Ville Lukkarila

On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata

Turku Centre *for* Computer Science

TUCS Dissertations
No 129, October 2010

On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata

Ville Lukkarila

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Auditorium XXI on October 29, 2010, at 12 noon.

University of Turku
Department of Mathematics
FI-20014, Turku

2010

Supervisors

Professor Jarkko Kari
Department of Mathematics
University of Turku
Finland

Reviewers

Professor Nicolas Ollinger
Laboratoire d'Informatique Fondamentale de Marseille
Université de Provence
France

Professor Ivan Rapaport
Departamento de Ingeniería Matemática
Universidad de Chile
Chile

Opponent

Professor Pekka Orponen
Department of Information and Computer Science
Aalto University
Finland

ISBN 978-952-12-2464-5
ISSN 1239-1883

Abstract

Cellular automata are models for massively parallel computation. A cellular automaton consists of cells which are arranged in some kind of regular lattice and a local update rule which updates the state of each cell according to the states of the cell's neighbors on each step of the computation.

This work focuses on reversible one-dimensional cellular automata in which the cells are arranged in a two-way infinite line and the computation is reversible, that is, the previous states of the cells can be derived from the current ones. In this work it is shown that several properties of reversible one-dimensional cellular automata are algorithmically undecidable, that is, there exists no algorithm that would tell whether a given cellular automaton has the property or not.

It is shown that the tiling problem of Wang tiles remains undecidable even in some very restricted special cases. It follows that it is undecidable whether some given states will always appear in computations by the given cellular automaton. It also follows that a weaker form of expansivity, which is a concept of dynamical systems, is an undecidable property for reversible one-dimensional cellular automata.

It is shown that several properties of dynamical systems are undecidable for reversible one-dimensional cellular automata. It shown that sensitivity to initial conditions and topological mixing are undecidable properties. Furthermore, non-sensitive and mixing cellular automata are recursively inseparable. It follows that also chaotic behavior is an undecidable property for reversible one-dimensional cellular automata.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Professor Jarkko Kari. His guidance and expertise have been priceless to my doctoral studies. This thesis contains results which have been achieved with the guidance of and in collaboration with Professor Kari. Without his expert guidance and broad knowledge, I would not have completed this work.

I am grateful to Professors Nicolas Ollinger and Ivan Rapaport for reviewing the thesis manuscript and for providing valuable comments. I am grateful to Professor Pekka Orponen for accepting the invitation to act as an opponent at the public disputation of my thesis.

I thank Professor Juhani Karhumäki, the Department of mathematics and Turku Centre for Computer Science (TUCS) for providing excellent working conditions. This work was financially supported (in chronological order) by Turku University Foundation, Maili Autio Fund of the Finnish Cultural Foundation, Fund of Vilho, Yrjö and Kalle Väisälä of the Finnish Academy of Science and Letters, and TUCS. The financial support is most gratefully acknowledged.

I would also like to thank Dr. Ari Renvall for giving his support to my doctoral studies. I also thank the personnel of both the Department of Mathematics and TUCS for a pleasant working environment.

Finally, I would like to thank my parents and Hanna for all the loving support.

Turku, August 2010 Ville Lukkarila

Contents

Chapter 1

Introduction

The main topic of this work is reversible cellular automata and their algorithmically undecidable dynamical properties. Parts of this work have appeared in [53, 67, 66, 68, 69, 70, 71].

Cellular automata are a simple formal model for the study of phenomena caused by local interaction of finite objects. A *cellular automaton* consists of a regular lattice of cells. Each cell has a state which is updated on every time step according to some rule which is the same for all the cells in the lattice. The locally used update rule is simply called a *local rule*. On every time step the next state of the cell is determined according to its own previous state and the previous states of a finite number of its neighbors. The state information of the entire lattice at any time step is called a *configuration* of the cellular automaton.

Cellular automata were introduced by von Neumann to study biologically motivated computation and self-replication [103]. The mathematical study of cellular automata in symbolic dynamics was initiated by Hedlund [38]. Although cellular automata may seem a simple model for computation, they can exhibit very complex behavior. A well-known example of such complex behavior is the Game of Life. Even though the rule according to which the lattice is updated is quite simple in the Game of Life, some state patterns interact in a somewhat complex manner. In fact, the Game of Life has been shown to be computationally universal. In particular, any Turing machine can be simulated with some cellular automaton in a natural way.

Cellular automata have been studied very extensively also as discrete time dynamical systems. Properties of dynamical systems such as injectivity, surjectivity, nilpotency, equicontinuity, sensitivity to initial conditions, topological transitivity, topological mixing, chaos, and different variants of expansivity have been studied in terms of cellular automata also. In particular, some properties have been studied in the sense of algorithmic decidability and undecidability. Nilpotency is an undecidable property even for

one-dimensional cellular automata [46, 48]. Injectivity and surjectivity are known to be decidable for one-dimensional cellular automata but undecidable for two-dimensional cellular automata [47]. It was mentioned in [24] that sensitivity, equicontinuity, transitivity and ergodicity are believed to be undecidable properties of cellular automata but no proof was given. It was shown in [28] that equicontinuity and sensitivity are undecidable properties for not necessarily reversible one-dimensional cellular automata. Recently, it was shown by Kari and Ollinger that equicontinuity is undecidable even for reversible one-dimensional cellular automata [54].

Some properties of cellular automata have been studied using *Wang tiles* and Wang tiles will be used in this work also. A Wang tile (or simply a *tile*) is a unit square tile with colored edges and a Wang tile set is a finite set of Wang tiles. A tiling by Wang tiles is a function which assigns a unique tile from the given tile set for each location of the plane. A tiling is said to be valid if the colors of neighboring tiles match. The tiling problem of Wang tiles (also known as the domino problem) is the decision problem of determining whether or not a given finite tile set admits a valid tiling. It was shown by Berger [7] that the tiling problem is undecidable. A simplified proof was given later by Robinson [92]. Both the proof of Berger and the proof of Robinson relied on the existence of an aperiodic Wang tile set, i.e. a Wang tile set which admits only non-periodic valid tilings.

A Wang tile set is said to be *2-way deterministic* if there are no two tiles in the tile set that have the same colors next to the bottom left corner or the top right corner. This means that the colors of the edges adjacent to the bottom left corner determine rest of the edge colors uniquely. The same holds also for the color of the edges adjacent to the top right corner. A Wang tile set is said to be *4-way deterministic* if there are no two tiles in the tile set that have the same colors next to any given corner. Then the tile is uniquely determined by the colors of the edges adjacent to any corner.

In Chapter 3 it is shown that the tiling problem of Wang tiles remains undecidable even if the structure of the given Wang tile set is restricted to 4-way deterministic tile sets and even if at most one kind of Wang tile can be placed between any two Wang tiles. It is known that 4-way deterministic aperiodic tile sets exist [55] but it has not been known whether or not the tiling problem is undecidable in the 4-way deterministic case. It is shown that the square tiling problem [35] remains NP-complete even when the instances are 4-way deterministic tile sets. This chapter is based on [67, 68, 70].

In Chapter 4 it is shown how a 2-way deterministic tile sets can be used to represent a Turing machine computation. This construction gives a simple description of a result of Dubacq stating that any (not necessarily reversible) Turing machine can be simulated in real time by a reversible one-dimensional cellular automaton [27]. This construction also gives a proof for

the undecidability of the tiling problem in the weaker 2-way deterministic case. Although the result is weaker, the construction is significantly simpler than the one in Chapter 3 and it is sufficient to be used in Chapter 5 since 4-way determinism is not required. This chapter is based on [66].

In Chapter 5 it is shown that left (or right) expansivity of reversible cellular automata is an undecidable property. An irreversible one-dimensional cellular automaton is said to be *positively expansive* if a difference between two different configurations propagates both to the left and to the right during a computation forward in time. A reversible one-dimensional cellular automaton is said to be *expansive* if a difference between two different configurations propagates both to the left and to the right during a computation forward and backward in time. A reversible one-dimensional cellular automaton is said to be *left expansive* if a difference between two different configurations propagates to the right during the computation forwards and backwards in time. This definition differs from the original definition given by Kurka [59] where only forward computation was considered in the sense of positive expansivity. From either one tile set construction (2-way or 4-way) it follows that it is undecidable whether a given one-dimensional cellular automaton is left expansive or not. This chapter is based on [53]. The undecidability status of expansivity in the reversible case and positive expansivity remain open problems.

In Chapter 6 it is shown that sensitivity to initial conditions is an undecidable property even for reversible one-dimensional cellular automata. Sensitivity to initial conditions (or *sensitivity*) means that from any open neighborhood of a configuration it is possible to pick another configuration so that the iterated images of the two configurations are no longer located within a same open neighborhood with a fixed general radius. Intuitively, sensitivity means that small changes in the initial configuration can always (for any configuration) result relatively large changes in the iterated image configurations after some number of time steps. It has been known that sensitivity is an undecidable property for not necessarily reversible one-dimensional cellular automata [28]. This chapter is based on [69, 71].

In Chapter 7 it is shown that topological mixing and topological transitivity are undecidable properties even for reversible one-dimensional cellular automata. In fact, non-sensitive reversible one-dimensional cellular automata and topologically mixing reversible one-dimensional cellular automata are recursively inseparable. This follows from a simple modification to the construction given in Chapter 6. Due to the close relation between transitivity and different definitions of chaotic behavior, it follows that chaotic behavior is an undecidable property both according to the definition of Devaney [26] and according to the definition of Knudsen [57]. This chapter is based on [71].

In Chapter 8 expansivity and sensitivity of linear cellular automata are briefly discussed. A cellular automaton is linear if the set of configurations admits an additive operation and the cellular automaton function is linear in this sense. The linear cellular automata are considered with the definition of [12, 49]. It is shown that expansivity, positive expansivity and sensitivity to initial conditions are decidable properties for linear cellular automata whose state sets consist of vectors over a ring. Also some minor linear algebraic results are presented. It is shown that a linear rule F is expansive if, and only if, the linear rule $F + F^{-1}$ is positively expansive. This chapter is previously unpublished work.

Many questions regarding expansive dynamical system and cellular automata remain open, including the following:

Open Problem Is expansivity an undecidable property for reversible cellular automata?

Open Problem Is positive expansivity an undecidable property for cellular automata?

Open Problem Is every expansive (reversible) cellular automata conjugate to a two-sided subshift of finite type?

Chapter 2

Preliminaries

2.1 Algorithms and undecidability

In this section the mathematical concept of algorithm is briefly reviewed. Due to the extensive theory which is connected to algorithms and undecidability, only the most basic definitions and notions will be presented and in an informal manner.

2.1.1 Algorithms

Recall that a *word* is a finite sequence of elements from a set Σ. The set Σ in then called an *alphabet*. A word (i.e. sequence) $(a_1, a_2, \ldots, a_n)$ is usually written in the form $a_1 a_2 \ldots a_n$. The empty sequence (i.e. ()) with no elements is called the *empty word* and it is denoted by ε. The set of all words over alphabet Σ is denoted by Σ^*. A *language* is a set of words.

In this work, an *algorithm* is a deterministic mechanical procedure which receives a finite input (encoded as a word) and after a finite time it halts and outputs an answer. A *decision algorithm* for a set $A \subseteq \Sigma^*$ is a deterministic mechanical procedure that receives a finite input $x \in \Sigma^*$ and outputs "1", if the input belongs to the set, and "0", if the input does not belong to the set. No matter what word is given to the algorithm, it always eventually halts and returns an answer.

A *semi-algorithm* of a set $A \subseteq \Sigma^*$ is a deterministic mechanical procedure that receives a finite input and outputs "1", if the input belongs to the set, and otherwise it either outputs "0" or does not halt. If the input word belongs to the set, the semi-algorithm returns a correct answer (which is "1"). If the input word does not belong to the set, the computation might never end. However, if the computation ever ends, a correct answer is returned.

The concept of algorithm is somewhat intuitive. However, there exists many strict mathematical models which are assumed to be equivalent to the

intuitive concept of an algorithm (by the so-called Church–Turing thesis). One choice to represent the workings of an algorithm is to represent the algorithm as a Turing machine. The concept of Turing machines is briefly reviewed in Section 2.1.2.

2.1.2 Turing machines

In what follows, only the most basic definitions of Turing machines are represented in an informal manner. A more thorough discussion of Turing machines can be found in [25, 40, 74].

A Turing machine is usually pictured as a simple mechanical device which has a finite instruction set, a finite memory and which scans an infinite tape back and forth. The tape contains cells and each of the cells can contain one letter from a finite alphabet. The mechanical part of the Turing machine is called the *(read-write) head* and it has a *state* (or *(read-write head) state*) from a finite state alphabet.

The head is always located on top of a single cell and on every time step the head reads the letter from the cell. Then, if the instruction set identifies the combination of the head's state and the tape letter, the machine replaces the tape letter and the state elements with new ones and possibly moves the head a one location either to the left or to the right. If no instruction is found for the combination of the machines inner state and the letter on the tape, the machine simply halts. The operation conducted by the read-write head is called a *transition* of the Turing machine. There exists different definitions of Turing machine depending on how the head is defined to move on each step of the computation. In particular, in this work two different, but computationally equivalent, definitions (Definitions 2.1.4 and 2.1.3) are used for Turing machine transitions depending on their suitable special properties.

There exist different variants of Turing machines with multiple tapes or read-write heads and also the tape can be infinite either to one direction only or to both directions. Different definitions for Turing machine appear in [54, 83, 92]. In addition, sometimes the tape is considered to move instead of the read-write head [54]. In this work only Turing machines with a moving read-write head are considered.

Definition 2.1.1. *A* Turing machine $\mathcal{M}$ *(or* TM *in short) is a quadruple*

$$\mathcal{M} = (Q, \Gamma, T, q_0),$$

where Q *is the* state set*,* Γ *is the* tape alphabet*,* T *is the* transitions table *and* $q_0 \in Q$ *is the* initial state*.*

In this work, the tape of a Turing machine is defined to be two-way infinite and symbol ε is the *empty tape letter* which denotes an empty tape

cell. The directions of the tape are referred to as *left* and *right*. The *read-write head* of a Turing machine is an object which moves on top of the tape and may modify the contents of each individual cell whenever it is located on top of the cell.

A *configuration* of a Turing machine (with a moving head) is an expression of the form

$$(q, i, c) \in Q \times \mathbb{Z} \times \Gamma^{\mathbb{Z}},$$

where q is the inner state of the read-write head, i is the current location of the read-write head and c is the current tape contents. The tape in a configuration is said to be *empty* if $c(j) = \varepsilon$ holds for every cell $j \in \mathbb{Z}$. A configuration is called *infinite* if there are infinitely many cells $j \in \mathbb{Z}$ such that $c(j) \neq \varepsilon$. Likewise, a configuration is called *finite* if there are finitely many cells $j \in \mathbb{Z}$ such that $c(j) \neq \varepsilon$. A Turing machine configuration is sometimes called an *instantaneous description* [39].

The transitions are the Turing machine's instruction set. The definition for the set of transitions varies in the literature and in this work two different definitions are used.

Remark 2.1.2. *The constructions in Chapter 3 and Chapter 4 use Definition 2.1.3 as was done already in Robinson's article [92].*

However, the construction in Chapter 6 uses Definition 2.1.4 because the construction relies on the reversible Turing machine halting problem. For the construction in Chapter 6 to work, it would be enough to consider "injective" Turing machines (see Remark 6.2.1, p. 94).

It is straightforward to see that that the definitions are equivalent in the sense that a computation by any Turing machine according to one definition can be simulated by a computation by some Turing machine according to the other definition.

Definition 2.1.3 (TM transitions as used in [35, 74, 92]). *A* transition table *is a set*

$$T \subseteq Q \times \Gamma \times Q \times \Gamma \times \{\triangleleft, \triangleright\}.$$

An element $(q, a, r, b, \star) \in T$ is interpreted so that the read write head in state q reading letter a does one of the following:

1. *It writes a letter b on the cell, switches to a new state r and moves one cell to the left if $\star = \triangleleft$. Then configuration (q, i, c) is mapped to configuration $(r, i - 1, d)$, where $d(j) = b$ if $j = i$ and $d(j) = c(j)$ otherwise.*

2. *It writes a letter b on the cell, switches to a new state r and moves one cell to the right if $\star = \triangleright$. Then configuration (q, i, c) is mapped to configuration $(r, i + 1, d)$, where $d(j) = b$ if $j = i$ and $d(j) = c(j)$ otherwise.*

That is, on every time step the read-write head first writes and then moves.

Definition 2.1.4 (TM transitions as used in [25, 54, 83]). *A* transition table *is a set*

$$T \subseteq (Q \times \Gamma \times Q \times \Gamma) \cup (Q \times Q \times \{\triangleleft, \triangleright\}) .$$

1. *An element* $(q, a, r, b) \in Q \times \Gamma \times Q \times \Gamma \subseteq T$ *is interpreted so that the read write head in state* q *reading letter* a *writes a letter* b *on the cell and switches to a new state* r*. Then configuration* (q, i, c) *is mapped to configuration* (r, i, d)*, where* $d(j) = b$ *if* $j = i$ *and* $d(j) = c(j)$ *otherwise.*

2. *An element* $(q, r, \star) \in Q \times Q \times \{\triangleleft, \triangleright\} \subseteq T$ *is interpreted so that the read write head in state* q *does one of the following:*

   (a) *It moves one cell to the left and switches to a new state* r *if* $\star = \triangleleft$*. Then configuration* (q, i, c) *is mapped to configuration* $(r, i-1, c)$*.*

   (b) *It moves one cell to the right and switches to a new state* r *if* $\star = \triangleright$*. Then configuration* (q, i, c) *is mapped to configuration* $(r, i+1, c)$*.*

That is, on every time step the read-write head either moves or writes but not both.

The two models for a Turing machine are computationally equivalent in the sense that any computable function that can be expressed with one transition model can be expressed with the other model also.

Lemma 2.1.5. *For any computation leading from a configuration* (a, i, c) *to a configuration* (b, j, d) *by a Turing machine* A *(resp.* B*) with a transition table according to Definition 2.1.3 (resp. Definition 2.1.4), there exists a Turing machine* A' *(resp.* B'*) with a transition table according to Definition 2.1.4 (resp. Definition 2.1.3) such that it can compute from a configuration* (q_a, i, c) *to a configuration* (q_b, j, d)*.*

Proof. It is shown that for every Turing machine with a transition table of one type it is possible to construct a Turing machine with a transition table of another type so that every single computational transition by the first Turing machine can be achieved as a combination of one or more transitions of the second Turing machine.

Suppose that the Turing machine A has a transition $t = (q, a, r, b, \triangleleft)$. Then the Turing machine A' is extended with transitions (q, a, q_t, b) and $(q_t, r, \triangleleft)$. Likewise, for a transition $s = (q, a, r, b, \triangleright)$, the Turing machine A' would be extended with transitions (q, a, q_s, b) and $(q_s, r, \triangleright)$.

Suppose that the Turing machine B has a transition $t = (q, a, r, b)$. Then the Turing machine B' is extended with transitions $(q, a, q_t, b, \triangleright)$ and $(q_t, x, r, x, \triangleleft)$ for any tape letter x. Suppose that the Turing machine B has a transition $(q, r, \triangleleft)$. Then the Turing machine B' is extended with transitions $(q, x, r, x, \triangleleft)$ for any tape letter x. Likewise, for a transition $(q, r, \triangleright)$, the Turing machine B' is extended with transitions $(q, x, r, x, \triangleright)$ for any tape letter x. □

A Turing machine with a transition table according to Definition 2.1.3 (resp. Definition 2.1.4) is said to *halt* on a configuration (q, i, c), where $c(i) = a$, if there are no transitions of the form $(q, a, r, b, \star)$ (resp. $(q, a, r, \star)$ or $(q, r, \star)$) in its transition table. That is, the Turing machine halts if there is no transition which could be applied to the current configuration.

A Turing machine is said to be *deterministic* if there is at most one transition that can be applied to a configuration. Otherwise, a Turing machine is said to be *nondeterministic*. In terminology, the family of nondeterministic Turing machines is considered to contain deterministic Turing machines also.

If a Turing machine $\mathcal{M} = (Q, \Gamma, T, q_0)$ is deterministic, then the *transition function* δ can be defined in the case of Definition 2.1.3 as

$$\delta(q, a) = \begin{cases} (r, b, \star) & \text{if } (q, a, r, b, \star) \in T \text{ and} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

or in the case of Definition 2.1.4 as

$$\delta(q, a) = \begin{cases} (r, \star) & \text{if } (q, r, \star) \in T, \\ (r, b) & \text{if } (q, a, r, b) \in T \text{ and} \\ \text{undefined} & \text{otherwise} \end{cases}$$

A deterministic Turing machine is said to be *injective* if every configuration has at most one preimage configuration. Although the computation by an injective Turing machine is "reversible" in terms of determining previous computation steps (if they exist), reversibility is defined differently for Turing machines [81, 83, 54]. The idea is to redefine Turing machine in such a way that also the inverse mapping is a Turing machine. For this reason, reversible Turing machines are defined according to [54].

The *reverse* of a transition according to Definition 2.1.4 is defined as

$$(q, a, r, b)^{-1} = (r, b, q, a),\ (q, r, \triangleleft)^{-1} = (r, q, \triangleright) \text{ and } (q, r, \triangleright)^{-1} = (r, q, \triangleleft).$$

The *reverse transition table* T^{-1} is defined as

$$T^{-1} = \left\{ t^{-1} \mid t \in T \right\}.$$

The *reverse* of a Turing machine $\mathcal{M} = (Q, \Gamma, T, q_0)$ is the Turing machine

$$\mathcal{M}^{-1} = (Q, \Gamma, T^{-1}, q_0).$$

A Turing machine (according to Definition 2.1.4) is said to be *reversible* (or *RTM* in short) if it is deterministic and its reverse is a deterministic Turing machine.

2.1.3 Decidability and undecidability

A Turing machine can return its output, for example, by writing on a specific part of the tape a specially formatted string. For example, a Turing machine answering to a yes/no question could give its output as either the letter "1" or the letter "0" written in the cell immediately to the right of the read-write head after it has halted. Alternatively, for as simple a set of outputs as "yes" or "no", the machine could also output its decision by halting in a special state such as q_{yes} and q_{no} (as in [35]). This work does not deal with the practical issues of implementing algorithms on Turing machines and therefore the output method is left undefined.

A set $A \subseteq \Sigma^*$ is said to be *recursive* (or *solvable* or *decidable*) if there exist a Turing machine such that for a given word $x \in \Sigma^*$ it would return "1", if $x \in A$, and "0", if $x \notin A$. The set A is said to be *recursively enumerable* (or *semi-solvable* or *semi-decidable*) if there exists a Turing machine such that for a given word x it would return "1" if, and only if, $x \in A$. Recursive enumerability of a set A means that the elements of A can be printed in some order, although it is not known in which order. A basic result is, that a set $A \subseteq \Sigma^*$ is recursive if, and only if, both A and $A^{\complement}$ are recursively enumerable.

The computational task or *problem* of determining whether or not a given word belongs to a given set A is called the *membership problem*. The membership problem is said to be *decidable*, if A is recursive, and *undecidable* otherwise. In general, a problem is said be *decidable* (or *solvable*), if there exists a Turing machine that always returns a correct answer for the problem. Otherwise, the problem is said to be *undecidable* (or *unsolvable*). A function is called *computable* (or *recursive*), if it can be implemented as a Turing machine, and *uncomputable* (or *non-recursive*) otherwise.

Two sets A and B are said to be *recursively inseparable* if there does not exist a recursive set C such that $A \subseteq C$ and $B \subseteq C^{\complement}$ [56, 91, 93]. In particular, if two sets are recursively inseparable, there does not exist a Turing machine that would distinguish the elements of the first set from the elements of the second set.

2.1.4 Turing machine halting problem

The *Turing machine halting problem* is the following decision problem: "Does the given Turing machine $\mathcal{M}$ eventually halt when started on an empty tape?" The halting problem is known to be undecidable even for reversible Turing machines [6, 62].

Theorem 2.1.6 ([6, 62]). *It is undecidable whether or not a given reversible Turing machine eventually halts when started on an empty tape and the initial state.*

A configuration is said to be *mortal* if the Turing machine eventually halts when it it starts on such a configuration. Conversely, a configuration is said to be *immortal* if the Turing machine never halts when it starts on the configuration.

The *Turing machine immortality problem* is the following decision problem: "Does the given Turing machine $\mathcal{M}$ have an immortal not necessarily finite configuration?" The immortality problem was shown to be undecidable for not necessarily reversible Turing machines in [39]. Later, it was shown to be undecidable for reversible Turing machines in [54]. If a machine eventually halts on every not necessarily finite configuration, there must exist an upper bound on how many computation steps the machine can execute before halting. Therefore the complement of the immortality problem is semi-decidable because for every hypothetical bound m all the finite configurations with only m non-empty cells on both sides of the read-write head can be checked.

2.1.5 Computational complexity

A (nondeterministic) Turing machine $\mathcal{M}$ is said to *accept* a word $x \in \Sigma^*$ if for input x the Turing machine outputs "yes" for some nondeterministic path of computation. That is, if there exists some path of transitions to an positive output, then the word is accepted. The language *recognized* by a nondeterministic Turing machine $\mathcal{M}$ is the set

$$L_{\mathcal{M}} = \{x \in \Sigma^* \mid \mathcal{M} \text{ accepts } x\}.$$

The *time* required by a nondeterministic Turing $\mathcal{M}$ machine to accept a word $x \in L_{\mathcal{M}}$ is defined to be minimum, over all accepting computations of M for x, of the number of computation steps until a result is returned [35].

The *time complexity function* $T_{\mathcal{M}} : \mathbb{N} \to \mathbb{N}$ for $\mathcal{M}$ is

$$T_{\mathcal{M}}(n) = \max\left(\{1\} \cup \{m \mid \exists x \in L_{\mathcal{M}} \cap \Sigma^n : \text{The time to accept } x \text{ by } \mathcal{M} \text{ is } m.\}\right)$$

A nondeterministic Turing machine is said to be of *polynomial time* if there exists a polynomial $p(n)$ such that $T_{\mathcal{M}}(n) \leq p(n)$ for all $n \geq 1$. A

language is said to be in the set P if it can be recognized in polynomial time using a deterministic Turing machine, that is,

$$\mathrm{P} = \{L \mid \exists \text{ polynomial time deterministic TM } \mathcal{M} : L = L_{\mathcal{M}}\}.$$

A language is said to be in the set NP if it can be recognized in polynomial time using a nondeterministic Turing machine, that is,

$$\mathrm{NP} = \{L \mid \exists \text{ polynomial time nondeterministic TM } \mathcal{M} : L = L_{\mathcal{M}}\}.$$

A language L is said to be in the set co-NP if $L^{\complement} \in \mathrm{NP}$.

A Turing machine is said to *compute* a function $f : A \to B$ if for any input $x \in A$ it outputs $f(x) \in B$. However, in this work the actual output method is left undefined as noted in Section 2.1.3.

A *polynomial time transformation* from a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$ is a function $f : \Sigma_1^* \to \Sigma_2^*$ that satisfies the following conditions:

1. There exists a polynomial time deterministic Turing machine that computes f.

2. For all $x \in \Sigma_1^*$, $x \in L_1$ if, and only if, $f(x) \in L_2$.

Two languages L_1 and L_2 are *polynomially equivalent* if there exist polynomial time transformations from L_1 to L_2 and from L_2 to L_1.

A language L is said to be NP-*complete* if the following conditions hold:

1. $L \in \mathrm{NP}$.

2. For all $L' \in \mathrm{NP}$, there exists a polynomial time transformation from L' to L.

An example of an NP-complete problem is the *satisfiability problem* (or *SAT* in short). Let $X = \{x_1, \ldots, x_m\}$ be a set of Boolean *variables*. A *truth assignment* for X is a function $t : X \to \{0, 1\}$. If $t(x) = 1$, it is said that x is "true" under t. If $t(x) = 0$, it is said that x is "false" under t. If $x \in X$, then x and $\neg x$ are *literals* over X.

A *Boolean expression* or *Boolean formula* is a finite expression describing a *Boolean function* (i.e. a function with a domain of the form $\{0, 1\}^m$ and a range of the form $\{0, 1\}$) using parentheses, a set of Boolean variables and logical connectives $\neg$ (NOT), $\wedge$ (AND), $\vee$ (OR), $\Leftarrow$ and $\Rightarrow$ (implication) and $\Leftrightarrow$ (if and only if). For example,

$$f(x_1, x_2, x_3) = (x_1 \wedge x_2) \Rightarrow \neg x_3$$

is a Boolean expression.

A Boolean expression $f(x_1, \ldots, x_m)$ is *satisfied* if $f(t(x_1), \ldots, t(x_m)) = 1$ for some truth assignment t.

A *clause* over X is a Boolean expression of the form $f(x_1, \ldots, x_m) = \bigvee_{j \in J} a_j$ where a_j are literals over X. The following decision problem is the satisfiability problem: "Given a set X of Boolean variables and a set $C = \{C_i \mid i \in I\}$ of clauses over X, does there exist a truth assignment such that the Boolean expression

$$f(x_1, \ldots, x_m) = \bigwedge_{i \in I} C_i = \bigwedge_{i \in I} \bigvee_{j \in J_i} a_{i,j}$$

is satisfied?"

Theorem 2.1.7 (Cook's Theorem [20, 21, 35]). *The satisfiability problem is* NP*-complete.*

It is a famous open problem whether P equals NP or not [21, 35, 88].

2.2 Symbolic dynamics and cellular automata

This section contains the basic definitions on dynamical systems, cellular automata and their dynamical properties.

2.2.1 Basic definitions

Definition 2.2.1. *Let X be a set. A family $\mathcal{T}$ of subsets of X is called a* topology *if the following conditions hold:*

1. $\emptyset \in \mathcal{T}$ *and* $X \in \mathcal{T}$*,*
2. *any union of elements of $\mathcal{T}$ is in $\mathcal{T}$,*
3. *any intersection of finitely many elements of $\mathcal{T}$ is in $\mathcal{T}$.*

The pair $(X, \mathcal{T})$ is called a topological space*. The elements of a topology are called* open *sets and their complements are called* closed *sets. A set is called* clopen *if it is both open and closed.*

A family of open sets $U = \{U_i \mid i \in I\} \subseteq \mathcal{T}$ is called an *open cover* if $X = \bigcup_{i \in I} U_i$. A set $V = \{U_i \mid i \in J \subseteq I\}$ is said to be a *finite subcover* of the open cover U if $X = \bigcup_{i \in J} U_i$ and J is finite. A topological space $(X, \mathcal{T})$ is *compact* if every open cover has a finite subcover. Let $x \in X$ and $Y \subset X$ be an open set. If $x \in Y$, then Y is called a *neighborhood* (or an *open neighborhood*) of x.

Definition 2.2.2. *Let X be a set. A function $d : X \times X \to \mathbb{R}$ is called a* metric *or* distance*, if the following conditions are satisfied:*

1. $d(x,y) = d(y,x)$ *for all* $x, y, \in X$,

2. $d(x,y) = 0$, *if, and only if,* $x = y$ *and*

3. $d(x,y) + d(y,z) \geq d(x,z)$ *for all* $x, y, z \in X$.

If the function d *is a metric, then* (X,d) *is called a* metric space.

Example: discrete topology A simple example of a topological space is $(X, 2^X)$ where X is any set and 2^X is the power set of X (i.e. the set which contains all the subsets of X). This particular topology 2^X is called a *discrete topology* and the space $(X, 2^X)$ is called a *discrete space*.

Example: Euclidean metric The set of real vectors $\mathbb{R}^n$ is a metric space with the *Euclidean metric*

$$d(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2},$$

where $\overrightarrow{x} = (x_1, \ldots, x_n)$ and $\overrightarrow{y} = (y_1, \ldots, y_n)$.

Let (X,d) be a metric space. The set $B_\varepsilon(x) = \{y \in X \mid d(x,y) < \varepsilon\}$ is called the *ε-ball* of point x or the *ε-neighborhood* of point x or simply a *ball* with radius ε centered at point x. A subset $Y \subseteq X$ of a metric space (X,d) is called *open* if for any point $y \in Y$ $B_\varepsilon(y) \subseteq Y$ for some $\varepsilon \in \mathbb{R}$. A topological space $(X, \mathcal{T})$ is called *metrizable* if there exists a metric d on X such that the topology (i.e. the family of open sets) defined by d is equal to the original topology $\mathcal{T}$.

Given two topological spaces $(X, \mathcal{T})$ and $(Y, \mathcal{S})$, a function $f : X \to Y$ is said to be *continuous* if for each open set $U \subset Y$ the set

$$f^{-1}(U) = \{x \in X \mid f(x) \in U\}$$

is open in X. Alternatively, f is continuous if $f^{-1}(V)$ is closed for each closed set $V \subseteq Y$. A continuous bijective function is called a *homeomorphism* if its inverse is continuous.

Definition 2.2.3. *A pair* (X,F) *is a* dynamical system *if* X *is a compact metric space and* $F : X \to X$ *is a continuous function.*

A dynamical system (X,F) is said to be *periodic* if there exists a positive integer p such that $F^p(x) = x$ for every $x \in X$. The dynamical system is said to be *ultimately periodic* if there exists positive integers p_0 and p such that $F^{p_0+p}(x) = F^{p_0}(x)$ for every $x \in X$. If F is a homeomorphism (i.e. continuous bijective function with continuous inverse), the dynamical system is said to be *invertible* (or *reversible*).

Definition 2.2.4. *Two dynamical systems* (X, F) *and* (Y, G) *are* conjugate *if there exists a bijective continuous function* $\pi : X \to Y$ *satisfying* $\pi \circ F = G \circ \pi$. *Then the function* π *is called a* conjugacy.

Notation 2.2.5. *Recall that, for any two sets* X *and* Y, *the set of functions from* X *to* Y *is denoted by* Y^X, *that is,*

$$Y^X = \{f \mid f : X \to Y\}.$$

Definition 2.2.6. *Let* $\mathbb{I}$ *denote either* $\mathbb{N}$ *or* $\mathbb{Z}$. *Let* A *be an alphabet and let* $F \subseteq A^*$. *The* shift function $\sigma : A^{\mathbb{I}} \to A^{\mathbb{I}}$ *is defined by* $\sigma(x)(i) = x(i+1)$. *If a set* $X \subseteq A^{\mathbb{I}}$ *is of the form*

$$X = \{x \in A^{\mathbb{I}} \mid \forall w \in F : \forall i \in \mathbb{I} : x[i, i + |w| - 1] \neq w\},$$

then the set X *is called a* shift space *(or* shift*).*

If $\mathbb{I} = \mathbb{N}$, then the shift is said to be *one-sided*. If $\mathbb{I} = \mathbb{Z}$, then the shift is said to be *two-sided*.

A shift is called a *subshift* if it is a subset of another shift. The sets $A^{\mathbb{N}}$ and $A^{\mathbb{Z}}$ are called *full shifts*. A pair (X, σ) is called *shift dynamical system* if X is a shift.

A shift X is said to be a *subshift of finite type* (SFT) or a *subshift of order* n if there exists an integer $n \in \mathbb{N}$ and a set $F \subseteq A^n$ such that

$$X = \left\{x \in A^{\mathbb{I}} \mid \forall w \in F \subseteq A^n : \forall i \in \mathbb{I} : x[i, i + n - 1] \neq w\right\}.$$

Elements of subshifts of finite type are described by a set of forbidden words F of bounded length n. A sequence belongs to the subshift of finite type if, and only if, it does not contains a factor word from the set of forbidden words.

Definition 2.2.7. *A pair* (A^M, F) *is said to be a* cellular automaton *(or* CA *in short) if the function* F *is defined according to a condition*

$$F(c)(\overrightarrow{x}) = f(c(\overrightarrow{x} + \overrightarrow{x_1}), \ldots, c(\overrightarrow{x} + \overrightarrow{x_n})), \tag{2.1}$$

for some function $f : A^n \to A$ *and where* $+$ *denotes the operation of the monoid* M.

The set A is called the *state set* of the cellular automaton (A^M, F). An element of A^M is called a *configuration*. The function F is called the *global rule* (or *global function*) and the function f is called the *local rule* (or *local function*) of the cellular automaton. The vector of elements $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) \in M^n$ is called the *neighborhood vector* (or simply *neighborhood*) of the cellular automaton.

Remark 2.2.8. *A more classical definition of a cellular automaton is that a cellular automaton is the dynamical system defined by a quadruple* $\mathcal{A} = (M, A, N, f)$*, where* M *is the monoid defining the structure of the lattice,* A *is the state set,* N *is the neighborhood and* $f : A^{|N|} \to A$ *is the local rule. This information describes the dynamical system of Definition 2.2.7 effectively (assuming that* M *has a finite presentation) and can be processed algorithmically if needed.*

This definition of neighborhood has nothing to do with the topological definition of an (open) neighborhood. The elements of M are called *cells* and the elements $\overrightarrow{x} + \overrightarrow{x_1}, \ldots, \overrightarrow{x} + \overrightarrow{x_n}$ are called *neighbors* of cell $\overrightarrow{x}$. The vector $(\overrightarrow{x}+\overrightarrow{x_1}, \ldots, \overrightarrow{x}+\overrightarrow{x_n})$ is called the neighborhood of cell $\overrightarrow{x}$. The integer $r = \max_{i=1}^{n} \|\overrightarrow{x_i}\|$ is called the *radius* of the neighborhood, the local rule or the cellular automaton.

If $M = \mathbb{N}^d$ or $M = \mathbb{Z}^d$ in the Definition 2.2.7, then the cellular automaton is said to be *d-dimensional*. The positive integer d is then known as the *dimension* of the cellular automaton. Cellular automata of the form $(A^{\mathbb{N}}, F)$ and $(A^{\mathbb{Z}}, F)$ are called *one-dimensional*. The topic of this work is cellular automata of the form $(A^{\mathbb{Z}}, F)$. For one-dimensional cellular automata the set of configurations is in fact a full shift. One-dimensional cellular automata global functions are then functions on full shifts.

A one-dimensional cellular automaton is *one-sided* if its shift space of configurations is one-sided. Likewise, a one-dimensional cellular automaton is *two-sided* if its shift space of configurations is two-sided.

For one-dimensional cellular automata, the integer $m = -\min_{i=1}^{n} x_i$ is called the *memory* and the integer $a = \max_{i=1}^{n} x_i$ is called the *anticipation* of the local rule. Clearly, the radius can be defined by $r = \max\{m, a\}$.

The distance between two configurations $c, e \in \mathbb{Z}^d$ can is defined to be

$$d(c, e) = \left(\frac{1}{2}\right)^{\min\{\|\overrightarrow{x}\| \mid c(\overrightarrow{x}) \neq e(\overrightarrow{x})\}},$$

where

$$\|(x_1, x_2, \ldots, x_d)\| = \max\{|x_1|, |x_2|, \ldots, |x_d|\}$$

and $d(c, e) = 0$, if $c = e$.

Function $d(\cdot, \cdot)$ is also a metric thus making the set of configurations a metric space. The balls in the metric are called *cylinders* and they form a basis for the topology. Radius r cylinder containing configuration c is the set

$$\mathrm{Cyl}(c, r) = \left\{ e \in A^{\mathbb{Z}^d} \mid c(\overrightarrow{x}) = e(\overrightarrow{x}) \text{ when } \|\overrightarrow{x}\| \leq -\log_2 r \right\}$$

For every radius r there are only finitely many cylinders and these cylinders are by definition disjoint. Therefore, radius r cylinders form a partition of the space of configurations. Hence, every cylinder is clopen (i.e. both open

and closed) because the complement of every cylinder is a union of other cylinders with the same radius.

In the one-dimensional case, one can define cylinders differently as sets

$$\mathrm{Cyl}(w,k) = \left\{ c \in A^{\mathbb{Z}} \mid c(i+k) = w(i) \text{ when } i \leq |w| - 1 \right\}$$

where $w \in A^*$ is a finite word and $w(i-1)$ denotes the ith letter of the word. The word consisting of states in locations i through j (when $i \leq j$) in a configuration c is denoted by $c[i,j]$.

The topology induced by the cylinders in known as the *Cantor topology* on $A^{\mathbb{Z}^d}$ and it is compact. In $A^{\mathbb{Z}^d}$ every (infinite) sequence has a converging subsequence. Because a metric space is compact if, and only if, every sequence has a converging subsequence, the metric space $A^{\mathbb{Z}^d}$ is compact [52].

Recall that a *translation* is a mapping such that

$$\tau_{\overrightarrow{y}}(c)(\overrightarrow{x}) = c(\overrightarrow{x} + \overrightarrow{y}),$$

where $c \in A^{\mathbb{Z}^d}$ and $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{Z}^d$.

Theorem 2.2.9 ([38]). *A function $F : A^{\mathbb{Z}^d} \to A^{\mathbb{Z}^d}$ is the global function of a cellular automaton if, and only if,*

1. *it is continuous and*
2. *it commutes with translations.*

Cellular automata are indeed dynamical systems, because the set of configurations is a compact metric space and global functions of cellular automata are continuous. In particular, d-dimensional cellular automata are dynamical systems of the form $(A^{\mathbb{Z}^d}, F)$.

A cellular automaton is said to be *injective* if its global function is injective (also known as one-to-one). A cellular automaton is said to be *surjective* if its global function is surjective (also known as onto). A cellular automaton is said to be *bijective* if its global function is both injective and surjective.

A cellular automaton (A^M, F) is said to be *reversible* (or *invertible* or *RCA* in short) if there exists a cellular automaton (A^M, G) such that $F \circ G = G \circ F = I$, where $I : A^M \to A^M$ is the identity function. Then the cellular automaton (A^M, G) is the *inverse* of the cellular automaton (A^M, F).

Theorem 2.2.10 ([38]). *A cellular automaton (A^M, F), where $M = \mathbb{Z}^d$ or $M = \mathbb{N}^d$, is reversible if, and only if, it is bijective.*

This theorem means that if the inverse function F^{-1} exists, it is the global rule of another cellular automaton. Furthermore, it is known that for cellular automata injectivity and bijectivity are equivalent and injectivity implies surjectivity [80, 84]. Injectivity (i.e. reversibility) and surjectivity are known to be decidable for one-dimensional cellular automata [3, 101] but undecidable for two-dimensional cellular automata [47].

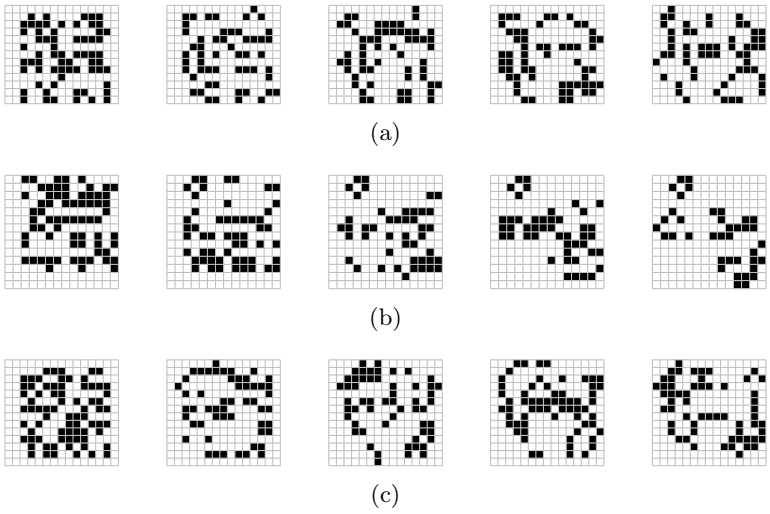


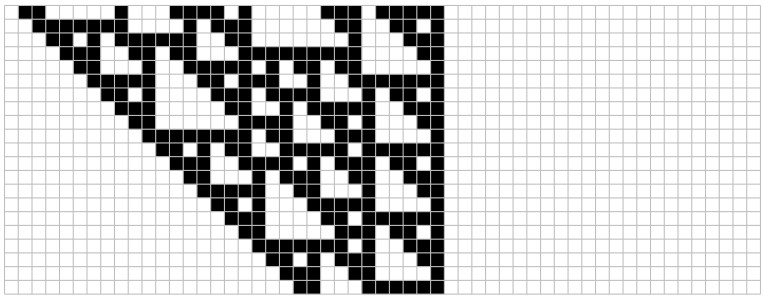
Figure 2.1: Examples of Game of Life computation on random finite initial configurations.

Example: John Conway's Game of Life Probably the most famous cellular automaton is the Game of Life by John Conway [34, 8]. The *Game of Life* (or simply *Life*) is a two-dimensional cellular automaton where each cell has a neighborhood of 3×3 cells centered at the cell and each cell is either *dead* or *alive*. A dead cell becomes alive if it has exactly three living neighbors. A living cell remains alive if and only if there are either two or three living cells in the neighboring cells surrounding it. In all other cases the cell will be dead in the next configuration. The motivation of such a local rule is to simulate artificial life. If a living cell has too many living neighbors, it dies of starvation. If it has too few neighbors, it dies of isolation. If there is the right number of living neighbors, a new living cell is born in place of a dead cell.

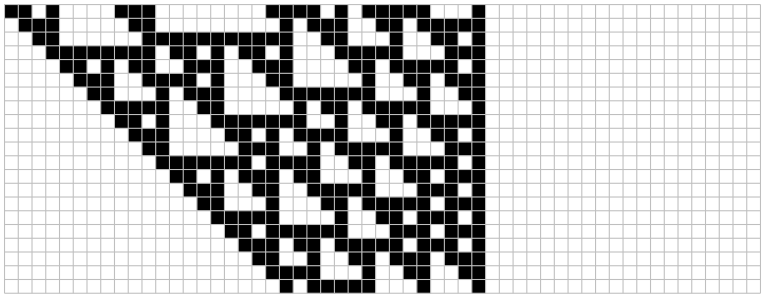
The Game of Life has been studied extensively. One notable property of Life is that seemingly "living" clusters of cells can appear from random initial configurations. From computational perspective, it has been shown that the Game of Life is computationally universal and it is undecidable whether a finite configuration dies or not [8, 51].

Figure 2.1 contains three examples of Game of Life computations (showing five consecutive configurations) starting with a random finite initial configuration.

Example: Rule 110 The *Rule 110* cellular automaton is a one-dimensional cellular automaton with a state set $\{0, 1\}$ and a local rule $f : \{0, 1\}^3 \rightarrow$



(a)



(b)

Figure 2.2: Examples of Rule 110 computation on random finite initial configurations. The computation advances upwards.

$\{0, 1\}$, which is defined by

$$f(1,1,1) = 0, \quad f(1,1,0) = 1, \quad f(1,0,1) = 1, \quad f(1,0,0) = 0,$$
$$f(0,1,1) = 1, \quad f(0,1,0) = 1, \quad f(0,0,1) = 1 \text{ and } \quad f(0,0,0) = 0.$$

The name of Rule 110 comes from the fact that the sequence of local rule's image states forms the binary presentation of number 110, that is,

$$110 = (01101110)_2.$$

The number, which is given by the sequence of image states of a one-dimensional cellular automaton's local rule, is called the *Wolfram number* of the cellular automaton. The Rule 110 is known to be computationally universal [19, 110].

Figure 2.2 contains two examples of Rule 110 computations starting with a random finite initial configuration.

2.2.2 Equicontinuity and sensitivity

Definition 2.2.11. *A point $x \in X$ is an* equicontinuity point *of function F if for any $\varepsilon > 0$ there exists $\delta > 0$ such that for any point $y \in X$ and*

integer $n \in \mathbb{N}$,

$$d(x, y) < \delta \Longrightarrow d(F^n(x), F^n(y)) < \varepsilon.$$

A dynamical system (X, F) *is* equicontinuous *if every point* $x \in X$ *is an equicontinuity point.*

Definition 2.2.12. *A dynamical system* (X, F) *is* sensitive to initial conditions *(or* sensitive*) if there exists such* $\varepsilon > 0$ *that for any* $x \in X$ *and* $\delta > 0$ *there exists a point* $y \in X$ *such that*

$$0 < d(x, y) < \delta \text{ and } d(F^n(x), F^n(y)) \geq \varepsilon$$

for some integer $n \in \mathbb{N}$*. If the constant* ε *exists, it is known as the* sensitivity constant*.*

It was shown in [28] that equicontinuity and sensitivity are undecidable properties for not necessarily reversible one-dimensional cellular automata. Recently, it was shown that equicontinuity is undecidable even for reversible one-dimensional cellular automata [54]. The proof for the reversible case was based on the undecidability of the immortality problem for reversible Turing machines.

Theorem 2.2.13 ([10]). *A cellular automaton is equicontinuous if, and only if, it is ultimately periodic.*

A state $q \in A$ is called *quiescent* if the local rule of the cellular automaton satisfies $f(q, \ldots, q) = q$. A cellular automaton is said to be *nilpotent* if it has a quiescent state q and there exists a bound n_0 such that

$$F^n(c)(i) = q$$

for every $n \geq n_0$, $i \in \mathbb{Z}$ and $c \in A^{\mathbb{Z}^d}$. A configuration c is called *quiescent* if $F(c) = c$.

Nilpotent cellular automata are all ultimately periodic and therefore also equicontinuous. It is undecidable whether or not a given one-dimensional cellular automaton is nilpotent [46, 48]. A closely related result is the uncomputability of topological entropy for not necessarily reversible one-dimensional cellular automata [42]. It is also undecidable whether an not necessarily reversible one-dimensional cellular automaton has a non-trivial conservation law or not [32].

Definition 2.2.14. *A word* $w \in A^*$ *is* blocking *if there exists a sequence of words* $(w_n)_{n=0}^{\infty}$ *such that* $w_n \in A^r$ *and there exists an integer* i *such that* $F^n(\mathrm{Cyl}(w, i)) \subseteq \mathrm{Cyl}(w_n, 0)$ *for any* $n \in \mathbb{N}$*.*

Theorem 2.2.15 ([10]). *Any equicontinuity point has an occurrence of a blocking word. Conversely, any point with infinitely many occurrences of blocking words arbitrarily far to the left and right of the origin is an equicontinuity point.*

For one-dimensional cellular automata sensitivity is equivalent to the nonexistence of equicontinuity points. However, for two-dimensional cellular automata this claim does not hold. There exists a 2-dimensional cellular automaton which is non-sensitive but it still has no equicontinuity points [95].

2.2.3 Topological transitivity and topological mixing

Definition 2.2.16. *A dynamical system (X, F) is* topologically transitive *(or* transitive*) if for all nonempty open subsets U and V of X there exists a positive integer n such that $F^n(U) \cap V \neq \emptyset$.*

Definition 2.2.17. *A dynamical system (X, F) is* topologically mixing *(or* mixing*) if for all nonempty open subsets U and V of X there exists a positive integer n_0 such that $F^n(U) \cap V \neq \emptyset$ for all $n \geq n_0$.*

Topological mixing is an even stronger property than transitivity. Clearly, a mixing dynamical system is transitive also.

It can be proved with a topological argumentation that a cellular automaton is topologically transitive if, and only if, it has a dense orbit (Definition 2.2.21) [52]. A short introduction to transitivity in terms of symbolic dynamics can be found in [65]. It is known that transitivity implies sensitivity [17].

2.2.4 Expansivity

Definition 2.2.18. *A reversible dynamical system (X, F) is* expansive *if there exists a constant ε such that for any two different points x and y*

$$d(F^n(x), F^n(y)) \geq \varepsilon \tag{2.2}$$

for some integer $n \in \mathbb{Z}$.

Definition 2.2.19. *A dynamical system (X, F) is* positively expansive *if there exists a constant ε such that for any two different points x and y equation* (2.2) *holds for some positive integer $n \in \mathbb{N}$.*

It is known that only one-dimensional cellular automata can be expansive or positively expansive [31, 98]. It can be shown that a reversible cellular automaton cannot be positively expansive [52]. Furthermore, positive expansivity implies topological mixing [9].

It is also known that any positively expansive cellular automaton is conjugate to a one-sided subshift of finite type [85]. More precisely, the subshift is the column subshift $\Sigma_{2r+1}(F)$ (see Definition 2.2.24), where r is the radius of the local rule. An expansive cellular automata is conjugate to a subshift of finite type if either memory or anticipation is non-positive [86]. It is an open problem whether or not an expansive cellular automaton is conjugate to a two-sided subshift of finite type. In general, if the memory of $(A^{\mathbb{Z}}, F)$ is m and the anticipation of $(A^{\mathbb{Z}}, F^{-1})$ is a, then $(A^{\mathbb{Z}}, F \circ \sigma^k)$ is conjugate to a subshift of finite type for all $k \geq \max\{m, a\}$ [11, 86].

Example: the shift function A simple example of a cellular automaton global function, which is reversible, expansive and transitive, is the *shift function* $\sigma : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$, which is defined by

$$\sigma(c)(i) = c(i+1).$$

A reversible cellular automaton $(A^{\mathbb{Z}}, F)$ is *left expansive* if there exists a constant ε such that for any two different configurations c and e with $c(i) \neq e(i)$ for some $i < 0$ equation (2.2) holds for some integer $n \in \mathbb{Z}$. Similarly, a reversible cellular automaton is *right expansive* if there exists a constant ε such that for any two different configurations c and e with $c(i) \neq e(i)$ for some $i > 0$ equation (2.2) holds for some integer $n \in \mathbb{Z}$.

A cellular automaton $(A^{\mathbb{Z}}, F)$ is *positively left expansive* if there exists a constant ε such that for any two different configurations c and e with $c(i) \neq e(i)$ for some $i < 0$ equation (2.2) holds for some positive integer $n \in \mathbb{N}$. Similarly, the cellular automaton is *positively right expansive* if there exists a constant ε such that for any two different configurations c and e with $c(i) \neq e(i)$ for some $i > 0$ equation (2.2) holds for some positive integer $n \in \mathbb{N}$.

The original definition of left and right expansivity given by Kurka actually meant positive left and right expansivity [59]. Here Kurka's left expansivity is called positive left expansivity and right expansivity is called positive right expansivity because of the similar difference between expansivity of reversible cellular automata and positive expansivity of the irreversible cellular automata. Intuitively positive left expansivity means that if two configurations differ by some cell then their later images will differ by cells further to the right. Similarly positive right expansivity means that if two configurations differ by some cell then their later images will differ by cells further to the left.

2.2.5 Chaos

Definition 2.2.20. *Let $P(F)$ denote the set of periodic points of a dynamical system (X, F), that is,*

$$P(F) = \{x \in X \mid \exists n \in \mathbb{N} \setminus \{0\} : F^n(x) = x\}.$$

A dynamical system (X, F) is said to have dense periodic points *if $P(F)$ is a dense subset of X, or equivalently, for any point $x \in X$ and any positive real number $\varepsilon > 0$ there exists a periodic point $y \in P(F)$ such that $d(x, y) < \varepsilon$.*

Definition 2.2.21. *A dynamical system (X, F) has a* dense orbit *if there exists a point $x \in X$ such that for every point $y \in X$ with any $\varepsilon > 0$ there exists an integer $n \in \mathbb{N}$ such that $d(F^n(x), y) < \varepsilon$.*

Definition 2.2.22 (Devaney's chaos [26]). *A dynamical system (X, F) is said to be* chaotic *according to Devaney's definition if*

1. *it is topologically transitive,*
2. *it has dense periodic points and*
3. *it is sensitive to initial conditions.*

For cellular automata, it is an open problem whether the set of periodic points is dense or not for every surjective cellular automaton [51]. If the answer is affirmative, chaos would become equivalent to transitivity, because transitivity implies sensitivity (for cellular automata) [17, 51].

Definition 2.2.23 (Knudsen's chaos [57]). *A dynamical system (X, F) is said to be* chaotic *according to Knudsen's definition if*

1. *it has a dense orbit and*
2. *it is sensitive to initial conditions.*

In [15] the authors reviewed some of the properties of discrete time dynamical systems in terms of cellular automata. The authors discussed also Knudsen's definition of chaotic behavior with respect to cellular automata.

In the case of reversible cellular automata, Devaney's and Knudsen's definitions of chaos are equivalent because a reversible cellular automaton has always dense periodic points and a cellular automaton is transitive if, and only if, it has a dense orbit. Since transitivity implies sensitivity, both definitions of chaos are equivalent to transitivity in the reversible case.

Definition 2.2.24. *A set*

$$\Sigma_k(F) = \left\{ x \in \left(A^k\right)^{\mathbb{I}} \mid \exists y \in A^{\mathbb{J}} : \forall i \in \mathbb{I} : F^i(y)[0, k-1] = x(i) \right\}$$

is called the column subshift *of the one-dimensional cellular automaton $(A^{\mathbb{J}}, F)$, where $\mathbb{I}$ and $\mathbb{J}$ equal either $\mathbb{N}$ or $\mathbb{Z}$.*

A cellular automaton is *regular* if for any positive integer k the finite words (i.e. words over alphabet A^k) appearing in the sequences of $\Sigma_k(F)$ as factors form together a regular language. Equivalently, cellular automaton is regular if $\Sigma_k(F)$ is *sofic* for every k. In particular, if the column subshift is of finite type, then the cellular automaton is regular. Every subshift of finite type is sofic, but not every sofic subshift is a subshift of finite type [65, p. 67]. Sometimes the denseness of periodic points is called regularity or topological regularity [15]. However, cellular automata, whose column subshifts are sofic subshifts, are called regular [58] because the factor words of the elements of a column subshift form a regular language in the sense of formal languages. In this work regularity means the regularity defined in [58] (i.e. the regularity of column subshift words). It has been shown by Di Lena that regularity is an undecidable property for one-dimensional not necessarily reversible cellular automata [63].

Using the undecidability of equicontinuity for reversible one-dimensional cellular automata [54], undecidability of regularity follows in the reversible case with the same method of proof:

Theorem 2.2.25. *It is undecidable whether or not a given one-dimensional reversible cellular automaton is regular.*

Proof. Assume that regularity is a decidable property among reversible one-dimensional cellular automata. A reversible cellular automaton is equicontinuous if, and only if, it is periodic. Every periodic cellular automaton is regular because a language of a column subshift consists of powers of a finite number of words.

Reversibility is a decidable property for one-dimensional cellular automata and equicontinuity is a decidable property among regular one-dimensional cellular automata because the graph presentation of the column subshift can be constructed algorithmically [63]. Therefore, by the assumption, it is possible to determine whether or not a given regular one-dimensional cellular automaton is periodic (i.e. both reversible and equicontinuous). This contradicts the undecidability of periodicity. □

2.2.6 Applications of cellular automata

Applications of cellular automata range both simulation and computational purposes. Cellular automata have been used to simulate biological (see, for example [4, 13]) physiological (see, for example [73, 90]) and physical phenomena (see, for example [33, 37]). Even quantum computation has been discussed in terms of cellular automata (see, for example [105]). In [4], vegetation growth is discussed in terms of cellular automata. In particular, [4, p. 115] contains a partial list of the simulation uses of cellular automata. In [33], a family of cellular automata called lattice gases are discussed. A

lattice gas (or *lattice gas automaton*) is a cellular automaton which is used to simulate the behavior of moving particles on a lattice.

Cellular automata have been used also both in private-key cryptography (see, for example [72, 97, 5]) and public-key cryptography [36, 45, 102]. For example, in [72] simple two-dimensional cellular automata are used to generate pseudo-random bits. In [45], the cryptosystem's security is based on the difficulty of inverting a 2-dimensional cellular automaton. This assumption is based on the undecidability of reversibility of 2-dimensional cellular automata [47]. The public key is a global function F of a cellular automaton and it is formed as a composition

$$F = F_n \circ \cdots \circ F_1$$

of n global functions of more "simple" reversible cellular automata. The private key is the sequence of functions $(F_1, \ldots, F_n)$. It is assumed that the cellular automata with global functions F_i are chosen so that the neighborhood of F^{-1} is very large compared to that of F. It is assumed, that in practice it is infeasible to compute the inverse mapping

$$F^{-1} = F_1^{-1} \circ \cdots \circ F_n^{-1}$$

without the knowledge of the functions F_i. The plaintext configuration p is mapped with F to produce the ciphertext configuration $c = F(p)$. Assuming the infeasibility of computing F^{-1} from F without the knowledge of $(F_1, \ldots, F_n)$, only the receiver can compute $p = F^{-1}(c)$. The practical implementation of functions F_i was suggested to be based on so-called *marker cellular automata*. In a marker cellular automata, the local rule is defined by a permutation and a set of patterns with the neighborhood as the domain. The state of a cell is mapped according to the permutation if, and only if, some state pattern from the pattern set is present in the neighborhood of the cell. The details of constructing suitable marker cellular automata have been recently discussed in [16].

2.3 Wang tiles

In this section the basic definitions and results on Wang tiles are presented. Wang tiles are used in this work because they can be used to prove some undecidability results for cellular automata.

2.3.1 Basic definitions

A *Wang tile* (or a *tile* in short) is a unit square with colored edges. The edges of a Wang tile are called *north*, *east*, *west* and *south* edges in a natural way. Each edge of a Wang tile has a *color* which is an element from a finite

set. For the given tile t, expressions t_N, t_E, t_W and t_S are used to denote north, east, west and south edge colors, respectively. A *Wang tile set* T (or a *tile set* in short) is a finite set of Wang tiles.

Given tile sets $T_1, \ldots, T_n$, a tile set $T \subseteq T_1 \times \ldots \times T_n$ is a *sandwich tile set*. Elements of T are called *sandwich tiles*. Tile set T_i is said to be *layer* i of the sandwich tile set. Let $t \in T$ be an element of a sandwich tile set and $t = (t_{i_1}, \ldots, t_{i_n})$. Then the colors of t are sequences of corresponding colors of the original tiles, for example, $t_N = (t_{i_1 N}, \ldots, t_{i_n N})$. Let S_i be a subset of tile set T_i for any $1 \leq i \leq n$. If $S_1 \times \ldots \times S_{i-1} \times \{t\} \times S_{i+1} \times \ldots \times S_n \in T$ it is said that the tile t on layer i is *paired* with tiles S_j on layer j.

Sandwich tiles are perhaps a more illustrative way to express that a tile set is a subset of a cartesian product of given tile sets.

Definition 2.3.1. *A* tiling *is a function $f : \mathbb{Z}^2 \to T$, which assigns a unique Wang tile for each integer pair of the plane. A tiling f is said to be* valid, *if for every pair $(x, y) \in \mathbb{Z}^2$ the tile $f(x, y) \in T$ matches its neighboring tiles (e.g. the south edge of tile $f(x, y)$ has the same color as the north edge of tile $f(x, y - 1)$).*

A tiling $f : \mathbb{Z}^2 \to T$ is called *periodic* with period $(a, b) \neq (0, 0)$ if $f(x, y) = f(x + a, y + b)$ for all $(x, y) \in \mathbb{Z}^2$. Otherwise the tiling f is called *non-periodic*. A tile set T is called *aperiodic*, if there exists some tiling with the tile set T, but no tiling with the tile set T is periodic. If the tile set T admits a periodic tiling $f : \mathbb{Z}^2 \to T$ with some period, then it admits also a *doubly periodic* tiling $g : \mathbb{Z}^2 \to T$, that is, there exists such non-zero integers a and b that $g(x, y) = g(x + a, y)$ and $g(x, y) = g(x, y + b)$ for all $(x, y) \in \mathbb{Z}^2$ [92].

A Wang tile set T is said to be *NW-deterministic*, if within the tile set there does not exist two different tiles with the same colors on the north and west edges. In general, a Wang tile set is *XY-deterministic*, if the colors of X- and Y-edges uniquely determine a tile in the given Wang tile set. A Wang tile set is *4-way deterministic*, if it is NE-, NW-, SE- and SW-deterministic. This definition makes a Wang tile set "deterministic" in all four diagonal directions and the rest of a valid tiling is always determined by a single infinite diagonal row of tiles to any diagonal direction. If one is given an infinite diagonal row of tiles which is already known to be part of a valid tiling, then the rest of the tiling is uniquely determined.

A Wang tile set is called *2-way deterministic*, if it is NE- and SW-deterministic. A 4-way deterministic tile set is 2-way deterministic.

Let T be a Wang tile set. The $n \times n$ *tile set* is the Wang tile set that has been constructed by taking every $n \times n$ cluster of matching tiles and mapping them to a single Wang tile as shown in Figure 2.3. The colors in the vertical direction are all the matching $n \times (n-1)$ clusters of tiles and the colors in the horizontal direction are all the matching $(n-1) \times n$ clusters of

tiles. The tile set which is the $n \times n$ tile set of the tile set T is denoted by $T^{n \times n}$. Clearly, if the original tile set T is 4-way deterministic so is the tile set $T^{n \times n}$.

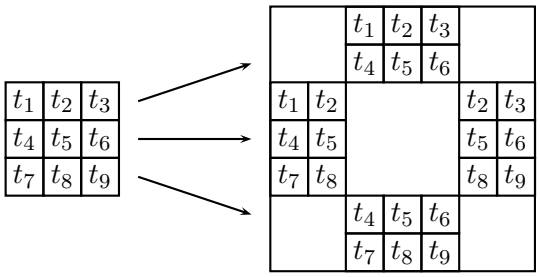


Figure 2.3: An $n \times n$ square (or rectangular) cluster of matching Wang tiles can be converted into a single Wang tile. This can be done by setting the $n-1$ outermost columns and rows of tiles as colors of the new tile.

Definition 2.3.2. *A function $f : T_1 \to T_2$ is called a* tile homomorphism *if it respects the colors, i.e. $f(t) = t'$ with ${t'}_N = g(t_N)$, ${t'}_E = g(t_E)$, ${t'}_W = g(t_W)$ and ${t'}_S = g(t_S)$, where g is a function from the set of the colors of the tile set T_1 to the set of the colors of the tile set T_2.*

The *homomorphic image* $f(T)$ of a tile set T is defined in the natural way as the set

$$f(T) = \{f(t) | t \in T\}.$$

If every tile in a given tile set T appears in a valid tiling, then the $n \times n$ tile set $T^{n \times n}$ can always be mapped tile homomorphically onto T, that is, there always exists a tile homomorphism f such that $f(T^{n \times n}) = T$. For example, f could map an element $t \in T^{3 \times 3}$ to the tile t_5 (denoted as in Figure 2.3) in the center of the 3×3 tile cluster represented by t. Furthermore, the homomorphic image of a valid tiling is also a valid tiling.

The following decision problem is referred to as the *tiling problem* of Wang tiles: "Given a Wang tile set T, does there exist a valid tiling of the plane using tiles of T?" A tiling $f : \mathbb{Z}^2 \to T$ is said to *contain* tile $t \in T$, if for some integers $x, y \in \mathbb{Z}$ equation $f(x, y) = t$ holds. The tiling problem of wang tiles is also known as the *domino problem*. The following decision problem is referred to as the *tiling problem with a seed tile*: "Given a Wang tile set T and a tile $t \in T$, does there exist a valid tiling of the plane that contains the tile t?"

If the tiling problem with a seed tile were decidable, then the tiling problem would be decidable. Let T be the tile set of the given instance of the tiling problem. Then the answer for the tiling problem is affirmative if,

and only if, for some tile $t \in T$ the answer for the tiling problem with a seed tile is affirmative considering the tile set T as the tile set of the instance and the tile t as the seed tile of the instance.

It is already known that the tiling problem is undecidable [92, 7]. Furthermore, it is known to be undecidable even when restricted to tile sets that are deterministic by one corner [46]. In this work it is shown that the tiling problem is undecidable for tile sets that are deterministic by all four corners. The proof relies on the 4-way deterministic aperiodic tile set given by Kari and Papasoglu [55].

2.3.2 Determinism in relation to cellular automata

Wang tiles have been used previously, for example, to prove undecidability of injectivity and surjectivity of two-dimensional cellular automata [47] and nilpotency of one-dimensional cellular automata [46]. The definition of determinism for tile sets was originally motivated by the theory of one-dimensional cellular automata [46, 51].

A tile set can be considered as a one-dimensional cellular automaton, if the tile set contains all the possible color pairs at one corner and the tile set is deterministic by the same corner. The tiles can be seen as states of the cells and the diagonal rows of tiles as configurations of the cellular automaton. Therefore, the rule, which determines whether or not neighboring tiles match, can in this case be considered as the local rule of a cellular automaton.

It has been shown that the tiling problem is undecidable with tile sets that are deterministic by at least one corner [46]. From this it follows that nilpotency of one-dimensional cellular automata is undecidable [46, 48]. Undecidability of nilpotency follows from the fact that a tiling error can be used to create a spreading quiescent state in the local rule and then every diagonal row of tiles leads to a quiescent configuration if, and only if, there exists no valid tiling.

If a Wang tile set is deterministic by two opposite corners, then the tile set can be seen as a subset of the state set of a reversible one-dimensional cellular automaton. For such a tile set the non-existence of a valid tiling is equivalent to having "forbidden" states to appear in every computation of the cellular automaton.

2.3.3 Mathematical self-assembly

Mathematical self-assembly is the concept of modelling chemical self-assembly with Wang tiles. One model for self-assembly is the *tile assembly model* [108], where one is given a tile set T, a *seed tile* $s \in T$, a *temperature* $\tau \in \mathbb{N}$ and a *glue function* $g : C \times C \to \mathbb{Z}$, where C is the set of colors of T and

$g(a, b) = g(b, a)$. The tiles represent molecules and the glue function is used to represent the bond strength between the edges of different molecules (i.e. tiles). In its simplest form, self-assembly is started from a single seed tile and tiles are added one by one to an existing connected cluster of tiles. A tile can be appended to an existing tile cluster if there is a slot around which the sum of the glue values between the tile and the earlier tiles exceeds or equals the temperature. A more thorough discussion of mathematical self-assembly can be found, for example, in [108, 1, 2].

One topic of mathematical self-assembly is the characterization of tile sets (and glue functions) that produce desired kind of tile clusters even if any finite number of tiles (except the seed tile) were removed from the tile cluster. For this reason, self-healing tile sets have been introduced in [109, 100]. The approach is to counter a sudden removal of any finite number of tiles (except the seed tile) from assembled tile clusters. A tile set is called *self-healing* if for any assembled tile cluster any damage caused by removing any finite number of tiles is eventually repaired (in linear time with respect to the number of removed tiles) so that every removed tile is restored in its original place in the tile cluster.

One possible application of 4-way deterministic tile sets is the modelling of one type of self-healing systems in mathematical self-assembly. In a way, 4-way deterministic tile sets are an analogy of self-healing tile sets in the case of classical tilings. If a 4-way deterministic tile set admits a valid tiling, then any continuous two-way infinite tile path, which intersects all rows and columns and is contained in a valid tiling, determines rest of the tiling uniquely. If any finite number of tiles is removed from a valid tiling, the missing tiles can be replaced only in a unique way to produce a valid tiling again. Any two adjacent neighbors determine a tile uniquely in a valid tiling. With regard to mathematical self-assembly, Theorem 3.4.2 gives a result which might be useful. It states that the tiling problem is undecidable even when the tile set is deterministic by any two, including opposite, edges. This follows from the undecidability of the tiling problem in the 4-way deterministic case and the fact that a 4-way deterministic tile set can be changed to a 2×2 tile set which is deterministic also with respect to colors on opposite edges of tiles.

2.3.4 Wang tiles in computer graphics

One practical application of Wang tiles is their use for texture generation in computer graphics [18, 107, 106]. Wang tiles can be used in cases when the texture image's size would exceed the texture size limit of the graphics card and the texture would consist of small square patterns which repeat in a non-periodic manner.

In practice, the Wang tile texture generation can be implemented (for both DirectX and OpenGL) by special programming languages known as *shading languages* which enable a programmer to write programs (known as *shaders*) directly for the GPU. With shading languages one can easily implement non-standard lighting models [89] and use procedural textures such as fractal patterns instead of images [94]. Examples of such programming languages are Microsoft's HLSL for DirectX, OpenGL's GLSL [94], NVIDIA's Cg [30] and Pixar's RenderMan.

Using shaders, a GPU can be sometimes used to speed up algorithms that execute massive floating-point number operations. For example, a GPU can be used to speed up the simulation of Lattice Boltzmann Method (LBM) [64] which can be seen as a cellular automaton. Also some simple cellular automata with a "discrete" state set can be simulated with a GPU. For example, the Game of Life can be simulated with standard OpenGL functions without the use of shaders [99, p. 627].

Chapter 3

On 4-way deterministic tile sets

In this chapter it is shown that the tiling problem of Wang tiles (also known as the domino problem) remains undecidable even when the instances are restricted to 4-way deterministic tile sets. This result is interesting in the sense that if a valid tiling exists, it is always completely determined by any diagonal row of tiles in the valid tiling.

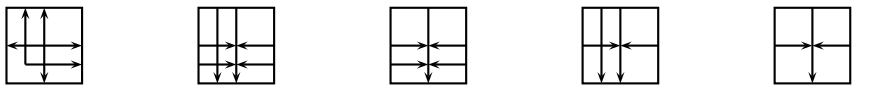
3.1 The aperiodic tile set

In this section Robinson's aperiodic tile set and an improved 4-way deterministic aperiodic tile set are discussed briefly. As before in [92], the undecidability proof of the tiling problem makes use of an aperiodic tile set. However, now the tile set needs to be 4-way deterministic and such a suitable tile set has been provided in [55].

3.1.1 The aperiodic tile set of Robinson

Earlier proofs of the undecidability of different variants of the tiling problem have relied heavily on the existence of aperiodic tile sets, that is, tile sets which admit only non-periodic valid tilings. A well-known aperiodic tile set is the tile set of Robinson which he used in proving the tiling problem to be undecidable [92]. Its 4-way deterministic version will be used in the following proofs also. A more thorough description of Robinson's tile set can be found in Appendix A.

A general outline of Robinson's tile set is shown in Figure 3.1. The tile set consists of tiles that are called *crosses* and tiles that are called *arms* as shown in the figure. The colors of the tiles are defined using patterns consisting of *single arrows* and *double arrows*. The arrows are colored either



(a) A cross. (b) An arm. (c) An arm. (d) An arm. (e) An arm.

Figure 3.1: The basic tiles of Robinson's tile set (with colors, reflections, rotations and parity constraints omitted).

red or blue. In a cross tile all the arrows have the same color. In an arm tile the horizontal arrows may have the same or a different color than the vertical arrows with the exception that the tiles of the form shown in Figure 3.1(e), where the horizontal arrows are required to have a different color than the vertical arrows. This constraint concerning the tiles of the form shown in Figure 3.1(e) is set to ensure that only squares with different color can intersect.

Two tiles are considered to match at their abutting edges if an arrow (of some particular type and color) exiting one of the tiles enters the other tile. Robinson's tile set has also some parity constraints that are not shown in the tiles in Figure 3.1. A complete description of Robinson's tile set can be found in [92] and the necessary properties of Robinson's tile set are briefly reviewed in Appendix A.

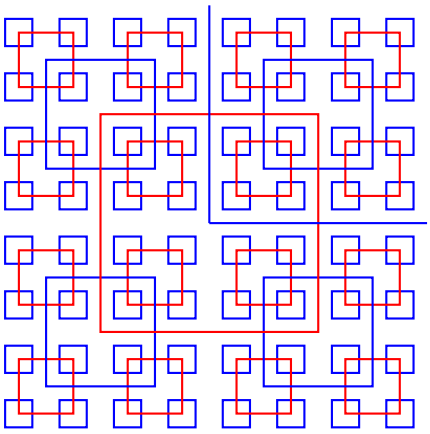


Figure 3.2: A part of the self-similar pattern generated by the tile set of Robinson (and the tile set of Kari and Papasoglu).

Robinson's tile set forces a self-similar pattern to be tiled, a part of which is shown in Figure 3.2. The tiling forced by the Robinson's tile set is divided into square areas bounded by blue squares or red squares. The edges of the

squares are formed from the double arrows found in the tiles in Figure 3.1. The crosses (i.e. the tiles of the form shown in Figure 3.1(a)) act as corners for the squares of different size and color.

More specifically, the tiling contains blue squares of height $2^{2n+1} + 1$ and red squares of height $2^{2(n+1)} + 1$, for every non-negative integer n. Furthermore, borders of squares of the same color never intersect. In the center of the area bounded by a blue square there is always some corner of a red square, and likewise in the center of the area bounded by a red square there is always some corner of a blue square. However, every blue cross is not located in the center of a red square because it is assumed that the smallest squares present in a valid tiling are the blue squares.

3.1.2 The 4-way deterministic aperiodic tile set

Kari and Papasoglu have constructed a 4-way deterministic tile set which will be used in this work instead of Robinson's tile set. Formally, the following theorem holds:

Theorem 3.1.1 ([55]). *There exists a 4-way deterministic tile set, which*

1. *admits a valid tiling and*

2. *can be mapped homomorphically onto Robinson's tile set.*

An implication of property 2 is the aperiodicity of the tile set of Kari and Papasoglu. However, the exact structure of the 4-way deterministic tile set is irrelevant. It is entirely sufficient to know that there exists a 4-way deterministic tile set which can be mapped homomorphically onto Robinson's tile set. That is, the tile set can be used in a similar way as Robinson's tile set and it is enough to refer to the 4-way deterministic aperiodic tile set as if it was the Robinson's tile set. The only difference is that each tile in the Robinson's tile set is represented by possibly more than one tile in the tile set of Kari and Papasoglu.

3.2 Drawing a diagonal line

In this section it is shown how to construct a 4-way deterministic tile set which can be used to draw a single diagonal line. One can construct a tile set D such that it is a union of disjoint tile sets D_1 and D_2 and it admits a valid tiling in which the tiles of D_1 are located only on a single two-way infinite diagonal row of tiles and no tile of D_2 is located on the same diagonal row.

In practical terms the tile set D is such that if the tiles of D_1 were colored black and the tiles of D_2 were colored white, then there would exist a valid

tiling such that it contains a single black diagonal line on white background. However, a tiling which contains the single diagonal line is not forced. For example, there might exist a valid tiling with no black diagonal lines or there might exist a valid tiling with multiple black diagonal lines or any other patterns. Fortunately, it is not necessary to force the diagonal line to be tiled. It is sufficient to know that there exists a tiling where all the black tiles are located on a single two-way infinite diagonal line.

The construction is quite tedious but in Section 3.3.5 it is essential in converting a "nondeterministic" set of tiles which represents a Turing machine into a tile set which is 4-way deterministic.

3.2.1 Structure of the tile set

The tile set D is constructed in four layers as follows:

Layer 1. The aperiodic tile set of Kari and Papasoglu.

Layer 2. The tiles in Figures 3.4, 3.5 and 3.6.

Layer 3. The tiles of layer 1 rotated by 180 degrees.

Layer 4. The tiles of layer 2 rotated by 180 degrees.

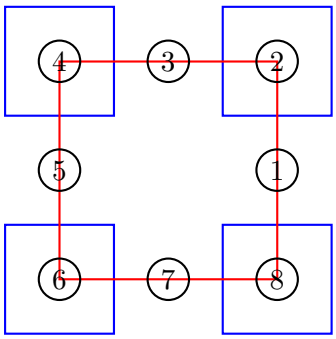


Figure 3.3: The tiles around the edges of blue and red squares in the non-periodic tiling which are used to set constraints between the aperiodic tile set and the final sandwich tile set.

The tile set D is constructed as a sandwich tile set with four layers. However, tiles of layers 1 and 2 will be paired together in a similar way as the tiles of layers 3 and 4. Therefore it is, for now, sufficient to describe how the tiles of layers 1 and 2 are paired together.

The first layer consists of the tiles of the aperiodic tile set of Kari and Papasoglu and the second layer consists of tiles from a simpler tile set. Each tile of the aperiodic tile set is paired with a specific tile set (one of the

sets shown in Figures 3.4, 3.5 and 3.6). Recall that in a valid tiling by the Robinson's tile set the squares are formed from the arrow patterns of the tiles in Figure 3.1. Then the corners and the edge centers (enumerated as in Figure 3.3) of a square can be described as follows in terms of the arrow patterns (which are used as colors in the aperiodic tile set):

1. A meeting point of two vertical double arrows at the east border of a square.

2. A cross in the northeast corner of a square.

3. A meeting point of two horizontal double arrows at the north border of a square.

4. A cross in the northwest corner of a square.

5. A meeting point of two vertical double arrows at the west border of a square.

6. A cross in the southwest corner of a square.

7. A meeting point of two horizontal double arrows at the south border of a square.

8. A cross in the southeast corner of a square.

It is possible to assume that one can make a distinction between the tiles that are located within a square of a certain fixed size and the tiles that are not. This can be done by forming a new tile set by taking sufficiently large $n \times n$ blocks (which do not contain tiling errors) of the original tiles and using $n - 1$ outermost tile rows and columns of tiles of the blocks as colors of the new tiles as in Figure 2.3. This construction maintains 4-way determinism. Therefore it is possible to assume that one can determine from the edge colors whether or not a given tile from the aperiodic tile set is located inside a blue 3×3 square in a valid tiling. Now the tiles on layer 1 and layer 2 are paired together according to the following rules:

1. Tiles at locations 1 on layer 1 are paired with the tiles in Figure 3.4(a) on layer 2.

2. The tiles at locations 2 on layer 1 are paired with the tiles in Figure 3.5(a) on layer 2.

3. The tiles at locations 3 on layer 1 are paired with the tiles in Figure 3.4(b) on layer 2.

4. The tiles at locations 4 on layer 1 which are located outside a 3×3 blue square are paired with the tiles in Figure 3.5(b) on layer 2. The tiles at locations 4 on layer 1 which are located within a 3×3 blue square are paired with the tiles in Figure 3.6(a) on layer 2.

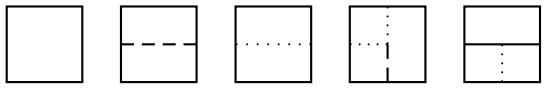
5. The tiles at locations 5 on layer 1 are paired with the tiles in Figure 3.4(c) on layer 2.

6. The tiles at locations 6 on layer 1 are paired with the tiles in Figure 3.5(c) on layer 2.

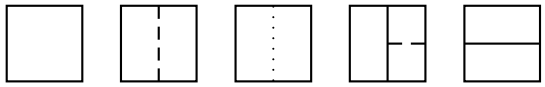
7. The tiles at locations 7 on layer 1 are paired with the tiles in Figure 3.4(d) on layer 2.

8. The tiles at locations 8 on layer 1 which are located outside a 3×3 blue square are paired with the tiles in Figure 3.5(d) on layer 2. The tiles at locations 8 on layer 1 which are located within a 3×3 blue square are paired with the tiles in Figure 3.6(b) on layer 2.

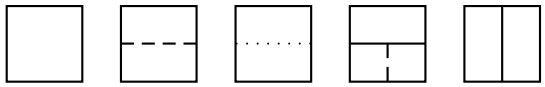
9. Other tiles than the ones in locations shown in Figure 3.3 are paired with the tiles in Figure 3.6(c).



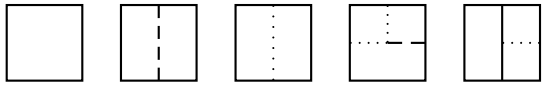
(a) The second layer tiles for tiles of the aperiodic tile set in locations 1



(b) The second layer tiles for tiles of the aperiodic tile set in locations 3

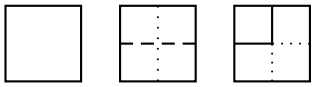


(c) The second layer tiles for tiles of the aperiodic tile set in locations 5



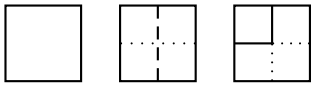
(d) The second layer tiles for tiles of the aperiodic tile set in locations 7

Figure 3.4: The tiles to be paired with the center tiles of blue and red square edges.

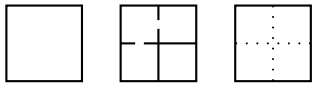


(a) The second layer tiles for tiles of the aperiodic tile set in locations 2

(b) The second layer tiles for tiles of the aperiodic tile set in locations 4

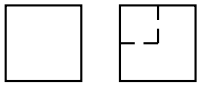


(c) The second layer tiles for tiles of the aperiodic tile set in locations 6

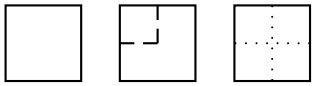


(d) The second layer tiles for tiles of the aperiodic tile set in locations 8

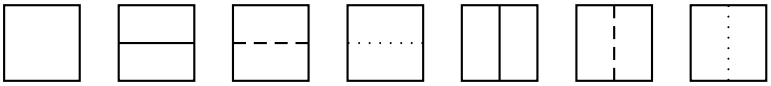
Figure 3.5: The tiles to be paired with the tiles in the corners of blue and red squares.



(a) The second layer tiles for tiles of the aperiodic tile set in locations 4 instead of the tiles in Figure 3.5(b) if the tile of the first layer is in the center of a 3×3 blue square.



(b) The second layer tiles for tiles of the aperiodic tile set in locations 8 instead of the tiles in Figure 3.5(d) if the tile of the first layer is in the center of a 3×3 blue square.



(c) The tiles of layer 2 to be paired with such tiles of layer 1 that are not located in any of the locations in Figure 3.3.

Figure 3.6: The tiles to be paired with the tiles in the center of the smallest blue squares and in other locations than the ones in Figure 3.3.

This new tile set is almost 4-way deterministic. All the tile sets in Figures 3.4, 3.5 and 3.6(c) are 4-way deterministic. However, the tile sets in Figures 3.6(a) and 3.6(b) cause the sandwich tile set to be not 4-way deterministic. Fortunately, these are the only sources of "nondeterminism".

Let T denote the newly constructed tile set with two layers. Let A denote the 3×3 tile set of the aperiodic tile set of Kari and Papasoglu. Let B denote the union of all tiles in Figures 3.4, 3.5 and 3.6. Then

$$T \subseteq A \times B.$$

Let $A_1 \subseteq A$ denote the tiles in locations 4 or 8 in the center of a 3×3 blue square. Let t_d denote the second tile found in the tile sets in Figures 3.6(a) and 3.6(b). Let $T_1 = A_1 \times \{t_d\}$ and $T_2 = T \setminus T_1$. Notice that no sandwich

tile of T_2 contains the tile t_d because by definition t_d is paired only with the tiles of A_1.

Now the tile set U_1 (which is used to draw a "dotted" diagonal line) is defined to be

$$U_1 = T_1 \times T_1^{\circlearrowleft}, \tag{3.1}$$

where $T_1^{\circlearrowleft}$ denotes the tile set which contains the tiles of T_1 after they have been rotated by 180 degrees. The tile set U_2 (which represents the background color) is defined to be

$$U_2 = T_2 \times T_2^{\circlearrowleft},$$

where $T^{\circlearrowleft}$ denotes the tile set which contains the tiles of T_2 after they have been rotated by 180 degrees. Now the tile set U is defined to be

$$U = U_1 \cup U_2.$$

This definition of the tile set U sets a constraint in equation (3.1) that on layer 2 the third tile in Figures 3.6(a) and 3.6(b) is used if, and only if, its rotated counterpart is used on layer 4. This will make the tile set 4-way deterministic because the tile t_d (the second tile Figures 3.6(a) and 3.6(b)), which causes the nondeterminism, is paired only with its rotated counterpart (which will be denoted by $t_d^{\circlearrowleft}$). The tile t_d cannot be distinguished from the blank tile by looking at the colors on the southeast corner whereas the tile $t_d^{\circlearrowleft}$ can be distinguished from the blank tile by looking at the colors on the southeast corner. To say it concisely, the union U of tile sets

$$\begin{aligned} U_1 &= A_1 \times \{t_d\} \times A_1^{\circlearrowleft} \times \{t_d^{\circlearrowleft}\} \text{ and} \\ U_2 &= T_2 \times T_2^{\circlearrowleft} \end{aligned}$$

is 4-way deterministic because the elements of U_1 can be distinguished from the elements of U_2 by watching the colors adjacent to any corner because no sandwich tile of T_2 contains the tile t_d.

Lemma 3.2.1. *The tile set U is 4-way deterministic.*

3.2.2 Drawing the diagonal recursively

In this section it is described how the line patterns drawn by the non-blank tiles in Figures 3.4, 3.5 and 3.6 can be used to distinguish the tiles in a single infinite diagonal row from the tiles in the rest of the tiling. The idea is that the tile set D is such that it admits a tiling which contains a fractal-like line pattern, part of which is shown in Figure 3.7. For brevity, the construction is described again only on layers 1 and 2 because for layers 3 and 4 it is similar.

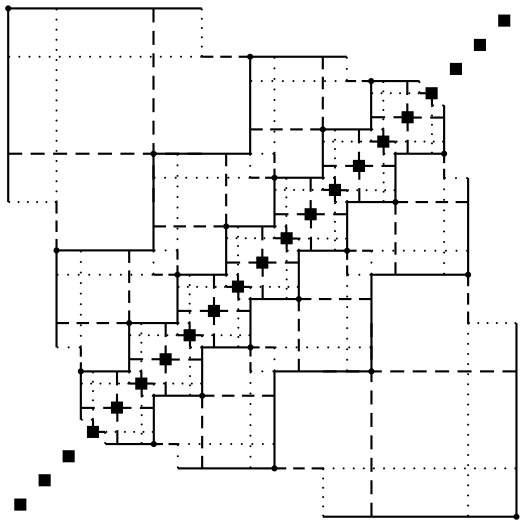


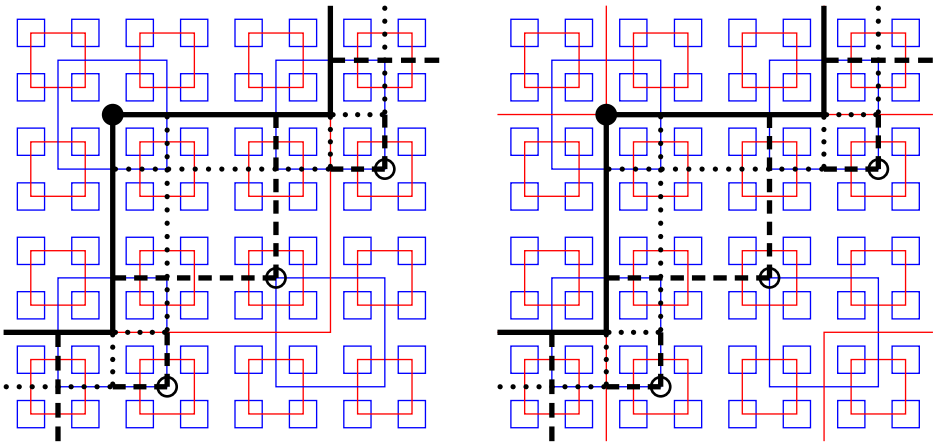
Figure 3.7: The structure of the tile set D is such that the line patterns on top of the non-periodic tiling can draw a recursive line pattern so that a single diagonal row of tiles is tiled with a different set of tiles than rest of the tiling. The locations of sandwich tiles containing tile t_d (and its rotated counterpart) are denoted by black squares.

The most important observation concerning the line pattern in Figure 3.7 is that the upper left half of it can be partitioned into smaller line patterns shown in Figures 3.8(a) and 3.8(b). The other half can also be partitioned into smaller line patterns which are the ones in Figures 3.8(a) and 3.8(b) after rotating by 180 degrees. It is enough to show that the upper left half of the pattern shown in Figure 3.7 can be drawn with the tiles of layer 2 of tile set T because the same pattern can be repeated on layer 4 as a reflection.

With respect to Figure 3.7, Figures 3.8(a) and 3.8(b) should be interpreted so that a line pattern starting from a location denoted by a filled circle is repeated with (approximately) half the size in locations denoted by the empty circles. The line pattern is repeated in smaller size over and over again until the locations denoted by the empty circles are found in the centers of the 3×3 blue squares in which case the pattern is no longer repeated.

Lemma 3.2.2. *The tile set T admits a valid tiling $f : \mathbb{Z}^2 \to T$ such that $f(x, y) \in T_1$ if, and only if, $x = y$ and $x, y \in 4\mathbb{Z}$.*

Proof. First, it can be noted (by listing all the possibilities) that all the different "line interactions" seen in Figures 3.8(a) and 3.8(b) in the locations in Figure 3.3 have been implemented as the tiles in Figures 3.4 and 3.5.



(a) A line pattern starting from the top left corner of a square.

(b) A line pattern starting from the bottom right corner of a square.

Figure 3.8: A recursive signal pattern is used to draw a diagonal line. The partial pattern beginning at the location denoted by a filled circle is recursively repeated with half the previous size at the locations denoted by empty circles.

Furthermore, the tiles used elsewhere than in locations 3.3 are exactly the tiles in Figure 3.6.

Second, it can be concluded with a simple inductive argument that the recursive pattern in Figures 3.8(a) and 3.8(b) can be drawn for arbitrary depth without tiling errors and therefore the tiling f exists.

It can be shown by induction that for every $N > 0$ there exists a (not necessarily valid) tiling f_N such that there is no tiling error in locations (x, y), where $y \leq x + N$, and $f_N(x, y) \in T_1$ if, and only if, $x = y$ and $x, y \in 4\mathbb{Z}$. Then, by the compactness of the space of tilings, there exists the desired kind of tiling f.

Let $N_1 = 0$ and let f_{N_1} be a tiling such that the tiling by the aperiodic tile set (on the first layer) is valid and the tiles in locations $(4k, 4k)$ (where $k \in \mathbb{Z}$) are either upper left corners or lower right corners of squares of size $2^2 + 1 = 5$ (and therefore in both cases centers of 3×3 squares). Let the second layer tiling is to have the second tile t_d in Figures 3.6(a) and 3.6(b) to be located in every location $(4k, 4k)$ whereas the blank tile is to be located in the rest of the locations of the diagonal row (x, x). The second layer is set to have the blank tile also to the right of the diagonal row. To say it concisely, $f_{N_1}(k, k) \in A \times \{t_d\}$ (where A is the aperiodic tile set) if, and only if $k \in 4\mathbb{Z}$. Then, clearly, the tiling is valid between the tiles in locations (x, y), $y \leq x$.

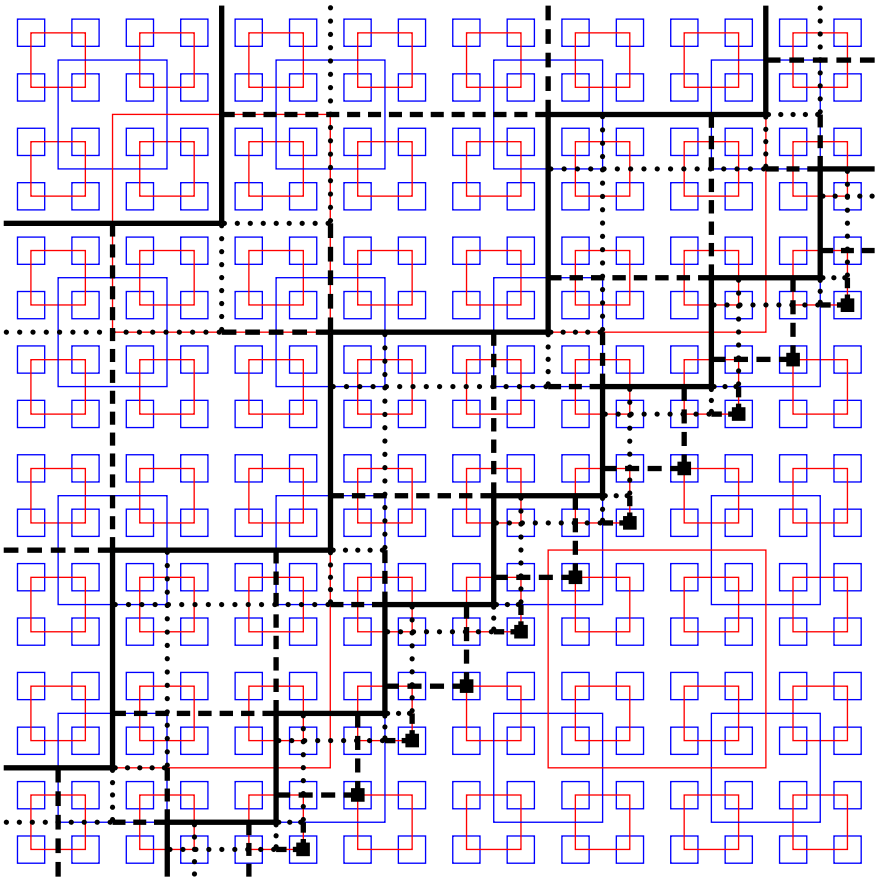


Figure 3.9: The location of the diagonal line is determined by tiles (denoted by small black squares in the picture) which are located in the centers of the 3×3 blue squares and receive dashed lines through their north and west edges.

Let $N_n = \sum_i^n \left(2^n + 2^{n-1}\right)$ (for $n > 1$), which is the distance from the pattern starting location (filled circles in Figure 3.8(a)) to the final diagonal line (filled squares in Figure 3.9), and assume that f_{N_n} is a tiling such that the underlying non-periodic tiling is valid and for every $k \in \mathbb{Z}$ the tile $f_{N_n}((2^n+1)k - N_n, (2^n+1)k + N_n)$ is either an upper left corner or a lower right corner of a $2^n + 1$ square paired with the second tile in Figures 3.5(b) and 3.5(d) and rest of the sandwich tiles in locations $(l - N_n, l + N_n)$ (where $l \mod 4 \neq 0$) have the blank tile on the second layer. Furthermore, it is assumed that tiling is valid for in locations (x, y) where $y \leq x + 2N_n$ and $f_{N_n}(k, k) \in A \times \{t_d\}$ if, and only if $k \in 4\mathbb{Z}$.

Now, let $f_{N_{n+1}}$ be any such tiling that the underlying non-periodic tiling is valid and for every $k \in \mathbb{Z}$ the tile $f_{N_{n+1}}((2^{n+1}+1)k - N_{n+1}, (2^{n+1}+1)k + N_{n+1})$ is either an upper left corner or a lower right corner of a $2^{n+1} + 1$ square paired with the second tile in Figures 3.5(b) and 3.5(d) and rest of the sandwich tiles in locations $(l - N_{n+1}, l + N_{n+1})$ (where $l \mod 4 \neq 0$) have the blank tile on the second layer. By looking at the Figures 3.8(a) and 3.8(b) it can be concluded that the lines starting from locations $((2^{n+1} + 1)k - N_{n+1}, (2^{n+1}+1)k + N_{n+1})$ can be constructed so that eventually the sandwich tiles in locations $(k - N_n, k + N_n)$ are the top left corners or bottom right of squares of size $2^n + 1$ and they have on the second layer the second tile in Figures 3.5(b) and 3.5(d) if, and only if, $k \in 4\mathbb{Z}$ and otherwise the blank tile. This reduces the construction of tiling $f_{N_{n+1}}$ to the construction of tiling $f_{N_{n+1}}$. Because the line patterns could be drawn without tiling error in the diagonal band of locations (x, y), where $y \leq x + 2N_{n+1}$ and $y \geq x + 2N_n$, the validity of the tiling $f_{N_{n+1}}$ in locations (x, y), where $y \leq x + 2N_{n+1}$, follows from the validity of the tiling f_{N_n} in locations (x, y), where $y \leq x + 2N_n$.

Finally, the requested tiling f exists because a tiling can be constructed so that it is valid arbitrarily far from the origin while having the tiles of $A \times \{t_d\}$ located in precisely the locations $(4k, 4k)$ (where $k \in \mathbb{Z}$). □

Lemma 3.2.3. *The 4-way deterministic tile set U admits a valid tiling $f : \mathbb{Z}^2 \to U$ such that $f(x, y) \in U_1$ if, and only if, $x = y$ and $x, y \in 4\mathbb{Z}$.*

Proof. Follows directly from the previous lemma because the tiles of T_1 and $T_1^{\circlearrowleft}$ are paired together which makes the tile set 4-way deterministic. □

Let D denote the 4×4 tile set of the tile set U, that is, $D = U^{4\times4}$. Consider the 4×4 tiles as 4×4 tile clusters in the sense of Figure 2.3. Then D_1 is the subset of D containing the tiles that have an element of U_1 located somewhere on the central diagonal (with positive slope). Tile set D_2 is the subset of D containing the tiles that do not have an element of U_1 located anywhere on the central diagonal. Clearly, $D = D_1 \cup D_2$.

Theorem 3.2.4. *There exists a tile set $D = D_1 \cup D_2$ (where $D_1 \cap D_2 = \emptyset$) such that it is 4-way deterministic and there exists a valid tiling $f : \mathbb{Z}^2 \to D$ such that $f(x, y) \in D_1$ if, and only if, $x = y$.*

Proof. Follows directly from the previous lemma because switching to a 4×4 tile set does not remove the 4-way determinism but enables to determine whether or not a given tile from U_2 is located diagonally between two tiles from U_1 in the tiling f given by the previous lemma. □

3.3 The tiling problem with a seed tile

In this section it will be shown that the tiling problem with a seed tile remains undecidable even if the instances are 4-way deterministic tile sets. This can be seen by applying an additional construction represented in Section 3.2 to the original tile set of Robinson.

3.3.1 The idea for the undecidability proof

The basic idea is to represent Turing machine configurations on horizontal tile rows as was already done in [104]. One tile at each row represents the read-write head and the current letter to be read. The rest of the tiles on the same row represent other cells of the tape located to the left and to the right from the read-write head. This tile set construction has been used earlier in [104, 92].

Unfortunately, the tile set construction does not provide a 4-way deterministic tile set. However, the tile set can be modified to make it 4-way deterministic by using the construction given in Section 3.2. After the modification, the undecidability with the 4-way deterministic instances follows directly.

The tile set is constructed (as a sandwich tile set) in four layers as follows:

Layer 1. The tiling representing a Turing machine computation.

Layer 2. Horizontal signals identifying the move tile pair (on layer 1) occurring on the particular row.

Layer 3. The tiles used to distinguish the leftmost move tiles from alphabet tiles of layer 1.

Layer 4. The tiles used to distinguish the rightmost move tiles from alphabet tiles of layer 1.

Eventually, the following theorem holds:

Theorem 3.3.1. *The tiling problem with a seed tile remains undecidable when restricted to tile sets that are 4-way deterministic.*

Proof (sketch). Section 3.3.2: The tiles of layer 1 can be used to represent an unbounded Turing machine computation started on a blank tape. This layer is forced to contain a tiling error if, and only if, the Turing machine eventually halts.

Section 3.3.4: The tile set can be modified to have no ambiguity between different action tiles or merging tiles at any corner. The tiles of layer 2 are used to distinguish one horizontal pair of an action tile and a merging tile from other tile pairs. This is done by choosing a unique color for any pair of move tiles. The new color is transferred horizontally and paired only with the particular move tiles. Hence, no two move tiles belonging to different read-write operations have the same color on their east edges or their west edges. This layer can be tiled in a valid way if the tiling on layer 1 is valid.

Section 3.3.5: The tile set can be modified to have no ambiguity between an action tile, a merging tile or a alphabet tile at any corner. The tiles of layer 3 are used to distinguish the leftmost tile of the tile pair representing a move from the alphabet tiles of the same row. Likewise, the tiles of layer 4 are used to distinguish the rightmost tile of the tile pair representing a move from the alphabet tiles of the same row. These layers can be tiled in a valid way if the tiling on layer 1 is valid.

The tile sets used on layers 2, 3 and 4 are all 4-way deterministic, so they do not create nondeterminism. The way they are paired with the original tile set on layer 1 makes the final tile set 4-way deterministic. □

Details of the proof are given in the following Sections 3.3.2–3.3.5.

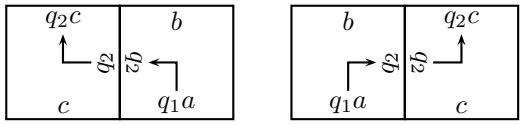
3.3.2 Layer 1: simulating an unbounded Turing machine computation

In this subsection a tile set used for representing a Turing machine computation will be presented [104]. The tiles are used on layer 1 to represent a Turing machine computation which is started from a blank tape.

Left moves Assume that the Turing machine contains move $\delta(q_1, a) = (q_2, b, \triangleleft)$. Then the tile combination in Figure 3.10(a) (for every letter c) is used to represent the move. Therefore the tile in Figure 3.11(a) and all the tiles in Figure 3.12(a) are added to the tile set.

Right moves Assume that the Turing machine contains move $\delta(q_1, a) = (q_2, b, \triangleright)$. Then the tile combination in Figure 3.10(b) (for every letter c) is used to represent the move and the tile in Figure 3.11(b) and all the tiles in Figure 3.12(b) are added to the tile set.

Tape contents For every tape alphabet element a, a tile of the form in Figure 3.13 is added to the tile set. This tile represents a single tape cell and its current contents.



(a) The left move $\delta(q_1, a) = (q_2, b, \triangleleft)$

(b) The right move $\delta(q_1, a) = (q_2, b, \triangleright)$

Figure 3.10: The tile combinations to represent different Turing machine moves.

(a) The action tile for the left move $\delta(q_1, a) = (q_2, b, \triangleleft)$

(b) The action tile for the right move $\delta(q_1, a) = (q_2, b, \triangleright)$

Figure 3.11: Action tiles

Initial configuration To force the Turing machine to start on a blank tape only, the tiles in Figure 3.14 are added to the tile set. One of these tiles (namely, the one in Figure 3.14(b)) is chosen to be the seed tile. If the seed tile is contained within a tiling, then a valid tiling will necessarily represent a non-halting Turing machine computation. The tiles in Figures 3.14(a) and 3.14(c) define the tape to be initially empty. In short, if the seed tile is located in the origin, then the Turing machine simulation is done in the first and the second quadrant of the plane. The tiles in Figures 3.14(d) and 3.14(e) allow the lower part of the plane to be always correctly tiled. The tile in Figure 3.14(d) is required to allow the south edge of the tile in Figure 3.14(b) to have a different color than the south edges of the tiles in Figures 3.14(a)

(a) Merging tiles for all the left moves.

(b) Merging tiles for all the right moves.

Figure 3.12: Merging tiles

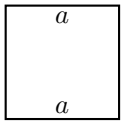
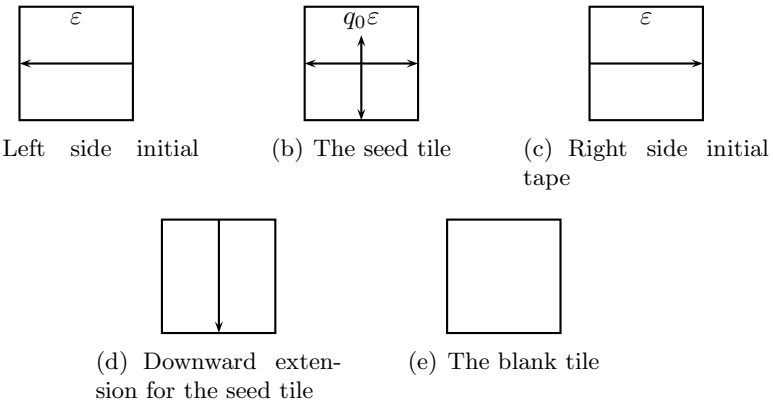


Figure 3.13: An alphabet tile

and 3.14(c). This is required to achieve SE- and SW-determinism. An example is given in Figure 3.15 on the use of tiles in Figure 3.14.



(a) Left side initial tape

(b) The seed tile

(c) Right side initial tape

(d) Downward extension for the seed tile

(e) The blank tile

Figure 3.14: Starting tiles

Following the terminology of Robinson [92], the tiles shown in Figure 3.11 are referred to as *action tiles*. Every Turing machine move is represented by a unique action tile. The tiles shown in Figure 3.12 are called *merging tiles* and the tiles of the form shown in Figure 3.13 are called *alphabet tiles*. The tiles in Figure 3.14 are referred to as *starting tiles*. The final tile set consists of a unique action tile for every non-halting move, all the possible merging tiles, all the possible alphabet tiles and the starting tiles.

Let q and a be a pair of a read-write head state and a tape letter for which the Turing machine transition $\delta(q, a)$ has not been defined. Then there will be no tile that would have the color qa on its south edge. Therefore, if the Turing machine eventually halts, that is, if at some moment of time the read-write head in state q reads letter a, then the tiling cannot be completed to cover the entire plane in a valid way. From this it follows that the simple version of the tiling problem with a seed tile is undecidable.

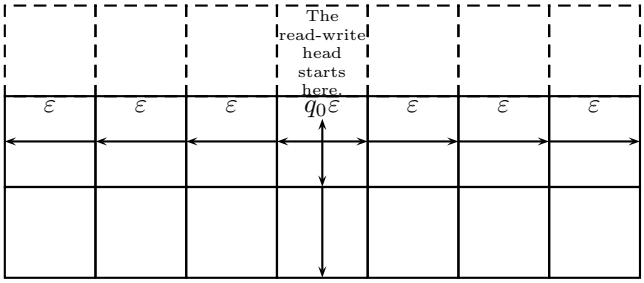


Figure 3.15: The tile pattern representing the initial configuration of the Turing machine computation.

3.3.3 Nondeterminism of the Turing machine tile set

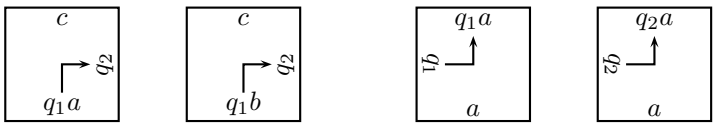
The tiles which are used for Turing machine simulation are themselves sufficient to show that the original tiling problem with a seed tile is undecidable. However, the tile set is not 4-way deterministic.

The tile set is not 4-way deterministic (only) because

Problem 1. two different action or merging tiles cannot always be distinguished from each other (see Figure 3.16 for the case of right moves) and

Problem 2. action and merging tiles cannot always be distinguished from the alphabet tiles (see Figure 3.17 for the case of right moves).

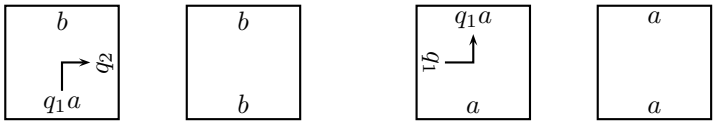
Fortunately, these are the only sources of nondeterminism in the tile set. The tiling problem (with a seed tile) will be seen to be undecidable in the 4-way deterministic case by modifying the Turing machine tile set so that problems 1 and 2 are solved (and the resulting tile set is 4-way deterministic otherwise also). Problem 1 will be removed in Section 3.3.4 and problem 2 will be removed in Section 3.3.5.



(a) Ambiguity at the northwest corner and northeast corner.

(b) Ambiguity at the southeast corner.

Figure 3.16: The tiles representing a Turing machine move cannot be distinguished.



(a) Ambiguity at the northwest corner.

(b) Ambiguity at the southeast corner.

Figure 3.17: A tile representing a Turing machine move and an alphabet tile cannot be distinguished.

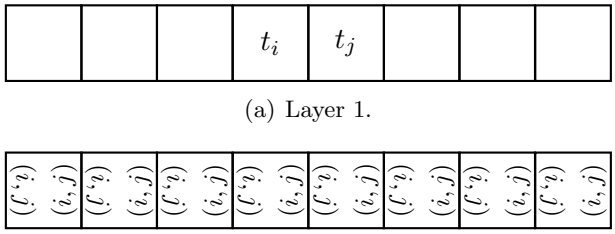
3.3.4 Layer 2: distinguishing different move tiles

Let the number of different action tiles and merging tiles of layer 1 be n for the given Turing machine. For simplicity, let the different action tiles and merging tiles be denoted by expressions $t_1, \ldots, t_n$, that is, integer i identifies tile t_i uniquely.

Let t_k be any action tile or merging tile occurring on layer 1. Then the tile is paired with the tiles of the form shown in Figure 3.18 with either $i = k$ or $j = k$. If the tile is of the form in Figures 3.11(b) or 3.12(a), it is required that $i = k$ and $j \neq k$. Otherwise, if the tile is of the form in Figures 3.11(a) or 3.12(b), it is required that $i \neq k$ and $j = k$. In other words, in the tile pair representing a move, the leftmost tile is identified by the first component and the rightmost tile is identified by the second component in the pair (i, j). This makes it possible to distinguish any two action tiles or merging tiles from each other just by observing the color of (either) the east edge or the west edge.

Figure 3.18: The tiles of layer 2.

All the other tiles are paired freely with all the tiles of the form shown in Figure 3.18, where $1 \leq i, j \leq n$. On a valid tiling, layer 2 consists of rows of tiles like the one in Figure 3.19(b) if the underlying tile row contains action tile and move tile pair t_i and t_j as in Figure 3.19(a).



(a) Layer 1.

(b) Layer 2.

Figure 3.19: Associating tiles t_i and t_j with the horizontal signal identifying the tile pair.

3.3.5 Layers 3 and 4: distinguishing move tiles from alphabet tiles

Let D be the tile set of Theorem 3.2.4 which is used to draw a northeast-southwest diagonal line 4-way deterministically. Let $D = D_1 \cup D_2$, where D_1 is the set of tiles used on the diagonal line only and let $D_2 = D \setminus D_1$. Let D^R be the tile set which has been constructed by interchanging the north edge colors and south edge colors in the tiles of set D and by replacing the east edge colors and the west edge color bijectively with a completely new set of colors. Tile set D^R can be used to tile a diagonal line in the northwest-southeast direction. Let D_1^R be the set of tiles located on this diagonal pattern and let again $D_2^R = D^R \setminus D_1^R$.

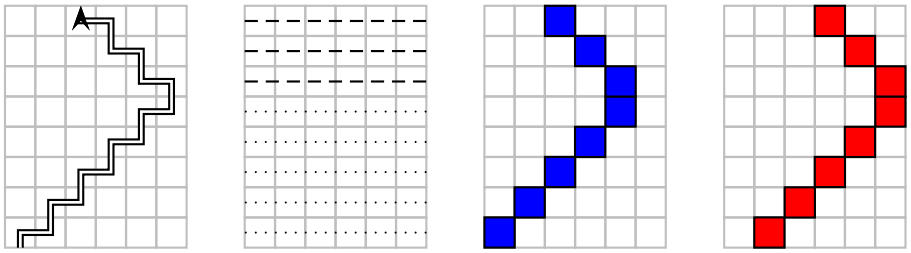
Now the tile set $D \cup D^R$ is 4-way deterministic and it can be used to draw any diagonally advancing zig-zag line with exactly the elements $D_1 \cup D_1^R$ on the zig-zag line. Using tile sets D and D^R one can distinguish action tiles and merging tiles from alphabet tiles by pairing the action tiles and the merging tiles of layer 1 with the tiles representing a diagonal line.

The tile set $D \cup D^R$ is used on both layer 3 and layer 4. The action tiles in Figure 3.11(a) are paired with the tiles of set D_1^R on layer 4. The action tiles in Figure 3.11(b) are paired with the tiles of set D_1 on layer 3. The merging tiles in Figure 3.12(a) are paired with the tiles of set D_1^R on layer 3. The merging tiles in Figure 3.12(b) are paired with the tiles of set D_1 on layer 4. These constraints force two diagonally advancing ziz-zag lines to be drawn side by side on layers 3 and 4. The line pattern of layer 3 is associated with merging tiles of left moves and action tiles of right moves (that is, the leftmost tile in a tile pair representing a move). The line pattern of layer 4 is associated with merging tiles of right moves and action tiles of left moves (that is, the rightmost tile in a tile pair representing a move).

All the alphabet tiles are paired with all the tiles in $D_2 \cup D_2^R$ on both layers and the starting tiles are paired freely with all the tiles $D \cup D^R$ on both

layers. This can be done because there is no ambiguity between starting tiles and other tiles.

The idea of this construction is better seen in Figure 3.20. The left side tiles are identified by the tiles on the diagonal lines on layer 3 and the right side tiles are identified by the tiles on the diagonal lines on layer 4.



(a) Read-write head movement on layer 1.

(b) Signals on layer 2 for different moves.

(c) Simulating a diagonal line on layer 3.

(d) Simulating a diagonal line on layer 4.

Figure 3.20: Distinguishing move tiles from alphabet tiles by using diagonal patterns that can be drawn 4-way deterministically.

If layer 1 has been tiled in a valid way, also layers 3 and 4 can be tiled correctly. This follows from the fact that a row of tiles of D can always be followed by a row of tiles D^R with matching colors since D^R is only a "reflected" version of D and D admits a valid tiling.

This construction distinguishes the action tiles and the merging tiles from the rest of the tiles. By looking at the tiles on layers 3 and 4 it can be determined whether or not the tile in question is an alphabet tile or not. The tile is an alphabet tile if, and only if, it is not a starting tile and it is not paired with a member of sets D_1 or D_1^R on layers 3 and 4.

3.4 The tiling problem without a seed tile

In this section the tiling problem without a seed tile is shown to be undecidable even for 4-way deterministic tile sets. The argumentation is quite similar to that of earlier proofs [46, 92]. The only difference is the requirement that the final tile set must be 4-way deterministic.

3.4.1 A brief outline of the argumentation

The undecidability is proven by reducing the tiling problem with a seed tile to the tiling problem. The reduction is done by converting an instance of the tiling problem with a seed tile to an instance of the tiling problem.

The idea of the reduction is to construct another tile set. For the new tile set, the answer for the tiling problem will be affirmative if, and only if, the answer for the tiling problem with a seed tile is affirmative for the original given tile set and the given seed tile.

The new tile set is such that certain areas of a valid tiling are used to simulate a tiling with the original tile set. These areas are referred to as free rows and free columns. Identifying the free areas in the earlier case [46] required some modifications to the original proof of Robinson [92]. Now the tile set construction for identifying the free areas is more complicated.

The construction of the new tile set relies heavily on the use of an aperiodic tile set. By using the square patterns generated by Robinson's tile set, copies of the seed tile are forced to be located at certain points of the plane.

In [46], Robinson's tile set was modified resulting a new aperiodic tile set which is deterministic by one corner. Later in [55] an aperiodic 4-way deterministic tile set was given which can be mapped homomorphically onto Robinson's original aperiodic tile set. This 4-way deterministic tile set will be used in the proof instead of Robinson's tile set. This set was already used in Section 3.2 to construct another tile set.

The new tile set is constructed in four layers for the given tile set T and a seed tile $t \in T$. The rough outline of the layers is the following:

Layer 1. The tiling forced by the 3×3 tile set (see the definition in Section 2.3) of the aperiodic tile set of Kari and Papasoglu.

Layer 2. The tiles to identify free areas.

Layer 3. A tiling simulating a tiling by the given tile set T.

Layer 4. The tiles to forward the colors of the tile set T on layer 3 from a free area to another free area border and from a red border to another red border without discarding any color information.

Theorem 3.4.1. *The tiling problem is undecidable even when restricted to tile sets that are 4-way deterministic.*

Proof (sketch). Section 3.4.2: It is possible to divide the plane into squares of increasing size using (the 3×3 tile set of) the aperiodic tile set of Kari and Papasoglu. The squares are colored either red or blue. No edges of two squares of the same color can coincide.

Section 3.4.3: Each of the red squares contains free areas that are not between any of the smaller red squares. The tiles which are used on free and non-free areas belong to two disjoint subsets of the modified aperiodic tile set.

Section 3.4.4: A finite area of a tiling by the original tile set is simulated on the free areas within the red squares. The size of the free area inside a

red square square is directly proportional to the size of the red square. One copy of the seed tile can be forced to be located in the center of every red square (and therefore in the center of every simulation area) by pairing only the seed tile with all blue cross tiles that are not part of a 3×3 blue square. Whether or not a cross tile is part of a 3×3 blue square is determined by using a 3×3 tile set (see Section 2.3 for the definition) version of the aperiodic tile set.

Section 3.4.5: The area consisting of disjoint free areas can be considered as a single continuous square. This is seen by transferring the colors between the free areas using a 4-way deterministic construction. Also, if the original tile set admits arbitrarily large squares to be tiled, so does the new tile set. This is possible because the colors next to a red edge on a free area are chosen nondeterministically and shared between all the red squares of the same size using horizontal and vertical signals containing the color information. Then the plane is tiled correctly if, and only if, on every red square the free areas are tiled correctly using the original tile set. Any valid tiling by the original tile set can be simulated using the new tile set without a tiling error. □

The details of the proof of Theorem 3.4.1 have been scattered to Sections 3.4.2–3.4.5.

Theorem 3.4.2. *The tiling problem is undecidable even when restricted to tile sets that are deterministic by any two edges.*

Proof. A 2×2 tile set (see the definition of an $n \times n$ tile set in Section 2.3) of a 4-way deterministic tile set is not only 4-way deterministic but also deterministic by opposite edges. Therefore, the claim follows from the previous theorem. □

3.4.2 Layer 1: the aperiodic tile set

The aperiodic tile set of Kari and Papasoglu is used as the first layer of the new sandwich tile set. This 4-way deterministic tile set can be mapped homomorphically onto Robinson's tile set and therefore it is possible to use the patterns of red squares and blue squares of Robinson's original tile set in the rest of the proof. The aperiodic tile set is used to admit only non-periodic valid tilings in which the seed tile is contained infinitely many times.

3.4.3 Layer 2: identifying the free areas

A tile within a red square is said to be located on a *free column* of the red square, if there are no smaller red squares above or below it within the red square. Likewise, a tile within a red square is said to be located on a *free row* of the red square, if there are no smaller red squares within the red square to the right or to the left from its position. A tile within a red square

is said to be *free*, if it is located on both a free row and a free column. For example, the free tiles of a $(2^6 + 1) \times (2^6 + 1)$ red square are shown in Figure 3.21. The aperiodic tile set will be modified so that it is possible to determine whether or not a given tile within a red square is located on a free area or not.

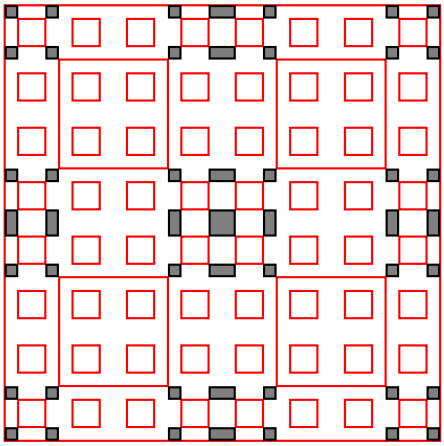
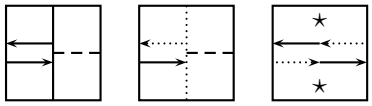


Figure 3.21: The free area of 9×9 squares within a red square spanning 65×65 squares.

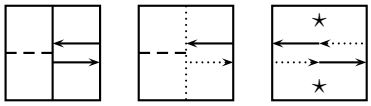
A row is said to be *red*, if it consists (only) of tiles having red horizontal arrows. Similarly, a row is said to be *blue*, if it consists (only) of tiles having blue horizontal arrows. Likewise, a column of tiles is said to be *red* or *blue* if it consists of tiles having red or blue vertical arrows, respectively. It needs to be noted that a row within a red square can be free only if it is a part of a blue row.

Robinson's tile set is such that in a valid tiling there is always an even number red columns and rows within a red square. On the area between two neighboring red squares of equal size there can be either an even or an odd number of red columns (if the squares lie on the same tile rows) or rows (if the squares lie on the same tile columns). The same properties hold for blue squares, columns and rows.

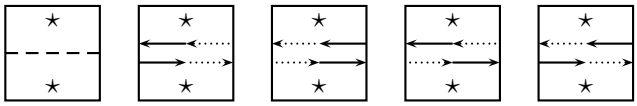
The number of columns and rows between two neighboring red squares of equal size is used to force certain line patterns to be drawn which eventually can be used to locally recognize the tiles that lie on free columns and rows. The line patterns on non-free areas consist of two signals which change their parity whenever they intersect a red column or row. The number of the red columns or rows between two neighboring squares of equal size affects on how the signals are initialized at the border of a red square.



(a) The tiles to be paired with west edge red double arrows



(b) The tiles to be paired with east edge red double arrows



(c) The tiles to be paired with any vertical red single arrows

Figure 3.22: The tiles to be paired with the tiles on blue rows and red columns. Expression $\star$ denotes either a solid or a dotted vertical line and it is the same for both the north edge and the south edge.

To identify the free rows (construction is only rotated to identify free columns), the aperiodic tile set is now modified as follows:

1. The tiles on the west edges of red squares are paired with the tiles in Figure 3.22(a).

2. The tiles on the east edges of red squares are paired with the tiles in Figure 3.22(b).

3. The tiles on red vertical single arrows are paired with the tiles in Figure 3.22(c).

4. The rest of the tiles are paired with such tiles that the colors originating from tiles in Figures 3.22(a), 3.22(b) and 3.22(c) can intersect unchanged.

The idea of the tiles in Figure 3.22 is that in a valid tiling a tile within a red square is located on a free row if, and only if, it contains a horizontal dashed line. If the tile is located on a blue row but not on a free area, then it must have a pair of horizontal arrows travelling to opposite directions. Each of these arrows is either solid or dotted. On a red column a solid arrow is changed to a dotted arrow and vice versa.

Every red column contains a single vertical line which is either solid or dotted. These vertical lines are used to determine how a dashed line representing a free row is swapped to the two arrows on the edge of a red square. Practically speaking, the vertical lines are used to identify nondeterministically, how many red columns there are between two nearest red squares of equal size. If there is an even number of red columns between the two squares, then both the east edge of the leftmost square and the west edge of

the rightmost square should contain a solid vertical line. If there is an odd number of red columns between the two squares, then both the east edge of the leftmost square and the west edge of the rightmost square should contain a dotted vertical line.

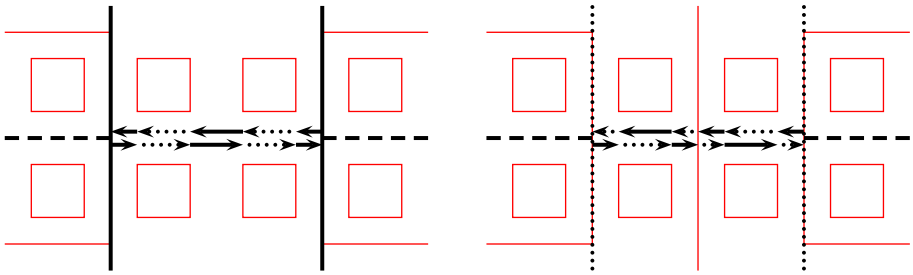
On the edge of a red square a dashed line representing a free row is replaced with a leftward arrow and a rightward arrow. The replacement is determined by the vertical line (which is either solid or dotted) on that red column. If an east edge contains a solid vertical line, the rightward arrow (which is emitted away from the square) must be solid also. If an east edge contains a dotted vertical line, the rightward arrow must be dotted also. If a west edge contains a solid vertical line, the leftward arrow must be solid also. If a west edge contains a dotted vertical line, the leftward arrow must be dotted also. When the vertical lines are assumed to be of the same type on opposite edges of neighboring red squares, the arrow which is absorbed on a red edge is uniquely determined by the emitted arrow on the edge of the opposite red square.

The solid or dotted black arrows emitted at the ends of free rows of two neighboring red squares of equal size may intersect a vertical edge of another (larger) red square only exactly in the middle between the two squares. Let the two squares be of size $2^{2n}+1$ and let them be aligned so that both their horizontal edges are located on the same rows, the leftmost square has its east edge located on column $x = -2^{2n-1}$, the rightmost square has its west edge located on column $x = 2^{2n-1}$ and the squares have free rows on a blue row in location $y = 0$. Then on intervals $(-2^{2n-1}, 0)$ and $(0, 2^{2n-1})$ there is an even number of red columns and the column $x = 0$ is either red or blue. Furthermore, on row $y = 0$ there are no vertical red square edges on intervals $(-2^{2n-1}, 0)$ and $(0, 2^{2n-1})$. The only possibility on row $y = 0$ to have a red vertical edge (i.e. a vertical red double arrow on Robinson's tiles) on interval $(-2^{2n-1}, 2^{2n-1})$ is in location $(x, y) = (0, 0)$. Because both the subintervals contain an even number of red columns, the horizontal black arrows are tiled correctly in location $(0, 0)$ using the tiles in Figure 3.22(c) if there is no red vertical edge present. If there is a red vertical edge present in location $(0, 0)$, then the horizontal black arrows are still drawn correctly using the third tile in Figures 3.22(a) and 3.22(b).

It should be noted that there exists such a tiling in which there is a blue row which does not intersect any red squares (a so-called fracture line). It follows that this row can therefore contain either a pair of horizontal arrows or a dashed line without causing a tiling error.

Lemma 3.4.3. *If a tile within a red square is located on a non-free row in a valid tiling, then it cannot contain a horizontal dashed line.*

Proof. If a tile on a non-free row contains a dashed line, the line would collide with an edge of a smaller red square. But this is not possible, because a

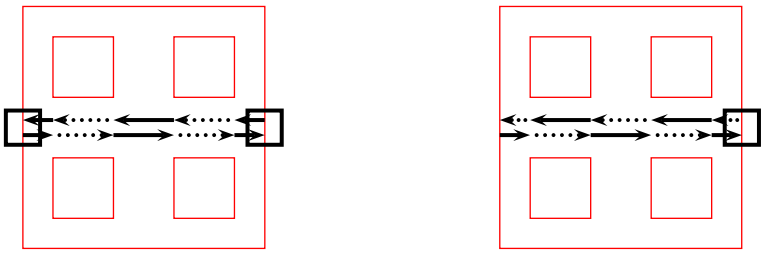


(a) Both the horizontal arrows are solid at the end of a free row if the area between the two squares contains an even number of red columns.

(b) The horizontal arrows are of different type at the end of a free row if the area between the two squares contains an odd number of red columns.

Figure 3.23: Different tiles are used at the end of free rows depending on the (even or odd) number of red columns between the two red squares.

dashed line coming from the outside of a square cannot collide with an edge of the square by the tiles in Figures 3.22(a) and 3.22(b). □



(a) If in the middle of the row both the leftward arrow and the rightward arrow are solid, then neither the east edge nor the west edge cannot be tiled correctly using the tiles in Figures 3.22(a) and 3.22(b).

(b) If in the middle of the row the leftward arrow is dotted and the rightward arrow is solid, then east edge cannot be tiled correctly using the tiles in Figure 3.22(b).

Figure 3.24: A free row can be tiled only with a horizontal dashed line. The locations denoted by black rectangles cannot be tiled.

Lemma 3.4.4. *If a tile within a red square is located on a free row in a valid tiling, then it must contain a horizontal dashed line.*

Proof. The arrows cannot be of the same type (solid or dotted) because then at the end of the free row the arrows would still be of the same type as shown in Figure 3.24(a). A leftward arrow and a rightward arrow of the

same type is not allowed inside the red square by the tiles in Figures 3.22(a) and 3.22(b).

If the arrows are not of the same type, then they will still meet the red square edges on the east side and the west side as the same arrow type combination because there is an even number of red columns within the square and at every red column the arrow types are swapped shown in Figure 3.24(b). But the same arrow combinations arriving at the both edges is not possible because on the west side the leftward arrow and the rightward arrow must be dotted and solid, respectively, whereas the east side the leftward arrow and the rightward arrow must be solid and dotted, respectively, by the third tile in Figures 3.22(a) and 3.22(b). □

By Lemmas 3.4.3 and 3.4.4, it is obvious that in a valid tiling a tile is located on a free row if, and only if, it has a horizontal dashed line. It remains to be shown that a valid tiling exists.

Lemma 3.4.5. *The modified tile set admits a valid tiling.*

Proof. A row which does not intersect with any red square edges can be tiled correctly. Therefore, it is enough to consider only rows that contain free tiles for some red squares. Consider row $y = 0$, which is assumed to contain free rows of red squares of size $2^{2n} + 1$.

Clearly, any free row itself within a square of size $2^{2n} + 1$ can be tiled correctly by the tiles containing a horizontal dashed line. It is therefore sufficient to show that no tiling error is introduced on row $y = 0$ on the area between two neighboring red squares of size $2^{2n} + 1$ assuming that at the ends of a free row the dashed line is replaced with a suitable pair of arrows. Let the two neighboring squares be located so that the leftmost square has its east edge on column $x = -2^{2n-1}$ and the rightmost square has its west edge on column $x = 2^{2n-1}$.

If the area in between (i.e. columns on interval $(-2^{2n-1}, 2^{2n-1})$) contains an even number of red columns (as shown in Figure 3.23(a)), the column $x = 0$ is blue and the row $y = 0$ can be tiled correctly by choosing the vertical lines to be solid on the edges of the red squares. The selection which is made between solid and dotted vertical lines on the east and west edges of red squares is "row invariant" because the selection will depend only on the number of red columns between the red squares.

If the area in between the two squares contains an odd number of red columns (as shown in Figure 3.23(b)), the vertical lines are chosen to be dotted on the edges of the red squares. Then the east edge of the leftmost red square has a dotted rightward arrow and a solid leftward arrow. If the interval $(-2^{2n-1}, 2^{2n-1})$ does not contain a vertical red edge, the arrows are swapped an odd number of times on the interval using the tiles in Figure 3.22(c) and the west edge of the rightmost square will have a solid rightward

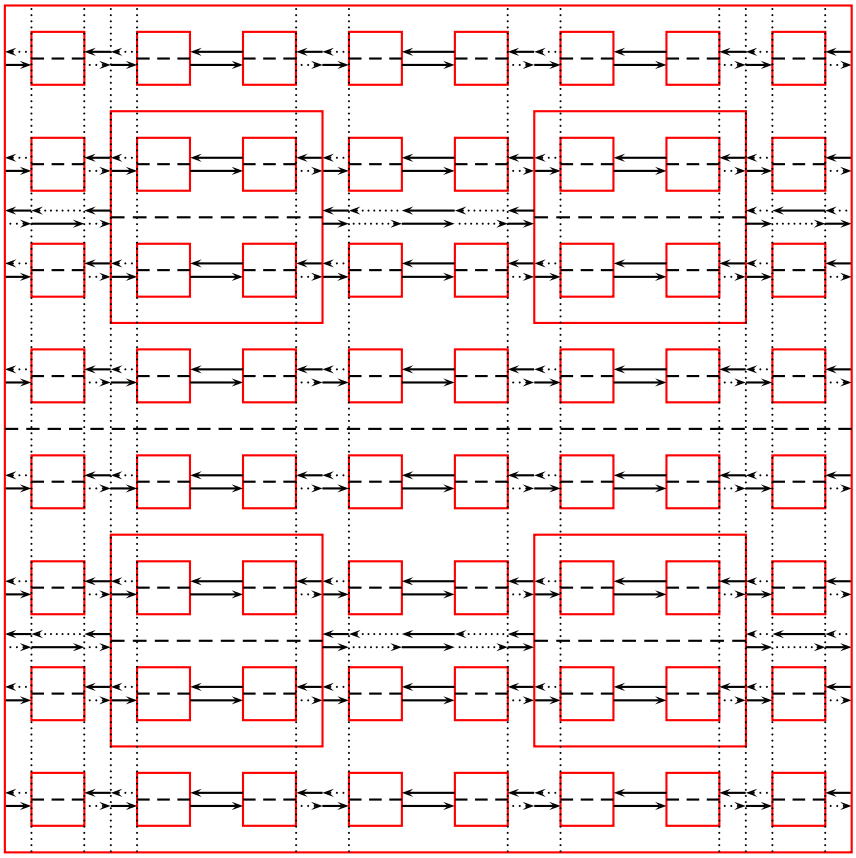


Figure 3.25: Signals on free rows within a 65×65 red square. For clarity, horizontal signals are shown only for the central row of each square and vertical solid signals are not shown.

arrow and a dotted leftward arrow. These arrow combinations are allowed by the tiles in Figures 3.22(a) and 3.22(b). Therefore, no tiling error is introduced if the interval $(-2^{2n-1}, 2^{2n-1})$ does not contain a vertical red edge and the vertical line components are chosen correctly on the edges of the red squares.

The only location on row $y = 0$ where the horizontal (solid or dotted) black arrows might meet a vertical red edge on interval $(-2^{2n-1}, 2^{2n-1})$ is on column $x = 0$. Because the arrows are swapped an even number of times on both intervals $(-2^{2n-1}, 0)$ and $(0, 2^{2n-1})$, the tile required on column $x = 0$ is exactly the third tile in Figures 3.22(a) and 3.22(b) or the fourth tile in Figure 3.22(c) (which are all the same). Therefore, no tiling error is introduced even if the interval $(-2^{2n-1}, 2^{2n-1})$ contains the only possible occurrence of a vertical red edge in the middle of the interval. □

It is shown in Figure 3.25 how the signals in tiles in Figure 3.22 are drawn within a 65×65 red square.

The following theorem follows as a combination of the previous lemmas and their "rotated" counterparts:

Theorem 3.4.6. *The 4-way deterministic aperiodic tile set of Kari and Papasoglu can be modified so that the set of tiles on free areas and the set of tiles on non-free areas are disjoint sets.*

3.4.4 Layer 3: simulating the original tile set

Assume that the given instance for the tiling problem with a seed tile is the tile set T and the seed tile t. A tiling by the original tile set T is simulated within all the red squares. However, since larger red squares contain smaller red squares, the simulation area cannot be the entire square itself. Instead, the simulation corresponding the particular red square is done on the free areas.

Lemma 3.4.7 ([92]). *For every $(4^n + 1) \times (4^n + 1)$ red square, the number of free columns is $2^n + 1$ and the number of free rows is $2^n + 1$.*

Lemma 3.4.7 states that the free area within a red square increases with respect to the size of the square. Hence, the tiling by the original tiles (on layer 3) can be arbitrarily large even when restricted to the free rows and free columns. Hence, it is enough to restrict the simulation only to free rows and free columns.

One copy of the seed tile can be forced to be located in the center of every red square (and therefore in the center of every simulation area) by pairing the seed tile with all blue cross tiles that are not part of a 3×3 blue square. Whether or not a cross tile is part of a 3×3 blue square is determined simply by using a 3×3 tile set version of the aperiodic tile set. The tiles that are used on this layer are the tiles of the original tile set T (which is assumed not to have blank as a color), the blank tile and some other tiles that have some sides colored blank and some colored with the colors of T. In Section 3.4.5 these tiles will be described in more detail and it will be shown to be possible to consider the area consisting of the free areas as a single continuous square.

3.4.5 Layer 4: joining the free areas

To treat all the free areas within a red square as a single continuous area and to allow any colors at the edges of red squares while maintaining the 4-way determinism, a set of tiles is added to the original tile set and these tiles are paired with another set of tiles as shown in Figure 3.26. Notice that the tiles in Figure 3.26 forward only colors of the north and south edges.

A similar construction is used to forward colors in the horizontal direction and the actual tile set is the combination of tiles in Figure 3.26 and their counterparts for the horizontal direction.

Let f be a valid tiling of the aperiodic tile set. A tile $f(x, y)$ is said to be located on the *north border*, *east border*, *west border* or *south border* of a free area if a tile $f(x, y-1)$, $f(x-1, y)$, $f(x+1, y)$ or $f(x, y+1)$, respectively, belongs to the free area and the tile $f(x, y)$ does not. Collectively, the tiles that do not belong to a free area but are located horizontally or vertically next to the free area are said to be located on a *border* of the free area. The locations enumerated in Figure 3.26 can be identified as disjoint subsets of the aperiodic tile set after the modifications of layer 2 (i.e. Section 3.4.3) and forming the 3×3 tile set of the new modified aperiodic tile set. By looking at the tiles in Figure 3.26, it is obvious that the final tile set is 4-way deterministic.

What needs to be shown is that tiles force arbitrarily large squares to be tiled with the tiles of the original tile set. This is done by erasing colors at the free area borders and forwarding them to the next free area unless the color is adjacent to a red edge. At the red edges the color is erased and forwarded to the next red square. Another way of looking at it is that the final colors next to red edges are chosen nondeterministically for all red squares of the same size.

For example, at the north border of a free area the last color (i.e. x in Figure 3.26) on layer 3 is erased and raised onto layer 4 to be transferred northwards. At the south border of another free area color x is lowered from layer 4 back to layer 3. The last colors next to red edges are chosen nondeterministically using the tiles in Figures 3.26(e) and 3.26(f). The principle of forwarding colors is represented in Figure 3.27.

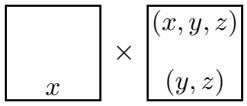
It can be concluded that any color is allowed on layer 3 next to a red edge and the free areas can be considered as a single continuous area. Therefore the new tile set can simulate tiling of arbitrarily large squares by the original tile set and Theorem 3.4.1 follows.

3.5 The square tiling problem is NP-complete

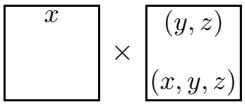
In this section it is shown that the square tiling problem is NP-complete even when the instance is a 4-way deterministic tile set.

3.5.1 Preliminaries

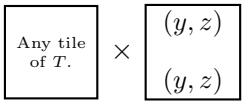
The *square tiling problem* is defined as follows [35]: "Given an integer N (in unary form) and a set of Wang tiles T, does there exist a valid tiling of an $N \times N$-square by these given tiles?" The square tiling problem is known to be NP-complete [35]. A variant of the square tiling problem is the *periodic*



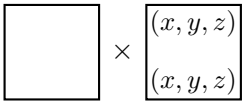
(a) The tiles at the north border of a free area.



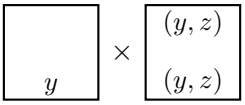
(b) The tiles at the south border of a free area.



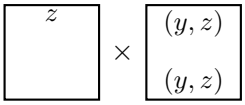
(c) The tiles on free areas.



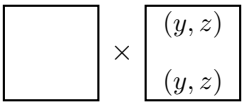
(d) The tiles on non-free areas.



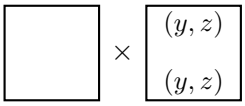
(e) The tiles on a red north edge next to a free area.



(f) The tiles on a red south edge next to a free area.



(g) The tiles on a red north edge next to a non-free area.



(h) The tiles on a red south edge next to a non-free area.

Figure 3.26: The tile construction on layers 3 and 4 to transfer vertical colors between the free areas inside a red square and between different red squares. Expressions x, y and z denote arbitrary colors of the given tile set T. The set T is assumed not to have blank color on any tile.

tiling problem: "Given an integer N (in unary form) and a set of Wang tiles T, does there exist a periodic tiling of the plane with a period less than N?" Also the periodic tiling problem is known to be NP-complete [35].

Mathematical self-assembly is the concept of modelling chemical self-assembly with Wang tiles [2, 1]. One is given a tile set T, a *seed tile* $s \in T$, a *temperature* $\tau \in \mathbb{N}$ and a *glue function* $g : C \times C \to \mathbb{Z}$, where C is the set of colors of T and $g(a, b) = g(b, a)$. The tiles represent molecules and the glue function is used to represent the bond strength between the sides of different molecules (i.e. tiles). In its simplest form, self-assembly is started from a single seed tile and tiles are added one by one to an existing connected cluster of tiles. A tile can be appended to an existing tile cluster if there is a slot around which the sum of the glue values between the tile and the earlier tiles exceeds or equals the temperature. A more thorough discussion of mathematical self-assembly can be found in [2, 1]. A tile cluster (of more than one tile) is called *terminal*, if no more tiles can be appended to it. The

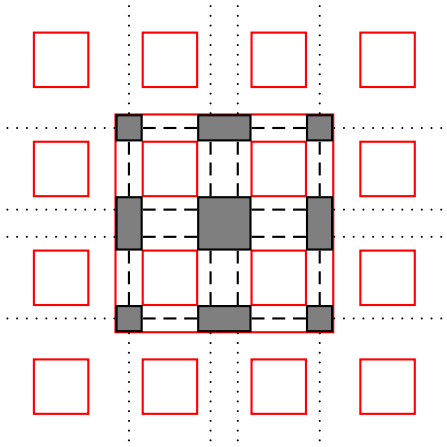


Figure 3.27: Forwarding the colors from one free area to another and from one red square to another using the construction in Figure 3.26. Dotted lines represent two-color vectors and dashed lines represent three-color vectors. Darkened rectangles represent free areas.

definition of glue function allows tiling errors (i.e. mismatching colors) in the assembled tile clusters. However, if the glue function is restricted not to allow tiling errors in the assembled clusters, then the concept of determinism by edge colors can be extended to self-assembly also.

The *unique shape problem* [2] is the following: "Given a tile set T, a seed tile $s \in T$, a glue function g, a temperature τ and a shape S, does the assembly always produce a terminal tile cluster of shape S?" The *unique shape model* is a model of mathematical self-assembly, where $g(a, a) > 0$, $g(a, b) = 0$ if $a \neq b$ and assembled tile clusters are distinguished only by their shape.

It is shown that the square tiling problem [35], the periodic tiling problem (with a bounded period) [35] and the complement of a special case of the unique shape problem [2] are NP-complete even if the given tile set is deterministic by any two sides. That is, the problems remain NP-complete even when the tile set is of a very restricted form. The reductions are done using the well-known *satisfiability problem* (or *SAT* in short), which is known to be NP-complete [20, 35].

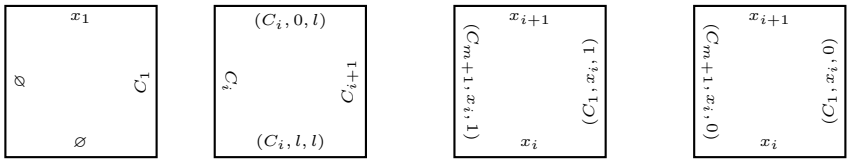
3.5.2 NP-completeness with 4-way deterministic tile sets

In this section it is shown that the NP-complete satisfiability problem can be reduced to the square tiling problem for 4-way deterministic tile sets. The construction is based on the construction given in [2], which again is based on the construction in [61]. In [2], the variables with values were

represented by the rows and the clauses were represented by the columns of the rectangle. The tile set constructed in [2] was only SW-deterministic.

In the following, the clauses are denoted by C_i, where $1 \leq i \leq m$, and the variables are denoted by x_j, where $1 \leq j \leq n$. The construction of the Wang tile set is done as follows:

1. A *seed tile* (as shown in Figure 3.28(a)) is added to the tile set. For every clause C_i a *clause tile* (as shown in Figure 3.28(b)) is added to the tile set for every $0 \leq l \leq n$. The clauses are represented by the columns in the rectangle to be assembled. For every variable x_i in the given instance of SAT, add the *valuation tiles* in Figures 3.28(c) and 3.28(d). A column of these tiles is used to represent all the possible valuations. The tile with the color $(C_{m+1}, x_i, 1)$ on its west edge represents valuation $x_i = 1$ and the tile with the color $(C_{m+1}, x_i, 0)$ on its west edge represents valuation $x_i = 0$. Each of the variables is represented by a row in the rectangle.



(a) The seed tile.

(b) A clause tile for clause C_i and $0 \leq l \leq n$.

(c) The valuation tile for $x_i = 1$.

(d) The valuation tile for $x_i = 0$.

Figure 3.28: The auxiliary tiles.

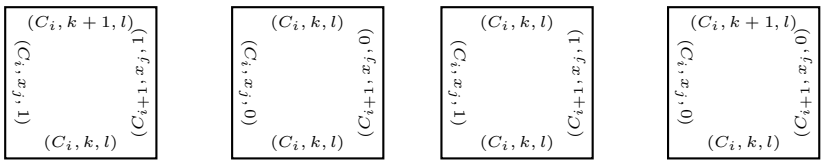
2. For every clause C_i, where $1 \leq i \leq m$, and variable x_j, where $1 \leq j < n$, add only one set of the tiles from figures 3.29(a), 3.29(b) and 3.29(c). For every clause C_i, where $1 \leq i \leq m$, and for the last variable x_n, add only those tiles (from one of the figures 3.29(a), 3.29(b) and 3.29(c)), which have north side colors of form (C_i, k, k), where $0 < k \leq n$. This restriction is set to force the top row to be tiled if, and only if, all the clauses (i.e. columns) have a literal, which is true in the arbitrary valuation given by the valuation tiles in the first column.

The rectangle is constructed so that the tile corresponding variable x_j in clause C_i having color (C_i, k, l) on its south side has color $(C_i, k+1, l)$ on its north side if either $x_j = 1$ and literal x_j belongs to the clause or $x_j = 0$ and negative literal $\neg x_j$ belongs to the clause. In other words, a counter is initiated for every clause and it is incremented on every true literal. Variable $l \leq n$ in Figures 3.28(b), 3.29(a), 3.29(b) and 3.29(c) is used to denote the

final counter value at the top row. It is required later to prove the NP-completeness of the periodic tiling problem and to have the top and bottom row have matching colors.

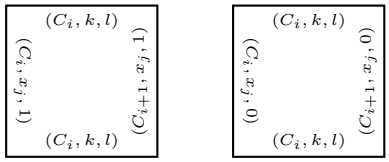
The west side color $(C_i, x_j, 1)$ represents value $x_j = 1$ for the tile in the row that represents clause C_i. Likewise, the color $(C_i, x_j, 0)$ represents value $x_j = 0$.

1. If the positive literal x_j belongs to clause C_i, add the tiles in Figure 3.29(a). Let the south side color be (C_i, k, l). If $x_j = 1$ (i.e. the west side color is $(C_i, x_j, 1)$), the north side color is $(C_i, k+1, l)$. Otherwise the north side color is (C_i, k, l).



(a) The tiles for positive literals x_j in clause C_i.

(b) The tiles for negative literals $\neg x_j$ in clause C_i.



(c) The tiles for literals x_j not in clause C_i.

Figure 3.29: The tiles to represent literals in the given formula.

2. If the negative literal $\neg x_j$ belongs to clause C_i, add the tiles in Figure 3.29(b). Let the south side color be (C_i, k, l). If $x_j = 0$, the north side color is $(C_i, k+1, l)$. Otherwise the north side color is (C_i, k, l).

3. If neither the positive literal x_j nor the negative literal $\neg x_j$ belongs to clause C_i, add the tiles in Figure 3.29(c). These tiles simply move the information on the truth state of the clause upwards. If the south side color is (C_i, k, l), then also the north side color is (C_i, k, l).

Theorem 3.5.1. *The square tiling problem is* NP*-complete even for instances with a 4-way deterministic tile set.*

Proof. It is quite obvious that the tile set constructed above is 4-way deterministic. All the tiles are determined uniquely by any two adjacent sides. The only case that is not as obvious is the uniqueness within the tiles in

Figures 3.29(a), 3.29(b) and 3.29(c). However, for every clause C_i and variable x_j only one of these sets is chosen and within each of these sets the tile is uniquely determined.

It is not necessary to show that a square can be tiled with the given tile set. For every instance of SAT it is possible to add dummy variables that belong to no clause so that the number of variables and the number of clauses are the same. It remains to be shown that an instance of SAT with m clauses and n variables has a solution if, and only if, an $(m+1) \times (n+1)$-rectangle can be tiled with the tile set corresponding to the instance.

It can be seen from the colors of the seed tile that it must be located in the lower left corner of the rectangle. This implies that the valuation tiles must be used to tile the leftmost column and the clause tiles to tile the bottom row. This implies further that the rest of the rectangle can be tiled correctly if, and only if, the given instance of SAT has a solution.

The reduction from the given instance of SAT to a 4-way deterministic tile set can clearly be done in polynomial time. Hence, the square tiling problem is NP-complete for 4-way deterministic tile sets. □

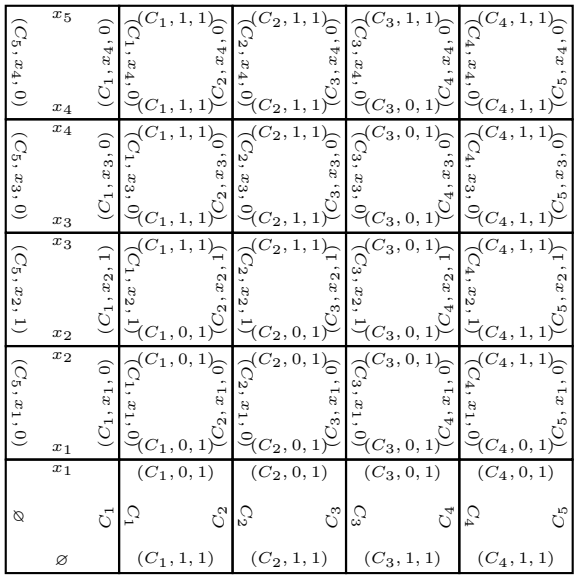


Figure 3.30: A valid tiling of a square with values $x_1 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_4 = 0$ for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$.

Figures 3.5.2 and 3.5.2 present examples of a valid tiling of a rectangle and an incomplete tiling of a rectangle. In Figure 3.5.2 the valuation (i.e. the leftmost column) is such that the rectangle can be completed. In Figure

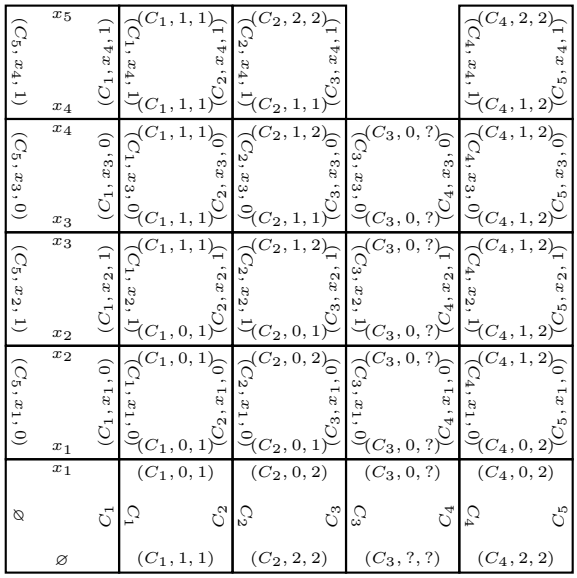


Figure 3.31: An incomplete tiling of a square with values $x_1 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_4 = 1$ for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$. There is no matching tile for the negative literal $\neg x_4$ in the third clause.

3.5.2 the valuation is such that the column representing the third clause cannot be tiled.

Theorem 3.5.2. *The periodic tiling problem is* NP*-complete even for instances with a 4-way deterministic tile set.*

Proof. A new tile set is constructed by modifying the previous one. The south side color of the seed tile in Figure 3.28(a) is changed to x_{n+1} (where n is the number of variables in the SAT instance) and west side of the seed tile is changed to C_{m+1}. This modified tile set admits a periodic tiling by $(m+1) \times (n+1)$-rectangles if, and only if, the given instance of SAT has a solution. □

One can improve the results concerning the square tiling problem (and the periodic tiling problem) with 4-way deterministic tile sets so that the square (and the rectangular period) can always be tiled correctly except for the top right corner. This can be seen by adding an additional row on top of the rectangle in a similar way as was done in [2]. With the new tile set, this "check row" is used to count the number of north side colors (C_i, k, k) (where $k > 0$) in the row below it. Then the rightmost tile in the check row can be tiled if, and only if, the number of true clauses equals the number of

clauses (and if value l can be chosen correctly for each column). Formally, the new tiles for the check row are the ones in Figure 3.32.

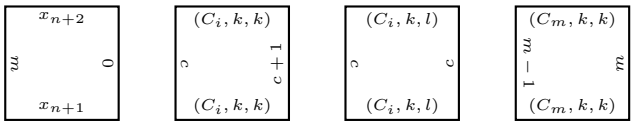


Figure 3.32: The check tiles. Expression m denotes the number of clauses, n the number of variables, $l \neq k$ is an integer and c, $0 \leq c \leq m-2$, is the counter for the "correct" north side colors (C_i, k, k), where $k > 0$.

3.5.3 The unique shape problem of self-assembly

By constructing the tile set as previously but setting no constraints for the north side colors (C_i, k, l) of the top row and adding the tiles for the check row, one has, again, a tile set construction reducing the SAT to the square tiling problem. Now however, the square can always be tiled correctly except for the top right corner. Knowing this, one can also conclude that a special case of the unique shape problem is co-NP-complete even with 4-way deterministic tile sets.

Theorem 3.5.3. *The unique shape problem is* co-NP*-complete with 4-way deterministic tile sets in the unique shape model.*

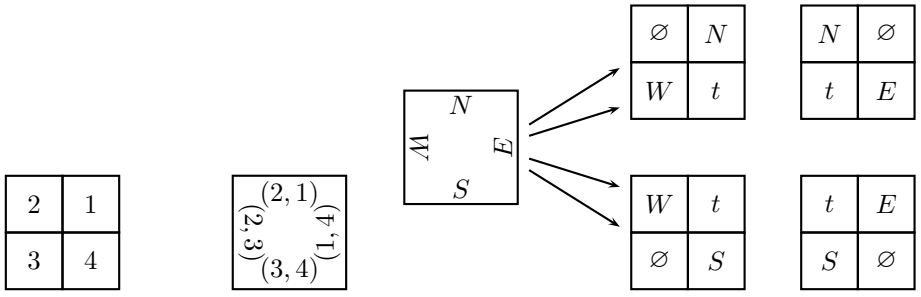
Proof. The problem is seen to be in co-NP as before in [2]. If a terminal tile cluster of a shape not contained in S or a (possibly nonterminal) tile cluster of size $|S|+1$ can be produced then the problem has a negative solution. Then the sequence (with size $\mathcal{O}(|S|^2)$) of nonterminal tile clusters leading to a tile cluster of the alternative shape can be considered as the polynomial time proof for the negative answer. For every step it can be checked, whether or not the new tile was added correctly.

The tile set for the reduction is the one of Theorem 3.5.1 with the tiles in Figure 3.32. The seed tile is the tile in Figure 3.28(a) and the temperature is $\tau = 2$. The glue function g will be defined to force the leftmost column and the bottom row to be assembled always by setting $g(C_i, C_i) = 2$, $g(x_j, x_j) = 2$, $g(a, a) = 1$ if $a \neq C_i, x_j$ and $g(a, b) = 0$ if $a \neq b$. Now rest of the rectangular area will be assembled one tile at a time towards the top right corner. Also, the assembly will always continue uninterrupted (at least) until the top right corner. If the given tile set uniquely assembles into a rectangle with its top right corner missing (i.e. the given shape S), then the given instance of SAT does not have a solution. Else, a complete

rectangle is produced and the given instance of SAT has at least one solution. Furthermore, with this construction, no mismatching colors occur in the assembled tile clusters. □

3.5.4 NP-completeness with determinism by opposite edges

A *corner tile* is a unit square, which is divided into four colored corners as in Figure 3.33(a). Corner tiles are also Wang tiles, since the color of an edge is determined by corner colors of that particular edge in the sense of Figure 3.33(a). The notions concerning determinism can therefore be extended to corner tile sets also.



(a) A corner tile and the corresponding Wang tile.

(b) From Wang tiles to corner tiles. Color t is a unique new color representing the old Wang tile

Figure 3.33: Corner tiles and Wang tiles.

Every Wang tile set can be converted into a corner tile set, which admits a valid tiling of a square if, and only if, the original Wang tile set admits a valid tiling of a square. This can be done by dividing each Wang tile into four corner tiles (as quadrants) as in Figure 3.33(b). The corners are colored so that the neighboring corner tiles match if, and only if, they are parts of the same original Wang tile or their corresponding original Wang tiles match. It is done simply by introducing a new color uniquely identifying the original tile and coloring the corners according to the side colors of the original tile. The technique of Figure 3.33(b) has been used earlier to transform Wang tile sets into corner tile sets [60].

Lemma 3.5.4. *If the given Wang tile set T is XY-deterministic, then the corner tile set (considered as a Wang tile set in the sense of figure 3.33(a)), which is constructed from set T by the operation in Figure 3.33(b), is also XY-deterministic.*

Proof. A corner tile set is always deterministic with respect to any two opposite sides and therefore it is enough to consider only determinism by

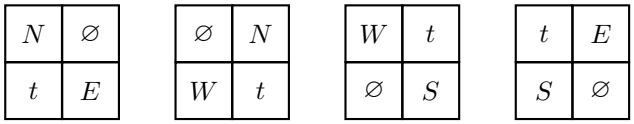


Figure 3.34: The new corner tiles for Wang tile t.

adjacent sides. Assume that the given tile set is SW-deterministic. The first one of the new tiles in Figure 3.34 is uniquely defined by both its south side and its west side, since no other corner tile has color t in its SW-corner. The second tile is uniquely defined by its south side, since no other corner tile has color t in its SE-corner. The third tile is uniquely defined by its south side and west side, since the original Wang tile set was assumed to be SW-deterministic. The fourth tile is uniquely defined by its west side, since no other corner tile has color t in its NW-corner. Argumentation for NW-, NE- and SE-determinism is similar. □

Theorem 3.5.5. *The problems of Theorems 3.5.1 and 3.5.2 and the complement problem of Theorem 3.5.3 remain* NP*-complete even if the given tile set is deterministic with respect to any two sides.*

For the square tiling problem and the (bounded) periodic tiling problem it would have been enough to consider 2×2 tile set formed of the original tile set to achieve the result of Theorem 3.5.5. However, the problem of Theorem 3.5.3 assumes a unique seed tile which is given in advance. Applying the 2×2 tile set construction to a seed tile and its neighbors produces multiple new seed tiles.

Proof. A tile set that is deterministic by any two sides can be constructed by applying the operation depicted in Figure 3.33(b) to the tile set of a particular theorem. This new tile set can tile a rectangle of twice the original height and width if, and only if, the given instance of SAT with m clauses and n variables has a positive solution. For the problem of Theorem 3.5.3, the temperature is chosen to be $\tau = 2$ and the glue function is defined to have value 2 between the equal colors of the leftmost column and the bottom row. The glue function has value 1 for other equal color pairs and value 0 for unequal color pairs. The co-NP-completeness (for the rectangular shapes with the four tiles of the original top right corner missing) follows as before. □

Chapter 4

On 2-way deterministic tile sets

In this chapter it is shown that the tiling problem of Wang tiles remains undecidable even when the instances are restricted to 2-way deterministic tile sets. This is a weaker result than that of Chapter 3. However, the 4-way deterministic construction is unnecessarily complex with respect to the further use in Chapter 5. In particular, to represent a Turing machine computation with a 4-way deterministic tile set, it would seem that the complicated diagonal line construction of Section 3.2 is necessary. However, a cellular automaton can be expansive without being based on a 4-way deterministic tile set and for that reason 4-way determinism is not required of the tile sets in Chapter 5. Also, the 2-way deterministic tile set construction can easily be converted into a reversible cellular automaton which can simulate an irreversible Turing machine unlike the 4-way deterministic construction. So, for a reader who is interested on cellular automata only, this chapter is meant to provide, is simplified proof of the results required in Chapter 5 without unnecessarily complicated constructions.

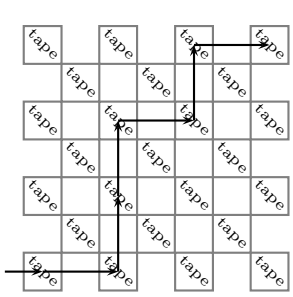
4.1 The tiling problem with a seed tile

It is known that the tiling problem is undecidable in the 4-way deterministic case with and without a seed tile. However, in the 4-way deterministic case the construction is unnecessarily complicated in terms of cellular automata. Therefore, it is shown in this section that the tiling problem with a seed tile is undecidable for 2-way deterministic tile sets. The construction is much more simple and it provides the same undecidability results with respect to cellular automata while the argumentation is more readable. Furthermore, the 2-way deterministic construction gives another proof for the fact that any Turing machine can be simulated with a reversible one-dimensional cel-

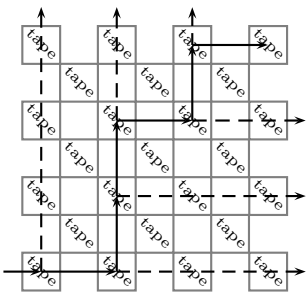
lular automaton, which could not be achieved with the 4-way deterministic construction.

4.1.1 The idea for the undecidability proof

The basic idea is to represent the Turing machine tape on diagonal rows as in [46]. It is easy to show that an arbitrary Turing machine computation can be represented on diagonal rows. The computation on diagonal rows is done in the manner of Figure 4.1(a). Every second diagonal row in the northwest-southeast direction is used to represent the Turing machine configuration at a certain moment. One tile at each diagonal row represents the read-write head and the current letter to be read. The other tiles of the diagonal row represent the other letters on the tape located to the left and to the right from the read-write head.



(a) The rough idea of representing Turing machine computation on diagonal rows.



(b) The Turing machine computation with additional information signals of earlier read-write operations.

Figure 4.1: The general idea of representing the computation on diagonal rows.

Since a Turing machine is a deterministic method of computation, the tile set constructed in this manner is clearly deterministic in (at least) one direction. More specifically, it is the direction to which the computation advances in time. To force determinism also in the opposite direction, some modifications are needed. On every operation of the read-write head, a "signal" is sent to the direction that is opposite to the read-write head movement. This signal contains information about the read-write operation which is currently being conducted and the direction from which the read-write head entered the current cell after the previous move. The computation with signals is represented in Figure 4.1(b). If the read-write head moves to the left, then the signal is sent towards east, and if the read-write head moves to the

right, then the signal is sent towards north. In practice, the signal is just a component of a side color which moves onward unobstructed. These signals containing information about the previous move and the current one are referred to as the *move signals*. The move signals are started on the tiles in Figures 4.2 and 4.3 (i.e. the tiles that represent the read-write head). The tiles in Figure 4.5 (i.e. the tiles that represent the tape) just move the possible move signals onward. This construction will make the tile set representing the given Turing machine 2-way deterministic.

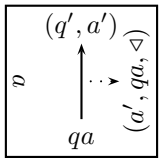
4.1.2 The tile set for the given Turing machine

In this subsection a 2-way deterministic tile set is constructed for the given Turing machine. In what follows, the diagonal rows of tiles are referred to as *diagonals* in short.

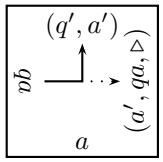
The tiles to represent read-write operations For every possible move of the Turing machine, either the tiles in Figure 4.2 and the tile in Figure 4.4(a), or the tiles in Figure 4.3 and the tile in Figure 4.4(b) are added to the tile set.

The tiles for a left move Assume that the Turing machine contains move $\delta(q, a) = (q', a', \triangleleft)$. Then the tiles in Figure 4.2 and the tile in Figure 4.4(a) are added to the tile set.

The tile in Figure 4.2(a) is used if the previous move was to the left and the current move is to the left. If the previous move was to the right, then the tile in Figure 4.2(b) is used.



(a) The new move is to the left and the previous move was to the left.

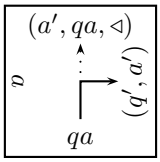


(b) The new move is to the left and the previous move was to the right.

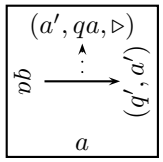
Figure 4.2: Action tiles for move $\delta(q, a) = (q', a', \triangleleft)$

The tiles for a right move Assume that the Turing machine contains move $\delta(q, a) = (q', a', \triangleright)$. Then the tiles in Figure 4.3 and the tile in Figure 4.4(b) are added to the tile set.

The tile in Figure 4.3(a) is used if the previous move was to the left and the current move is to the right. If the previous move

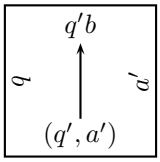


(a) The new move is to the right and the previous move was to the left.

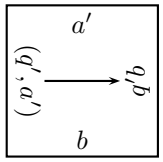


(b) The new move is to the right and the previous move was to the right.

Figure 4.3: Action tiles for move $\delta(q, a) = (q', a', \triangleright)$



(a) The tile for read-write operation $\delta(q, a) = (q', a', \triangleleft)$.



(b) The tile for read-write operation $\delta(q, a) = (q', a', \triangleright)$.

Figure 4.4: Merging tiles. The tiles depend on the new state q', the new letter a' to be written, the move direction and on the new letter b to be read.

was to the right and the current move is to the right, then the tile in Figure 4.3(b) is used.

The tiles in Figures 4.2 and 4.3 will be called *action tiles* and the tiles in Figures 4.4 will be called *merging tiles*. Together these tile are referred to as *move tiles* or the tile set $M_{\mathcal{M}}$.

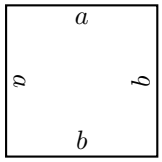
The tiles to represent tape contents For every state q and every element a, b and c of the tape alphabet, the tiles in Figure 4.5 are added to the tile set. The tile in Figure 4.5(a) is used to represent a cell (or the border between two cells if $a \neq b$) of the tape without any information about an earlier read-write operation.

The tiles in Figure 4.5(b) represent tape contents likewise, but contain also information about a read-write operation during which the read-write head moved to the left. That is, the east side and the west side have colors of form $(\cdot, qc, \cdot)$ if, and only if, there exist a move of form $\delta(q, c) = (\cdot, \cdot, \triangleleft)$.

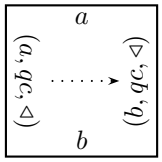
The tiles in Figure 4.5(c) are similar to the tiles in Figure 4.5(b) with the exception that they contain information about a read-write oper-

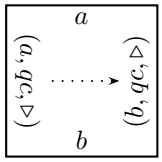
ation during which the read-write head moved to the right and not to the left. The north side and the south side have colors of form $(\cdot, qc, \cdot)$ if, and only if, there exist a move of form $\delta(q, c) = (\cdot, \cdot, \triangleright)$.

The tile set is being constructed so, that if the seed tile (i.e. the tile in Figure 4.6(c)) is located on an even diagonal, then on every odd diagonal expressions a and b in Figure 4.5 are equal.

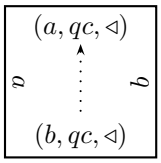


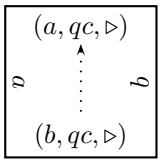
(a) A tile without read-write information.





(b) The tiles with information about a move to the left.





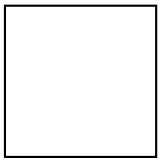
(c) The tiles with information about a move to the right.

Figure 4.5: The tiles to represents the letters on the tape. Here q denotes an arbitrary state and letters a, b and c denote arbitrary elements of the tape alphabet.

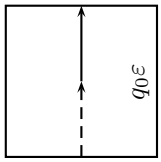
The tiles that are used to represent the tape contents of the given Turing machine $\mathcal{M}$ are referred to as *alphabet tiles* or as the tile set $A_{\mathcal{M}}$.

The starting tiles To force the Turing machine to start on a blank tape only, the tiles in Figure 4.6 are added to the tile set. One of these tiles (namely, the tile in Figure 4.6(c)) is chosen to be the seed tile. If the seed tile is contained within a tiling, then the tiling represents a Turing machine computation. Other tiles in Figure 4.6 force the Turing machine to start on a blank tape. The blank initial configuration of the Turing machine is represented by the tile pattern shown in Figure 4.8. In short, if the seed tile is located in the origin, then the Turing machine simulation is done in the first quadrant.

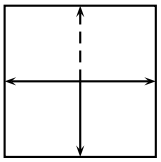
For the given Turing machine $\mathcal{M}$, the tiles in Figure 4.6 are referred to as *starting tiles* or as the tile set $S_{\mathcal{M}}$.



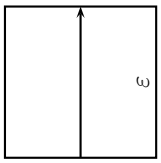
(a) The blank tile.



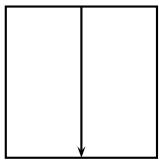
(b) The tile to place the read-write head.



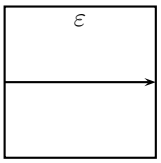
(c) The seed tile.



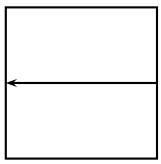
(d) The west border.



(e) The west border extension.



(f) The south border.



(g) The south border extension.

Figure 4.6: Starting tiles.

For every Turing machine $\mathcal{M}$, the tile set constructed using the method above is denoted by $T_{\mathcal{M}}$, that is,

$$T_{\mathcal{M}} = M_{\mathcal{M}} \cup A_{\mathcal{M}} \cup S_{\mathcal{M}}.$$

An example of a Turing machine operation is shown in Figure 4.7.

Let (q, a) be any preimage pair for which the transition $\delta(q, a)$ is not defined. Then there will be no tile that would have the color qa on its west side or south side. Therefore, if the Turing machine eventually halts, that is, if at some moment of time the read-write head in state q reads letter a, then the tiling cannot be completed to cover the entire plane in a valid way.

Lemma 4.1.1. *For any given Turing machine $\mathcal{M}$, the tile set $T_{\mathcal{M}}$ is both NE- and SW-deterministic.*

Proof. The tile set is SW-deterministic, since clearly it has no two tiles having same colors on the south side and the west side.

Similarly, the tile set is NE-deterministic. No two tiles in Figures 4.2, 4.4, 4.3 and 4.5 have the same colors on the north side and the east side. □

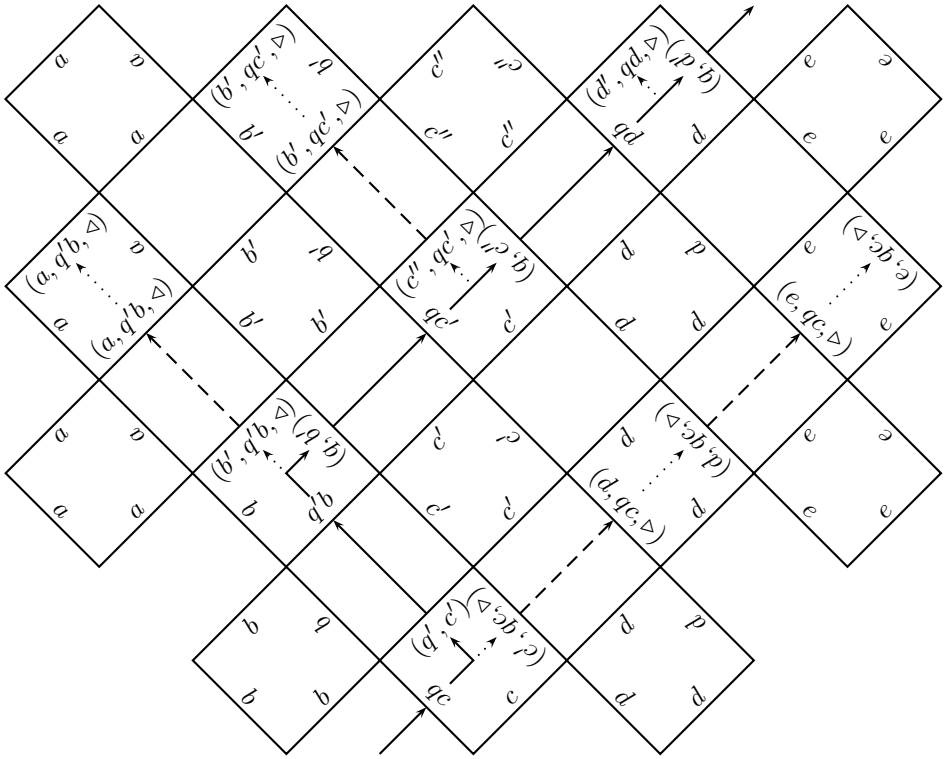


Figure 4.7: Rewrite operation $abqcd \vdash aq'bc'de \vdash ab'qc'de \vdash ab'c''qde$. The computation advances upwards, that is, towards northeast. For brevity, not all tiles are drawn completely.

Theorem 4.1.2. *The following question is undecidable: "Given a Turing machine $\mathcal{M}$, does the tile set $T_{\mathcal{M}}$ admit a valid tiling of the plane containing the tile in Figure 4.6(c)?"*

Proof. The tile set $T_{\mathcal{M}}$ quite obviously corresponds the actions and configurations of the given Turing machine $\mathcal{M}$. Requiring the seed tile to be the tile in Figure 4.6(c), the structure in Figure 4.8 is forced to be tiled on the plane.

The structure in Figure 4.8 obviously corresponds the initial configuration with a blank tape. Therefore, the plane can be tiled correctly if, and only if, the given Turing machine does not eventually halt (when started on a blank tape). The halting problem with a blank tape is undecidable. □

Since the tiling problem with a seed tile is a generalization of the problem in Theorem 4.1.2, it is seen that the tiling problem with a seed tile is undecidable for tile sets that are 2-way deterministic. Moreover, the tile set

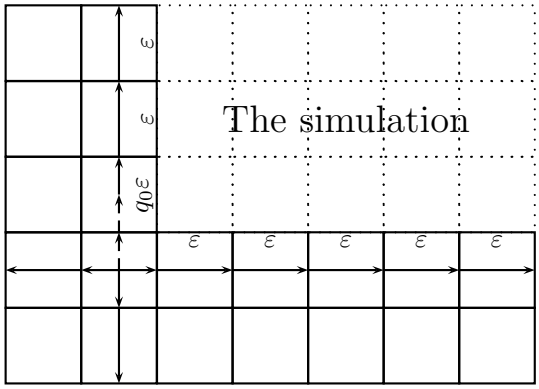


Figure 4.8: Using the tiles in Figure 4.6 to start the Turing machine simulation on a blank tape.

$T_{\mathcal{M}}$ would be NE-deterministic even if the Turing machine $\mathcal{M}$ was nondeterministic. No matter what the state q and letter a are, the tiles in Figures 4.2, 4.4 and 4.3 are uniquely defined by the colors of their north and east sides.

A 2-way deterministic tile set can be constructed even for any nondeterministic Turing machine. This tile set is constructed by modifying the tile set $T_{\mathcal{M}}$. Modification is based on using signals containing information about the particular move that was chosen. These signals could be referred to as *decision signals*. The tile in Figure 4.2 is modified so, that it sends a decision signal to the left and backwards in time (i.e. towards west since the computation advances towards northeast). Likewise, the tile in Figure 4.3 is modified to send a decision signal to the right and backwards in time (i.e. towards south). Furthermore, the tiles in Figures 4.5 and 4.6 are modified to allow crossings with any kinds of decision signals. It is quite straightforward to see, that the new modified tile set is indeed both NE- and SW-deterministic.

4.2 The tiling problem without a seed tile

In this section the tiling problem without a seed tile is shown to be undecidable for those tile sets that are deterministic by two opposite corners. The reduction procedure is almost identical to that of Section 3.4. That is, the tile set which simulates the Turing machine computation is much simpler and the complex construction of Sections 3.2 and 3.3 can be skipped entirely by using the tile set of Section 4.1.2.

The new tile set is constructed in four layers for the given tile set T and a seed tile $t \in T$. The outline of the layers is the following (as in Section 3.4):

Layer 1. The tiling forced by the 3×3 tile set (see the definition in Section 2.3) of the aperiodic tile set of Kari and Papasoglu. (Section 3.4.2)

Layer 2. The tiles to identify free areas. (Section 3.4.3)

Layer 3. A tiling simulating a tiling by the given tile set T. (Section 3.4.4)

Layer 4. The tiles to forward the colors of the tile set T on layer 3 from a free area to another free area border and from a red border to another red border without discarding any color information. (Section 3.4.5)

Theorem 4.2.1. *The tiling problem is undecidable for tile sets that are 2-way deterministic.*

4.3 Tile sets and cellular automata

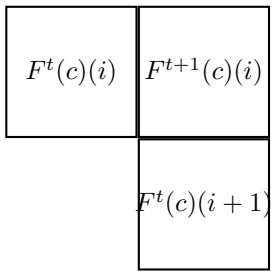
In this section it is reviewed how a tile set can be interpreted as a cellular automaton and how a computation by an irreversible Turing machine can be simulated with a reversible cellular automaton.

4.3.1 Interpreting tile set as a cellular automaton

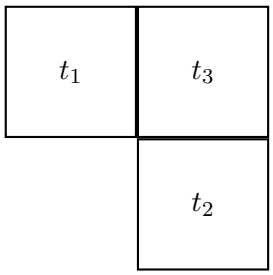
Following the presentation in [46], it is possible to regard Wang tile sets (that are deterministic at least in one direction) as one-dimensional cellular automata.

If the given tile set is, say, SW-deterministic, it is possible to consider the tiles as states of a cellular automaton. As shown in Figure 4.9, with a cellular automaton the next state of a cell is determined with a similar manner as the next tile (to the northeast) in a tiling with a SW-deterministic tile set. With a cellular automaton the new state depends on the old states and in a tiling (with a SW-deterministic tile set) the new tile is determined by the colors of its neighbors.

It should be noted, that the given Wang tile set may not contain all the possible color pairs in the southwest corners of the tiles. If the given tile set T is assumed to be deterministic in only one direction, say, by the southwest corner, it is enough to add a tile to the original tile set for every missing southwest corner color pair. For example, if there is no tile t with $t_W = x$ and $t_S = y$ in the given tile set T, a tile t with $t_N = t_E = z$, $t_W = x$ and



(a) States $c(i)$ and $c(i+1)$ determine the next state of the cell $c(i)$.



(b) Tiles t_1 and t_2 determine tile t_3 in a tiling by a SW-deterministic tile set.

Figure 4.9: The tiles of a SW-deterministic tile set can be considered as states of a cellular automaton.

$t_S = y$, where z is any color of the tile set T, could be added to the tile set while maintaining SW-determinism.

If the given tile set is assumed to be both NE- and SW-deterministic, equally many color pairs are missing in the northeast corners and the southwest corners. It is trivial to construct (for example, by some ordering method) a one-to-one correspondence between the missing colors in the southwest corners and the missing colors of the northeast corners. This bijection can clearly be considered as a NE- and SW-deterministic set of tiles. Moreover, the union of the initial tile set and this new tile set is both NE- and SW-deterministic tile set containing N^2 tiles, where N is the number of colors in the original tile set. Let $T^{\complement}$ denote the new tile set which was constructed for the missing color pairs. Now it can be seen that the tile set $T \cup T^{\complement}$ can be considered as a reversible cellular automaton.

A bit more formally, for the NE- and SW-deterministic tile set T, a reversible cellular automaton $(T \cup T^{\complement}, F_T)$ simulates the tiling procedure. The global rule F_T is defined using local rule

$$f_T(x, y) = z \quad \text{if } x, y \in T \cup T^{\complement}, x_E = z_W \text{ and } y_N = z_S.$$

The function $f_T : (T \cup T^{\complement})^2 \to T \cup T^{\complement}$ is total and well-defined, since the tile set $T \cup T^{\complement}$ is both NE- and SW-deterministic.

4.3.2 Universality of one-dimensional reversible cellular automata

It has been shown by Morita and Harao that reversible one-dimensional cellular automata are computationally universal [82]. More precisely, they have shown that any reversible Turing machine can be simulated with some

reversible one-dimensional cellular automaton. Since any Turing machine can be simulated with a reversible Turing machine [6], the universality of one-dimensional reversible cellular automata follows.

However, the requirement of reversibility for the given Turing machine is not necessary for the machine to be simulated with a reversible one-dimensional cellular automaton. In fact, Dubacq has given a construction for a family of reversible cellular automata to simulate any irreversible Turing machine in real time [27]. Dubacq's approach was more from the cellular automata point of view. The construction of the family of tile sets given in Section 4.1.2 gives a tiling view for Dubacq's result. Namely, a Turing machine configuration can be represented by a cellular automaton configuration where exactly one of the cells contains a move tile representing the read-write head and all the rest of the cells have alphabet tiles as states. The cellular automaton executes one Turing machine computation step in two time steps. If the initial configuration c of the cellular automaton (with global rule F) represented valid Turing machine configuration, so does $F^{2t}(c)$. The additional move signals can be ignored and the configuration of the Turing machine computation can be read directly.

Theorem 4.3.1 ([27]). *Any Turing machine can be simulated using a reversible one-dimensional cellular automaton in real time.*

Proof. A given Turing machine can be simulated with the cellular automaton $(T \cup T^{\complement}, F_T)$, where T is the tile set constructed in Section 4.1.2. Because for every computation step of the Turing machine the cellular automaton $(T \cup T^{\complement}, F_T)$ conducts two computation steps, the cellular automaton with global rule F_T^2 simulates the given Turing machine in real time. □

Corollary 4.3.2 ([82]). *Reversible one-dimensional cellular automata are computationally universal.*

Chapter 5

Undecidability results regarding expansivity

In this chapter it is shown that left expansivity and right expansivity (which are weaker properties than expansivity) are undecidable properties. The result follows from the facts that tiling problem is undecidable for tile sets that are at least 2-way deterministic.

5.1 Some technical definitions and results

A cellular automaton $(A^{\mathbb{Z}}, F)$ is said to be *globally immortal* with respect to subset $B \subseteq A$ if there exists a configuration $c \in B^{\mathbb{Z}}$ such that $F^n(c)(i) \in B$, for all integers $n \in \mathbb{N}$, $i \in \mathbb{Z}$. The following decision problem is referred to as the *global immortality problem*: "Given a cellular automaton $(A^{\mathbb{Z}}, F)$ and subset $B \subseteq A$, is $(A^{\mathbb{Z}}, F)$ globally immortal with respect to B?"

A cellular automaton is said to be *locally immortal* with respect to subset $B \subseteq A$ if there exists a configuration $c \in B^{\mathbb{Z}}$ such that $F^n(c)(0) \in B$, for all integers $n \in \mathbb{N}$. The following decision problem is referred to as the *local immortality problem*: "Given a cellular automaton $(A^{\mathbb{Z}}, F)$ and subset $B \subseteq A$, is $(A^{\mathbb{Z}}, F)$ locally immortal with respect to B?"

These definitions of "immortality" are motivated by the immortality problem of Turing machines [39]. Hooper has shown that it is undecidable if a given Turing machine eventually halts on every extended configuration (i.e. the tape may contain infinitely many non-blank cells).

The problems of global immortality and local immortality are slightly related to the nilpotency. In a nilpotent cellular automaton every cell enters to a quiescent state q within a bounded number of time steps. In the immortality problems the elements of $A \setminus B$ correspond to the quiescent state of a nilpotent cellular automaton with the distinction that a cell does not

have to remain in a state of $A \setminus B$. With this difference the immortality problems make sense for reversible cellular automata also.

Theorem 5.1.1. *The global immortality problem is undecidable for reversible one-dimensional cellular automata.*

Proof. Undecidability of the question follows by a reduction from the problem of Theorem 4.2.1 (or Theorem 3.4.1). Let $(T \cup T^{\complement}, F_T)$ be the cellular automaton constructed in Section 4.3.1 and let $B = T$ in the definition of the problem.

Assume first, that the given tile set T admits a valid tiling. Then one can choose any northwest-southeast diagonal row of tiles of the valid tiling to be the configuration c. Since the tile set is NE-deterministic and configuration c is part of a valid tiling, $F_T^n(c)(i) \in T$, for all integers $n, i \in \mathbb{Z}$.

Assume second, that the given tile set T does not admit a valid tiling. If for some configuration c (considered again as a northwest-southeast diagonal row of a valid tiling) the condition of the problem did hold, then it would be possible to construct a valid tiling. However, this contradicts the assumption.

Hence, a configuration c exists if, and only if, the tile set T admits a valid tiling. □

Any surjective cellular automaton with global rule F and radius r can be converted into a mixing cellular automaton by forming a new global rule which is a composition of the original global rule and a sufficiently high power of the shift function σ. For example, composition $F \circ \sigma^{r+1} : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ is a mixing cellular automaton. The idea is illustrated better in Figure 5.1. By applying a shift map, the preimage of the requested pattern is moved away from the origin.

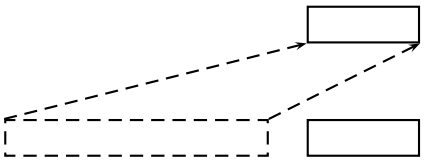


Figure 5.1: Combining the global rule with a shift function causes the preimages of two different patterns to be transferred to disjoint locations.

Lemma 5.1.2. *Let $(A^{\mathbb{Z}}, F)$ be a surjective cellular automaton with radius r. Then $(A^{\mathbb{Z}}, F \circ \sigma^{r+1})$ is topologically mixing.*

Lemma 5.1.3. *Let $(A^{\mathbb{Z}}, F)$ be an injective cellular automaton with radius r_1 and let its inverse local rule have radius r_2. Then $(A^{\mathbb{Z}}, F \circ \sigma^{r+1})$ is expansive where $r = \max(r_1, r_2)$.*

Corollary 5.1.4. *The global immortality problem is undecidable for reversible one-dimensional cellular automata that are both expansive and topologically mixing.*

Proof. The property in question is shift-invariant. Therefore, by using Lemma 5.1.2 and Lemma 5.1.3 the claim follows from Theorem 5.1.1. □

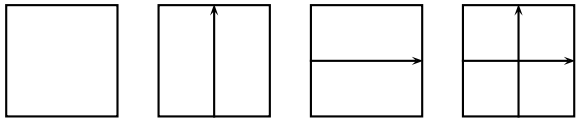
Theorem 5.1.5. *The local immortality problem is undecidable for reversible one-dimensional cellular automata that are left expansive.*

Proof. Let T_1 be the tile set in Fig. 5.2(a) and let T_2 be the tile set in Fig. 5.2(b). Let t_b be the blank tile in set T_1. Form a new sandwhich tile set

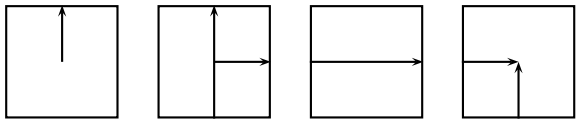
$$A = (T \times T_1) \cup (T^{\complement} \times T_2)$$

and let $B = T \times \{t_b\}$.

This tile set can also be considered as a reversible cellular automaton because all the possible color pairs occur both at northeast and southwest corners. The cellular automaton simulates tiling on two layers with some additional constraints. On the first layer are tiles $T \cup T^{\complement}$ and on the second layer are tiles $T_1 \cup T_2$. Let G be the global rule of this cellular automaton.



(a) Tile set T_1.



(b) Tile set T_2.

Figure 5.2: New tiles to pair with the original tile set.

Let $c \in B^{\mathbb{Z}}$, that is, let c be a configuration consisting of states $T \times \{t_b\}$ only. If c is diagonal row of tiles in a valid tiling, then $G^n(c) \in B^{\mathbb{Z}}$ for every $n \in \mathbb{Z}$. If T does not admit a valid tiling then for every c there exists such integers $n \in \mathbb{Z}$ and $i > 0$ that $G^n(c)(i) \in T^{\complement} \times T_2$. (If such integer $i > 0$ did not exist, a valid tiling would exist.) Due to the structure of tile sets T_1 and T_2, once the "arrow signal" is generated using the tiles of T_2, it is always forwarded at least to the left (i.e. northwest) and either forward (i.e. north) or backward (i.e. west) in time without ever being cancelled. That is, if an arrow signal is present in a cell i at some moment, then the signal will be present in every cell j, where $j < i$, at some moment. This implies

that $G^n(c)(0) \in A \setminus B$ for some $n \in \mathbb{Z}$. Hence $G^n(c)(0) \in B$ for every $n \in \mathbb{Z}$ if, and only if, T admits a valid tiling. This would conclude the proof for reversible cellular automata.

For left expansive cellular automata, the rule G is modified so that underlying tiling rule of state components $T \cup T^{\complement}$ is combined with the shift function σ. Let this new global rule be F. This converts the tiling procedure by $T \cup T^{\complement}$ into an expansive cellular automaton. The second rule on the second layer is not modified because tile set T_1 defines an expansive cellular automaton and tile set T_2 defines a left expansive cellular automaton. Therefore using either tiles T_1 or T_2 on predefined locations atop tiles $T \cup T^{\complement}$ defines a left expansive cellular automaton. □

It is not known whether the local immortality problem is decidable or undecidable for expansive cellular automata. If the problem was an undecidable for expansive cellular automata, then also expansivity would be an undecidable property. This would follow by applying the same construction as later in the proof of Theorem 5.2.1.

Theorem 5.1.6. *The local immortality problem is undecidable for reversible one-dimensional cellular automata that are both left expansive and topologically mixing.*

Theorem 5.1.6 will be proved by modifying the cellular automaton in the proof of Theorem 5.1.5 so that it becomes mixing.

The only reason, why the cellular automaton is not already mixing, is that the signals may cause dependence between two configurations in the same computation. That is, configurations of two different cylinders might not appear in the same orbit because the contents of one cylinder might cause some undesired pattern of signals (of the ones generated by T_1 and T_2) appear throughout the computation near the origin. For the given two open set U and V, this might prevent from finding a configuration $c \in U$ so that $F^k(c) \in V$ for some k (i.e. the cellular automaton would not be even transitive). The problem is illustrated in Figure 5.3.

Unfortunately, the shift function cannot be applied to the signals of tile sets T_1 and T_2 because then a row of tiles without the signals might appear even if there was tiling error (i.e. a state from $T^{\complement} \times T_2$) in the tiling represented by the computation. Combining the original rule with shift function would make the cellular automaton mixing but the problem might not be undecidable anymore. Therefore, the signals in tiles T_1 and T_2 need to be modified.

Proof (sketch). Let $(T \cup T^{\complement}, G)$ be the cellular automaton defined by the 2-way deterministic tile set T. It can be assumed that $(T \cup T^{\complement}, G \circ \sigma^{r+1})$ is expansive and mixing for some positive integer r. The theorem will be proved by a reduction from the problem of Corollary 5.1.4.

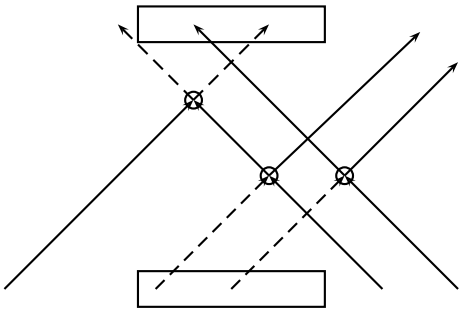


Figure 5.3: The signal generated by the tile set T_2 may cause dependence between cells of two configurations of the same orbit in the proximity of origin. Solid lines represent the signals in the tiles. Dashed lines represent absence of the signals. Circles denote tiling errors (where T_2 is used). The rectangles represent domains defining the different cylinders.

The state set $T \cup T^{\complement}$ and the rule $G \circ \sigma^{r+1}$ is modified by adding three different kinds of signals. The first set of signals consists of signals that move with a speed of one cell per time step either from left to right or right to left. For the sake of the argument, let these signals be called *type 1 signals*. The second set consists of signals that move with a speed of one cell per two time steps either from left to right or right to left. These signals will be called *type 2 signals*.

The third set of signals consists of signals that move with a speed of one cell per time step either from left to right or right to left. These signals will be called *type 3 signals* and they correspond the signals of tile sets T_1 and T_2. If a tiling error occurs in the underlying tiling, the type 3 signals in the same cell are modified as in tiles T_2 if, and only if, there is no type 1 and type 2 signal present in the cell. That is, type 1 and type 2 signals are used to "cancel out" the effects of tiling errors on type 3 signals. Signals of type 1 and 2 are not altered at any point. Let the new global rule be denoted by F.

Given two radius r cylinders $C_1 = \mathrm{Cyl}(c_1, r)$ and $C_2 = \mathrm{Cyl}(c_2, r)$, it is shown in Figure 5.4 how type 1 and 2 signals can be used to cancel out tiling errors everywhere else except in the immediate vicinity of the origin. Type 1 and 2 signal from different sides of the origin coincide after $t_0 = 2r + 2$ time steps. Therefore, configurations belonging to any two cylinders can appear in the same trajectory for every $t \geq \max(t_0, n_0)$ time steps apart, where n_0 is the bound given implicitly by Lemma 5.1.2. Hence, cellular automaton F is mixing. Also, adding signals type 1, 2 and 3 does not remove left expansivity.

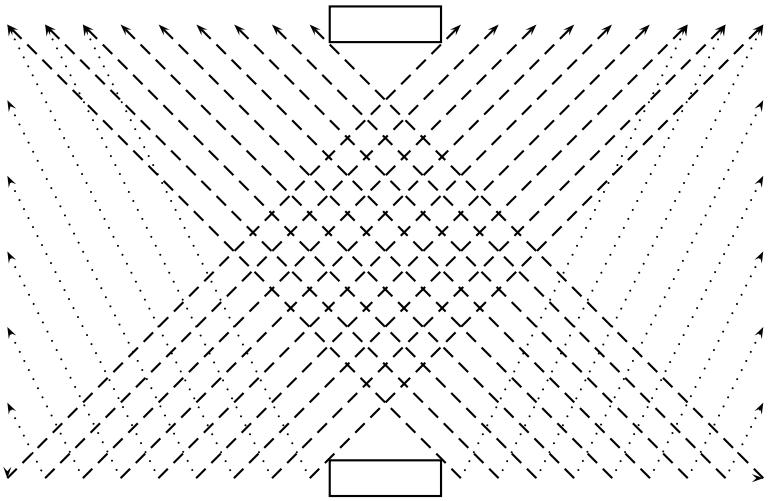


Figure 5.4: If two configurations are sufficiently far apart in the same orbit, type 1 and 2 signals can be used to cancel out tiling errors that would cause constraints between the cylinders of the two configurations. Dashed lines denote type 1 signals, dotted lines denote type 2 signals and the rectangles denote the cell state sequences defining the cylinders.

It is now undecidable whether or not there exists a configuration c such that state $F^n(c)(0)$ contains none of the signals type 1, 2 or 3 for any integer n. □

Interpreting the definition of local immortality problem in a different way, Corollary 5.1.7 follows.

Corollary 5.1.7. *Given a left expansive and topologically mixing reversible cellular automaton $(A^{\mathbb{Z}}, F)$ and clopen set $C \subseteq A^{\mathbb{Z}}$, it is undecidable whether or not there exists a configuration $c \in C$ such that $F^n(c) \in C$ for every $n \in \mathbb{Z}$.*

5.2 Undecidability of left expansivity

In this section it is shown that is is undecidable whether or not the given cellular automaton is left expansive. However, it still remains an open problem whether or not expansivity or positive expansivity are a decidable properties.

Theorem 5.2.1. *Given a reversible cellular automaton $(A^{\mathbb{Z}}, F)$, it is undecidable whether or not F is left (or right) expansive .*

Proof. Let $((T \cup S)^{\mathbb{Z}}, G)$ be the cellular automaton of Theorem 5.1.5. The states that do not contains the signals generated by tile T_2 are denoted by

T and the states that do contain the signals are denoted by S. The new state set is $A = (T \cup S) \times \{0,1\} \times \{0,1\}$. The new rule is defined by

$$F(c)(i) = \begin{cases} (G(c_1)(i), \sigma(c_2)(i), c_3(i)) & \text{if } c_1(i) \in T, \\ (G(c_1)(i), c_3(i), \sigma(c_2)(i)) & \text{if } c_1(i) \in S, \end{cases}$$

where $c \in A^{\mathbb{Z}}$, $c(i) = (c_1(i), c_2(i), c_3(i))$, $c_1 \in (T \cup S)^{\mathbb{Z}}$, $c_2 \in \{0,1\}^{\mathbb{Z}}$ and $c_3 \in \{0,1\}^{\mathbb{Z}}$. That is, the new cellular automaton contains three different layers. On the first layer the original cellular automaton is simulated. The second and the third layer contain only binary states that either remain in place or move one cell per time step to the left. If the state of the first layer belongs to S then the second and the third component are interchanged. The idea is that if all the cells of the first layer are in states of T, the states of the second layer are moved one step to the left and the states of the third layer remain unchanged. The cellular automaton $(A^{\mathbb{Z}}, F)$ is left expansive if, and only if, the answer to the question of Theorem 5.1.5 is affirmative for $((T \cup S)^{\mathbb{Z}}, G)$.

Assume that for every initial configuration every cell of $((T \cup S)^{\mathbb{Z}}, G)$ must enter to a state in S at some point. Due to the compactness, this happens for every cell infinitely often and there exists a bound N such that during N time steps every cell must enter to a state in S at least once. This means that a binary state value of originally the third component of cell i must travel at least one cell to the left for every N time step. This means that no single state of the second of the third layer remains in place but travel infinitely far to the left eventually. Hence the new cellular automaton is left expansive.

Assume that for some initial configuration c cell i of $((T \cup S)^{\mathbb{Z}}, G)$ does not enter a state in S ever. Let c_0 be a configuration such that $c_0(j) = 0$ for every j. Let c_1 be a configuration such that $c_1(j) = 1$ if $i = j$ and $c_1(j) = 0$ otherwise. Then $d(F^n(c, c_0, c_0), F^n(c, c_0, \sigma^{-i}(c_1))) \geq \varepsilon$ for any ε by choosing suitably large i. That is, the information of the difference only at the third component of cell i does not travel arbitrarily far to the left. Hence the new cellular automaton is not left expansive. □

Chapter 6

Undecidability of sensitivity to initial conditions

In this chapter it is shown that sensitivity to initial conditions is an undecidable property for reversible cellular automata. The result follows by constructing a cellular automaton which is sensitive if, and only if, a given reversible Turing machine does not eventually halt on an empty tape.

Previously it has been known that sensitivity is an undecidable property for not necessarily reversible cellular automata [28].

In this chapter's cellular automata construction the blocking words (if they exist) are certain words representing a halting Turing machine simulation with some correctly aligned additional "signals" and labels. This is different from the construction of [28] where the Turing machine simulation is conducted between words consisting of quiescent states which act as blocking words.

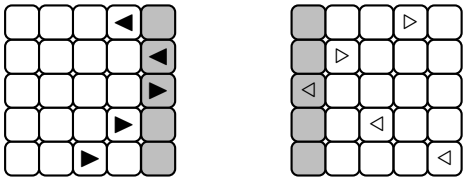
6.1 Concept of signals

In what follows, the concept of a *signal* is used frequently. Although it is a very informal concept, formally a signal could be described as just a state or a component of a state travelling a (piecewise) linear path. A signal has speed, that is, a number of cells it travels per one time step and a direction to which it moves.

Let $|i - j| < |v|$. If a signal with speed $v > 0$ in location i is said to *collide* with or *bounce* off the cell in location $j > i$, its speed is changed from v to $-v$ and its new location is $(j + 1) - (|v| - |i - j|)$. Similarly, if a signal with speed $v < 0$ in location i bounces off the cell in location $j < i$, its new location is $(j-1)+(|v|-|i-j|)$. If the signal is located between two cells whose distance is less than the velocity of the signal and both of which it would bounce off, the same idea of changing the speed to the opposite

and computing the new location is applied repeatedly. If a signal, which is located at cell i, does not have its path intersect any cells from which it would bounce off in locations j, where $|i - j| < |v|$, then its new location is simply $i + v$.

Geometrically speaking, if cell i occupies the unit interval $[i - \frac{1}{2}, i + \frac{1}{2}]$ and a signal travelling to the right is represented by a line intersecting a point (i, t) at time t with a slope of $1/v$, the signal is reflected as a line with respect to a vertical axis $x = j + \frac{1}{2}$. Similarly, a signal travelling to the left as a line through a point (i, t) at time t with a slope of $1/v$ is reflected with respect to a vertical axis $x = j - \frac{1}{2}$. The idea of signals is shown in Figure 6.1.



(a) A signal with speed 1 bounces back from a cell to the right.

(b) A signal with speed 2 bounces back from a cell to the left.

Figure 6.1: Examples of how signals are represented on a cellular automaton and how they bounce back from certain cells.

The concept of signals is used in the following constructions because that way the main ideas of the proofs can be expressed more efficiently.

6.2 Outline of the construction

For each reversible Turing machine, the cellular automaton is constructed in four layers. The construction is presented in this way to make it more readable. The local rule on each of these layers maps states of the neighborhood into a new state component of the layer depending on the state components of the particular layer and the previous layers. That is, each layer has a single purpose and its contents depend only on the contents of the layer itself and the contents of the underlying layers in the previous configuration.

Layer 1. The configuration is split into areas on each of which the Turing machine is simulated.

Layer 2. For each simulation area on layer 1, a signal is used to check the validity of the initial tape configuration.

Layer 3. If the contents of layers 1 and 2 do not match a valid simulation, *error signals* are generated to forward this information.

Layer 4. This layer contains *border signals* and two types of *activation signals*. A border signal can block the propagation of activation signals. If a signal of layer 3 is present in the same cell as a border signal on this layer, the border signal can be made to change its state using the activation signals so that further activation signals can pass through it.

The construction is such that the cellular automaton consisting of only the first three layers always has equicontinuity points. However, the fourth layer is constructed in such a way that in the final construction blocking word sequences exist if, and only if, certain states can be avoided in the blocking word sequences of the cellular automaton consisting of the first three layers.

The idea of the construction is to use the first and the second layer to simulate periodically a computation by a reversible Turing machine starting on an empty tape. The Turing machine computation is simulated on the first layer on a finite area. The configuration of the cellular automaton is divided into areas on each of which the Turing machine computation is simulated (on arbitrary input). The construction on the second layer is used to detect whether or not the Turing machine simulation was started on a specific initial state and an empty tape. The construction for the first layer is practically the same as in [54] where equicontinuity of reversible cellular automata was proven to be an undecidable property.

The third layer is used to forward information about the validity of the computation. That is, if the first layer does not contain periodically a simulation of a Turing machine computation started on an empty tape, a set of signals is generated to spread this information. The signals symbolizing a failed attempt to find a positive solution to the halting problem are restricted inside the particular simulation area of the Turing machine. Therefore, each of the simulation areas is in fact a blocking word sequence when layer 4 is not considered, because the contents of one simulation area are unaffected by the contents of the other simulation areas. The rule which is used on the third layer to draw the signals is determined locally by the contents of the first and second layer.

A *simulation error* is said to be present in a cell, if the contents of the first and second layer in the neighborhood of the cell are such that the Turing machine simulation should be considered failed in the search for a positive solution to the halting problem. Then, on the third layer, a different local rule is used depending on whether the cell has a simulation error present or not.

The fourth layer contains three different kinds of signals. One of the signal types is used to prevent the other two kinds of signals from crossing a simulation area. However, if a simulation area contains a negative instance to the halting problem, the signals used as border signals can be made to switch to an unblocking state and the other two signal types can pass through the simulation area. Therefore, the cellular automaton is sensitive if, and only if, the problem of Theorem 2.1.6 has a negative answer. Eventually, the state set of the cellular automaton is $A_1 \times A_2 \times A_3 \times A_4$, where A_i denotes the state components of layer i.

Remark 6.2.1. *The construction in this chapter does not actually require the Turing machine to be reversible. It is enough to assume that the Turing machine is injective as a dynamical system. In that case the Turing machine computation can be reversed using a cellular automaton rule by choosing the neighborhood of the local rule large enough.*

For one-dimensional reversible cellular automata the size and location of inverse neighborhood is polynomially bounded with respect to the number of states and the length of the neighborhood [23, 22]. Furthermore, the inverse rule of a one-dimensional cellular automaton can be computed in polynomial time [101].

6.3 Layer 1: representing Turing machine computation

Let Q be the state set of the Turing machine and let the initial state of the Turing machine be denoted by q_0. Let Γ denote the tape alphabet of the Turing machine. Then the cell states representing the read-write head reading a single tape letter are $\{\blacktriangle, \blacktriangledown\} \times Q \times \Gamma$. The cell states representing a single tape letter are $\{\triangleright, \triangleleft\} \times \Gamma$.

Symbols $\blacktriangle$ and $\blacktriangledown$ are used together with the state set Q to distinguish which rule, the original Turing machine or its inverse, is used in the local rule of the cellular automaton. Therefore, the state set representing the Turing machine on the first layer is

$$A_1 = (\{\blacktriangle, \blacktriangledown\} \times Q \times \Gamma) \cup (\{\triangleright, \triangleleft\} \times \Gamma).$$

Let sets $\{\triangleright\} \times \Gamma$ and $\{\triangleleft\} \times \Gamma$ be denoted by T_L and T_R, respectively. The elements of T_L are used to represent the tape contents to the left from the read-write head and the elements of T_R are used to represent the tape contents to the right from the read-write head. The elements of T_L and T_R will be called *left tape states* and *right tape states*, respectively. Let ε denote the empty tape symbol. Let H denote the set of states containing the read-write head.

By associating each tape letter with either expression $\triangleright$ or $\triangleleft$ the tape is divided into areas where each read-write head occurrence is preceded by a certain number of elements of T_L on the left and succeeded by a certain number of elements of T_R on the right. Therefore, the configuration is always partitioned into disjoint areas in each of which the Turing machine is simulated on some configuration.

A *simulation area* in a configuration c is an integer interval of cell locations $S = \{i, \ldots, j\}$, where $i \leq j$ and

1. if $k, k+1 \in S$ and $c(k+1) \in T_L$ then $c(k) \in T_L$,
2. if $k, k+1 \in S$ and $c(k) \in T_L$ then $c(k+1) \in T_L \cup H \cup T_R$,
3. if $k, k+1 \in S$ and $c(k+1) \in H$ then $c(k) \in T_L$,
4. if $k, k+1 \in S$ and $c(k) \in H$ then $c(k+1) \in T_R$,
5. if $k, k+1 \in S$ and $c(k+1) \in T_R$ then $c(k) \in T_L \cup H \cup T_R$,
6. if $k, k+1 \in S$ and $c(k) \in T_R$ then $c(k+1) \in T_R$, and
7. $c(i-1) \notin T_L$ and $c(j+1) \notin T_R$.

This definition splits a configuration into sequences of consecutive cells where there is first a certain number of left tape states, then possibly a state representing a read-write head, and finally a certain number of right tape states. With this definition, a domain which does not contain a state representing the read-write head may be a simulation area.

The rightmost cell of a simulation area is called the *right border* of the simulation area. Likewise, the leftmost cell of the simulation area is called the *left border* of the simulation area. If the simulation area contains only one cell (in which case it is in a state from H), the left border and the right border are the same cell. A left border cell and a right border cells are identifiable using a local rule.

By labelling the cell states representing the different sides of the Turing machine tape, the read-write heads are forced not to enter other simulation areas. If the read-write head moves to the left, it labels the previous cell to belong to the right side of the tape. Similarly, if the read-write head moves to the right, it labels the previous cell to belong to the left side of the tape. This way the simulation area maintains its constant width.

Expressions $\blacktriangle$ and $\blacktriangledown$ are used to denote the application of the Turing machine transition function as the local rule and the corresponding inverse operation. By using values $\blacktriangle$ and $\blacktriangledown$ together with the original states of the Turing machine, the cellular automaton becomes reversible even if the transition function of the Turing machine was only a partial function. Let $c \in A_1^{\mathbb{Z}}$ and $c(i) = (\blacktriangle, q, a)$. If the Turing machine move defined by pair

(q, a) cannot be executed, state $(\blacktriangle, q, a)$ is replaced with state $(\blacktriangledown, q, a)$. That is, state component $\blacktriangle$ is replaced with $\blacktriangledown$ if $\delta(q, a)$ is undefined, $\delta(q, a)$ defines a left move but $c(i-1) \notin T_L$ or $\delta(q, a)$ defines a right move but $c(i+1) \notin T_R$. Similarly, if the inverse move cannot be executed, state $(\blacktriangledown, q, a)$ is replaced with state $(\blacktriangle, q, a)$. With this construction an eventually halting computation with the Turing machine always leads to a periodic computation with the cellular automaton. In the peculiar case of the read-write head being located between a right tape cell to the left and a left tape cell to the right, the read-write head state $(\blacktriangle, q, a)$ is constantly swapped with $(\blacktriangledown, q, a)$.

A read-write head is forced to be always present within a simulation area by defining the alternative cases to be simulation errors. That is, if a cell represents a left tape cell and the cell to its right represents a right tape cell, the cell is defined to have a simulation error. Similarly, if the cell represents a right tape cell and the cell to its left represents a left tape cell, the cell has a simulation error. With these constraints, a missing read-write head is always dealt with on the third layer.

It is shown in Figure 6.2 how a read-write head moves within a simulation area.

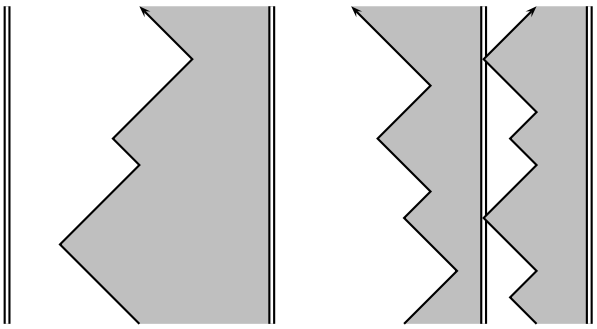


Figure 6.2: Labelling cells to represent either the left or the right side of the Turing machine tape. The zig-zag arrows represent read-write heads. The vertical double lines represent the borders between simulation areas.

6.4 Layer 2: verifying initial configuration

On the second layer, the validity of the Turing machine initial configuration is verified. If the Turing machine does not erase the tape periodically and re-enter the initial state, simulation errors are found present in the computation and they will affect the computation on the third layer.

The existence of the empty initial configuration is determined by defining signals which travel with twice the speed of the read-write head, that is, two cells per time step. These signals will be called *verification* signals. They simply bounce between the left border and the right border of the simulation area without ever passing from one simulation area to another. The greater speed of the signal is used to ensure that a verification signal intersects the path of the read-write head only once during a pass from the left side border to the right side border.

The movement of the verification signal is presented in Figure 6.3.

Let expressions ▶ and ◀ denote a verification signal moving to the right and a verification signal moving to the left, respectively. The existence of the verification signal is forced by using states $\triangleright$ or $\triangleleft$. State $\triangleright$ (resp. $\triangleleft$) tells that the verification signal is located to the right (resp. left) from the cell. The idea is the same as on the first layer with the read-write head. A verification signal moving to the right can move only as far as there are states $\triangleleft$ to its right. The verification signal at location i moving to the right (i.e. state ▶) bounces back from cell i if cell $i+1$ has value $\triangleright$ or from cell $i+1$ if cell $i+1$ has value $\triangleleft$ and cell $i+2$ has value $\triangleright$. Similarly, a verification signal moving to the left can move only as far as there are states $\triangleright$ to its left. The verification signal at location i moving to the left (i.e. state ◀) bounces back from cell i if cell $i-1$ has value $\triangleleft$ or from cell $i-1$ if cell $i-1$ has value $\triangleright$ and cell $i-2$ has value $\triangleleft$. This way the configuration is divided into disjoint, consecutive and continuous areas on each of which a single verification signal bounces back and forth.

The state of a cell is changed from $\triangleleft$ to $\triangleright$ or from $\triangleright$ to $\triangleleft$ when the path of the verification signal crosses with the cell. Assume that a verification signal is contained in cell location i. Then the verification signal is replaced with state $\triangleright$ or $\triangleleft$, if the verification signal can move to the right or left, respectively. If the verification signal moves from cell location i to $i+2$, the state components of both cells i and $i+1$ are replaced with value $\triangleright$. Similarly, if the verification signal moves from cell location i to $i-2$, the state components of both cells i and $i-1$ are replaced with value $\triangleleft$.

Use of the verification signal is essentially the same method as dividing the configuration into simulation areas on the first layer according to left tape cells and right tape cells. Now the verification signal acts like the read-write head on the first layer changing the state component which points towards its location. The final state component set of the second layer is

$$A_2 = \{\triangleright, \triangleleft, ▶, ◀\}.$$

To ensure that the areas where the verification signals bounce back and forth are located exactly the same way as the simulation areas, the alternative case is defined to be a source of simulation errors. This can be done

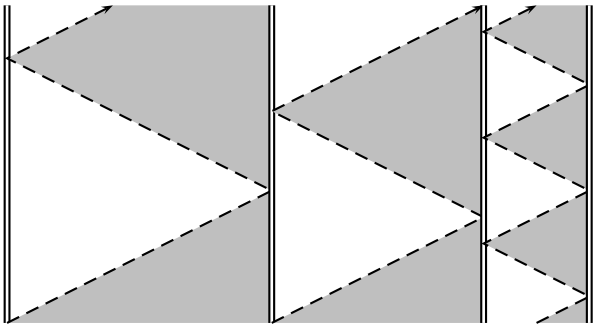


Figure 6.3: Drawing a verification signal to verify the initial configuration. The signal bounces between the borders of simulation areas.

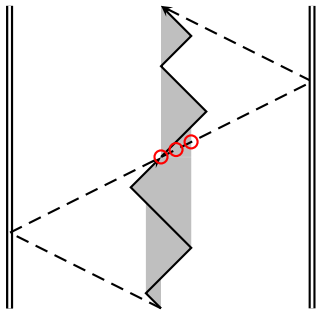
by defining appearances of the state components pair $\triangleleft$ and $\triangleright$ to be simulation errors unless value $\triangleleft$ is located on the right border of a simulation area and value $\triangleright$ is located on the left border of another simulation area. If a right border and a left border to its right both contain either value $\triangleleft$ or $\triangleright$, it is considered a simulation error. It is considered a simulation error, if there are two verification signals side by side on top of the same simulation are. Hence, unless the domain on which a verification signal moves back and forth does not match a domain of a simulation area, simulation errors occur.

It is also considered a simulation error, if the path of a verification signal intersects with the path of a read-write head when the read-write head is not in the initial state. Likewise, it is considered a simulation error, if the verification signal intersects a cell which contains a different Turing machine tape symbol than the empty tape symbol ε. Therefore, simulation errors occur if the Turing machine simulation is not started on an empty tape with the read-write head being in the initial state.

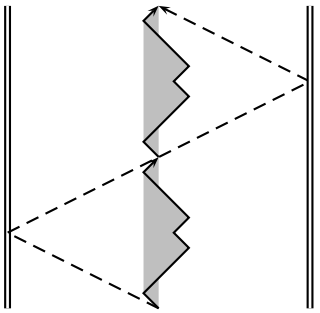
It is shown in Figure 6.4 how the verification signal sweeps the simulation area back and forth in search for signs of incorrect initial configuration.

6.5 Layer 3: detecting incorrect cell state combinations

The third layer is used to react on the simulation errors detected on the first two layers. This is done by introducing a new set of signals called *error signals*. An error signal travels one cell either to the left or right per one time step and it bounces back from a left or right simulation area border, respectively. An error signal always travels a straight line unless it collides with a simulation area border or there is a simulation error present in the



(a) A simulation is executed unsuccessfully: The verification signal intersects the non-empty tape cells (denoted by circles) before the read-write head is able to erase them.



(b) An eventually halting simulation is executed successfully: The Turing machine starts and halts periodically on an empty tape and the verification signal intersects the read-write head path only at the initial configuration.

Figure 6.4: Pairing the Turing machine simulation and the verification signal for the initial configuration. The solid arrow denotes the read-write head, the gray area denotes non-empty tape cells and the dashed arrow denotes the verification signal. The simulation area borders are denoted by vertical double lines.

cell. In a cell where there is a simulation error present, the propagation of error signals is determined according to a different rule. If there is no error signal entering a cell which contains a simulation error, then two error signals are created, one of which travels to the left and another to the right. A single error signal passes through a cell with a simulation error unchanged. If two error signals enter a cell with a simulation error, both of them are erased. An outline of these different cases is shown in Figure 6.5. It is shown in Figure 6.5(a) how error signals propagate when no simulation error is present and in Figure 6.5(b) it is shown how error signals are modified when a simulation error is present.

As explained already in the sections describing layers 1 and 2, occurrences of the following cell state combinations are defined to be simulation errors:

1. A simulation area does not contain a state representing the read-write head, that is, at two adjacent cells belonging to the same simulation area the leftmost cell has value $\triangleright$ (representing a left tape state) and the rightmost cell has value $\triangleleft$ (representing a right tape state) on layer 1.

2. A read-write head collides with a simulation area border on layer 1.

(a) At a location where no simulation error is present, the error signals can intersect and propagate freely.

(b) At a location where simulation error is present, a single error signal can propagate freely but two error signals are either created or erased together.

Figure 6.5: The error signals are allowed to propagate freely if no simulation error is encountered. If a simulation error is encountered, a single signal can propagate freely but two signals are either created or erased.

3. A verification signal intersects a cell where the tape is non-empty on layer 1.

4. A verification signal intersects the path of the read-write head when the read-write head is not in the initial state on layer 1.

5. Of two adjacent cells belonging to the same simulation area one has value $\triangleright$ and the other has value $\triangleleft$ for the component of layer 2. That is, either the verification signal is missing or the areas on the first two layers do not match.

6. Two verification signals are located in two neighboring cells on top of the same simulation area, that is, the areas on the two layers do not match.

7. The component of layer 2 has value $\triangleleft$ on the left border, that is, the areas on the two layers do not match.

8. The component of layer 2 has value $\triangleright$ on the right border, that is, the areas on the two layers do not match.

At these locations the error signals are modified according to Figure 6.5(b).

The cellular automaton consisting only of layers 1, 2 and 3 is equicontinuous on all configurations containing only finitely long simulation areas. This follows from the fact that the contents of one simulation area do not affect another simulation area.

Using the error signals it would be possible to allow a new set of diagonally advancing signals to cross simulation area borders. That is, at the location where an error signal bounces back from a simulation area border,

a signal of another type would be allowed to cross the border. However, this is not enough to remove the blocking property of multiple simulation areas next to each other. Namely, a signal crossing one simulation area border might always "strangely" bounce back from the border of the next simulation area where an error signal might not be present. This might happen, for example, if adjacent simulation areas contain the same Turing machine computation but in a different stage. For this reason, the blocking property of the simulation area border should be more controlled and it should not depend only on the error signals.

On the other hand, if reversibility was not a requirement, then the simulation area border could be permanently changed to a state which allows information pass through it. Therefore, in the not necessarily reversible case, layer 4 would not be needed in its current complexity.

Formally, the state component of the third layer can be presented by elements of

$$A_3 = \{\Diamond, \blacktriangleright, \blacktriangleleft, \blacklozenge\},$$

where different elements represent different combinations of error signals moving left and right.

Let $(A^{\mathbb{Z}}, F)$ be the cellular automaton consisting of layers 1, 2 and 3. That is, the state set is

$$A = A_1 \times A_2 \times A_3,$$

and the global rule F is defined as described in Sections 6.3, 6.4 and 6.5. Now the following theorem follows directly:

Theorem 6.5.1. *Given a non-sensitive reversible one-dimensional cellular automaton $(A^{\mathbb{Z}}, F)$ and a subset $E \subseteq A$ of its state set, it is undecidable whether or not there exists an equicontinuity point $x \in A^{\mathbb{Z}}$ such that $F^n(x)(0) \in A \setminus E$ for every $n \in \mathbb{N}$.*

Proof. Let E be the set of states that contain error signals. By the construction of layers 1, 2 and 3 it is undecidable whether or not all finite simulation areas will generate error signals. If error signals appear in a simulation area, then the error signals necessarily visit every cell of the simulation area according to the rules in Figure 6.5. Therefore, if the reversible Turing machine does not eventually halt, every cell in every finite simulation area will at some point contain error signals.

If the reversible Turing machine does eventually halt, then a finite simulation area exists on which no error signals appear and this area can be chosen to overlap the origin.

An infinite simulation area might never contain an error signal if it consists of, say, tape states of one side only. However, a configuration which

contains a one-way or two-way infinite simulation area cannot be an equicontinuity point because the contents of the infinite simulation area can be chosen to contain or not an error signal which travels through every cell of the simulation area. □

6.6 Layer 4: possible sensitivity

Now a new cellular automaton $(B^{\mathbb{Z}}, G)$ is constructed by adding a new layer to the cellular automaton $(A^{\mathbb{Z}}, F)$ of Theorem 6.5.1. The construction could be done explicitly by using the construction $(A^{\mathbb{Z}}, F)$ but this is not necessary.

The new layer consists of states from set A_4. That is, the state set A of $(A^{\mathbb{Z}}, F)$ is replaced with

$$B = A \times A_4 = A_1 \times A_2 \times A_3 \times A_4,$$

and the new global rule G is defined accordingly. The states $E \times A_4$ will be called *error states* (motivated by Theorem 6.5.1) in the new cellular automaton.

Layer 4 consists of adding three different types of signals on top of cellular automaton $(A^{\mathbb{Z}}, F)$. One of these signal types is a vertical signal, which will be used to either block or let the other two signal types to pass. The signals will be defined in such a way that a vertical signal can remain in a blocking state indefinitely if, and only if, the answer to the question of Theorem 6.5.1 is positive.

The three signal types used on the new layer will be called *border* signals and *activation* signals of *type 1* and *type 2*. A detailed description of the interaction of these signal types is shown in Figures 6.6 and 6.7. An active border signal is represented by a double vertical line and an inactive border signal is represented by a single vertical line. An activation signal of type 1 is represented by a dashed arrow and an activation signal of type 2 is represented by a dotted arrow.

A border signal is a signal which travels only vertically. It is represented by using a ternary component in every cell with one of the values ◊, □ and ■. An absent border signal is represented by value ◊ whereas a present border signal is represented by values □ and ■. The border signal is said to be *inactive* (resp. *active*) if the component has value □ (resp. ■). The main idea is that an active border signal blocks the propagation of activation signals indefinitely if the cell does not enter an error state. If a cell i containing a border signal is in such a state that it blocks activation signals, then a signal coming from the left bounces back from the cell i and a signal coming from the right bounces back from the cell $i + 1$. For example, let two border signals be located in cells i and j with $i < j$ and assume that the signals are in an active state and the cells i and j do not enter error states.

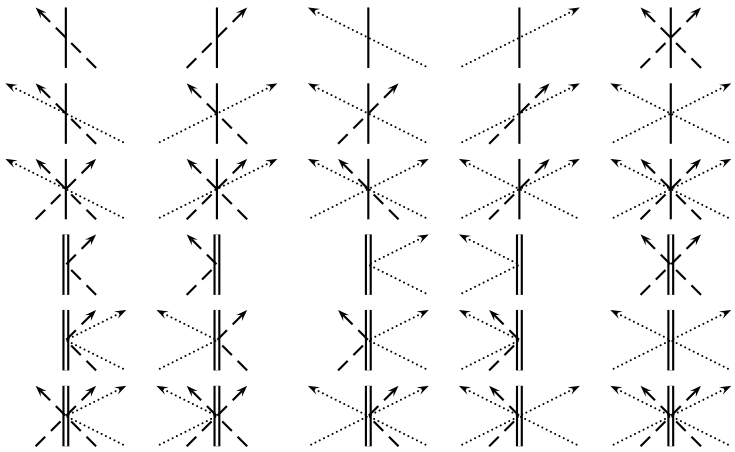


Figure 6.6: The interaction of different signal types when the cell containing the border signal is not in an error state. The first three rows of pictures show how activation signals pass through inactive border signals. The second three rows of pictures show how activation signals bounce back from active border signals.

Then activation signals located in between the cells i and j will bounce back and forth between the cells $i+1$ and j.

An activation signal of type 1 is a signal which travels to the left or to the right one cell per one time step. An activation signal of type 2 is a signal which travels to the left or to the right two cells per one time step. The activation signals are used to change the states of border signals. The state of a border signal is changed if, and only if, the cell is in an error state and there is a single type 1 activation signal coming either from the left or from the right and a type 2 activation signal coming from the right. Let a cell in location i contain a border signal. If the cell i is in an error state and the cell $i+1$ contains a type 2 activation signal moving to the left, a type 1 activation signal moving to the right in the cell i bounces back from cell i and the state of the border signal is changed. If the type 1 activation signal is moving to the left in the cell $i+1$, it bounces back from cell $i+1$ and the state of the border signal is changed.

Formally, the state component of the fourth layer is expressed by elements of

$$A_4 = \{\Diamond, \Box, \blacksquare\} \times \{\Diamond, \blacktriangleright, \blacktriangleleft, \blacklozenge\} \times \{\Diamond, \blacktriangleright, \blacktriangleleft, \blacklozenge\},$$

where the sets represent different signal types and their elements represent different combinations of a particular signal type.

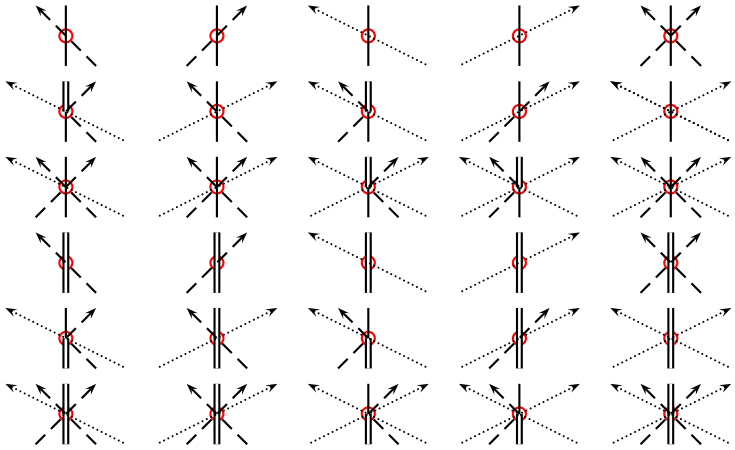


Figure 6.7: The interaction of different signal types when the cell containing the border signal is in an error state. The border signal changes its blocking state if there is a single activation signal of type 1 coming either from the left or from the right and an activation signal of type 2 coming from the right. In this case the type 1 activation signal bounces back.

A detailed description of the local rule for the signals can be found in Figures 6.6 and 6.7. The description of the local rule can be summarized as a following list of rules:

1. Any type of activation signal travels through an inactive border signal if no error state is present.

2. Any type of activation signal bounces off an active border signal if no error state is present.

3. A type 1 activation signal travels through any border signal if the cell is in an error state and no type 2 activation signal is coming from the right.

4. A type 2 activation signal always travels through any border signal if the cell is in an error state.

5. The state of the border signal is changed if, and only if, (1) the cell is in an error state, (2) there is a single activation signal of type 1 coming either from the left or from the right and (3) an activation signal of type 2 coming from the right. In this case the type 1 activation signal bounces back.

6.7 Undecidability of sensitivity

In this section it is concluded that the cellular automaton constructed in previous sections is sensitive to initial conditions if, and only if, the given reversible Turing machine does not eventually halt on an empty tape. This follows from the fact that the contents of a simulation area which contains an eventually halting Turing machine simulation can be used to construct a blocking word.

Recall that the cellular automaton $(B^{\mathbb{Z}}, G)$ was constructed in Section 6.6 by adding an additional layer of signals to the cellular automaton $(A^{\mathbb{Z}}, F)$ of Section 6.5.

Lemma 6.7.1. *If the question of Theorem 6.5.1 has a positive answer for the cellular automaton $(A^{\mathbb{Z}}, F)$, then the cellular automaton $(B^{\mathbb{Z}}, G)$ has a blocking word.*

Proof. Let $c_A \in A^{\mathbb{Z}}$ be an equicontinuity point such that $F^n(c_A)(0) \in A \setminus E$ for every $n \in \mathbb{N}$. Then, by the definition of an equicontinuity point, there exists a positive integer k such that the word $w = c_A[-k, k]$ is a blocking word and for every configuration $c \in A^{\mathbb{Z}}$ with $c[-k, k] = w$ condition $F^n(c)(0) \in A \setminus E$ holds.

Define word $w_{B,\blacksquare} \in B^*$ by setting

$$\begin{array}{ll} w_{B,\blacksquare}(i) = (w(i), \Diamond, \Diamond, \Diamond) & \text{if } 0 \leq i < k, \\ w_{B,\blacksquare}(i) = (w(i), \blacksquare, \Diamond, \Diamond) & \text{if } i = k \text{ and} \\ w_{B,\blacksquare}(i) = (w(i), \Diamond, \Diamond, \Diamond) & \text{if } k < i \leq |w| - 1. \end{array}$$

Now word $w_{B,\blacksquare} w_{B,\blacksquare}$ is a blocking word for the cellular automaton $(B^{\mathbb{Z}}, G)$. This follows from the fact that active border signals (i.e. values $\blacksquare$) in the centers of words $w_{B,\blacksquare}$ can never be changed inactive. □

Lemma 6.7.2. *If the question of Theorem 6.5.1 has a negative answer for the non-sensitive cellular automaton $(A^{\mathbb{Z}}, F)$, then for the cellular automaton $(B^{\mathbb{Z}}, G)$ and any word $u \in B^*$ there exists a configuration $c \in \mathrm{Cyl}(u, 0)$ and a positive integer t_u^+ such that the configuration $G^t(c)$ does not contain any active border signals for any $t \geq t_u^+$.*

Proof. Let $w \in A^*$ be again a blocking word for the cellular automaton $(A^{\mathbb{Z}}, F)$. The blocking word exists because $(A^{\mathbb{Z}}, F)$ is non-sensitive. Define word $w_{B,\Diamond} \in B^*$ of the same length by setting

$$w_{B,\Diamond}(i) = (w(i), \Diamond, \Diamond, \Diamond)$$

whenever $0 \leq i < |w|$. The word $w_{B,\Diamond}$ is defined so that it does not contain any border signals.

Let $u \in B^*$ any word whose active border signals are supposed to be changed inactive eventually in a new configuration $c \in \mathrm{Cyl}(u, 0)$. Then the configuration $c \in B^{\mathbb{Z}}$ is constructed by first defining configuration c_n (where n is the number of border signals contained in the word u) so that

$$\begin{array}{ll} c_n(i) = w_{B,\Diamond}((i+1) \bmod |w_{B,\Diamond}|) & \text{if } i < 0, \\ c_n(i) = u(i) & \text{if } 0 \leq i < |u| \text{ and} \\ c_n(i) = w_{B,\Diamond}((i-|u|) \bmod |w_{B,\Diamond}|) & \text{if } |u| \leq i. \end{array}$$

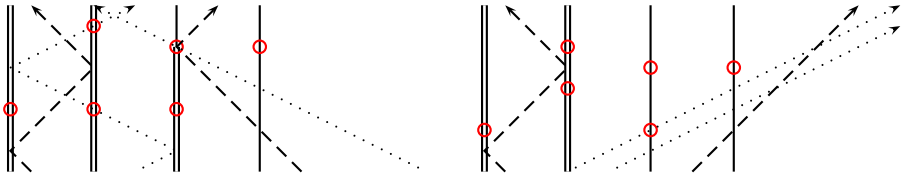
That is, configuration c_n is such that word u is located in the origin and word $w_{B,\Diamond}$ repeatedly appears to the left and to the right of the occurrence of the word u. Moreover, all the border signals are located within domain $[0, |u|-1]$. Let the border signals be located in locations $b_1, b_2, \ldots, b_n$ where $b_i < b_{i+1}$, $0 \leq b_1$ and $b_n < |u|$. Also, every cell of c_n (and its modifications) will enter an error state infinitely often, thanks to Theorem 6.5.1.

Second, the configuration c_n is modified iteratively so that all the border signals are changed inactive. Assume that c_k is a configuration and t_k is a time such that $G^t(c_k)$ has neither active border signals nor activation signals in locations $(b_k, b_n]$, and further, all activation signals to the left of cell b_1 are moving to the left and all activation signals to the right of cell b_n are moving to the right for every integer $t \geq t_k$. That is, after t_k time steps b_k is the rightmost cell containing an active border signal. Furthermore, it is assumed that no activation signals pass through it to the right after time step t_k without a modification to the the initial configuration. Now the configuration c_k is modified (in two steps) further to produce configuration c_{k-1} where cell b_{k-1} contains the rightmost active border after $t_{k-1} > t_k$ time steps. The steps of erasing an active border signal are illustrated in Figure 6.8.

Recall that a border signal state is changed if, and only if, (1) the cell containing the border signal is in an error state, (2) there is a single type 1 activation signal coming either from the left or from the right and (3) a type 2 activation signal coming from the right intersects the border signal at the same time.

1. Let $a > |u|$ be an integer such that $G^{t_k+a}(c_k)(b_k)$ is an error state. First, an activation signal of type 1 moving to the left is placed to the location $b_k + t_k + a + 1$ if, and only if, there is no type 1 signal coming from the left. That is, either an activation signal coming from the left is located in the cell b_k at time step $t_k + a$ or an activation signal coming from the right is located in the cell $b_k + 1$ at time step $t_k + a$.

   Second, an activation signal of type 2 is set to be located in the cell $b_k + 2(t_k + a) + 1$. These two different activation signals meet the border signal in cell b_k and together change it inactive. Integer a is



(a) Step 1 of erasing an active border signal. An activation signal of type 1 and an activation signal of type 2 meet the border signal in a cell which is in an error state.

(b) Step 2 of erasing an active border signal. The redundant activation signals are allowed to scatter away before erasing the next active border signal.

Figure 6.8: Two activation signals (first of type 1 and second of type 2) are added to the initial configuration to erase an active border signal.

taken large enough to make sure that an added activation signal of any type does not hit any earlier stage border signals in cells b_i, where $i > k$, while they are still in an active state.

2. A band of e (i.e. sufficiently many) cells in the locations $[b_k + 2(t_k + a) + 2, b_k + 2(t_k + a) + 1 + e]$ are set not to contain any activation signals of type 2 moving to the left. Then the type 1 activation signal that was used to change the border signal in the cell b_k and all the activation signals contained between the border signals in the cells b_{k-1} and b_k and moving to the right (at time step $t_k + a$) can move beyond the last border signal in the cell b_n without changing the already inactive border signals back to active. Any activation signal moving to the left (at time step $t_k + a$) either crosses the border signal in cell b_{k-1} (when it changes inactive or enters an error state) while moving to the left or collides with it and starts moving to the right. In any case, due to finiteness of u, in a finite number of time steps all the activation signals of type 1 or 2 that would cross the border signal in cell b_{k-1} while moving to the right have also done so. Within this time there should be no type 2 activation signals coming from the right so that no border signals are changed back to active state.

   Clearly, e can be chosen to be a finite positive integer because word u contains only finitely many activation signals bouncing back and forth between the border signals.

Let the modified configuration (produced from c_k) be denoted by c_{k-1}. Now, for some positive integer $t_{k-1} > t_k$ configuration $G^t(c_{k-1})$ contains neither active border signals nor activation signals in locations $(b_{k-1}, b_n]$, and further, all activation signals to the left of cell b_1 are moving to the left and all activation signals to the right of cell b_n are moving to the right for every integer $t \geq t_{k-1}$ and no activation signals meet with border b_{k-1} when it enters an error state. Now this iterative procedure is repeated n times to change the state of each border signal inactive and to allow enough time pass for all the activation signals to move beyond all the border signals. Eventually this procedure gives the configuration $c = c_0$.

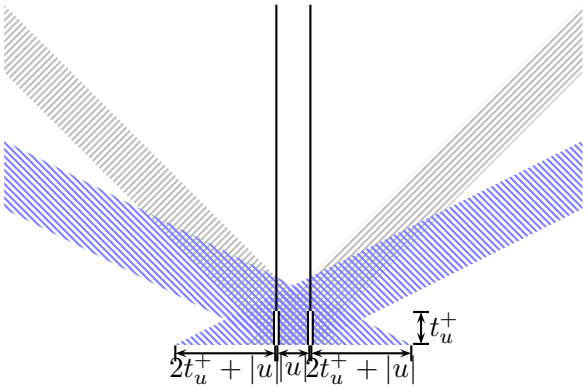
Assume that the active border signals in the configuration c are erased in t_u^+ time steps. Then only $2t_u^+ + |u|$ cells to the left and right of word u contain activation signals. That is, the configuration c was chosen in such a way that the activation signals appear only in the locations shown in Figure 6.9(a). Then for some positive integer t_u^+ configurations $G^t(c)$ do not contain any active border signals for $t \geq t_u^+$. □

An example on the usage of the activation signals to change the border signals inactive is shown in Figure 6.10. However, the contents of Figure 6.10 does not follow the strict guidelines given in the proof of Lemma 6.7.2. The state of a border signal is changed if the cell containing the border signal enters an error state and there is a single activation signal of type 1 coming either from the left or from the right and an activation signal of type 2 coming from the right.

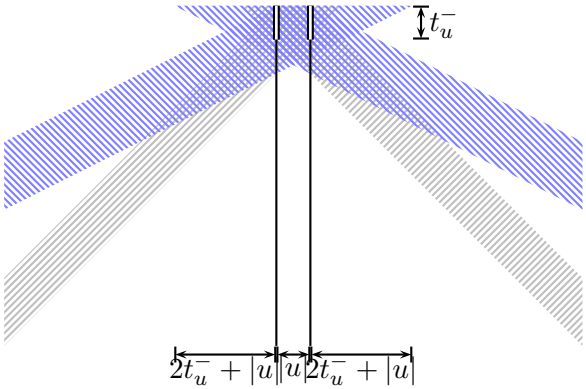
Because the signal interaction on layer 4 is almost identical for both the forward rule and the inverse rule, a similar lemma holds for the inverse rule also. The only difference is that because an activation signal of type 2 has the "activation property" only when it is coming from the right, the iterative process places the required signals to the left from the word occurrence instead of placing them to the right when working with the inverse rule. To say it more practically, the set of rules in Figure 6.7 is invariant with respect to a rotation by 180 degrees. For the inverse rule, the locations of the signals of layer 4 are shown in Figure 6.9(b).

Lemma 6.7.3. *If the question of Theorem 6.5.1 has a negative answer for the non-sensitive cellular automaton $(A^{\mathbb{Z}}, F)$, then for the cellular automaton $(B^{\mathbb{Z}}, G)$ and any word $u \in B^*$ there exists a configuration $c \in \mathrm{Cyl}(u, 0)$ and a positive integer t_u^- such that the configuration $G^{-t}(c)$ does not contain any active border signals for any $t \geq t_u^-$.*

Theorem 6.7.4. *The cellular automaton $(B^{\mathbb{Z}}, G)$ is sensitive if, and only if, the question of Theorem 6.5.1 has a negative answer for the cellular automaton $(A^{\mathbb{Z}}, F)$.*



(a) All active border signals contained in u can be changed inactive after t_u^+ applications of the forward rule by choosing the contents of the $2t_u^+ + |u|$ cells to its left and to its right.



(b) All active border signals contained in u can be changed inactive after t_u^- applications of the inverse rule by choosing the contents of the $2t_u^- + |u|$ cells to its left and to its right.

Figure 6.9: If the question of Theorem 6.5.1 has a negative answer, all border signals contained in the word u can be changed inactive in a finite number of time steps. Double lines represent the locations between which active border signals may appear. Hash fill with positive slope and hash fill with negative slope represent the possible locations for activation signals of type 1 and type 2, respectively.

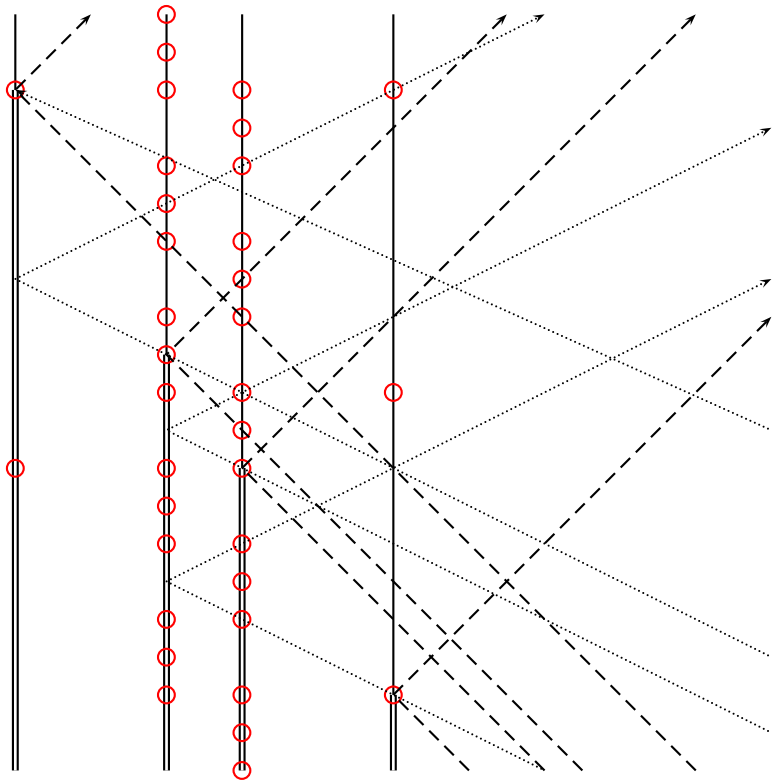


Figure 6.10: Any finite number of border signals can be changed to inactive state in which they can remain indefinitely long if every cell containing a border signal will enter an error state. Error states in the cells containing border signals are denoted by circles.

Proof. If the answer to the question of Theorem 6.5.1 is positive, then there exists an equicontinuity point for the cellular automaton $(A^{\mathbb{Z}}, F)$ with a cell that does not enter an error state ever. Then, according to Lemma 6.7.1, there exists a blocking word for the cellular automaton $(B^{\mathbb{Z}}, G)$. Hence, the cellular automaton $(B^{\mathbb{Z}}, G)$ is not sensitive.

If the answer to the question of Theorem 6.5.1 is negative, then the cellular automaton $(B^{\mathbb{Z}}, G)$ is sensitive by Lemma 6.7.2, because at some point border signals no longer block activation signals. □

Undecidability of sensitivity now follows from Theorem 6.5.1.

Corollary 6.7.5. *It is undecidable whether or not a given reversible one-dimensional cellular automaton is sensitive.*

Chapter 7

Undecidability of topological mixing and transitivity

In this chapter it is shown that sets of non-sensitive and topologically mixing reversible cellular automata are recursively inseparable sets. This is achieved by modifying the construction of Chapter 6. It directly follows that transitivity, topological mixing and (both Devaney's and Knudsen's) chaotic behavior are undecidable properties for reversible cellular automata.

It is described how the cellular automaton constructed in Section 6 can be modified in such a way that it will be topologically mixing and topologically transitive if, and only if, the original cellular automaton (of Section 6) is sensitive. The idea is to modify the cellular automaton in such a way that unless the original cellular automaton has a blocking word sequence, the contents of a simulation area (and actually any finite pattern in the original cellular automaton) can be shifted freely to the left and to the right. The shift effect is achieved by adding new states, called shift signals, that advance diagonally and do not affect the computation with the original states but act as a "filling" material. However, the computation with the original states does affect the propagation of the shift signals. To be precise, if a left shift signal encounters an active border signal (as defined in Section 6.6), it is changed to a right shift signal. Similarly, if a right shift signal encounters an active border signal, it is changed to a left shift signal. Therefore, the modified cellular automaton (with shift signals) is mixing if, and only if, blocking words do not exist for the original cellular automaton $(B^{\mathbb{Z}}, G)$.

7.1 Shift signals

Let the cellular automaton constructed in Section 6.6 be again denoted by $(B^{\mathbb{Z}}, G)$. The state set is modified in such a way that the modified

cellular automaton is topologically mixing if, and only if, the original cellular automaton is sensitive.

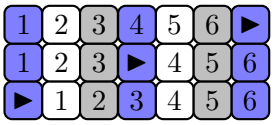
First, the state set B is swapped to the cartesian product B^N for some $N > r$, where r is the radius of the original local rule g. Because type 2 activation signals move with a speed of two cells per one time step, the value of r is at least 2. The value of N is not fixed until the proofs of Lemmas 7.2.3 and 7.2.4 where the reason for using a certain value for N is also seen. The local rule is modified accordingly by considering the N-tuples of states to form a single configuration consisting of the original states in a natural, sequential way. In terms of symbolic dynamics, the cellular automaton, which is modified this way, could be denoted by $((B^N)^{\mathbb{Z}}, \gamma_N \circ G \circ \gamma_N^{-1})$ using the notation in [65]. The function γ_N is the function which rearranges the cell structure by replacing every N consecutive cells with a vector containing the states as elements.

The goal of this change is to reduce the effective radius of the local rule. That is, the Nth iteration of the new local rule has radius r. In the modified cellular automaton N consecutive cells from the original cellular automaton $(B^{\mathbb{Z}}, G)$ occupy a single cell in the new cellular automaton.

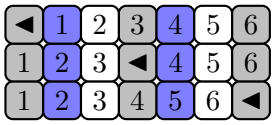
Second, the state set B^N is extended with additional states ▶ and ◀. State ▶ represents a *left shift signal* and state ◀ represents a *right shift signal*. The left shift signals and the right shift signals, together called *shift signals*, are empty place holders which are used to shift the location of states belonging to the original state set. A left shift signal can travel three cells to the right per one time step (as shown in Figure 7.1(a)) and a right shift signal can travel three cells to the left per one time step (as shown in Figure 7.1(b)).

The shift signals are not allowed to appear on arbitrarily many consecutive cells. The state ◀ signifying a right shift signal can be located only on cells at locations i, where $i \bmod 3 = 1$. Similarly, the state ▶ signifying a left shift signal can be located only on cells at locations i, where $i \bmod 3 = 2$. Therefore, at least the cells in locations i, where $i \bmod 3 = 0$, are in states from the original state set. With these constraints, the original local rule can be used to compute the next configuration from the original states found between the shift signals. At least every third cell does not contain a shift signal and therefore the radius of the new local rule remains finite.

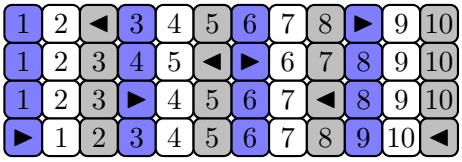
The collisions of the shift signals are defined differently from the description given in Section 6.6. This follows from the fact that the shift signals form a disjoint subset of the state set and their locations are restricted. If a shift signal does not encounter an active border signal, it travels a straight path as shown in Figure 7.2(a). If a shift signal encounters an active border signal, it is swapped to a shift signal travelling to the opposite direction (as shown in Figure 7.2(b)). In other words, a shift travels a straight path if,



(a) A left shift signal is used to move states of the original state set to the left.



(b) A right shift signal is used to move states of the original state set to the right.



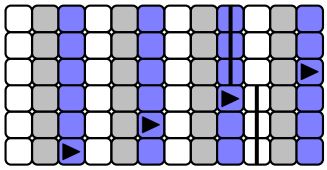
(c) The possible locations for the shift signals are restricted so that they can intersect while preserving the number of cells in the original states.

Figure 7.1: The use of the shift signals. Both shift signals work as a filling material moving elements of the original state set. For simplicity, the states of the original state set are denoted only by the numbers of their relative positions.

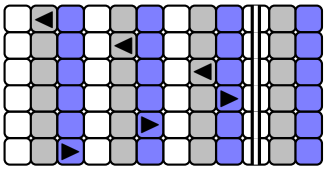
and only if, it does not bounce back from an active border signal. If the shift signal encounters an active border signal, it is bounced back.

However, because shift signals are restricted to only certain locations whereas a border signal can be located anywhere, it needs to be clarified where a left signal is swapped to a right signal and vice versa. In short, a left shift signal that would intersect with an active border signal is replaced with a right shift signal in the first possible location to the left of the border signal. Similarly, a right shift signal that would intersect with an active border signal is replaced with a left shift signal in the first possible location to the right of the border signal. The case of the left shift signal is shown in Figure 7.2(c). The collisions are defined in a similar way to right shift signals.

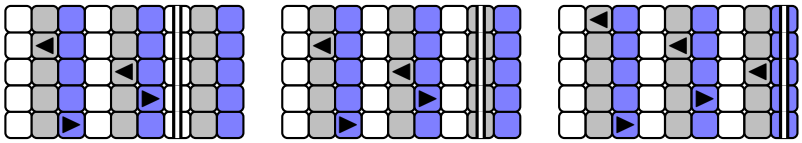
If the shift signal is located between two active border signals whose distance is less than the shift signal's movement amount, then the new location and movement direction is determined repeatedly in a natural way. That happens if a region between two active border signals contains only one cell in location $i \bmod 3 = 1$ or $i \bmod 3 = 2$, then the possible single shift signal in that location remains in place. This rule follows from the fact that the shift signal in considered to bounce back from both of the borders and change its direction twice.



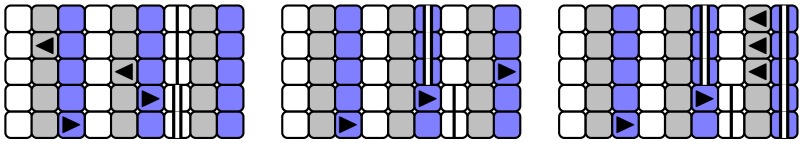
(a) A shift signal can travel through an inactive border signal.



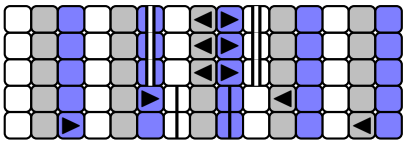
(b) A shift signal bounces back from an active border signal.



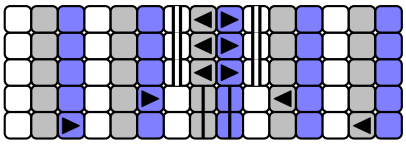
(c) The different cases of a left shift signal bouncing back.



(d) Other miscellaneous examples.



(e) An example of two shift signals ending up between two active border signals.



(f) Another example of two shift signals ending up between two active border signals.

Figure 7.2: Shift signals travel a straight diagonal path if, an only if, they do not encounter active border signals.

To enforce the constraint on the locations of shift signals, the state set of the cellular automaton is further modified from B^N to

$$C = B^N \times \left(B^N \cup \{\blacktriangleleft\}\right) \times \left(B^N \cup \{\blacktriangleright\}\right)$$

and the new local rule is defined accordingly. That is, the new local rule splits the new input cells into consecutive cell triplets in the old cell structure and then applies the old local rule.

Let the new global rule be denoted by H. Then the new cellular automaton is $(C^{\mathbb{Z}}, H)$, which was constructed by first joining consecutive cells to form N-tuples and second by adding the shift signals.

In terms of the original cellular automaton $(B^{\mathbb{Z}}, G)$, on every time step a shift signal shifts N states to the left or to the right by N cells.

Theorem 7.1.1. *The cellular automaton* $(C^{\mathbb{Z}}, H)$ *is reversible.*

Proof. The cellular automaton is seen to be reversible by showing that (1) the locations of the active border signals in the previous configuration can be uniquely determined and (2) the previous locations of the shift signals are then uniquely determined. These conditions are enough, since the underlying cellular automaton $(B^{\mathbb{Z}}, G)$ is already reversible.

First, from configuration $F(c)$ it is possible to determine the previous state of each border signal. If a border signal has been active in the previous configuration, then it has not been moved by a shift signal. Therefore, it can be determined how the previous configuration c is split into consecutive and disjoint regions which are bordered by active border signals.

Second, the next location of a shift signal in the configuration c is determined only by the border signals which are currently active. Because the active borders in the previous configuration c can be uniquely determined from configuration $F(c)$, the previous shift signal locations can be uniquely determined from the configuration $F(c)$. Therefore, the previous locations of shift signals are uniquely determined. □

7.2 Undecidability of topological mixing

In this section it is shown that the cellular automaton constructed in previous sections is topologically mixing if, and only if, the given injective Turing machine does not eventually halt on an empty tape.

If the given injective Turing machine eventually halts, it is possible to construct a blocking word which cannot be moved with the shift signals in the cellular automaton constructed in Section 7.1. To be exact, if the Turing machine eventually halts, there can exist a border signal which always remains in the active state. Then the shift signals simply bounce away from the border signal without moving it and the cellular automaton is not even sensitive.

If the Turing machine does not eventually halt, no blocking word sequence exists and eventually all simulation areas can be moved to the left and to the right at will (by modifying the initial configuration). If the Turing machine does not eventually halt, all the active border signals in a finite segment can be changed to inactive border signals. Once all the border signals are in inactive state, the contents of any finite segment can be shifted to the left or to the right by setting sufficiently many shift signal states to the initial configuration.

Lemma 7.2.1. *Suppose the question of Theorem 6.5.1 has a negative answer for the cellular automaton* $(A^{\mathbb{Z}}, F)$*. Then for the cellular automaton* $(C^{\mathbb{Z}}, H)$ *and any word* $u \in C^*$ *there exists a configuration* $c \in \mathrm{Cyl}(u, 0)$ *and a positive*

integer t_u^+ such that for any $t \geq t_u^+$ the configuration $H^t(c)$ does not contain any active border signals.

Proof. The proof is similar to that of Lemma 6.7.2. The shift signals only disperse away from the location of the word u when all the active border signals have been changed inactive. □

Similarly, Lemma 7.2.2 follows.

Lemma 7.2.2. *Suppose the question of Theorem 6.5.1 has a negative answer for the cellular automaton $(A^{\mathbb{Z}}, F)$. Then for the cellular automaton $(C^{\mathbb{Z}}, H)$ and any word $u \in C^*$ there exists a configuration $c \in \mathrm{Cyl}(u, 0)$ and a positive integer t_u^- such that for any $t \geq t_u^-$ the configuration $H^{-t}(c)$ does not contain any active border signals.*

Lemma 7.2.3. *Let $u \in C^*$ and $v \in C^*$ be two words of equal length k and assume that the question of Theorem 6.5.1 has a negative answer for the cellular automaton $(A^{\mathbb{Z}}, F)$ so that the value $t = \max(t_u^+, t_v^-)$ exists. Then there exists a positive integer m_0 and a configuration $c_m \in C^{\mathbb{Z}}$ for any positive integer $m \geq m_0$ such that*

1. *$c_m \in \mathrm{Cyl}(u, 0)$,*
2. *$H^m(c_m) \in \mathrm{Cyl}(v, 4\lceil t/N \rceil + 3k + 4\lceil rm/N \rceil)$ and*
3. *$H^i(c_m)$ does not contain any active border signals for $i \in [t, (m - t)]$.*

The idea of the proof is to place the words u and v with a suitable distance so that the contents of one word does not affect the contents of another. That is, the activation signals dispersing away from word u do not have time to reach the position of word v in the "condensed" cell structure.

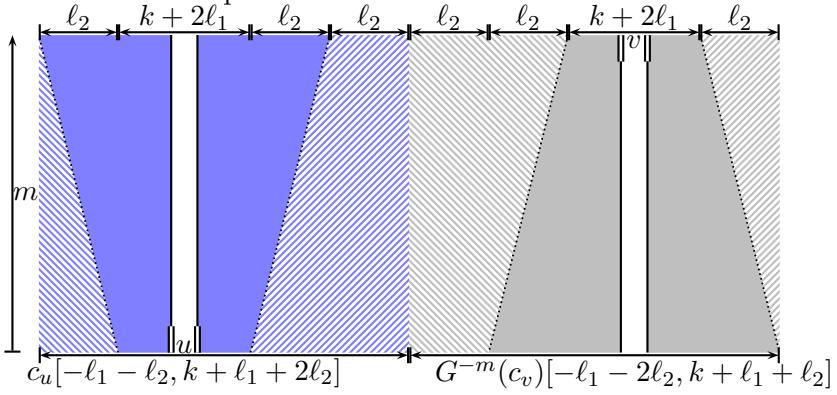


Figure 7.3: The configuration constructed in the proof of Lemma 7.2.3. The areas denoted by solid and line fill do not contain border signals. Activation signals are not found on the areas denoted by line fill.

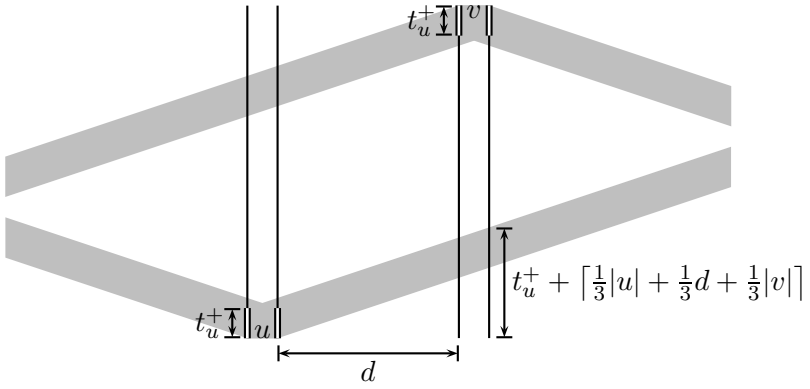


Figure 7.4: The shift signals originating from the word u pass through the future domain of word v in $t_u^+ + \lceil \frac{1}{3}|u| + \frac{1}{3}d + \frac{1}{3}|v| \rceil$ time steps. The solid gray areas denote the locations in which the shift signals may appear.

Proof. Let $\ell_1 = 2\lceil t/N \rceil + k$, $\ell_2 = \lceil rm/N \rceil$ and $\ell = \ell_1 + 2\ell_2$, where k is the length of words u and v and r is the radius of the cellular automaton $(B^{\mathbb{Z}}, G)$. Multiple $N\ell_1 = 2N\lceil t/N \rceil + Nk$ gives an upper bound for the number of states from B which need to be redefined around the words to change the active border signals inactive. That is, ℓ_1 is the equivalent bound of the new cell structure to the bounds "$2t_u^+ + |u|$" and "$2t_u^- + |u|$" in Figures 6.9(a) and 6.9(b). The bound ℓ_2 is simply chosen in a suitable way to have enough space between the two words.

Let c_u and c_v be the configurations constructed with the method of the proof of Lemma 7.2.1 and its analogy for the inverse rule, respectively. Let $d = 2\ell_1 + 4\ell_2$. That is,

$$d = 4\lceil t/N \rceil + 2k + 4\lceil rm/N \rceil$$

and it is the distance between the domain of u and the future domain of v. Then, the configuration c_m is constructed by first setting

$$\begin{aligned} c_m(i) &= c_u(i) && \text{if } i < k + \ell, \text{ and} \\ c_m(i) &= G^{-m}(c_v)(i - d) && \text{if } k + \ell \leq i. \end{aligned}$$

Second, those shift signals which are to be eventually located within word v are placed to suitable locations in the initial configuration.

However, a shift signal exiting one word must not affect the formation of the second word. Therefore, the number of time steps m is bounded from below by equation

$$m \geq t_u^+ + \left\lceil \frac{1}{3}|u| + \frac{1}{3}d + \frac{1}{3}|v| \right\rceil + t_v^-$$

which comes from the fact that after $\lceil \frac{1}{3}|u| + \frac{1}{3}d + \frac{1}{3}|v| \rceil$ time steps the left shift signals (which have slope $\frac{1}{3}$) exiting the domain of word u have passed through the future domain of word v as shown in Figure 7.4. Because $d = 2\ell_1 + 4\ell_2$, it follows it would be sufficient to have condition

$$\begin{aligned} & t_u^+ + \left\lceil \frac{1}{3}k + \frac{1}{3}\left(2(2\lceil t/N \rceil + k) + 4(\lceil rm/N \rceil)\right) + \frac{1}{3}k \right\rceil + t_v^- \\ & \leq t + \frac{1}{3}k + \frac{1}{3}\left\lceil (4t/N + 2 + 2k + 4rm/N + 4) \right\rceil + \frac{1}{3}k + t + 3 \\ & \leq \frac{4}{3}k + \frac{1}{6}m + 3t + 6 \leq m, \end{aligned}$$

where $N = 8r$, hold. The previous inequality would hold if

$$m - \frac{1}{6}m \geq \frac{4}{3}k + 3t + 6$$

which would hold if $m \geq 2k + 3t + 6$. Therefore, it can be chosen that $m_0 = 2k + 3t + 6$.

Finally, no active border signals appear in the time window $[t, (m-t)]$ because the activation signals originally found in the configuration c_u do not have enough time to meet with the border signals located in the word v. Similarly, the signals that change the border signals inactive in c_v do not have enough time to meet with the border signals located in the word u. This follows from the fact that the distance even from the "seam" location to either one of the words is $N(\ell_1 + 2\ell_2) > 4m$ in terms of the old cell structure where a type 2 activation signal (i.e. the fastest signal that possibly matters) travels with a speed of two cells per one time step. In short, no active border signals are generated in other locations than in the domains of u and v and none on the interval $[t, (m-t)]$ because the words are chosen to appear far enough from each other. □

Lemma 7.2.4. *Let $u \in C^*$ and $v \in C^*$ be two words of equal length k and assume that the question of Theorem 6.5.1 has a negative answer for the cellular automaton $(A^{\mathbb{Z}}, F)$ so that the value $t = \max(t_u^+, t_v^-)$ exists. Then there exists a positive integer n_0 and a configuration $c_n \in C^{\mathbb{Z}}$ for any positive integer $n \geq n_0$ such that*

1. *$c_n \in \mathrm{Cyl}(u, 0)$ and*
2. *$H^n(c_n) \in \mathrm{Cyl}(v, 0)$.*

Proof. Let c be the configuration given by Lemma 7.2.3 and which therefore depends on integer $m = n$. The configuration c_n is constructed by modifying configuration c by adding shift signals between the states in c. Let $\ell_1 =$

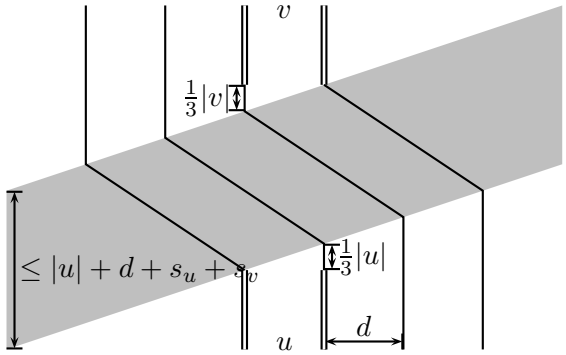


Figure 7.5: To have words u and v in the same position, at most $|u| + d + s_u + s_v$ left shift signals are required. This shift effect can be achieved in $|u| + d + s_u + s_v + \lceil \frac{1}{3} \max(|u|, |v|) \rceil$ time steps, where d is the distance between the occurrences of the word u and the word v without adding any shift signals to the initial configuration. Expressions s_u and s_v denote the number of shift signals contained in u and v, respectively.

$2\lceil t/N \rceil + k$, $\ell_2 = \lceil rn/N \rceil$ and $\ell = \ell_1 + 2\ell_2$, where k is the length of words u and v and r is the radius of the cellular automaton $(B^{\mathbb{Z}}, G)$. Let the number of shift signals contained in word u and v be s_u and s_v, respectively. The distance between the word occurrences is again $d = 2\ell_1 + 4\ell_2$.

An upper bound for the number s of left shift signals required to shift the word v to appear in the original domain of word u is given by condition

$$\begin{aligned} s &\leq k + d + s_u + s_v \\ &= k + 2\ell_1 + 4\ell_2 + s_u + s_v \\ &= k + 2(2\lceil t/N \rceil + k) + 4\lceil rm/N \rceil + s_u + s_v \\ &\leq k + 4t/N + 4 + 2k + 4rn/N + 4 + s_u + s_v \\ &\leq 5k + 4t/N + 4rn/N + 8. \end{aligned}$$

The number s of shift signals is restricted only by equation

$$n \geq t + \left\lceil \frac{1}{3}k \right\rceil + s + t$$

which follows from the time window enforced by the appearance of the active border signals as shown in Figure 7.5. The coefficient $\frac{1}{3}$ follows from the fact that a left shift signal has slope $\frac{1}{3}$. However, the equation holds if

$$n \geq t + k + 5k + 4t/N + 4rn/N + 8 + t = 6k + 2t + 4t/N + 4rn/N + 8.$$

By fixing the constant N to have value $8r$ as already in the proof of Lemma 7.2.3, it follows from the previous equation that the shift signals can be used

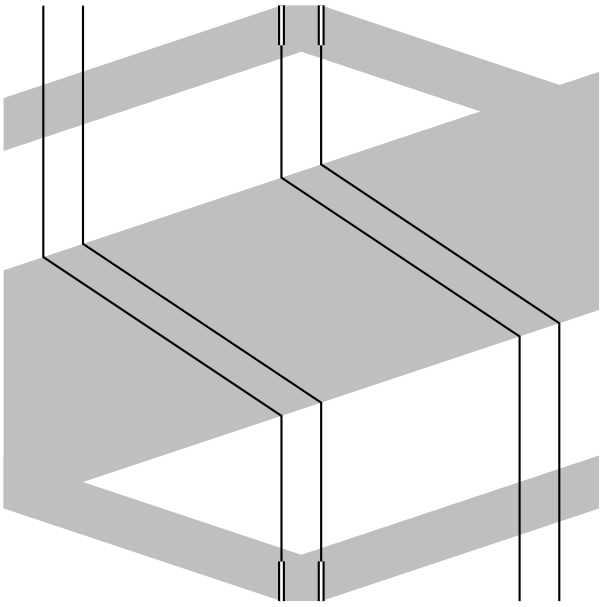


Figure 7.6: The possible locations (denoted by gray fill) of the shift signals in the computation with the initial configuration which is constructed in the proof of Lemma 7.2.4.

if $n \geq 12k + 6t + 16$ and if the constraint m_0 of Lemma 7.2.3 holds also. Now the bound n_0 is given by

$$n_0 = 12k + 6t + 8 \geq \max(12k + 6t + 16, 2k + 3t + 6).$$

For any $n \geq n_0$ the new configuration can be constructed by modifying the configuration given by Lemma 7.2.3 by adding sufficiently many left shift signals to the left of the cell in the location $-3t$ and rearranging the initial locations of the shift signals that will become part of the word v near the origin eventually. During the computation the shift signals will be located as shown in Figure 7.6. □

Theorem 7.2.5. *For reversible one-dimensional cellular automata, the sets of topologically mixing and non-sensitive cellular automata are recursively inseparable.*

Proof. Assume that the question of Theorem 6.5.1 has a negative answer for the cellular automaton $(A^{\mathbb{Z}}, F)$. Then, by Lemma 7.2.4, the cellular automaton $(C^{\mathbb{Z}}, H)$ is topologically mixing.

Assume that the answer to the question of Theorem 6.5.1 is positive. Then the blocking word constructed in the proof of Lemma 6.7.1 can be

modified to be used in the new cell structure of the cellular automaton $(C^{\mathbb{Z}}, H)$. The word remains blocking because shift signals simply turn away from states in C containing active border signals from B as vector elements. Hence, the cellular automaton is not sensitive to initial conditions. □

Corollary 7.2.6. *The following dynamical properties are undecidable for reversible one-dimensional cellular automata:*

1. *sensitivity to initial conditions,*
2. *topological mixing and*
3. *topological transitivity.*

Because a reversible cellular automaton has dense periodic points, the cellular automaton of Theorem 7.2.5 is chaotic if, and only if, it is transitive. Because transitivity was shown to be undecidable in the reversible case, the undecidability of Devaney's chaos follows.

Corollary 7.2.7. *It is undecidable whether or not a given reversible one-dimensional cellular automaton is chaotic according to Devaney.*

A cellular automaton is transitive if, and only if, it has a dense orbit. Transitivity was seen to be an undecidable property, so undecidability of Knudsen's chaos follows. In particular, in the case of reversible cellular automata, Devaney's and Knudsen's definitions of chaos are equivalent.

Corollary 7.2.8. *It is undecidable whether or not a given reversible one-dimensional cellular automaton is chaotic according to Knudsen.*

Chapter 8

Some results on linear cellular automata

In this chapter linear cellular automata are discussed. It is shown that there exists a close relationship between expansive and positively expansive linear cellular automata. In particular, a reversible linear cellular automaton function F is expansive if, and only if, the function $F + F^{-1}$ is positively expansive (see Corollary 8.5.2). It is also shown, for example, that expansivity and sensitivity are decidable properties for linear cellular automata. The dynamical properties have been studied earlier in a less general setting [43, 14, 75, 96, 76]. Here the linear cellular automata are considered with a more general definition of [12, 49].

8.1 Brief overview

A subclass of cellular automata is the family of linear cellular automata in which the global rule is linear, that is, a sum of the images of two configurations is equal to the image of the sum of the two configurations. The linear cellular automata whose state set is a finite commutative ring will be called linear cellular automata with a single state variable (SSLCA) and they can be represented with Laurent polynomials. A larger class of linear cellular automata is the one where the state sets are finite dimensional vector spaces over finite commutative rings. These cellular automata are also called linear cellular automata with multiple state variables (MSLCA). The multiple state variable can be represented with matrices over Laurent polynomials of the ring. The multiple state variable linear cellular automata were introduced independently in [12] and [49].

Unlike cellular automata in general, so far all the dynamical properties of linear cellular automata have turned out to be decidable. In fact, it has been shown that linear cellular automata cannot be universal [12]. For single

state variable linear cellular automata over $\mathbb{Z}_m$ many dynamical properties are decidable with easy to check criterions [43, 14, 75, 96, 76]. It seems that dynamical properties of multiple state variable linear cellular automata have not been studied except in articles [12] and [49]. For multiple state variable linear cellular automata nilpotency [12], surjectivity [49] and injectivity [12, 49] are known to be decidable properties.

In what follows it is shown that also expansivity, positive expansivity and sensitivity to initial conditions are decidable properties for multiple state variable linear cellular automata. The algorithms do not consist of simple criterions as in the case of linear cellular automata with a single state variable [76]. There even might not exist as simple criterions for different dynamical properties since linear cellular automata with multiple state variables can exhibit much more complex behavior than the linear cellular automata with a single state variable. For example, one-sided reversible cellular automata cannot exist in the single state variable case. Another example is that one-sided expansive (reversible) cellular automata do not exist in the single state variable case but they do exist in the multiple state variable case.

8.2 Finite commutative rings

The reader is assumed to have knowledge of basic algebra such as rings and fields.

The *characteristic* $\operatorname{char}(R)$ of ring R is the least positive integer such that $\operatorname{char}(R) \cdot a = \sum_{i=1}^{\operatorname{char}(R)} a = 0_R$. If such an integer does not exist, the characteristic is defined to be $\operatorname{char}(R) = 0$.

An element $a \in R$ is called *nilpotent*, if for some positive integer n equation $a^n = 0_R$ holds. Similarly, a set $A \in R$ is called *nilpotent* if there exists a positive integer n such that $A^n = \{0_R\}$.

The *nilpotency* $\operatorname{nil}(a)$ of a nilpotent element $a \in R$ is the least positive integer for which $a^{\operatorname{nil}(a)} = 0_R$. Similarly, the nilpotency $\operatorname{nil}(A)$ of a nilpotent set A is the least positive integer with which $A^{\operatorname{nil}(A)} = \{0_R\}$.

A *local* ring is a commutative ring with identity having a unique maximal ideal [41]. There exists a broader definition for a local ring in the case of non-commutative rings [79, p. 82]. In the case of commutative rings, alternative definitions of local rings can be found in [87, 44]. However, in this work a local ring is simply a commutative ring with identity having a unique maximal ideal.

Theorem 8.2.1 ([79]). *Let R be a finite commutative ring with identity $1_R \neq 0_R$. The ring R has a unique decomposition*

$$R = L_1 \oplus \ldots \oplus L_n$$

into a direct sum of local rings.

Moreover, there exists orthogonal idempotents e_i in R with $\sum_{i=1}^{m} e_i = 1_R$ and $L_i L_j = \{0_R\}$ for every $i \neq j$ [79].

In a finite commutative local ring with identity the zero-divisors are all nilpotent and they form the maximal ideal [49, 78].

8.3 Linear cellular automata

The following definition of linear cellular automata is a generalization of the one found in [12]. The only difference is that the state set can consist of vectors over a finite commutative ring with identity and not just a finite field. Earlier, linear cellular automata have been defined, for example, only with a state set $\mathbb{Z}_m$ [43]. The linear cellular automata defined with a state set consisting of vectors over a field or a commutative ring were independently introduced in [12] and [49].

Definition 8.3.1. *An m-dimensional cellular automaton $(A^{\mathbb{Z}^m}, F)$ is said to be* linear*, if*

1. *the state set $A = R^n$ is the set of n-element vectors over a finite commutative ring R with identity and*

2. *the function F is defined by*

$$F(c)(\overrightarrow{x}) = A_1 c(\overrightarrow{x} + \overrightarrow{x}_1) + \cdots + A_l c(\overrightarrow{x} + \overrightarrow{x}_l),$$

   where $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l \in R^m$ are the neighborhood vectors of length m and $A_1, \ldots, A_l \in \mathcal{M}_{n\times n}(R)$.

Following [49], a linear cellular automaton is said to be a *multiple state variable linear cellular automaton* (MSLCA) if the state set is not a finite commutative ring with identity, that is, $n \geq 2$ in Definition 8.3.1. A linear cellular automaton is said to be a *single state variable linear cellular automaton* (SSLCA) if the state set is simply the finite commutative ring with identity, that is, $n = 1$ in Definition 8.3.1. No distinction was made depending on the size of the state vectors in [12].

Definition 8.3.2. *A* Laurent polynomial *$f(x_1, \ldots, x_m)$ of m variables $x_1, \ldots, x_m$ over a ring R is*

$$f(x_1, \ldots, x_m) = \sum_{|i_j| \leq k} a_{i_1,\ldots,i_m} x_1^{i_1} \cdots x_m^{i_m},$$

where $a_{i_1,\ldots,i_m} \in R$ and k is a finite bound. The set of Laurent polynomials of m variables $x_1, \ldots, x_m$ over ring R is denoted by $R[x_1, x_1^{-1}, \ldots, x_m, x_m^{-1}]$.

Definition 8.3.3. *A* Laurent series $f(x_1, \ldots, x_m)$ *of* m *variables over a ring* R *is*

$$f(x_1, \ldots, x_m) = \sum_{i_j \in \mathbb{Z}} a_{i_1,\ldots,i_m} x_1^{i_1} \cdots x_m^{i_m},$$

where $a_{i_1,\ldots,i_m} \in R$. *The set of Laurent polynomials of* m *variables* $x_1, \ldots, x_m$ *over ring* R *is denoted by* $R[[x_1, x_1^{-1}, \ldots, x_m, x_m^{-1}]]$.

Notation 8.3.4. *For brevity, if* $X = \{a_1, \ldots, a_m\}$, $R[a_1, \ldots, a_m]$ *will also be denoted by* $R[X]$. *Likewise, if* $X = \{a_1, \ldots, a_m\}$, *then* $R[[a_1, \ldots, a_m]]$ *will also be denoted by* $R[[X]]$.

If $X = \{x\}$, then $R[X]$ is used to denote a polynomial ring of a single variable x. If $X = \{x, x^{-1}\}$, then $R[X]$ denotes a Laurent polynomial ring of a single variable x. If R is commutative, then $R[x_1, x_1^{-1}, \ldots, x_m, x_m^{-1}]$ is an infinite commutative ring.

It is straightforward to see, that a configuration of an m-dimensional linear cellular automaton of n state variables from a finite commutative ring R with identity can be represented by a Laurent series of m variables with coefficients from R^n. Then the global function of the cellular automaton can be interpreted as a linear transformation or a matrix of Laurent polynomials whose domain is the set of Laurent series vectors.

Let $M \in \mathcal{M}_{n\times n}(R[X])$ be matrix which represents an m-dimensional linear cellular automaton with state set R^n. Then F_M is used to denote the cellular automaton function defined by matrix M, that is, the cellular automaton in question is $((R^n)^{\mathbb{Z}^m}, F_M)$.

Definition 8.3.5. *A cellular automaton function* $F : (A^n)^{\mathbb{Z}^m} \to (A^n)^{\mathbb{Z}^m}$ *is said to have a* one-sided *neighborhood with respect to coordinate* i *if either* $\overrightarrow{x}_j \in \mathbb{Z}^{i-1} \times \{0, 1, \ldots\} \times \mathbb{Z}^{m-i}$ *or* $\overrightarrow{x}_j \in \mathbb{Z}^{i-1} \times \{0, -1, \ldots\} \times \mathbb{Z}^{m-i}$ *for every vector* $\overrightarrow{x}_j$ *in the neighborhood of the local rule.*

Definition 8.3.6. *A cellular automaton function* $F : (A^n)^{\mathbb{Z}^m} \to (A^n)^{\mathbb{Z}^m}$ *is said to have a* strictly one-sided *neighborhood with respect to coordinate* i *if either* $\overrightarrow{x}_j \in \mathbb{Z}^{i-1} \times \{1, 2, \ldots\} \times \mathbb{Z}^{m-i}$ *or* $\overrightarrow{x}_j \in \mathbb{Z}^{i-1} \times \{-1, -2, \ldots\} \times \mathbb{Z}^{m-i}$ *for every vector* $\overrightarrow{x}_j$ *in the neighborhood of the local rule.*

That is, a cellular automaton function has a one-sided neighborhood with respect to the ith coordinate if either all the effective neighbors of a cell are located to the left or all the effective neighbors of a cell are located to the right from the origin along the ith coordinate axis.

Notation 8.3.7. *Let* $X = \{x_1, x_1^{-1}, \ldots, x_m, x_m^{-1}\}$. *Let* $f(x_1, \ldots, x_m) \in R[X]$ *be a non-zero Laurent polynomial and let* $x \in X$.

Then $\deg_x f(x_1, \ldots, x_m)$ *is used to denote the smallest integer* k *such that* $x^{-k} f(x_1, \ldots, x_m) \in R[X \setminus \{x\}]$.

Notation 8.3.8. *For brevity, a notation $X[m]$ is used to refer to the set of m variables $\{x_1, , \ldots, x_m\}$ of a polynomial and a notation $X(n)$ is used to refer to the set of m variables $\{x_1, x_1^{-1}, \ldots, x_m, x_m^{-1}\}$ of a Laurent polynomial.*

The dynamical properties of linear cellular automata with a single state variable have been widely studied (see for example [43, 14, 75, 96, 76]), but it seems that multiple state variable linear cellular automata have not been studied except in articles [12] and [49].

For linear cellular automata, injectivity [12, 49] and surjectivity [49] can be determined by studying the determinant of the matrix which defines the function. Nilpotency of a linear cellular automaton can be determined by computing n iterations of the function or by determining the characteristic equation (which will be equal to $M^n = 0$ if, and only if, M is nilpotent) [12].

Example: An expansive one-sided linear cellular automaton There exists no one-sided expansive rules in the case of single state variable linear cellular automata over $\mathbb{Z}_m$. However, for multiple state variable linear cellular automata there exists. Let $M = \begin{pmatrix} x^{-1} & 1+x^{-1} \\ 1 & 1 \end{pmatrix}$. Then F_M is a reversible one-dimensional cellular automaton over state set $\mathbb{Z}_2^2$. Then $M^{-1} = \begin{pmatrix} 1 & 1+x^{-1} \\ 1 & x^{-1} \end{pmatrix}$ modulo 2. Therefore, $(\mathbb{Z}_2^{2^{\mathbb{N}}}, F_M)$ is a one-sided reversible cellular automaton. The sum of the two matrices is $M + M^{-1} = \begin{pmatrix} 1+x^{-1} & 0 \\ 0 & 1+x^{-1} \end{pmatrix}$ modulo 2. The characteristic polynomial of $M + M^{-1}$ is

$$\chi_{M+M^{-1}}(\lambda) = \lambda^2 + (1 + x^{-2}).$$

By Theorem 8.5.6, the cellular automaton $(\mathbb{Z}_2^{2^{\mathbb{N}}}, F_{M+M^{-1}})$ is positively right expansive. By Theorem 8.5.1, the cellular automaton $(\mathbb{Z}_2^{2^{\mathbb{N}}}, F_M)$ is right expansive, so it is an expansive one-sided cellular automaton.

8.4 Some technical lemmas

In this section some technical lemmas will be represented. These lemmas will be used later to study the linear cellular automata and linear combinations of their iterations.

The following result was already used in [75, 76]:

Lemma 8.4.1. *Let integer n be a multiple of $m!k^m$ for some integer k. Then $\binom{n}{i}$ is divisible by k^m for every $0 < i < m$.*

Proof. Let j be a positive integer so that $n = m!k^m j$. Because $m!k^m - i = i\left(\frac{m!}{i}k^m - 1\right)$, any binomial coefficient

$$\begin{aligned}\binom{m!k^m j}{i} &= \frac{(m!k^m j)!}{(m!k^m j - i)!(i!)} = \frac{m!k^m j \cdots (m!k^m j - i + 1)}{i \cdots 1} \\ &= \frac{m!k^m j}{i} \cdot \frac{m!k^m j - 1}{1} \cdot \frac{m!k^m j - 2}{2} \cdots \frac{m!k^m j - i + 1}{i - 1}\end{aligned}$$

is divisible by k^m whenever $0 < i < m$. □

Recall, that $N(R)$, $U(R)$ and $Z(R)$ denote the sets of nilpotent elements, units and zero-divisors, respectively, of a ring R.

Lemma 8.4.2. *Let L be a finite commutative local ring with identity. Let $f(x) \in L[x] \setminus N(L)[x]$. Then for some positive integer n*

$$f(x)^n = \sum_{i=0}^{k} a_i x^i,$$

where $a_k \in U(L)$.

Proof. Assume that $f(x) = \sum_{i=0}^{l} b_i x^i$, where $b_j \in U(L)$ for some j and $b_i \in N(L)$ for $i > j$. Denote $A = \sum_{i=0}^{j} b_i x^i$ and $B = \sum_{i=j+1}^{l} b_i x^i$. Because L is local, $N(L)$ is the set of zero-divisors. Choose $n = \operatorname{nil}(N(L))! q^{\operatorname{nil}(N(L))}$, where q is the characteristic of L. Then, by Lemma 8.4.1,

$$f(x)^n = (A + B)^n = A^n + \sum_{i=\operatorname{nil}(N(L))}^{n} \binom{n}{i} A^{n-i} B^i = A^n,$$

because $B^i = 0$ when $i \geq \operatorname{nil}(N(L))$. Now $f(x)^n = A^n$ is of the desired form. □

Lemma 8.4.3. *Let R be a finite commutative ring with identity and $M \in \mathcal{M}_{n\times n}(R[X(m)])$. Let $K = |R|^{n(2rn+3)^m}$. There exists such integers N_1 and N_2 that $0 \leq N_1 < N_2 < K$ and*

$$M' = M^{N_2} - M^{N_1} = \sum_{i=0}^{2m} M_i,$$

where matrices M_i commute and for every i there exist an integer j so that either

$$M_i \in \mathcal{M}_{n\times n}(x_j^{-1} R[X(m) \setminus \{x_j\}]) \text{ or } M_i \in \mathcal{M}_{n\times n}(x_j R[X(m) \setminus \{x_j^{-1}\}]).$$

Proof. Let the characteristic equation of M be $\chi_M(\lambda) = \lambda^n - \sum_{i=0}^{n-1} a_i\lambda^i$, where $a_i \in F[X(m)]$. Every power λ^k of λ greater than λ^{n-1} can be expressed as a decomposition (which is also unique)

$$\lambda^k = \sum_{i_1,\ldots,i_m\in\mathbb{Z}} f_{k,(i_1,\ldots,i_m)}(\lambda)x_1^{i_1}\cdots x_m^{i_m}, \tag{8.1}$$

where $f_{k,(i_1,\ldots,i_m)}(\lambda) \in F[\lambda] \setminus \lambda^n F[\lambda]$.

For some two distinct powers λ^{N_2} and λ^{N_1} the decompositions (8.1) agree on the coefficients of terms $x_1^{i_1}\cdots x_m^{i_m}$, where $-(rn+1) \leq i_j \leq (rn+1)$ and $1 \leq j \leq m$. Then the coefficients of the same terms in the decomposition of $\lambda^{N_2} - \lambda^{N_1}$ are all equal to zero. That is,

$$\lambda^{N_2} - \lambda^{N_1} = \sum_{i_1,\ldots,i_m\in\mathbb{Z}} g_{(i_1,\ldots,i_m)}(\lambda)x_1^{i_1}\cdots x_m^{i_m}, \tag{8.2}$$

where $g_{(i_1,\ldots,i_m)}(\lambda) \in R[\lambda]\setminus\lambda^n R[\lambda]$ and if $-(rn+1) \leq i_j \leq (rn+1)$ whenever $1 \leq j \leq m$, then $g_{i_1,\ldots,i_m}(\lambda) = 0$. Because the set of indices given by constraints $-(rn+1) \leq i_j \leq (rn+1)$ and $1 \leq j \leq m$ is finite, the integers N_1 and N_2 can be chosen to satisfy $0 \leq N_1 < N_2 < K$. This is actually a simple application of the pigeon hole principle.

Finally, because all elements of matrix M^{n-1} are at most of degree $r(n-1)$ (with respect to any $x \in X(m)$), the claim follows by substituting M for λ in equation (8.2). All the terms $g_{(i_1,\ldots,i_m)}(M)x_1^{i_1}\cdots x_m^{i_m}$ commute and either

$$g_{(i_1,\ldots,i_m)}(M)x_1^{i_1}\cdots x_m^{i_m} \in \mathcal{M}_{n\times n}(x_j^{-1}R[X(m)\setminus\{x_j\}])$$

or

$$g_{(i_1,\ldots,i_m)}(M)x_1^{i_1}\cdots x_m^{i_m} \in \mathcal{M}_{n\times n}(x_jR[X(m)\setminus\{x_j^{-1}\}]).$$

The terms can be summed in a suitable way to produce matrices M_i which also commute. □

In the one-dimensional case the decomposition of commuting matrices simplifies into a sum of two matrices.

Corollary 8.4.4. *Let R be a finite commutative ring with identity and let $M \in \mathcal{M}_{n\times n}(R[x,x^{-1}])$. Let $K = |R|^{n(2rn+3)}$. There exists such integers N_1 and N_2 that $0 \leq N_1 < N_2 < K$ and*

$$M' = M^{N_2} - M^{N_1} = M_1 + M_2,$$

where matrices M_1 and M_2 commute,

$$M_1 \in \mathcal{M}_{n\times n}(xR[x]) \text{ and } M_2 \in \mathcal{M}_{n\times n}(x^{-1}R[x^{-1}]).$$

Lemma 8.4.5. *Let R be a finite field and let $M \in \mathcal{M}_{n\times n}(xR[x])$ represent a cellular automaton with strictly one-sided neighborhood. If the cellular automaton $((R^n)^{\mathbb{Z}}, F_M)$ is surjective, then all the column subshits are of finite type of the second order.*

Proof. Let the state set of the cellular automaton be denoted by $A = R^n$. Let the radius of the local rule be denoted r. Let $c \in A^{\mathbb{Z}}$ be any configuration. Since the local rule is strictly one-sided, word $F_M(c)[0, k]$ is determined uniquely by word $c[-r, k]$.

Let i be a non-negative integer. Let c be any configuration such that $F_M^i(c)[-|u| + 1, 0] = u$. Let u and v be such words (of equal length) that there exists a configuration c' such that

$$F_M^i(c')[-|u| + 1, 0] = u \text{ and } F_M^{i+1}(c')[-|u| + 1, 0] = v.$$

Clearly, v depends only on $F_M^i(c')[-|u| + 1 - r, 0]$. Because the local rule is surjective, strictly one-sided and linear, for every word w and integer i there exists a configuration $c_{w,i}$ such that $F_M^i[-|w| + 1 - r, 0 - r] = w$ and $F_M^j(k) = \mathbf{0}$ for all integers $j \leq i$ and $k \geq -r$. By linearity, assume that $w = F_M^i(c')[-|u|+1-r, 0-r]-F_M^i(c)[-|u|+1-r, 0-r]$ (where subtraction is done cell-wise). Then $c'' = c + c_{w,i}$ (where addition is done cell-wise) is a configuration such that $F_M^j(c'')[-|u| + 1, 0] = F_M^j(c)[-|u| + 1, 0]$ for every $j \leq i$ and $F_M^{i+1}(c'')[-|u| + 1, 0] = F_M^{i+1}(c')[-|u| + 1, 0] = v$. This means that to make the word appear after any sequence of i words, it is sufficient to modify the cells to the left, which is possible due to linearity. Hence, the column subshift of width $|u|$ is of the second order. □

8.5 Expansivity

In this section it is shown that expansivity and positive expansivity are decidable properties for linear cellular automata whose states sets are sets of vectors over finite commutative rings with identity.

The following Theorem 8.5.1 states that among linear cellular automata there is a strong relationship between expansive and positively expansive cellular automata. In particular, it follows that among linear cellular automata expansivity can be algorithmically reduced to positive expansivity.

Please note, that in the following elements of set $L + N(L)[\lambda]$ are polynomials of the form

$$f(\lambda) = \sum_{i=0}^{k} a_i \lambda^i,$$

where $a_i \in N(L)$ whenever $i > 0$. Then the elements of $L[\lambda] \setminus (L + N(L)[\lambda])$ are of the form

$$f(\lambda) = \sum_{i=0}^{k} a_i \lambda^i,$$

where $a_j \in U(L)$ for some $j > 0$.

Theorem 8.5.1. *Let L be a finite commutative local ring with identity. Let $M \in \mathcal{M}_{n\times n}(L[x, x^{-1}])$ and suppose that $((L^n)^{\mathbb{Z}}, F_M)$ is a reversible cellular automaton. The following conditions are equivalent:*

1. *The cellular automaton $((L^n)^{\mathbb{Z}}, F_M)$ is left (right) expansive.*
2. *For some two polynomials $f(\lambda), g(\lambda) \in L[\lambda] \setminus (L + N(L)[\lambda])$ the cellular automaton given by matrix $N = f(M) + g(M^{-1})$ is positively left (right) expansive.*
3. *For any two polynomials $f(\lambda), g(\lambda) \in L[\lambda] \setminus (L + N(L)[\lambda])$ the cellular automaton given by matrix $N = f(M) + g(M^{-1})$ is positively left (right) expansive.*

Proof. Let $f(\lambda)$ and $g(\lambda)$ be any two polynomials satisfying the assumptions. Denote $f(\lambda) = \sum_{i=0}^{d_f} a_i \lambda^i$ and $g(\lambda) = \sum_{i=0}^{d_g} b_i \lambda^i$, where it is possible to assume that $a_{d_f}, b_{d_g} \in U(R)$ by Lemma 8.4.2. For clarity, denote $h(\lambda) = f(\lambda) + g(\lambda^{-1})$.

Assume that $((L^n)^{\mathbb{Z}}, F_M)$ is not left expansive. Then there exists a configuration $c(x) \in L[[x^{-1}]]^n \setminus x^{-1}L[[x^{-1}]]^n$ such that $M^k c(x) \in L[[x^{-1}]]^n$ for all $k \in \mathbb{Z}$. Then also $h(M)^k c(x) \in L[[x^{-1}]]^n$ for all $k \in \mathbb{N}$, and $((L^n)^{\mathbb{Z}}, F_{h(M)})$ cannot be positively left expansive.

Assume that $((L^n)^{\mathbb{Z}}, F_{h(M)})$ is not positively left expansive. Then there exists a configuration $c(x) \in L[[x^{-1}]]^n \setminus x^{-1}L[[x^{-1}]]^n$ such that $h(M)^k c(x) \in L[[x^{-1}]]^n$ for all $k \in \mathbb{N}$. Next it will be proven that also

$$M^k c(x) \in x^{r \max(d_f, d_g)} L[[x^{-1}]]^n \text{ for all } k \in \mathbb{Z}.$$

That is, the same configuration $c(x) \in L[[x^{-1}]]^n \setminus x^{-1}L[[x^{-1}]]^n$ will prove that F_M is not left expansive.

It will be proven by induction that a matrix M^i can be presented as a sum of the form

$$M^i = \sum_{j=0}^{J_i} \sum_{k=-d_g}^{d_f} a_{j,k} M^k h(M)^j, \text{ where } a_{j,k} \in L \text{ and } J_i \in \mathbb{N}. \tag{8.3}$$

Clearly, the decomposition of Equation (8.3) exists for $-d_g \leq i \leq d_f$.

Consider the positive powers M^i where $i > d_f$. Then

$$M^i = a_{d_f}^{-1} \left(M^{i-d_f} h(M) - (h(M) - a_{d_f} M^{d_f}) M^{i-d_f} \right).$$

By the inductive hypothesis M^{i-d_f} is of the form given in Equation (8.3). Then the former term $M^{i-d_f} h(M)$ is of the form given in Equation (8.3). Because the Laurent polynomial $h(M) - a_{d_f} M^{d_f}$ contains terms of degree $d_f - 1$ at most, then the Laurent polynomial $(h(M) - a_{d_f} M^{d_f}) M^{i-d_f}$ contains terms of degree $i-1$ at most. Furthermore, the Laurent polynomial $(h(M) - a_{d_f} M^{d_f}) M^{i-d_f}$ contains terms of degree $i - d_f - d_g > -d_g$ at least. Therefore, the latter term is of the form given in Equation (8.3). Hence, M^i can be decomposed into the form given in Equation (8.3) when $i > d_f$.

The case for $i < -d_g$ is similar.

Now it can be concluded that every matrix M^i, where $i \in \mathbb{Z}$, can be presented in the form given in Equation (8.3).

Finally, it follows that

$$M^i c(x) \in x^{r \max(d_f, d_g)} L[[x^{-1}]]^n \text{ for all } i \in \mathbb{N}$$

using Equation (8.3) and the assumption that $h(M)^i c(x) \in L[[x^{-1}]]^n$ for all $i \in \mathbb{N}$, where r is the maximum of the radii of the local rules of the cellular automata $((L^n)^{\mathbb{Z}}, F_M)$ and $((L^n)^{\mathbb{Z}}, F_M^{-1})$. That is, no non-zero states propagate beyond cell $r \max(d_f, d_g)$ because $((L^n)^{\mathbb{Z}}, F_{h(M)})$ is not positively left expansive. Hence, the cellular automaton $((L^n)^{\mathbb{Z}}, F_M)$ cannot be left expansive.

Because both polynomials $f(\lambda)$ and $g(\lambda)$ were arbitrary, it was shown that all the conditions 1, 2 and 3 are equivalent. □

Corollary 8.5.2. *Let L be a finite commutative local ring with identity and let $M \in \mathcal{M}_{n \times n}(L[x, x^{-1}])$. Suppose that M defines a reversible cellular automaton. The following conditions are equivalent:*

1. *The cellular automaton given by M is left (right) expansive.*
2. *For some two elements $a, b \in U(L)$ the cellular automaton given by $aM + bM^{-1}$ is positively left (right) expansive.*
3. *For any two elements $a, b \in U(L)$ the cellular automaton given by $aM + bM^{-1}$ is positively left (right) expansive.*

The following lemma states the a linear cellular automaton is positively expansive if, and only if, any non-trivial linear combination of its powers is positively expansive.

Lemma 8.5.3. *Let L be a finite commutative local ring with identity, $M \in \mathcal{M}_{n\times n}(L[x, x^{-1}])$ and $f(\lambda) \in L[\lambda] \setminus (L + N(L)[\lambda])$. The cellular automaton given by M is positively left (right) expansive if, and only if, the cellular automaton given by $f(M)$ is positively left (right) expansive.*

Proof. By Lemma 8.4.2, it is possible to assume that the coefficient of the highest term is a unit.

First, assume that M is positively left expansive. Denote $C_n(x) = (L[[x]] \setminus xL[[x]])^n$ and let

$$h = \min\left\{m \in \mathbb{N} \mid \forall c(x) \in C_n(x^{-1}) : M^m c(x) \notin (L[[x^{-1}]])^n\right\}.$$

The finite constant h exists because M defines a positively left expansive cellular automaton. Let the characteristic of L be q. Let m be a positive integer and $k = (hm)!q^{hm}$. Let the degree of $f(\lambda)$ be d_f Then $f(M)^k = \sum_{i=0}^{kd_f-hm} c_i M^i + c_{kd_f} M^{kd_f}$ (for some coefficients c_i), that is, the highest terms in the polynomial are $c_j M^j$ (for some $j \leq kd_f - hm$) and $c_{kd_f} M^{kd_f}$. By positive left expansivity, for every configuration $c(x) \in C_n(x^{-1})$ there are non-zero states in configuration $c_{kd_f} M^{kd_f} c(x)$ further to the right than in configuration $c_j M^j c(x)$. This means that the rightmost non-zero states caused by the highest term $c_{kd_f} M^{kd_f} c(x)$ cannot be cancelled out by any of the lower terms $c_i M^i c(x)$, $i \leq j$. Therefore, $f(M)$ must define a positively left expansive cellular automaton.

Second, assume that M is not positively left expansive. Then there exists such $c(x) \in (L[[x^{-1}]])^n$ that $M^l c(x) \in (L[[x^{-1}]])^n$ for any $l \geq 0$. Then also $f(M)^l c(x) \in (L[[x^{-1}]])^n$ for any $l \geq 0$. Then $f(M)$ cannot be positively left expansive.

The result follows for positive right expansivity in a similar manner. □

Theorem 8.5.4 ([63]). *Positive expansivity, equicontinuity and nilpotency are decidable properties for regular cellular automata*

The previous theorem by Di Lena [63] can be strengthened to produce the following lemma:

Lemma 8.5.5. *Positive left expansivity and positive right expansivity are decidable properties for regular cellular automata.*

The proof is essentially the same as in [63].

Proof. Assuming that the cellular automaton $(A^{\mathbb{Z}}, F)$ is regular, the graph which describes the column subshift $\Sigma_k(F)$ (for any $k > 0$) can be algorithmically constructed [63].

Let $k = 2r + 1$ where r is the radius of the local rule of the function F. Recall that the graph contains the (infinite path) $p \in \left(A^k\right)^{\mathbb{N}}$ if, and only if,

$p \in \Sigma_k(F)$. Now the cellular automaton is left expansive if, and only if, there are no two such paths $p_1 \in \left(A^k\right)^{\mathbb{N}}$ and $p_1 \in \left(A^k\right)^{\mathbb{N}}$ that $p_1(0) \neq p_2(0)$ but $p_1(i)(j) = p_2(i)(j)$ for every $i \geq 0$ and $j > 0$. That is, the cellular automaton is left expansive if, and only if there are no two such paths the (state vector) nodes of the paths agree on every component except on the first one. If there were, it would mean that the difference in the first component might not propagate to the right and the rule would not be left expansive.

Right expansivity can be determined in a similar manner. □

The following Theorem 8.5.6 gives an easy method for generating expansive cellular automata. That is, choosing suitable Laurent polynomials for coefficients for a polynomial $p(\lambda) \in F[x, x^{-1}][\lambda]$ guarantees that the companion matrix of $p(\lambda)$ represents a positively expansive cellular automaton.

Theorem 8.5.6. *Let F be a finite field and let $M \in \mathcal{M}_{n\times n}(F[x, x^{-1}])$. Suppose that for an integer $k \in \mathbb{N}$ matrix M can be represented by a relation*

$$M^k = \sum_{i=0}^{k-1} f_i(x)M^i, \tag{8.4}$$

where $f_i(x) \in F[x, x^{-1}]$ for every i. If $\deg_x(f_0(x)) > \deg_x(f_j(x))$ for every j, $0 < j < k$, then $((F^n)^{\mathbb{Z}}, F_M)$ is positively left expansive.

Proof. Suppose that matrix M would not define a positively left expansive cellular automaton. Then there exists a configuration $c(x) \in (F[[x^{-1}]] \setminus x^{-1}F[[x^{-1}]])^n$ such that $M^m c(x) \in (F[[x^{-1}]])^n$ for all $m \in \mathbb{N}$. By assumption, $M^k = \sum_{i=1}^{k-1} f_i(x)M^i + f_0(x)$. Let p be the characteristic of field F and let h be a positive integer. Now

$$\left(f_i(x)M^i\right)^{p^h} c(x) \in (x^{p^h \deg_x f_i(x)}F[[x^{-1}]])^n \text{ for all } 0 < i < k$$

and

$$f_0(x)^{p^h} Ic(x) \notin (x^{p^h \deg_x f_i(x)}F[[x^{-1}]])^n \text{ for any } i \neq 0,$$

This means that the the non-zero states propagate further to the right with multiplication by $f_0(x)^{p^h} I$ than with multiplication by any higher term $f_i(x)M^i$. Therefore, the linear combination (8.4) of the smaller powers of M produces a positively left expansive cellular automaton. But this is not possible, because M was assumed not to define a positively left expansive cellular automaton. □

Lemma 8.5.7. *Let R be a finite commutative ring with identity and let*

$$M = \begin{pmatrix} a(x) & b(x) \\ c(x) & d(x) \end{pmatrix} \in \mathcal{M}_{2\times 2}(R[x, x^{-1}])$$

and let $\det M = 1$. *Then* $(\left(R^2\right)^{\mathbb{Z}}, F_M)$ *is left (right) expansive if, and only if, the Laurent polynomial* $a(x)+d(x) \in R[x, x^{-1}]$ *represents a positively left (right) expansive linear cellular automaton.*

Proof. Because $\det M = 1$,

$$M^{-1} = \begin{pmatrix} d(x) & -b(x) \\ -c(x) & a(x) \end{pmatrix}.$$

Now

$$M + M^{-1} = \begin{pmatrix} a(x) + d(x) & 0 \\ 0 & a(x) + d(x) \end{pmatrix}.$$

The cellular automaton defined by this matrix is a direct product of identical single state variable linear cellular automata. By Theorem 8.5.1, the cellular automaton $(\left(R^2\right)^{\mathbb{Z}}, F_M)$ is left (right) expansive if, and only if, the Laurent polynomial $a(x) + d(x)$ represents positively left (right) expansive linear cellular automaton which happens when $\deg_x(a(x) + d(x)) > 0$ (or $\deg_{x^{-1}}(a(x) + d(x)) > 0$ for right expansive rules). □

Theorem 8.5.8. *Let* R *be a finite commutative ring with identity and let* $M \in \mathcal{M}_{n\times n}(R[x, x^{-1}])$. *It is decidable whether or not the cellular automaton given by a non-invertible matrix* M *is positively left (right) expansive.*

Proof. The property needs to be tested for every local ring in the decomposition of the ring.

By Corollary 8.4.4, a linear combination N of some powers of M can be expressed as a sum $N = M_1 + M_2$ of two commuting matrices $M_1 \in \mathcal{M}_{n\times n}(xR[x])$ and $M_2 \in \mathcal{M}_{n\times n}(x^{-1}R[x^{-1}])$ with strictly one-sided neighborhoods. By Lemma 8.5.3, M is positively left (right) expansive if, and only if, N is positively left (right) expansive.

By Lemma 8.4.5, both of the matrices M_1 and M_2 represent regular cellular automata. Now the cellular automaton given by M is positively left expansive if, and only if, matrix M_1 is positively left expansive. Likewise, the cellular automaton given by M is positively right expansive if, and only if, matrix M_2 is positively right expansive. Because positive left expansivity and positive right expansivity are decidable properties for regular cellular automata, decidability of positive left expansivity and positive right expansivity follows. □

Theorem 8.5.9. *Let* R *be a finite commutative ring with identity and let* $M \in \mathcal{M}_{n\times n}(R[x, x^{-1}])$. *It is decidable whether or not the cellular automaton* $((R^n)^{\mathbb{Z}}, F_M)$ *given by an invertible matrix* M *is left (right) expansive.*

Proof. The property needs to be tested for every local ring in the decomposition of the ring.

The claim follows by Corollary 8.5.2 and Theorem 8.5.8. □

8.6 Sensitivity to initial conditions

In this section it is shown that sensitivity to initial conditions is a decidable property for linear cellular automata whose states sets are sets of vectors over finite commutative rings with identity.

Again, the results are first represented in the case of a finite field and later generalized to the case of a finite commutative ring with identity.

The following lemma states the obvious fact that a matrix defines a linear cellular automaton, which is sensitive to initial conditions,

Lemma 8.6.1. *Let R be a finite commutative ring with identity and $M \in \mathcal{M}_{n\times n}(R[X(m)])$. Then the m-dimensional linear cellular automaton given by matrix M is sensitive to initial conditions if, and only if, it is not equicontinuous.*

Proof. Clearly, if the powers of the variables do not grow arbitrarily large in the powers of M then the matrix is periodic and hence the rule must be equicontinuous.

Also, if the powers of the variables do grow arbitrarily large in the powers of M, then some unit vector of the basis acts as a non-zero configuration from which non-zero states propagate arbitrarily far. Then the rule must be sensitive. □

The following lemma states that determining the equicontinuity status of a linear cellular automaton can be reduced to determining the equicontinuity status of any non-trivial linear combination of its powers.

Lemma 8.6.2. *Let F be a finite field, $M \in \mathcal{M}_{n\times n}(F[X(m)])$ and $f(\lambda) \in F[\lambda]\setminus F$. Then M defines an equicontinuous cellular automaton if, and only if, $f(M)$ defines an equicontinuous cellular automaton.*

Proof. Denote $f(\lambda) = \sum_{i=0}^{k} a_i\lambda^i$. Assume that M is equicontinuous. Then matrix M is ultimately periodic which means that there exists such non-negative integers p_0 and p that $M^{p_0} = M^{p_0+kp}$ for every positive integer k. Then there exists an upper bound on the degrees of Laurent polynomials in the powers of M. Because the powers of M commute, the same bound eventually restricts the elements of matrix $f(M)$ which is a linear combination of some powers of M.

Assume that M is sensitive. Then matrix M is not ultimately periodic. Then for some $f(x_1,\ldots,x_m) \in (F[[X(m)\setminus\{x\}]])^n$, some $x \in X(m)$ and every positive integer s there exists a positive integer t such that

$$M^l f(x_1,\ldots,x_m) \in (F[X(m)])^n \setminus (x^s F[[X(m)\setminus\{x\}]])^n,$$

if $l = t$ and

$$M^l f(x_1,\ldots,x_m) \in (x^s F[[X(m)\setminus\{x\}]])^n,$$

if $l < t$. That is, t is the moment when non-zero states starting from the origin spread beyond cell s with respect to the coordinate axis of variable $x \in X$. Let h be maximal so that $t = kp^h + u$, where p is the characteristic of F. Let $g(x_1, \ldots, x_m) = M^u f(x_1, \ldots, x_m)$. Then

$$\left(a_k M^k\right)^{p^h} g(x_1, \ldots, x_m) \in (F[[X(m)]])^n \setminus (x^s F[[X(m) \setminus \{x\}]])^n$$

and

$$\left(a_i M^i\right)^{p^h} g(x_1, \ldots, x_m) \in (x^s F[[X(m) \setminus \{x\}]])^n \text{ when } i < k.$$

That is, with configuration $g(x_1, \ldots, x_m)$ the consecutive powers of M do not cancel each other out. Therefore, for configuration $g(x_1, \ldots, x_m)$ and every positive integer s there exists an positive integer p^h such that

$$(f(M))^{p^h} g(x_1, \ldots, x_m) \in (F[[X(m)]])^n \setminus (x^s F[[X(m) \setminus \{x\}]])^n.$$

This means that also $f(M)$ is not ultimately periodic and therefore the rule it gives is sensitive to initial conditions. □

Lemma 8.6.3. *Let F be a finite field and $M \in \mathcal{M}_{n\times n}(F[X(m)])$. The cellular automaton $((F^n)^{\mathbb{Z}^m}, F_M)$ is sensitive to initial conditions if, and only if, any cellular automaton $((F^n)^{\mathbb{Z}^m}, F_{M'})$ given by Lemma 8.4.3 is not nilpotent.*

Proof. Assume that $((F^n)^{\mathbb{Z}^m}, F_M)$ is sensitive to initial conditions. Then also $((F^n)^{\mathbb{Z}^m}, F_{M'})$ must be sensitive by Lemma 8.6.2 and therefore it cannot be nilpotent.

Assume that $((F^n)^{\mathbb{Z}^m}, F_{M'})$ is not nilpotent. Let the characteristic of field F be p. Recall that $M' = \sum_{i=0}^{2m} M_i$ is a sum of commuting matrices with strictly one-sided neighborhoods. This means that after p^h time steps every cell in the initial configuration which affects a cell in location $\overrightarrow{x}$ is located at least a distance of $d(p^h)$ apart from the cell in location $\overrightarrow{x}$. Furthermore, $d(p^h) > d(p^{h-1})$ for every $h > 0$. This follows from the fact that p^hth powers of M' do not contain mixed terms of matrices M_i. That is, the effective cells will spread arbitrarily far from the cell in location $\overrightarrow{x}$ whenever the number of iterations is a power of the characteristic.

Because $((F^n)^{\mathbb{Z}^m}, F_{M'})$ was assumed not to be nilpotent, the effective cells will keep on spreading away from the cell in location $\overrightarrow{x}$ and their number will not drop to zero. Therefore, $((F^n)^{\mathbb{Z}^m}, F_{M'})$ must be sensitive so $((F^n)^{\mathbb{Z}^m}, F_M)$ must also be sensitive by Lemma 8.6.2. □

The following theorem gives a simple and practical method to test sensitivity of a given linear rule. The matrix given by Lemma 8.4.3 might impractical to calculate for Lemma 8.6.3, but Theorem 8.6.4 requires only powers $I, M, \ldots, M^{n-1}$ and their linear combinations to be computed.

Theorem 8.6.4. *Let F be a finite field and $M \in \mathcal{M}_{n\times n}(F[X(m)])$. The cellular automaton $((F^n)^{\mathbb{Z}^m}, F_M)$ is sensitive to initial conditions if, and only if,*

1. *every non-zero matrix $\sum_{i=1}^{n-1} a_i M^i$ (where $\{0\} \neq \{a_1, \ldots, a_{n-1}\} \subseteq F$) is not nilpotent and*
2. *the characteristic equation $\lambda^n = \sum_{i=1}^{n-1} a_i \lambda^i$ of matrix M is such that $F \not\supseteq \{a_1, \ldots, a_{n-1}\} \subseteq F[\lambda]$.*

Proof. Assume first, that both the conditions 1 and 2 hold. Let

$$\lambda^n = \sum_{i_1,\ldots,i_m \in \mathbb{Z}} f_{(i_1,\ldots,i_m)}(\lambda) x_1^{i_1} \cdots x_m^{i_m}$$

be the sum given by the characteristic equation of M rearranged to a Laurent polynomial with coefficients from $F[\lambda]$. If condition 1 holds, then every matrix $f_{(i_1,\ldots,i_m)}(M)$ is not nilpotent. Suppose that the matrix M was ultimately periodic (i.e. not sensitive). Then every matrix $f_{(i_1,\ldots,i_m)}(M)$ would also be ultimately periodic by Lemma 8.6.2. Then there exists a bound k for all matrices $f_{(i_1,\ldots,i_m)}(M)$ such that every power of a matrix $f_{(i_1,\ldots,i_m)}(M)$ contains no term with a power higher than x_j^k or lower than x_j^{-k} for any variable x_j. That is, the occurrences of powers of any variable x_j remain bounded. Let p be the characteristic of F and choose $h > 0$ so that $p^h > 2k$. Then for any $t > 0$ the non-zero terms in any two non-zero matrices $(f_{(i_1,\ldots,i_m)}(M) x_1^{i_1} \cdots x_m^{i_m})^{p^{ht}}$ and $(f_{(i'_1,\ldots,i'_m)}(M) x_1^{i'_1} \cdots x_m^{i'_m})^{p^{ht}}$ do not overlap. Then it follows that every matrix $M^{np^{ht}}$ in non-zero and the positive or the negative powers of at least one of the variables $x_1, \ldots, x_m$ will grow without a bound in matrices $M^{np^{ht}}$ when t grows. Hence, the matrix is not ultimately periodic and the cellular automaton is sensitive to initial conditions.

Assume second, that at least one of the conditions does not hold. If condition 1 does not hold, then the matrix is ultimately periodic by Lemma 8.6.2. If condition 2 does not hold, then the matrix is ultimately periodic because none of the coefficients of the characteristic polynomial are polynomials. Then the powers of any x_j or x_j^{-1} do not grow any larger than what has already been encountered in matrices $I, M, \ldots, M^{n-1}$. □

Theorem 8.6.5. *Let L be a finite commutative local ring with identity and let $M \in \mathcal{M}_{n\times n}(L[X(m)])$. Then matrix M defines a sensitive cellular automaton over state set L^n if, and only if, matrix*

$$N = M + \mathcal{M}_{n\times n}(N(L)[X]) \in \mathcal{M}_{n\times n}(L/N(L)[X(m)])$$

defines a sensitive cellular automaton over state set $(L/N(L))^n$.

Proof. First, assume that N defines a sensitive cellular automaton. This means that the powers of the variables with coefficients of the form $a + N(L) \in L/N(L)$ where $a \in U(L)$ will grow arbitrarily large. Then the powers of the variables with coefficients from $U(L)$ will grow arbitrarily large in the powers of M. Therefore also M must be sensitive.

Assume that M defines a sensitive cellular automaton. This means that the powers of the variables with coefficients from $U(L)$ must grow arbitrarily large. Then N would be sensitive. If this did not happen, say for some $k \in \mathbb{N}$ and every power M^l the coefficient a of a non-zero term $ax_1^{i_1} \cdots x_m^{i_m}$ found in M^l would be such that $a \in N(L)$ whenever $|i_j| > l$ for some j, with $1 \leq j \leq m$. Because of nilpotency of the elements of $N(L)$, the coefficient a of a non-zero term $ax_1^{i_1} \cdots x_m^{i_m}$ found in M^l would be such that $a = 0$ whenever $|i_j| > l + r\mathrm{nil}\,(N(L))$ for some j, with $1 \leq j \leq m$, which means that M is not sensitive after all. □

Theorem 8.6.6. *Let R be a finite commutative ring with identity and $M \in \mathcal{M}_{n\times n}(R[X(m)])$. Then it is decidable whether the m-dimensional linear cellular automaton $((R^n)^{\mathbb{Z}^m}, F_M)$ is sensitive to initial conditions or not.*

Proof. Because it is decidable whether a linear cellular automaton is nilpotent or not, the claim follows (either by Lemma 8.6.3 or) by Theorem 8.6.4 and Theorem 8.6.5 when the matrix defines a sensitive rule in at least one of the local rings given by Theorem 8.2.1. □

Appendix A

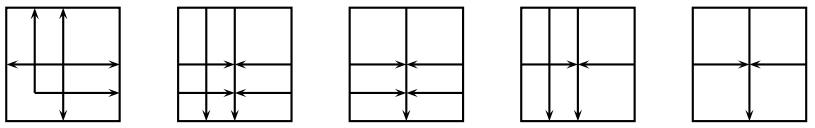
On Robinson's tile set

In this appendix the Robinson's tile set's structure and its tilings are briefly reviewed to provide the reader with the details used in Chapter 3.

Robinson's tile set is constructed in two layers. On the first layer there are the *basic tiles* and on the second layer there are the *parity tiles*.

A.1 The basic tiles

The *basic tiles* of Robinson's tile set construction are the ones shown in Figure A.1 with all their reflected and rotated variants. The tiles which are rotated or reflected variants of the tile in Figure A.1(a) are called *crosses*. The tiles which are rotated or reflected variants of the tiles in Figures A.1(b)–A.1(e) are called *arms*. Each tile has a *central arrow* at the center of each four sides and possibly a *side arrow*. A cross is said face the directions of its side arrows. The arrow that runs through an arm is called the *principal arrow* of the arm and the direction of the principal arrow is called the *direction* of the arm [50].



(a) A cross. (b) An arm. (c) An arm. (d) An arm. (e) An arm.

Figure A.1: The basic tiles of Robinson [92, p. 182] (with rotations and reflection omitted).

A.2 The parity tiles

A position $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ is said to be

1. *odd-odd*, if $(x + 1, y + 1) \in 2\mathbb{Z} \times 2\mathbb{Z}$,
2. *odd-even*, if $(x + 1, y) \in 2\mathbb{Z} \times 2\mathbb{Z}$,
3. *even-odd*, if $(x, y + 1) \in 2\mathbb{Z} \times 2\mathbb{Z}$, and
4. *even-even*, if $(x, y) \in 2\mathbb{Z} \times 2\mathbb{Z}$.

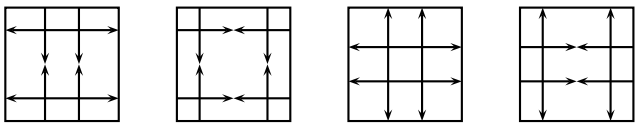


Figure A.2: The parity tiles of Robinson [92, p. 185].

The tiles in Figure A.2 are called *parity tiles*.

The basic tiles and parity tiles are paired (to form a sandwich tile set) as follows:

1. Crosses are paired with the parity tile in Figures A.2(b) and A.2(c).
2. Horizontal arms are paired with the parity tile in Figures A.2(a) and A.2(b).
3. Vertical arms are paired with the parity tile in Figures A.2(b) and A.2(d).

This causes the crosses to appear in alternate columns and in alternate rows, say in the odd-odd positions [92]. If the odd-odd positions are filled with crosses, then the odd-even positions are filled with horizontal arms and the even-odd positions are filled with vertical arms. Both crosses and arms with any orientation can appear in even-even positions.

A.3 Colors

The arrows in the basic tiles (see Figure A.1) of Robinson's tile set are colored as follows [50]:

1. The side arrows of each cross are both red or both blue. The crosses at odd-odd positions have blue side arrows. In this way the 3×3 squares are forced to be blue.

2. In each arm the horizontal side arrows have the same color and the vertical side arrows have the same color. In this way the color is transmitted unchanged through the arm. If an arm contains both horizontal and vertical side arrows then these side arrows have different colors.

3. In the neighboring tiles the matching rule is that the meeting arrow heads and tails must have the same color.

A.4 Square patterns in a valid tiling

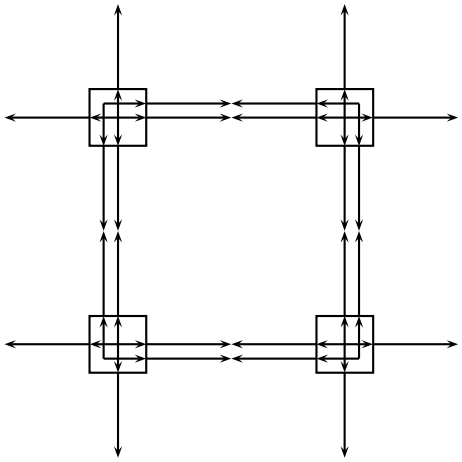


Figure A.3: The patterns formed by the side arrows of the cross tiles are interpreted as red and blue squares.

In a valid tiling the closed pattern consisting of side arrows originating from four crosses (as shown in Figure A.3) is a *square*. The squares are indeed squares and not just rectangles, i.e. their width and height are equal. Moreover, both the width and the height and of a square are equal to $2^n + 1$ for some $n \in \mathbb{N}$.

The parity constraints force the smallest squares to be blue. Because squares of the same color cannot intersect, it follows that blue squares are of height $2^{2n+1} + 1$, where $n \in \mathbb{N}$, and red squares are of height $4^{n+1} + 1$, where $n \in \mathbb{N}$.

There is always four squares of height $2^n + 1$ centered at the four corners of a square of height $2^{n+1} + 1$ as illustrated in Figure A.4. The four smaller squares are of different color than the larger square.

Now the tiles in the corners and in the middle of edges of squares can be identified as subsets of the tile set. For example, a tile in the top left

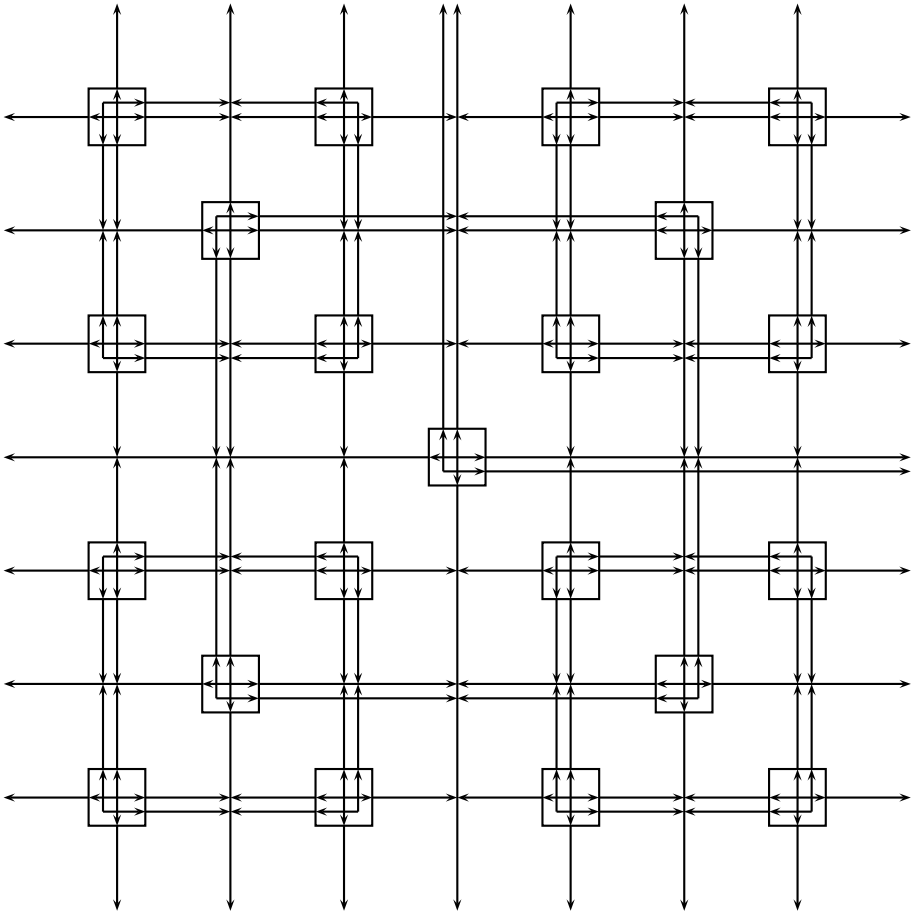


Figure A.4: A red (resp. blue) square of height $2^n + 1$ is always centered at a corner of a blue (resp. red) square of height $2^{n+1} + 1$.

corner of a square is a cross with double arrows travelling rightwards and downwards. A tile in the middle of the left edge of a square has two meeting horizontal double arrows with the horizontal arrows aligned on the left and in the middle of the square.

Figure A.5 shows an example of how a blue 9×9 square is centered at the bottom left corner a red 17×17 square. Likewise, there are red 5×5 squares centered at the corners the blue 9×9 square.

In a valid tiling there can exist a single column or a single row arms which does not intersect any colored square. There exists at most one such horizontal line and one such vertical line and they can divide the plane into

1. two half planes,

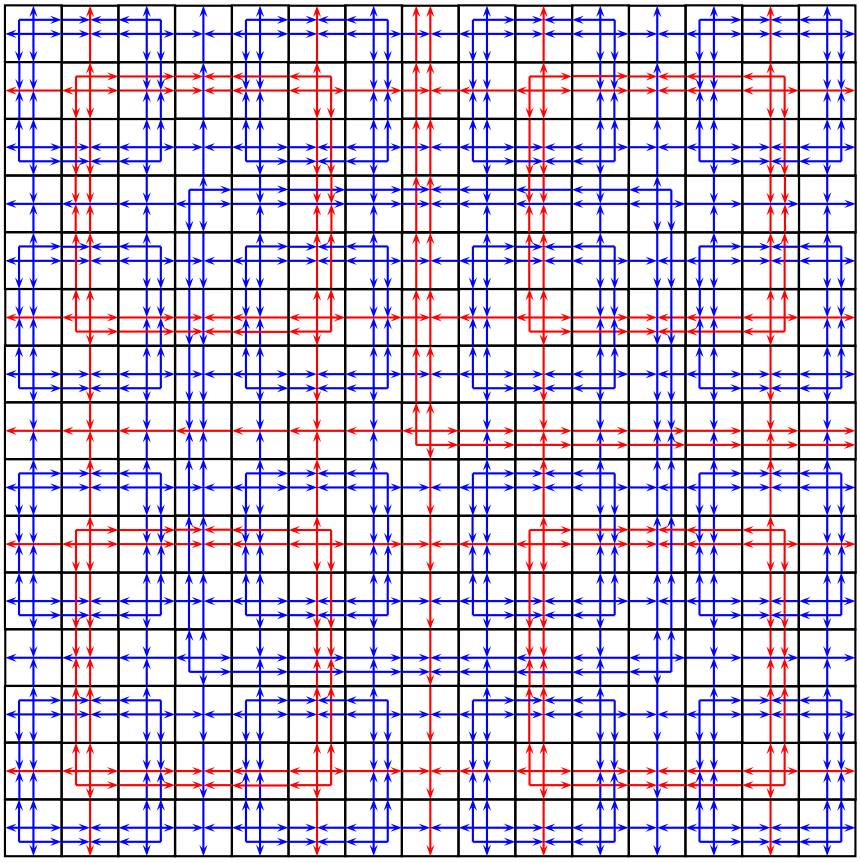


Figure A.5: A example of a tile pattern in a valid tiling. A blue 9×9 square is centered at the bottom left corner of a red 17×17 square.

2. a half plane and two quarter planes, or

3. four quarter planes.

Such lines are called *fracture lines* in [29, p. 199]. The lines could be though of as to originate from "infinitely large" squares. A cross in such a line would not be in the center of any colored square. Such a column or row is called a *fault line* if the colored squares on opposite sides are not aligned symmetrically [92, p. 189].

However, fracture lines and even fault lines do not affect the proof of tiling problem's undecidability in a negative way. Regardless of the fault lines, arbitrarily large red squares are always tiled in the different quadrants. If arbitrarily large areas can be tiled in the quadrants, then the whole plane can be tiled without fault lines.

Likewise, the fault lines do not enable an incorrect simulation of another tiling on the free areas within squares. That is, the presence of fault lines does not remove the existence or arbitrarily large combined free areas within red squares. More precisely, if a free row within a red square were tiled as if it were non-free (and hence not used for simulation purposes), it would cause a tiling error by Lemma 3.4.4 (as shown in Figure 3.24) regardless of the alignment of nearby squares of the same size.

Bibliography

[1] L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, D. Kempe, P. M. de Espanés, and P. W. K. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings 34th Annual ACM Symposium on Theory of Computing*, pages 23–32, 2002.

[2] G. Aggarwal, Q. Cheng, M. H. Goldwasser, M.-Y. Kao, P. M. de Espanés, and R. T. Schweller. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005.

[3] S. Amoroso and Y. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tesselation structures. *Journal of Computer System Sciences*, 6:448–464, 1972.

[4] H. Baltzer, P. W. Braun, and W. Köhler. Cellular automata models for vegetation dynamics. *Ecological modelling*, 107:113–125, 1998.

[5] F. Bao. Cryptanalysis of a new cellular automata cryptosystem. In *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 2003.

[6] C. H. Bennett. Logical reversibility of computation. *IBM Journal of research and development*, 6:525–532, 1973.

[7] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[8] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays II.* Academic Press, 1982.

[9] F. Blanchard and A. Maass. Dynamical properties of expansive one-sided cellular automata. *Israel Journal of Mathematics*, 99:149–174, 1997.

[10] F. Blanchard and P. Tisseur. Some properties of cellular automata with equicontinuity points. *Annales de l'Institute Henri Poincaré*, 36(5):562–582, 2000.

[11] M. Boyle and W. Krieger. Periodic points and automorphisms of the shift. *Trans. Amer. Math. Soc.*, 302:125–149, 1987.

[12] L. Le Bruyn and M. Van den Bergh. Algebraic properties of linear cellular automata. *Linear Algebra and its Applications*, 157:217–234, 1991.

[13] G. Cattaneo, A. Dennunzio, and F. Farina. A full cellular automaton to simulate predator-prey systems. In *ACRI 2006*, volume 4173 of *Lecture Notes in Computer Science*, pages 446–451. Springer-Verlag, 2006.

[14] G. Cattaneo, E. Formenti, G. Manzini, and L. Margara. Ergodicity, transitivity, and regularity for linear cellular automata over $\mathbb{Z}_m$. *Theoretical Computer Science*, 233:147–164, 2000.

[15] G. Cattaneo and L. Margara. Topological definitions of chaos applied to cellular automata dynamics. In *MFCS 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 816–824. Springer-Verlag, 1998.

[16] A. Clarridge and K. Salomaa. A cryptosystem based on the composition of reversible cellular automata. In *LATA 2009*, volume 5457 of *Lecture Notes in Computer Science*, pages 314–325. Springer-Verlag, 2009.

[17] B. Codenotti and L. Margara. Transitive cellular automata are sensitive. *Amer. Math. Monthly*, 103:58–62, 1996.

[18] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *Proceedings of SIGGRAPH 2003*, volume 22 of *ACM Transactions of Graphics*, pages 287–294, 2003.

[19] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.

[20] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[21] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[22] E. Czeizler. *The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory*. PhD thesis, Turku Centre for Computer Science, 2007.

[23] E. Czeizler and J. Kari. A tight linear bound on the neighborhood of inverse cellular automata. In *Proceedings of Automata, Languages and Programmin 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 410–420. Springer-Verlag, 2005.

[24] M. D'amico, G. Manzini, and L. Margara. On computing the entropy of cellular automata. *Theoretical Computer Science*, 290:1629–1646, 2003.

[25] M. Davis. *Computability and unsolvability*. McGraw-Hill, 1958.

[26] R. L. Devaney. *An introduction to chaotic dynamical systems*. Addison-Wesley, 1989.

[27] J.-C. Dubacq. How to simulate any Turing machine by reversible one-dimensional cellular automaton. *International Journal of Foundations of Computer Science*, 6(4):395–402, 1995.

[28] B. Durand, E. Formenti, and G. Varouchas. On undecidability of equicontinuity classification for cellular automata. *Discrete Mathematics and Theoretical Computer Science*, pages 117–128, 2003.

[29] B. Durand and V. Poupet. Asymptotic cellular complexity. In *DLT 2009*, volume 5583 of *Lecture Notes in Computer Science*, pages 195–206. Springer-Verlag, 2009.

[30] R. Fernando and M. J. Kilgard. *The Cg Tutorial*. Addison-Wesley, 2003.

[31] M. Finelli, G. Manzini, and L. Margara. Lyapunov exponents versus expansivity and sensitivity in cellular automata. *J. Complexity*, 14:210–233, 1998.

[32] E. Formenti, J. Kari, and S. Taati. The most general conservation law for a cellular automaton. In *Proceedings of CSR 2008*, volume 5010 of *Lecture Notes in Computer Science*, pages 194–203. Springer-Verlag, 2008.

[33] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Physical review letters*, 56(14):1505–1508, 1986.

[34] M. Gardner. Mathematical games. *Scientific American*, 223(4):120–123, 1970.

[35] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman and company, New York, 1979.

[36] P. Guan. Cellular automata public key cryptosystem. *Complex Systems*, 1:51–57, 1987.

[37] J. Hardy, O. de Pazzis, and Y. Pomeau. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. *Physical review A*, 13(5):1949–1961, 1976.

[38] G. Hedlund. Endomorphisms and automorphisms of shift dynamical systems. *Math. Systems Theory*, 3:320–375, 1969.

[39] P. Hooper. The undecidability of the Turing machine immortality problem. *The Journal of Symbolic Logic*, 31(2):219–234, 1966.

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

[41] T. W. Hungerford. *Algebra*. Holt, Rinehart and Winston, 1974.

[42] L. P. Hurd, J. Kari, and K. Culik. The topological entropy of cellular automata is uncomputable. *Ergodic Theory and Dynamical Systems*, 12:255–265, 1992.

[43] M. Ito, N. Osato, and M. Nasu. Linear cellular automata over $\mathbb{Z}_m$. *Journal of Computer System Sciences*, 27:125–140, 1983.

[44] I. Kaplansky. *Commutative Rings*. The University of Chicago Press, 1974.

[45] J. Kari. Cryptosystems based on reversible cellular automata. Manuscript, 1992.

[46] J. Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM Journal on Computing*, 21:571–586, 1992.

[47] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer System Sciences*, 48:149–182, 1994.

[48] J. Kari. Rice's theorem for the limit sets of cellular automata. *Theoretical Computer Science*, 127:229–254, 1994.

[49] J. Kari. Linear cellular automata with multiple state variables. In *STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 110–121. Springer-Verlag, 2000.

[50] J. Kari. Tilings and patterns. Lecture notes, University of Turku, 2004.

[51] J. Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, 334:3–33, 2005.

[52] J. Kari. Cellular automata. Lecture notes, University of Turku, 2009.

[53] J. Kari and V. Lukkarila. Some undecidable dynamical properties for one-dimensional reversible cellular automata. In *Algorithmic Bioprocesses*, pages 639–660. Springer, 2009.

[54] J. Kari and N. Ollinger. Periodicity and immortality in reversible computing. In *MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 419–430. Springer-Verlag, 2008.

[55] J. Kari and P. Papasoglu. Deterministic aperiodic tile sets. *Geometric and functional analysis*, 9:353–369, 1999.

[56] S. C. Kleene. A symmetric form of Gödel's theorem. *Indagationes Mathematicae*, 12:244–246, 1950.

[57] C. Knudsen. Chaos without nonperiodicity. *The American Mathematical Monthly*, 101:563–565, 1994.

[58] P. Kurka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory and Dynamical Systems*, 17:417–433, 1997.

[59] P. Kurka. Topological dynamics of cellular automata. In B. Markus and J. Rosenthal, editors, *Codes, Systems and Graphical models*, volume 123 of *IMA Volumes in Mathematics and its Applications*, pages 447–498. Springer-Verlag, 2001.

[60] A. Lagae, J. Kari, and P. Dutré. Aperiodic sets of square tiles with colored corners. Technical Report CW460, Katholieke Universiteit Leuven, 2006. `www.cs.kuleuven.ac.be/publicaties/rapporten/cw/cw460.html`.

[61] M. Lagoudakis and T. LaBean. 2D DNA self-assembly for satisfiability. In *Proceedings of the 5th DIMACS Workshop on DNA Based Computers held at MIT, Cambridge*, 1999.

[62] Y. Lecerf. Machines de Turing réversibles. *C. R. Acad. Sci. Paris*, 257:2597–2600, 1963.

[63] P. Di Lena. Decidable propertied for regular cellular automata. In *Fourth IFIP conference on Theoretical Computer Science - TCS 2006*, pages 185–196. Springer-Verlag, 2006.

[64] W. Li, Z. Fan, X. Wei, and A. Kaufman. Flow simulation with complex boundaries. In M. Pharr, editor, *GPU Gems 2*, chapter 47, pages 747–764. Addison-Wesley, 2004.

[65] D. Lind and B. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

[66] V. Lukkarila. On the tiling problem and reversible cellular automata. Technical Report 788, TUCS, 2006.

[67] V. Lukkarila. The square tiling problem is NP-complete for deterministic tile sets. Technical Report 754, TUCS, 2006.

[68] V. Lukkarila. The 4-way deterministic tiling problem is undecidable. In M. Hirvensalo, V. Halava, I. Potapov, and J. Kari, editors, *Proceedings of the Satellite Workshops of DLT 2007*, volume 45 of *TUCS general publications*, pages 100–105. TUCS, 2007.

[69] V. Lukkarila. On undecidability of sensitivity of reversible cellular automata. In A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martinez, K. Morita, and T. Worsch, editors, *AUTOMATA 2008*, pages 100–105. Luniver Press, 2008.

[70] V. Lukkarila. The 4-way deterministic tiling problem is undecidable. *Theoretical Computer Science*, 410:1516–1533, 2009.

[71] V. Lukkarila. Sensitivity and topological mixing are undecidable for reversible one-dimensional cellular automata. *Journal of Cellular Automata*, 5(3):241–272, 2010.

[72] M. Madjarova, M. Kakuta, M. Yamaguchi, and N. Ohyama. Two-dimensional cellular automata for pseudo-random pattern generators and for highly secure stream ciphers. *Optical Review*, 5(3):143–151, 1998.

[73] D. Makowiec. On modeling of the heart pacemaker by cellular automata — topology issue. In A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martinez, K. Morita, and T. Worsch, editors, *AUTOMATA 2008*, pages 586–600. Luniver Press, 2008.

[74] Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill, 1974.

[75] G. Manzini and L. Margara. Invertible linear cellular automata over Z_m: Algorithmic and dynamical aspects. *Journal of Computer and System Sciences*, 56:60–67, 1998.

[76] G. Manzini and L. Margara. A complete and efficiently computable topological classification of d-dimensional linear cellular automata over Z_m. *Theoretical Computer Science*, 221:157–177, 1999.

[77] C. R. F. Maunder. *Algebraic Topology*. Van Nostrand Reinhold Company, London, 1970.

[78] N. McCoy. *Rings and Ideals*. 1948.

[79] B. R. McDonald. *Finite rings with identity*. Marcel Dekker, Inc., 1974.

[80] E. F. Moore. Machine models of self-reproduction. In *Proceedings of the Symposium in Applied Mathematics*, volume 14, pages 17–33, 1962.

[81] K. Morita. Universality of a reversible two-counter machine. *Theoretical Computer Science*, 168:303–320, 1996.

[82] K. Morita and M. Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *Trans. IEICE Japan*, E72:758–762, 1989.

[83] K. Morita and Y. Yamaguchi. A universal reversible turing machine. In *MCU 2007*, volume 4664 of *Lecture Notes in Computer Science*, pages 90–98, 2007.

[84] J. Myhill. The converse to Moore's Garden-of-Eden theorem. In *Proceedings of the American Mathematical Society*, volume 14, pages 685–686, 1963.

[85] M. Nasu. *Textile systems for endomorphisms and automorphisms of the shift*, volume 546 of *Mem. Amer. Math. Soc.* AMS, 1995.

[86] M. Nasu. Textile systems and one-sided resolving automorphisms and endomorphisms of the shift. *Ergodic Theory and Dynamical Systems*, 28:167–209, 2008.

[87] D. G. Northcott. *Ideal Theory*, volume 42 of *Cambridge Tracts in Mathematics and Mathematical Physics*. Cambridge University Press, 1953.

[88] C. M. Papadimitrou. *Computational complexity*. Addison-Wesley, 1994.

[89] F. Pellacini and K. Vidimce. Cinematic lighting. In R. Fernando, editor, *GPU Gems 1*, chapter 10, pages 167–183. Addison-Wesley, 2004.

[90] G. J. Pettet, C. P. Please, R. L. Colasanti, R. A. Dawson, and J. Malda. A cellular automaton simulation of calcium driven tissue differentiation in human skin equivalent models. In A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martinez, K. Morita, and T. Worsch, editors, *AUTOMATA 2008*, pages 601–610. Luniver Press, 2008.

[91] M. O. Rabin. On recursively enumerable and arithmetic models of set theory. *The Journal of Symbolic Logic*, 23(4):408–416, 1958.

[92] R. M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12:177–209, 1971.

[93] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

[94] R. J. Rost. *OpenGL Shading Language*. Addison-Wesley, second edition, 2007.

[95] M. Sablik and G. Theyssier. Topological dynamics of 2D cellular automata. In A. Beckmann and C. Dimitracopoulos, editors, *CiE 2008*, volume 5028 of *Lecture Notes in Computer Science*, pages 523–532. Springer-Verlag, 2008.

[96] T. Sato. Ergodicity of linear cellular automata over $\mathbb{Z}_m$. *Information Processing Letters*, 61:169–172, 1997.

[97] S. Sen, C. Shaw, D. R. Chowdhuri, N. Ganguly, and P. P. Chaudhuri. Cellular automata based cryptosystem (CAC). In *ICICS 2002*, volume 2513 of *Lecture Notes in Computer Science*, pages 303–314. Springer-Verlag, 2002.

[98] M. A. Shereshevsky. Expansiveness, entropy and polynomial growth for groups acting on subshifts by automorphisms. *Indag. Math. N. S.*, 4:203–210, 1993.

[99] D. Shreier, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley, sixth edition, 2008.

[100] D. Soloveichik, M. Cook, and E. Winfree. Combining self-healing and proofreading in self-assembly. *Natural Computing*, 7:203–218, 2008.

[101] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5:19–31, 1991.

[102] R. Tao and S. Chen. On finite automaton public-key cryptosystem. *Theoretical Computer Science*, 226:143–172, 1999.

[103] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[104] H. Wang. Proving theorems by pattern recognition II. *Bell Systems Technical Journal*, 40:1–41, 1961.

[105] J. Watrous. On one-dimensional quantum cellular automata. In *Proceedings of 36th FOCS*, pages 528–537, 1995.

[106] L.-Y. Wei. Tile based texture mapping. In M. Pharr, editor, *GPU Gems 2*, chapter 12, pages 189–199. Addison-Wesley, 2004.

[107] L.-Y. Wei. Tile-based texture mapping on graphics hardware. In T. Akenine-Möller and M. McCool, editors, *Proceedings of Graphics Hardware 2004*, pages 55–63, 2004.

[108] E. Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, California Institute of Technology, 1998.

[109] E. Winfree. Self-healing tile sets. In J. Chen, N. Jonoska, and G. Rozenberg, editors, *Nanotechnology: science and computation*, pages 55–78. Springer-Verlag, 2006.

[110] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

Index

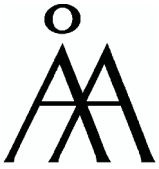
Turku Centre for Computer Science
TUCS Dissertations

95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming
128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Information Technologies

Turku School of Economics
- Institute of Information Systems Sciences

ISBN 978-952-12-2464-5
ISSN 1239-1883

Ville Lukkarila

On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata