Descriptive complexity of sensitivity for cellular automata

Tom Favereau[1] and Ville Salo[2]

[1]Mines Nancy engineering school, France,
`tom.favereau@etu.mines-nancy.univ-lorraine.fr`
[2]Turku university, Finland, `vosalo@utu.fi`

March 31, 2025

Abstract

We study the computational complexity of determining whether a cellular automaton is sensitive to initial conditions. We show that this problem is Π_2^0-complete in dimension 1 and Σ_3^0-complete in dimension 2 and higher. This solves a question posed by Sablik and Teyssier [ST11].

1 Introduction

Cellular automata are discrete dynamical systems that exhibit complex behavior despite their simple local rules. There have been numerous attempts to classify such systems, with the first notable classification proposed by Wolfram for one-dimensional cellular automata. However, this classification lacked rigorous formality, which led Kůrka [Kur97] to propose a more formal classification of one-dimensional cellular automata based on their sensitivity to initial conditions.

Other classification schemes have also been proposed, such as Culik's classification. The arithmetical complexity of Culik's classes has been established and demonstrated in [Sut89]. The decidability of Kůrka's classes, particularly the problem of sensitivity to initial conditions, was studied by Durand et al. [DFV03], while the reversible case was addressed by Lukkarila [Luk10].

Sablik and Theyssier [ST11] showed that Kůrka's classification no longer holds in higher dimensions and that additional classes must be introduced. In the same article, they investigated the arithmetical complexity of Kůrka's classes. They demonstrated that sensitivity to initial conditions is Π_2^0 in one dimension, but did not prove completeness. For dimensions three and higher, they showed it to be Σ_3^0-hard. However, they did not provide results for two dimensions.

Understanding the dynamical properties of such systems, particularly their sensitivity to initial conditions, is crucial for characterizing their behavior. In this paper, we address these open questions and provide the exact complexity of sensitivity for all dimensions. Specifically, we demonstrate that sensitivity to initial conditions is Π_2^0-complete in one dimension and Σ_3^0-complete in two dimensions and higher. As a corollary, we prove that the finite nilpotency problem for cellular automata is Π_2^0-complete. Additionally we explore an application of our results by constructing a cellular automaton whose sensitivity is equivalent to the truth of the twin prime conjecture. These results not only fill the gaps left by previous research but also exposes the precise difference in complexity between one-dimensional and higher-dimensional cellular automata with respect to sensitivity.

2 Preliminaries

We will first recall some basic definitions that we will need throughout this document.

Definition 1 (Cellular Automaton)**.** A cellular automaton is a quadruple (d, S, N, f) where d is the dimension of the CA, S is a finite set of states, N is a finite subset of $\mathbb{Z}^d$ called the neighborhood, and $f : S^N \to S$ is the local transition function. This induces a global function $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$.

The sensitivity to initial conditions is a property of dynamical systems that states that for any configuration, we can always find an arbitrarily close configuration that tends to diverge from our original configuration. First we endow $S^{\mathbb{Z}^d}$ with the following metric:

$$d(x, y) = 2^{-\min\{\|n\|_\infty | x(n) \neq y(n)\}}$$

We formalize the concept of sensitivity in the following definition:

Definition 2 (Sensitivity)**.** A cellular automaton F is **sensitive** to initial conditions if there exists $\epsilon > 0$ such that for all configurations x and $\delta > 0$, there exists $y \in B_\delta(x)$ and $n \in \mathbb{N}$ such that $d(F^n(x), F^n(y)) > \epsilon$.

In one dimension, a famous result states a connection between sensitivity and having words that block information, i.e., blocking words. Blocking words are characterized by a length that no information can cross. Indeed, to truly prevent information from propagating, such words must have a length larger than the automaton's radius. We provide a definition of an m-blocking word:

Definition 3 (m-blocking word)**.** A word $w \in S^*$ together with an integer p is **m-blocking** if for any configurations u, v containing w as a factor at the same position, for all $n \in \mathbb{N}$, $F^n(u)_{|[p,p+m]} = F^n(v)_{|[p,p+m]}$.

Hence, we can state the well-known characterization of sensitivity in one dimension:

Theorem 1. A one-dimensional cellular automaton with radius r is not sensitive if and only if it has an r-blocking word.

We will also need a notion of blocking word in higher dimensions. In dimension d, we want an m-blocking word to be any pattern that contains a subpattern of size m^d that no information can cross.

Definition 4 (Higher dimensional m-blocking word)**.** For any finite subset $K \subset \mathbb{Z}^d$ together with a vector $p \in \mathbb{Z}^d$ such that $\sigma^p([0,m]^d) \subseteq K$, a pattern $\pi \in S^K$ is **m-blocking** if for any configurations u, v containing π at the position K, for all $n \in \mathbb{N}$, $F^n(u)_{|\sigma^p([0,m]^d)} = F^n(v)_{|\sigma^p([0,m]^d)}$.

With that definition, we state the following lemma that shows how one can extend the blocking word characterization for higher-dimensional cellular automata.

Lemma 1. A cellular automaton is not sensitive if and only if it has arbitrarily large blocking words.

Proof. If F is a sensitive cellular automaton, there exists $\epsilon > 0$ such that for all x and $\delta > 0$, there exists $y \in B_\delta(x)$ and n such that $d(F^n(x), F^n(y)) > \epsilon$. This means that there exists $m > -\log_2(\epsilon)$ such that arbitrarily close configurations will eventually differ on $[0,m]^d$, hence there is no m-blocking word.

Conversely, if F is not sensitive, for every $\epsilon > 0$ and $m > -\log_2(\epsilon)$, we can find a pattern π such that every configuration that matches with π at position 0 will never differ on $[0,m]^d$. Hence, we have arbitrarily large blocking words. $\square$

Turing machines are abstract computational models that provide a formal definition of a computable function. These machines can perform basic operations such as reading, writing, and moving the head. Their power lies in their ability to simulate any algorithm or computational process. We will use them to prove our complexity results.

A Turing machine is formally defined as a 4-tuple (Q, A, δ, Γ) where Q is a finite set of states, including an initial state q_i and a halting state q_h; A is a finite alphabet that includes a blank symbol; $\Gamma \subseteq A$ is the input alphabet; and $\delta : Q \times A \to Q \times A \times \{-1, 0, 1\}$ is the transition function, such that for all $a \in A$, $\delta(q_h, a) = (q_h, a, 0)$.

The transition function δ determines the machine's behavior: given a current state and a symbol under the head, it specifies the next state, the symbol to write, and the direction to move the head (-1 for left, 0 for stay, 1 for right).

Given an enumeration of Turing machines, we denote $W_e := \{x \mid M_e(x) \downarrow\}$ the set of inputs on which the machine halts. We introduce the two sets we will use in our proof:

$$\mathrm{TOT} := \{e \mid W_e = \mathbb{N}\}$$

and

$$\mathrm{COF} := \{e \mid W_e \text{ is cofinite}\}$$

Lemma 2. TOT is Π_2^0-complete and COF is Σ_3^0-complete.

Proof. The reader should find a proof of these results in [Rog87]. □

Finally, we state a basic result: one-way tape Turing machines are equivalent to two-way tape Turing machines.

Proposition 1. Given a two-way Turing machine M_2, there exists a one-way Turing machine M_1 such that for all $x \in \mathbb{N}$, M_2 halts on x if and only if M_1 halts on x.

Proof. The backward direction is trivial. For the converse, we can simulate any two-way Turing machine on a one-way tape by instructing the machine, when it wants to move left to 0, to copy all its non-blank symbols one step to the right. □

In particular, this proposition allows us to consider the sets TOT and COF as sets of indices of one-way Turing machines.

3 Main Results

In this section, we present our main results concerning the complexity of determining sensitivity for cellular automata. These theorems provide a complete characterization of the sensitivity's problem complexity across different dimensions.

For one-dimensional cellular automata, we establish:

Theorem 2. The problem of determining whether a one-dimensional cellular automaton is sensitive is Π_2^0-complete.

For higher dimensions, we state:

Theorem 3. For $d > 1$, the problem of determining whether a d-dimensional cellular automaton is sensitive is Σ_3^0-complete.

The proofs of these theorems rely on embedding Turing machines into cellular automata and reducing the problems TOT and COF to the respective sensitivity problem. We present the proof of Theorem 2 in Section 4, and the proof of Theorem 3 in Section 5. Applications are given in Section 6.

4 One-dimensional Cellular Automata

In this section, we prove our first result: the Π_2^0-completeness of the sensitivity problem for one-dimensional cellular automata. We begin with a simple lemma.

Lemma 3. The problem of determining whether a given cellular automaton has an m-blocking word is in Σ_2^0.

Proof. Observe that in the definition of a blocking word, we only need to quantify over finite objects. Hence, having an m-blocking word can be written as:

$\exists K \subset \mathbb{Z}^d$ finite, $\exists p \in \mathbb{Z}^d, \exists w \in S^K, \forall u, v \in (S^d)^*, n \in \mathbb{N}, (\sigma^p([0,m]^d) \in w \wedge w \in u \wedge w \in v) \implies F^n(u)_{|\sigma^p([0,m]^d)} = F^n(v)_{|\sigma^p([0,m]^d)}$

This formulation clearly places the problem in Σ_2^0. □

4.1 Upper Bound on Complexity of the Sensitivity Problem

The sensitivity problem for one-dimensional cellular automata is in Π_2^0. The proof follows directly from Lemma 3. We note that this result was already known from Sablik and Teyssier [ST11].

Lemma 4. The problem of determining whether a one-dimensional cellular automaton is sensitive to initial conditions is in Π_2^0.

Proof. A cellular automaton is sensitive to initial conditions if and only if it does not have a blocking word of any length. This is clearly a Π_2^0 formula, as it negates a Σ_2^0 formula (from Lemma 3). □

4.2 Construction of G_e

To prove Π_2^0-hardness, we reduce from TOT, which is known to be Π_2^0-hard. Given a number e in, we denote M_e as the associated Turing machine. Let us first recall that we do not change the problem by requiring M_e to operate only on a one-way infinite tape.

It is important to note that we must consider all possible configurations, including those where the Turing machine has performed an incorrect computation. These "degenerate" configurations arise because all configurations can be initial configuration. We will explain how to handle this issue in the subsequent parts of the construction.

We construct a cellular automaton G_e as follows:

The state set of G_e is $(\Gamma \cup \{\sqcup\}) \times (Q_e \times A) \times (\{<, >, p\} \cup X)$, where: Γ is the input alphabet of M_e, A is the tape alphabet of M_e, $<$ and $>$ are delimiter symbols that partition the space into computational blocks, X is a set of elements that handle the extension of the computational zone and restart the Turing machine on the input tape to prevent degenerate configurations and Q_e is the set of states of M_e.

It is a three-tape cellular automaton with an input tape that allows computation to restart, a working tape where the Turing machine performs computations, and a delimiter tape that manages the computational zones.

The cellular automaton G_e simulates multiple copies of M_e, where each copy operates within its own computational block, delimited by sequences of $<$ and $>$ symbols. A typical configuration is shown in Figure 1.

The transition rules of G_e enforce the following behaviors:

$\cdots$	$>$	$>$	$>$	$>$	x_l	$<$	$<$	$<$	$>$	$>$	x_r	$<$	$<$	$\cdots$
$\cdots$	a'	a'	b'	q, b'	c'	c'	d'	d'	a'	b'	q, c'	d'	a'	$\cdots$
$\cdots$	a	a	b	b	c	c	d	d	a	b	c	d	a	$\cdots$

Figure 1: Structure of computational blocks in G_e. The upper row shows the delimiters and machine heads, while the lower rows show the tape content.

1. The set X contains x_r, which moves to the right on computational blocks, x_l, which moves to the left, and x_0, which moves to the left while resetting the computation on the witness tape. Symbols move back and forth in the computational zone and always try to extend it to the right when possible. When this happens, the Turing machine restarts on the input tape (preventing degenerate configurations).

2. `<` and `>` delimit computational blocks. x_r can only move to the right if it is to the left of a `<`; otherwise, it becomes an x_l. x_0 and x_l can only move to the left if they are to the right of a `>`; otherwise, they become an x_r.

3. The blank symbol on the witness tape propagates to the right on the computational block, preventing the input from being divided by a blank.

4. When a machine halts, it writes a blank symbol that propagates throughout the computational block.

5. The x symbol can extend its computational block if and only if the machine to its right has been destroyed.

6. p is a particle that always travels to the right on blank symbols of the first tape and gets destroyed when it encounters non-blank cells. It allows us to establish sensitivity.

We provide the complete rule of the cellular automaton G_e in Figure 2.

4.3 Lower Bound and Π_2^0-Hardness

Lemma 5. For any $e \in \mathbb{N}$, the Turing machine M_e halts on all inputs if and only if the cellular automaton G_e is sensitive.

Proof. We now prove that $e \in \mathrm{TOT}$ if and only if G_e is sensitive.

If $e \in$ TOT: We need to show that G_e is sensitive. Let x be any configuration and P any pattern in x. We can take $y \in \mathrm{cyl}(P)$ such that y has only blank symbols (on all tapes) to the right of P, and to the left of P, there is an p particle moving to the right and blank symbols.

Since $e \in \mathrm{TOT}$, all blocks in P will eventually be destroyed. Indeed, in the rightmost computational block in P, the x symbols will be able to extend the

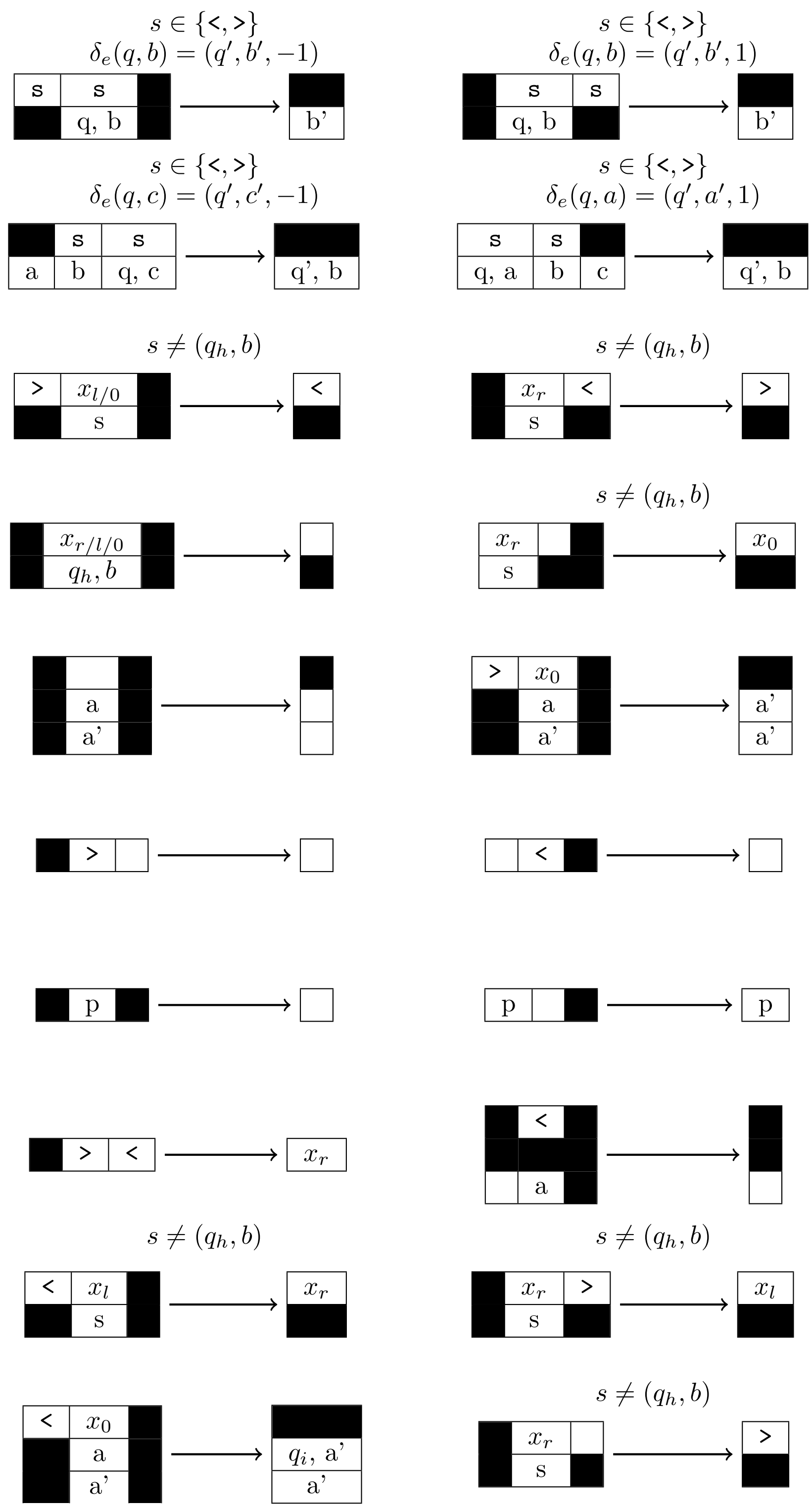

Figure 2: Transition function of the cellular automaton G_e

computational block arbitrarily far, allowing the Turing machine to compute on its input without being restarted and with arbitrary large space. Hence, the rightmost Turing machine will halt and erase itself in finitely many steps, leaving space for the machine to its left. Therefore, in finitely many steps, P will become blank. Thus, the particle we placed to the left of P can traverse through P. This establishes sensitivity.

If $e \notin$ TOT: Then there exists an input n such that M_e runs forever on n without halting. Consider the word pattern uu where u is a computational block of size radius of G_e containing input n. The machine simulated in the second copy of u will run forever and never be destroyed. Consequently, the machine in the first copy of u is confined to its computational block and will also run forever. This creates a barrier that no information can cross. Therefore uu is an r-blocking word, proving that G_e is not sensitive.

□

Thus, we have shown that G_e is sensitive if and only if $e \in$ TOT. Since the TOT problem is Π_2^0-hard,

Proof of Theorem 2. By Lemma 4, we know that the sensitivity problem is in Π_2^0, and by Lemma 5, we have hardness. We conclude that determining sensitivity for one-dimensional cellular automata is Π_2^0-complete. This proves the theorem. □

Corollary 1. The problem of determining whether a cellular automaton is nilpotent on finite configurations is Π_2^0-complete.

Proof. Given a one dimensional cellular automaton G, finite nilpotency is a Π_2^0 property since it requires

$$\forall w \in A^*, \forall k \in \mathbb{N}, \exists n \in \mathbb{N}, G(0^k w 0^k) = 0.$$

We obtain hardness by reducing TOT and removing particle p in our construction of G_e. Hence, from our previous proof, all finite patterns P will eventually become blank if and only if the given Turing machine halts on every input. We can then lift the result to higher dimensions using slices. Therefore, finite nilpotency is Π_2^0-complete. □

5 Higher Dimensional Cellular Automata

In this section, we extend our analysis to cellular automata in two or more dimensions and prove Theorem 3. x

5.1 Upper Bound: Σ_3^0

We begin by showing that for any $d > 1$, the sensitivity problem for d-dimensional cellular automata belongs to the arithmetical hierarchy. First, we recall a crucial lemma from our previous discussion, Lemma 1: A cellular automaton is not sensitive if and only if it has an arbitrarily large blocking word. We then state the following lemma.

Lemma 6. For any $d > 1$, the problem of determining whether a d-dimensional cellular automaton is sensitive is Σ_3^0.

Proof. Recall that the property "there exists an M-blocking word" is Σ_2^0 by Lemma 3. Therefore, by Lemma 1, being non-sensitive is Π_3^0, as it requires the existence of blocking words for all sufficiently large M. Consequently, being sensitive is Σ_3^0. This establishes that the sensitivity problem is at most Σ_3^0. $\square$

5.2 Lower Bound: Σ_3^0-Hardness

In this section, we aim to prove the following result:

Theorem 4. The problem of determining whether a two-dimensional cellular automaton is sensitive is Σ_3^0-complete.

To prove Σ_3^0-hardness, we reduce from the Cofinite Set (COF) problem, which is known to be Σ_3^0-hard. For each e, let M_e be the associated Turing machine. We construct a 2D cellular automaton G_e as follows:

The states of G_e are $A \times Q_e \times T \times P \times \{\text{red1}, \text{red2}, \text{red3}, \text{red4}, \text{white}, \text{green}\}$, where A is the tape alphabet of M_e, Q_e is the set of states of M_e, P is a set of particles, T is a set of symbols for tentacles and the colors red, white, and green delimit the computational zones. We describe the cellular automaton's behavior as follows:

Each red cell can have at most two red neighbors. A red cell marked with 1 can have either a 1 neighbor to the west and a 1 neighbor to the east, a 2 neighbor to the west and a 1 neighbor to the east, or a 4 neighbor to the north and a 1 neighbor to the west. For other marker rules are given figure 3. Consequently, the only valid red zones are those forming spirals. Thus, if we place a finite pattern P in a sea of blank symbols, after finitely many steps, only red spirals will remain. However, a red spiral can either spiral inward, leading to a conflict and the destruction of the zone, or spiral outward, eventually reaching the boundary of P, again causing a conflict and the destruction of the zone. As a result, after finitely many step, the only remaining red zones are are the one forming rectangular loop, representing the tape. The first cell of the tape is chosen to be the bottom-right cell, and the length of the tape is considered as the input.

P is a set of particles. There are two types of particles: p particles, which send a signal to a Turing machine to process one step of computation, and q particles, which place a Turing machine at the beginning of the tape in the initial state. The particles p and q are divided into right and left particles (p_r, p_l) that move to the right and left, respectively. When they meet at the beginning of a tape, they send a signal through that tape. If a particle meets an obstacle (i.e. a red zone) it divide into an up particle and a down particle (p_{ru}, p_{lu}, p_{rd}, p_{ld}) that follow the red loop and reform when meeting again. The key point of this construction is that reversible until particle meets allowing us to send particle where we want outside red zone.

When a signal enters a red zone, it follows the loop guided by the 1, 2, 3, and 4 symbols. When it encounters a Turing machine head, it allows the head to process one step of computation. The Turing machine head is guided along the loop by the markers. If it encounters another head on the tape, it erases that head. When it halts, it erases itself and the red zone.

When a machine runs out of space, it requests the creation of a tentacle that operates with T symbols. Tentacles always deploy to the east, or to the north if there is a red zone in the east position. Hence, a green cell always has exactly two green neighbors, except for the first one, which must have a red neighbor with a Turing machine to the west and the last one. In any other case, the green zone is destroyed. Since red zones are always rectangular and not connected to each other, the green zone can always extend. The set T contains copies of states of M_e and includes a signal s_h that is sent when the machine halts to destroy the connected red zone. By the first rule, when two green zones meet, they annihilate each other.

We give below the main rules of the cellular automaton G_e in Figure 3.

Lemma 7. For any $e \in \mathbb{N}$, the Turing machine M_e halts on all but finitely many inputs if and only if the associated cellular automaton G_e is sensitive.

Proof. We prove that $e \in \mathrm{COF}$ if and only if G_e is sensitive.

If $e \in$ COF: Let x be any configuration and P be a pattern in x. In finite time, P will be constituted of rectangular red loops, white zones, and eventually tentacles. Let us first remark that we can send particles to meet anywhere outside the red zones, as the particle behavior is reversible until the particles meet. Hence, we just need to look backward in time. It follows that the only inaccessible areas are the interiors of the red zones. For any red zone, we can send a particle q to place a Turing machine on the zone. We can then send processing particles in those zones. Because inaccessible zones are surrounded by red zones, if these zones are sufficiently large, they must be surrounded by a large red area (which is the input of the zone). Since $e \in \mathrm{COF}$, there exists an N such that for all $y > N$, M_e halts on y. Therefore, any sufficiently large block will have its border destroyed, allowing us to send particles through it. by Lemma 1, G_e is sensitive.

If $e \notin$ COF: Then there exist infinitely many y such that $M_e(y)$ does not halt. We can construct arbitrarily large blocking words as follows: For any $M > 0$, choose $y > M$ such that $M_e(y)$ does not halt. Construct a red rectangle with perimeter y. This rectangle forms an M-blocking word, as:

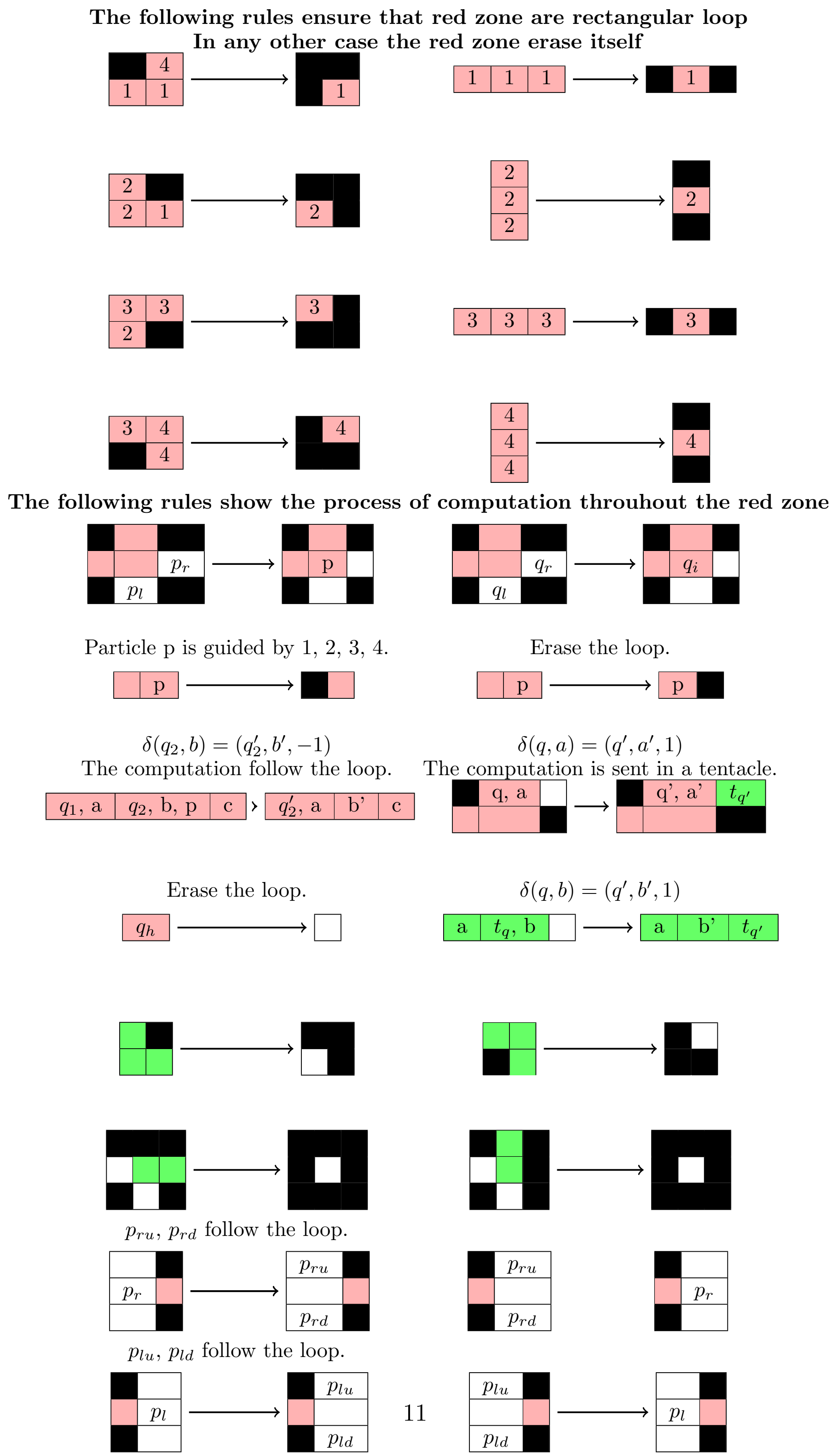

Figure 3: Main rules for the transition function of the 2 dimensional cellular automaton G_e

1. The Turing machine simulating $M_e(y)$ will never halt, so the rectangle will never be destroyed. 2. No particles can penetrate the red border. 3. The interior of the rectangle is inaccessible to any external influence.

Since we can construct such blocking words for arbitrarily large M, by Lemma 1, G_e is not sensitive.

□

Proof of theorem 4. By Lemma 6, we know that the sensitivity problem for two-dimensional cellular automata is Σ_3^0, and by Lemma 7, we have established hardness. Hence, the theorem is proved. □

Proof of theorem 3. By Lemma 6, we know that the sensitivity problem for d-dimensional cellular automata is Σ_3^0. We obtain the hardness by reducing from the two-dimensional problem using slicing. Therefore, the theorem is proved. □

This construction establishes a reduction from COF to the sensitivity problem, proving that the latter is Σ_3^0-hard. Combined with our earlier upper bound, this shows that the sensitivity problem for $n > 1$ dimensional cellular automata is Σ_3^0-complete.

6 Application

In this section, we explore applications of our results in number theory, specifically in relation to the twin prime conjecture.

Proposition 2. The twin prime conjecture is a Π_2^0 statement in the arithmetical hierarchy.

Proof. The twin prime conjecture can be stated as:

$$\forall n, \exists p (p > n \wedge \text{Prime}(p) \wedge \text{Prime}(p+2))$$

□

We construct a Turing machine M that, given an input n in unary (represented by a string of 0s), halts if and only if it finds a pair of twin primes larger than or equal to n.

Theorem 5. There exists a cellular automaton G_e that is sensitive to initial conditions if and only if the twin prime conjecture is true.

Proof. By our construction, the Turing machine M halts on all inputs if and only if the twin prime conjecture is true. Using our reduction from the TOT problem to the sensitivity problem for cellular automata, we can construct a cellular automaton G_e that is sensitive to initial conditions if and only if M halts on all inputs. Therefore, G_e is sensitive to initial conditions if and only if the twin prime conjecture is true. □

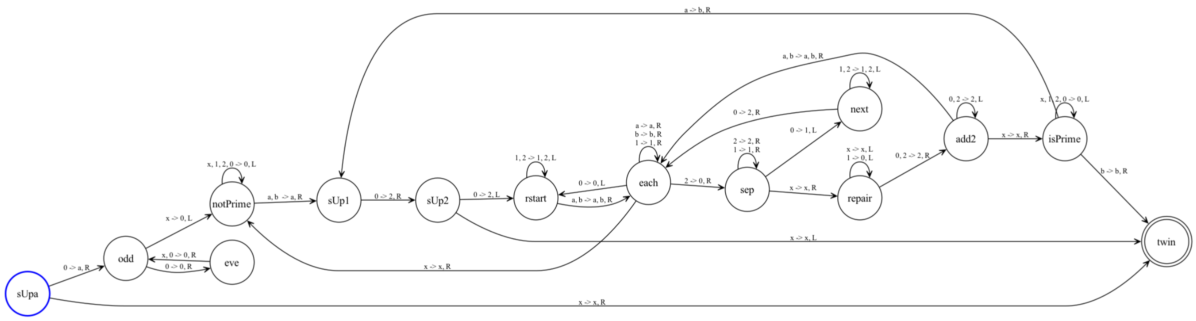

Figure 4: A turing machine that halts on input n if there is twin primes greater than n

We have implemented this cellular automaton, which can be explored interactively at the following URL: https://tom-favereau.github.io/misc/ca.html. The source code for the automaton is also available on GitHub at https://github.com/tom-favereau/twin_prime_automaton.

While we do not believe this to be a suitable approach for solving the twin prime conjecture, we argue that it demonstrates concrete applications of the arithmetic hierarchy that are not often explored. Furthermore, this construction illustrates how complexity classifications can provide unexpected connections between different areas of mathematics.

7 Conclusion and Future Work

In this paper, we have demonstrated that the problem of determining sensitivity to initial conditions for cellular automata always falls within the arithmetic hierarchy. Specifically, we have shown that this problem is Π_2^0-complete for one-dimensional cellular automata and Σ_3^0-complete for cellular automata of dimension two and higher. These results provide a complete characterization of the complexity of the sensitivity problem across all dimensions.

Additionally, we have proven that the problem of determining finite nilpotency for one-dimensional cellular automata is Π_2^0-complete. This result further enriches our understanding of the complexity landscape for cellular automata properties. We have also constructed a cellular automaton that is sensitive to initial conditions if and only if the twin prime conjecture is true.

As a corollary to our main results, we can conclude that the problem of determining non-sensitivity is Σ_2^0-complete for one-dimensional cellular automata and Π_3^0-complete for higher dimensions. This complementary result complete our analysis of Kůrka's classes.

However, it is important to note that our reductions are not reversible. Consequently, the complexity of the sensitivity problem for reversible cellular automata remains an open question. This presents an interesting avenue for future research, as reversible cellular automata form an important subclass with unique properties and applications. However it should be solved with some trivial techniques that we won't develop here.

Furthermore, our work does not address the case of expansive cellular automata, which constitute the final class in Kůrka's classification. For expansive cellular automata, we not only lack results concerning their place in the arithmetic hierarchy but also have no definitive answer regarding the decidability or undecidability of the problem. This gap in our understanding presents another significant direction for future investigations.

In conclusion, while our results provide a comprehensive complexity analysis for the sensitivity problem in general cellular automata, they also highlight important open questions and future directions.

References

[DFV03] Bruno Durand, Enrico Formenti, and Georges Varouchas. On undecidability of equicontinuity classification for cellular automata. *Discrete Mathematics & Theoretical Computer Science*, DMTCS Proceedings vol. AB, Discrete Models for Complex Systems (DMCS'03), Jan 2003.

[Kur97] Petr Kurka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory and Dynamical Systems*, 17(2):417–433, April 1997.

[Luk10] Ville Lukkarila. *On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata*. Ph.d. thesis, University of Turku, 2010.

[Rog87] Hartley Rogers. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, USA, 1987.

[ST11] M. Sablik and G. Theyssier. Topological dynamics of cellular automata: Dimension matters. *Theory Comput Syst*, 48(4):693–714, 2011.

[Sut89] Klaus Sutner. A note on culik-yu classes. *Complex Systems*, 3(1):107–115, 1989.