

The Complexity of Reversible Cellular Automata

K. Sutner

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

Abstract. We study the orbits of reversible one-dimensional cellular automata. It is shown that the Turing degree structure of the orbits of these automata is the same as for general cellular automata. In particular there are reversible cellular automata whose orbits have arbitrary recursively enumerable degree.

1 Computation in Cellular Automata

It was shown by Bennett [1] that reversible Turing machines can compute any partial recursive function. More precisely, given a partial recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ the function $\hat{f}(x) = \langle x, f(x) \rangle$ can be computed by a reversible Turing machine where $\langle \cdot, \cdot \rangle$ is any effective pairing function. If f happens to be injective then f itself can be computed by a reversible Turing machine. Morita demonstrated that the same holds true for one-dimensional cellular automata [9, 7], even in the sense that any irreversible cellular automaton can be simulated by a reversible one on finite configurations. The focus in the references is on the use of Turing machines and cellular automata as transducers, machines that convert input into output, at least if the input belongs to the domain of the corresponding partial recursive function. The orbits of the instantaneous descriptions coding these inputs under the action of the Turing machine are finite; they end when a certain halting state is reached. Davis [4] proposed to use finite orbits of instantaneous descriptions to define universality of Turing machines: a Turing machine is universal if the collection of instantaneous descriptions with finite orbits is recursively enumerable complete. There are two distinct advantages of this approach when one considers generalizations to realm of cellular automata. First, it bypasses the issue of input/output conventions and, second, one deals with a larger set of configurations, rather than just the ones that directly code steps in a computation.

Thus we will consider the complete (forward) orbit

$$\text{Orb}_\rho = \{ (X, \rho^t(X)) \mid t \geq 0, X \text{ finite configuration} \}$$

of a given cellular automaton with global map ρ . We adopt as measure of the complexity of ρ the Turing degree of Orb_ρ . In particular a universal cellular automaton has to have a recursively enumerable complete orbit. Note that in this case the orbit can be used to compute an arbitrary partial recursive function, albeit in a somewhat more complicated fashion than with a standard simulation. More precisely, given a partial recursive function f and some argument x we can first use the orbit as an oracle to determine whether x lies in the domain of f , and, if so, use repeated queries to find the y such that $f(x) = y$. The second step exploits the fact that $f^{-1}(y)$ is recursively enumerable uniformly in y .

One can also think of the complete orbit as a decision problem, the Reachability Problem for ρ : given two configurations X and Y determine whether Y lies in the orbit of X . A closely related problem is Confluence: given two configurations X and Y determine whether the orbits of X and Y overlap. Both problems are naturally recursively enumerable (r.e. for short) regardless of the cellular automaton in question. It is shown in [15] that one can select the difficulty of the Confluence and Reachability problems arbitrarily within the r.e. degrees for one-dimensional cellular automata.

Theorem 1. *For any two recursively enumerable degrees d_1 and d_2 there is a one-dimensional cellular automaton whose Reachability Problem is of degree d_1 and whose Confluence Problem is of degree d_2 .*

It is clear that theorem 1 cannot hold for reversible cellular automata: X and Y are confluent if, and only if, one configuration appears in the orbit of the other. Thus the degree of the Reachability problem is an upper bound for the degree of the Confluence problem for any reversible cellular automaton. It follows that erasing information is an indispensable component in some of the constructions dealing with the complexity of orbits of cellular automata. On the other hand, we will show that the Reachability problem alone can have arbitrary r.e. degree, so that the difficulties one encounters in the classification of general cellular automata is fully reflected in the smaller class of reversible cellular automata.

In section 2 we will give a brief review of Bennett's and Morita's constructions and show how to modify them to obtain a reversible cellular automaton whose Reachability problem is of arbitrary r.e. degree. We close with a few open problems in section 3. For background in recursion

theory we refer the reader to the literature, in particular [6] or [11]. Our notation is compatible with the last reference.

2 Reachability in Reversible Cellular Automata

We start with a brief review of the construction of a reversible universal Turing machine and the analogous construction of a reversible cellular automaton that can simulate a universal Turing machine in a simple manner. Details concerning the physical relevance of reversibility in computation and a discussion of time/space trade-offs involved in reversible computation can be found in [2, 3]. In [8] Morita gives an overview of results on reversible cellular automata.

2.1 Constructing Reversible Cellular Automata

The construction of a reversible Turing machine by Bennett [1] uses a history tape that records all moves of a given Turing machine. When the machine halts, the output is copied to a special output tape, and then the history is used to undo the computation. In the end, we are left with only the input and the output of the computation. Since we are interested in acceptors rather than transducers we can avoid Bennett's copying trick and revert to Lecerf's method [5]: we undo the computation and are left solely with the input, provided that the input is accepted by the machine. Otherwise, the simulation fails to terminate.

Suppose M is a Turing machine on state set Q and tape alphabet Γ . As is customary, we think of an instantaneous description (ID) of a Turing machine M as a word in $\Gamma^*Q\Gamma^*$ where Γ denotes the tape alphabet of the Turing machine, including the blank symbol, and Q its state set. We assume that these expressions are uniquely parsable. We write ID_M for the collection of all IDs of machine M and $I \rightarrow_M J$ to indicate that M acts on ID I and produces J in one step. Reversibility of a Turing machine is usually defined as a local property of the transition function of the machine, see [1, 8]: one has to make sure that a given state p and tape symbol a can have at most one predecessor state and symbol. As a consequence, any ID has at most one predecessor under \rightarrow_M . For the construction of reversible Turing machines it is advantageous to represent the transitions of the Turing machine in a slightly non-standard way as follows, see [1]. The transition function comes in three parts. There are read/write transitions $\delta : Q \times \Gamma \rightarrow Q \times \Gamma$, left-shift transitions $\delta_L : Q \rightarrow Q$ and right-shift transitions $\delta_R : Q \rightarrow Q$. All these functions

are partial and we insist that their domains do not overlap with respect to states.

Machine M acts on IDs in the natural way. For example, for any read/write transition $\delta(p, a) = (q, b)$ we have

$$a_1 a_2 \dots a_n p a b_1 b_2 \dots b_m \rightarrow_M a_1 a_2 \dots a_n q b b_1 b_2 \dots b_m$$

and for any left-shift $\delta_L(p) = q$ we get

$$a_1 a_2 \dots a_n a p b_1 b_2 \dots b_m \rightarrow_M a_1 a_2 \dots a_n q a b_1 b_2 \dots b_m$$

Likewise a right-shift moves the head to the right, ignoring the tape inscription. It is not hard to see that the Turing machine acts reversibly on ID_M if, and only if, the read/write, left-shift and right-shift functions are injective and if their ranges are disjoint with respect to states. Thus, unlike with cellular automata, there is a simple local condition to test reversibility of Turing machines. Also, unlike with cellular automata, the action of the Turing machine on its IDs may well be partial.

By keeping a history of the computation and writing the transitions in the special format just described to verify reversibility, one can now establish the following theorem.

Theorem 2 (Lecerf, Bennett). *For any r.e. set W there is a reversible Turing machine that accepts W .*

To establish an analogue of Lecerf's and Bennett's theorem for Turing machines in the realm of one-dimensional cellular automata, Morita uses a special type of cellular automaton called a *partitioned cellular automaton (PCA)*, see [9, 7].

Theorem 3 (Morita, Harao). *For any reversible Turing machine there is a reversible one-dimensional cellular automaton that simulates the Turing machine.*

It follows that some reversible one-dimensional cellular automaton will have certain orbits whose degree is any chosen r.e. degree. To extend this result to all orbits we have to modify the construction so as to insure that the cellular automaton cannot perform computations of unintended complexity. For our purposes, it suffices to consider cellular automata of width 3, defined in terms of a *local map* or *local rule* $\rho : \Sigma^3 \rightarrow \Sigma$ where Σ is the alphabet of the automaton. We define the set of *configurations* of ρ to be $\mathcal{C} = \Sigma^\infty$, the set of all biinfinite words over Σ . The local map

acts on \mathcal{C} via the *global map* $\rho^\infty(X)(i) = \rho(X(i-1), X(i), X(i+1))$. We will abuse notation and denote the global map simply by ρ . Since we are interested in the computational complexity of the orbits of the global map we will focus on *finite configurations*, i.e., configurations that differ from a special *quiescent configuration* $X_q(i) = q$, where $q \in \Sigma$ is fixed, in only finitely many places. X_q is required to be a fixed point of ρ . We denote the collection of all finite configurations \mathcal{C}_f .

One can think of a PCA as a cellular automaton whose cells are divided into multiple tracks; specifically Morita uses an alphabet of the form $\Sigma = \Sigma_1 \times \Sigma_2 \times \Sigma_3$. The configurations can be written as (X, Y, Z) where $X \in \Sigma_1^\infty$, $Y \in \Sigma_2^\infty$ and $Z \in \Sigma_3^\infty$. Now consider the shearing map σ defined by $\sigma(X, Y, Z) = (\text{RS}(X), Y, \text{LS}(Z))$ where RS and LS denote the right and left shift, respectively. Given any function $f : \Sigma \rightarrow \Sigma$ we can define a map $\rho : \mathcal{C} \rightarrow \mathcal{C}$ by $\rho = f \circ \sigma$ where f is assumed to be applied point-wise. Since the shearing map is bijective, ρ will be bijective if, and only if, f is so bijective. The map ρ can be construed as the composition of the global maps of a cellular automaton of width 3 with a trivial cellular automaton of width 1, and so is indeed the global map of a cellular automaton of width 3.

To describe such PCAs it suffices to specify a collection of rewrite rules of the form

$$\langle x, y, z \rangle \rightarrow \langle x', y', z' \rangle,$$

where $x, x' \in \Sigma_1$, $y, y' \in \Sigma_2$, and $z, z' \in \Sigma_3$, as long as the right hand sides of these rules are distinct. The corresponding local change in a configuration is of the form

$$\begin{array}{|c|c|c|c|c|c|} \hline \cdot & \cdot & x & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & y & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & z & \cdot \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|} \hline \cdot & \cdot & \cdot & x' & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & y' & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & z' \\ \hline \end{array} \quad (1)$$

More precisely, given a partial injective function $f : \Sigma \rightarrow \Sigma$ the functional digraph of f consists of a collection of cycles and paths, possibly of length 0. We can extend f to a bijection f^c , the *completion* of f , by adding an edge to each maximal path that connects the target to the source of such a path. For paths of length 0, corresponding to triples neither in the domain nor range of f , this means that all such triples will be fixed points of the completion. We will refer to any triple not in the domain of f as *improper*.

As a simple example of this construction we can define a reversible cellular automaton that simulates particles bouncing between domain walls. The alphabets are $\Sigma_1 = \{0, r\}$, $\Sigma_2 = \{0, l, L, r, R, W\}$ and $\Sigma_3 = \{0, l\}$. The map f is given by the following table.

$$\begin{array}{l|l}
\langle 0, 0, 0 \rangle \rightarrow \langle 0, 0, 0 \rangle & \langle 0, l, 0 \rangle \rightarrow \langle 0, 0, l \rangle \\
\langle 0, W, 0 \rangle \rightarrow \langle 0, W, 0 \rangle & \langle r, W, 0 \rangle \rightarrow \langle 0, L, 0 \rangle \\
\langle r, 0, 0 \rangle \rightarrow \langle 0, r, 0 \rangle & \langle 0, W, l \rangle \rightarrow \langle 0, R, 0 \rangle \\
\langle 0, r, 0 \rangle \rightarrow \langle r, 0, 0 \rangle & \langle 0, R, 0 \rangle \rightarrow \langle r, W, 0 \rangle \\
\langle 0, 0, l \rangle \rightarrow \langle 0, l, 0 \rangle & \langle 0, L, 0 \rangle \rightarrow \langle 0, W, l \rangle
\end{array}$$

As figure 1 shows, the global map handles collisions between particles properly despite the fact that no explicit provisions were made in f for this case. One should note, though, that on random configurations, unlike the carefully chosen initial conditions in the figure, the behavior of this RCA becomes somewhat more complicated. For example, the typical period length is exponential using cyclic boundary conditions whereas configurations as in the figure have periods linear in the size of the configuration.

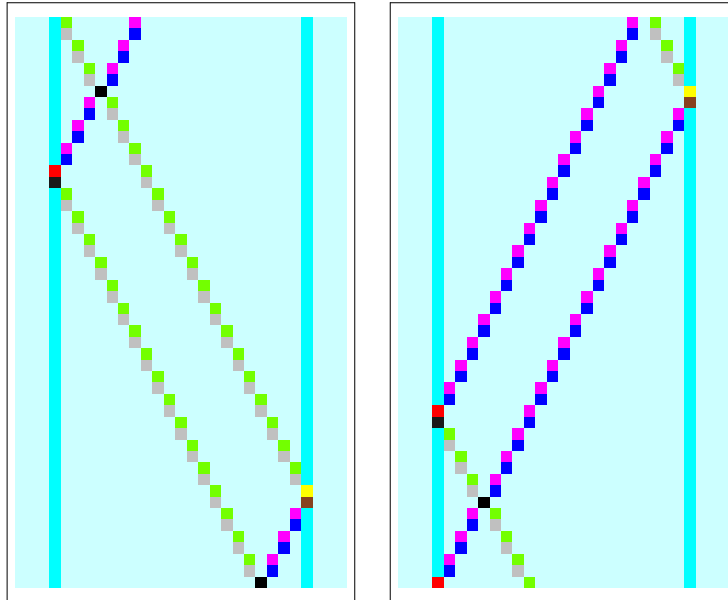


Fig. 1. A reversible cellular automaton simulating particles bouncing between domain walls.

2.2 Stable Reversible Turing Machines

When dealing with all orbits of a cellular automaton a direct simulation of a reversible Turing machine using Morita's approach meets with two principal obstructions. First, since we are dealing with all possible finite configurations of the RCA rather than just the ones that correspond to a step in computation of the Turing machine in some obvious way, we have to make sure the orbits of unintended configurations do not affect the degree of Orb_ρ . Second, even if we start with a reversible Turing machine that somehow addresses the first issue, we have to define a cellular automaton whose global map is injective on all configurations without affecting the simulation of the Turing machine.

We will first contend with the problem of constructing a suitable reversible universal Turing machine. Let M be an arbitrary Turing machine accepting some r.e. set $W \subseteq \mathbb{N}$. We may safely assume that M starts its computation on input $x = x_1x_2 \dots x_n$ in ID $I_x = q_0 \cdot x_1x_2 \dots x_n$, meaning that the head is positioned at the last blank square before the input (\cdot will be used throughout to denote a blank symbol). If x is accepted, M winds up in the halting ID $H_x = q_H \cdot$ and stops; moreover, the tape head at this point is in the same position as in I_x . Certainly any Turing machine can be primitive recursively transformed into another that satisfies these constraints.

Call an ID of a Turing machine *admissible* if it lies in the orbit of some initial ID I_x under the action of the machine. Inadmissible IDs may cause problems in simulations that attempt to preserve the degree of the acceptance set of the Turing machine, since they may give rise to computations of higher complexity. It is easy to see that the collection of admissible IDs may well be r.e.-complete, so there is no effective way of eliminating inadmissible IDs. However, one can still avoid inadmissible IDs in the sense that one can force all their orbits to be finite. For example, it was shown by Davis in [4] that any total recursive function can be computed by a Turing machine M that halts on all its IDs. Davis' proof uses a special representation of total recursive functions, rather than special Turing machines as in the following argument.

Since we are dealing with acceptors we define a Turing machine to be *stable* if the orbit of any inadmissible ID is finite, see [12, 13] for an application of stability to classification problems of cellular automata. The references contain a careful discussion, including coding details, of a construction similar to the one in the next lemma, so we will omit details here. Since we are interested in reversible machines the construction can

be somewhat simplified by using a history mechanism similar to Bennett's method rather than the time stamps used in the references.

Lemma 1. *For any recursively enumerable set W there is a stable reversible Turing machine that accepts W .*

Proof. Start with a Turing machine M accepting W as described above. We construct a new machine M' that simulates M in a circuitous fashion by maintaining the history of the computation performed so far, and by executing repeated checks if the history really describes the alleged corresponding ID. For input x the initial tape inscription of machine M' for input x has the form

$$\$_0\$ x \$ I_x \$_1 \tag{2}$$

where I_x is the initial ID of x for M and the head is positioned on separator symbol $\$_0$. The part between $\$_0$ and $\$$ will be used to record the history of the computation of M on x . In general, the tape inscriptions of M' look like so:

$$\$_0H\$ x \$ I \$_1 \tag{3}$$

The markers $\$$ are stationary, but $\$_0$ and $\$_1$ can be moved to accommodate growing and shrinking blocks of symbols. The tape head sweeps back and forth between the two outermost markers. Of course, one can avoid the use of actual endmarkers by the usual coding conventions at the cost of introducing more states in M' . We may assume without loss of generality that the degree of irreversibility of M is no higher than 2, meaning that there are at most two predecessor IDs for each ID of M . In this case a single bit suffices to disambiguate any transition of the Turing machine, so we may use a block of 0's and 1's as history. One can think of $\$_0H\$$ as a binary stack: at each move of the Turing machine M a bit is pushed onto this stack (for non-ambiguous moves we push 0 by default).

M' works in two main phases, compute and undo, and the states used during these phases are distinct. In the compute phase, rather than just simulating M step by step, keeping track of the history of transitions used, after each simulation step M' enters a verification procedure. During verification a copy of the history stack and ID is made in the space to the right of $\$_1$. Then M' attempts to unravel ID I popping bits off the copied stack to disambiguate the steps of M . If, in the end, $I_x \in \text{ID}_M$ is reached when all the history bits have been used up, verification succeeds, we remove all non-blank entries in the scratch-space and perform the next simulation step. If verification fails M' halts without accepting.

If, at some point, M reaches a halting ID, M' enters its undo phase during which the history bits are removed and used to run the computation of M leading to the halting ID backward. Unlike with the compute phase, the actual history and ID are used rather than copies. When inscription (2) is reached, M' also halts.

It is easy to see that M' accepts W . At bit more tedious is to check that M' is indeed reversible. The crucial point is that whenever a single step of M is simulated in the computation phase, the state of M' carries information about which transition was used, and pushes the appropriate bit onto the history stack, thus maintaining reversibility.

Lastly, we have to make sure that M' is stable. To this end consider an inadmissible ID J of M' . If the tape inscription is not of the form (3) or if the tape head is not located somewhere between $\$0$ and $\$1$ the orbit of $J \in \text{ID}_{M'}$ will be finite since the sweeping tape head will, after finitely many steps, encounter an inappropriate tape symbol, including possibly the blank symbol, and stop without accepting. So suppose J is of the proper form as in (3) but still inadmissible. If the state of M' belongs to the compute phase, a verification procedure must be initiated after finitely many steps. Since J is inadmissible there must be a mismatch between the alleged history and the ID of M , so that M' halts. If M' is in its undo phase the same argument applies. Hence M' is stable and we are done. \square

By adding Bennett's trick of copying the output to another tape before the computation is reversed, a similar construction can be used to compute recursive functions reversibly.

Corollary 1. *For any partial recursive function $g : \mathbb{N} \rightarrow \mathbb{N}$ there exists a stable reversible Turing machine that computes the function $\hat{g}(x) = \langle x, g(x) \rangle$.*

Of course, there is no need to use an actual pairing function here, one can simply adopt the convention that the output is represented by the ID $q_h \cdot x \cdot g(x)$. If g is already injective, x can be omitted.

2.3 Degrees of Reversible Cellular Automata

We are now ready to show that the Reachability problem for reversible cellular automata can assume any r.e. degree.

Theorem 4. *For any r.e. degree d there is a one-dimensional reversible cellular automaton whose complete orbit has degree d .*

Proof. Let M be a reversible stable Turing machine accepting an r.e. set W of degree d as in lemma 1. We will use a 3-track cellular automaton to simulate M , though it will be convenient to think of the top and bottom tracks as being divided into two sub-tracks: one of the sub-tracks is dedicated to simulating the Turing machine, and the other serves as a signal channel. For the time being, we ignore the signal channels.

The track alphabets are then $\Sigma_1 = Q_R \cup \{\bullet, \square\}$, $\Sigma_2 = \Gamma \cup Q\Gamma \cup \{\square\}$ and $\Sigma_3 = Q_R \cup \{\bullet\}$. As before, symbol \bullet denotes the blank symbol for the Turing machine and \square can be construed as a special signal that is used to extend accepting computations to be two-way infinite. The transitions of the Turing machine are expressed in the cellular automaton by the rules

$$\begin{aligned} \langle \bullet, pa, \bullet \rangle &\rightarrow \langle \bullet, qb, \bullet \rangle && \text{read/write transition } \delta(p, a) = (q, b) \\ \langle \bullet, pa, \bullet \rangle &\rightarrow \langle q, a, \bullet \rangle && \text{right-shift transition } \delta_R(p) = q \\ \langle q, a, \bullet \rangle &\rightarrow \langle \bullet, qa, \bullet \rangle && \\ \langle \bullet, pa, \bullet \rangle &\rightarrow \langle \bullet, a, q \rangle && \text{left-shift transition } \delta_L(p) = q \\ \langle \bullet, a, q \rangle &\rightarrow \langle \bullet, qa, \bullet \rangle && \end{aligned}$$

To avoid having to deal with computations that are finite sequences, add the rules

$$\begin{aligned} \langle \square, \square, \bullet \rangle &\rightarrow \langle \bullet, q_0 \bullet, \bullet \rangle && q_0 \text{ initial state} \\ \langle \bullet, q_H \bullet, \bullet \rangle &\rightarrow \langle \square, \square, \bullet \rangle && q_H \text{ final state} \\ \langle \square, a, \bullet \rangle &\rightarrow \langle \square, a, \bullet \rangle && \\ \langle \bullet, \bullet, \bullet \rangle &\rightarrow \langle \bullet, \bullet, \bullet \rangle && \text{quiescence} \end{aligned}$$

The last two rules are not superfluous despite our convention about f^c , see below. Thus, \square acts like a seek-right operation that finds a specially marked position on the “tape” and then initiates the actual computation of the Turing machine. Assuming the input is accepting, the marker is reinserted, and \square will move further to the right in the top track upon completion of the computation. Let f be the partial function from $\Sigma = \Sigma_1 \times \Sigma_2 \times \Sigma_3$ to itself, corresponding to the local rules, D its domain of definition, and f^c its completion.

In the actual construction we replace Σ_1 by $\Sigma_1 \times \{0, 1\}$ and likewise Σ_3 by $\Sigma_3 \times \{0, 1\}$. We will write the extra bits as subscripts. The local rules from above then take the form

$$\begin{aligned} \langle \bullet_u, pa, \bullet_v \rangle &\rightarrow \langle \bullet_u, qb, \bullet_v \rangle \\ \langle \bullet_u, pa, \bullet_v \rangle &\rightarrow \langle q_u, a, \bullet_v \rangle \end{aligned}$$

and so forth. For any of the triples $\langle x, y, z \rangle$ not in D we set

$$\langle x_u, y, z_v \rangle \rightarrow \langle x'_u, y', z'_u \rangle$$

where $f^c(x, y, z) = (x', y', z')$. Thus, the signal bits in the top and bottom sub-tracks remain unchanged whenever one of the local replacements is performed that are determined by f . However, when a triple not in D occurs, the signal bits from the top and bottom sub-tracks are interchanged. For example, suppose two heads are trying to move into the same tape cell in a configuration representing an inadmissible ID. Then we have

$$\langle p_u, a, q_v \rangle \rightarrow \langle p_v, a, q_u \rangle$$

For any input $x = x_1x_2\dots x_n$ to the Turing machine define the configuration C_x , suppressing signal bits, to be

$$\dots \begin{array}{|c|c|c|c|c|c|c|c|} \hline \square & \square & \square & \square & \dots & \square & \square & \square \\ \hline \square & q_0 & x_1 & x_2 & \dots & x_n & \square & \square \\ \hline \square & \square & \square & \square & \dots & \square & \square & \square \\ \hline \end{array} \dots$$

The signal bits in the top track are all 0, and in the bottom track they are all 1. We consider $\langle \bullet_0, \bullet, \bullet_1 \rangle$ to be the quiescent symbol of the RCA just defined, so that C_x is indeed a finite configuration.

Any configuration that appears in the two-way orbit of C_x , x any input, will be called *admissible*. An easy induction shows that the triples appearing in any admissible configuration will always lie in the domain of f . Thus, in the orbit of an admissible configuration, the signal bits are never interchanged and undergo a simple left/right shift at every step.

Claim: The Reachability problem has degree d .

Let x be some natural number and C_x and C_x^H the corresponding initial and accepting configurations of ρ . Then $x \in W$ if, and only if, $(C_x, C_x^H) \in \text{Orb}_\rho$ so that $W \leq_T \text{Orb}_\rho$.

For the opposite direction consider two arbitrary finite configurations X and Y . Using W as an oracle we have to determine whether Y appears in the orbit of X . For the sake of simplicity we will only consider the case where X contains a single contiguous block of non-quiescent symbols; the argument easily carries over to the general case. We distinguish several cases depending on the number of appearances of state symbols $p \in Q$ in X . To avoid extra cases, a symbol \square in the top track should be considered a state in this context. An inspection of the rule table for our RCA shows that the number of state symbols is preserved under application of the global map. Moreover, in the absence of any state symbols, the global map simply acts as a right-shift on the top track and a left-shift on the bottom track; it is the identity on the middle track. To see this recall that

the completion f^c of f is defined to be the identity on all triples outside of the domain and range of f .

When there is exactly one state symbol, X may lie in the two-way orbit of some configuration C_x . We can decide whether this is indeed the case, and determine the corresponding input x , by evolving X until a verification phase has been passed successfully, or until the undo phase is successfully completed. If there is a failure at some point, machine M stops without accepting, in which case the argument from the no-state case above applies, even if we ignore the signal bits in the top and bottom track. Otherwise we can use our oracle to test whether $x \in W$. If so, the orbit of X is trivial (a signal \square moving to the right) after a finite initial segment, corresponding to the actual computation of M on x . Otherwise the computation diverges, meaning that the history part of the inscription also grows indefinitely. In either case we can easily decide whether Y appears in the orbit of X .

For the last case assume that there are at least two state symbols in X . We may safely assume that at least two state symbols appear in the same region between the endmarkers of M . Otherwise the evolution proceeds much as in the one-state case, though there may ultimately be a collision between expanding regions representing computations of M . If such a collision occurs an improper triple is generated and the signals in the outer tracks are interchanged, which, as we will see shortly, suffices to make the orbit of X decidable.

So suppose we have at least two states in one block of X representing a tape inscription of M , positioned between the endmarkers. A priori, multiple state symbols might interact in an unintended way to produce orbits of degree $d \not\leq W$. To see that this cannot happen, note that the tape heads in M sweep out the region between the markers. Hence unboundedly many improper triples must appear in the orbit, either because of a collision between the state symbols, or because of state/symbol combinations for which the transition function is undefined. Now consider the signal bits in the top track that are being shifted to the right by the action of the global map. The bits originating in the quiescent part of X to the left of the non-quiescent block are all 0 by definition. When one of these bits appears in an improper triple in some configuration $\rho^t(X)$ in the orbit, it is switched to the lower track and now travels to the left. Note that it may happen that, at some later time, the bit appears in another improper triple and is switched back up. Furthermore, if the signal bit in the top track was 1 at this time, the bits would be restored to the same order as in the quiescent state. By considering the light-cones of all

switching events one can see that there has to be an event whose light-cone has a left edge that does not intersect any of the other light-cones. The 0 bit in the lower track from this event will escape into the quiescent region, and will henceforth move to the left indefinitely. But then membership in the orbit of X is decidable and we are through. \square

As the proof shows, the orbits of the RCA are all decidable except for those that correspond to actual computations of the simulated Turing machine M . It is only those orbits that force the Reachability problem to have the chosen r.e. degree.

3 Conclusion and Problems

We have shown that reversible cellular automata are indistinguishable from general cellular automata as far as the Turing degrees of their orbits on finite configurations are concerned. Davis suggested in [4] to use the set of all configurations that have finite orbits as a measure for the complexity of a Turing machine and in particular to define a notion of universal Turing machine. We do not know if an analogous approach is possible for reversible cellular automata: can the set of configurations that have finite orbits as sets be used to distinguish universal automata?

Another interesting question is the complexity of the orbits of reversible cellular automata on spatially periodic configurations. It is known for general one-dimensional cellular automata that it is undecidable whether for some fixed k all spatially periodic configurations of length n evolve to a limit cycle of length $O(n^k)$, for all n , see [13]. In the general case this holds even for $k = 0$. It is unknown whether a similar undecidability result obtains for reversible cellular automata. Wolfram gives the outline of a proof of computational universality of elementary cellular automaton number 110 in [16]. The argument depends on the use of periodic background patterns, on ordinary finite configurations Reachability for rule 110 is trivially decidable, see [14]. Since backgrounds may obscure the presence of signals, it is unknown whether theorem 4 carries over to this larger class of configurations.

References

1. C. H. Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, pages 525–532, 1973.
2. C. H. Bennett. The thermodynamics of computation—a review. *IJTP*, 21(12):905–940, 1982.

3. C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, pages 766–776, 1989.
4. M. Davis. *A note on universal Turing machines*, pages 167–175. In [10], 1956.
5. Y. Lecerf. Machine de Turing réversible. Insolubilité récursive en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un “isomorphisme de codes”. *C. R. Acad. Sci. Paris*, 257:2597–2600, 1963.
6. M. Lerman. *Degrees of Unsolvability*. Perspectives in Mathematical Logic. Springer Verlag, 1983.
7. K. Morita. Computation universality of one-dimensional reversible cellular automata. *Information Processing Letters*, 42:325–329, 1992.
8. K. Morita. Reversible cellular automata. *J. Information Processing Society of Japan*, 35:315–321, 1994.
9. K. Morita and M. Harao. Computation universality of 1 dimensional reversible (injective) cellular automata. *Transactions Institute of Electronics, Information and Communication Engineers, E*, 72:758–762, 1989.
10. C. E. Shannon and J. McCarthy. *Automata Studies*. Princeton University Press, 1956.
11. R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer Verlag, 1987.
12. K. Sutner. A note on Culik-Yu classes. *Complex Systems*, 3(1):107–115, 1989.
13. K. Sutner. Classifying circular cellular automata. *Physica D*, 45(1–3):386–395, 1990.
14. K. Sutner. Almost periodic configurations on linear cellular automata. Submitted, 2003.
15. K. Sutner. Cellular automata and intermediate degrees. *Theoretical Computer Science*, 296:365–375, 2003.
16. S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.