

# Cellular Automata and Intermediate Degrees

K. Sutner

*Computer Science Department  
Carnegie Mellon University  
Pittsburgh, PA 15213*

---

## Abstract

We study a classification of cellular automata based on the Turing degree of the orbits of the automaton. The difficulty of determining the membership of a cellular automaton in any one of these classes is characterized in the arithmetical hierarchy.

*Key words:* Cellular automata, intermediate degrees

---

## 1 Classifications

The study of cellular automata leads to many computationally hard problems. Except for a few select properties, such as injectivity, openness or surjectivity of the global map of a one-dimensional cellular automaton, most important features of cellular automata are undecidable. Indeed, when phrased as decision problems, it turns out that questions about natural properties of cellular automata are typically complete in their respective complexity classes. In all cases, the difficulties arise from the fact that cellular automata are computationally universal and can be used to simulate, say, universal Turing machines.

Thus it is not surprising that any general classification of cellular automata meets with considerable difficulties. The heuristic classification proposed by Wolfram in [11,14] attempts to distinguish between cellular automata on the basis of the structure of typical orbits of configurations under long term evolution. Wolfram's classification is based on extensive simulations of one-dimensional cellular automata and employs clearly discernible visual patterns. Briefly, Wolfram's classes can be described thus:

- Class 1: Configurations evolve to a fixed, homogeneous state.
- Class 2: Configurations evolve to simple, separated, periodic structures.

- Class 3: Configurations evolve to chaotic, aperiodic patterns.
- Class 4: Configurations evolve to complex patterns of localized structures.

In [3] Culik and Yu gave a formal definition of an analogous classification that is amenable to precise complexity considerations. The simplest class in their hierarchy consists of all cellular automata whose orbits end in a fixed point, and the second class is comprised of those automata whose orbits are ultimately periodic. Automata in the third class are characterized by having decidable orbits. Not surprisingly, the Culik-Yu classes themselves are all undecidable, see [3]. Indeed, it turns out that the first two classes are  $\Pi_2$ -complete, and the last is  $\Sigma_3$ -complete, see [9].

In this paper we propose a much more fine-grained hierarchy based on arbitrary recursively enumerable degrees. As Wolfram points out in [12], his classes admit different levels of prediction. In the first two levels, it is easy to determine properties of the orbit of a configuration. For example, we can generate any ultimately periodic orbit by brute-force enumeration, and determine its transient and period, the length of the limit cycle. Level 3 is vastly more complicated, but may still yield to a decision procedure in the sense that for any particular cellular automaton in this class an algorithm may exist that predicts whether a configuration occurs in the orbit of another. The last level, however, is computationally universal and requires explicit simulations. Thus, membership in an orbit is semi-decidable, but may fail to be decidable. Wolfram also mentions a classification based on the growth rate of configurations. For a discussion of problems with this approach see [1].

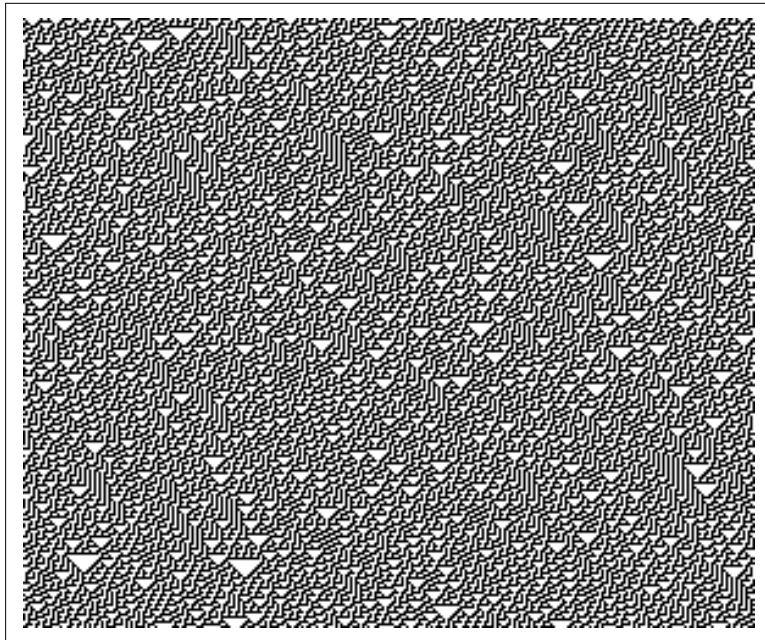


Fig. 1. Elementary cellular automaton number 30.

We would argue that the real situation is perhaps somewhat more complicated. As a case in point, take the notorious elementary cellular automaton rule 30, whose local rule is given by the Boolean function  $\rho(x, y, z) = x \oplus (y \vee z)$  where  $\oplus$  denotes exclusive or. This well-known automaton produces apparently random patterns, and is used as the default random number generator in the computer algebra system Mathematica, see [13]. Figure 1 shows a fragment of the orbit of a one-point seed configuration from time 750 to 1000. The structure of the patterns produced by rule 30 stands in stark contrast to those obtained from elementary cellular automaton number 110 with Boolean local rule  $\rho(x, y, z) = (\bar{x} \wedge y \wedge z) \oplus y \oplus z$ . Surprisingly rule 110 turns out to be computationally universal and is thus a member of Wolfram’s Class 4, see [14]. The universality proof outlined in chapter 11 of the reference is based on embedding cyclic tag systems and makes heavy use of persistent “local structures” in the evolution of certain configurations, which appear to be entirely absent in rule 30. One possible explanation for the behavior of rules like number 30 could be that their orbits are too complicated to admit a recursive decision procedure for the Reachability problem: given a source configuration  $X$  and a target configuration  $Y$ , does  $Y$  appear in the orbit of  $X$ . On the other hand, the orbits appear to be too chaotic to allow one to embed computationally universal systems.

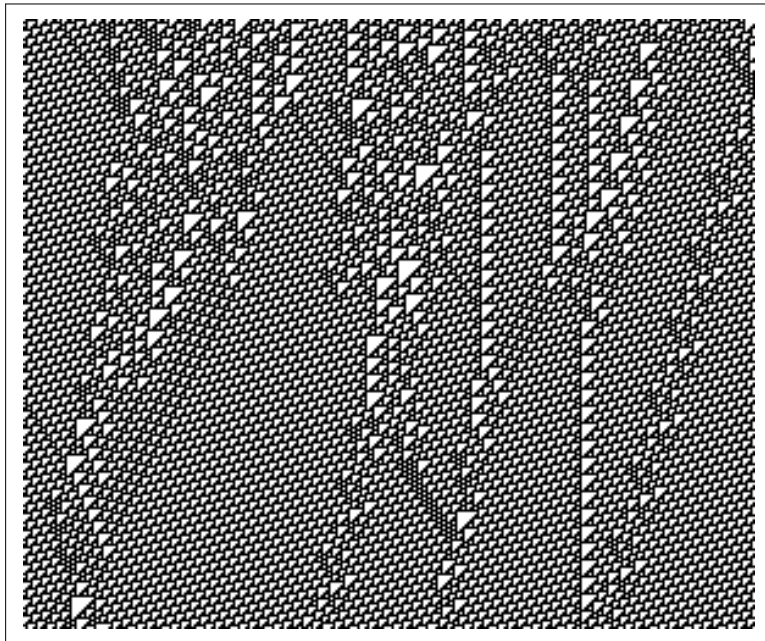


Fig. 2. Elementary cellular automaton number 110.

Thus, the orbits of rule 30, or, at the very least, of similar Class 3 automata, could be of intermediate degree: undecidable, yet less complicated than the Halting problem. The existence of intermediate recursively enumerable degrees is far from obvious and indeed was an open problem for more than two decades, see [7]. Constructions of intermediate r.e. sets require complicated

enumeration procedures, such as finite injury priority arguments, and fail to produce natural examples. As a matter of experience, concrete and natural r.e. problems in mathematics and computer science all appear to be either decidable or complete.

The appropriate setting for our considerations are *effective dynamical systems* of the form  $\mathcal{C} = \langle C, \rightarrow \rangle$ . Here  $\mathcal{C}$  is the space of *configurations*, and  $\rightarrow$  is the “next configuration” relation, which is always assumed to be deterministic. Configurations are finitary objects, and can be construed as finite words over some suitable alphabet  $\Sigma$ . We require  $C \subseteq \Sigma^*$  to be primitive recursive, so that it is easy to determine whether a given string codes a configuration. Moreover, the next configuration relation is required to be presented by a primitive recursive function. Hence, it is computationally easy to generate the unique next configuration, given a current configuration. Any classical model of computation such as Turing machines gives rise to such effective dynamical systems: the configurations are the instantaneous descriptions (IDs) of the Turing machine, and the next configuration relation is simply the one-step relation of the Turing machine. For us, the other important class of examples is derived from cellular automata. In order to obtain finitary objects, we have to restrict ourselves to finite configurations, or, more precisely, to configurations with finite support. Here we assume that the alphabet of the cellular automaton contains a special quiescent state. The support of a configuration is then the collection of cells that fail to be in this quiescent state. Note that we could also use spatially periodic configurations or even recursive configurations of the cellular automaton as the underlying space of an effective dynamical system. We will briefly comment on these alternatives below, but, as is customary, we focus on the finite configuration case.

Write  $\xrightarrow{*}$  for the transitive reflexive closure of  $\rightarrow$ . The Reachability problem in  $\mathcal{C}$  then takes the form: given two configurations  $X$  and  $Y$ , does  $X \xrightarrow{*} Y$  hold? Since  $\xrightarrow{t}$ , the  $t$ -fold composition of  $\rightarrow$  with itself, is primitive recursive, uniformly in  $t$ , it is clear that Reachability is always semi-decidable for any such system  $\mathcal{C}$ . Now consider specifically a one-dimensional cellular automaton  $\rho$ . Define the (*complete*) *orbit* of  $\rho$  to be

$$\text{Orb}_\rho = \{ (X, Y) \mid X \xrightarrow{*}_\rho Y \}$$

where  $X \rightarrow_\rho Y$  means that configuration  $Y$  is obtained from  $X$  by a single application of the global map of the cellular automaton. It is clear that  $\text{Orb}_\rho$  is semi-decidable for any  $\rho$ . For certain rules,  $\text{Orb}_\rho$  is certainly decidable, whereas for others its degree is  $\emptyset'$ . This suggests the following classification. For any recursively enumerable (r.e.) set  $A$  define

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

to be the collection of all cellular automata whose orbit has the same Turing degree as  $A$ . Thus,  $\mathbb{C}_\emptyset$  is none other than the third Culik-Yu class. Note that these classes are pairwise disjoint, if one prefers to deal with a cumulative hierarchy one can instead consider

$$\mathbb{C}_{\leq A} = \{ \rho \mid \text{Orb}_\rho \leq_T A \}$$

We will show below that all these classes are non-empty, and that the computational complexity of determining membership in any such class depends on  $A$ , and ranges from  $\Sigma_3$ -complete to  $\Sigma_4$ -complete.

The upper semi-lattice  $\mathcal{R}$  of the r.e. degrees has been the object of careful studies over the last few decades, and carries a complicated and rich structure. For example, there are r.e. sets  $A$  and  $B$  that are incomparable with respect to Turing reductions. Hence, there are cellular automata whose orbits are incomparable in the sense that the ability to perform simulations of one automaton at no cost would be of no help in making predictions for the other automaton, and vice versa. Another fundamental result concerning  $\mathcal{R}$  is Sacks' density theorem: for any two r.e. sets  $A$  and  $B$  such that  $A <_T B$ , there exists an intermediate set  $C$ :  $A <_T C <_T B$ . Thus, between any two levels of predictability of orbits there exists yet another one. As regards the decidability of the theory of  $\mathcal{R}$ , one can show that every countable partial order can be embedded into  $\mathcal{R}$ , which implies that the existential theory of  $\mathcal{R}$  is decidable. However, the full theory of  $\mathcal{R}$  is undecidable, see [6,7]. Thus, any orbit based classification of cellular automata is bound to be either too coarse, or highly complicated.

To keep this note reasonably short we will assume that the reader is familiar with basic recursion theory. Good expositions focusing on degree theory can be found in [6], [4], or [7]. Our notation is compatible with the last reference and we will refrain from redefining notions found there. In section 2 we will show how to construct cellular automata whose orbits have a given Turing degree of complexity. Hardness results concerning our classification are presented in section 3, and the last section contains a few comments about difficulties in applying this framework to concrete cellular automata such as rule 30.

## 2 Stable Simulations

We assume a standard enumeration  $M_e$  of one-tape Turing machines. For the sake of simplicity, let us agree that  $M_e$  takes as input a single natural number  $x$ . We have a natural encoding  $x \mapsto I_x$  that assigns an initial instantaneous description to each natural number  $x$ , and the computation of  $M$  on input  $x$  begins at  $I_x$ . Since we are only interested in acceptors, there is no need for

special output conventions. Let us assume, though, that  $M_e$  erases its tape before dropping into its accepting state. We obtain an enumeration of all r.e. sets by setting  $W_e = \{x \in \mathbb{N} \mid M_e \text{ accepts } x\}$ . Each machine  $M$  gives rise to a dynamical system  $\mathcal{C}_M = \langle C, \rightarrow_M \rangle$ . We may safely assume that the configurations here are given by the regular language  $\Sigma^*Q\Sigma^*$  where  $\Sigma$  denotes the tape alphabet of  $M$ ,  $Q$  its state set, and  $\Sigma$  and  $Q$  are disjoint. The next configuration relation  $\rightarrow_M$  corresponds to a single step in operation of  $M$ , augmented so that halting configurations are fixed points of  $\rightarrow_M$ . Thus acceptance of  $M$  on input  $x$  corresponds to the orbit of  $I_x$  in  $\mathcal{C}_M$  containing a special accepting configuration  $I_a = q_a$ .

The fundamental problem with this setup is that the system  $\mathcal{C}_M$  may contain very complicated orbits even though the Turing machine performs only trivial computations: the computations of  $M$  correspond solely to the orbits of the initial configurations  $I_x$ ,  $x \in \mathbb{N}$ . However, there may well be other instantaneous descriptions whose orbits are more complicated. For example,  $M$  may simply erase its input and accept, but it could contain a copy of a universal Turing machine whose states are not reachable from the initial state of  $M$ . The halting set of  $M$  is then trivial, but the degree of the orbits of  $\mathcal{C}_M$  is  $\emptyset'$ .

To address this problem, let us call an ID  $D$  *accessible* if, for some number  $x$ ,  $I_x \xrightarrow{*}_M D$ . Since the collection of inaccessible IDs fails to be decidable in general we cannot simply exclude them from the dynamical system – recall that we require the carrier set to be primitive recursive. However, we can modify the Turing machine so that inaccessible IDs do not artificially inflate the Turing degree of the orbits of  $\mathcal{C}_M$ . To this end, let us define a Turing machine to be *stable* if it halts on all its inaccessible configurations. As we will see, for stable machines the orbits of  $\mathcal{C}_M$  have the same Turing degree as the halting set of  $M$ . See [2] and [5] for similar problems in connection with register machines and Turing machines, and an application to Thue systems. The basic idea is this: while an adversary may try to produce an instantaneous inscription that causes the machine to perform an erroneous and complicated computation, we can spoil any such attempts by modifying the finite state control of the machine so as to recognize these tampered IDs after a finite number of steps. A variant of this construction can be found in [9], where all the coding issues are addressed carefully.

**Lemma 1** *For every Turing machine  $M$  there is a stable Turing machine  $M'$  that accepts the same set of inputs as  $M$ .*

**PROOF.** The main idea is to have machine  $M'$  perform a highly redundant simulation of  $M$ , keeping track of the original input  $x$  and the number of steps in the alleged computation. To this end, augment the tape alphabet of  $M$  by a new blank symbol, a tag 1 for unary counters, and markers  $\#_1, \dots, \#_6$ . A

typical tape inscription of  $M'$  has the form

$$\#_1 x \#_2 s \#_3 D \#_4 t \#_5 E \#_6,$$

flanked by blank symbols. Here  $x$  is the input of  $M$ , say, written in unary,  $s$  and  $t$  are unary counters, and  $D, E \in \Sigma^* Q \Sigma^*$  are IDs of  $M$ . The interpretation is that  $x$  is the alleged input of a computation of  $M$ , and  $s$  the alleged number of steps in the computation needed to reach  $D$  from  $I_x$ . The inscription is *correct* if indeed  $I_x \xrightarrow{s}_M D$ . In order to test correctness, machine  $M'$  uses the  $t$  and  $E$  components. Initially,  $t$  is set to 0 and  $E$  to  $I_x$ . The machine then simulates  $M$  on  $E$ , increasing the counter  $t$  as appropriate. If, after  $s$  steps,  $E$  fails to be equal to  $D$  machine  $M'$  halts in a non-accepting state. Also note that  $M'$  can verify the form of its tape inscription, if a malformed inscription is encountered the machine also halts in a non-accepting state. Otherwise, it simulates one more step in the computation of  $M$ , leading to a tape inscription of the form

$$\#_1 x \#_2 s1 \#_3 D' \#_4 t \#_5 E \#_6,$$

where  $D \rightarrow_M D'$ .  $M'$  then performs another correctness test, and so on. If an accepting ID of  $M$  is encountered,  $M'$  erases its tape and halts. The states of  $M'$  controlling both phases are distinct.

By alternating between correctness testing and actual simulation steps, we can force  $M'$  to enter a proper correctness test after finitely many steps regardless of the starting ID  $X$  of  $M'$ . To see this, note first that any simulation phase is necessarily finite: either we encounter a malformed tape inscription, and halt immediately, or we simply replace  $D$  by  $D'$ ,  $s$  by  $s1$ . If  $X$  places the machine into a state used during correctness testing, it may falsely conclude that the given inscription is correct. For example, the time stamps may be wrong, and  $D$  may in fact be inaccessible. However, the testing phase is finite, and must ultimately be followed by another simulation phase, which is in turn finite. But after the next simulation step  $M'$  performs a complete correctness test, will discover inaccessibility, and halt.

Since the construction of  $M'$  just outlined is certainly primitive recursive and uniform in  $M$ , we have a primitive recursive function  $S$  such that  $W_e = W_{S(e)}$  and  $M_{S(e)}$  is stable. It follows that ALL, the set of all indices  $e$  such that  $M_e$  halts on all configurations is  $\Pi_2$ -complete, analogous to the well-known fact that  $\text{TOT} = \{ e \mid W_e = \mathbb{N} \}$  is  $\Pi_2$ -complete. Note, though, that the latter is an index set whereas the former is not. For the purposes of this paper, it is more convenient to work with index sets, see section 3 below.

The second step in the simulation is to replace a Turing  $M$  machine by a suitable one-dimensional cellular automaton  $\rho$ . This time we have to contend

with configurations of the cellular automaton that do not correspond to IDs of the Turing machine, but it is entirely straightforward to make sure that any such configuration evolves to a quiescent one in linear time. For example, assuming the standard coding of an inscription as a string  $upv \in \Sigma^*Q\Sigma^*$  where the special center symbol  $p$  indicates the state and head position of the machine, and  $uv$  its tape inscription, we can augment the alphabet of the cellular automata to contain a special quiescent state 0, and tag all the symbols in  $\Sigma$  by “left” and “right” indicators. A legitimate configuration of the cellular automaton then is a string in  $0\Sigma_{\text{left}}^*Q\Sigma_{\text{right}}^*0$ . Left symbols in  $\Sigma$  may be flanked on the left only by 0, and on the right only by an element of  $Q$ . A similar rule applies to  $\Sigma_{\text{right}}$ . The cellular automaton can then recognize malformed configurations such as  $0upvqw0$  and evolve them to the quiescent configuration in linear time.

More precisely, a configuration of  $\rho$  may contain several blocks of symbols, separated by 0 cells, each of which may or may not correspond to an ID of  $M$ . The blocks that fail to correspond to IDs will evolve to blank cells in linear time, but the others may coexist, so that in effect we may be simulating several computations of the Turing machine in parallel. If two such blocks expand during the evolution of the configuration and collide, they too evolve to quiescence. As we will see, this complication is entirely technical and does not affect our results.

Again the construction of  $\rho$  from  $M$  is primitive recursive and uniform, so there is a primitive recursive function  $R$  such that  $\rho = R(M)$ .

By combining these two steps, we can produce cellular automata whose orbits have exactly a given, arbitrary r.e. degree.

**Lemma 2** *Let  $M$  be a Turing machine accepting  $W \subseteq \mathbb{N}$ , and  $\rho$  the cellular automaton simulating the stable version of  $M$ . Then  $\text{Orb}_\rho \equiv_T W$ .*

**PROOF.** It is easy to see that  $W \leq_T \text{Orb}_\rho$ . For  $x \in W$  iff  $M$  accepts  $x$ , iff the stable version  $M'$  accepts  $x$ . Recall that  $M'$  erases its tape before accepting, so, unlike with intermediate tape inscription,  $I_a$  carries no time-stamps. But then  $M'$  accepts  $x$  iff  $I_x \xrightarrow{*}_\rho I_a$  where  $I_a$  denotes the accepting ID of Turing machine  $M'$ .

For the opposite direction, consider two configurations  $X$  and  $Y$  of  $\rho$ . We can evolve  $X$  until only 0-separated blocks survive that each code an ID of the stable machine  $M'$ , all along testing for the appearance of  $Y$ . If no such block ever appears, then the orbit of  $X$  under  $\rho$  is finite, and we can check by brute force if  $Y$  occurs in it.

First assume that there is just one proper block coding an ID of  $M'$ . Since the IDs of  $M'$  contain  $x$ , the original input for the computation of  $M$ , we can use oracle  $W$  to determine if the machine halts on  $x$ . If so, the orbit is again finite and we can generate it completely, and search for  $Y$ . Otherwise the orbit is infinite, but in order for  $Y$  to appear in the orbit of  $X$  it must also carry the time-stamps included in the tape inscriptions of  $M'$ . Hence we have a bound on the number of steps of the evolution of  $X$  we have to follow to decide whether  $Y$  will appear.

If there are several proper blocks, the argument still applies, though some of these blocks may collide during the computation and turn into quiescent cells. Repeated queries to oracle  $W$  may be needed, but it still holds true that  $\text{Orb}_\rho$  is Turing reducible to  $W$ .

Hence we can primitive recursively compute  $\rho = R(S(e))$  such that  $\text{Orb}_\rho \equiv_T W_e$ . We will use this fact below to establish hardness of the classes  $\mathbb{C}_A$ .

The argument above can be modified in a number of ways by slightly altering the type of the tagged instantaneous descriptions, or by adjusting the behavior of the simulating machine. One can then establish the following result that combines restrictions on the complexity of the Reachability problem with similar restrictions on the complexity of the closely related Confluence problem. The latter asks whether for two given configurations  $X$  and  $Y$  wind up on the same limit cycle, more precisely whether there is a configuration  $Z$  such that  $X \xrightarrow{*} Z$  and  $Y \xrightarrow{*} Z$ . For a proof see [8].

**Theorem 3** *Given any two recursively enumerable degrees  $d_1$  and  $d_2$  there is a cellular automaton  $\rho$  such that the Reachability problem for  $\rho$  has degree  $d_1$ , and the Confluence problem for  $\rho$  has degree  $d_2$ .*

At any rate, from the definition of our degree-based classification we immediately get the next corollary.

**Corollary 4** *The classes  $\mathbb{C}_A$  where  $A$  is an arbitrary recursively enumerable set are all non-empty.*

It should be noted, though, that our classification is a bit coarse at the bottom end:  $\mathbb{C}_\emptyset$  does not distinguish between, say, a cellular automaton all of whose configurations evolve to a quiescent fixed point, and another one that has more complicated albeit still decidable orbits. The following results concerning these low classes are established in [9].

**Theorem 5** *It is  $\Pi_2$ -complete to determine if all orbits evolve to a fixed point. Likewise, it is  $\Pi_2$ -complete to determine if all orbits are ultimately periodic.*

If we consider only spatially periodic configurations, all orbits are by necessity ultimately periodic. Nonetheless, one still cannot distinguish effectively between periods of different lengths.

**Theorem 6** *For spatially periodic configurations it is  $\Pi_0$ -complete to determine if all orbits evolve to a fixed point.*

As a matter of fact, there is an infinite hierarchy based on the length of the limit cycles as a function of the size of the source configuration. For example, it is  $\Sigma_2$ -complete to test whether the length of the limit cycle is  $O(n^k)$  for a fixed degree  $k$ , where  $n$  is the size of the initial configuration. This holds even when  $k = 0$ , i.e., when the limit cycles are required to be of bounded length. Likewise, one can show there are no computable bounds on the lengths of the limit cycles in this setting, see [10].

### 3 Hardness Results

We have seen that all the classes  $\mathbb{C}_A$  and thus  $\mathbb{C}_{\leq A}$  are non-trivial. Note, though, that by a theorem of Lachlan and Yates, see [6], there exist non-recursive sets  $A$  and  $B$  such that  $\mathbb{C}_{\leq A} \cap \mathbb{C}_{\leq B} = \mathbb{C}_\emptyset$ . Thus, even given the orbits of two cellular automata  $\rho \in \mathbb{C}_{\leq A}$  and  $\sigma \in \mathbb{C}_{\leq B}$  as oracles, we still cannot answer reachability questions about any other cellular automaton that is not already in  $\mathbb{C}_\emptyset$ .

The complexity of the classes  $\mathbb{C}_A$  can be described as follows. Recall that  $\Sigma_k^A$  is the collection of all sets definable by a  $\Sigma_k$  formula whose matrix is a predicate that is recursive in  $A$ . Thus,  $\Sigma_k^\emptyset$  is the same as  $\Sigma_k$ .

**Theorem 7** *Let  $A$  be an arbitrary recursively enumerable set. Then class  $\mathbb{C}_A$  is  $\Sigma_3^A$ -complete.*

**PROOF.** Consider the index set  $G(A) = \{e \mid W_e \equiv_T A\}$ . It follows from lemma 2 that  $e \in G(A)$  iff  $\rho = R(S(e))$  lies in class  $\mathbb{C}_A$ . As we have seen,  $\rho$  can be constructed primitive recursively from  $e$ . By Yates' index theorem,  $G(A)$  is  $\Sigma_3^A$ -complete, see [7].

Specializing to the bottom and top end of the hierarchy we immediately obtain the two following corollaries.

**Corollary 8**  *$\mathbb{C}_\emptyset$  is  $\Sigma_3$ -complete.*

**Corollary 9**  *$\mathbb{C}_{\emptyset'}$  is  $\Sigma_4$ -complete.*

**PROOF.** The triple-jump  $A^{(3)}$  is  $\Sigma_3^A$ -complete. Hence, for  $A = \emptyset'$  we get  $A^{(4)}$ , and thus a  $\Sigma_4$ -complete set.

Thus, testing whether a cellular automaton is computationally universal is as hard as  $\text{COMP} = \{e \mid W_e \equiv_T \emptyset'\}$ . Note that there are other classes with the same complexity. For example, for any high r.e. set  $A$ , i.e.,  $A' \equiv_T \emptyset''$ ,  $\Sigma_3^A$  is the same as  $\Sigma_4$ . Likewise, at the other end of the hierarchy, if  $A$  is low, i.e.,  $A' \equiv_T \emptyset'$ , then  $\Sigma_3^A$  is the same as  $\Sigma_3$ . Thus, testing membership here is no harder than testing membership for the bottom class  $\mathbb{C}_\emptyset$ .

With regard to the cumulative version of the classification we have the following analogous characterization.

**Theorem 10** *Let  $A$  be any recursively enumerable set such that  $A <_T \emptyset'$ . Then class  $\mathbb{C}_{\leq A}$  is  $\Sigma_3^A$ -complete.*

**PROOF.** The argument is analogous to the last theorem, with the modification that the appropriate index set is now  $G(\leq A) = \{e \mid W_e \leq_T A\}$ . Since  $G(\leq A)$  is still  $\Sigma_3^A$ -complete for  $A <_T \emptyset'$  our claim follows.

Of course,  $\mathbb{C}_\emptyset = \mathbb{C}_{\leq \emptyset}$ . However,  $\mathbb{C}_{\leq \emptyset'}$  is trivial and comprises all cellular automata. A similar result holds for lower bounds on the complexity of a cellular automaton. To simplify notation, write  $\mathbb{C}_{\geq A}$  for the collection of all automata whose orbits have Turing degree at least the degree of  $A$ .

**Theorem 11** *Let  $A$  be an arbitrary recursively enumerable but non-recursive set. Then class  $\mathbb{C}_{\geq A}$  is  $\Sigma_3^A$ -complete.*

**PROOF.** This time we use the index set  $G(\geq A) = \{e \mid W_e \geq_T A\}$ . Again,  $G(\geq A)$  is  $\Sigma_3^A$ -complete as long as  $A$  is not recursive, and our claim follows.

Thus, the difficulty of pinpointing the degree of the orbit of a cellular automaton is the same as the difficulty of establishing upper and lower bounds, disregarding the obvious exceptions at  $\emptyset$  and  $\emptyset'$ .

## 4 Conclusion

We have shown that there is a natural classification of cellular automata based on the Turing degree of the orbits of configurations under the global rule of

the automaton. Since the upper semi-lattice of recursively enumerable degrees has an extremely rich and complicated structure, one should expect similarly complicated behaviors from cellular automata. Of course, the real challenge is to identify concrete cellular automata that belong to one of the intermediate classes  $\mathbb{C}_A$ ,  $\emptyset <_T A <_T \emptyset'$ .

Candidates for such intermediate cellular automata are those whose orbits are chaotic, but appear not to carry enough structure to express universal computation. Perhaps the simplest example for this type of cellular automaton is the pseudo-random number generator, elementary cellular automaton number 30. Note, though, that rule 30 has the property that a finite configuration of size  $n$  always expands to a configuration of size  $n + 2$  in one step. Thus, configurations carry an implicit time-stamp, and rule 30 belongs to  $\mathbb{C}_\emptyset$ , albeit for the wrong reasons. The same problem arises with any automaton whose finite configurations are monotonically increasing in size. In order to accommodate this effect, it is tempting to modify the definition of orbit to include sub-configurations rather than full finite configurations. More precisely, write  $Y \sqsubseteq Z$  if  $Z$  is a finite configuration, represented as a finite word over some suitable alphabet, and  $Y$  is a factor of  $Z$ . Then define

$$\text{Orb}_\rho^* = \left\{ (X, Y) \mid \exists Z \left( Y \sqsubseteq Z \text{ and } X \xrightarrow{\rho}^* Z \right) \right\}.$$

Note that for  $\rho = R(S(e))$  from above it is still true that  $W_e \leq \text{Orb}_\rho^*$ , but it may well happen that, say,  $W_e = \emptyset$  whereas  $\text{Orb}_\rho^*$  has degree  $\emptyset'$ .

More generally, there is the issue of information hiding in classical computations versus the full display of all steps in an orbit. For example, in the classical Friedberg-Muchnik construction of two incomparable r.e. sets, a universal Turing machine is used to handle the infinite list of conflicting requirements. But in the end only the two sets are produced as output, and these two sets are indeed of intermediate degree. We are currently not aware of any technique that could be used to address this problem. The Principle of Computational Equivalence, proposed by Wolfram in [14] would suggest that no natural solution exists at all.

## References

- [1] J. T. Baldwin and S. Shelah. On the classifiability of cellular automata. *Theoretical Computer Science*, 230(1-2):117–129, 2000.
- [2] M. Davis. *A note on universal Turing machines*, pages 172–175. Princeton University Press, 1956.

- [3] K. Culik II and Sheng Yu. Undecidability of CA classification schemes. *Complex Systems*, 2(2):177–190, 1988.
- [4] M. Lerman. *Degrees of Unsolvability*. Perspectives in Mathematical Logic. Springer Verlag, 1983.
- [5] J. C. Shepherdson. Machine configuration and word problems of given degree of unsolvability. *Zeitschrift f. Math. Logik u. Grundlagen d. Mathematik*, 11:149–175, 1965.
- [6] R. A. Shore. *The recursively enumerable degrees*, volume 140 of *Studies in Logic*, chapter 6, pages 169–197. North-Holland, 1999.
- [7] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer Verlag, 1987.
- [8] K. Sutner. Cellular automata and intermediate reachability problems. To appear in *Fundamenta Informaticae*.
- [9] K. Sutner. A note on Culik-Yu classes. *Complex Systems*, 3(1):107–115, 1989.
- [10] K. Sutner. Classifying circular cellular automata. *Physica D*, 45(1–3):386–395, 1990.
- [11] S. Wolfram. Computation theory of cellular automata. *Comm. Math. Physics*, 96(1):15–57, 1984.
- [12] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.
- [13] S. Wolfram. *The Mathematica Book*. Wolfram Media, Cambridge UP, 4th edition, 1999.
- [14] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.