain computation diverges, converges on 0, or converges on 1.

Thus every configuration Y_e has a recursive predecessor. Now suppose that such a predecessor (or rather, an index thereof) could be computed effectively. Then the set $A := \{ e \mid X(0) = a,\ X \text{ the predecessor of } Y_e \}$ would be recursive. By the S_m^n theorem there is a primitive recursive function p such that $\{p(e)\}(x) \simeq 1$ for all x if and only if $\{e\}(e)$ converges and $\{p(e)\}(x)$ diverges for all x otherwise. But then $\{e\}(e)$ converges if and only if $p(e) \notin A$. It follows from the undecidability of the Halting problem that A is not recursive, and we have the desired contradiction. ∎

Note that the configurations Y_e from above provide a simple proof for the assertion that in two-dimensional CAs a recursive target configuration may fail to have a recursive predecessor, but still not be a Garden-of-Eden. To this end one arranges all these configurations into a two-dimensional target configuration, say parallel to the x-axis in the upper half-plane and separated by blanks. Any predecessor of this configuration restricted to the y-axis is essentially the characteristic function of the diagonal set $K = \{ e \mid \{e\}(e) \text{ converges} \}$. Hence no predecessor can be recursive.

4. Conclusion

We have shown that reversibility of a one-dimensional cellular automaton is decidable in quadratic time using standard graph theoretic algorithms. Similarly one can decide in quadratic time whether the limit set of the automaton is the whole space of configurations. Both properties are known to be undecidable even in two dimensions (see [9]).

Our proof shows that one can determine in polynomial time whether certain restricted nondeterministic automata accept every string over their input alphabet. The automata under consideration here have the special property that all states are initial and final. Such devices are usually referred to as Fischer automata. We do not know whether acceptance of all inputs can be tested in polynomial time for arbitrary Fischer automata. For arbitrary nondeterministic automata the problem is known to be **PSPACE**-complete.

By unfolding de Bruijn graphs into an associated infinite graph, we have shown that all recursive configurations with an arbitrary predecessor must already have a recursive predecessor. However, the recursive predecessor in general cannot be effectively computed from the target configuration. Again, there are no analogous results for two-dimensional CAs; in a two-dimensional CA even a trivial target configuration may have only nonrecursive predecessors (see [5, 13]).

References

[1] S. Amoroso and Y. N. Patt, "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tesselation Structures," *Journal of Computers and System Science*, **6** (1972) 448–464.

[2] A. W. Burks, *Essays on Cellular Automata* (University of Illinois Press, Urbana-Champaign, Illinois, 1970).

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability* (Freeman, 1979).

[4] U. Golze, "Differences between 1- and 2-Dimensional Cell Spaces," in *Automata, Languages and Development*, edited by A. Lindenmayer and G. Rozenberg, pp. 369–384 (North-Holland, 1976).

[5] W. Hanf, "Nonrecursive Tilings of the Plane *I*," *Journal of Symbolic Logic*, **39**(2) (1974) 283–285.

[6] T. Head, "Linear Cellular Automata: Injectivity from Ambiguity," forthcoming.

[7] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, 1979).

[8] K. Culik II, "On Invertible Cellular Automata," *Complex Systems*, **1**(6) (1987) 1035–1044.

[9] J. Kari, "The Undecidability of Injectivity and Surjectivity of Two-Dimensional CAs," CA'89, Los Alamos, New Mexico.

[10] A. Maruoka and M. Kimura, "Injectivity and Surjectivity of Parallel Maps for Cellular Automata," *Journal of Computers and System Science*, **18** (1979) 47–57.

[11] K. Mehlhorn, *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness* (Springer-Verlag, Berlin, 1984).

[12] E. F. Moore, "Machine Models of Self-Reproduction," in *Essays on Cellular Automata* [2].

[13] D. Myers, "Nonrecursive Tilings of the Plane *II*," *Journal of Symbolic Logic*, **39**(2) (1974) 286–294.

[14] J. Myhill, "The Converse of Moore's Garden-of-Eden Theorem," in *Essays on Cellular Automata* [2].

[15] M. Nivat and D. Perrin, "Ensembles Reconnaisable de Mots Biinfinis," Technical Report 84–68, Université Paris 7.

[16] D. Richardson, "Tesselation with Local Transformation," *Journal of Computers and System Science*, **6** (1972) 373–388.

[17] J. R. Shoenfield, *Mathematical Logic* (Addison-Wesley, 1967).

[18] K. Sutner, "The Complexity of Finite Cellular Automata," forthcoming.

[19] K. Sutner and W. Maass, "Motion Planning among Time-Dependent Obstacles," *Acta Informatica*, **26** (1988) 93–122.

[20] S. Wolfram, "Computation Theory of Cellular Automata," *Communications in Mathematical Physics*, **96**(1) (1984) 15–57.