Università degli Studi di Trieste

AUTOMATA 2023

The 29th International Workshop on Cellular Automata and Discrete Complex Systems

Exploratory Proceedings

Luca Manzoni
Luca Mariot

August 30 – September 1, 2023

Preface

This volume contains the exploratory papers and the extended abstracts accepted and presented at the 29th International Workshop on Cellular Automata and Discrete Complex Systems, AUTOMATA 2023, which was held in Trieste, Italy, on August 30 - September 1, 2023. The workshop was organized by the Department of Mathematics and Geosciences of the University of Trieste, and it hosted the annual meeting of the IFIP working group 1.5.

The current focus of AUTOMATA encompasses a wide range of aspects and features pertaining to cellular automata and discrete complex systems. While not exhaustive, the current topics include dynamics, topology, ergodicity, algebraic properties, algorithmic considerations, complexity analysis, emergence of properties, formal languages, symbolic dynamics, tilings, models of parallelism and distributed systems, timing schemes, synchronous and asynchronous models, phenomenological descriptions, scientific modeling, and practical applications.

Overall, AUTOMATA 2023 received 11 submissions in the exploratory track. Each paper was reviewed in single-blind mode by at least two program committee members. Following the review and discussion phases, the committee decided to accept 2 submissions as extended abstract and 8 submissions as exploratory papers, with 2 submissions later withdrawn by the authors. The remaining submissions are included in these proceedings volume and were presented at the workshop. We wish to thank all authors who submitted to the exploratory track of AUTOMATA 2023.

We are thankful to the program committee of AUTOMATA 2023, for its great help in reviewing and selecting the submitted papers. We further thank the members of the local organizing committee of AUTOMATA 2023, namely Giulia Bernardini, Giuliamaria Menara and Gloria Pietropolli. Finally, we are also grateful for the support by the Department of Mathematics and Geosciences and the University of Trieste.

August 2023

Luca Manzoni
Luca Mariot

Organization – AUTOMATA 2023

Steering Committee

Pedro Paulo Balbi	Universidade Presbiteriana Mackenzie, Brazil
Nazim Fatès	INRIA Nancy, France
Pierre Guillon	Université d'Aix-Marseille, France
Dipanwita Roy Chowdhury	IIT Kharagpur, India
Hector Zenil	University of Cambridge, UK

Program Committee

Jan Baetens	Ghent University, Belgium
Pedro Paulo Balbi	Universidade Presbiteriana Mackenzie, Brazil
Alonso Castillo-Ramirez	University of Guadalajara, Mexico
Sukanta Das	IIEST Shibpur, India
Alberto Dennunzio	University of Milano-Bicocca, Italy
Andreas Deutsch	TU Dresden, Germany
Nazim Fatès	INRIA Nancy, France
Enrico Formenti	Université Côte d'Azur, France
Maximilien Gadouleau	Durham University, United Kingdom
Anahí Gajardo	Universidad de Concepción, Chile
Pierre Guillon	Université d'Aix-Marseille, France
Tomasz Gwizdałła	University of Lodz, Poland
Rolf Hoffmann	TU Darmstadt, Germany
Jarkko Kari	University of Turku, Finland
Martin Kutrib	University of Giessen, Germany
Andreas Malcher	University of Giessen, Germany
Luca Manzoni	University of Trieste, Italy (co-chair)
Luca Mariot	University of Twente, the Netherlands (co-chair)
Kenichi Morita	Hiroshima University, Japan
Kévin Perrot	Université d'Aix-Marseille, France
Dipanwita Roy Chowdhury	IIT Kharagpur, India
Ville Salo	University of Turku, Finland
Biplab K. Sikdar	IIEST Shibpur, India
Georgios Ch. Sirakoulis	Democritus University of Thrace, Greece
Siamak Taati	American University of Beirut, Lebanon
Guillaume Theyssier	Université d'Aix-Marseille, France
Ilkka Törmä	University of Turku, Finland
Hiroshi Umeo	Osaka Electro-Communication University, Japan

Organizing Committee

Giulia Bernardini	University of Trieste, Italy
Luca Manzoni	University of Trieste, Italy (co-chair)
Luca Mariot	University of Twente, the Netherlands (co-chair)
Giuliamaria Menara	University of Trieste, Italy
Gloria Pietropolli	University of Trieste, Italy

List of Contributions

Extended Abstracts

Exploratory Papers

Searching for number-conserving non-uniform binary cellular automata

Barbara Wolnik[1] and Maciej Dziemiańczuk[2]

[1] Institute of Mathematics, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, 80-308 Gdańsk, Poland
[2] Institute of Informatics, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, 80-308 Gdańsk, Poland

Abstract. We consider one-dimensional cellular automata whose cells can use different local rules to update their states. We study such cellular automata on the infinite grid and on its subgrids in the context of number conservation.

Keywords: Cellular automata · non-uniform cellular automata · number conservation.

We investigate non-uniform CAs (ν-CAs), *i.e.*, one-dimensional CAs acting on the infinite grid (or on some of its subgrids), for which cells can use different local rules to update their states. The infinite grid is denoted by $\mathcal{G}^{\infty}_{-\infty}$, *i.e.*

$$\mathcal{G}^{\infty}_{-\infty} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}.$$

Additionally, we consider the following subgrids of $\mathcal{G}^{\infty}_{-\infty}$:

$\mathcal{G}^{n}_{-\infty} = \{\ldots, n-2, n-1, n\}$, for $n \in \mathbb{Z}$;
$\mathcal{G}^{\infty}_{m} = \{m, m+1, m+2, \ldots\}$, for $m \in \mathbb{Z}$;
$\mathcal{G}^{n}_{m} = \{m, m+1, \ldots, n-1, n\}$, for $m, n \in \mathbb{Z}$ satisfying $m \leq n$.

If the type of the considered grid is irrelevant, we simply use the symbol $\mathcal{G}$, as this should not lead to confusion.

A configuration is any function from the grid $\mathcal{G}$ to the binary state set $\{0, 1\}$ and the set of all configurations is denoted by X. The value at a cell $i \in \mathcal{G}$ in a configuration $\mathbf{x} \in X$ is denoted by x_i. Most often, a configuration $\mathbf{x} \in X$ is conveniently represented as a sequence $(x_i)_{i\in\mathcal{G}}$

For a configuration $\mathbf{x}$, we denote by $\mu(\mathbf{x})$ the sum of all states in $\mathbf{x}$, *i.e.*, $\mu(\mathbf{x}) = \sum_{i\in\mathcal{G}} x_i$. A configuration $\mathbf{x}$ is said to be *finite* if the number of cells i for which $x_i = 1$ is finite (*i.e.*, if $\mu(\mathbf{x}) < +\infty$).

Let $\mathcal{S}$ be a finite set of binary local rules. Each rule sequence $[f_i]_{i\in\mathcal{G}}$, where $f_i \in \mathcal{S}$ for any $i \in \mathcal{G}$, induces a ν-CA $H : X \to X$ defined by, for any $\mathbf{x} \in X$ and $i \in \mathcal{G}$:

$$H(\mathbf{x})_i = f_i(x_{i-1}, x_i, x_{i+1}), \tag{1}$$

with the convention that if $j \notin \mathcal{G}$ we use *null boundary conditions* and in the case of finite subgrids also *periodic boundary conditions*.

The main property of ν-CAs that is investigated in our research is *number conservation*. In the binary case, it states that a given ν-CA preserves the number of ones in any finite configuration throughout its evolution.

Definition 1. *A ν-CA H on $\mathcal{G}$ is called number-conserving if $\mu(H(\mathbf{x})) = \mu(\mathbf{x})$ for each finite configuration $\mathbf{x} \in X$.*

Let us note that the above definition is one of two ones being considered for the infinite grid, however, in the case of the ν-CAs considered by us, both of them are equivalent (see Proposition 1 in [1]).

At the beginning of the research, we focused on to the so-called **non-uniform elementary** CAs (ν-ECAs), in which each cell is allowed to have its own local updating rule belonging to Wolfram's set of 256 elementary local rules. In our previous paper [2], we managed to describe in detail all number-conserving ν-ECAs on finite grids both with periodic and null boundary conditions. The main result of [2] states that each number-conserving ν-ECA on a finite grid of length at least five cells (regardless of the boundary conditions: periodic or null) is either a classical (uniform) number-conserving ECA or it is a conglomerate of components, which can be of four different types only and each of them has a very poor dynamics. Moreover, each component of the conglomerate "lives its own life", as if there were impermeable barriers between adjacent components. The characterization obtained allows, *inter alia*, to enumerate all number-conserving ν-ECAs on finite grids and and those that are reversible. Surprisingly, the numbers obtained are closely related to the Fibonacci sequence.

Since the structure of the number-conserving ν-ECAs does not change with the increase of the grid (no matter how long the considered grid is, all number-conserving ν-ECAs on it are built with the same four mentioned types of components), one might erroneously conclude that the same would be true of the infinite grid. Of course, all examples of number-conserving ν-ECAs found for finite grids can be, in some way, "converted" to the infinite grid. It turns out, however, that the infinite grid admits many other types of number-conserving ν-ECAs (see [3]). This means that when considering number conversation for non-uniform cellular automata, the infinite grid cannot be treated as a limiting case of finite grids, *i.e.*, there are number-conserving non-uniform cellular automata on the infinite grid that have no analogous counterpart on finite grids. Thus, unfortunately, computer experiments are useless to assist in finding these other types.

Now we try to describe what all ν-CAs look like if we increase the radius to 3/2. Right from the start, we are faced with an incredibly great computational complexity, even if we only consider finite grids. However, we were able to obtain some promising results that give hope for solving this problem in the future.

References

[1] A. Dennunzio, E. Formenti, and J. Provillard, "Local rule distributions, language complexity and non-uniform cellular automata," *Theoretical Computer Science*, vol. 504, pp. 38–51, Sep. 2013. DOI: 10.1016/j.tcs.2012.05.013.

[2] B. Wolnik, M. Dziemiańczuk, and B. De Baets, "Non-uniform number-conserving elementary cellular automata," *Information Sciences*, vol. 626, pp. 851–866, 2023. DOI: 10.1016/j.ins.2023.01.033.

[3] B. Wolnik, M. Dziemiańczuk, and B. De Baets, "Non-uniform number-conserving elementary cellular automata on the infinite grid: A tale of the unexpected," submitted to Information Sciences, 2023.

Exploring Solutions to the Odd-sized Grid Classification Problem in Cellular Automata.

Anna Nenca[1][0000−0003−2746−1061] and Barbara Wolnik[2,3][0000−0003−2935−5529]

[1] Institute of Informatics, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, 80-308 Gdańsk, Poland
[2] Institute of Mathematics, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, 80-308 Gdańsk, Poland
[3] KERMIT, Department of Data Analysis and Mathematical Modelling, Faculty of Bioscience Engineering, Ghent University, Coupure links 653, B-9000 Gent, Belgium

Abstract. We consider one-dimensional grids $\mathcal{G}_n = \{0, 1, \ldots, n-1\}$, (where n is a positive odd integer) with periodic boundary condition and with the state set $Q = \{0, 1\}$. We are interested in finding a local rule with the smallest possible radius that can correctly classify each initial configuration according to its parity.

Keywords: Cellular automata · parity problem · finite grid.

The parity problem is a challenge in cellular automata (CA) where the goal is to classify initial configurations into two classes based on their parity. If the initial configuration contains an odd number of 1s, the CA should converge to a fixed point of all 1s. Conversely, if the initial configuration contains an even number of 1s, the CA should converge to a fixed point of all 0s. This problem is considered more difficult than the density classification problem (DCP) because a simple flip in any input bit can alter the output.

However, the solution to the parity problem exists. In a paper by Betel et al. [1], they described a local rule called the BFO rule with a radius of 4 (neighborhood size of 9) that solves the parity problem. The construction of the BFO rule involved analyzing the properties of the De Bruijn graph of the rule, which preserves parity and converges correctly to a homogeneous configuration for any initial configuration.

The paper also demonstrated that no binary CA with a radius of at most two (neighborhood size not greater than 5) can solve the parity problem for all odd-sized grids. The question of whether a rule with a radius of 3 (neighborhood size of 7) exists to solve the parity problem remains open. The authors attempted to find such a rule using a sophisticated evolutionary algorithm.

We introduce new approach to the parity problem. A perfect solution needs to keep the parity of any configuration, thus it has to be number-conserving considered as a function in the field $\mathbb{Z}_2$. Most tools used in the field of number conservation will be useful also in the area of the parity problem.

References

1. H. Betel, P. P. Oliveira, and P. Flocchini. Solving the parity problem in one-dimensional cellular automata. *Natural Computing: An International Journal*, 12(3):323–337, 2013.

Pathfinding Neural Cellular Automata with Local Self-Attention

Felix Reimers[1,2], Sanyam Jain[1], Aarati Shrestha[1], and Stefano Nichele[1,3]

[1] Østfold University College, Halden, Norway
[2] Alfred-Wegener-Institut, Bremerhaven, Germany
[3] Oslo Metropolitan University, Oslo, Norway
{stefano.nichele}@hiof.no

Abstract. Current artificial intelligence systems are rather rigid and narrow, if compared to the adaptivity and the open-endedness of living organisms. Neural Cellular Automata (NCA) are an extension of traditional CA, where the transition rule is replaced by a neural network operating on local neighborhoods. NCA provide a platform for investigating more biologically plausible features of emergent intelligence. However, an open question is how can collections of cells in an NCA be trained to collectively explore an environment in search for energy sources and find suitable paths to collect them. In this work, we utilize an NCA equipped with a local self-attention mechanism trained with gradient descent for pathfinding. Our results show that NCA can be trained to achieve such task and collect energy sources, while being able to redistribute the available energy to neighboring alive cells. Ongoing work is exploring how those abilities may be incorporated in NCA to solve tasks with increased adaptivity and general intelligence.

Keywords: Neural Cellular Automata · Local Attention · Neural Network · Artificial Intelligence · Artificial Life.

1 Introduction

Cellular Automata (CA) have often been used to study how simple individual agents may self-organise with local information and give rise to an emergent collective behavior, in order to perform a global task. Recently, Neural Cellular Automata (NCA) have been proposed as an extension to traditional CA, where the transition rule is replaced by a neural network [1,2]. Such NCA models may be used as bottom-up machine learning tools [4] or, most notably, as substrates to explore how features of intelligence may emerge in artificial organisms [5,6]. One of the most essential tasks that even simple biological organisms, such as slime mold [7] or caenorhabditis elegans [8], can perform in order to survive is the exploration of their environment to reach energy sources.

In this work, we investigate if an agent, i.e., a collection of cells in a neural cellular automaton controlled by a uniform neural network using local information, can be trained to seek out energy sources and thus navigate around the grid to

survive for a prolonged period of time. In order to succeed, an agent would need to be able to move and redistribute the available energy across neighboring cells, as well as the ability of pathfinding to detect new energy sources. As typically done in NCA models, we use multiple CA layers (channels) besides the standard visible CA channel (which represents the alive cells and their respective states). We include an energy channel, representing the amount of energy present in a specific location (either in the environment or in a living cell) which is consumed by cells to stay alive, and a chemical channel representing the concentration of chemicals in a site. The chemical channel works similarly to pheromones, which increase in the presence of a living organism or dissipate over time otherwise. We test our trained NCA to reach different concentrations of energy in the environment, and investigate their emergent behavior and survival.

The long term aim of this exploratory work is to investigate how artificial intelligence (AI) systems may become more open-ended [9], adaptive [10,11], and general, by exploiting some of the features of cellular automata and dynamical complex systems [12].

2 Background and Related Work

Classical CA are discrete computational models consisting of a grid of cells that evolve over time based on a set of predefined rules. CAs are widely used to study complex systems, where local interactions gives rise to an emergent behaviour at the global level. Extensions to classical CA have recently been introduced, such as Lenia (Continuous CA) [6] and Neural CA [1,2], that have proven to be a valuable tool for studying biologically-relevant dynamical processes, as pattern formation and morphogenesis.

In the work herein, we use the Neural CA proposed in [2] (with the additions outlined in the Section 3), where the authors introduce a differentiable framework for CA, i.e., trainable with gradient descent. Neural networks are used to control the state update of a 2D CA consisting of a visible layer and several hidden layers. Such Neural CA allows the growth and development of complex structures, demonstrating robustness to perturbations and regeneration. The use of linear convolution operations, combined with local cell updates, produces a highly complex multi-level update rule. Another Neural CA approach is presented in [1], where neuroevolution is used to train the neural update rules.

The work in [3] applies Neural CA (NCA) for bottom-up classification, exploring whether CA can achieve global agreement on the composition of handwritten digits by utilizing local message passing. This research addresses the problem of how cell collectives determine their anatomical structure and classify the large-scale morphology that they are part of. Another application domain includes the development of a control system for a cart-pole agent [16] via self-organisation. The work in [17] uses Heterogeneous CA update rules as testbed for an Ising model and spiking neural networks to model biological neural networks. In [4] a Neural CA using a Vision Transformer including self-attention is used to perform reconstruction of images. Very recent work [15] uses Isotropic Neu-

ral CA [2], allowing symmetry breaking. Such work focuses on enabling an inner sense of orientation in cellular systems, which parallels how real-world organisms develop organs that have a sense of directionality. In [13] a NCA framework that incorporates goal embeddings through iterative sampling is introduced, allowing for effective guidance of cells towards desired behaviors. Their approach demonstrates the ability to achieve diverse behaviors in a morphing image experiment, although it lacks generative capabilities to learn latent distributions. Finally, the work in [14] preliminarily explores how NCA agents may be able to reach energy sources in a similar fashion to slime molds.

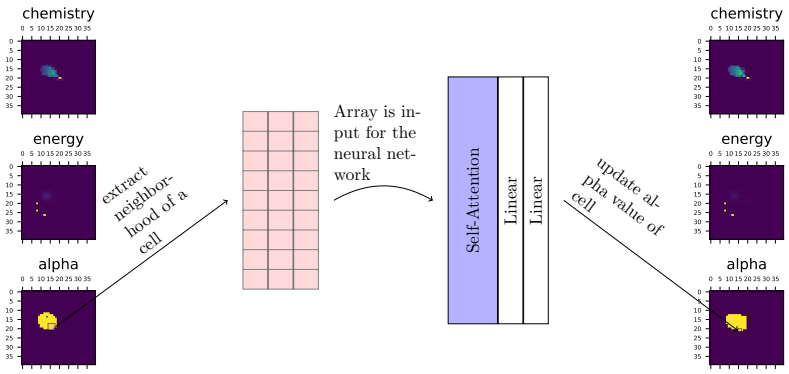


Fig. 1: An update step of the cellular automaton by the neural network. On the left is the CA grid. From this the neighborhood of a single cell is extracted ($9 \times 3 = 27$). The resulting array propagates through the network and an update is computed, which is added to the alpha channel of the grid seen on the right. Before and after, the deterministic updates are performed on the chemistry and energy channels.

3 Proposed Method

In this study, we use a two-dimensional cellular automaton with continuous values (allowing for a differentiable update function). It has a discrete 40 by 40 grid with each cell having 3 channels. The channels are a chemistry channel, an energy channel, and the alpha channel (representing the alive cells). The cell's state is either updated by the output of an uniform neural network (for the alpha

channel) or by fixed update rules (for the other channels). Our implementation supports additional hidden CA channels (currently not used in this work).[4]

The neural network consists of three layers, similar to the architecture in [2]. It operates locally: the channels of a single cell, and its eight neighboring cells, are the input to the neural network ($9 \times 3 = 27$ in total). The network computes an update for the alpha channel, which is then applied to the CA. Only the cells which are "alive", e.g. with an alpha channel over 0.1, and its neighbors are updated. The alpha channels are clipped to values between 0 and 1. The energy channel is initialized at the start of the training or testing. The values of the energy are updated deterministically, depending on the alpha channel: Energy, which is present in a cell with an alive alpha channel, is distributed within the neighborhood proportionally to their relative alpha channels, e.g. a cell with a high alpha value receives more energy from neighboring cells than a cell with a low alpha value. New energy deposits may be introduced during training. Otherwise, the cells will inevitably die after some iterations, as every update in an iteration costs a certain amount of energy. This introduces a notion of scarcity, similar to how it can be found in natural environments. Similarly, the chemistry channel is updated: On every alive cell, the chemistry value is increased, while on cells that are not alive the value is decreased until the initial value of 0 is reached. Those deterministic updates happen after the update of the alpha channel of the neural network.

The neural network is updating cells one at a time, but with the same topology and the same synaptic weights. This means, that there is only one agent operating on the grid, but for every update it can only use the locally available information. This reduces computational complexity, but also simulates either a single organism or a group of organisms which are similar in their genetic code but can still act independently based on their surrounding.

An example of the CA neural network and update step is shown in Figure 1.

3.1 NCA Architecture with Local Attention

The first layer is a self-attention layer. The motivation for including a self-attention layer is to enable learning which parts of the neighborhood are more important and should have more focus. As input it takes nine cells, the currently observed cell as well as its eight neighbors. All channels are used, which means, that the input to the neural network is an array of size 3 by 9. The layer has 633 trainable parameters. An important aspect of this is that the attention is applied only locally; cells which are not in the immediate neighborhood are disregarded. The self-attention mechanism is the canonical self-attention [18], widely used in Large Language Models.

Next comes two linear layers, the last layers output is used as an update to the alpha channel of the current cell. To keep those linear layers local, 1-D convolutional layers are used in the implementation (as done in the original

[4] The repository with the code is openly available at the following link: `https://github.com/Deskt0r/LocalAttentionNCA`

paper [2]). The first linear layer has 544 and the last linear layer has 33 trainable parameters. All layers are fully connected.

This procedure loops through all alive cells and all cells neighboring an alive cell (those can be "resurrected"). Each of those cells receives an update based on its neighbors. The implementation of the attention mechanism is canonical, the locality is achieved by padding the cellular grid and extracting patches of the grid according to the aliveness of the cells.

3.2 Training Routine

The neural network is trained by gradient-based optimization with error backpropagation, i.e., differentiable programming, as typically done in Deep Learning. To train the neural network, a pool based approach is used. Every sample in the pool is a grid which is initialized by setting the channels of a central cell such that it is alive and has a relatively large amount of energy. Furthermore, a second cell, randomly chosen in the vicinity of the center towards on of the four edges, is chosen for every batch as a target and also initialized with some energy. In total there are 8 batches with 4 samples.

For the loss, a mask is calculated which gives every cell a value corresponding to its distance to the target. It is important here to scale those values correctly, in order to give the cells an incentive to not just spread in all directions but in the desired one. The loss is the sum of the products of the alpha values with the values of the distance mask. In this work, there is no penalty term for the consumption of energy in the loss.

During the training phase, the network and the fixed rules update the cellular grid for a random number of iterations between 8 and 16 during each training step. Afterwards, the loss is calculated and backpropagated. This allows the calculation of a gradient. After a number of training steps, the target is changed. First only the location, but in a similar distance. Later on, the target is also put in a greater distance from the center. Furthermore, to avoid catastrophic forgetting, the sample of the batch with the highest loss may be reinitialized. The training runs for a total of 100 steps. This means that the network has to adapt to a changing environment over time, although that change is only partly random.

It could be noticed that depending on the random initialization of the weights of the network, the cells are prone to "die" before being able to be trained, especially when the network runs for a high amount of iterations before the weight's update. This is mitigated by re-initializing the grid in such cases. An example of the training loss for the CA model is shown in Figure 2.

As can be seen in the plot, the loss decreases over time, but is rather volatile. The two big spikes at iteration 15 and 20 can likely be explained by the introduction of targets further away, which poses a new challenge to the neural network that is has to overcome, i.e., a kind of curriculum learning.

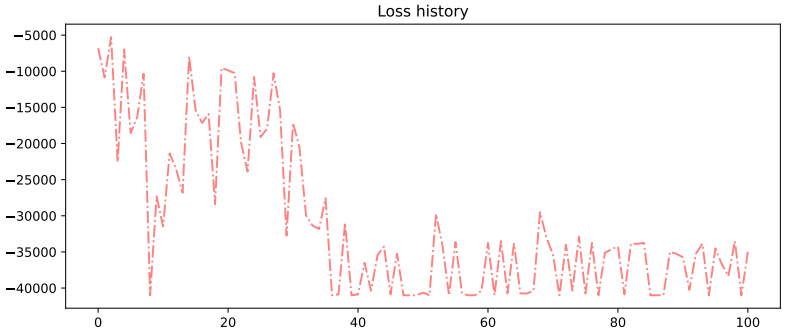


Fig. 2: The loss during the training of the model. Time on the x axis and loss value on the y axis.

4 Experiments

4.1 Experimental Setup

To test the performance of the network, first we executed cases where the target energy pocket is close to the center (Fig. 3 top), as it is in the initial steps of the training. Next, a target further away is picked (Fig. 3 bottom). To see how well the learned behavior generalizes, we also used two energy distribution patterns greatly differing from the ones during the training: In one case, the energy is distributed in a spiral (Fig. 5), in the other one there are two curved lines of energy that can be followed (Fig. 4). While we repeated the experiments several times confirming that the target behaviors are learned, the results have to be seen as qualitative rather then quantitative. Videos of the results can be seen at the following link: `https://github.com/Deskt0r/LocalAttentionNCA`.

5 Results and Discussion

Our tests indicate that for the "survival" of the cells, i.e. the amount of iterations in which the initialized or new cells have an alpha value above 0.1, the distance between the cells and energy deposits and the size of the initial energy available are of high importance, while the shape in which the energy is distributed seems less important.

In Fig. 3 it can be seen that when the next energy depot from the center is too far away, the cells are unavailable to reach it, even if there is enough energy initially available. This shows, that the network has not really learned to search for energy. When it is able to reach the deposit, the cells survive for the whole 100 iterations. In the latter case it dies out after 13 iterations. In Fig. 5 and Fig. 4, the cells are following the laid out pattern, even though they did not encounter such during training. It can not be seen on the single frames in the figures, but

the cells show an oscillating behaviour in those experiments. Furthermore, when the cells reach the last energy deposit, the energy seems to spread through the alive cells. During the expansion on the other hand, the energy seems to spread just thickly enough to keep the cells alive.

On a more general level, being able to learn properties typically available in biological organisms such as environment exploration, pathfinding, energy usage, etc., in cellular systems such as Neural CA, may allow the development of more open-ended and general AI systems. Different tasks may be encoded in such systems in the form of environmental signals and energy sources, and the solution to the task would result from the self-organising process of the CA. This exploratory work fits well with the emerging area of self-organising AI [12].

6 Conclusion

Our results show that it is possible to train a neural network, such that the cells under its control can seek energy sources. Similar to a normal cellular automaton, cells receive an update based on their neighborhood, but the neural network with self-attention allows for significantly more complex behaviour.

Further work could include a more sophisticated training routine exposing the network to a larger variety of situations. This could include energy deposits randomly spawning on the grid. Furthermore, a channel for hidden information could be used, which would allow cells to communicate with each other or store information for later use. It would be interesting to see how complex behaviour of the cells can emerge with regards to searching for energy, splitting and joining up, as well as long term energy preservation.

Acknowledgments

We would like to thank the Helmholtz Information and Data Science Academy (HIDA) and the Norwegian Artificial Intelligence Research Consortium (NORA) for providing a stipend for Felix Reimers through a mobility collaboration agreement between HIDA and NORA. We would like to thank Østfold Univerisity College for additional financial support.

References

1. Nichele, S., Ose, M. B., Risi, S., and Tufte, G. (2017). *CA-NEAT: evolved compositional pattern producing networks for cellular automata morphogenesis and replication.* IEEE Transactions on Cognitive and Developmental Systems, 10(3), 687-700.
2. Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). *Growing neural cellular automata.* Distill, 5(2), e23.
3. Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M., & Greydanus, S. (2020). *Self-classifying mnist digits.* Distill, 5(8), e00027-002.
4. Tesfaldet, M., Nowrouzezahrai, D., and Pal, C. (2022). *Attention-based Neural Cellular Automata.* Advances in Neural Information Processing Systems, 35, 8174-8186.
5. Gregor, K., and Besse, F. (2021). *Self-organizing intelligent matter: A blueprint for an AI generating algorithm.* arXiv preprint arXiv:2101.07627.
6. Hamon, G., Etcheverry, M., Chan, B. W. C., Moulin-Frier, C., and Oudeyer, P. Y. (2022). *Learning sensorimotor agency in cellular automata.*
7. Beekman, M., and Latty, T. (2015). *Brainless but multi-headed: decision making by the acellular slime mould Physarum polycephalum.* Journal of molecular biology, 427(23), 3734-3743.
8. Qin, J., and Wheeler, A. R. (2007). *Maze exploration and learning in C. elegans.* Lab on a Chip, 7(2), 186-192.
9. Stanley, K. O. (2019). *Why open-endedness matters.* Artificial life, 25(3), 232-235.
10. Pontes-Filho, S., Walker, K., Najarro, E., Nichele, S., and Risi, S. (2022). *A single neural cellular automaton for body-brain co-evolution.* In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 148-151).
11. Nadizar, G., Medvet, E., Nichele, S., and Pontes-Filho, S. (2022). *Collective control of modular soft robots via embodied Spiking Neural Cellular Automata.* arXiv preprint arXiv:2204.02099.
12. Risi, S. (2021). *The future of artificial intelligence is self-organizing and self-assembling.* Available at: sebastianrisi.com.
13. Sudhakaran, S., Najarro, E., and Risi, S. (2022). *Goal-Guided Neural Cellular Automata: Learning to Control Self-Organising Systems.* arXiv preprint arXiv:2205.06806.
14. Barbieux, A., and Canaan, R. (2023). *EINCASM: Emergent Intelligence in Neural Cellular Automaton Slime Molds.* arXiv preprint arXiv:2305.13425.
15. Randazzo, E., Mordvintsev, A., and Fouts, C. (2023). *Growing Steerable Neural Cellular Automata.* arXiv preprint arXiv:2302.10197.
16. Variengien, A., Pontes-Filho, S., Glover, T. E., and Nichele, S. (2021). *Towards Self-organized Control: Using Neural Cellular Automata to Robustly Control a Cart-pole Agent.* Innovations in Machine Intelligence (IMI), vol. 1, pp. 1-14.
17. Khajehabdollahi, S., Giannakakis, E., Buendia, V., Martius, G., and Levina, A. (2023). *Locally adaptive cellular automata for goal-oriented self-organization.* arXiv preprint arXiv:2306.07067.
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I. (2017). *Attention is all you need.* Advances in neural information processing systems, 30.

A Appendix

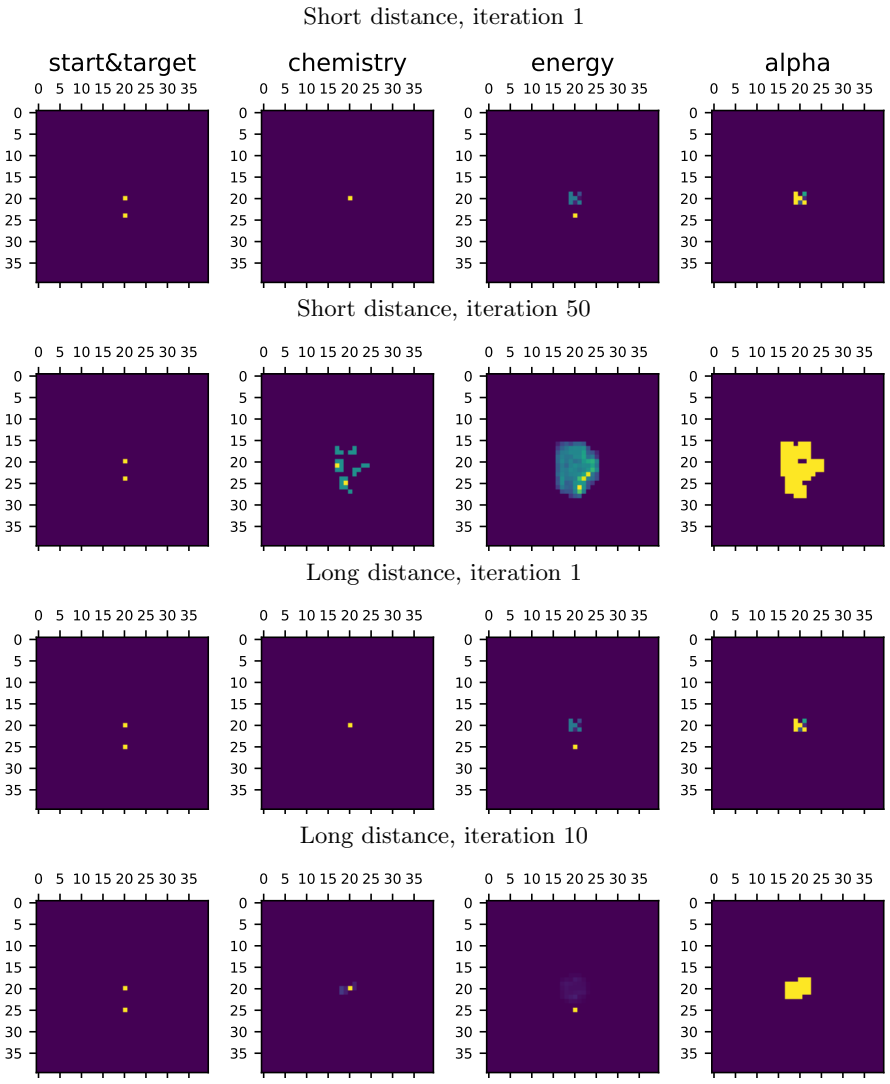


Fig. 3: *Short and long distance.* From left to right: seed of the grid with energy at the starting point and additional locations, chemistry channel, energy channel, alpha channel. All energy depots are initialized with a value of 5. The single additional energy deposit is located below the center. Short distance: 4 cells away; large distance: 5 cells away.

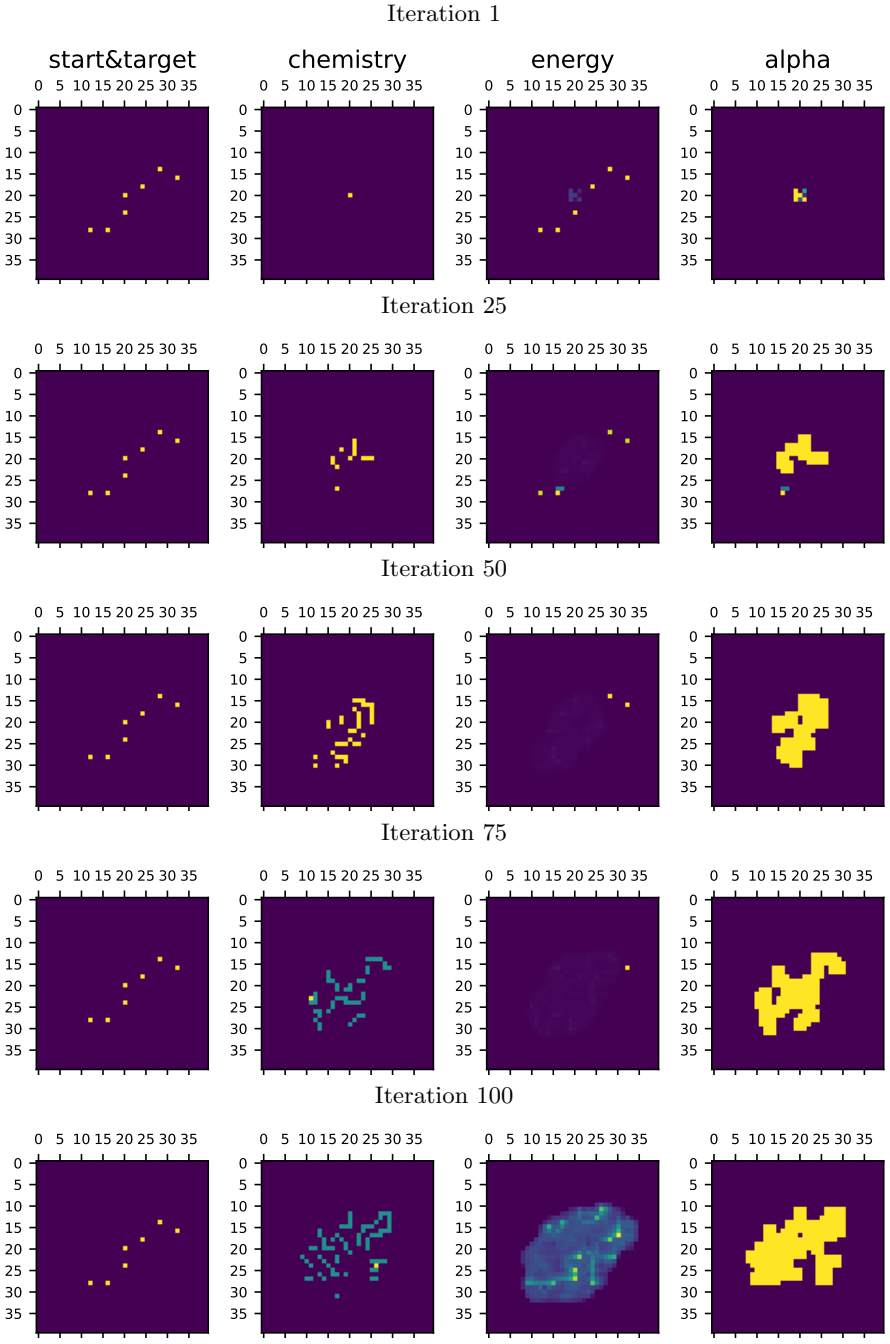


Fig. 4: *Split.* From left to right: seed of the grid with the energy depot at the starting point and additional locations, chemistry channel, energy channel, alpha channel. The energy is located in two paths from the center. All energy depots are initialized with a value of 5.

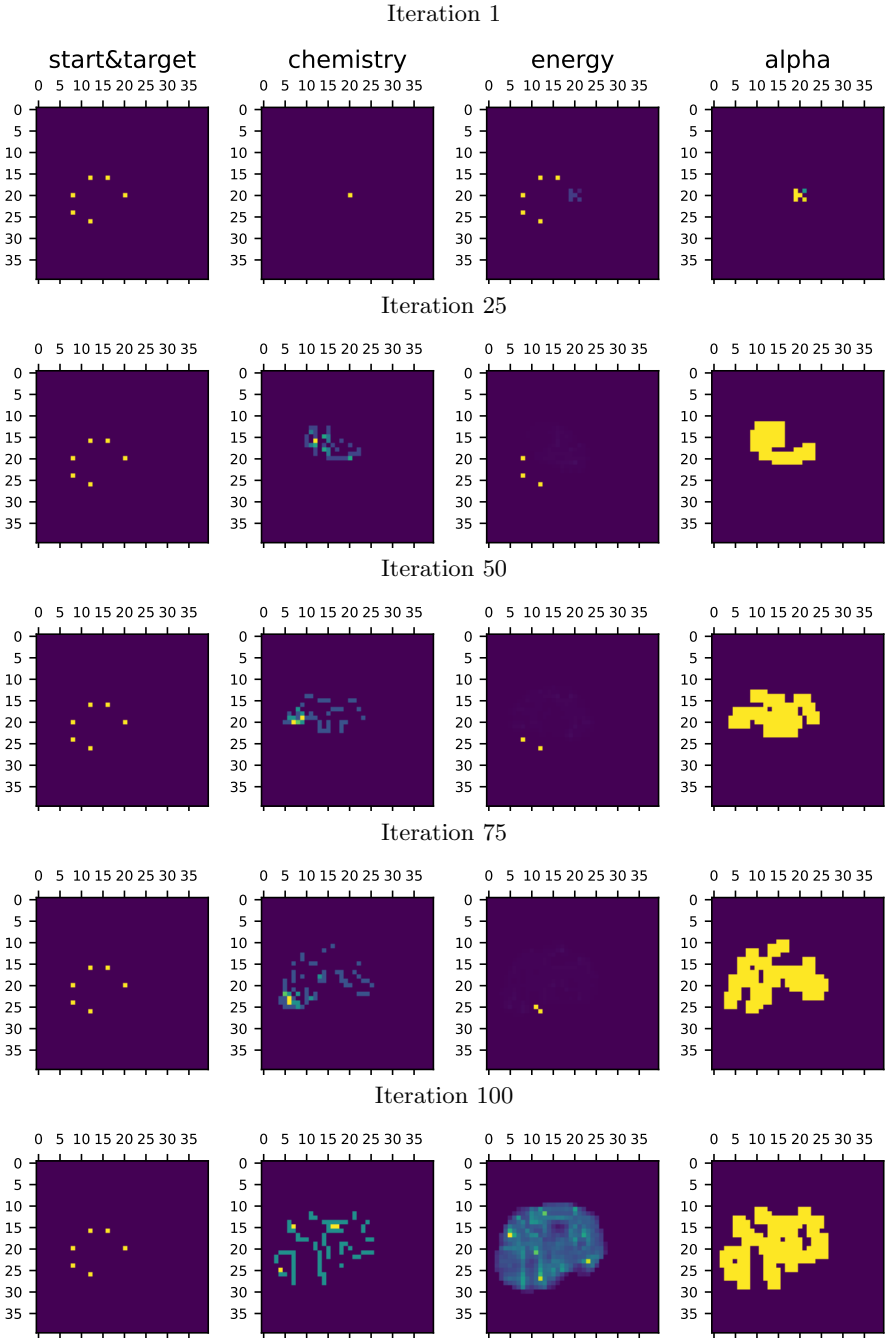


Fig. 5: *Spiral.* From left to right: seed of the grid with the energy depot at the starting point and additional locations, chemistry channel, energy channel, alpha channel. The energy is located in spiral starting from the center. All energy depots are initialized with a value of 8.

Dill maps in the Weyl-like space associated to the Levenshtein distance.

Firas BEN RAMDHANE[1] and Pierre GUILLON[2]

[1,2]Aix Marseille Univ, CNRS, Centrale Marseille, I2M, Marseille, France,
[1]Sfax University, Faculty of Sciences of Sfax, Tunisia.
[1]firasbenramdhane@math.cnrs.fr.
[2] pguillon@math.cnrs.fr.

Abstract. The Weyl pseudo-metric is a shift-invariant pseudo-metric over the set of infinite sequences, that enjoys interesting properties and is suitable for studying the dynamics of cellular automata. It corresponds to the asymptotic behavior of the Hamming distance on longer and longer subwords. In this paper we characterize well-defined dill maps (which are a generalization of cellular automata and substitutions) in the Weyl space and the sliding Feldman-Katok space where the Hamming distance appearing in the Weyl pseudo-metrics is replaced by the Levenshtein distance.

Keywords: The Weyl pseudo-metric · Feldman-Katok pseudo-metric · Cellular automata · Dill maps · Substitutions · Edit distances · Levenshtein distance · Symbolic dynamical systems · non-compact dynamical systemms.

1 Basic definitions and notations

Word combinatorics. We fix once and for all an *alphabet* A of finitely many *letters*. A *finite word* over A is a finite sequence of letters in A; it is convenient to write a word as $u = u_{[\![0,|u|[\![}$ to express u as the concatenation of the letters $u_0, u_1, \ldots, u_{|u|-1}$, with $|u|$ representing the length of u, *i.e.,* the number of letters appearing in u, and $[\![0, |u|[\![= \{0, ..., |u| - 1\}$. The unique word of length 0 is the empty word denoted by λ. A *configuration* $x = x_0x_1x_2\ldots$ over A is the concatenation of infinitely many letters from A. The set of all finite (resp. infinite) words over A is denoted by A^* (resp. $A^{\mathbb{N}}$), A^n is the set of words of length $n \in \mathbb{N}$ and $A^+ = A^* \setminus \{\lambda\}$.

Topologies over $A^{\mathbb{N}}$***.*** Most classically, the set $A^{\mathbb{N}}$ is endowed with the product topology of the discrete topology on each copy of A. The topology defined on $A^{\mathbb{N}}$ is metrizable, corresponding to the *Cantor distance* denoted by $\mathfrak{d}_C$ and defined as follows:

$$\mathfrak{d}_C(x, y) = 2^{-\min\{\, n\in\mathbb{N} | x_n \neq y_n\}}, \forall x \neq y \in A^{\mathbb{N}}, \text{ and } \mathfrak{d}_C(x, x) = 0, \forall x \in A^{\mathbb{N}}.$$

Topological dynamical systems were studied using other topologies on the set of infinite words, such as the Besicovitch and Weyl spaces [BFK97] and the Feldman-Katok space [RG22]. The Weyl space and a similar space that are defined using pseudo-metrics depend on the two following distances are of interest to us in this paper.

Definition 1. *1. The* Hamming distance *denoted by d_H and defined over finite words of the same length u, v by: $d_H(u,v) = |\{ i \in [\![0, |u|[\![\mid u_i \neq v_i \}|$.*

2. The Levenshtein distance *d_L is defined over $u, v \in A^*$ as follows:*

$$d_L(u,v) = \frac{1}{2} \min \left\{ m + m' \middle| \exists j_1 < \cdots < j_m, j'_1 < \cdots < j'_{m'}, D_{j_1} \circ \ldots \circ D_{j_m}(u) = D_{j'_1} \circ \ldots \circ D_{j'_{m'}}(v) \right\},$$

where D_j is deletion *operation at position $j \in [\![0, |u|[\![$ is defined over word $u \in A^*$ as follows: $D_j(u) = u_0 u_1 \ldots u_{j-1} u_{j+1} \ldots u_{|u|-1}$.*

Definition 2. *1. The Weyl pseudo-metric, denoted by $\hat{\mathfrak{d}}_H$, is defined as follows:*

$$\hat{\mathfrak{d}}_H(x,y) = \limsup_{\ell \to \infty} \max_{k \in \mathbb{N}} \frac{d_H(x_{[\![k,k+\ell[\![}, y_{[\![k,k+\ell[\![})}{\ell}, \forall x, y \in A^{\mathbb{N}},$$

2. The sliding pseudo-metric associated to the Levenshtein distance, denoted by $\hat{\mathfrak{d}}_L$, is defined as follows:

$$\hat{\mathfrak{d}}_L(x,y) = \limsup_{\ell \to \infty} \max_{k \in \mathbb{N}} \frac{d_L(x_{[\![k,k+\ell[\![}, y_{[\![k,k+\ell[\![})}{\ell}, \forall x, y \in A^{\mathbb{N}}.$$

It is easy to verify that these are pseudo-metrics *i.e.,* symmetric, zero over diagonal pairs, and satisfies the triangular inequality. On the other hand, these are not distances since we can find two different configurations between which the pseudometric is worth zero (for example, we can take two configurations with finitely many of differences). Hence, it is relevant to quotient the space of configurations by the equivalence of zero distance, in order to get a separated topological space:

Definition 3. *The relation $x \sim_{\hat{\mathfrak{d}}_*} y \iff \hat{\mathfrak{d}}_*(x,y) = 0$, is an equivalence relation. The quotient space $A^{\mathbb{N}} / \sim_{\hat{\mathfrak{d}}_*}$ called the Weyl space for $\hat{\mathfrak{d}}_* = \hat{\mathfrak{d}}_H$ and the sliding Feldman-Katok space when $\hat{\mathfrak{d}}_* = \hat{\mathfrak{d}}_L$, denoted $X_{\hat{\mathfrak{d}}_*}$, where $\hat{\mathfrak{d}}_*$ represent previous pseudo-metrics. We denote by $x_{\hat{\mathfrak{d}}_*}$ the equivalence class of $x \in A^{\mathbb{N}}$ in the quotient space. Any map F of $A^{\mathbb{N}}$ to itself such that $\hat{\mathfrak{d}}_*(x,y) = 0 \implies \hat{\mathfrak{d}}_*(F(x), F(y)) = 0$ for all $x, y \in A^{\mathbb{N}}$, induces a well-defined map $F_{\hat{\mathfrak{d}}_*} : X_{\hat{\mathfrak{d}}_*} \to X_{\hat{\mathfrak{d}}_*}$ over the quotient space. A map $F : A^{\mathbb{N}} \mapsto A^{\mathbb{N}}$ is $\hat{\mathfrak{d}}_*$-constant if for all $x, y \in A^{\mathbb{N}}$, $\hat{\mathfrak{d}}_*(F(x), F(y)) = 0$.*

Dill maps. Dill maps were defined in [ST15], and generelize both substitutions [FBF+02] and cellular automata [BR10]. Here we give an equivalent definition to [ST15, Definition 2].

Definition 4.

1. *A map* $F : A^{\mathbb{N}} \mapsto A^{\mathbb{N}}$ *is a* dill map *if there exist a diameter* $\theta \in \mathbb{N} \setminus \{0\}$ *and a* local rule $f : A^{\theta} \to A^{+}$ *such that for all* $x, y \in A^{\mathbb{N}}$*:* $F(x) = f(x_{[\![0,\theta[\![})f(x_{[\![1,\theta+1[\![})f(x_{[\![2,\theta+2[\![}) \cdots$.
2. *The* lower norm $|f|$ *and the* upper norm $\|f\|$ *of a dill map* F *with diameter* θ *and local rule* f *are defined by:* $|f| = \min\left\{ |f(u)| \,\middle|\, u \in A^{\theta} \right\}$ *and* $\|f\| = \max\left\{ |f(u)| \,\middle|\, u \in A^{\theta} \right\}$.
3. *We extend the local rule into a self-map* $f^* : A^* \to A^*$ *by:* $f^*(u) = f(u_{[\![0,\theta[\![})f(u_{[\![1,1+\theta[\![}) \cdots f(u_{[\![|u|-\theta,|u|[\![}})$, *for* u *such that* $|u| \geq \theta$ *and* $f^*(u) = \lambda$ *if* $|u| < \theta$.
4. *If* $\|f\| = |f|$*, then we say that* F *is* uniform.

When it is clear from the context, we may identify a dill map with its local rule.

Remark 5.

1. *Uniform dill maps with* $|f| = \|f\| = 1$ *are called cellular automata.*
2. *The local rule of a dill maps with diameter* $\theta = 1$ *is called substitution. In this case, we denote* τ *for the local rule and* $\bar{\tau}$ *for the dill map.*
3. *The composition of a substitution* τ *and a cellular automaton local rule* f *with diameter* θ *is a dill map local rule* $\tau \circ f$ *with diameter* θ.

Example 6.
1. *The shift is the CA with diameter* $\theta = 2$ *and local rule* f *defined by* $f(u_0u_1) = u_1$ *for all* $u_0, u_1 \in A$.
2. *Let* $A = \{a, b\}$. *The* Xor *is the CA with diameter* $\theta = 2$ *and local rule* f *defined by:* $f(aa) = f(bb) = a$ *and* $f(ab) = f(ba) = b$.
3. *The* Fibonacci *substitution defined over* A *by:* $\tau(a) = ab$ *and* $\tau(b) = a$.
4. *Let* f *be the local rule of the Xor CA and* τ *be the Fibonacci substitution. Then* $\tau \circ f$ *is a local rule of a dill map with diameter* 2 *and defined as follows:*

$$\tau \circ f(aa) = \tau \circ f(bb) = ab \text{ and } \tau \circ f(ba) = \tau \circ f(ab) = a.$$

In the Cantor space, an elegant characterization of cellular automata was given by Curtis, Hedlund and Lyndon in [Hed69] as follows: A function $F : A^{\mathbb{N}} \to A^{\mathbb{N}}$ is a cellular automaton if and only if it is continuous with respect to the Cantor metric and shift-equivariant (*i.e.,* $F(\sigma(x)) = \sigma(F(x))$, for all $x \in A^{\mathbb{N}}$). Similarly to the case of cellular automata, we gave a characterization of dill maps à la Hedlund. Recall that $\mathbb{N}$ can be naturally endowed with the discrete topology.

Theorem 7 ([RG22, Theorem 11])**.** *A function* $F : A^{\mathbb{N}} \to A^{\mathbb{N}}$ *is a dill map if and only if it is continuous over the Cantor space and there exists a continuous map* $s : A^{\mathbb{N}} \to \mathbb{N}$ *such that for all* $x \in A^{\mathbb{N}}$*:* $F(\sigma(x)) = \sigma^{s(x)}(F(x))$.

Before proving our main results, let us mention that this paper is a continuation of our previous work [RG22], and we suggest that the reader check it out.

2 Lipschitz property of dill maps with respect to $\hat{\mathfrak{d}}_H$

It is known since [BFK97] that every cellular automaton induces a (well-defined) Lipschitz function over the Weyl space. Some dill maps, on the contrary, are not well-defined.

Example 8. *The Fibonacci substitution is not well-defined over the Weyl space $X_{\hat{\mathfrak{d}}_H}$. For example, $\hat{\mathfrak{d}}_H(a^\infty, ba^\infty) = 0$ but $\hat{\mathfrak{d}}_H(\overline{\tau}(a^\infty), \overline{\tau}(ba^\infty)) = \hat{\mathfrak{d}}_H((ab)^\infty, (ba)^\infty) = 1$.*

Let us denote, for a uniform dill map F with local rule f and diameter θ:

$$d_f^+ = \max\left\{ d_H(f(u), f(v)) \middle| \, u, v \in A^\theta \right\} \text{ and } d_f^- = \min\left\{ d_H(f(u), f(v)) \middle| \, u \neq v \in A^\theta \right\}.$$

Lemma 9. *Let F be a uniform dill map with diameter θ and local rule f. Then for all $\ell, k \in \mathbb{N}$, for $m = \left\lceil \frac{k}{|f|} \right\rceil$, and for $p = \left\lfloor \frac{\ell+k}{|f|} \right\rfloor - (m+1)$:*

$$d_H(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) \leq d_H(x_{[\![m,m+p+\theta[\![}, y_{[\![m,m+p+\theta[\![}) \theta d_f^+ + 2\,|f|\,, \forall x, y \in A^{\mathbb{N}}.$$

Proof. Let $x, y \in A^{\mathbb{N}}$ and $\ell, k \in \mathbb{N}$. Since F is uniform, we can write:

$$F(x)_{[\![k,k+\ell[\![} = v f^*(x_{[\![m,m+p+\theta[\![}) w, \text{ and } F(y)_{[\![k,k+\ell[\![} = v' f^*(y_{[\![m,m+p+\theta[\![}) w',$$

where $|v| = |v'| \leq |f|$ and $|w| = |w'| \leq |f|$. By additivity, we can write then:

$$\begin{aligned}
d_H(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) - 2\,|f| &\leq \sum_{i=0}^{p} d_H(f(x_{[\![m+i,m+i+\theta[\![}), f(y_{[\![m+i,m+i+\theta[\![})) \\
&\leq \sum_{\substack{i=0 \\ x_{[\![m+i,m+i+\theta[\![} \neq y_{[\![m+i,m+i+\theta[\![}}}^{p} d_H(f(x_{[\![m+i,m+i+\theta[\![}), f(y_{[\![m+i,m+i+\theta[\![})) \\
&\leq \sum_{\substack{i=0 \\ \exists j \in [\![m+i,m+i+\theta[\![, x_j \neq y_j}}^{p} d_f^+ \quad \leq \sum_{j \in [\![m,m+p+\theta[\![} \sum_{\substack{i \in]\!]j-m-\theta, j-m]\!] \\ x_j \neq y_j}} d_f^+ \\
&\leq \sum_{\substack{j \in [\![m,m+p+\theta[\![\\ x_j \neq y_j}} \theta d_f^+ = d_H(x_{[\![m,m+p+\theta[\![}, y_{[\![m,m+p+\theta[\![}) \theta d_f^+. \quad \square
\end{aligned}$$

Proposition 10. *Let F be a uniform dill map with diameter θ and local rule f. Then:*

$$\hat{\mathfrak{d}}_H(F(x), F(y)) \leq \frac{\theta d_f^+}{|f|} \cdot \hat{\mathfrak{d}}_H(x, y), \forall x, y \in A^{\mathbb{N}}.$$

Proof. Let us prove that F is $\frac{\theta d_f^+}{|f|}$-Lipschitz with respect to $\hat{\mathfrak{d}}_H$. According to Lemma 9, for large enough ℓ, for $k \in \mathbb{N}$, $m = \left\lceil \frac{k}{|f|} \right\rceil$ and $p = \left\lfloor \frac{\ell+k}{|f|} \right\rfloor - (m+1)$ we obtain:

$$d_H(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) \leq d_H(x_{[\![m,m+p[\![}, y_{[\![m,m+p[\![}) \theta d_f^+ + \theta^2 d_f^+ + |f|\,.$$

Hence:

$$\frac{d_H(F(x)_{\llbracket k,k+\ell\rrbracket}, F(y)_{\llbracket k,k+\ell\rrbracket})}{\ell} \leq \frac{\max_{h\in\mathbb{N}} d_H(x_{\llbracket h,h+p\rrbracket}, y_{\llbracket h,h+p\rrbracket})\theta d_f^+ + \theta^2 d_f^+ + 2\,|f|}{\ell}$$
$$\leq \frac{\theta d_f^+}{|f|} \cdot \frac{\max_{h\in\mathbb{N}} d_H(x_{\llbracket h,h+p\rrbracket}, y_{\llbracket h,h+p\rrbracket})}{p} + \frac{\theta^2 d_f^+ + 2\,|f|}{\ell}.$$

Since this was true for every k and since $p \to \infty$ when $\ell \to \infty$, we obtain:

$$\hat{\mathfrak{d}}_H(F(x), F(y)) \leq \frac{\theta d_f^+}{|f|}\hat{\mathfrak{d}}_H(x, y). \quad \square$$

Proposition 11. *Let F be a dill map with diameter $\theta \in \mathbb{N} \setminus \{0\}$ and local rule f. If $F_{\hat{\mathfrak{d}}_H}$ is well-defined, then F is either constant or uniform.*

Proof. Assume that F is non-uniform, *i.e.,* there are two words u and v of equal length such that $|f^*(u)| \neq |f^*(v)|$. One can assume that their longest common suffix has length $\theta - 1$. Indeed, otherwise let $a \in A$, $u' = u_{\llbracket |u|-\theta+1,|u|\rrbracket}a^{\theta-1}$ and $v' = v_{\llbracket |u|-\theta+1,|v|\rrbracket}a^{\theta-1}$; one can note that $f^*(ua^{\theta-1}) = f^*(u)f^*(u')$ and $f^*(va^{\theta-1}) = f^*(v)f^*(v')$, so that either $\left|f^*(ua^{\theta-1})\right| \neq \left|f^*(va^{\theta-1})\right|$, or $|f^*(u')| = \left|f^*(ua^{\theta-1})\right| - |f^*(u)| \neq \left|f^*(va^{\theta-1})\right| - |f^*(v)| = |f^*(v')|$. Note that both of these pairs of words share a common suffix of length at least $\theta-1$. Assume also without loss of generality that $k = |f^*(u)| - |f^*(v)| > 0$.

- First assume that there exist $w \in A^*$ and $i \in \mathbb{N}$ such that $f^*(w)_i \neq f^*(w)_{i+k}$. By our previous assumption, we know that for $z = w^\infty$ and $w' = u_{\llbracket |u|-\theta,|u|\rrbracket}z_{\llbracket 0,\theta\rrbracket} = v_{\llbracket |v|-\theta,|v|\rrbracket}z_{\llbracket 0,\theta\rrbracket}$ we have: $F(uz) = f^*(u)f^*(w')F(z)$. According to the proof of [RG22, Theorem 20], we obtain: $\hat{\mathfrak{d}}_H(F(uz), F(vz)) \geq \mathfrak{d}_H(F(uz), F(vz)) \geq \frac{1}{|f^*(z_{\llbracket 0,|w|+\theta\rrbracket})|}$. Since $|u| = |v|$, $\hat{\mathfrak{d}}_H(uz, vz) = 0$, so that F is not well-defined with respect to $\hat{\mathfrak{d}}_L$.
- Otherwise, for all $w \in A^*$, $i \in \llbracket 0, |f^*(w)| - k\rrbracket$, we have $f^*(w)_i = f^*(w)_{i+k}$. Then for every $x \in A^{\mathbb{N}}$, $F(x)$ is k-periodic and thus $F(x) = (f^*(x_{\llbracket 0,k+\theta\rrbracket})_{\llbracket 0,k\rrbracket})^\infty$. Assuming that F is well-defined, we get $\mathfrak{d}_H(F(x), F(0^{k+\theta}x_{\llbracket k+\theta,\infty\rrbracket})) = 0$. According to Proposition [CFMM97, Proposition 3], we can deduce that $F(x) = F(0^{k+\theta}x_{\llbracket k+\theta,\infty\rrbracket})$. Then, for every $x \in A^{\mathbb{N}}$, $F(x) = (f^*(0^{k+\theta})_{\llbracket 0,k\rrbracket})^\infty$. Hence, F is constant. $\square$

We now reach necessary and sufficient conditions for dill maps to induce well-defined maps over this space.

Theorem 12. *Let F be a dill map with diameter θ and local rule f. Then the following statements are equivalent:*

1. *$F_{\hat{\mathfrak{d}}_H}$ is well-defined.*
2. *F is $\frac{\theta d_f^+}{|f|}$-Lipschitz with respect to $\hat{\mathfrak{d}}_H$.*
3. *F is either constant or uniform.*

Proof. $2 \implies 1$ is clear from the definition of Lipschitz maps. Implication $3 \implies 2$ follows from Proposition 10. Implication $1 \implies 3$ follows from Proposition 11. $\square$

3 Lipschitz property of dill maps with respect to $\hat{\mathfrak{d}}_L$

In [RG22], we proved that all dill maps are well-defined in the Feldman-Katok space. However, by changing the Feldman-Katok pseudo-metric for $\hat{\mathfrak{d}}_L$, one can find that not all dill maps are well-defined. See for instance the following example:

Example 13. *Let τ be a substitution defined over $A = \{0,1\}^{\mathbb{N}}$ by $\tau(0) = 0$ and $\tau(1) = 11$. Let $x = (0,1)^{(n,n)_{n\in\mathbb{N}\setminus\{0\}}}$ and $y = (0,1)^{(n+1,n-1)_{n\in\mathbb{N}\setminus\{0\}}}$. Note that for all $j \in \mathbb{N}$, since $s_j := j(j+1) = \sum_{i=1}^{j} 2i$, we obtain: $d_H(x_{[\![s_j,s_{j+1}[\![}, y_{[\![s_j,s_{j+1}[\![}) = 1$. Let $\ell \in \mathbb{N}\setminus\{0\}$ and $k \in \mathbb{N}$. For $p = \min\{\, j \in \mathbb{N} |\, s_j \geq k\}$, $m = \max\{\, j \in \mathbb{N} |\, s_j \leq k+\ell\}$ and by subadditivity:*

$$\begin{aligned} d_H(x_{[\![k,k+\ell[\![}, y_{[\![k,k+\ell[\![}) &= d_H(x_{[\![k,s_p[\![}, y_{[\![k,s_p[\![}) + d_H(x_{[\![s_p,s_m[\![}, y_{[\![s_p,s_m[\![}) + d_H(x_{[\![s_m,k+\ell[\![}, y_{[\![s_m,k+\ell[\![}) \\ &\leq 1 + d_H(x_{[\![s_p,s_m[\![}, y_{[\![s_p,s_m[\![}) + 1 \leq (m-p) + 2. \end{aligned}$$

Moreover, since $\ell + k \geq m^2 + m$ and $k \leq p^2 + p$, we obtain $\frac{m-p}{\ell} \leq \frac{1}{m+p+1}$, and thus:

$$\frac{d_H(x_{[\![k,k+\ell[\![}, y_{[\![k,k+\ell[\![})}{\ell} \leq \frac{m-p+2}{\ell} \leq \frac{2}{m+p+1} + \frac{2}{\ell}.$$

Since m tends to ∞ when ℓ tends to ∞: $\lim_{\ell\to\infty} \dfrac{d_H(x_{[\![k,k+\ell[\![}, y_{[\![k,k+\ell[\![})}{\ell} = 0, \forall k \in \mathbb{N}$.
Thus $\hat{\mathfrak{d}}_L(x,y) = \hat{\mathfrak{d}}_H(x,y) = 0$. On the other hand, it is clear that:

$$\overline{\tau}(x) = (0,1)^{(n,2n)_{n\in\mathbb{N}\setminus\{0\}}} \text{ and } \overline{\tau}(y) = (0,1)^{(n+1,2(n-1))_{n\in\mathbb{N}\setminus\{0\}}}.$$

For $\ell \in \mathbb{N}\setminus\{0\}$ and $k = \sum_{i=1}^{\ell}(i+1) + \sum_{i=1}^{\ell} 2(i-1)$, hence: $\overline{\tau}(y)_{[\![k,k+\ell[\![} = 0^\ell$. In contrast, since $\left(\sum_{i=1}^{\ell} i + \sum_{i=1}^{\ell} 2i\right) - \ell = k$, we obtain $\overline{\tau}(x)_{[\![k,k+\ell[\![} = 1^\ell$. Thus:

$$d_L(\overline{\tau}(x)_{[\![k,k+\ell[\![}, \overline{\tau}(y)_{[\![k,k+\ell[\![}) = d_L(0^\ell, 1^\ell) = \ell.$$

Hence, $\max_{h\in\mathbb{N}} d_L(\overline{\tau}(x)_{[\![h,h+\ell[\![}, \overline{\tau}(y)_{[\![h,h+\ell[\![}) = \ell$. Therefore, $\hat{\mathfrak{d}}_L(\overline{\tau}(x), \overline{\tau}(y)) = 1$. In conclusion, $\overline{\tau}$ is not well-defined with respect ot $\hat{\mathfrak{d}}_L$.

Before giving a caracterization of well-defined dill-maps with respect to $\hat{\mathfrak{d}}_L$, let us recall a result from [RG22] to be used in the proof of the main result of this section.

Lemma 14 ([RG22])**.** *Let F be a dill map with diameter θ and local rule f. Then for all $\ell \in \mathbb{N}$ and $u, v \in A^\ell$, we have: $d_L(f^*(u), f^*(v)) \leq \|f\|\,(2\theta-1)d_L(u,v) - \frac{\big||f^*(u)|-|f^*(v)|\big|}{2}$.*

Now let us characterize dill maps which induce a well-defined function over $X_{\hat{\mathfrak{d}}_L}$.

Definition 15. *We say that a dill map with diameter θ and local rule f is* diamond-uniform *if for every ℓ and $u, v \in A^\ell$ such that $u_{[\![0,\theta[\![} = v_{[\![0,\theta[\![}$ and $u_{[\![\ell-\theta,\ell[\![} = v_{[\![\ell-\theta,\ell[\![}$, one has $|f^*(u)| = |f^*(v)|$.*

It is clear that all uniform dill maps are diamond-uniform. Here is an example of non-uniform dill map which is diamond-uniform.

Example 16. *Let F be the dill map with diameter $\theta = 2$ and local rule f defined by:*

$$f(aa) = ab, f(bb) = ba, f(ab) = a \text{ and } f(ba) = bab.$$

It is clear that for any $x \in A^{\mathbb{N}}$, $F(x) = (ab)^\infty$ if x start by the letter a and $F(x) = (ba)^\infty$ otherwise. And thus, F is neither constant nor uniform. However, F is $\hat{\mathfrak{d}}_L$-constant, since for every $x, y \in A^{\mathbb{N}}$, $\hat{\mathfrak{d}}_L(F(x), F(y)) = 0$. So, $F_{\hat{\mathfrak{d}}_L}$ is well-defined.

Theorem 17. *If F is a dill map, then $F_{\hat{\mathfrak{d}}_L}$ is well-defined if and only if F is $\hat{\mathfrak{d}}_L$-constant or diamond-uniform.*

In order to prove this, here is the key point about diamond-uniformity.

Lemma 18. *If F is a dill map with diameter θ and local rule f, then the following statements are equivalent:*

1. *F is diamond-uniform.*
2. *$|f^*(u)| - |f^*(v)|$ is uniformly bounded, for every u, v with equal length.*

Proof.

$1 \Rightarrow 2$ Let u, v have equal length. Then

$$\begin{aligned} |f^*(u)| - |f^*(v)| &= \left|f^*(a^{\theta-1}ua^{\theta-1})\right| - \left|f^*(a^{\theta-1}va^{\theta-1})\right| - \left|f^*(a^{\theta-1}u_{[\![0,\theta-1[\![})\right| + \\ &\quad + \left|f^*(a^{\theta-1}v_{[\![0,\theta-1[\![})\right| - \left|f^*(u_{[\![|u|-\theta-1,|u|[\![}a^{\theta-1})\right| + \left|f^*(v_{[\![|u|-\theta-1,|u|[\![}a^{\theta-1})\right| \\ &\leq 2(\theta-1)(d_f^+ - d_f^-). \end{aligned}$$

$2 \Rightarrow 1$ Assume, by contrapositive, that there exist $\ell \in \mathbb{N}$ and $u, v \in A^\ell$ such that:

$$u_{[\![0,\theta-1[\![} = v_{[\![0,\theta-1[\![}, u_{[\![\ell-\theta+1,\ell[\![} = v_{[\![\ell-\theta+1,\ell[\![} \text{ and } |f^*(u)| - |f^*(v)| \neq 0.$$

Then $\left|f^*(u^k)\right| - \left|f^*(v^k)\right| = (|f^*(u)| - |f^*(v)|)k$, thanks to the common prefixes and suffixes. Hence $|f^*(u)| - |f^*(v)|$ is not bounded (and not upper-bounded, up to swapping u and v). □

Proposition 19. *Any diamond-uniform dill map F with local rule f and diameter θ is $(2\theta - 1) \times \dfrac{\|f\|}{|f|}$-Lipschitz with respect to $\hat{\mathfrak{d}}_L$.*

Proof. Let $x, y \in A^{\mathbb{N}}$. For large enough ℓ and $k \in \mathbb{N}$ let us denote:

$$m = \max\left(\min\left\{ i \in \mathbb{N} \middle| |f^*(x_{[\![0,i+\theta[\![})| \geq k \right\}, \min\left\{ i \in \mathbb{N} \middle| |f^*(y_{[\![0,i+\theta[\![})| \geq k \right\}\right), \text{ and}$$
$$p = \min\left(\max\left\{ i \in \mathbb{N} \middle| |f^*(x_{[\![0,m+p[\![})| \leq k + \ell \right\}, \min\left\{ i \in \mathbb{N} \middle| |f^*(y_{[\![0,m+p[\![})| \leq k + \ell \right\}\right).$$

Since F is a diamond-uniform dill map and thanks to 2, there exist $C > 0$ such that:

$$\big|\big|f^*(x_{[\![0,m[\![})\big| - \big|f^*(y_{[\![0,m[\![})\big|\big| \leq C \text{ and } \big|\big|f^*(x_{[\![0,m+p[\![})\big| - \big|f^*(y_{[\![0,m+p[\![})\big|\big| \leq C.$$

And thus we can write,

$$F(x)_{[\![k,k+\ell[\![} = uvf^*(x_{[\![m,m+p[\![})wz \text{ and } F(y)_{[\![k,k+\ell[\![} = u'v'f^*(y_{[\![m,m+p[\![})w'z'$$

where $|u|, |u'|, |z|, |z'| < \|f\|$, and $|v|, |v'|, |w|, |w'| < C$. Hence, by subadditivity:

$$\begin{aligned} d_L(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) &\leq d_L(uv, u'v') + d_L(f^*(x_{[\![m,m+p[\![}), f^*(y_{[\![m,m+p[\![})) + d_L(wz, w'z') \\ &\leq 2(\|f\| + C) + d_L(f^*(x_{[\![m,m+p[\![}), f^*(y_{[\![m,m+p[\![})). \end{aligned}$$

According to Lemma 14 we obtain:

$$d_L(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) \leq 2(\|f\|+C)+(2\theta-1)\,\|f\|\,d_L(x_{[\![m,m+p[\![}, y_{[\![m,m+p[\![}).$$

Since $\ell \geq \big|f^*(x_{[\![m,m+p[\![})\big| \geq (p-\theta)\,|f|$ and by subadditivity:

$$\begin{aligned} \frac{d_L(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![})}{\ell} &\leq \frac{\|f\|\,(2+\theta(2\theta-1)) + 2C}{\ell} + \frac{(2\theta-1)\,\|f\|\,d_L(x_{[\![m,m+p-\theta[\![}, y_{[\![m,m+p-\theta[\![})}{|f|\,(p-\theta)} \\ &\leq \frac{\|f\|\,(2\theta^2-\theta+2)+2C}{\ell} + (2\theta-1)\frac{\|f\|}{|f|} \times \max_{h\in\mathbb{N}} \frac{d_L(x_{[\![h,h+p-\theta[\![}, y_{[\![h,h+p-\theta[\![})}{p-\theta}. \end{aligned}$$

Since this was true for every $k \in \mathbb{N}$ and since $p \to \infty$ when $\ell \to \infty$:

$$\hat{\mathfrak{d}}_L(F(x), F(y)) \leq (2\theta-1)\frac{\|f\|}{|f|}\hat{\mathfrak{d}}_L(x,y). \quad \square$$

Proof of Theorem 17. According to Proposition 19, if F is diamond-uniform then $F_{\hat{\mathfrak{d}}_L}$ is well-defined. Suppose now that F is neither $\hat{\mathfrak{d}}_L$-constant nor diamond-uniform *i.e.,* there exists $x, y \in A^{\mathbb{N}}$ such that $\hat{\mathfrak{d}}_L(F(x), F(y)) > 0$, and there exists $m \in \mathbb{N}$, $u, v \in A^m$ such that $u_{[\![0,\theta-1[\![} = v_{[\![0,\theta-1[\![}$, $u_{[\![m-\theta+1,m[\![} = v_{[\![m-\theta+1,m[\![}$ and $\alpha := |f^*(u)| - |f^*(v)| > 0$. Let us define the following configurations:

$$\begin{aligned} z &:= ux_{[\![0,\alpha[\![}y_{[\![0,\alpha[\![}ux_{[\![0,2\alpha[\![}y_{[\![0,2\alpha[\![}ux_{[\![0,3\alpha[\![}y_{[\![0,3\alpha[\![}\dots, \text{ and} \\ w &:= vx_{[\![0,\alpha[\![}y_{[\![0,\alpha[\![}vx_{[\![0,2\alpha[\![}y_{[\![0,2\alpha[\![}vx_{[\![0,3\alpha[\![}y_{[\![0,3\alpha[\![}\dots. \end{aligned}$$

Let $(k_\ell)_{\ell\in\mathbb{N}}$ such that for all $\ell \in \mathbb{N}$: $\max_{k\in\mathbb{N}} d_L(F(x)_{[\![k,k+\ell[\![}, F(y)_{[\![k,k+\ell[\![}) = d_L(F(x)_{[\![k_\ell,k_\ell+\ell[\![}, F(y)_{[\![k_\ell,k_\ell+\ell[\![})$. This pair of patterns appears in (z,w), in the zone where $n\alpha \geq k_\ell+\ell$. Hence $\hat{\mathfrak{d}}_L(z,w) = 0$ but $\hat{\mathfrak{d}}_L(F(z), F(w)) \geq \hat{\mathfrak{d}}_L(F(x), F(y)) > 0$. $\square$

4 Conclusion and perspectives

In this paper, we characterized well-defined dill maps over the Weyl space, indeed, we find the same result as in the case of Besicovitch space in [RG22]. In addition,

we showed that not all dill maps are well-defined with respect to the sliding Feldman-Katok space, in contrast the Feldman-Katok space, where all dill maps are well-defined [RG22, Corollary 46]. Those spaces were constructed using two pseudo-metrics depending on two different edit distances over finite words (the Hamming distance and the Levenshtein distance). One natural question is which properties on distance d make all dill maps are well-defined in the corresponding pseudo-metric space?

References

BFK97. François Blanchard, Enrico Formenti, and Petr Kůrka. Cellular automata in the Cantor, Besicovitch, and Weyl topological spaces. *Complex Systems*, 11:107–123, 1997.

BR10. Valérie Berthé and Michel Rigo, editors. *Combinatorics, automata, and number theory.*, volume 135 of *Encyclopedia of Mathematics and Its Applications*. Cambridge: Cambridge University Press, 2010.

CFMM97. Gianpiero Cattaneo, Enrico Formenti, Luciano Margara, and Jacques Mazoyer. A shift-invariant metric on $S^{\mathbb{Z}}$ inducing a non-trivial topology. In *International Symposium on Mathematical Foundations of Computer Science*, pages 179–188. Springer, 1997.

FBF$^+$02. N. Pytheas Fogg, Valérie Berthé, Sébastien Ferenczi, Christian Mauduit, and Anne Siegel, editors. *Substitutions in dynamics, arithmetics and combinatorics*, volume 1794 of *Lecture Notes in Mathematics*. Berlin: Springer, 2002.

Hed69. Gustav A Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, 3(4):320–375, 1969.

RG22. Firas Ben Ramdhane and Pierre Guillon. Cellular automata and substitutions in topological spaces defined via edit distances. *arXiv:2203.16226*, 2022.

ST15. Ville Salo and Ilkka Törmä. Block maps between primitive uniform and Pisot substitutions. *Ergodic Theory and Dynamical Systems*, 35(7):2292–2310, 2015.

Words fixing the kernel network and maximum independent sets in graphs

Maximilien Gadouleau[1][0000−0003−4701−738X] and David C. Kutner[1][0000−0003−2979−4513]

Durham University, UK
{m.r.gadouleau,david.c.kutner}@durham.ac.uk

Abstract. The simple greedy algorithm to find a maximal independent set of a graph can be viewed as a sequential update of a Boolean network, where the update function at each vertex is the conjunction of all the negated variables in its neighbourhood. In general, the convergence of the so-called kernel network is complex. A word (sequence of vertices) fixes the kernel network if applying the updates sequentially according to that word always leads to a fixed point whatever the initial configuration. We prove that determining whether a word fixes the kernel network is coNP-complete. We also consider the so-called permis, which are permutation words that fix the kernel network. We exhibit large classes of graphs that have a permis, but we also construct many graphs without a permis.

Keywords: Boolean networks · Graph theory · Maximal independent set · Kernel.

1 Introduction

1.1 Background

A simple greedy algorithm to find a maximal independent set in a graph works as follows. Starting with the empty set, visit each vertex in the graph, and add it to the set whenever none of its neighbours are already in the set. This can be interpreted in terms of Boolean networks as follows: starting with the all-zero configuration x, update one vertex v at a time according to the update function $\bigwedge_{u\sim v} \neg x_u$. Once all vertices have been updated, we obtain the final configuration y where the set of ones is a maximal independent set, regardless of the order in which the vertices have been updated.

We refer to the Boolean network where the update function is the conjunction of all the negated variables in the neighbourhood of a vertex as the *kernel network* on the graph. We use this terminology, as kernels are the natural generalisation of maximal independent sets to digraphs (they are the independent dominating sets). This class of networks has been the subject of some study; we refer to two works in particular.

In [2], the fixed points of different conjunctive networks on graphs are studied. In particular, it is shown that the set of fixed points of the kernel network is the

set of (configurations whose coordinates equal to one are) maximal independent sets of the graph. They further prove that for square-free graphs, the kernel network is the conjunctive network that maximises the number of fixed points.

In a completely different setting, Yablo discovered the first non-self-referential paradox in [10]. The construction for this paradox implicitly applies the fact that the kernel network on a transitive tournament on $\mathbb{N}$ has no fixed point. The study of acyclic digraphs that admit a paradox is continued further in [8], where the kernel network is referred to as an $\mathcal{F}$-system.

Boolean networks are used to model networks of interacting entities. As such, it is natural to consider a scenario whereby the different entities update their state at different times. This gives rise to the notion of sequential (or asynchronous) updates. The problem of whether a Boolean network converges sequentially goes back to the seminal result by Robert on acyclic interaction graphs [9]; further results include [6,5,7]. Recently, [1] introduced the concept of a fixing word: a sequence of vertices such that updating vertices according to that sequence will always lead to a fixed point, regardless of the initial configuration. Large classes of Boolean networks have short fixing words [1,4].

1.2 Contributions and outline

Our main result is to show that determining whether a word fixes the kernel network is an NP-hard problem. Our seminal remark is that if w is any permutation of the vertices, then w maps any configuration to an independent set (we say w prefixes the kernel network) and w maps any independent set to a kernel (we say w suffixes the kernel network), and as such ww fixes the kernel network. Once again, whether a word prefixes or suffixes the kernel network only depends on the set of vertices visited by the word. Determining whether a word prefixes the kernel network can be done in polynomial time, while it is coNP-complete to determine whether it suffixes the kernel network. We then determine the sets of vertices S for which there exists a word fixing the kernel network that only visits S; deciding whether S is one such set is NP-hard. We use the intractability of that last problem to prove our main result.

We then go back to our interpretation of the kernel network in terms of the greedy algorithm for finding a maximal independent set. In that algorithm, the initial configuration is fixed and the permutation of vertices is arbitrary. We then consider fixing the permutation and varying the initial configuration instead. Thus we introduce the notion of a permis, i.e. a permutation that fixes the kernel network. We exhibit large classes of graphs which do have a permis, and some examples and constructions of graphs which do not have a permis.

The rest of the paper is organised as follows. Some necessary background on Boolean networks is reviewed in Section 2. In Section 3 we classify those words which prefix or suffix the kernel network and show that determining whether a word fixes the kernel network is coNP-complete. Lastly, we study graphs that have a permis in Section 4.

Due to space limitations, some proofs are given in the appendix, and their sketches are given in the main text instead. We presume that the reader is familiar with some basic graph theory; otherwise, they are directed to [3].

2 Preliminaries

2.1 Boolean networks

A **configuration** on a graph $G = (V, E)$ is $x \in \{0,1\}^V = (x_v : v \in V)$, where $x_v \in \{0,1\}$ is the state of the vertex v for all v. We denote $\mathbf{1}(x) = \{v \in V : x_v = 1\}$ and $\mathbf{0}(x) = \{v \in V : x_v = 0\}$. For any set of vertices $S \subseteq V$, we denote $x_S = (x_v : v \in S)$. We denote the all-zero (all-one, respectively) configuration by 0 (by 1, respectively), regardless of its length.

A **Boolean network** is a mapping $\mathsf{F} : \{0,1\}^V \to \{0,1\}^V$. For any Boolean network F and any $v \in V$, the update of the state of vertex v is represented by the network $\mathsf{F}^v : \{0,1\}^V \to \{0,1\}^V$ where $\mathsf{F}^v(x)_v = \mathsf{F}(x)_v$ and $\mathsf{F}^v(x)_u = x_u$ for all other vertices u. We extend this notation to words as follows: if $w = w_1 \dots w_l$ then

$$\mathsf{F}^w = \mathsf{F}^{w_l} \circ \cdots \circ \mathsf{F}^{w_2} \circ \mathsf{F}^{w_1}.$$

Unless otherwise specified, we let x be the initial configuration, $w = w_1 \dots w_l$ a word, $y = \mathsf{F}^w(x)$ be the final configuration, and for all $0 \le a \le l$, $y^a = \mathsf{F}^{w_1 \dots w_a}(x)$ be an intermediate configuration, so that $x = y^0$ and $y = y^l$.

The set of **fixed points** of F is $\mathrm{Fix}(\mathsf{F}) = \{x \in \{0,1\}^V : \mathsf{F}(x) = x\}$. The word w **fixes** F if for all x, $\mathsf{F}^w(x) \in \mathrm{Fix}(\mathsf{F})$.

2.2 The kernel network

Let $G = (V, E)$ be a graph. The **kernel network** on G, denoted as $\mathsf{K}(G)$ is defined by

$$\mathsf{K}(x)_v = \bigwedge_{u \sim v} \neg x_u,$$

with $\mathsf{K}(x)_v = 1$ if $N(v) = \emptyset$.

An **independent set** I is a set such that $ij \notin E$ for all $i, j \in I$. The collection of all configurations x of G such that $\mathbf{1}(x)$ is an independent set of G is denoted by $\mathrm{I}(G)$. A **dominating set** D is a set such that for every vertex $v \in V$, either $v \in D$ or there exists $u \in D$ such that $uv \in E$. A **kernel** K is a dominating independent set. Equivalently, a kernel is a **maximal independent set** of G, i.e. an independent set K such that there is no independent set $J \supset K$. The collection of all configurations x of G such that $\mathbf{1}(x)$ is a kernel of G is denoted by $\mathrm{K}(G)$. It is easily seen (for instance, in [2]) that $\mathrm{Fix}(\mathsf{K}(G)) = \mathrm{K}(G)$.

3 Words fixing the kernel network

We now focus on words fixing the kernel network. Whether a word fixes the kernel network does not only depend on the set of vertices it visits. For example, if G is the path on the three vertices a, b, c with edges ab, bc, then $w = abc$ does not fix $\mathsf{K}(G)$ (if $x = 111$, then $y = 001$), while it is easily checked that $w = acb$ does fix $\mathsf{K}(G)$. In general, characterising the fixing words for $\mathsf{K}(G)$ remains an open problem. However, we manage to prove that deciding whether a word fixes the kernel network is computationally hard.

We define Fixing Word to be the decision problem asking, for an instance (G, w), whether w fixes $\mathsf{K}(G)$.

Theorem 1. Fixing Word *is coNP-complete.*

The rest of this section is devoted to the proof of Theorem 1.

The set of vertices visited by a word w is denoted by $[w] = \{v \in V : \exists a, v = w_a\}$. A **permutation** of V (or of G) is a word $w = w_1 \dots w_n$ such that $[w] = V$ and $w_a \neq w_b$ for all $a \neq b$. If w is a permutation of G, then ww fixes $\mathsf{K}(G)$: for any initial configuration x, $\mathsf{K}^w(x) \in \mathrm{I}(G)$; then for any $y \in \mathrm{I}(G)$, $\mathsf{K}^w(y) \in \mathrm{K}(G)$.

Accordingly, we say that w^{p} **prefixes** $\mathsf{K}(G)$ if $\mathsf{K}^{w^{\mathrm{p}}}(x) \in \mathrm{I}(G)$ for all $x \in \{0,1\}^n$, and that w^{s} **suffixes** $\mathsf{K}(G)$ if $\mathsf{K}^{w^{\mathrm{s}}}(y) \in \mathrm{K}(G)$ for all $y \in \mathrm{I}(G)$. In that case, for any word ω, $w^{\mathrm{p}}\omega$ also prefixes $\mathsf{K}(G)$ and ωw^{s} also suffixes $\mathsf{K}(G)$. Clearly, if $w = w^{\mathrm{p}}w^{\mathrm{s}}$, where w^{p} prefixes $\mathsf{K}(G)$ and w^{s} suffixes $\mathsf{K}(G)$, then w fixes $\mathsf{K}(G)$. We can be more general, as shown below.

Proposition 1. *If $w = w_1 \dots w_l$ where $w_1 \dots w_a$ prefixes $\mathsf{K}(G)$, $w_b \dots w_l$ suffixes $\mathsf{K}(G)$, and $[w_b \dots w_a]$ is an independent set of G for some $0 \leq a, b \leq l$, then w fixes $\mathsf{K}(G)$.*

Proof. First, suppose $a < b$, so that $w = w_1 \dots w_a \dots w_b \dots w_l$. As mentioned above, $w^{\mathrm{p}} = w_1 \dots w_{b-1}$ prefixes $\mathsf{K}(G)$ and $w^{\mathrm{s}} = w_b \dots w_l$ suffixes $\mathsf{K}(G)$, hence $w = w^{\mathrm{p}}w^{\mathrm{s}}$ fixes $\mathsf{K}(G)$.

Second, suppose $a \geq b$, so that $w = w_1 \dots w_b \dots w_a \dots w_l$. It is easily seen that if $u \not\sim v$, $\mathsf{K}^{vv} = \mathsf{K}^v$ and $\mathsf{K}^{uv} = \mathsf{K}^{vu}$. As such,

$$\mathsf{K}^w = \mathsf{K}^{w_1 \dots w_b w_b \dots w_a w_a \dots w_l} = \mathsf{K}^{w_1 \dots w_b \dots w_a w_b \dots w_a \dots w_l},$$

and again if we let $w^{\mathrm{p}} = w_1 \dots w_a$ and $w^{\mathrm{s}} = w_b \dots w_l$, we have $\mathsf{K}^w = \mathsf{K}^{w^{\mathrm{p}}w^{\mathrm{s}}}$, hence w fixes $\mathsf{K}(G)$. □

We now characterise the words that prefix (or suffix) the kernel network. Interestingly, those properties depend only on $[w]$.

Proposition 2. *Let G be a graph. Then w prefixes $\mathsf{K}(G)$ if and only if $[w]$ is a vertex cover of G.*

Proof. Suppose $[w]$ is a vertex cover of G and that $y = \mathsf{K}^w(x) \notin \mathrm{I}(G)$, i.e. $y_{uv} = 11$ for some edge uv of G. Without loss, let the last update in $\{u, v\}$ be

v, i.e. there exists a such that $w_a = v$ and $w_b \notin \{u, v\}$ for all $b > a$. Let $z = \mathsf{K}^{w_1 \dots w_{a-1}}(x)$, then $z_u = y_u = 1$ hence $y_v = 0$, which is the desired contradiction.

Conversely, if $[w]$ is not a vertex cover, then there is an edge $uv \in E$ such that $[w] \cap \{u, v\} = \emptyset$. Therefore, for any x with $x_{uv} = 11$, we have $y_{uv} = 11$ as well. □

Corollary 1. *Given a graph G and a word w, determining whether w prefixes $\mathsf{K}(G)$ is in P.*

A subset S of vertices of a graph is a **colony** if there exists an independent set I such that $S \subseteq N(I)$. Alternatively, a colony is a set S such that $V \setminus S$ contains a maximal independent set. A subset W of vertices is a **dominion** if there exists $v \in V \setminus W$ such that $W \cap N(v)$ is a colony of $G - v$. A **non-dominion** is a set of vertices that is not a dominion.

Proposition 3. *Let G be a graph. Then the word w suffixes $\mathsf{K}(G)$ if and only if $[w]$ is a non-dominion of G.*

Proof. Suppose $[w]$ is a dominion of G, i.e. there exists an independent set I and a vertex $v \notin [w]$ such that $W = [w] \cap N(v)$ is in the neighbourhood of I. Let x such that $x_I = 1$ and $x_{V \setminus I} = 0$, and let $y = \mathsf{K}^w(x)$. Then for any $u \in W$, u has a neighbour in I, hence $y_u = 0$; thus $y_{N[v]} = 0$ and w does not suffix K.

Conversely, suppose there exists x and v such that $y = \mathsf{K}^w(x)$ with $y_{N[v]} = 0$. Then $x_{N[v]} = 0$. Let $W = [w] \cap N(v)$ and $I = \mathbf{1}(y) \cap N(W)$; we note that I is an independent set. For each $u \in W$, we have $y_u = 0$ hence there exists $i \in I$ such that $u \in N(i)$. Therefore, $W \subseteq N(I)$ and W is a colony of $G - v$. □

The Colony (respectively, Dominion, Non-Dominion) problem asks, given a graph G and set T, if T a colony (resp. a dominion, a non-dominion) of G.

Theorem 2. *Given G and w, determining whether w suffixes $\mathsf{K}(G)$ is coNP-complete.*

Proof (Sketch). The proof is by successive reductions: Set Cover to Colony to Dominion. The full proof is given in Appendix A.

We now characterise the sets of vertices S visited by fixing words of the kernel network. Interestingly, those are the same sets S such that ww is a fixing word for any permutation w of S.

Proposition 4. *Let S be a subset of vertices of G. The following are equivalent.*

1. *There exists a word w with $[w] = S$ that fixes $\mathsf{K}(G)$.*
2. *For all w^{p}, w^{s} such that $[w^{\mathrm{p}}] = [w^{\mathrm{s}}] = S$, the word $w^{\mathrm{p}} w^{\mathrm{s}}$ fixes $\mathsf{K}(G)$.*
3. *S is a vertex cover and a non-dominion.*

Proof. Clearly, $2 \implies 1$. We now prove $1 \implies 3$. Since w prefixes $\mathsf{K}(G)$, $S = [w]$ is a vertex cover by Proposition 2; similarly, since w suffixes $\mathsf{K}(G)$, $S = [w]$ is a non-dominion by Proposition 3. Finally, we prove $3 \implies 2$. Since S is a vertex cover, then by Proposition 2 w^{p} prefixes $\mathsf{K}(G)$; similarly, by Proposition 3 w^{s} suffixes $\mathsf{K}(G)$. Therefore, $w^{\mathrm{p}} w^{\mathrm{s}}$ fixes $\mathsf{K}(G)$. □

Let Fixing Set be the decision problem, where the instance is (G, S) and the question is: does there exist a word w with $[w] = S$ that fixes $\mathsf{K}(G)$, or equivalently is S a vertex cover and a non-dominion?

Theorem 3. Fixing Set *is coNP-complete.*

Proof. The proof is by reduction from Non-Dominion (which is coNP-complete) to Fixing Set. The full proof is given in Appendix B.

We now finalise the proof of Theorem 1.

Proof (of Theorem 1). Fixing Word is in coNP; the certificate being a configuration x such that $\mathsf{K}^w(x) \notin \mathrm{K}(G)$. The proof of hardness is by reduction from Fixing Set, which is coNP-complete, as shown in Theorem 3. Let (G, S) be an instance of Fixing Set, then consider the instance $(G, w = \omega\omega)$ of Fixing Word, where ω is a permutation of S. Then Proposition 4 shows that w fixes $\mathsf{K}(G)$ if and only if S is a vertex cover and a non-dominion.

The complexity of determining the length of a shortest fixing word for the kernel network remains open.

Question 1. What is the complexity of the following optimisation problem: given G, what is the length of a shortest fixing word for $\mathsf{K}(G)$?

4 Graphs with a permis

Let $G = (V, E)$ be a graph. The greedy algorithm to find a maximal independent set of G fixes the initial configuration x to 0, and varies the permutation w, while always obtaining a maximal independent set $y \in \mathrm{K}(G)$. We now turn the tables, and instead consider fixing the permutation to some w and varying the initial configuration x; we want to find a permutation that guarantees that we always obtain a maximal independent set $y \in \mathrm{K}(G)$. As such, we call a permutation of G that fixes $\mathsf{K}(G)$ a **permis** for G.

We now investigate which graphs have a permis. We first exhibit large classes of graphs that do have a permis in Theorem 4. For that purpose, we need to review some graph theory first.

A graph is a **comparability graph** if there exists a partial order $\sqsubseteq$ on V such that $uv \in E$ if and only if $u \sqsubset v$. The following are comparability graphs: complete graphs, bipartite graphs, permutation graphs, and interval graphs.

A vertex is **simplicial** if its neighbourhood is a clique, i.e. if $N[s] \subseteq N[v]$ for all $v \in N[s]$.

We now introduce an operation on graphs, that we call **graph composition**. Let H be an n-vertex graph, $G_1, \ldots, G_n$ other graphs, then the composition $H(G_1, \ldots, G_n)$ is obtained by replacing each vertex v of H by the graph G_v, and whenever $uv \in E(H)$, adding all edges between G_u and G_v. This construction includes for instance the disjoint union of two graphs: $G_1 \cup G_2 = \bar{K}_2(G_1, G_2)$; the full union with all edges between G_1 and G_2: $K_2(G_1, G_2)$; adding an open twin (a new vertex v' with $N(v') = N(v)$ for some vertex v of H): $H(K_1, \ldots, K_1, \bar{K}_2, K_1, \ldots, K_1)$; similarly, adding a closed twin ($N[v'] = N[v]$).

Theorem 4. *Let G be a graph. If G satisfies any of the following properties, then G has a permis:*

1. *G has at most seven vertices, and is not the heptagon C_7;*
2. *G is a comparability graph;*
3. *the set of simplicial vertices of G is a dominating set;*
4. *$G = H(G_1, \ldots, G_n)$, where each of $H, G_1, \ldots, G_n$ has a permis.*

Proof (Sketch). The proof of 1 is by computer search. For 2, the permis goes through the vertices "from lowest to highest" according to $\sqsubseteq$. For 3, G has a maximal independent set M of simplicial vertices, and the permis visits M last. For 4, we can reduce ourselves to the case where only one vertex b is blown up into a graph G_b. Then the permis for G is obtained by taking the permis for H and replacing the update of b by a permis for G_b. Everything works as though the other vertices see $\bigvee_{v \in G_b} x_v$. The full proof is given in Appendix C.

We now exhibit classes of graphs without a permis. As mentioned in Theorem 4, the smallest graph without a permis is the heptagon.

Proposition 5. *For all $2k+1 \geq 7$, the odd hole C_{2k+1} does not have a permis.*

Proof. Let w be a permutation, and orient the edges such that $a \to b$ if and only if $a = w_i, b = w_j$ with $j > i$. We shall prove that there cannot be two consecutive arcs in the same direction; this shows that the direction of arcs must alternate, which is impossible because there is an odd number of arcs in the cycle. We do this by a case analysis on the arcs preceding those two consecutive arcs.

We consider six vertices f, e, d, c, b, a, where the last two arcs $c \to b \to a$ are in the same direction. The first case is where $d \to c$. In that case, if $(x_a, x_b, x_c) = (1, 1, 1)$, then $(y_b, y_c, y_d) = (0, 0, 0)$ as shown below (top left) along with the other three cases.

	x			1		1		1		x	0			0			1		1
		d	$\longrightarrow$	c	$\longrightarrow$	b	$\longrightarrow$	a			f	$\longleftarrow$	e	$\longrightarrow$	d	$\longleftarrow$	c	$\longrightarrow$	b	$\longrightarrow$	a
	y	0		0		0				y			1		0		0		0		
x	1		1				1		1	x			1		0			1		1	
	e	$\longleftarrow$	d	$\longleftarrow$	c	$\longrightarrow$	b	$\longrightarrow$	a			f	$\longrightarrow$	e	$\longrightarrow$	d	$\longleftarrow$	c	$\longrightarrow$	b	$\longrightarrow$	a
y			0		0		0			y	0		1		0		0		0		

□

Say a set of vertices S is **tethered** if there is an edge st between any $s \in S$ and any $t \in T = N(S) \setminus S$.

Proposition 6. *Let G be a graph. If G has a tethered set of vertices S such that $G[S]$ has no permis, then G has no permis.*

Proof (Sketch). Let x be a configuration such that $x_S \neq 0$ and $x_T = 0$. Then updating T will have no effect, and we have $y_T^a = 0$ throughout (for every $0 \leq a \leq l$). Therefore, the updates in S are the same as the updates in $G[S]$: $\mathsf{K}^w(x;G)_S = \mathsf{K}^{\hat{w}}(x_S;G)$, where $\hat{w}$ represents the updates of S only. Since $\hat{w}$ does not fix $G[S]$, w does not fix G. The full proof is Appendix D.

Propositions 5 and 6 yield perhaps the second simplest class of graphs without a permis. The **wheel graph** is $W_{n+1} = K_2(C_n, K_1)$.

Corollary 2. *For all $2k+2 \geq 8$, the wheel graph W_{2k+2} does not have a permis.*

An interesting consequence of Proposition 6 is that having a permis is not a graph property that can be tested by focusing on an induced subgraph, even if the latter has all but seven vertices. Indeed, for any graph H, the graph $G = K_2(C_7, H)$ does not have permis, since the heptagon is tethered in G. Conversely, for any graph H without a permis, adding a pending vertex v' to each vertex v of H yields a graph G where the set of simplicial vertices is a dominating set (all the vertices v' form a maximal independent set of simplicial vertices). Therefore, some graphs with an induced heptagon do have a permis.

In general, the characterisation of graphs with a permis remains open.

Question 2. What is the complexity of the following decision problem: given G, does G have a permis?

References

1. Julio Aracena, Maximilien Gadouleau, Adrien Richard, and Lilian Salinas. Fixing monotone boolean networks asynchronously. *Information and Computation*, 274(104540), October 2020.
2. Julio Aracena, Adrien Richard, and Lilian Salinas. Number of fixed points and disjoint cycles in monotone boolean networks. *accepted in SIAM journal on Discrete mathematics*, 2017.
3. J.A. Bondy and U.S.R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, 2008.
4. Maximilien Gadouleau and Adrien Richard. On fixable families of boolean networks. In *Proc. Workshop on Asynchronous Cellular Automata*, pages 396–405, September 2018.
5. E. Goles and M. Noual. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 48(5):646–662, 2012.
6. Eric Goles. Dynamics of positive automata networks. *Theoretical Computer Science*, 41:19–32, 1985.
7. Mathilde Noual and Sylvain Sené. Synchronism versus asynchronism in monotonic boolean automata networks. *Natural Computing*, Jan 2017.
8. Landon Rabern, Brian Rabern, and Matthew Macauley. Dangerous reference graphs and semantic paradoxes. *Journal of Philosophical Logic*, 42(5):727–765, 2013.
9. F. Robert. Iterations sur des ensembles finis et automates cellulaires contractants. *Linear Algebra and its Applications*, 29:393–412, 1980.
10. Steven Yablo. Paradox without self-reference. *Analysis*, 53(4):251–252, 1993.

A Proof of Theorem 2

Proof. We prove that the DOMINION problem is NP-complete. It is in NP: the certificate is the pair (v, I) where $W \cap N(v) \subseteq N(I)$.

We show NP-hardness by first reducing SET COVER to COLONY and then reducing COLONY to DOMINION.

Theorem 5. COLONY *is NP-complete.*

Proof. The proof is by reduction from SET COVER, which is NP-complete. In SET COVER, the input is a finite set of elements $X = \{x_1, \ldots, x_n\}$, a collection $C = \{C_1, C_2, \ldots, C_m\}$ of subsets of X, and an integer k. The question is whether there exists a subset $S \subseteq C$ of cardinality at most k such that $\cup_{C_i \in S} C_i = X$.

We first construct the graph G on $n + mk$ vertices. G consists of: vertices $Q_j = \{q_j^1, \ldots, q_j^k\}$, for each $j \in [m]$; vertices v_i for each $i \in [n]$; edges from each vertex in Q_j to v_i, whenever $x_i \in C_j$; edges connecting $\{q_1^l, q_2^l, \ldots, q_m^l\}$ in a clique, for each $l \in [k]$. Let the target set $T = \{v_1, \ldots, v_n\}$. This concludes our construction; an illustrative example is shown in Fig. 1.

We now show that if (X, C, k) is a yes-instance of SET COVER, then (G, T) is a yes-instance of COLONY. Let $S \subseteq C$ be a set cover of X of cardinality at most k. We obtain the set I as follows:

$$I = \{q_j^a : C_j \text{ is the } a\text{th element of } S\}.$$

Note that every node in I exists in G since S has cardinality at most k (the last subset to appear in S is its kth element exactly). Further, I is an independent set, since by construction every node q_j^a is adjacent to some other node q_l^b if and only if $a = b$. Lastly, every node $v_i \in S$ is incident to some node in I; for any i, $\exists j : v_i \in C_j$. Then necessarily $\exists a : q_j^a \in I$, and by construction (v_i, q_j^a) is an edge in G.

Conversely, if (G, T) is a yes-instance of COLONY then (X, C, k) is a yes-instance of SET COVER. Let I be an independent set in G which colonizes T. By construction of G, I has cardinality at most k. Suppose otherwise, for contradiction - then by the pigeon-hole principle there is some clique C_j such that $|C_j \cap I| \geq 2$, contradicting that I is an independent set. We obtain the set S of cardinality $|I|$ as follows:

$$S = \{C_j : \exists a \text{ such that } q_j^a \in I\}.$$

We now show S is a set cover of X. For each $i \in [n]$, v_i must be adjacent to some node in I; denote this node q_j^a - now by construction x_i is in the set C_j, and $C_j \in S$.

□

Theorem 6. DOMINION *is NP-complete.*

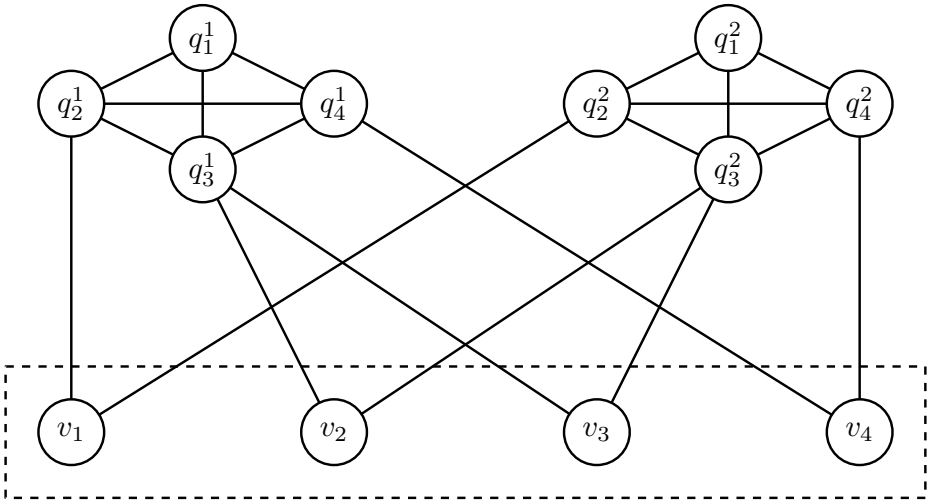


Fig. 1. Illustration of the reduction from Set Cover to Colony (the set T is the nodes in the dashed box). Here the Set Cover instance has $C_1 = \emptyset, C_2 = \{x_1\}, C_3 = \{x_2, x_3\}, C_4 = \{x_4\}$, with $k = 2$. Observe that both the Set Cover instance and the Colony instance are no-instances.

Proof. The proof is by reduction from Colony, which is NP-complete, as proved in Theorem 5. Let (G, S) be an instance of Colony, and construct the instance $(\hat{G}, \hat{S})$ as follows.

Let $G = (V, E)$ and denote $T = V \setminus S$. Then consider a copy $T' = \{t' : t \in T\}$ of T and an additional vertex $\hat{v} \notin V \cup T'$. Let $\hat{G} = (\hat{V}, \hat{E})$ with $\hat{V} = V \cup T' \cup \{\hat{v}\}$ and $\hat{E} = E \cup \{tt' : t \in T\} \cup \{s\hat{v} : s \in S\}$, and $\hat{S} = S \cup T'$. This construction is illustrated in Fig. 2.

We only need to prove that S is a colony of G if and only if $\hat{S}$ is a colony of $\hat{G}$. Firstly, if S is a colony of G, then there exists an independent set I of G such that $S \subseteq N(I; G)$. Then $\hat{S} \cap N(\hat{v}; \hat{G}) = S$ is contained in $N(I; \hat{G} - \hat{v})$, thus $\hat{S}$ is a dominion of $\hat{G}$.

Conversely, if $\hat{S}$ is a dominion of $\hat{G}$, then there exists $u \in \hat{V} \setminus \hat{S}$ such that $\hat{S} \cap N(u; \hat{G})$ is a colony of $\hat{G} - u$. Then either $u = \hat{v}$ or $u \in T$. Suppose $u = t \in T$, then $t' \in \hat{S}$ is an isolated vertex of $G - t$, hence $\hat{S} \cap N(t; \hat{G})$ is not a colony of $\hat{G} - t$. Therefore, $u = \hat{v}$ and there exists an independent set $\hat{I}$ of $\hat{G} - \hat{v}$ such that $\hat{S} \cap N(\hat{v}; \hat{G}) = S$ is contained in $N(\hat{I}; \hat{G})$. Since $S \subseteq V$ and $N(S; \hat{G} - \hat{v}) \subseteq V$, we obtain $S \subseteq N(\hat{I} \cap V; \hat{G} - \hat{v}) \cap V = N(\hat{I} \cap V; G)$, where $I = \hat{I} \cap V$ is an independent set of G. Thus, S is a colony of G. □

□

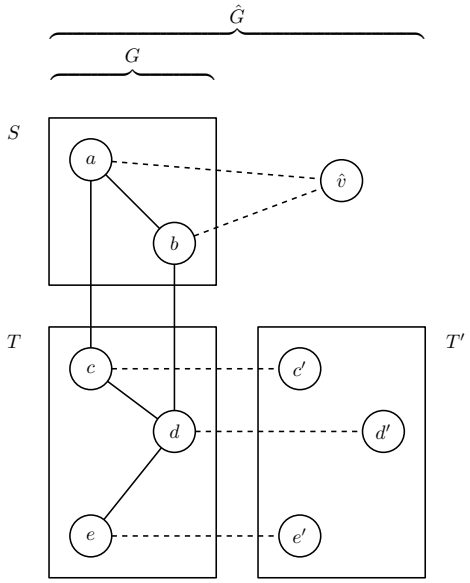


Fig. 2. Example reduction from a no-instance of Colony (G, S) to the corresponding no-instance of Dominion $(\hat{G}, \hat{S})$, with $\hat{S} := S \cup T'$.

B Proof of Theorem 3

Proof. Membership of coNP is known: the no-certificate is a permutation of S which does not fix the kernel network (by Proposition 4). The hardness proof is by reduction from Non-Dominion, which is coNP-complete, as proved in Theorem 6. Let (G, S) be an instance of Non-Dominion, and construct the instance $(\hat{G}, \hat{S})$ as follows.

Let $G = (V, E)$ and $T = V \setminus S$. For any $t \in T$, let $G_t = (V_t \cup \{\hat{t}\}, E_t)$ be the graph defined as follows: $V_t = \{u_t : u \in V \setminus t\}$ is a copy of all the vertices apart from t, which is replaced by a new vertex $\hat{t} \notin V_t$, and $E_t = \{a_t b_t : ab \in E, a, b \neq t\} \cup \{s_t\hat{t} : st \in E, s \in S\}$ is obtained by removing the edges between t and the rest of T. Then G is the disjoint union of all those graphs, i.e. $G = \bigcup_{t \in T} G_t$, while $\hat{S} = \bigcup_{t \in T} V_t$. For the sake of simplicity, we shall use the notation $A_t = \{u_t : u \in A\}$ for all $A \subseteq V \setminus \{t\}$.

By construction, $\hat{G} - \hat{S}$ is the empty graph on $\{\hat{t} : t \in T\}$, hence $\hat{S}$ is a vertex cover of $\hat{G}$. All we need to show is that $\hat{S}$ is a non-dominion of $\hat{G}$ if and only if S is a non-dominion of G. We have that $\hat{S}$ is a dominion of $\hat{G}$ if and only if there exists $\hat{t}$ and an independent set $\hat{I}$ of $\hat{G} - \hat{t}$ such that $W = \hat{S} \cap N(\hat{t}; \hat{G}) = (S \cap N(t; G))_t$ is contained in $N(\hat{I}; \hat{G})$. We have $\hat{I} \cap V_t = I_t$ for some independent set I of G. Since $W \subseteq V_t$ and $N(W; \hat{G} - \hat{t}) \subseteq V_t$, we have $W \subseteq N(\hat{I} \cap V_t; \hat{G}) \cap V_t = N(I; G)_t$, which is equivalent to S being a dominion of G.

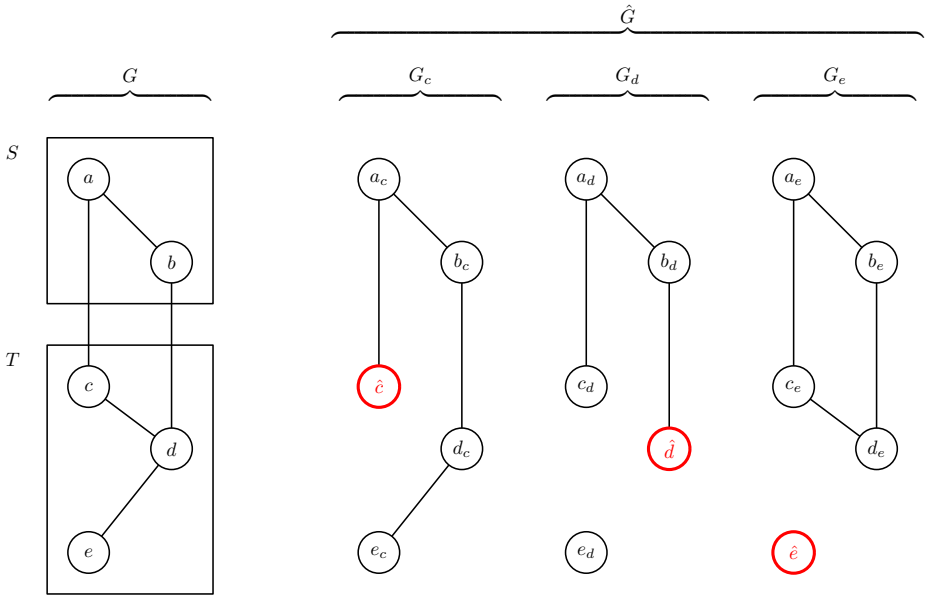


Fig. 3. Example reduction from a no-instance of Non-Dominion (G, S) to the corresponding no-instance of Fixing Set $(\hat{G}, \hat{S})$, with $\hat{S} = V_c \cup V_d \cup V_e$.

C Proof of Theorem 4

Lemma 1. *Let G be a graph such that its set S of simplicial vertices is a dominating set. Then S contains a maximal independent set.*

Proof. Suppose for the sake of contradiction that S does not contain a maximal independent set, i.e. that no dominating set contained in S is independent. Let $T \subseteq S$ be a minimal dominating set, and let t, t' be adjacent vertices of T so that $N[t] = N[t']$. Thus $T \setminus \{t'\}$ is still a dominating set, which is the desired contradiction. □

Proof (of Theorem 4).

1. Proof by computer search. Code available at `https://github.com/dave-ck/MISMax` with some further explanation.
2. Let G be a comparability graph, and order its vertices so that $v_i \sqsubseteq v_j \implies i \leq j$. Then let $w = w_1 \dots w_n$ with $w_i = v_i$ for all $i \in [n]$. For the sake of contradiction, suppose that $y_{N[w_i]} = 0$ for some $i \in [n]$. Since $\mathsf{K}(y^{i-1})_{w_i} = y_{w_i} = 0$, we have $y^{i-1}_{N[w_i]} \neq 0$. Therefore, let $j = \max\{k \in [n] : w_k \sim w_i, y^{i-1}_{w_j} = 1\}$; since $y^{i-1}_{w_k} = y_{w_k} = 0$ for all $k \leq i-1$, we have $j \geq i+1$. But $\mathsf{K}(y^{j-1})_{w_j} = y_{w_j} = 0$, thus there exists $l = \max\{k \in [n] : w_k \sim w_j, y^{j-1}_{w_j} = 1\}$. Again, $l \geq j+1$, and hence $y^{i-1}_{w_l} = 1$. However, $w_l \sim w_j$ and $w_j \sim w_i$ imply that $w_l \sim w_i$, thus $l \in \{k \in [n] : w_k \sim w_i, y^{i-1}_{w_j} = 1\}$ and $l \leq j$, which is the desired contradiction.

3. By Lemma 1, let M be a maximal independent set of simplicial vertices, and let w be a permutation of G such that the vertices of M appear last: $w_1, \ldots, w_{n-|M|} \notin M$ and $w_{n-|M|+1}, \ldots, w_n \in M$. Suppose for the sake of contradiction that $y_{N[v]} = 0$ for some vertex v. Then there exists $m \in M$ such that $N[m] \subseteq N[v]$, thus $y_{N[m]} = 0$. Suppose $m = w_{a+1}$, then $y^a_{N(w_{a+1})} = y_{N(m)} = 0$, hence $y_m = \mathsf{K}(y^a)_{w_{a+1}} = 1$, which is the desired contradiction.
4. It is easily shown that a graph composition can be obtained by repeatedly blowing up one vertex b into the graph G_b at a time. As such, we only need to prove the case where $G = H(K_1, \ldots, K_1, G_b, K_1, \ldots, K_1)$, where the vertices are sorted according to a permis $\hat{w} = \hat{w}_1 \ldots \hat{w}_b \ldots \hat{w}_n$ of H. For any configuration x of G, let $\hat{x}$ be the configuration of H such that $\hat{x}_u = x_u$ for all $u \neq \hat{w}_b$ and $\hat{x}_{\hat{w}_b} = \bigvee_{v \in G_b} x_v$. Let w^b be a permis of G_b and consider the permutation w of G given by $w = \hat{w}_1 \ldots \hat{w}_{b-1} w^b \hat{w}_{b+1} \ldots \hat{w}_n$. We then prove that $y \in \mathrm{K}(G)$ by considering the three main steps of w. We denote the vertex set of G_b as V_b.
   - Step 1: before the update of G_b. It is easy to show that for any $1 \leq a < b$, we have $\mathsf{K}^{w_1 \ldots w_a}(x; G)_{G - V_b} = \mathsf{K}^{\hat{w}_1 \ldots \hat{w}_a}(\hat{x}; H)_{H - \hat{w}_b}$.
   - Step 2: update of G_b. Note that V_b is a tethered set of G, so let $T = N(V_b; G) \setminus V_b$. Let $\alpha = y^{b-1}$ be the initial configuration and $\beta = y^{b-1+|V_b|}$ be the final configuration of the update of G_b. If $\alpha_T \neq 0$, then the whole of G_b will be updated to 0: $\beta_{V_b} = 0$. Otherwise, it is as if G_b is isolated from the rest of the graph and $\beta_{V_b} = \mathsf{K}^{w^b}(\alpha_{V_b}; G_b)$. In either case, we have $\hat{\beta} = \mathsf{K}^{(\hat{w}_b)}(\hat{\alpha}; H)$.
   - Step 3: after the update of G_b. Again, we have for all $b < a \leq n$, $\mathsf{K}^{w_{b+1} \ldots w_a}(\beta; G)_{G - V_b} = \mathsf{K}^{\hat{w}_{b+1} \ldots \hat{w}_a}(\hat{\beta}; H)_{H - \hat{w}_b}$.

   In conclusion, we have $y_{G - V_b} = \mathsf{K}^{\hat{w}}(\hat{x}; H)_{H - \hat{w}_b}$, and if $\mathsf{K}^{\hat{w}}(\hat{x})_{\hat{w}_b} = 0$ then $y_{V_b} = 0$ else $y_{V_b} = \mathsf{K}^{w^b}(x_S; G_b)$. In either case, we obtain that $y \in \mathrm{K}(G)$.

□

D Proof of Proposition 6

Proof. Let w be a permutation of G and $\hat{w}$ be the subsequence of w satisfying $[\hat{w}] = S$. Let $\hat{x}$ be a configuration of $G[S]$ which is not fixed by $\hat{w}$: $\mathsf{K}^{\hat{w}}(\hat{x}; G[S]) \notin \mathrm{K}(G[S])$. We first note that $\hat{x} \neq 0$ and that for all $0 \leq a \leq |\hat{w}|$, $\mathsf{K}^{\hat{w}_1 \ldots \hat{w}_a}(\hat{x}; G[S]) \neq 0$.

Let $T = N(S) \setminus S$ and $U = V \setminus (S \cup T)$ and $x = (x_S = \hat{x}, x_T = 0, x_U)$. We prove by induction on $0 \leq b \leq |w|$ that

$$y^b := \mathsf{K}^{w_1 \ldots w_b}(x; G) = \left(y^b_S = \mathsf{K}^{\hat{w}_1 \ldots \hat{w}_{b'}}(\hat{x}; G[S]), y^b_T = 0, y^b_U\right),$$

where b' is defined by $[\hat{w}_1 \ldots \hat{w}_{b'}] = S \cap [w_1 \ldots w_b]$. The base case $b = 0$ is clear. Suppose it holds for $b - 1$.

- Case 1: $w_b \in S$. Then $b' = (b-1)' + 1$ and $w_b = \hat{w}_{b'}$. Since $y^{b-1}_T = 0$, we have

$$y^b_{w_b} = \mathsf{K}(y^{b-1}; G)_{w_b} = \mathsf{K}(y^{b-1}_S; G[S])_{w_b} = \mathsf{K}(\mathsf{K}^{\hat{w}_1 \ldots \hat{w}_{b'-1}}(\hat{x}; G[S]); G[S])_{\hat{w}_{b'}} = \mathsf{K}^{\hat{w}_1 \ldots \hat{w}_{b'}}(\hat{x}; G[S])_{w_b},$$

and hence $y^b_S = \mathsf{K}^{\hat{w}_1 \cdots \hat{w}_{b'}}(\hat{x}; G[S])$.
- Case 2: $w_b \in T$. Then $b' = (b-1)'$. Since $y^{b-1}_S \neq 0$, we have $\mathsf{K}(y^{b-1}; G)_{w_b} = 0$ and hence $y^b_T = 0$.
- Case 3: $w_b \in U$. This case is trivial.

For $b = |w|$ we obtain $y = \mathsf{K}^w(x; G) = (\mathsf{K}^{\hat{w}}(\hat{x}; G[S]), 0, y_U)$, for which $y_S \notin \mathrm{K}(G[S])$, and hence $y \notin \mathrm{K}(G)$. □

On the complexity of freezing automata networks of bounded pathwidth*

Eric Goles[1], Pedro Montealegre[1], Martín Ríos-Wilson[1], and G. Theyssier[2]

[1] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago
[2] I2M (Aix-Marseille University, CNRS)

Abstract. An automata network is a graph of entities, each holding a state from a finite set and evolving according to a local update rule which depends only on its neighbors in the network's graph. It is freezing if there is an order on the states such that the state evolution of any node is non-decreasing in any orbit. They are commonly used to model epidemic propagation, diffusion phenomena like bootstrap percolation or cristal growth.
Previous works have established that, under the hypothesis that the network graph is of bounded treewidth, many problems that can be captured by trace specifications at individual nodes admit efficient algorithms. In this paper we study the even more restricted case of a network of bounded pathwidth and show two hardness results that somehow illustrate the complexity of freezing dynamics under such a strong graph constraint. First, we show that the trace specification checking problem is NL-complete. Second, we show that deciding first order properties of the orbits augmented with a reachability predicate is NP-hard.

1 Introduction

Automata networks (AN) are finite dynamical systems that can be seen as the finite and non-uniform counterpart of cellular automata on arbitrary graphs. An automata network is *freezing* if there is an order on the states such that the state evolution of any node is non-decreasing in any orbit. Several models that received a lot of attention in the literature are actually freezing automata networks, for instance: bootstrap percolation which has been studied on various graphs [1, 4, 3, 11], epidemic [6] or forest fire [2] propagation models, cristal growth models [17, 10] and various models of self-assembly tilings [18].

The freezing condition has strong implications on the computational complexity of these systems. For instance, following previous works on cellular automata [14, 8], it was established in [9] that a large set of problems specified by traces at individual nodes are actually NC when considering freezing automata networks of bounded treewidth. This result in particular captures the problem

* Research partially supported by projects STIUC-AMSUD 22-STIC-02 (all authors), Fondecyt-ANID 1200006 (EG), FONDECYT-ANID 1230599 (PM), ANID FONDECYT Postdoctorado 3220205 (MRW)

of nilpotency, a property which can be expressed in the language of orbits by: all orbits converge to the same fixed point. The nilpotency problem is typical of the computational complexity collapse when the freezing condition is combined by a condition on the structure of the network.

This paper aims at better understanding this complexity collapse by giving lower bounds for freezing automata networks on the simplest network structure: graphs of *bounded pathwidth* (intuitively, that are structurally close to a line or a cycle).

First, we consider regular trace properties (*i.e.* regular expressions specifying allowed traces at each node) and show that the problem of existence of an orbit following the constraints is NL-complete (Theorem 2). Note that this problem is similar to some well-studied problems in 1D cellular automata like cylinder-to-cylinder reachability which can also be expressed as a regular expression of traces. It is striking to compare the finite context with the NL upper bound above to the infinite context, where freezing cellular automata have actually an undecidable cylinder-to-cylinder reachability problem [14].

Second, we study another family of problems : properties defined by first order logic on configuration with equality, a predicate $x \to y$ meaning that y can be reached from x in one step, and a predicate $x \to^+ y$ meaning that configuration y can be reached from configuration x in some number of steps. This logic denoted FO$^+$also captures nilpotency by $\exists! y, \forall x : x \to^+ y$. Our second main result is that, although nilpotency is co-NL (Corollary 1), the model checking of FO$^+$is NP-hard even for freezing automata networks defined on a line (Theorem 3).

Our results contribute to the following global picture where each cell of the table is divided between the general case (lower left in black) and the freezing case (upper right in blue).

	Infinite 1D CA	Finite bounded pathwidth AN
Nilpotency	Decidable [14, Theorem 2] / Undecidable [12]	co-NL (Corollary 1) / PSPACE-complete [7, Corollary 3.2]
Regular trace properties	Undecidable [14, Theorem 5] / Undecidable	NL-complete (Theorem 2) / PSPACE-complete [7, Theorem 3.3]
FO$^+$	Open / Undecidable [12]	NP-hard (Theorem 3) / PSPACE-complete [7, Corollary 3.2]

2 Definitions and notations

Given a graph $G = (V, E)$ and a vertex v we will call $N(v)$ the neighborhood of v and δ_v to the degree of v. In addition, we define the closed neighborhood of v

as the set $N[v] = N(v) \cup \{v\}$ and we use the following notation $\Delta(G) = \max_{v \in V} \delta_v$ for the maximum degree of G. We will use the letter n to denote the order of G, i.e. $n = |V|$. Also, if G is a graph and the set of vertices and edges is not specified we use the notation $V(G)$ and $E(G)$ for the set of vertices and the set of edges of G respectively. In addition, we will assume that if $G = (V, E)$ is a graph then, there exists an ordering of the vertices in V from 1 to n. During the rest of the text, every graph G will be assumed to be connected and undirected. We define a *class* or a *family* of graphs as a set $\mathcal{G} = \{G_n\}_{n \geq 1}$ such that $G_n = (V_n, E_n)$ is a graph and $|V_n| = n$.

Non-deterministic freezing automata networks. Let Q be a finite set that we will call an *alphabet*. We define a non-deterministic automata network in the alphabet Q as a tuple $(G = (V, E), \mathcal{F} = \{F_v : Q^{N(v)} \to \mathcal{P}(Q) | v \in V\}))$ where $\mathcal{P}(Q)$ is the power set of Q. To every non-deterministic automata network we can associate a non-deterministic dynamics given by the global function $F : Q^n \to \mathcal{P}(Q^n)$ defined by $F(x) = \{x \in Q^n | x_v \in F_v(x), \forall v\}$.

Definition 1. *Given a a non-deterministic automata network $(G, \mathcal{F})$ we define an orbit of a configuration $x \in Q^n$ at time t as a sequence $(x_s)_{0 \leq s \leq t}$ such that $x_0 = x$ and $x_s \in F(x_{s-1})$. In addition, we call the set of all possible orbits at time t for a configuration x as $\mathcal{O}(x, t)$. Finally, we also define the set of all possible orbits at time t as $\mathcal{O}(\mathcal{A}, t) = \bigcup_{x \in Q^n} \mathcal{O}(x, t)$*

We say that a non-deterministic automata network $(G, \mathcal{F})$ defined in the alphabet Q satisfies the *freezing property* or simply that it is *freezing* if there exists a partial order $\leq$ in Q such that for every $t \in \mathbb{N}$ and for every orbit $y = (x_s)_{0 \leq s \leq t} \in \mathcal{O}(\mathcal{A}, t)$ we have that $x_s^i \leq x_{s+1}^i$ for every $0 \leq s \leq t$ and for every $0 \leq i \leq n$.

Path decompositions and pathwidth. Let $G = (V, E)$ be a connected graph. A subgraph P of G is said to be a path if $V(P) = \{v_1, \ldots, v_k\}$ where every v_i is different and $E(P) = \{v_1v_2, v_2v_3 \ldots, v_{k-1}v_k\}$. Now we present a graph parameter called *pathwidth* which, generally speaking, indicates how similar a graph is to a path graph. More precisely, we have the following definition:

Definition 2. *Given a graph $G = (V, E)$ a path decomposition is pair $\mathcal{D} = (P, \Lambda)$ such that P is a path graph and Λ is a family of subsets of nodes $\Lambda = \{X_t \subseteq V |\ t \in V(T) = \{1, \ldots, s\}\}$, called bags, such that:*

- *Every node in G is in some X_t, i.e: $\bigcup_{t \in V(P)} X_t = V$,*
- *For every $e = uv \in E$ there exists $t \in V(P)$ such that $u, v \in X_t$,*
- *For every $u, v, w \in V(P)$ if $1 \leq u < v < w \leq s$ then, $X_u \cap X_v \subseteq X_w$.*

We define the width of a path decompostion $\mathcal{D}$ as the amount $\text{width}(\mathcal{D}) = \max_{t \in V(P)} |X_t| - 1$. Given a graph $G = (V, E)$, we define its pathwidth as the parameter $\text{path}(G) = \min_{\mathcal{D}} \text{width}(\mathcal{D})$. In other words, the pathwidth is the minimum width of a path decomposition of G. Note that, if G is a connected graph such that $|E(G)| \geq 2$ then, G is a path if and only if $\text{path}(G) = 1$.

It is known that a path decomposition can be computed in **DLOGSPACE** [13].

Specification checking problem. Now, we introduce a decision problem called *specification checking problem.* Roughly, this problem ask for the existence of an orbit in the automata network that verifies some trace constraints at each node. The information of allowed traces at each node is called a *specification*: a specification of length t is a map $\mathcal{E}_t : V \to \mathcal{P}(Q^t)$ such that, for every $v \in V$, the sequences in $\mathcal{E}_t(v)$ are non-decreasing (and thus respect the freezing condition). We say that $\mathcal{E}_t$ is satisfiable by $\mathcal{A}$ if there exists an orbit $O \in \mathcal{O}(\mathcal{A}, t)$ such that $O_v \in \mathcal{E}_t(v)$ for every $v \in V$. We observe that the number of freezing traces of length t is polynomial in t so $\mathcal{E}_t$ can be represented in polynomial size in V and t.

Also, in the absence of explicit mention, all the considered graphs will have bounded degree Δ by default, so a freezing automata network rule can be represented as the list of local update rules for each node which are maps of the form $Q^\Delta \to \mathcal{P}(Q)$ whose representation as transition table is of size $O\big(|Q|^{\Delta+1}\big)$. The specification checking problem (SPEC) introduced in [9] asks whether a given freezing automata network satisfies a given specification. If $\mathcal{E}_t$ is a satisfiable t-specification for some automata network $\mathcal{A}$ we write $\mathcal{A} \models \mathcal{E}_t$.

In [9] it is shown that many well-known and well-studied decision problems related to the dynamics of automata networks are somehow related to SPEC. These problems are: the prediction problem, the predecessor problem, the nilpotency problem and the asynchronous reachability problem. Recall that nilpotency is the property that there is a configuration x such that all orbits end up in x and x is a fixed point. Most of these problems are sub-problems of SPEC. In the case of Nilpotency, an efficient parallel Turing reduction can be constructed [9].

In this paper, we focus on a variant of the specification problem were admissible traces are represented as regular expressions. More precisely, a regular (Q, V)-specification is a map from V to regular expressions over alphabet Q. We therefore consider the Regular Specification Checking Problem or simply REGSPEC which is the same as SPEC except that the specification must be a regular specification. It is interesting to observe that REGSPEC with fixed degree and fixed treewidth and with alphabet as unique parameter is $W[2]$-hard [9].

3 NL-completeness of REGSPEC problem

In this section, we explore different results for the complexity of REGSPEC when the pathwidth of the underlying interaction graph is bounded. We start this section by showing that REGSPEC is in **NL**. This is a direct extension of the results on bounded treewidth in [9] and the technique used in [14] for the prediction problem in one dimensional freezing cellular automata. Then, we show that the problem is actually **NL**-complete by showing a logspace reduction from (s, t)-connectivity.

Theorem 1. *The REGSPEC problem is in **NL** for bounded pathwidth freezing AN.*

The complement of the nilpotency problem can be reduced to instances of REGSPEC in such a way that we keep the strong complexity upper-bounds from the previous theorem.

Corollary 1. *The nilpotency problem is in* ***co-NL*** *for bounded pathwidth freezing AN.*

Now we show show that REGSPEC is **NL**-complete and thus, it is most likely that the previous algorithm is the best we can do, unless **NL** = **DLOGSPACE**.

Now we introduce the main result of the section.

Theorem 2. *The Regular Specification Checking problem (REGSPEC) is* ***NL****-complete when restricted to bounded degree and bounded pathwidth interaction graphs.*

The proof of this result is divided in what we call *phases.* The main idea of the proof is to construct a non-deterministic automata network $\mathcal{A}_D = (G_D, \mathcal{F}_D)$ defined over a two dimensional grid of size $k \times d$ where $d = n^{\mathcal{O}(1)}$ and $k = \mathcal{O}(1)$. Of course, since k is constant, then $\mathcal{A}_D$ has bounded pathwidth. This automata network will non-deterministically guess a sequences of blocks (a structure representing edges in the interaction graph of $\mathcal{A}$, see Figure 1 for more details). We call this part the *selection phase.* Then, the next part of the proof consists in showing that $\mathcal{A}_D = (G_D, \mathcal{F}_D)$ is capable of deterministically verifying if an initial condition corresponds to a sequence of valid edges, i.e. if it corresponds to a sequence of blocks and they actually represent edges in G_D. We call this phase a *verification phase* In order to perform this task, we use a construction based on using signals that will collide at specific locations as a way to verify the distance between two given cells. In addition, it would be essential to save (as a constant layer) the information contained in the incidence matrix of D. Generally speaking, once $\mathcal{A}_D$ has verified that the sequence of blocks is valid, it will compare two subsequent blocks (which represent a pair of edges) in order to verify if they are incident. If in any part of its dynamics $\mathcal{A}_D$ locally detects some error (by the application of its local rule), it will spread an error state that will led the system to an attractor corresponding to a uniform configuration in which any cell will be in this particular error state. However, if the process runs flawless, then the system will reach an attractor in which all the cells are in a particular success state. We will code, by using a specification $\mathcal{E}_D$ (given in the input of REGSPEC), a specific requirement for the initial configuration (more precisely, we will ask the initial configuration to have the incidence matrix of D, markers and information about the nodes (s, t)) in order to allow $\mathcal{A}_D$ to have enough information to start the selection and verification process. In addition, we will code in this specification only the orbits that will reach this specific success state. By doing this, we will show that $\mathcal{A}_D \models \mathcal{E}_D$ if and only if there is a path between s, t in D. Thus, the reduction will consist on constructing $(\mathcal{A}_D, \mathcal{E}_D)$ from (D, s, t) in **DLOGSPACE**.

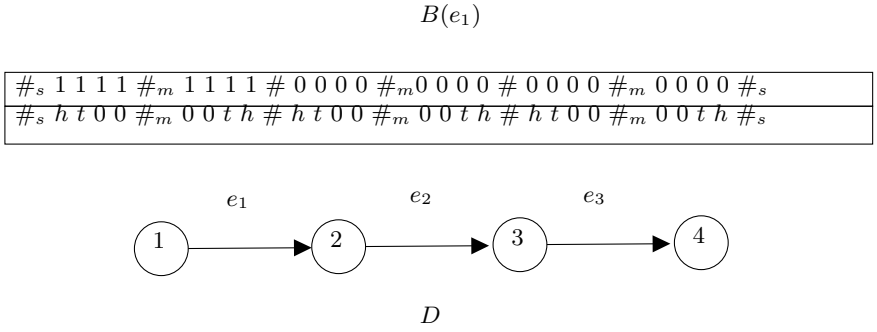


Fig. 1. An example of a block for some graph D.

4 Hardness of FO^+ model checking

$FO(=, \to)$ denotes the first order logic over configurations using equality and a predicate $x \to y$ meaning that y can be reached from x in one step. It is well-known that this logic can be efficiently dealt with using finite automata theory when configurations are one-dimensional. For instance the model-checking of this logic is decidable on one-dimensional CAs [5, 16]. In this subsection, we study first-order properties of the dynamics enriched with a new predicate $x \to^+ y$ expressing that configuration y can be reached from configuration x in some unknown number of steps. We denote this logic $FO^+ = FO(=, \to, \to^+)$. Adding the predicate $\to^+$ allows to express properties like nilpotency:

$$\exists x, \forall z, (z \to^+ z) \implies z = x,$$

which is equivalent in the deterministic case to $\exists x, \forall y, y \to^+ x$. The model checking of FO^+ is therefore undecidable for general 1D CA [12] and **PSPACE-complete** for AN of bounded pathwidth [7]. However, nilpotency is a decidable property for 1D freezing CA [14] and **co-NL** for bounded pathwidth freezing AN (Corollary 1). It is therefore interesting to figure out what is the complexity of the model checking of FO^+ for freezing AN of bounded pathwidth.

The goal of this section is to show that despite considering only "one-dimensional" networks and having the freezing constraint, we can encode bi-dimensional domino problems in FO^+ and thus get a **NP-hard** lower bound. The precise NP-hard problem we consider in this subsection to reduce from is the following.

Lemma 1 (HV-domino CSP). *Let Q be a large enough alphabet. The following problem is NP-complete:*

- ***input:*** *for each $1 \leq i, j \leq n$, two lists of constraints: $H_{i,j} \subseteq Q^2$ and $V_{i,j} \subseteq Q^2$.*
- ***question:*** *does there exist a configuration $a \in Q^{\{1,\dots,n\}\times\{1,\dots,n\}}$ such that for all $1 \leq i, j \leq n$ the local constraints are satisfied, i.e.*

$$(a_{i,j}, a_{i+1,j}) \in H_{i,j} \text{ (if } i < n) \text{ and } (a_{i,j}, a_{i,j+1}) \in V_{i,j} \text{ (if } j < n).$$

We can now show a lower bound on the model checking of a single formula of FO^+. Let $P_x(x)$ denote the formula of FO^+ expressing that configuration x has at least k preimages, formally:

$$P_k(x) \equiv \exists x_1 \neq x_2 \neq \cdots \neq x_k, \bigwedge_{1 \leq i \leq k} x_i \to x.$$

We will consider the following formula ϕ:

$$\phi \equiv \exists x : x \to x$$
$$\wedge \big(\forall y, \forall y^1, \forall z, (\neg P_1(y) \wedge \neg P_2(y^1) \wedge y \to y^1 \wedge y^1 \to^+ z \wedge z \to x \wedge z \neq x) \Rightarrow \neg P_2(z)\big).$$

It expresses that there exists a fixed point x such that considering any orbit starting from a configuration y without preimage, which is the unique preimage of its successor y^1, and leading to x, then the configuration z occurring in the orbit just before reaching x has only 1 preimage.

The main result of this section is that the FO^+ model checking problem is already hard for formula ϕ. The proof uses the HV-domino CSP of Lemma 1. For each HV-domino CSP problem, we build a deterministic automata one-dimensional network that essentially checks that a configuration $(a_{i,j})$ satisfies the HV-domino constraints. By one-dimensional we mean a graph which is a line with self-loops on each node. In this automata network, configurations $(a_{i,j})$ are layed out as one-dimensional configurations so that $a_{i,j}$ and $a_{i+1,j}$ are neighbors in the graph, and therefore H-constraints can be checked locally. However, $a_{i,j}$ and $a_{i,j+1}$ are far away in the graph, so V-constraints require the dynamics of the automata network to be checked. The key idea is to use formula ϕ above to characterize the part of the dynamics of the automata network that checks all V-constraints for a given candidate configuration $(a_{i,j})$: intuitively, quantifying over orbits starting from a configuration y without preimage and being the unique preimage of its successor ensures that the orbit contains some well-initialized computation, and predicate $\neg P_2$ on the configuration before reaching the fixed point codes the fact that the output of the computation is correct. The fixed point configuration x in formula ϕ represents a candidate configuration $(a_{i,j})$ (cleaned from any trace of computation) and, by construction of the automata network, the second part of the formula expresses that for any well-initialized test of a V-constraint the output of the test is a success. Formula ϕ uses predicate $\neg P_2$ to characterize some specific configurations: the key corresponding trick in the construction below is to make Cartesian products of some alphabet with $\{0, 1\}$ and ensure that the action of the automata network almost always reset to 1 the value of such a $\{0, 1\}$-component in at least one node. This ensures that the configuration obtained after one step has at least two preimages. The situations where it is not the case are exceptional and well-controlled: this helps to identify possible candidates for configurations y, y^1 and z in formula ϕ.

Theorem 3. *Checking whether a given deterministic freezing automata network $(G, \mathcal{F})$ verifies ϕ is NP-hard, even when restricted to bounded alphabet, and degree 3 and pathwidth 1.*

References

1. Hamed Amini and Nikolaos Fountoulakis. Bootstrap percolation in power-law random graphs. *Journal of Statistical Physics*, 155(1):72–92, feb 2014.
2. Per Bak, Kan Chen, and Chao Tang. A forest-fire model and some thoughts on turbulence. *Physics Letters A*, 147(5):297 – 300, 1990.
3. József Balogh and Béla Bollobás. Bootstrap percolation on the hypercube. *Probability Theory and Related Fields*, 134(4):624–648, jul 2005.
4. József Balogh, Béla Bollobás, Hugo Duminil-Copin, and Robert Morris. The sharp threshold for bootstrap percolation in all dimensions. *Transactions of the American Mathematical Society*, 364(5):2667–2701, may 2012.
5. Olivier Finkel. On decidability properties of one-dimensional cellular automata. *J. Cellular Automata*, 6(2-3):181–193, 2011.
6. M.A. Fuentes and M.N. Kuperman. Cellular automata and epidemiological models with spatial dependence. *Physica A: Statistical Mechanics and its Applications*, 267(3–4):471 – 486, 1999.
7. Guilhem Gamard, Pierre Guillon, Kevin Perrot, and Guillaume Theyssier. Rice-like theorems for automata networks. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
8. E. Goles, N. Ollinger, and G. Theyssier. Introducing freezing cellular automata. In *Exploratory Papers of Cellular Automata and Discrete Complex Systems (AUTOMATA 2015)*, pages 65–73, 2015.
9. Eric Goles, Pedro Montealegre, Martín Ríos Wilson, and Guillaume Theyssier. On the impact of treewidth in the computational complexity of freezing dynamics. In Liesbeth De Mol, Andreas Weiermann, Florin Manea, and David Fernández-Duque, editors, *Connecting with Computability*, pages 260–272, Cham, 2021. Springer International Publishing.
10. Janko Gravner and David Griffeath. Cellular automaton growth on z2: Theorems, examples, and problems. *Advances in Applied Mathematics*, 21(2):241 – 304, 1998.
11. Alexander E. Holroyd. Sharp metastability threshold for two-dimensional bootstrap percolation. *Probability Theory and Related Fields*, 125(2):195–224, 2003.
12. J. Kari. The Nilpotency Problem of One-dimensional Cellular Automata. *SIAM Journal on Computing*, 21:571–586, 1992.
13. Shiva Kintali and Sinziana Munteanu. Computing bounded path decompositions in logspace. In *Electron. Colloquium Comput. Complex.*, volume 19, page 126. Citeseer, 2012.
14. Nicolas Ollinger and Guillaume Theyssier. Freezing, Bounded-Change and Convergent Cellular Automata. *Discrete Mathematics & Theoretical Computer Science*, vol. 24, no. 1, January 2022.
15. Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Invent. Math.*, 12:177–209, 1971.
16. Klaus Sutner. Model checking one-dimensional cellular automata. *J. Cellular Automata*, 4(3):213–224, 2009.
17. S. M. Ulam. On some mathematical problems connected with patterns of growth of figures. In A. W. Bukrs, editor, *Essays on Cellular Automata*, pages 219–231. U. of Illinois Press, 1970.
18. Andrew Winslow. A brief tour of theoretical tile self-assembly. In *Cellular Automata and Discrete Complex Systems - 22nd IFIP WG 1.5 International Workshop, AUTOMATA 2016, Zurich, Switzerland, June 15-17, 2016, Proceedings*, pages 26–31, 2016.

A Proof of Theorem 1

Proof. Let $t \in \mathbb{N}$ a time, $\mathcal{A} = (G, \mathcal{F})$ a non-deterministic automata network and $\mathcal{E}_t$ a t-specification. First note that if G has bounded pathwidth, one can compute a path decomposition $\mathcal{P} = (X_1, \ldots, X_p)$ in **DLOGSPACE** where $p \leq \mathrm{pw}(G)$ [13]. Now note that we can adapt the previous **NC** algorithm to an **NL** algorithm in this particular context. First, observe that the dynamic programming lemma in [9] is also valid in this case, but now, because the decomposition is a path, there is only one bag for each level. Then, observe that testing whether a trace in compact representation (as explained earlier and presented in [9]) belongs to some regular language can be done in **DLOGSPACE**. Then, the algorithm will reproduce the same procedure than the algorithm in [9] , but, instead of parallelizing the information for the nodes in a bag storing it in different processors, it will handle this information non-deterministically. More precisely, an algorithm can guess a trace for each bag X_l from $l = 1$ to $l = p$ while ensuring that each node (that can appear in various bags) has the same trace in all guesses: this can be done because, by definition of a path decomposition, a node appears in an interval of $[1, p]$. This is the major difference with [9] that has to deal with tree decompositions. Thus, REGSPEC problem is in **NL**. □

B Proof of Corollary 1

Proof. For a freezing AN F over alphabet Q, the property of **not** being nilpotent is equivalent to the existence of a pair of orbits that ends up in two fixed points that differ at some node. For any pair of states q and q' and some node v, denote by $\mathsf{NONIL}(q, q', v)$ the problem of existence of two orbits in F that end respectively in states q and q' at node v. $\mathsf{NONIL}(q, q', v)$ is actually a REGSPEC problem for the AN $F \times F$ over alphabet $Q \times Q$ given by the following regular expression for trace at node v: $(Q \times Q)^*(q, q')^+$. Then, non-nilpotency can be expressed as the disjunction

$$\bigvee_{v \in V} \bigvee_{q \neq q'} \mathsf{NONIL}(q, q', v).$$

From this, we deduce a **NL** algorithm for non-nilpotency: choose non-deterministically one of the polynomially many instances of NONIL above and solve it in **NL** as an instance of the REGSPEC problem (Theorem 1). We deduce that nilpotency is **co-NL**. □

C Proof of Theorem 2

In order to show the main result, we need some technical definitions. Let Γ be a finite set and $\alpha \in \Gamma$. We call a string $y \in \Gamma^*$ an *α-marker* in $i \in [|y|]$ of length $l \in \mathbb{N}$ if $y|_{[i,i+l]} = \alpha \cdots \alpha$.

Given a directed graph (G, E) represented by its (oriented) incidence matrix M_G, we define for each $e = (u, v) \in E$ a block representing e as a $2 \times 2mn$ matrix $B(e)$ such that:

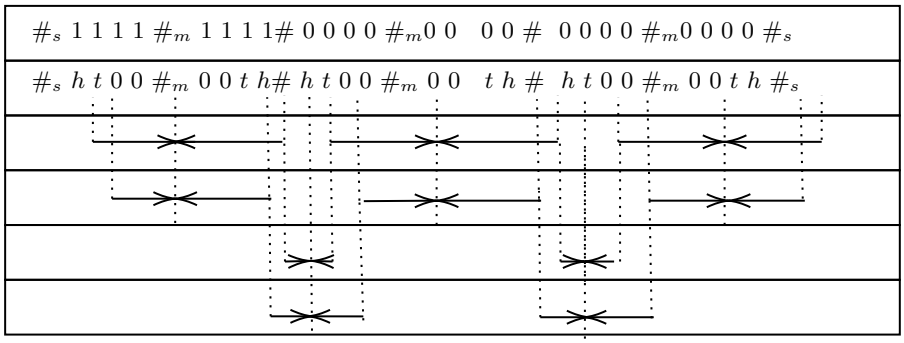


Fig. 2. Example of the verification dynamics for a periodic pattern. In this case, the pattern is given by the second row of a block representing the edge $(1, 2)$ in some graph.

1. $B(e)$ has $2m+1$ special symbols located at specific positions. More precisely $B(e)_{i,(n+1)(j-1)+1} \in \{\#_s, \#_m, \#\}$ for $i = 1, 2, 3$ and $j = 1, \ldots, 2m$.
2. Its first row $B(e)_{1\cdot} \in \{0,1\}^{(2n+2)m+1}$ is a 1-marker in $(2n+2)(e-1)+2$ of length n and a $1-$marker in $(3n+3)(e-1)+2$ of length n; and
3. $B(e)_{2\cdot} \in \{0,1\}^{(2n+2)m+1}$ is a periodic sequence of period $2n$ containing the row of M corresponding to e and a copy of this row in reverse order. More precisely, $B(e)_{2,[(2n+2)(i-1)+1,(2n+2)i]} = M_{\cdot,e}$ and $B(e)_{2,[(2n+2)i,(2n+2)(i+1)]} = \sigma(M_{\cdot,e})$ where σ is the permutation such that $\sigma(\{1, \ldots, k\}) = \{k, \ldots, 1\}$ for some fixed $k \in \mathbb{N}$.

For an example of a block for a some graph D see Figure 4.

First, observe that SPEC is in **NL** as a consequence of the Theorem 1. Now for the **NL**-hardness, let us take the problem STCON consisting in given a digraph $D = (N, A)$ and two nodes $s, t \in V$ deciding whether there exists a path connecting s and t. Let $(D = (N, A), s, t)$ be an instance of STCON. Observe that any path between s and t can be seen as a sequence of edges $e_1, \ldots, e_\ell$ such that $e_1 = (s, v)$, $e_i = (u', v') \implies e_{i+1} = (v', w')$ for some $u', v', w' \in N$, $i \in \{2, \ldots \ell - 1\}$ and $e_\ell = (w, t)$ for some $v, w \in N$. Besides, since each edge can be represented by a block then, an (s, t)-path P can be represented as a sequences of blocks $B(e_1), \ldots, B(e_\ell)$. Now, the main idea of the proof is to construct a non-deterministic automata network $\mathcal{A}_D = (G_D, \mathcal{F}_D)$ defined over a two dimensional grid of size $k \times d$ where $d = n^{\mathcal{O}(1)}$ and $k = \mathcal{O}(1)$. Of course, since k is constant, then $\mathcal{A}_D$ has bounded pathwidth. This automata network, will non-deterministically guess a sequences of blocks representing edges in A (selection phase). Needless to say that, the first part of the proof will be showing that $\mathcal{A}_D = (G_D, \mathcal{F}_D)$ is capable of deterministically verify if a n initial condition corresponds to a sequence of valid edges, i.e. if it corresponds to a sequence of blocks and they actually represent edges in A (verification phase). In order to perform these task, we use a construction based on using signals that will collide at specific locations as a way to verify the distance between two given cells. In addition, it would be essential to save (as a constant layer) the infor-

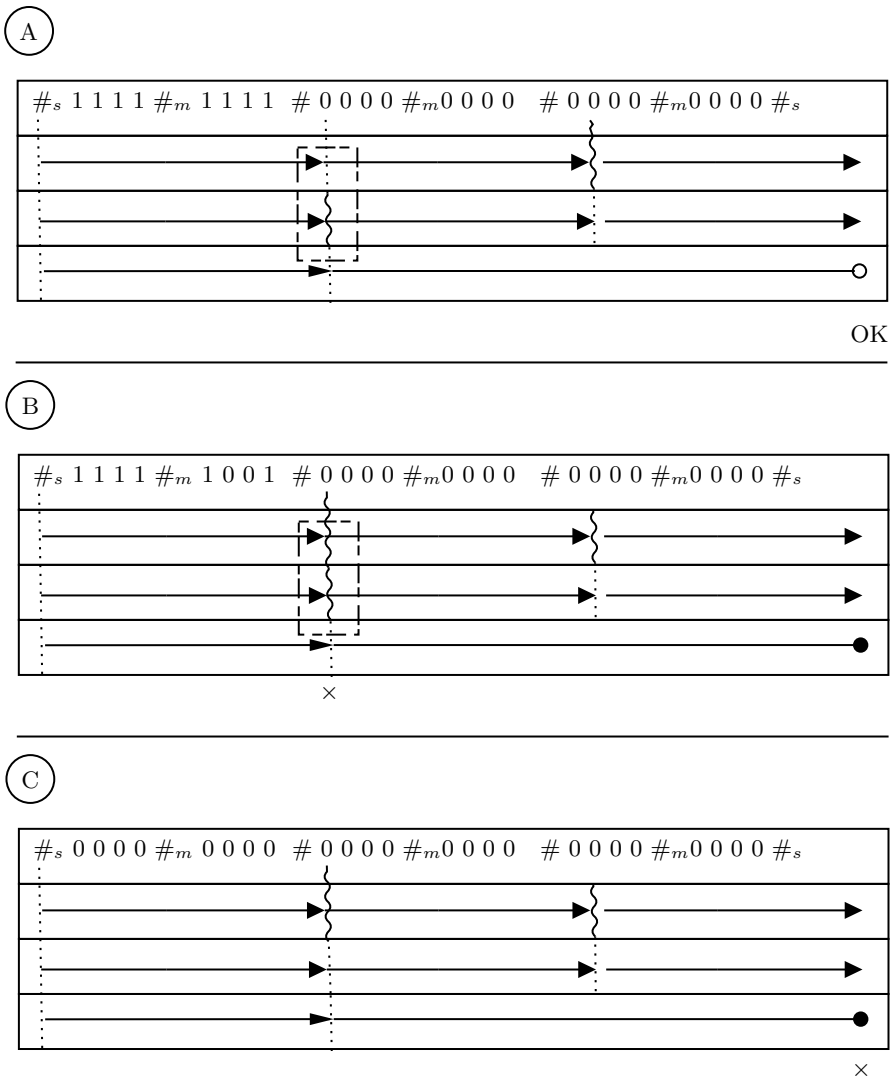
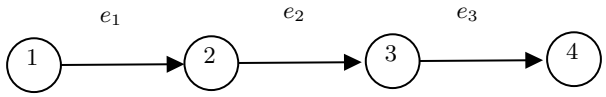


Fig. 3. Example of the verification dynamics for a marker. (Upper panel) A successful verification of a marker. If exactly one zone has only cells in state 1 an acceptance state will be reached. (Middle panel) An error in the verification raised by a zone in which cells in state 0 and 1 were identified. In this case, an error state is propagated. (Lower panel) An error in the verification raised by the signal only reading cells in state 0. In this case, an error state is propagated.

$B(e_1)$

$\#_s$ 1 1 1 1 $\#_m$ 1 1 1 1 # 0 0 0 0 $\#_m$0 0 0 0 # 0 0 0 0 $\#_m$ 0 0 0 0 $\#_s$
$\#_s$ h t 0 0 $\#_m$ 0 0 t h # h t 0 0 $\#_m$ 0 0 t h # h t 0 0 $\#_m$ 0 0 t h $\#_s$



D

Fig. 4. An example of a block for some graph D.

mation contained in the incidence matrix of D. Generally speaking, once $\mathcal{A}_D$ has verified that the sequence of blocks is valid, it will compare two subsequent blocks B_i and B_{i+1} in order to verify that if $B_i = (u, v)$ then $B_{i+1} = (v, w)$ for some $u, v, w \in N$. If in any part of its dynamics $\mathcal{A}_D$ detects some error, it will spread an error state that will lead the system to an attractor corresponding to a uniform configuration in which any cell will be in this particular error state. However, if the process runs flawless, then the system will reach an attractor in which all the cells are in a particular success state. We will code, by using a specification $\mathcal{E}_D$ (given in the input of SPEC), a specific requirement for the initial configuration (more precisely, we will ask the initial configuration to have the incidence matrix of D, markers and information about the nodes (s, t)) in order to allow $\mathcal{A}_D$ to have enough information to start the selection and verification process. In addition, we will code in this specification only the orbits that will reach this specific success state. By doing this, we will show that $\mathcal{A}_D \models \mathcal{E}_D$ if and only if there is a path between s, t in D. Thus, the reduction will consist on constructing $(\mathcal{A}_D, \mathcal{E}_D)$ from (D, s, t) in **DLOGSPACE**.

Now, we give details on the construction of $\mathcal{A}_D$ and $\mathcal{E}_D$:

1. $Q_D = Q_P \cup Q_{\text{core}} \cup Q_{\text{signal}}$ where $Q_{\text{core}} = \{\text{Success}, \text{Accept}, \text{Error}, 1, 0, \#_s, \#_m, \#\}$, $Q_P = \{P_1, \ldots, P_4\}$ are the states which indicate the different phases that are specified in the paragraph bellow and Q_{signal} are the states used in order to propagate signals (for example, the ones that we have used them on the previous lemmas).
2. Since we need to code an arbitrary path, d will be of size at most $m \times b$ where $b = (2n + 2)m + 1$ is the size of a block. Observe that size can be fixed since we can always assume that thexre is a loop in the terminal node t so we can consider that all the paths are coded by m blocks with possible padding of blocks $B((t, t))$. Thus, $d = n^{\mathcal{O}(1)}$.
3. k will be the number of rows of the grid. We will essentially use one row for the incidence matrix (incidence row), two rows for the blocks (selection row) and a constant number of rows that we will call working rows in which the signals will move and collide (working and verification rows).

4. $\mathcal{E}_D$ will code only initial conditions in which one row of the grid (incidence row) will have $\mathcal{O}(m)$ copies of the incidence matrix in the same format than the second row in blocks i.e. there are markers at specific positions and we code the different columns in the zones defined by the markers (Figure 5).
5. $\mathcal{E}_D$ will code orbits in which the selection row of size d will have marked in the first block a symbol indicating "head" in the position associated to node s (see Figure 4).
6. $\mathcal{E}_D$ will code orbits in which the selection row of size d will have marked in the last block a symbol indicating "tail" in the position associated to node t (see Figure 4).
7. $\mathcal{E}_D$ will code orbits which will reach a uniform success state.

Now we will describe the dynamics of the automata network $\mathcal{A}_D$.

Initialization In $t = 0$, since by construction of the specification $\mathcal{E}_D$, we can consider only orbits in which the incidence row, all the special symbols and the position of the source and terminal node (as an h and t symbol fixed in its correspondent positions) are well coded and fixed in the initial condition.

Selection phase First, the local rules will non-deterministically guess the states of the rest of the cells in the selection row. This process is performed cell by cell, by sending a traveling signal in one of the working rows. This signal starts on a starting symbol $\#_{\mathrm{s}}$ and finishes in a terminal symbol $\#_{\mathrm{s}}$. After doing that, the signal comes back from the terminal symbol to the starting symbol and writes a change of phase state in all the cells on the working row.

Verification phase After that, verification phase starts. The process has two main subphases:

A local phase: First, each block is internally verified. More precisely, the local rules will verify that $B(e)$ has the correct formatting on its two rows and that it correspond to an actual edge in D, i.e. $e \in A$:

1. *Verification of the first row.* For each part of size $2n$ defined by two different special symbols (i.e. the space bounded by pairs $(\#_s, \#), (\#, \#_s)$ or $(\#, \#)$)), three different signals will start from one symbol to the one in its left (see Figure 3). The first signal will change of state if and only if it reads a cell in state 0. If it remains in initial state it will be interpreted as success otherwise, if it has changed, then it will be interpreted as error. The second one will do the same thing but for cells in state 0. Finally, the third signal will start from a cell marked with $\#_s$ and will go through the row until another $\#_s$ symbol is reached. This signal will verify that there is exactly one block which has marked success for the first signal and error for the second one. Otherwise, it will change to an error state that will be spread to all the cells (see Figure 3 for examples).
2. *Verification of the second row.* In order to verify that the coding of the second row is correct, we need to check that each row has exactly two symbols: h

and t and that the configuration is symmetric related to the cells marked with symbols $\#_m$. In order to do that, from each symbol h and t a signal is sent through two working rows (one signal to the right and one to the left, see Figure 2 for details.) Then, the local rules in the cells holding the state $\#_m$ will change to the success if exactly two signals arrive at the same time. More precisely, this last procedure is implemented by sending a two state signal, one marking the starting part of the signal and one marking the rest. If the latter condition does not hold, the cells marked by $\#_m$ will spread an error state (see Figure 2). Observe that this procedure works since: i) if two cells are holding the same state and they are at the same distance of the cell marked by $\#_m$ then, the two equal signals will arrive at the same time to the cell holding the sate $\#_m$; and ii) since the coding considers a constant amount of special symbols (more precisely h and t) then, the local rule is freezing.

3. *Verification of the edge that is coded in the block.* At this point, if no error state has been produced by the dynamics, it means that the coding of each block is coherent, but we are not sure that it actually represents an edge $e \in A$. In fact, we have coded in the first selection row some number i referencing a column of the adjacency matrix of D but, we need to check whether the second selection row contains the same information than the i-column of the adjacency matrix. This last part is performed in the following way: a signal will be transmitted over a working row in order to identify the information in the two selecction rows of the block. Since each block has a marker in its first row, the signal can hold a state while it is in the same position than the cells in state one in the marker. Thus, this state will indicate the local rule to perform a comparison between the second row of the block and the correspondent part of the incidence row. For more details see Figure 5. While verifications are being run, the local rule will write an acceptance state or an error state in some working row. Finally, a third signal will verify that all the cells in the latter working row are in the acceptance state and will spread the error state if not. Finally, if no error state has been spread, the local rule updates the state of the cells in the working row holding the change of phase state.

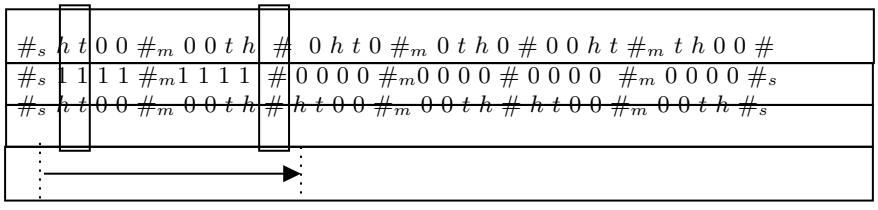


Fig. 5. Example of adjacency verification. In the first row, the incidence matrix of D is coded. In this case a signal verifies that the edge $(1, 2)$ is in the graph D (see Figure 4)

A pair-wise coherent phase: similarly to the verification of the second row in the selection row on the previous phase, this phase starts by sending multiple signals that are sent from the cells in the selection row with states given by the symbols marking the tails and the heads of the coded edge in the second row of the selection row on each block. These signals are sent through two different working tapes. Each of these signals will carry a special state indicating if its origin was a head or a tail. The local rule in the cells with a special symbol ($\#_s$) will verify whether a head signal has collided with a tail signal (see Figure 6). If exactly one of this collision take place, the local rule will write an accept state in one of the working rows. Finally, in other working row, a signal starting from the starting symbol will verify that at the position of the beginning (ending) of a block an accept state is written in the previous working row. The local rule will update the cells in that working row to an error state that will spread if at least one the verifications is not correct. Otherwise, it will update the cells to the success state.

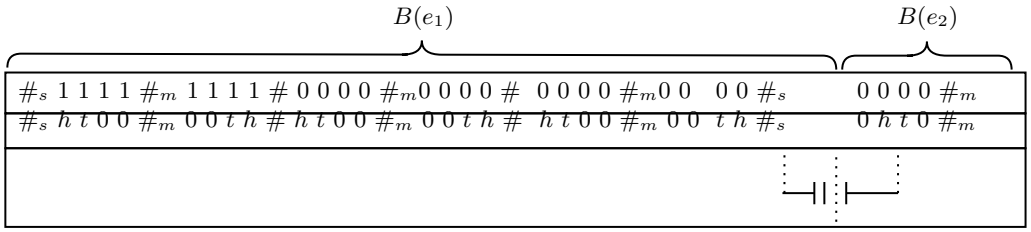


Fig. 6. Example of a verification dynamics which compares blocks and checks if the corresponding edges are both incident to the same node. If exactly one tail signal collide with a head it means that the previous property is verified and an error state is spread otherwise.

Now, we turn into show that the reduction hold. First, observe that the construction of $(\mathcal{A}_D, \mathcal{E}_D)$ can be done in **DLOGSPACE** since local rules does not depend on the structure of D and thus, we only need to store partial information related to the structure of the incidence matrix of D in order to define the specification. Then, we have that if there is a path between s and t on D, by construction, there must be at least one orbit of $\mathcal{A}_D$ which satisfies $\mathcal{E}_D$. Conversely, if $\mathcal{A}_D \models \mathcal{E}_D$ then, there exist at least one initial condition which codes a sequence of edges in D which leads the system to a uniform success fixed point. By construction, this attractor is only reachable (starting from the set of valid initial conditions) after all the previous phases are successfully performed by the dynamics. Then, we deduce that $\mathsf{STCON} \leq_m^{\mathbf{DLOGSPACE}} \mathsf{SPEC}$ and thus, SPEC is **NL**-hard. The theorem holds.

D Proof of Lemma 1

Proof. There exists a Turing machine working in polynomial time (and space) that on input (Ψ, v) where Ψ is a SAT formula and v a candidate valuation checks whether v satisfies Ψ. Then for any given SAT formula Ψ, one can produce in LOGSPACE a set of HV-domino constraints that accepts only bi-dimensional configurations $(a_{i,j})$ which represent a valid space-time diagram of the above machine which are correctly initialized and with Ψ enforced as the first component of the input. The encoding of space-time diagram of Turing machine inside domino constraints is well-known and usually presented through a fixed set of so-called Wang tiles (see for example [15]), which are just a uniform set of horizontal constraints $H \subseteq Q^2$ and vertical constraints $V \subseteq Q^2$. Note that since the HV-domino constraints considered here are non-uniform, we can hard-code the initial state of the machine in the lower-left corner of the configuration, the encoding of Ψ in the initial row, and the accepting state of the machine in the top row. The reduction from SAT to the HV-domino CSP follows. □

E Proof of Theorem 3

Proof. We proceed by reduction from the HV-domino CSP: given n and constraints $(H_{i,j})$ and $(V_{i,j})$, we build a deterministic automata network $(G_N, \mathcal{F})$ with $N = n^2$ which verifies ϕ if and only if the CSP has a solution. G_N is the graph with nodes $V = \{1, \dots, N\}$ and edges (i, i) for all $i \in V$ and $(i, i+1)$ for all $i < N$ and $(i, i-1)$ for all $i > 0$. G_N has pathwidth 1 and degree 3. The automata network $\mathcal{F}$ has four components plus a global error state and uses alphabet $Q' = Q \times \{0,1\} \times Q_h \times Q_t \cup \{\bot\}$ (where Q is the alphabet of the HV-domino CSP). The freezing order on $Q \times \{0,1\} \times Q_h \times Q_t$ is simply the product of orders on each component, and this order is extended to Q' by taking $\bot$ as a maximal element. The overall behavior is as follows (see Figure 7 and Figure 8).

- $\bot$ is an invariable spreading error state: as soon as some node is in state $\bot$, its neighbors change to $\bot$ in one step.
- The Q-component contains a candidate configuration $(a_{i,j})$ written as a one-dimensional word $a_{1,1} \cdots a_{1,n} a_{2,1} \cdots a_{2,n} \cdots a_{n,1} \cdots a_{n,n}$. The block of nodes $(j-1)n+1$ to $(j-1)n+n$ will be referred to as *block* j and it contains line j of the matrix $(a_{i,j})$ in its Q-component. This component never changes, except when an error state $\bot$ invades the network, or when some H-constraint $H_{i,j}$ is violated at some node in which case a $\bot$ state is generated. The freezing order on this component can be chosen arbitrarily.
- The $\{0,1\}$-component is called *dummy component* which never changes, has no influence on other components, and is just here to ensure that any configuration leading to $\bot^N$ has enough preimages (see Claim E below).
- The Q_h component handles a global control *head* whose main behavior is a back-and-forth movement from node 1 to node N and back to node 1. More precisely, the head do so on a set Σ of well-formed configurations and

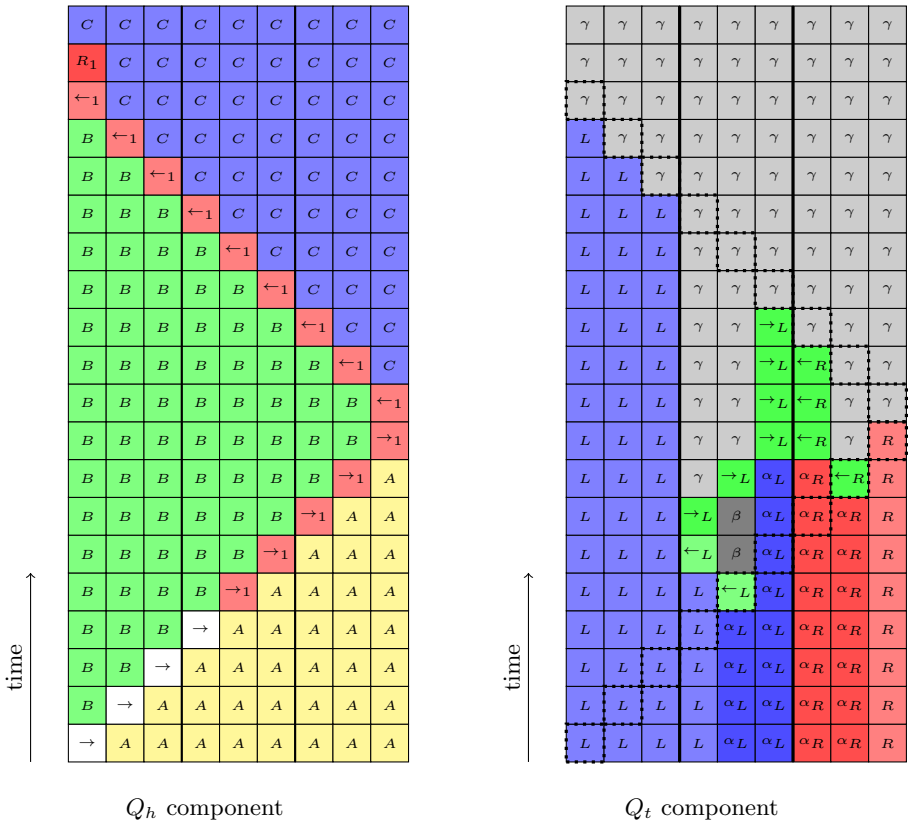


Fig. 7. Example of valid orbit with $n = 3$ starting from a configuration testing the V-constraint $V_{2,2}$ and resulting in a positive output. The trajectory of the Q_h head is reproduced on the Q_t-component to clarify the interactions. The vertical thick lines represent separations between consecutive blocks.

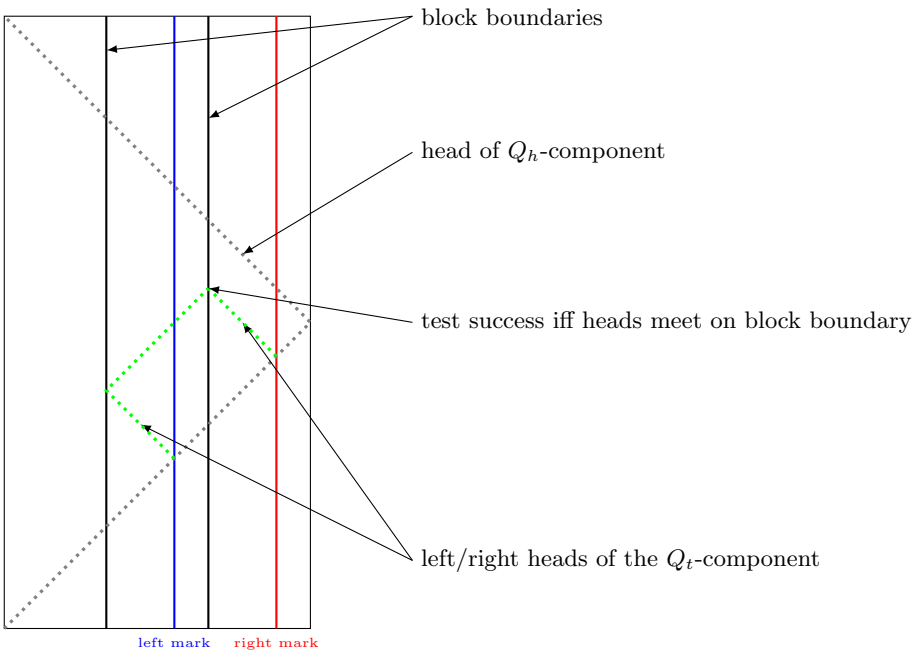


Fig. 8. Euclidean non-discretized representation of the verification process ensuring that the marks in two consecutive blocks have the same offset within the block and thus mark two positions which are vertical neighbors in the matrix $(a_{i,j})$, *i.e.* of the form $a_{i,j}$ and $a_{i+1,j}$.

any ill-formed configuration is detected locally and generates an error state $\perp$. We set $Q_h = \{0,1\} \times \{0,1\} \times \{A, B, C, \rightarrow, \rightarrow_0, \rightarrow_1, \leftarrow_1, \leftarrow_0, R_0, R_1\}$ with freezing order $0 < 1$ on the two $\{0,1\}$ components and $A <\rightarrow<\rightarrow_0<\rightarrow_1< B <\leftarrow_1<\leftarrow_0< R_0 < R_1 < C$ on the remaining component. Σ is defined by forbidding a set of pairs of states to occur two adjacent symbols $c_i c_{i+1}$ from the third component. Moreover, we add the constraint that node 1 cannot be in state A, and that the first $\{0,1\}$ component at this node must be 1. Precisely, configurations authorized in Σ are the following (without considering the $\{0,1\}$ components):

1. $\rightarrow A^{N-1}$ or $\rightarrow_0 A^{N-1}$ or $\rightarrow_1 A^{N-1}$,
2. $B^i \rightarrow A^{N-i-1}$ or $B^i \rightarrow_0 A^{N-i-1}$ or $B^i \rightarrow_1 A^{N-i-1}$,
3. $B^{N-1} \rightarrow_0$ or $B^{N-1} \rightarrow_1$,
4. $B^{N-1} \leftarrow_0$ or $B^{N-1} \leftarrow_1$,
5. $B^i \leftarrow_0 C^{N-i-1}$ or $B^i \leftarrow_1 C^{N-i-1}$,
6. $\leftarrow_1 C^{N-1}$ or $\leftarrow_0 C^{N-1}$,
7. $R_1 C^{N-1}$ or $R_0 C^{N-1}$,
8. C^N.

The head is the unique arrow occurring in each configuration and its dynamics is as follows. It moves to the right in a background of As and letting symbols B behind (configuration types 1, 2 and 3). At each move to the right, the first $\{0,1\}$ component of the node left by the head is reset to 1. When doing so it can turn at some point to state $\rightarrow_1$ or $\rightarrow_0$ depending on the layer of states Q_t as detailed below: these states represent a head holding a YES/NO bit of information about the output of the test process happening on component Q_t. This bit must appear before reaching node N and once appeared, this bit of information never changes in the future. When reaching node N the head starts to move to the left, progressing in a background of Bs and letting symbols C behind (configuration types 4 and 5). At each move to the left, the second $\{0,1\}$ component of the node left by the head is reset to 1. The fact that some $\{0,1\}$ component is reset to 1 at each head move ensures that the corresponding configurations have more than one preimage (which is a key aspect when considering formula ϕ). Finally, the head reaches node 1 and must hold the output bit of the test process (configuration type 6), maintain it one step (configuration type 7), and finally erase it (type 8). Also, when reaching a configuration of type 6 at node 1, the bit of the second $\{0,1\}$ component is reset to 1 when the head at node 1 is $\leftarrow_1$ and unchanged when it is $\leftarrow_0$. This bit is reset to 1 in any case for configurations of type 7. As a result, a configuration of type 7 has exactly one preimage if and only if it is $R_1 C^{N-1}$.

– The Q_t component is the *test* component, its role is to mark two positions in the configuration and interact with the head component in order to check a single V-constraint on the candidate configuration hold in the Q-component. More precisely, the test component ensures that the two marked positions are at distance n (*i.e.* they correspond to two vertical neighbors in the grid $(a_{i,j})$) and gathers locally at some node the information on the corresponding pair of states $a_{i,j}a_{i,j+1}$ and the constraint $V_{i,j}$ so that the head can check

whether $a_{i,j}a_{i,j+1} \in V_{i,j}$. See Figure 8 for an Euclidean intuition of how the distance equality test works. This behavior is implemented using alphabet $Q_t = Q \times Q \times \{L, R, \alpha_L, \alpha_R, \beta, \gamma, \leftarrow_L, \leftarrow_R, \rightarrow_R\}$ with freezing order: $L < R < \alpha_L < \alpha_R <\leftarrow_L< \beta <\rightarrow_L<\leftarrow_R< \gamma$. The third sub-component of Q_t is used to mark two positions in the configuration as well as check that the distance between the two marked positions is exactly n so that they indeed correspond to a pair of positions (i, j) and $(i, j + 1)$ in in the matrix $(a_{i,j})$. Its behavior is based on a set of valid configurations Σ^+ defined by local rules and synchronized with the Q_h component. States $\leftarrow_L, \leftarrow_R, \rightarrow_R$ are called "left/right arrows" of the Q_t-component and are generated at specific positions when the global Q_h-head passes by (see Figure 8). The two Q-sub-components of Q_t are forced to hold states $a_{i,j}$ and $a_{i,j+1}$ respectively on valid configurations, and allow to check the V-constraint $(a_{i,j}, a_{i,j+1}) \in V_{i,j}$. The conditions defining Σ^+ are local (*i.e.* they can be defined as a list of admissible pair of states between neighboring nodes) and a $\perp$ state is triggered whenever and wherever an invalid local pattern is detected. The conditions are the following:

- First, in the absence of a left-moving head in the Q_h component, the two Q-sub-component must be uniform: each one is of the form q^N for some $q \in Q$. When there is a left-moving head in the Q_h component, each Q-sub-component is of the form: $q^i q_0^{N-i}$ where q_0 is the maximal state of Q and i is the position of the Q_h head.
- Then, there are five types of admissible configurations on the third sub-component of Q_t:
  1. $L^*\alpha_L^+\alpha_R^+R^*$ and α_L and α_R segments are forbidden to cross a block boundary (*i.e.* node (n, j) has an α_L if and only if $(1, j + 1)$ has an α_R),
  2. $L^* \leftarrow_L \beta^*\alpha_L^*\alpha_R^*R^*$,
  3. $L^*\gamma^* \rightarrow_L \beta^*\alpha_L^*\alpha_R^*R^*$ or $L^*\gamma^* \rightarrow_L \beta^*\alpha_L^*\alpha_R^* \leftarrow_R \gamma^*R^*$,
  4. $L^*\gamma^* \rightarrow_L\leftarrow_R \gamma^*R^*$,
  5. any configuration of the form $c\gamma^*$ where c is the prefix of a configuration of type 3 or 4.
- Type 4 configurations are only authorized when $\rightarrow_L$ and $\leftarrow_R$ states meet at a bloc boundary, *i.e.* are at positions of the form (n, j) and $(1, j + 1)$ (respectively).
- Moreover, only L, α_L, α_R and R are authorized in a node whose Q_h component is in state A, therefore a type 1 configuration on the Q_h component admits only a type 1 configuration on the Q_t component.
- Finally, in configurations of type 1, let (i, j) (*i.e.* $n(j - 1) + i$) be the leftmost node in state α_L and let m be the rightmost node in state α_R. Denote by a and b the states of the first and second Q-sub-component respectively. Then it must hold that a is the state of the Q-component (the global one of the alphabet Q') of node (i, j) and b is the state of the Q-component of node m.

The dynamics of this Q_t-component is as follows and respects the type order of configuration described above:

- Type 1 configurations don't change until the head of the Q_h-component arrives at node (i, j) where it generates a $\leftarrow_L$ state.
- Then, $\leftarrow_L$ propagates in the L background, letting β states behind and until position $(1, j)$ is reached (*i.e.* the first position to the left which is at the beginning of a bloc). Then, the arrow bounces by turning into $\rightarrow_L$ and starts to progress to the right letting γ states behind.
- Meanwhile, when the Q_h head reaches position m (the rightmost node in state α), it launches a $\leftarrow_R$ state in the Q_t layer which starts to propagate to the left letting γ sates behind.
- Also, when the Q_h head bounces on node N and starts to propagate to the left, it writes q_0 on each Q-sub-component and γ on the third sub-component of Q_t, thus erasing progressively any information about the marked positions and the V-constraint being tested.
- The dynamics ends into the fixed point equal to q_0^N on each Q-sub-component and γ^N on the third sub-component.

Finally the Q_t-component influences the Q_h-component as follows: when the head of the Q_h-component of type $\rightarrow$ reaches node (i, j) it becomes $\rightarrow_1$ if $(a, b) \in V_{i,j}$ (where a and b are the states of the Q-sub-components) and $\rightarrow_0$ else.

Let us now prove that this construction has the desired property. Let's call valid orbit any orbit without occurrence of $\perp$.

Claim (ϕ checks V-constraints on valid orbits). Consider any valid orbit starting from a configuration y without preimages, with $y \rightarrow y^1$ and $\neg P_2(y^1)$, and reaching a fixed point x. Then y is of type 1 on components Q_h and Q_t. Moreover, a correctly encoded test of V-constraint $V_{i,j}$ is encoded in component Q_t and the configuration z such that $y \rightarrow^+ z$ and $z \rightarrow x$ verifies $\neg P_2(z)$ if and only if $a_{i,j}a_{i,j+1} \in V_{i,j}$.

Proof. Since there is no occurrence of $\perp$, the whole orbit belongs to Σ^+. A configuration of type 8 or 9 in the Q_h component always has a preimage so y is not of this type. A configuration of type 2,3,4,5,6 or 7 has a moving head that reset some $\{0, 1\}$ component to 1, so it cannot be the unique preimage of its successor, contradicting the hypothesis on y^1. Therefore y is of type 1 on components Q_h and Q_t. Then, by construction, the marked positions in the Q_t component are at distance n and there is a well-formed V-constraint test happening (otherwise a $\perp$ would be generated later in the orbit). The dynamics of the automata networks then ensures that the Q_h heads holds the bit of information corresponding to the validity of the encoded V-constraint: it is 1 if and only if $a_{i,j}a_{i,j+1} \in V_{i,j}$. The dynamics ends in a fixed point x which has a configuration of type 8 on the Q_h-component. Already when reaching a configuration of type 6 or 7 or 8 on the Q_h-component, all the Q_t-component has been reset to a default value. Therefore it holds that the bit of information in the head is 1 if and only if the type 7 configuration reached $z = R_1C^{N-1}$ has a unique preimage. □

From the construction and Claim E it should be clear that if the HV-domino CSP has a solution $(a_{i,j})$, then one can encode it into a fixed point configuration

x that satisfies the orbit property expressed in ϕ for all admissible choices of initial configuration y (because all admissible V-constraint tests are satisfied by the CSP solution). In this case the automata network verifies ϕ.

Conversely, if the automata network verifies ϕ and if the fixed point x can be chosen to be a configuration without $\perp$, then this configuration encodes a solution to the HV-domino CSP by Claim E and because any valid V-constraint test can be encoded in an appropriate initial configuration y. It remains to discard the possibility that ϕ is valid because x is chosen to be the invalid fixed-point $\perp^N$, this is the purpose of the following claim.

Claim (ϕ discards invalid orbits). Consider three configurations y, z, x such that $y \to^+ z \to x$ and $x \to x$ and $x \neq z$. If $\neg P_2(z)$ then x cannot be the configuration $\perp^N$.

Proof. First z must have an occurrence of $\perp$ because it is impossible that the preimage z' of z be everywhere correct and in one step becomes a configuration z everywhere incorrect but without occurrence of $\perp$: indeed, by construction, the changes not involving $\perp$ state that can occur in a configuration in one step are only in the neighborhood of arrow states of both Q_h and Q_t components, and they have a bounded number of occurrences by definition of Σ^+. Moreover, there must be an occurrence of $\perp$ in z at position i such that $z'(i) \neq \perp$. Indeed, otherwise it would imply $z = \perp^N$ which is impossible under the hypothesis. Therefore by just changing the dummy component at i in z' we produce another preimage of z, so $P_2(z)$ holds which is a contradiction. □

We have thus shown that the HV-domino CSP has a solution if and only if the automata network verifies ϕ. The theorem follows since the construction can be computed efficiently (actually in LOGSPACE). □

On the Dynamics of Bounded-Degree Automata Networks

Julio Aracena[4], Florian Bridoux[3], Pierre Guillon[1], Kévin Perrot[2], Adrien Richard[3], and Guillaume Theyssier[1]

[1] Aix-Marseille Université, CNRS, I2M UMR7373, Marseille, France
[2] Aix-Marseille Université, Univ. Toulon, CNRS, LIS UMR7020, Marseille, France
[3] Univ. Côte d'Azur, CNRS, I3S UMR 7271, Sophia Antipolis, France
[4] Departamento de Matemáticas, Universidad de Concepción, Chile

Abstract. Automata networks can be seen as bare finite dynamical systems, but their growing theory has shown the importance of the underlying communication graph of such networks. This paper tackles the question of what dynamics can be realized up to isomorphism if we suppose that the communication graph has bounded degree. We prove several negative results about parameters like the number of fixed points or the rank. We also give bounds on the complexity of the problem of recognizing such dynamics. However, we leave open the embarrassingly simple question of whether a dynamics consisting of a single cycle can be realized with bounded degree.

1 Introduction

One possible definition for a boolean automata network is simply a self-map $F : \{0,1\}^n \to \{0,1\}^n$. This definition forgets about the computational aspect of the model, which consists in, seen from a dual point of view, a set of n automata linked by some arcs, and each holding a bit that they can update depending on that of their incoming neighbors.

As a model of computation generalizing finite cellular automata, this communication graph is quite relevant, and it is natural to constrain it, and in particular try to restrict the possible degrees: a small degree indeed represents simple local computations. Note indeed that a complete communication graph can yield any dynamics $F : \{0,1\}^n \to \{0,1\}^n$. The minimal communication graph, often called interaction graph, plays an important role in automata network theory (see[**?**] for a survey). It was already established that some dynamics requires high degree and even a dense communication graph [**?**].

In this paper, we address the question of how restrictions on the communication graph, and in particular bounding its degrees, can impose restrictions on the possible dynamics. For instance, in Figure 1, one can see three (families of) graphs representing possible dynamics. Which are the ones that can be realized by communication graphs with small degree?

In Section 3, we establish bounds on different parameters of the dynamics depending on the degree of communication graphs. This in particular allows to

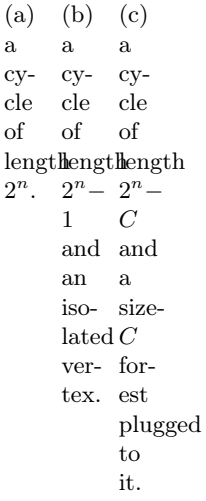


Fig. 1. Three examples of dynamics on 2^4 configurations.

show that the family of dynamics from Figure 1(c) cannot be realized with a bounded-degree communication graph. In Section 4, we give some constructions using feedback shift registers that in particular allows to realize dynamics of the type from Figure 1(b) with communication graphs of degree 2. Finally, in Section 5, we give upper and lower bounds for the computational complexity of recognizing dynamics that can be realized with a bounded-degree communication graph.

However, we leave open the question about the minimum degree necessary to realize dynamics from Figure 1(a). Prior to this work, J. Aracena has communicated to us the conjecture that such dynamics requires unbounded degree. This also appears in [**?**] with various intermediate results.

2 Definitions and notations

Consider a finite alphabet Q with $q = |Q|$ symbols. Without loss of generality, $Q = \{0, \dots, q-1\}$. Consider also a set $V = \{1, \dots, n\}$ of n *nodes*. A *configuration* $x = (x_i)_{i \in V} \in Q^V$ is a function $V \to Q$. For every $U \subseteq V$, we denote $x_U : U \to Q$ the restriction of x to U, *i.e.*, $x_i = (x_U)_i$ for every $i \in U$. Given a *pattern* $u \in Q^U$, we define the *cylinder* $[u] = \{x \in Q^V : x_U = u\}$.

An *automata network* (AN) is a map $F : Q^V \to Q^V$. It can be represented as a *dynamics graph*, like those from Figure 1, by linking each configuration x to its image $F(x)$. This graph is denoted by $\mathcal{D}(F)$. A configuration x such that $F(x) = x$ is called a fixed point, and the number of fixed points of F is denoted $\mathrm{fp}(F)$. The *rank* of F is its number of images and is denoted by $\mathrm{rk}(F)$. The set of ANs with alphabet of size q and with n nodes is denoted $\mathcal{F}(n, q)$.

A *communication graph* for F is a graph over vertex set V such that for every $i \in V$, and every $x, x' \in Q^V$ which agree over the in-neighborhood $N^-(i) \subset V$ of i, $F(x)_{N^-(i)} = F(x')_{N^-(i)}$. In other words, the value $F(x)_i$ is updated thanks to a local function $f_i : Q^V \to Q$ which depends only on the values $x_{N^-(i)}$. For $U \subseteq V$, we may also denote $f_U(x) = F(x)_U$. The *interaction graph* of F, denoted $G(F)$, is the minimal communication graph of F. Its *degree* is the maximum in-degree of a vertex in $G(F)$. We denote by $\mathcal{F}(n, q, d)$ the set ANs from $\mathcal{F}(n, q)$ which can be defined with a communication graph of degree at most d.

3 Non-local dynamics

Here we prove that some dynamics are intrinsically non-local in the sense that they cannot be realized by bounded-degree networks, even up to isomorphism.

Our first result shows that if $G(F)$ has bounded degree and F is not the identity, then the number of fixed points of F cannot be close to q^n.

Proposition 1. *Let $F \in \mathcal{F}(n, q, d)$ with $\mathrm{fp}(F) < q^n$. Then $\mathrm{fp}(F) \leq q^n - q^{n-d}$.*

Proof. Since F is not the identity map, there exists $x \in Q^V$ such that $f_i(x) \neq x_i$ for some $i \in V$. There are two cases. If $i \notin N^-(i)$, then for every pattern $u \in Q^{V \setminus \{i\}}$, there is a unique configuration $y \in [u]$ such that $f_i(y) = y_i = f_i(u)$; then, $\mathrm{fp}(F) \leq q^{n-1} \leq q^n - q^{n-d}$. If $i \in N^-(i)$, then let $u = x_{N^-(i)}$; for every configuration $y \in [u]$, $f_i(y) = f_i(x) \neq x_i = y_i$ and y is not a fixed point. Therefore, $\mathrm{fp}(F) \leq q^n - q^{n-d}$. □

Remark 1. The bound from the previous lemma is tight: indeed let $F(x) = x$ if $x_{1,\dots,d} \neq 0^d$ and $\pi x_1 x_{2,\dots,n}$ otherwise, where π is a permutation of Q without fixed point. Then F is an AN of degree $n-1$ with $q^n - q$ fixed points.

Proposition 1 can be generalised to the powers of F. First, note that if $F \in \mathcal{F}(q, n, d)$ then $F^k \in \mathcal{F}(q, n, d^k)$ for every $k \geq 1$ (because from $G(F)$ of degree $\leq d$ we obtain a communication graph for F^k by putting an edge for each path of length k). By combining this remark and Proposition 1, we obtain that, if $\mathrm{fp}(F^k) < q^n$ then $\mathrm{fp}(F^k) \leq q^n - q^{n-d^k}$.

As an application, we can easily find bijections without fixed points that force large communication degrees. Suppose for instance that the dynamics of $F \in \mathcal{F}(2, n)$ consists of $2^{n-1} - 2$ limit cycles of length 2 and one limit cycle of length 4. Then F^2 has exactly $2^n - 4$ fixed points. Denoting by d the degree of $G(F)$, we obtain that $2^n - 4 = \mathrm{fp}(F^2) \leq 2^n - 2^{n-d^2}$ and thus $d \geq \sqrt{n-2}$.

Remark 2. There are about $e^{\sqrt{2^n}}$ nonisomorphic bijective AN, but only $(q^{q^d})^n$ AN with degree $\leq d$. So few bijective AN have a realization with bounded degree.

Our second result shows that if $G(F)$ has bounded degree and F is not a bijection, then the rank of F cannot be close to q^n.

Theorem 1. *Let $F \in \mathcal{F}(n, q, d)$ with $rk(F) < q^n$. Then $rk(F) \leq q^n - \frac{n}{d+1}$.*

In particular, the family of dynamics depicted in Figure 1(c) is impossible to realize with bounded-degree ANs. However, Theorem 1 fails among bijective ANs of fixed degree, such as the dynamics depicted in Figure 1(c), as we will see in Section 4.

The key part of the proof of Theorem 1 consists in proving that AN dynamics cannot be close to bijective without being bijective (Lemma 2). We need some definitions. We say that $F \in \mathcal{F}(n,q)$ is *k-balanced* ($k \leq n$) if for any $U \subseteq V$ with $|U| = k$ and for any pattern $u \in Q^U$, it holds $\left|F^{-1}([u])\right| = q^{n-k}$. Note that if F is bijective then it is k-balanced for all $1 \leq k \leq n$. Moreover, if F is $(k+1)$-balanced, then it is k-balanced.

Given any property $\mathcal{P} \subseteq \mathcal{F}(n,q)$ of ANs, we say that a given $F \in \mathcal{F}(n,q)$ is *k-almost $\mathcal{P}$* ($k \leq q^n$) if there exists $F' \in \mathcal{P}$ such that F and F' differ on at most k configurations, *i.e.* $\left|\{x \in Q^V : F'(x) \neq F(x)\}\right| \leq k$. Observe that if the base property $\mathcal{P}$ is invariant under isomorphism, then being k-almost $\mathcal{P}$ is also invariant under isomorphism. For instance, *being k-almost bijective* is invariant under isomorphism.

Lemma 1. *$F \in \mathcal{F}(n,q)$ is k-almost bijective if and only if $rk(F) \geq q^n - k$.*

Proof. $\Rightarrow$: Suppose $F \in \mathcal{F}(n,q)$ is k-almost bijective. Then there exists $F' \in \mathcal{F}(n,q)$ bijective and $X \subseteq Q^V$ with $|X| = q^n - k$ and $F'(X) = F(X)$. Since F' is bijective $|F'(X)| = |X| = q^n - k$ and since $F'(X) = F(X) \subseteq F(Q^V)$ we have $\mathrm{rk}(F) \geq q^n - k$.

$\Leftarrow$: Suppose that F has rank $q^n - k$. Let $Y = \{y^1, y^2, \ldots, y^{q^n-k}\} \subseteq F(Q^V)$. Let $X = \{x^1, x^2, \ldots, x^{q^n-k}\}$ with $F(x^i) = y^i$ for all $1 \leq i \leq q^n - k$. Let $\overline{Y} = \{\overline{y}^1, \ldots, \overline{y}^k\} = Q^V \setminus Y$ and $\overline{X} = \{\overline{x}^1, \ldots, \overline{x}^k\} = Q^V \setminus X$. Let $F' \in \mathcal{F}(n,q)$ such that $F'(x^i) = y^i$ for all $1 \leq i \leq q^n - k$ and $F'(\overline{x}^i) = \overline{y}^i$ for all $1 \leq i \leq k$. F' is bijective and differs from F is k configurations. So F is k-almost bijective. □

Lemma 2. *Let $1 \leq k < n$. If $F \in \mathcal{F}(n,q)$ is k-balanced and k-almost bijective, then it is bijective.*

Proof. Suppose by contradiction that F is not bijective. If F is k-almost bijective, then there is some bijective $F' \in \mathcal{F}(n,q)$ which differs from F over a set of exactly $1 \leq \ell \leq k$ configurations, denoted $X = \{x_1, \ldots, x_\ell\}$. Because F' is bijective, the configurations $F'(x_1), \ldots, F'(x_\ell)$ are all distinct; moreover we cannot have $F(X) = F'(X)$ because $F(Q^V) \neq F'(Q^V)$ since one is bijective and not the other. So there are two cases:

Case 1: Assume that there exists $\leq j \leq \ell$ such that $F(x_j) \notin F'(X)$. In this case, the set $F'(X) \cup \{F(x_j)\}$ contains $\ell + 1$ distinct configurations. Let us inductively build a subset $U \subseteq V$ with $|U| = \ell$ such that the restrictions of these configurations to U are all distinct, *i.e.*,

$$\left|\{x_U \in Q^U : x \in F'(X)\} \cup \{F(x_j)\}\right| = \ell + 1.$$

When adding a configuration x in the set, either it is different over U to all previously included ones, then one does not need to change U; otherwise it is

equal to at most one d over U, then simply add an automaton that distinguishes x from d. Then this contradicts the fact that F is ℓ-balanced because the pattern $F(x_j)_U \in Q^U$ has at least one pre-image more under F than it has under F', which must be ℓ-balanced because bijective. Indeed, the pattern $F(x_j)_U$ has no pre-image by F' in X and F is similar to F' on $Q^V \setminus X$ but $x_j \in X$ is a pre-image of $F(x_j)_U$ by F.

Case 2: Otherwise, there are $x_i \neq x_j$ such that $F(x_i) = F(x_j)$. Then, following the same idea as in the previous case, we can find $U \subseteq V$ of size ℓ such that the restrictions to U of configurations $F'(X)$ are all distinct. For such an U, the pattern $F(x_j)_U \in Q^U$ has at least one pre-image more under F than it has under F', which contradicts the fact that both F and F' are ℓ-balanced. □

Lemma 3. *Consider $F \in \mathcal{F}(n, q, d)$ and $U \subseteq V$ with $k = |U| \leq \lfloor n/d \rfloor$. Then for any pattern $u \in Q^U$, the number of pre-images under F of the corresponding cylinder is a multiple of q^{n-kd}.*

Proof. Since the degree of $G(F)$ is upper-bounded by d, f_U only depends of $Y = \bigcup_{i \in U} N^-(i)$ and $|Y| \leq kd$. In other words, for every $x \in Q^U$ such that $f_U(x) = u$, we have $f_U([x_U]) = u$. Hence, $|F^{-1}([u])| = |\{v \in Q^Y \mid f_U(v) = u\}| q^{n-|Y|}$. Since $|Y| \leq k \cdot d$, this is a multiple of q^{n-kd}. □

Combining Lemma 2 and Lemma 3 we obtain the following.

Lemma 4. *Let $F \in \mathcal{F}(n, q, d)$ and $1 \leq k \leq \lfloor n/d \rfloor$. If F is k-almost bijective but not bijective then $k \geq q^{n-dk}$.*

Proof. By Lemma 2 F cannot be k-balanced, so there is a cylinder $u \in Q^U$ with $|U| = k$ such that $\alpha = |F^{-1}([u])| > q^{n-k}$. However, Lemma 3 gives that $\alpha = mq^{n-kd}$ for some $m > 0$. We deduce that $\alpha \geq q^{n-k} + q^{n-kd}$, so at least q^{n-kd} changes in F are necessary to recover k-balance (hence bijectivity). Since F is assumed k-almost bijective, we deduce $k \geq q^{n-dk}$. □

Proof (of Theorem 1). Let k such that $\mathrm{rk}(F) = q^n - k$ ($k \geq 1$). If $k > \lfloor n/d \rfloor$ we are done. Otherwise, by Lemma 4 we have $k \geq q^{n-dk}$ hence, $\log_q(k) \geq n - dk$ and $(d+1)k \geq \log_q(k) + dk \geq n$ (because $k \geq \log_q(k)$). □

Here is another application of Lemma 3.

Proposition 2. *Let $F \in \mathcal{F}(n, q, d)$ such that F is not constant. Then the number of preimages of any configuration is upper-bounded by $q^n - q^{n-d}$.*

Proof. Let $y \in Q^V$. Let us prove that $|F^{-1}(y)| \leq q^n - q^{n-d}$. Since F is not constant, there exists $z \in F(Q^V)$ such that $z_i \neq y_i$ for some $i \in V$. Since $F^{-1}([z_i]) \neq \emptyset$, by Lemma 3, $F^{-1}([z_i]) \geq q^{n-d}$. Furthermore, since $F^{-1}([z_i]) \cap F^{-1}(y) = \emptyset$, $|F^{-1}(y)| \leq q^n - q^{n-d}$. □

It is tight because we can have $F(x) = 0^n$ if $x_{1,\dots,d} \neq 0^d$ and 10^{n-1} otherwise.

4 Realization results via feedback shift registers

In this section, we are interested in realizing examples of AN with *almost degree* 1, *i.e.*, whose all but one nodes have degree at most 1.

One important tool for this is the following. Let $g : \{0,1\}^n \to \{0,1\}$, and $F_g : \{0,1\}^n \to \{0,1\}^n$ be the corresponding *feedback shift register* (FSR), that is, $F_g(x) = F_g(x_1, \ldots, x_n) = (x_2, \ldots, x_{n-1}, g(x))$. $G(F_g)$ is thus obtained from the path $n \to n-1 \to \cdots \to 1$ by adding an arc from i to 1 whenever g depends on input i: it has almost degree 1.

The *de Bruijn graph* of order n (over alphabet $\{0,1\}$) has set of vertices $V = \{0,1\}^n$ and set of arcs $E = \{(au, ub) : a, b \in \{0,1\}, u \in \{0,1\}^{n-1}\}$.

Proposition 3. *For any n and any $1 \leq k \leq 2^n$, the de Bruijn graph of order n admits a cycle of length k.*

Proof. The de Bruijn graph admits a *Eulerian* cycle because it is connected and all vertices have equal in- and out-degree. Since the Bruijn graph of order n+1 is the line digraph of the Bruijn graph of order n, we deduce that the de Bruijn graph admits a Hamiltonian cycle. Cycles of each length $0 < k < 2^n$ are a consequence of [**?**, Theorem 4]. □

Proposition 4. *For any n and any $0 \leq k \leq 2^n$, there exists $F : \{0,1\}^n \to \{0,1\}^n$ with almost degree 1 and whose maximum limit cycle has length k.*

Proof. Consider some cycle $C \subseteq \{0,1\}^n$ given by Proposition 3, and the feedback shift register F_g, where

$$g(x) = \begin{cases} b & \text{if } x = au \text{ and } au \to ub \in C \\ 0 & \text{otherwise.} \end{cases}$$

F_g has almost degree 1, and has the cycle C in its dynamics. To conclude the proof, it is sufficient to observe that the dynamics on the complement of C consists in adding 0 at node n and shifting node $i+1$ to node i for $i < n$. Therefore, the only possible cycle created by this part of the dynamics is possibly the fixed point $0 \cdots 0$. □

Now let us try to decrease the degree of the special vertex. Suppose that g is *additive*, *i.e.*, consider $\{0,1\}$ as the ring $\mathbb{Z}/2\mathbb{Z}$, and $g(x) = \sum_{i=1}^{n} a_i x_i$, for some coefficients $a_1, \ldots, a_n \in \{0,1\}$. In the corresponding interaction graph, $N^-(1) = \{i \mid a_i \neq 0\}$. The *characteristic polynomial* of g is $P = 1 + \sum_{i \in N^-(1)} X^i$. If P has degree n, then it is called *primitive* if it is irreducible and does not divide $X^k - 1$ for any $1 \leq k < 2^n - 1$. We say that P is a *trinomial* if it contains 3 terms, that is, if $P = X^n + X^k + 1$ for some $1 \leq k < n$.

Theorem 2 ([?]). *F_g has a limit cycle of length $2^n - 1$ and a fixed point if and only if P is* primitive *of degree n.*

Let us say that n is a *Mersenne exponent* if $2^n - 1$ is a (Mersenne) prime number.

Proposition 5. *For any Mersenne exponent $n \leq 3021377$, there is some order-n AN of degree 2 and almost degree 1 whose dynamics is the union of a limit cycle of length $2^n - 1$ and a fixed point.*

This corresponds to Example 1 and Figure 1(b).

Proof. If n is a Mersenne exponent and P has degree n, then P is primitive if and only if it is irreducible. For every Mersenne exponent $n \leq 3021377$, there exists at least one primitive trinomial P of degree n [**?**]. □

Example 1. Let $n = 5$, $q = 2$, and define the AN $F : \{0,1\}^5 \to \{0,1\}^5$ with $f_i(x) = x_{i-1}$ for $i \in \{2,3,4,5\}$, and $f_1(x) = x_3 \oplus x_5$ where $\oplus$ is the binary xor. Its dynamics has one fixed point and one cycle of length $2^n - 1$, while its interaction graph has degree 2 (see Figure 2), hence $F \in \mathcal{F}(5,2,2)$.

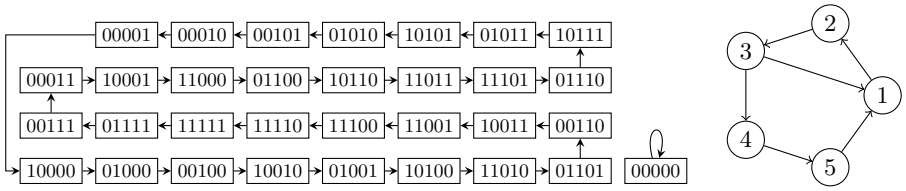


Fig. 2. Dynamics (left) and interaction graph (right) of the AN from Example 1.

5 Complexity of recognizing bounded-degree dynamics

Fix d and q, and consider the following decision problem called BDD (bounded-degree dynamics): given $F \in \mathcal{F}(n,q)$ represented by Boolean circuits, is there some $F' \in \mathcal{F}(n,q,d)$ such that $\mathcal{D}(F)$ and $\mathcal{D}(F')$ are isomorphic?

Theorem 3. *The problem BDD is in PSPACE for every d, q, and co-NP-hard for any $q \geq 2$ and $d \geq 1$.*

Proof. For the upper bound, a naive algorithm solving BDD consists in guessing $F' \in \mathcal{F}(n,q,d)$ (whose size is polynomial in F thanks to the bounded-degree condition) and checking that $\mathcal{D}(F)$ and $\mathcal{D}(F')$ are isomorphic. Given that planar graph isomorphism is computable with a LOGSPACE Turing machine M [**?**] and that $\mathcal{D}(F)$ and $\mathcal{D}(F')$ are at most exponentially larger than the input (Boolean circuit for F), we can test isomorphism of $\mathcal{D}(F)$ and $\mathcal{D}(F')$ in PSPACE by simulating each reading step of the read-only input tape of M by an evaluation of circuit in polynomial time (testing $F(x) = y$ is the same as testing the presence of the corresponding arc in $\mathcal{D}(F)$). This gives an algorithm in NP with an oracle in PSPACE, *i.e.*, an algorithm in the complexity class PSPACE.

For the co-NP-hardness we reduce from UNSAT. Given a propositional formula ϕ on p variables $v_1, \ldots, v_p$, we construct $F \in \mathcal{F}(n,q)$ on $|V| = p + d$

automata, with $P = \{v_1, \ldots, v_p\}$, $D = \{t_1, \ldots, t_d\}$ and $V = P \cup D$. Let $Q = \{0, \ldots, q-1\}$, and for $x \in Q^V$, consider the valuation $\theta(x_P)$ sending each 0 to false and other symbols to true. Set the local functions to be the identity $f_i(x) = x_i$ for every $i \in V \setminus \{t_d\}$, and:

$$f_{t_d}(x) = \begin{cases} x_{t_d} + 1 \mod q & \text{if } x_D = a^d \text{ and } \phi(\theta(x_P)), \\ x_{t_d} & \text{otherwise.} \end{cases}$$

If ϕ is unsatisfiable, then t_d depends only on D and F has degree d, hence it is a positive instance of BDD. Otherwise, F is not the identity, and it has:
- $(q^d - 1)q^p = q^n - q^{n-d}$ fixed points with $x_D \neq a^d$,
- at least one additional fixed point with $x_D = a^d$ and $\theta(x_P)$ satisfying ϕ.

Proposition 1 then implies that it is a negative instance of BDD. □

If we drop the isomorphism condition from the above problem, we get another one called BDIG (bounded-degree interaction graph): given $F \in \mathcal{F}(n, q)$ represented by Boolean circuits, is there some $F' \in \mathcal{F}(n, q, d)$ such that $\mathcal{D}(F) = \mathcal{D}(F')$? or, equivalently, is the degree of the interaction graph of F bounded by d?

Theorem 4. *The problem BDIG is co-NP-complete.*

Proof. The lower bound is given by the same reduction as in the proof of Theorem 3. For the upper bound, a simple co-NP algorithm consists in guessing an automaton $i \in V$, $d+1$ configurations $x^1, \ldots, x^{d+1}$, and $d+1$ distinct automata $i_1, \ldots, i_{d+1}$, then checking for each $j \in \{1, \ldots, d+1\}$ that $f_i(x^j) \neq f_i(x^j + e_{i_j})$. For each j, it checks whether x^j witnesses the effective dependency of i on automaton i_j. It is possible to guess $d+1$ such witnesses if and only if the interaction graph of F has degree at least $d+1$. □

6 Acknowledgments

This work was supported by ECOS-ANID project C19E02 between France and Chile, and ANR-18-CE40-0002 FANs. The authors thank Anahí Gajardo, Diego Maldonado, and Christopher Thraves, who participated in elaborating some ideas for the presented results. Figure 1 is built with Ti*k*Z library OODGraph.

Exhaustive Generation of Linear Orthogonal Cellular Automata*

Enrico Formenti[1][0000−0002−1007−7912] and Luca Mariot[2][0000−0003−3089−6517]

[1] Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S), Université Côte d'Azur, 2000, route des Lucioles - Les Algorithmes, bât. Euclide B, 06900 Sophia Antipolis, France
enrico.formenti@unice.it

[2] Semantics, Cybersecurity and Services Group, University of Twente, Drienerlolaan 5, 7522NB, Enschede, The Netherlands
l.mariot@utwente.nl

Abstract. We consider the problem of exhaustively visiting all pairs of linear cellular automata which give rise to orthogonal Latin squares, i.e., linear Orthogonal Cellular Automata (OCA). The problem is equivalent to enumerating all pairs of coprime polynomials over a finite field having the same degree and a nonzero constant term. While previous research as showed how to count all such pairs for a given degree and order of the finite field, no practical enumeration algorithms have been proposed so far. Here, we start closing this gap by addressing the case of polynomials defined over the field $\mathbb{F}_2$, which corresponds to binary CA. In particular, we exploit Benjamin and Bennett's bijection between coprime and non-coprime pairs of polynomials, which enables us to organize our study along three subproblems, namely the enumeration and count of: (1) sequences of constant terms, (2) sequences of degrees, and (3) sequences of intermediate terms. In the course of this investigation, we unveil interesting connections with algebraic language theory and combinatorics, by devising an enumeration algorithm and an alternative derivation of the counting formula for this problem.

Keywords: cellular automata, Latin squares, polynomials, finite fields, Euclid's algorithm, regular languages, compositions

1 Introduction

Orthogonal Cellular Automata (OCA) are pairs of CA whose global rules form orthogonal Latin squares, introduced by Mariot et al. at AUTOMATA 2016 [15]. As such, OCA have several practical applications in cryptography and coding theory, including the design of threshold secret sharing schemes [16, 18], primary constructions for partial spread bent functions [8, 9], correlation immune functions [19] and pseudorandom number generators with guaranteed diffusion

* This paper is a shortened version of [6], which is currently under submission at *Cryptography and Communications.*

properties [13, 14]. There is an algebraic characterization of OCA when the underlying local rules are linear, which relates the orthogonality of the resulting Latin squares to the coprimality of the polynomials induced by the rules [17]. Thus, the question of enumerating all pairs of linear OCA boils down to enumerating a specific kind of coprime polynomials over a finite field—namely, those pairs where both polynomials have the same degree and a nonzero constant term.

In general, coprime polynomials over finite fields have been studied in depth in the literature, also due to their numerous cryptographic and coding-theoretic applications [22, 3, 5]. Most of the related works in this area concern counting results with no restrictions on the constant terms nor the degree of the involved polynomials [4, 20, 1]. Benjamin and Bennett [1] described a simple bijection between coprime and non-coprime pairs for polynomials over the field $\mathbb{F}_2$. Such a bijection is proved by using Euclid's algorithm and its reversed application, dubbed dilcuE's algorithm. On the other hand, there are fewer works explicitly considering the exhaustive enumeration of coprime polynomial pairs [7].

In this paper, we show how to exhaustively generate all linear binary OCA by enumerating all coprime polynomial pairs over the finite field $\mathbb{F}_2$, where both polynomials have the same degree and a nonzero contant term. To this end, we take inspiration from Benjamin and Bennett's bijection and show that the sequences of quotients visited by dilcuE's algorithm for these coprime pairs are characterized by three independent components: the *constant terms*, the *degrees* and *intermediate terms*. From these results, we present the pseudocode of a combinatorial algorithm that generates all pairs of coprime polynomials of degree n and nonzero constant term. Further, we also provide an alternative derivation of the recurrence proved in [17] to count all such pairs for $q = 2$, which can be considered as an indirect proof of correctness of our enumeration algorithm.

2 Background Definitions

Cellular Automata (CA) are usually regarded as a parallel computational model or a particular kind of discrete dynamical systems. Here, on the other hand, we interpret CA as algebraic systems, specifically as a type of vectorial mappings composed of uniform local coordinate functions. Since we are interested only in binary CA, we can define them in terms of vectorial Boolean functions:

Definition 1. *A Cellular Automaton (CA) of length $m \in \mathbb{N}$, diameter $d \leq m$, and local rule $f : \mathbb{F}_2^d \mathbb{F}_2$ is a function $F : \mathbb{F}_2^m \to \mathbb{F}_2^{m-d+1}$ defined for all $x \in \mathbb{F}_2^m$ as:*

$$F(x_0, \ldots, x_{m-1}) = (f(x_0, \ldots, x_{d-1}), f(x_1, \ldots, x_d), \ldots, f(x_{m-d}, \ldots, x_{m-1})) \ . \tag{1}$$

Hence, each output coordinate $i \in \{0, \cdots, m-d\}$ of a CA F is is defined as the application of the local rule f to the neighborhood composed by the i-th input cell and the $d-1$ cells to its right.

Next, we give the formal definition of orthogonal Latin squares:

Definition 2. *Let* $[N] = \{1, \cdots, N-1\}$ *for all* $N \in \mathbb{N}$. *A Latin square of order* N *is a* $N \times N$ *matrix* L *with entries over* $[N]$ *such that each row and each column of* L *is a permutation of* $[N]$. *Two Latin squares* L_1, L_2 *of order* N *are called orthogonal if the function* $H : [N] \times [N] \to [N] \times [N]$ *defined for all* $(i, j) \in [N] \times [N]$ *as* $H(i, j) = (L_1(i, j), L_2(i, j))$ *is bijective.*

Intuitively, two Latin squares are orthogonal if and only if their *superposition* yields all pairs in the Cartesian product $[N] \times [N]$ exactly once.

Suppose now that $m = 2(d-1)$ and $F : \mathbb{F}_2^{2(d-1)} \to \mathbb{F}_2^{d-1}$ is a CA equipped with a *bipermutive* local rule, i.e. $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is of the form $f(x_0, \cdots, x_{d-1}) = x_0 \oplus g(x_1, \cdots, x_{d-2}) \oplus x_{d-1}$. In this case, Mariot et al. [15] showed that F defines a Latin square L_F of order $N = 2^{d-1}$. The idea is to encode blocks of $d-1$ bits in their decimal form. Then, the left and right $(d-1)$-bit blocks composing the input vector of F are used to index respectively the row and the column coordinates of L_F, while the output $(d-1)$-bit block of F represents the entry to be placed at those coordinates.

Let us further assume that the local rule is *linear*. This means that $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is defined as $f(x_0, \cdots, x_{d-1}) = x_0 \oplus a_1 x_1 \oplus \cdots \oplus a_{d-2} x_{d-2} \oplus x_{d-1}$, where $a_i \in \mathbb{F}_2$ for all $i \in \{1, \cdots, d-2\}$. In other words, f is an $\mathbb{F}_2$-linear combination of the neighborhood cells, with the property that the leftmost and righmost cells are always XORed, while $g : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ is any Boolean function over the $d-2$ central cells. In this case, a polynomial of degree $n = d-1$ in the ring $\mathbb{F}_2[X]$ can be naturally associated to the local rule:

$$f \mapsto P_f(X) = 1 + a_2 X + \cdots + a_{n-1} X^{n-1} + X^n \ . \tag{2}$$

Hence, we use the coefficients of the linear local rule to index the increasing powers of the indeterminate X in the polynomial P_f. The polynomial is thus monic of degree n and with a nonzero constant term.

In [15], the authors showed that two CA respectively defined by linear bipermutive local rules $f, g : \mathbb{F}_2^d \to \mathbb{F}_2$ give rise to a pair of orthogonal Latin squares if and only if their associated polynomials are relatively prime, i.e. if and only if $\gcd(P_f, P_g) = 1$. Therefore, the problem of counting and enumerating all pairs of linear Orthogonal Cellular Automata (OCA) of diameter d is equivalent to counting and enumerating all pairs of coprime polynomials of degree $n = d-1$ with a nonzero constant term. The counting question has already been settled in [17] for any finite field order q, where q is a power of a prime.

3 Problem Statement

In this paper, we focus on the enumeration problem for the binary case $q = 2$. To this end, we define the set S_n for all $n \in \mathbb{N}$ as:

$$S_n = \{f \in \mathbb{F}_2[x] : x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 : a_0 = 1\} \ , \tag{3}$$

that is, S_n is the set of binary polynomials of degree n with nonzero constant term. Further, let A_n and B_n be respectively defined for $n \in \mathbb{N}$ as:

$$A_n = \{(f, g) \in S_n^2 : \gcd(f, g) = 1\} \ , \ \ B_n = \{(f, g) \in S_n^2 : \gcd(f, g) \neq 1\} \ . \tag{4}$$

Thus, A_n and B_n are the sets of pairs of polynomials of degree n and nonzero constant terms that are respectively coprime and non-coprime. Clearly, it holds $A_n \cup B_n = S_n^2$ and $A_n \cap B_n = \varnothing$. We now formally state the problem addressed in this paper. Given $n \in \mathbb{N}$, we aim to:
(i) *Enumeration*: Find an algorithm to exhaustively generate all elements of A_n.
(ii) *Counting*: Derive a formula for $a_n = |A_n|$.
This problem has already been solved in [1]) for the case of generic polynomials of degree n (i.e. with unconstrained constant terms). The idea is to define a bijection between the sets of coprime and non-coprime pairs as follows:

1. Apply Euclid's algorithm to a non-coprime pair (f, g).
2. Change the last remainder from 0 to 1, and invert Euclid's algorithm (i.e. apply dilcuE's algorithm) using the same sequence of quotients computed for (f, g) in reverse order.
3. By construction, the pair (f', g') at the end of dilcuE's algorithm is coprime.

The crucial remark in the above procedure is that *the family of all sequences of quotients defines a bijection between coprime and non-coprime pairs.* The last remainder, either 1 or 0, defines whether the pair is respectively coprime or not.

Notice that if one employs this procedure starting from a non-coprime pair $(f, g) \in B_n$, in general the resulting coprime pair (f', g') will not belong to A_n, i.e. either f' or g' could have a null constant term. Nonetheless, since f' and g' are coprime, it cannot be the case the both of them have a null constant term, otherwise they would have a factor x in common. Therefore, we need to see how changing the last remainder in Euclid's algorithm affects the constant terms of the intermediate remainders, and thus the constant terms of f' and g'.

Let us start with the following remarks:

Remark 1. Let (f, g) be two polynomials of degree n. Then:
(i) The first quotient obtained from Euclid's algorithm is always $q_1 = 1$. Indeed, since f and g both have degree n, the long division stops immediately after dividing x^n by x^n.
(ii) Suppose that $\gcd(f, g) = 1$. Then, when the last pair of remainders is $(r_k(x), 1)$, if we apply Euclid's algorithm for one further step we will always obtain the pair $(1, 0)$ with quotient $r_k(x)$. In fact we can write the division of $r_k(x)$ and 1 as $r_k(x) = r_k(x) \cdot 1 + 0$.

Suppose that $q_1, q_2, \cdots, q_k$ is a sequence of quotients that yield a coprime pair $(f, g) \in A_n$ when applied in reverse order from $(1, 0)$ through dilcuE's algorithm. We denote these quotients as:

$$
\begin{array}{lcccc}
 & \overbrace{\phantom{x^{d_1}}}^{\text{degrees}} & \overbrace{\phantom{+ q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x +}}^{\text{intermediate terms}} & \overbrace{}^{\text{constant terms}} \\
q_1 \to & x^{d_1} & + q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x + & s_1 \\
q_2 \to & x^{d_2} & + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x + & s_2 \\
\vdots \to & \vdots & + \quad \vdots \quad + \cdots + \quad \vdots \quad + & \vdots \\
q_k \to & x^{d_k} & + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x + & s_k
\end{array}
$$

with $d_1, \cdots, d_k \in \mathbb{N}$ being the *degrees* of the quotients, $q_{i,j} \in \mathbb{F}_2$ the coefficients of the *intermediate terms*, and $s_i \in \mathbb{F}_2$ the *constant terms*.

Notice that a sequence of quotients is defined by independently choosing each of these three elements. Since our goal is to obtain a pair $(f, g) \in A_n$, the following two constraints hold:

- The sum of the degrees d_i equals n. This ensures that both polynomials have degree n, since the first quotient is equal to 1 by Remark 1(i).
- The sequence of constant terms is such that the constant terms of the two last remainders are respectively 1 and 0 (due to Remark 1(ii)), while the first two remainders (i.e., the reconstructed pair) must have constant term 1.

The intermediate terms, on the contrary, do not have any constraints and can be chosen freely. Thus, given the degree n and the length k of the quotients' sequence, enumerating the sequences of intermediate terms amount to enumerating all binary strings of length $n - k$, which are in total $I_{n,k} = 2^{n-k}$.

4 Constant Terms Sequences as a Regular Language

The generic step i of Euclid's algorithm applied to $(f, g) \in S_n$ corresponds to the Euclidean division:

$$r_i(x) = q_{i+1}(x) r_{i+1}(x) + r_{i+2}(x) \ , \tag{5}$$

where $r_i(x)$ and $r_{i+1}(x)$ are respectively the dividend and the divisor polynomial, $q_{i+1}(x)$ is the quotient, and $r_{i+2}(x)$ is the remainder of the division between $r_i(x)$ and $r_{i+1}(x)$. For $i = 1$, one has $r_1(x) = f(x)$ and $r_2(x) = g(x)$. Then, the process is repeated by shifting the divisor to become the dividend, whereas the remainder becomes the divisor.

We interpret the presence or the absence of the constant terms in r_i, r_{i+1} as the *state* of a discrete dynamical system, described by a pair (c_i, c_{i+1}) where $c_i, c_{i+1} \in \mathbb{F}_2$ respectively denote the constant terms of r_i and r_{i+1}. Remark that if $(f, g) \in S_n$ then $(c_i, c_{i+1}) \in (\mathbb{F}_2^2)^* = \{(1,1), (1,0), (0,1)\}$, since by Equation (5) $(c_i, c_{i+1}) = (0,0)$ for any i implies that f and g both have a null constant term.

The transition function $\delta : (\mathbb{F}_2^2)^* \times \mathbb{F}_2 \to (\mathbb{F}_2^2)^*$ maps a pair (c_i, c_{i+1}) and a constant term s_{i+1} of the $(i+1)$-th quotient to the next pair (c_{i+1}, c_{i+2}) using Equation (5). Figure 1 depicts the truth table and the transition graph of δ.

We now consider this dynamical system as a *Finite State Automaton* (FSA). The reason is that we can characterize the "correct" sequences of constant terms s_i (i.e., those that give a pair $(f', g') \in A_n$ at the end of dilcuE's algorithm) as the words of the language recognized by the FSA. Hence, we need to define the initial and accepting states of the automaton. Given $(f, g) \in S_n^2$, the sequence $q_1, q_2, \cdots$ of quotients computed through Euclid's algorithm induces a path on the FSA graph. This path starts from the state $(1, 1)$, and it is labelled by the constant terms $s_1, s_2, ...$ of the quotients.

Notice that the FSA is *permutative*: by taking two distinct states and reading the same constant term s_{i+1}, the two output states are distinct as well. This can be easily checked from the truth table of δ. A simple induction argument shows

(c_i, c_{i+1})	s_{i+1}	$\delta((c_i, c_{i+1}), s_{i+1})$
$(1,1)$	0	$(1,1)$
$(1,1)$	1	$(1,0)$
$(1,0)$	0	$(0,1)$
$(1,0)$	1	$(0,1)$
$(0,1)$	0	$(1,0)$
$(0,1)$	1	$(1,1)$

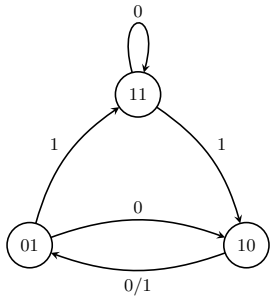


Fig. 1: Transition table and graph realizing δ.

that this property holds also for sequences of constant terms. Hence, if one starts from two different initial states and apply the same sequence of constant terms, the final states are also different. The inverse automaton, which corresponds to dilcuE's algorithm, is thus obtained by simply inverting the arrows in the transition graph of the original FSA. The initial state of this inverse automaton will be $(1,0)$, on account of Remark 1. For the accepting state, we should select intuitively $(1,1)$, since we want a pair of A_n at the end of dilcuE's algorithm. Notice however that the first quotient in Euclid's algorithm (therefore, the last one in dilcuE's) is always 1, due to Remark 1(i). Hence, we can shorten the sequence of quotients by one element, and append 1 to it. Consequently, since the only way to reach $(1,1)$ in the inverse FSA by reading a 1 is from $(1,0)$, we can define $(1,0)$ also as the only accepting state.

The classic state elimination method [10] gives us the following regular expression for the language recognized by the inverse FSA:

$$L_r = (0(0+1) + (10^*1(0+1)))^* \ . \tag{6}$$

We have thus obtained the following result:

Lemma 1. *The sequences of constant terms for the quotients visited by dilcuE's algorithm when generating a coprime pair $(f,g) \in A_n$ form a regular language L_r, whose regular expression is defined by Equation* (6).

Hence, enumerating the sequences $s_1, \cdots, s_k$ is equivalent to generating all words of length k in L_r. Several algorithms are available for this task, see e.g. [12]. To count the number of words, we apply the *Chomsky-Schützenberger enumeration theorem* [2], which uniquely identifies a regular language L with a rational Formal Power Series (FPS) $\mathcal{F}_L = \sum_{k=0}^{\infty} \ell_k X^k$. This leads us to the following:

Lemma 2. *The generating function $G(X)$ for the FPS associated to L_r and the recurrence for the number ℓ_k of words of length $k \in \mathbb{N}$ in L_r are given by:*

$$G(X) = \frac{1-X}{1-X-2X^2} \ , \ \ell_k = \frac{2^k + 2\cdot(-1)^k}{3} \ . \tag{7}$$

Proof. Omitted due to limited space (see [6]). □

5 Quotients' Degrees Sequences and Compositions

The only constraint enforced on the sequences of quotients' degrees $d_1, \cdots, d_k$ is that they must sum to the final degree n. The order of the summands is relevant: permuting the degrees gives rise to a different sequence of quotients. Hence, we want to enumerate and count the number of ways in which n is obtained as an ordered sum of k natural numbers. These are also known as *k-compositions* of $n \in \mathbb{N}$ in combinatorics [21], and a simple way to represent them by means of $n-1$ *boxes* interleaved by n occurrences of 1:

$$1 \overbrace{\square 1 \square \ldots \square 1 \square}^{n-1} 1 \ ,$$

where each box can be either a comma (,) or a plus (+). A comma separates two different parts in a composition, while a plus adds two adjacent 1s together. Notice that we cannot take the composition where all boxes are set to +, since a sequence composed of just one quotient cannot occur in dilcuE's algorithm.

Once the length k of the sequence of quotients is fixed, generating the corresponding degrees is equivalent to the enumeration of all binary strings of length $n-1$ with $k-1$ ones in them, which can be accomplished, for instance, by one of the several algorithms described by Knuth [11]. Further, the number of all compositions of n of length k is given by the binomial coefficient $\binom{n-1}{k-1}$. Therefore, we obtained the following result:

Lemma 3. *The number of sequences of degrees $d_1, \cdots, d_k$ of the final degree $n \in \mathbb{N}$ for the quotients visited by dilcuE's algorithm is:*

$$D_{n,k} = \binom{n-1}{k-1} \ . \tag{8}$$

Putting together the results presented above, the following pseudocode exhaustively enumerates all pairs of coprime polynomials in A_n for a given input degree n and quotients' sequence length $2 \leq k \leq n$:

- For each composition *comp* of n of length k do:
  - (1) Generate the degrees' sequence *deg* corresponding to *comp*
  - (2) For each intermediate terms sequence *seq* do:
    - (2a) Adjoin *seq* to *deg* to get a quotients's sequence *quot*
    - (2b) For each constant term sequence *const* of length k do:
      - Adjoin *const* to *quot*
      - Apply DilcuE's algorithm from $(1, 0)$ by using the sequence *quot*

Finally, the Lemma below provides a different derivation for the number $a_n = |A_n|$ of coprime polynomials of degree n and with a nonzero constant term.

Lemma 4. *The number of pairs of coprime polynomials of degree n with nonzero constant term is equal to:*

$$a_n = \sum_{k=2}^{n} 2^{n-k} \cdot \binom{n-1}{k-1} \cdot \frac{2^k + 2 \cdot (-1)^k}{3} = 2 \cdot \frac{4^{n-1} - 1}{3} \quad . \tag{9}$$

Proof. Omitted due to limited space (see [6]). □

References

1. Benjamin, A.T., Bennett, C.D.: The probability of relatively prime polynomials. Mathematics Magazine **80**(3), 196–202 (2007)
2. Chomsky, N., Schützenberger, M.P.: The algebraic theory of context-free languages. In: Studies in Logic and the Foundations of Mathematics, vol. 26, pp. 118–161. Elsevier (1959)
3. Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Trans. Inf. Theory **30**(4), 587–593 (1984)
4. Corteel, S., Savage, C.D., Wilf, H.S., Zeilberger, D.: A pentagonal number sieve. J. Comb. Theory, Ser. A **82**(2), 186–192 (1998)
5. Fitzpatrick, P.: On the key equation. IEEE Trans. Inf. Theory **41**(5), 1290–1302 (1995)
6. Formenti, E., Mariot, L.: An enumeration algorithm for binary coprime polynomials with nonzero constant term. CoRR **abs/2207.00406** (2022)
7. Fragneto, P., Rimoldi, A., Sala, M.: An approach to create coprime polynomial pairs (2005)
8. Gadouleau, M., Mariot, L., Picek, S.: Bent functions from cellular automata. IACR Cryptol. ePrint Arch. p. 1272 (2020)
9. Gadouleau, M., Mariot, L., Picek, S.: Bent functions in the partial spread class generated by linear recurring sequences. Des. Codes Cryptogr. **91**(1), 63–82 (2023)
10. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation, 3rd Edition. Pearson international edition, Addison-Wesley (2007)
11. Knuth, D.: The art of computer programming, vol. 4, pre-fascicle 3a (2011)
12. Mäkinen, E.: On lexicographic enumeration of regular and context-free languages. Acta Cybern. **13**(1), 55–61 (1997)
13. Mariot, L.: Hip to be (latin) square: Maximal period sequences from orthogonal cellular automata. In: Ninth International Symposium on Computing and Networking, CANDAR 2021, Matsue, Japan, November 23-26, 2021. pp. 29–37. IEEE (2021)
14. Mariot, L.: Enumeration of maximal cycles generated by orthogonal cellular automata. Natural Computing pp. 1–15 (2022)
15. Mariot, L., Formenti, E., Leporati, A.: Constructing orthogonal latin squares from linear cellular automata. CoRR **abs/1610.00139** (2016)
16. Mariot, L., Formenti, E., Leporati, A.: Enumerating orthogonal latin squares generated by bipermutive cellular automata. In: Dennunzio, A., Formenti, E., Manzoni, L., Porreca, A.E. (eds.) Cellular Automata and Discrete Complex Systems - 23rd IFIP WG 1.5 International Workshop, AUTOMATA 2017, Milan, Italy, June 7-9, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10248, pp. 151–164. Springer (2017)

17. Mariot, L., Gadouleau, M., Formenti, E., Leporati, A.: Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. **88**(2), 391–411 (2020)
18. Mariot, L., Leporati, A.: Inversion of mutually orthogonal cellular automata. In: Mauri, G., Yacoubi, S.E., Dennunzio, A., Nishinari, K., Manzoni, L. (eds.) Cellular Automata - 13th International Conference on Cellular Automata for Research and Industry, ACRI 2018, Como, Italy, September 17-21, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11115, pp. 364–376. Springer (2018)
19. Mariot, L., Manzoni, L.: On the linear components space of s-boxes generated by orthogonal cellular automata. In: Bastien Chopard, e.a. (ed.) ACRI 2022, Proceedings. Lecture Notes in Computer Science, vol. 13402, pp. 52–62. Springer (2022)
20. Reifegerste, A.: On an involution concerning pairs of polynomials over $_{\mathrm{f2}}$. J. Comb. Theory, Ser. A **90**(1), 216–220 (2000)
21. Riordan, J.: Introduction to combinatorial analysis. Courier Corporation (2012)
22. Shparlinski, I.: Finite Fields: Theory and Computation: The meeting point of number theory, computer science, coding theory and cryptography, vol. 477. Springer Science & Business Media (2013)