

Cellular Automata and Intermediate Reachability Problems

K. Sutner

Computer Science Department

Carnegie Mellon University

Pittsburgh, PA 15213

sutner@cs.cmu.edu

Abstract. We exhibit one-dimensional cellular automata whose reachability and confluence problems have arbitrary r.e. degree of unsolvability.

Keywords: Cellular automata, intermediate degrees, confluence problem

1. Motivation

Computational problems associated with infinite cellular automata are notoriously hard: except for a few select problems, such as the injectivity of one-dimensional cellular automata, they are all undecidable. As a matter of fact, the problems are typically complete in their respective complexity classes. For example, the question of whether a finite configuration on a one-dimensional cellular automaton evolves to another, is recursively enumerable (r.e.), and is complete in the class of r.e. sets. The question of whether all finite configurations on a one-dimensional cellular automaton evolve to a fixed point is Π_2^0 -complete. Even the analogous question for all spatially periodic configurations is still Π_1^0 -complete.

The ultimate reason for these completeness results is that some cellular automata can simulate computationally universal systems, and in particular Turing machines. Indeed, even an exceedingly simple cellular automaton such as the elementary automaton with rule number 110 turns out to be computationally universal, see chapter 11 in [16]. The local map of this automaton is given by the Boolean function $\rho(x, y, z) = (\bar{x} \wedge y \wedge z) \oplus y \oplus z$ where \oplus denotes exclusive or, and provides enough persistent, localized structures in certain configurations to embed certain types of cyclic tag systems, thus establishing universality. The heuristic classification proposed by Wolfram in [14] suggests that there might be cellular automata that are associated with decision problems of intermediate degree: rules that are too complicated to allow for recursive solutions, but too chaotic to allow one to embed computationally universal systems. A potential example for this type of cellular automaton is rule 30 with Boolean function

$\rho(x, y, z) = x \oplus (y \vee z)$. Even on one-point seed configurations rule 30 produces seemingly random patterns, and can indeed be used as a pseudo-random number generator. On the other hand, there appear to be no structures in the evolution of configurations under rule 30 such as “moving particles” that might be exploited in a universality argument. This lead Wolfram to ask whether rule 30 might be associated with problems that are neither decidable nor complete, [15]. Recursively enumerable problems of this type are said to be of intermediate degree. The existence of intermediate degrees is far from obvious and was an open problem for about a quarter of a century, see [7]. Indeed, all known constructions require complicated enumeration arguments (finite injury priority arguments) and produce somewhat artificial examples. On the other hand, natural r.e. problems in mathematics and computer science all appear to be either decidable or complete. Indeed, the Principle of Computational Equivalence, proposed by Wolfram in [16] would suggest that natural cellular automata—as opposed to the carefully constructed ones below—can not have orbits of intermediate degree.

In this paper we will therefore tackle the question of whether such intermediate cellular automata exist at all, and what shape the specific decision problems might take. As a general framework, we will consider effective dynamical systems of the form $\langle \mathcal{C}, \rightarrow \rangle$ where \mathcal{C} is the space of *configurations*, and \rightarrow is the “next configuration” relation. We are here only interested in the deterministic case where for each configuration X there exists precisely one configuration Y such that $X \rightarrow Y$. By effective we mean that the configurations can be coded as integers in some natural way, and the next configuration relation is primitive recursive in the sense that the corresponding relation on code numbers is so primitive recursive. Any cellular automaton when restricted to finite, spatially periodic, or recursive configurations provides an example of such a system. Other examples include the instantaneous descriptions of a Turing machine, or some other model of computation such as a register machine, together with the one-step relation, or, more generally, rewrite systems.

To analyze the complexity of the system from a computational point of view, one studies various decision problems associated with the system. To this end let $\overset{*}{\rightarrow}$ be the transitive reflexive closure of \rightarrow . The most important decision problem is arguably the *Reachability Problem*: given two configurations X and Y , is Y reachable from X , i.e., is $X \overset{*}{\rightarrow} Y$? A closely related but different question, motivated by the study of rewrite systems, is the *Confluence Problem*: given two configurations X and Y , is there a configuration Z that is reachable from both X and Y ? More formally, define $\overset{t}{\rightarrow}$ to be the t -fold composition of \rightarrow with itself. It is easy to see that $X \overset{t}{\rightarrow} Y$ is primitive recursive, uniformly in t . Then the Reachability and Confluence Problem amount to determining, given configurations X and Y , whether

$$\begin{aligned} &\exists t \geq 0 (X \overset{t}{\rightarrow} Y), \\ &\exists Z \in \mathcal{C}, t, s \geq 0 (X \overset{s}{\rightarrow} Z \text{ and } Y \overset{t}{\rightarrow} Z), \end{aligned}$$

respectively. Since configurations are finitary objects, both problems are r.e. For systems obtained from cellular automata, the predicates in these formulae are primitive recursive, uniformly in their parameters and the underlying cellular automaton.

The construction of a cellular automaton ρ for which the problems have intermediate degree meets with two central obstructions. First, one has to be careful to translate an r.e. set $A \subseteq \mathbb{N}$ into a Turing machine M_A for which reachability or confluence on the space of instantaneous descriptions is of exactly the same Turing degree as A (rather than just at least of the degree of A). Second, in the translation to

cellular automata, one has to contend with arbitrary configurations, not just those that correspond to meaningful instantaneous descriptions of the Turing machine. We will tackle these problems in the next section, and then apply the results to the construction of suitable cellular automata. In [8, 9] similar techniques are used in the context of cellular automata. The first reference in particular contains a very careful explanation of the necessary coding machinery, an issue we will completely ignore here. See [2] and [5] for similar problems in connection with register machines and Turing machines, and an application to Thue systems, and [1] in the context of cellular automata. To keep this note reasonably short we will assume that the reader is familiar with basic recursion theory. Good expositions can be found in [6], [4], or [7]; the latter two in particular focus on degree theory.

2. Pinning Down Degrees

Consider an arbitrary r.e. set $A \subseteq \mathbb{N}$. There is a primitive recursive predicate $\varphi(x, y)$ such that A is the domain of φ , i.e.,

$$x \in A \iff \exists y \varphi(x, y)$$

It is straightforward to construct a Turing machine M that, on input x , will search for a corresponding witness y . It will erase the tape and terminate after finitely many steps whenever $x \in A$, and will fail to halt otherwise. Consider the space of instantaneous descriptions of this Turing machine. The special version of the Reachability Problem where the source configuration X is constrained to be an initial instantaneous description I_x corresponding to some input value x , and the target configuration Y is constrained to be the halting configuration is of the same degree of unsolvability as A . However, the general Reachability Problem may well have higher degree than A : we could ask questions about configurations that do not occur in any of the actual computations of M , and it is not clear how oracle A could help to answer such questions.

More precisely, let us call an instantaneous description D *accessible* if there exists some input x such that $I_x \xrightarrow{*} D$; and *inaccessible* otherwise. There is no effective method to eliminate inaccessible instantaneous descriptions since this set may well be Π_1^0 -complete. Hence, we need to modify the Turing machine itself to avoid problems with inaccessibility. Let us say that a Turing machine is *stable* iff it halts on all inaccessible instantaneous descriptions. The following result is shown in [8].

Theorem 2.1. For each Turing machine M there is a stable Turing machine M' which halts on exactly the same inputs as M . Moreover, M' can be constructed primitive recursively from M .

The trick is to attach tags to the original instantaneous descriptions of M to obtain tape inscriptions of the form

$$\# x \# s \# D \#$$

where D is an instantaneous description of the old machine M . A tagged inscription is correct if indeed $I_x \xrightarrow{s} D$ in the old machine M . The new machine M' checks the correctness of its tape. If it is found incorrect, the machine erases its tape and halts. Otherwise, it increments s and replaces D by the next configuration. It then returns to a correctness check, and so forth. Applying this stability construction to the naive machine M we obtain the machine M_A .

It is straightforward to simulate the Turing machine M_A by a one-dimensional cellular automaton ρ . Moreover, we can choose the simulation in such a way that any configuration ρ which does not consist

of finitely many blocks representing tagged tape inscriptions, separated by blank cells, will evolve to the quiescent configuration consisting only of blank cells. Furthermore, we can set things up so that the tagged blocks grow indefinitely in both directions unless the original machine M halts on input x . Since these basic blocks may grow during evolution, there may be collisions. We assume that in case of a collision all non-blank cells involved in the collision will be erased.

3. Reachability and Confluence

We now design cellular automata based on stable Turing machines. The construction for the two following theorems is a bit more complicated than strictly necessary and is motivated by the corollary below.

Theorem 3.1. For any recursively enumerable degree d there is a one-dimensional cellular automaton whose Reachability Problem for finite configurations is of degree d .

Proof:

Consider a r.e. set A of degree d , and let M_A be the stable Turing machine constructed in the last section. Let ρ be the CA simulating M_A , again as described in the last section. Modify ρ so that when a witness y such that $\varphi(x, y)$ holds has been found, and when the underlying Turing machine therefore halts, the CA resets the tagged block to

$$\# x \# 0 \# I_x \#$$

and the computation of M_A starts all over. Hence, any configuration will ultimately evolve to one of the three following scenarios:

- the all-blanks configuration,
- a single tagged block,
- a collection of separate tagged blocks that evolve periodically without collisions.

We claim that the Reachability Problem for this CA has precisely degree d . To see this, first note that

$$x \in A \iff \# x \# 1 \# I_x \# \xrightarrow{*} \# x \# 0 \# I_x \#$$

Hence, A is Turing reducible to the Reachability Problem for ρ .

On the other hand, suppose we have A as an oracle, and we are given two configurations X and Y . We can evolve X until only blank-separated tagged blocks survive, all the while testing whether Y occurs in the process. Suppose otherwise, so we are now dealing with tagged blocks only. Each such block B uniquely identifies an integer x corresponding to the actual input of the Turing machine. If $x \in A$, then the evolution of B produces a finite orbit (there might be a collision with another block, but in that case all cells turn blank, and finiteness is guaranteed). Hence we can check by brute force if Y ever occurs in this finite orbit. Now suppose $x \notin A$. Again, we can wait until we are dealing with tagged blocks only. Indeed, after a while there will be at most one such block left (recall that the blocks are growing), whose evolution will continue indefinitely without ever reaching a cycle. The only way Y can appear in this infinite orbit is if it is of the form $\# x \# t \# D \#$. But then t imposes a bound on the number of steps we have to follow the orbit to decide whether Y occurs. \square

Theorem 3.2. For any recursively enumerable degree d there is a one dimensional cellular automaton whose Confluence Problem for finite configurations is of degree d .

Proof:

The argument is very similar to the one in the last theorem. However, this time we use tagged blocks of the form

$$\# \beta \# x \# s \# D \#$$

where β is a single bit. Moreover, when the underlying Turing machine halts, the CA sets the tagged block to

$$\# 0 \# x \# s \#$$

which pattern will be a fixed point of the CA.

To see that the Confluence Problem has the same degree as A , observe that

$$x \in A \iff \# 0 \# x \# 0 \# I_x \#, \# 1 \# x \# 0 \# I_x \# \text{ conflue}$$

On the other hand, with A as an oracle, we can decide confluence. For the sake of simplicity, let us only consider configurations of the form $\# \beta \# x \# s \# D \#$ which agree in their x component. If $x \in A$, then all such configurations will ultimately evolve to $\# 0 \# x \# t \#$ where t is the number of steps the Turing machine needs to halt (we may safely assume that there are no further instantaneous descriptions after halting), provided that $s \leq t$. If $x \notin A$, then the configurations have to agree in their β components in order to confluence. In either case, confluence is decidable relative to A . \square

Corollary 3.1. 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 .

Proof:

Note that the CA constructed in theorem 3.1 has decidable Confluence Problem, whereas the CA in theorem 3.2 has decidable Reachability Problem. For example, consider two configurations X and Y for the first CA. Evolve both X and Y until there is either just one tagged block left, or until all the blocks have gone through a full period (the case where one of the configurations evolves to all-blanks is easily dealt with). In the first case, we claim that X and Y confluence if, and only if, the corresponding inputs x and y are the same. Clearly, $x = y$ is necessary, since these parts of the configuration never change. But $x = y$ is also sufficient for confluence, since either both blocks reset to $\# x \# 0 \# I_x \#$ at some time, or they both will evolve to a block $\# x \# t \# D \#$ for some sufficiently large t . In the second case, both orbits are finite and we can check confluence by brute force.

The argument for the second CA is quite similar. Note that the “time field” s is preserved, even when the Turing machine halts and a transition is made to tagged blocks of the form $\# 0 \# x \# s \#$ is made. As a consequence, reachability is decidable: we can simply follow the evolution for the required number of steps.

Hence, we can combine the two CAs by splitting cells into an upper track and a lower track, and using the first rule on the upper track, and the second rule on the lower track. One can easily check that this product CA has the right properties with respect to confluence. As regards reachability, there is a slight complication since the set of times t such that the upper-track source configuration evolves to the target configuration at time t may be semi-periodic: $t = t_0 + \mathbb{N}p$. However, the lower-track machine that deals with confluence can be checked for reachability on this restricted set of admissible times. \square

4. Garden-of-Eden Problems

Historically one of the first problems associated with cellular automata to receive close scrutiny was the Garden-of-Eden question: does a given configuration on a cellular automaton have a predecessor? To be more precise, we consider the question whether a given finite configuration X has another finite configuration as its immediate predecessor under some global map ρ . The most remarkable property of this problem is that its complexity depends strongly on the dimension of the cellular automaton. For one-dimensional cellular automata the problem can be translated into an acceptance problem for finite state machines and is easily seen to be decidable in polynomial time. However, the Garden-of-Eden problem becomes undecidable for two-dimensional cellular automata (and hence for all higher dimensions). If the predecessor is required to be finite the problem is r.e.-complete, but for arbitrary predecessors the problem is Π_1^0 -complete, see [17]. For finite predecessors, we can select any r.e. degree.

Theorem 4.1. For any recursively enumerable degree d there is a two-dimensional cellular automaton whose Garden-of-Eden Problem for finite configurations is of degree d .

Proof:

It is standard to stack up instantaneous descriptions of a Turing machine to generate a two-dimensional pattern. It is not hard to construct a two-dimensional cellular automaton that has a finite predecessor for a configuration coding just the initial instantaneous description of a Turing machine, if, and only if, the machine halts on that input. \square

A similar argument also shows that for infinite predecessor configurations the negation of the problem can also be chosen of arbitrary r.e. degree.

The Garden-of-Eden Problem is different from the Reachability and Confluence Problems also when one considers spatially periodic configurations (or, what amounts to the same, finite cellular automata with periodic boundary conditions). Both Reachability and Confluence are easily seen to be PSPACE-complete in this case, regardless of dimension. However, the Garden-of-Eden Problem is co-NLOG-complete for one-dimensional cellular automata, and co-NP-complete in the two- or higher-dimensional case, see [11].

Another surprising feature of the Garden-of-Eden problem can be observed with respect to recursive configurations. It is shown in [10] that for one-dimensional cellular automata, it is reasonable to consider the subspace of recursive configurations with respect to Gardens-of-Eden: any recursive configuration that has a predecessor already has a recursive predecessor. However, this predecessor cannot in general be computed effectively from the target. Perhaps surprisingly, no intermediate degrees occur in this case. The core of the argument is to translate the computational resource of time into space in the context of configurations.

Theorem 4.2. For any one-dimensional cellular automaton ρ and recursive configurations, the Garden-of-Eden Problem is either decidable, or co-r.e.-complete.

Proof:

Consider an one-dimensional CA ρ . If ρ is surjective there are no Gardens-of-Eden, and the problem is trivially decidable. So suppose ρ is not surjective, and, by compactness, let $w = w_1w_2 \dots w_n$ be a

finite configuration that does not occur in any configuration of $\rho(X)$. Without loss of generality suppose that the alphabet of the CA contains a symbol 0 such that ${}^\omega 0^\omega$ (a bi-infinite sequence of 0's) has a predecessor. For each natural number e define a recursive target configuration X_e as follows. If the e th partial recursive function on input e fails to halt, $X_e = {}^\omega 0^\omega$. Otherwise, let t be minimal such that the e th partial recursive function on input e halts after t steps, and define:

$$X_e(p) = \begin{cases} w_i & \text{if } p = t + i, i = 1, \dots, n, \\ 0 & \text{otherwise.} \end{cases}$$

A moment's thought will reveal that X_e is indeed recursive, uniformly in e . But by definition X_e has a predecessor under ρ if, and only if, the e th partial recursive function on input e fails to halt, a Π_1^0 -complete problem. \square

5. Conclusion

We have shown that the Reachability Problem and the Confluence Problem for finite configurations on linear cellular automata can have arbitrary recursively enumerable degree of complexity. Regrettably, the corresponding cellular automata are artificial, and are deliberately designed to achieve given r.e. degrees of unsolvability. It would be interesting to see if natural examples of cellular automata with Reachability or Confluence Problem of intermediate degree can be found.

Note that rule 30 is not a candidate for these specific decision problems; both the Reachability and Confluence Problem are trivially decidable, albeit for the wrong reason: the length of the support of any finite configuration is monotonically increasing under rule 30. As a consequence, one can easily determine the number of steps in the evolution one has to follow to determine whether configurations are reachable or confluence. However, a modified version of Reachability where only a part of a configuration (perhaps a two-dimensional patch in the usual unfolding) is specified, could potentially have any r.e. degree.

The existence of cellular automata with intermediate Reachability and Confluence problems suggests that the whole standard classification into four types as in [3, 13, 16], three of which yield decidable orbits, is perhaps too coarse. Indeed one can show that for any r.e. degree there is a cellular automaton whose Reachability Problem has that particular degree, see [12]. Within this classification natural subclasses have decision problems of complexity Σ_4^d where d is the degree in question. For example, it is Σ_3^0 -complete to determine whether a cellular automaton has decidable Reachability Problem, but Σ_4^0 -complete to determine whether it is computationally universal.

Acknowledgments

I am grateful to the unknown referees who have helped in improving the presentation of this paper. Stephen Wolfram's skepticism about intermediate degrees has been a very productive challenge.

References

- [1] Baldwin, J. T., Shelah, S.: On the Classifiability of Cellular Automata, *Theoretical Computer Science*, **230**(1-2), 2000, 117–129.
- [2] Davis, M.: *A note on universal Turing machines*, Princeton University Press, 1956, 172–175.
- [3] Culik, K., Yu, S.: Cellular Automata, $\omega\omega$ -regular Sets, and Sofic Systems, *Discrete Applied Mathematics*, **32**, 1991, 85–101.
- [4] Lerman, M.: *Degrees of Unsolvability*, Perspectives in Mathematical Logic, Springer Verlag, 1983.
- [5] Shepherdson, J. C.: Machine Configuration and Word Problems of Given Degree of Unsolvability, *Zeitschrift f. Math. Logik u. Grundlagen d. Mathematik*, **11**, 1965, 149–175.
- [6] Shoenfield, J. R.: *Mathematical Logic*, Addison Wesley, 1967.
- [7] Soare, R. I.: *Recursively Enumerable Sets and Degrees*, Perspectives in Mathematical Logic, Springer Verlag, 1987.
- [8] Sutner, K.: A note on Culik-Yu classes, *Complex Systems*, **3**(1), 1989, 107–115.
- [9] Sutner, K.: Classifying Circular Cellular Automata, *Physica D*, **45**(1–3), 1990, 386–395.
- [10] Sutner, K.: De Bruijn Graphs and Linear Cellular Automata, *Complex Systems*, **5**(1), 1991, 19–30.
- [11] Sutner, K.: The Complexity of Finite Cellular Automata, *Journal of Computer and Systems Sciences*, **50**(1), 1995, 87–97.
- [12] Sutner, K.: Cellular Automata and Intermediate Degrees, To appear in TCS.
- [13] Wolfram, S.: Computation Theory of Cellular Automata, *Comm. Math. Physics*, **96**(1), 1984, 15–57.
- [14] Wolfram, S.: Universality and Complexity in Cellular Automata, *Physica D*, **10**, 1984, 1–35.
- [15] Wolfram, S.: Private communication, 1999.
- [16] Wolfram, S.: *A New Kind of Science*, Wolfram Media, 2002.
- [17] Yaku, T.: The Constructibility of a Configuration in a Cellular Automaton, *Journal of Computer and Systems Sciences*, **7**, 1973, 481–496.