Jarkko Kari (Ed.)

Cellular Automata and Discrete Complex Systems

LNCS 9099

21st IFIP WG 1.5 International Workshop, AUTOMATA 2015
Turku, Finland, June 8–10, 2015
Proceedings

ifip

Springer

Lecture Notes in Computer Science **9099**

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

More information about this series at http://www.springer.com/series/7407

Jarkko Kari (Ed.)

Cellular Automata and Discrete Complex Systems

21st IFIP WG 1.5 International Workshop, AUTOMATA 2015
Turku, Finland, June 8–10, 2015
Proceedings

Springer

Editor
Jarkko Kari
University of Turku
Turku
Finland

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-662-47220-0 ISBN 978-3-662-47221-7 (eBook)
DOI 10.1007/978-3-662-47221-7

Library of Congress Control Number: 2015938087

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

Preface

The 21st International Workshop on Cellular Automata and Discrete Complex Systems, AUTOMATA 2015, was held in Turku, Finland, during June 8–10, 2015. It was organized by the Department of Mathematics and Statistics of the University of Turku, and the conference venue was the Educarium building of the university. The event was an IFIP Working Conference and it hosted a meeting of the IFIP Working Group 1.5.

AUTOMATA 2015 continued an annual series of events established in 1995 as a forum for the collaboration of researchers in the field of cellular automata and related discrete complex systems. Topics of interest include, for example, the following aspects and features of such systems: dynamical, topological, ergodic, and algebraic aspects, algorithmic and complexity issues, emergent properties, formal language processing aspects, symbolic dynamics, models of parallelism and distributed systems, timing schemes, phenomenological descriptions, scientific modeling and practical applications. The conference attracted a good number of submissions, which indicates a continued interest in the topics.

There were four invited talks in the conference, and I wish to thank the speakers Andreas Deutsch, Turlough Neary, Ville Salo, and Luke Schaeffer for accepting the invitation and for their presentations. The invited contributions are included in this volume.

There were 33 submissions in the conference. Each submission was reviewed by three Program Committee members. Based on the reviews and discussions the committee decided to accept 15 papers to be presented in the conference and to be included in the proceedings. I would like to thank all authors for their contributions. The conference program involved also short presentations of exploratory papers that are not included in this book, and I wish to extend my thanks to the authors of the exploratory submissions.

I am indebted to the Program Committee and the additional reviewers for their help in selecting the papers. I extend my thanks to the members of the Local Organizing Committee. I am also grateful for the support by the Federation of Finnish Learned Societies, Turku Centre for Computer Science, the University of Turku, and the City of Turku. Finally, I acknowledge the excellent cooperation from the Lecture Notes in Computer Science team of Springer for their help in producing this volume in time for the conference.

March 2015 Jarkko Kari

Organization

Program Committee

Matthew Cook	University of Zurich and ETH Zurich, Switzerland
Pedro de Oliveira	Universidade Presbiteriana Mackenzie, Brazil
Nazim Fatès	Inria Nancy Grand-Est, France
Enrico Formenti	Université Nice Sophia Antipolis, France
Eric Goles	Adolfo Ibáñez University, Chile
Jarkko Kari	University of Turku, Finland
Martin Kutrib	Universität Giessen, Germany
Andreas Malcher	Universität Giessen, Germany
Kenichi Morita	Hiroshima University, Japan
Nicolas Ollinger	Université d'Orléans, France
Ivan Rapaport	Universidad de Chile, Chile
Klaus Sutner	Carnegie Mellon University, USA
Véronique Terrier	Université de Caen, France
Guillaume Theyssier	CNRS, Université de Savoie, France
Hiroshi Umeo	Osaka Electro-Communication University, Japan
Thomas Worsch	Karlsruhe Institute of Technology, Germany

Organizing Committee

Mikhail Barash	University of Turku, Finland
Jarkko Kari	University of Turku, Finland
Michal Szabados	University of Turku, Finland
Ilkka Törmä	University of Turku, Finland
Sonja Vanto	University of Turku, Finland

Additional Reviewers

Aubrun, Nathalie
Barbieri, Sebastián
Boyer, Laurent
Capobianco, Silvio
Chassaing, Philippe
Das, Sukanta
Delacourt, Martin
Deutsch, Andreas
Fukś, Henryk
Grandjean, Anaël
Hellouin De Menibus, Benjamin
Holzer, Markus
Inokuchi, Shuichi
Jakobi, Sebastian
Landman, Kerry
Lee, Jia
Leporati, Alberto
Manzoni, Luca
Marcovici, Irène
Meckel, Katja

Meunier, Pierre-Étienne
Namiki, Takao
Richard, Gaétan
Romashchencko, Andrei
Salomaa, Kai
Sirakoulis, Georgios Ch.
Spencer, Jason
Szabados, Michal
Taati, Siamak
Truthe, Bianca
Törmä, Ilkka
Vanier, Pascal
Wendlandt, Matthias
Yunès, Jean-Baptiste
Zinoviadis, Charalampos

Contents

Cellular Automaton Models for Collective Cell Behaviour

Andreas Deutsch(✉)

Cente for Information Services and High Performance Computing,
Technische Universität Dresden, Dresden, Germany
andreas.deutsch@tu-dresden.de

1 Introduction

Biological organisms are complex systems characterized by collective behaviour emerging out of the interaction of a large number of components (molecules and cells). In complex systems, even if the basic and local interactions are perfectly known, it is possible that the global (collective) behaviour can not be obviously extrapolated from the individual properties. Collective dynamics of migrating and interacting cell populations drive key processes in tissue formation and maintenance under normal and diseased conditions. For revealing the principles of tissue organization, it is fundamental to analyze the tissue-scale consequences of intercellular interaction [11,22]. Only an understanding of the dynamics of collective effects at the molecular and cellular scale allows answering biological key questions such as: what enables ensembles of molecules to organize themselves into cells? How do ensembles of cells create tissues and whole organisms? What is different in diseased tissues as malignant tumors? Mathematical models for spatio-temporal pattern formation can contribute to answer these questions. The first models of spatio-temporal pattern formation focused on the dynamics of diffusible morphogen signals and have been formulated as partial differential equations (e.g. [30]). In addition to diffusible molecular signals, the role of cells in morphogenesis can not be neglected. Living cells possess migration strategies that go beyond the merely random displacements of non-living molecules (diffusion) [22]. More and more evidence exists about how the self-organization of interacting and migrating cells contributes to the formation of order in a developing organism. Thereby, both the particular type of cell interaction and migration are crucial and suitable combinations allow for a wide range of patterns. The question is: What are appropriate mathematical models for analyzing organization principles of moving and interacting cells cells?

Cellular automata (CA), in particular lattice-gas cellular automata (LGCA) can model the interplay of cells with themselves and their heterogeneous environment [17]. These models describe interactions at a cell-based local scale. Cell-based models (for a review see [2,20]) are required if one is attempting to extract the organization principles of interacting cell systems down to length scales of the order of a cell diameter to link the individual (microscopic) cell dynamics with a particular collective (macroscopic) phenomenon.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 1–10, 2015.
DOI: 10.1007/978-3-662-47221-7_1

2 Cellular Automata and Lattice-Gas Cellular Automata

Cellular automaton models are a spatio-temporal modelling formalism and have been introduced by J. v. Neumann and S. Ulam in the 1950s as a model of individual (self-)reproduction [12]. They consist of a regular spatial grid in which each grid point (or site) can have a finite, typically small number of discrete states. The next state of a site depends on the states in the neighboring sites and a next-state function, which can be deterministic or stochastic. Cellular automata provide simple models of self-organizing complex systems in which collective behaviour can emerge out of an ensemble of many interacting "simple" components - being it molecules, cells or organisms [15,16,41].

Here, we suggest lattice-gas cellular automata which can be viewed as models for collective behaviour emerging from microscopic migration and interaction processes. LGCA can be used to model the interplay of cells with each other and with their heterogeneous environment by describing interactions at a cell-based (microscopic) scale and facilitating both efficient simulation and theoretical analysis of emergent, tissue-scale (macroscopic) parameters [17]. Historically, LGCA have been introduced as models of gas and fluid flows, through implementing simplistic local collisions. Often, the overall macroscopic behaviour of the system can be approximated very well if averages over larger spatial scales are considered [23,40]. In a biological context, LGCA particles are interpreted as cells and cell migration is modeled by updating cell positions at each time step based on local cell interactions. Local cell interactions are described by problem-specific LGCA transition rules. These transition rules are different from the rules that have been used for modelling fluid flows. LGCA transition rules in models of cell migration, in general, do not assume energy or momentum conservation. Biological LGCA models can be classified as stochastic cellular automata with time-discrete, synchronous updates consisting of stochastic interaction and subsequent deterministic migration steps. Implementing movement of individuals in traditional synchronous-update cellular automaton models is not straightforward, as one site in a lattice can typically only contain one individual, and consequently movement of individuals can cause collisions when two individuals want to move to the same empty site. In a lattice-gas model this problem is avoided by having separate channels for each direction of movement and imposing an exclusion principle. In addition, rest channels can be added for non-moving cells. The deterministic movement steps are alternated with stochastic interaction steps, in which processes affecting cell number, e.g., birth and death can be implemented (Fig. 1).

A major advantage of LGCA models compared to other cell-based models for interacting cell systems, such as interacting particle systems, e.g. [29,39], asynchronous cellular automata, e.g. [4–6], further agent-based models [24] or systems of stochastic differential equations [37], is their computational efficiency.

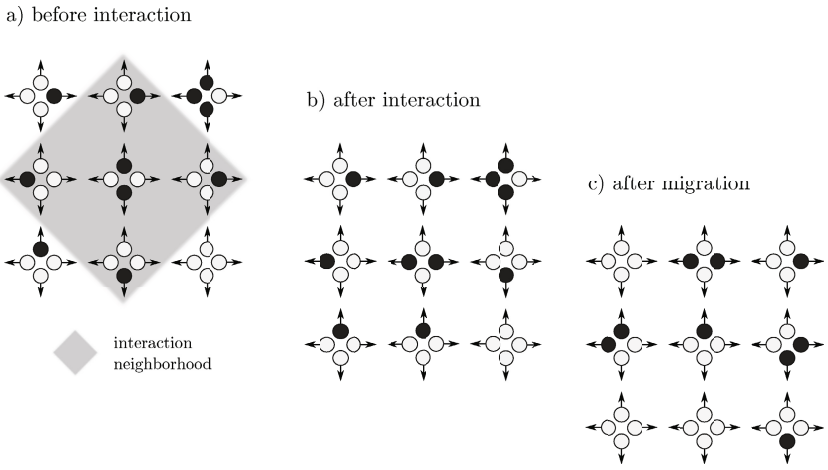


Fig. 1. Interaction and migration in a "biological LGCA": Filled circles denote occupied cells and open circles empty cells. a) part of the lattice before interaction, interation neighborhood is shown. b) after interaction: the configuration of the cells at several nodes has changed. c) after migration: each cell has moved to the nearest neighbor corresponding to its orientation indicated by arrows (the lattice outside the part shown is assumed to be empty, i.e. there is no migration of cells from outside).

3 LGCA Modelling and Analysis

The LGCA idea has led to models of morphogenetic motion describing migration of individual cells during spatio-temporal pattern formation in microorganisms, cell cultures and developing organisms [7,17,35][1]. The essential modelling idea is the definition of appropriate transition probabilities characterizing specific cell interactions. In particular, morphogenetic cell motion may be influenced by the interaction of cells with components of their immediate local surrounding through haptotaxis or differential adhesion, interaction with the basal lamina and the extracellular matrix, contact guidance, contact inhibition, and processes that involve cellular responses to signals that are propagated over larger distances (e.g. chemotaxis). LGCA models have also been used to study emergent collective behaviour in cell swarming [14], angiogenesis [32] and Turing pattern formation [19] (Fig. 2). Moreover, there are methods for parameter estimation in LGCA models [31].

Deciphering the principles of collective cell behaviour is especially important for a better understanding of tumour growth and invasion and might allow to develop new therapy concepts. Besides more and more molecular investigations, mathematical modelling of selected aspects of tumour growth has become attractive within the last years (e.g. [1,34,36]). Cellular automaton models have been

[1] A simulation platform containing various LGCA models with biological motivations can be found at www.biomodelling.info.

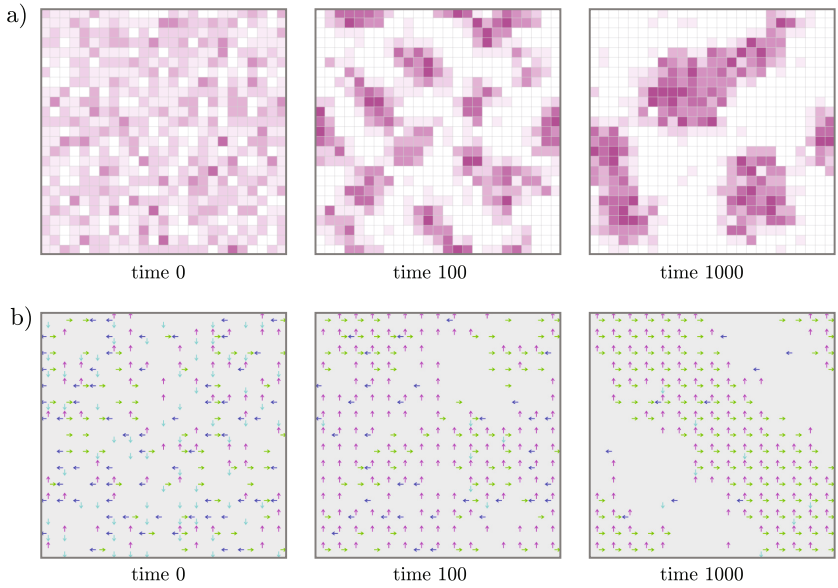


Fig. 2. Spatio-temporal pattern formation in simulations (from a random initial cell population) for a) LGCA model of adhesive interaction, b) LGCA model of alignment interaction (see [14] and [17] for details). Colours indicate different cell numbers (a) and different cell directions (b).

suggested for various aspects of tumour growth [34]. In particular, simulations and analysis of appropriate LGCA models permit to characterize different growth and invasion scenarios [9,18,28] (Fig. 3).

For a number of LGCA models a corresponding lattice-Boltzmann approximation can be adopted to analyze the emergence of spatio-temporal patterns [17]. The idea of the lattice-Boltzmann (mean-field) approximation is the reduction of the description of a system with many interacting individuals (many degrees of freedom), such as the CA, to the level of an effective, average description for the behaviour of a single individual (low degree of freedom). The application of the mean-field approximation allows for the transition from a microscopic to a macroscopic description of the CA dynamics. The central step is the derivation of a spatio-temporal mean-field approximation of the stochastic automaton process. Disregarding the spatial aspect completely leadd to qualitatively wrong model predictions (Fig. 4). Based on a spatio-temporal mean-field description of the microscopic process, one can often calculate a corresponding macroscopic partial differential equation by means of a Chapman-Enskog expansion technique [16]. For example, from the mean-field PDE for a proliferation process one can derive important macroscopic observables of biological growth, such as total number of particles, per capita growth rate and invasion speed, and reveal their dependence on the microscopic growth and migration parameters [28].

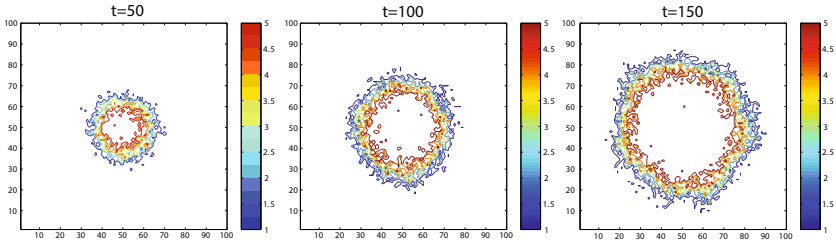


Fig. 3. Simulation of the spatio-temporal development of a LGCA growth model starting from a localized initial cell population in the centre of the lattice. The three pictures show simulation snapshots for subsequent time steps. Contour plot where colors indicate the cell density (from [27] with permission).

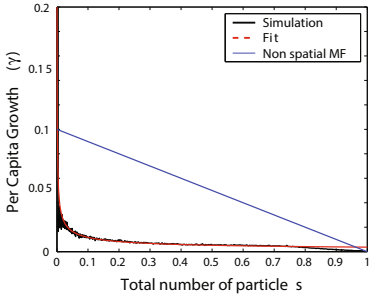


Fig. 4. LGCA growth model: evolution of the per capita growth rate as a function of the total population density. Comparison of simulations, a spatial mean-field approximation (fit) and a non-spatial mean-field approximation. The non-spatial mean-field approximation completely fails to describe the LGCA dynamics (from [27] with permission).

Typical examples for observables to study emergent collective behaviour are cell density patterns and related quantities such as the dynamics of moving cell fronts and cluster size distributions [8,26,32]. Cell density patterns can often be assessed experimentally and thus provide a means to relate LGCA model predictions to experimental observation. However, there are further experimental observables - particularly the trajectories of individual, selected cells - that characterize emergent collective behaviour but which are not directly deducible from cell density patterns. Classical LGCA, however, do not distinguish individual cells. We have developed an approach to overcome this limitation and which allows to translate classical LGCA into models where individual cells can be tracked (Fig. 5). This is achieved by modifying the state space and the transition rules such that a cell identity can be transferred [33].

Based on these individual-based LGCA the establishment of a connection between the LGCA transition rules and a stochastic differential equation (SDE) description for the trajectories of single cells becomes possible. Comparable

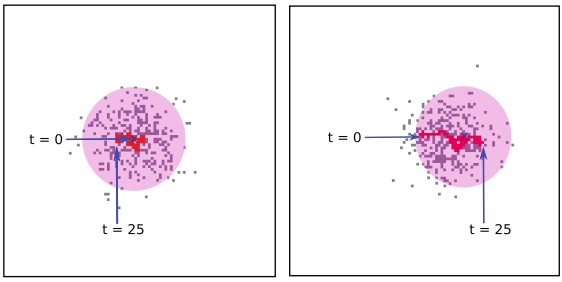


Fig. 5. LGCA simulations of single cell migration in a heterogeneous environment: Trajectories of selected labeled cells are shown in red. The positions of the labeled cells for $t = 0$ and $t = 25$ are marked by blue arrows. Simulation of the LGCA model for different sensitivities to the environment $\beta = 0$ (A) and $\beta = 0.5$ (B) (from [33] with permission).

approaches to derive equations for tagged particles in cellular automaton models [3,25] are only applicable to equilibrium systems, i.e. where the cellular environment is in a steady state. Our approach allows to tackle LGCA models for non-equilibrium systems, i.e. the cellular environment can evolve over time [33]. For fairly general, cell number conserving LGCA transition rules, we have shown that the trajectories of individual cells can be described by an SDE whose parameters can be calculated directly from the LGCA transition rules. Cell number conserving LGCA transition rules are applicable when cell migration is on a much faster timescale than cell birth and death. The SDE description is derived in the limit when lattice spacing and time step length tend to zero under diffusive scaling. We have shown, for a variety of concrete models, that there is good agreement between LGCA simulations and the SDE approximation (Fig. 6). Individual-based LGCA facilitate individual cell trajectory analysis while still allowing efficient simulations, thus opening up novel possibilities to investigate LGCA behaviour and to compare model and experiment at the individual cell level. Notice that the SDE approximation can be extended to other cell-based models such as interacting particle systems and general stochastic CA.

4 Outlook

The mean-field (Boltzmann) equation characterizing a given LGCA model arises under the assumption that the probability of finding two cells at specific positions is given by the product of corresponding single particle distribution functions, i.e. any correlations are neglected and distributions fully factorize. It is a challenge to include two-, three-, etc. particle distribution functions which will allow a systematic study of correlation effects. If pair correlations are taken into account, but third and higher order correlations are neglected, a generalized Boltzmann equation for the single particle distribution function is obtained, coupled to the so-called ring equation describing the evolution of the pair correlation function.

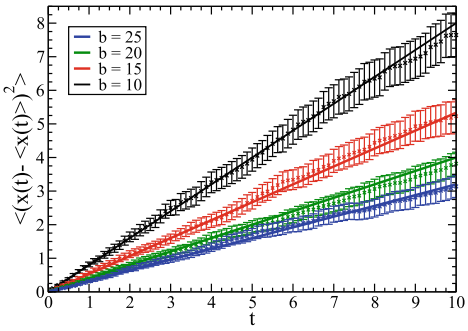


Fig. 6. Symmetric random walk in the low density regime. Simulations were carried out for a single cell on a two-dimensional square lattice lattice. Distribution $f(x)$ of the cell position of a single cell at time $t = 10$. Time development of the mean square displacement of an individual cell. Simulation data (symbols) and corresponding solutions (lines) of stochastic differential equation for different numbers of rest channels are shown. All solutions were obtained for deterministic initial data. The simulation data were averaged over 500 independent simulations for each case. LGCA simulations and the corresponding SDE approximations show good agreement for the mean square displacement (from [33] with permission).

For the adhesive (density-dependent) cellular automaton the ring equation has been successfully evaluated [13]. It is a challenge to determine corresponding equations for other cellular automata. This analysis could particularly improve our understanding of short and long time behaviour.

The need for discrete models, especially cellular automata, goes beyond the analysis of collective behaviour in interacting cell populations. A discrete cell-oriented approach is also required if the dynamic system behaviour depends on fluctuations at the individual cell level. This is, for example, the case at the front of invading tumours and crucial for the formation of metastases. Experimental findings of Bru et al. [10] indicate that many tumours share the same surface dynamics. This finding motivated the analysis of the tumour interface by means of a fractal scaling analysis [21].

In order to represent more detail at the individual cell level, multiscale models have been developed. For example, the user-friendly simulation platform Morpheus integrates cell-based models, ordinary differential equations for subcellular dynamics and reaction-diffusion systems for the extracellular environment [38]. While there exists a rich theory on ordinary and partial differential equations, the theory for cell-based models, in particular cellular automata is comparably young. Based on the variability in the local dynamics, we demonstrated that the "interaction-modul oriented" cellular automaton modelling provides an intuitive and powerful approach to capture essential aspects of collective cellular dynamics [17]. In conclusion, there are both challenging future perspectives with regards to interesting biological applications of the lattice-gas cellular automaton idea and possible refinements of analytical tools for the investigation of lattice-gas

cellular automata. Hopefully, the potential of cellular automata for modelling essential aspects of biological systems will be further exploited in the future.

Acknowledgments. AD acknowledges the support of the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" and thanks H. Hatzikirou, C. Mente, R. Müller and J. Starruss (Dresden) for comments and discussions.

References

1. Anderson, A.R.A., Quaranta, V.: Integrative mathematical oncology. Nature Rev. Cancer **8**, 227–234 (2008)
2. Anderson, A., Chaplain, M., Rejniak, K. (eds.): Single Cell-Based Models in Biology and Medicine. Birkhauser, Basel (2007)
3. Arratia, R.: The motion of a tagged particle in the simple symmetric exclusion system on Z. Ann. Prob. **11**, 362–373 (1983)
4. Badoual, M., Deroulers, C., Aubert, M., Grammaticos, B.: Modelling intercellular communication and its effects on tumour invasion. Phys. Biol. **7**(4), 046013 (2010)
5. Binder, B.J., Landman, K.A., Newgreen, D.F., Simkin, J.E., Takahashi, Y., Zhang, D.: Spatial analysis of multi-species exclusion processes: application to neural crest cell migration in the embryonic gut. Bull. Math. Biol. **74**(2), 474–490 (2012)
6. Bloomfield, J.M., Sherratt, J.A., Painter, K.J., Landini, G.: Cellular automata and integro-differential equation models for cell renewal in mosaic tissues. J. R. Soc. Interface **7**(52), 1525–1535 (2010)
7. Börner, U., Deutsch, A., Bär, M.: A generalized discrete model linking rippling pattern formation and individual cell reversal statistics in colonies of myxobacteria. Phys. Biol. **3**(2), 138–146 (2006)
8. Böttger, K., Hatzikirou, H., Chauviere, A., Deutsch, A.: Investigation of the migration/proliferation dichotomy and its impact on avascular glioma invasion. Math. Model. Nat. Phenom. **7**(1), 105–135 (2012)
9. Böttger, K., Hatzikirou, H., Voss-Böhme, A., Cavalcanti-Adam, E.A., Herrero, M.A., Deutsch, A.: Emerging Allee effect in tumor growth. Plos. Comp. Bio., in press (2015)
10. Brú, A., Albertos, S., Subiza, J.L., López García-Asenjo, J.A., Brú, I.: The universal dynamics of tumor growth. Biophys. J. **85**, 2648–2961 (2003)
11. Bryant, D.M., Mostov, K.E.: From cells to organs: building polarized tissue. Nature Rev. Mol. Cell Biol. **9**, 887–901 (2008)
12. Burks, A.W.: Essays on Cellular Automata. University of Illinois Press, Urbana IL (1970)
13. Bussemaker, H.: Analysis of a pattern-forming lattice-gas automaton: mean field theory and beyond. Phys. Rev. E **53**, 1644–1661 (1996)
14. Bussemaker, H.J., Deutsch, A., Geigant, E.: Mean-field analysis of a dynamical phase transition in a cellular automaton model for collective motion. Phys. Rev. Lett. **78**, 5018–5021 (1997)
15. Casti, J.L.: Alternate realities. John Wiley, New York (1989)
16. Chopard, B., Droz, M.: Cellular Automata Modeling of Physical Systems. Cambridge University Press, New York (1998)
17. Deutsch, A., Dormann, S.: Cellular Automaton Modeling of Biological Pattern Formation: Characterization, Applications, and Analysis. Birkhauser, Boston (2005). (2nd ed 2015)

18. Dormann, S., Deutsch, A.: Modeling of self-organized avascular tumor growth with a hybrid cellular automaton. Silico Biol. **2**(3), 393–406 (2002)
19. Dormann, S., Deutsch, A., Lawniczak, A.T.: Fourier analysis of Turing-like pattern formation in cellular automaton models. Future Gener. Comp. Sy. **17**(7), 901–909 (2001)
20. Drasdo, D.: On selected individual-based approaches to the dynamics in multicellular systems. In: Alt, W., Chaplain, M., Griebel, M., Lenz, J. (eds.) Models of Polymer and Cell Dynamics. Birkhäuser, Basel (2003)
21. Franciscis, S., Hatzikirou, H., Deutsch, A.: Analysis of lattice-gas models for tumor growth by means of fractal scaling. Acta Phys. Pol. B. Proc. Suppl. **2**(4), 167 (2011)
22. Friedl, P., Gilmour, D.: Collective cell migration in morphogenesis, regeneration and cancer. Nature Rev. Mol. Cell Biol. **10**, 445–457 (2009)
23. Frisch, U., Hasslacher, B., Pomeau, Y.: Lattice-gas automata for the Navier-Stokes equation. Phys. Rev. Lett. **56**(14), 1505–1508 (1986)
24. Galle, J., Hoffmann, M., Aust, G.: From single cells to tissue architecturea bottom-up approach to modelling the spatio-temporal organisation of complex multicellular systems. J. Math. Biol. **58**, 261–283 (2009)
25. Harris, T.E.: Diffusion with collisions between particles. J. App. Prob. **2**, 323–338 (1965)
26. Hatzikirou, H., Basanta, D., Simon, M., Schaller, K., Deutsch, A.: Go or grow: the key to the emergence of invasion in tumour progression? Math. Med. Biol. **29**(1), 49–65 (2006)
27. Hatzikirou, H., Brusch, L., Schaller, C., Simon, M., Deutsch, A.: Prediction of traveling front behavior in a lattice-gas cellular automaton model for tumor invasion. Comput. Math. Appl. **59**(7), 2326–2339 (2010)
28. Hatzikirou, H., Deutsch, A.: Cellular automaton modelling of tumor invasion. In: Meyers, R. (ed.) Encyclopedia of Complexity and Systems Science. Springer, New York (2009)
29. Liggett, T.M.: Interacting particle systems. Springer (1985)
30. Meinhardt, H.: Models of Biological Pattern Formation. Academic Press, London (1982)
31. Mente, C., Prade, I., Brusch, L., Breier, G., Deutsch, A.: Parameter estimation with a novel gradient-based optimization method for biological lattice-gas cellular automaton models. J. Math. Biol. **63**(1), 173–200 (2010)
32. Mente, C., Prade, I., Brusch, L., Breier, G., Deutsch, A.: A lattice-gas cellular automaton model for in vitro sprouting angiogenesis. Acta Phys. Pol. B **5**(1), 99–115 (2012)
33. Mente, C., Voss-Böhme, A., Deutsch, A.: Analysis of individual cell trajectories in lattice-gas cellular automaton models for migrating cell populations. Bull. Math. Biol. (2015, to appear)
34. Moreira, J., Deutsch, A.: Cellular automaton models of tumour development - a critical review. Adv. Compl Syst. (ACS) **5**(2), 1–21 (2002)
35. Moreira, J., Deutsch, A.: Pigment pattern formation in zebrafish during late larval stages: A model based on local interactions. Developm. Dyn. **232**(1), 33–42 (2004)
36. Preziosi, L. (ed.): Cancer Modelling and Simulation. Chapman Hall/CRC Press, Boca Raton, Florida, USA (2003)
37. Rejniak, K.A., Anderson, A.R.A.: Hybrid models of tumor growth. Wiley Interdiscip. Rev. Syst. Biol. Med. **3**(1), 115–125 (2011)
38. Starruß, J., de Back, W., Brusch, L., Deutsch, A.: Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology. Bioinformatics **30**, 1331–1332 (2014)

39. Voss-Böhme, A., Deutsch, A.: The cellular basis of cell sorting kinetics. J. Theor. Biol. **263**(4), 419–436 (2010)
40. Wolf-Gladrow, D.A.: Lattice-Gas Cellular Automata and Lattice Boltzmann Models. Springer, Berlin (2000)
41. Wolfram, S.: A new kind of science. Wolfram Media, Inc (2002)

Tag Systems and the Complexity of Simple Programs

Turlough Neary[1(✉)] and Damien Woods[2]

[1] Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland
tneary@ini.phys.ethz.ch
[2] Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA
woods@caltech.edu

Abstract. In this mini-survey we discuss time complexity and program size results for universal Turing machines, tag systems, cellular automata, and other simple models of computation. We discuss results that show that many of the simplest known models of computation including the smallest known universal Turing machines and the elementary cellular automaton Rule 110 are efficient simulators of Turing machines. We also recall a recent result where the halting problem for tag systems with only 2 symbols (the minimum possible) is proved undecidable. This result has already yielded applications including a significant improvement on previous undecidability bounds for the Post correspondence problem and the matrix mortality problem.

1 Introduction

This brief survey is concerned with time complexity and program size results for universal Turing machines, tag systems, cellular automata and other simple models of computation. We pay particular attention to tag systems as they are at the center of many of the results we discuss. Here we provide only a brief glimpse at the above mentioned topics and we direct the reader who wishes to learn more to other related surveys [17,27].

2 Program Size of Small Universal Machines

In 1956 Shannon [35] considered the question of finding the smallest possible universal Turing machine, where size is the number of states and symbols. Figure 1 summarises the state of the art for the smallest known standard, weakly and semi-weakly universal Turing machines. Here we say a machine is standard if it

Turlough Neary is supported by Swiss National Science Foundation grants 200021-141029 and 200021-153295. Damien Woods is supported by NASA grant NNX13AJ56G, and National Science Foundation grants 0832824 & 1317694 (The Molecular Programming Project), CCF-1219274 and CCF-1162589.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 11–16, 2015.
DOI: 10.1007/978-3-662-47221-7_2

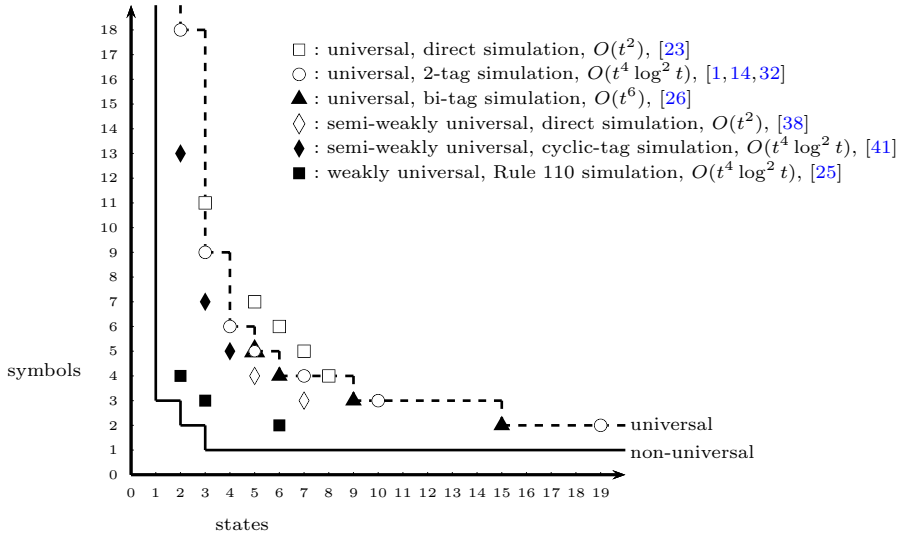


Fig. 1. State-symbol plot of small universal Turing machines. The type of simulation is given for each group of machines. For example, each machine plotted as a hollow circle simulates 2-tag systems, a technique introduced by Minsky [20]. The technique of Rule 110 simulation was devised by Cook [4] and later improved upon in [26] to give the weakly universal machines plotted as solid squares. Direct simulation indicates that a machine was proved universal by simulating Turing machines directly rather than via some other system. Simulation time overheads are given in terms of simulating a single-tape deterministic Turing machine that runs in time t.

is deterministic and has a single-tape. Semi-weakly universal machines generalise the standard model by allowing an infinitely repeated word on one side of the input, and the (standard) infinitely repeated blank symbol on the other. Weakly universal machines are a further generalisation on the standard model as they allow an infinitely repeated word to the left of the input, and another infinitely repeated word to the right. It is often the case that generalising the model allows us to find smaller universal programs. This notion is borne out when we compare the standard, weak and semi-weak machines in Figure 1.

In Figure 1 the machines with the state-symbol pairs $(2, 18)$, $(3, 9)$, $(4, 6)$, $(5, 5)$, $(6, 4)$, $(9, 3)$, and $(15, 2)$ are the smallest known standard machines and these machines define the universal curve (dashed line). Figure 1 also gives a non-universal curve. This curve is a lower bound that gives the state-symbol pairs for which it is known that the halting problem is decidable [11,13,28, 29]. It is currently unknown whether all of the lower bounds in Figure 1 hold for weak and semi-weak machines. For example, the non-universality results of Pavlotskaya [28,29] were proven under the assumption that the (standard) blank symbol is infinitely repeated to the left and right of the input.

3 Time Efficiency of Small Universal Machines

Cocke and Minsky [3] proved 2-tag systems universal via an exponentially slow simulation of Turing machines. So for many years the smallest known universal Turing machines were also exponentially slow. However, following the introduction of a new efficient 2-tag system algorithm for simulating Turing machines [40], the smallest known universal machines were found to be efficient simulators of Turing machines. To be more exact, given a single tape Turing machine M that runs in time t the small machines given by hollow circles in Figure 1 simulate M in time $O(t^8 \log^4 t)$ (this overhead was later improved [21] to $O(t^4 \log^2 t)$). Another consequence of this result is that many other systems [10,12,15,16,33,34,36] that simulate 2-tag systems, either directly or via a chain of simulations, are polynomial (instead of exponential) time simulators of Turing machines.

Rule 110 was proved universal by Matthew Cook via an impressive and intricate simulation of cyclic tag systems, the result is described in [39] and a full proof is given in [4]. Unfortunately, cyclic tag systems simulated Turing machines via Cocke and Minsky's exponentially slow 2-tag algorithm and so Rule 110 and the small weakly universal machines mentioned earlier were exponentially slow simulator of Turing machines. However, in [22] it was shown that cyclic tag systems simulate Turing machines in polynomial time. As a result, the following problem is now P-complete for Rule 110: Given a number t in unary, an initial configuration of Rule 110 and a cell c_i, predict the state of c_i after t timesteps. Rule 110 is the simplest (one-dimensional, nearest neighbour) cellular automaton that has been shown to have a P-complete prediction problem.

4 Universality of Binary Tag Systems

Tag systems are a type of rewriting system that use a very simple form of rule. A tag system acts on a dataword, which is a string of symbols taken from a finite alphabet Σ. There is a fixed set of rules $R : \Sigma \to \Sigma^*$ and a deletion number $\beta \in \mathbb{N}$. In a single timestep, the leftmost symbol σ_j of the dataword is read, if there is a rule $\sigma_j \to \alpha_j$ then the string α_j is appended to the right of the dataword and the leftmost β symbols are deleted. As an example we give the first 4 steps of Post's [31] binary tag system with deletion number 3 and the rules $0 \to 00$ and $1 \to 1101$ on the input 0101110.

$$0101110 \vdash 111000 \vdash 0001101 \vdash 110100 \vdash 1001101 \vdash \cdots$$

A tag system computation halts if its dataword is shorter than its deletion number. Surprisingly, the halting problem for the simple tag system given above (which Post discovered in the 1920s) remains open to this day [6]. Another remarkably simple tag system with an open halting problem was found by De Mol [5] when she reduced the well known Collatz problem to the halting problem for a tag system with only 3 rules and deletion number 2.

Tag systems were introduced by Post [30,31] and proved universal by Minsky [19]. Soon after this Cocke and Minksy [3] proved 2-tag systems (tag systems with deletion number 2) universal. As mentioned earlier, 2-tag systems have been used to prove universality for many of the smallest known universal Turing machines [1,14,20,32] and are central to many other universality results [10,12,15,16,33,34,36]. Given that tag systems have been so useful in the search for new simple universal systems, it is surprising that there have been no attempts to simplify tag systems since the 1960s. All known universal tag systems [3,4,37] have a large number of symbols and thus a large number of rules. Recently [24] it was shown that tag systems with only 2 symbols (the minimum possible) are universal by showing that the simulate cyclic tag systems. Applications have already been found for this result. The undecidable halting problem for binary tag systems reduces to the Post correspondence problem for 5 pairs of words. The previous bound for undecidability in this problem, which is due to Matiyasevich and Sénizergues [18], was 7 pairs. Following this new result, only the cases for 3 and 4 pairs of words remain open, as the problem is known to be decidable for 2 pairs [7]. Applying the reductions of Halava and Harju [8], and Cassaigne and Karhumäki [2] to the Post correspondence problem for 5 pairs of words shows that the matrix mortality problem is undecidable for sets with six 3×3 matrices and for sets with two 18×18 matrices. The previous bounds for the undecidability in this problem was seven 3×3 matrices and two 21×21 matrices [9].

Looking to the future, we expect that binary tag systems will have an important role in proving further undecidability results and in the search for new simple models of computation.

References

1. Baiocchi, C.: Three small universal turing machines. In: Margenstern, M., Rogozhin, Y. (eds.) MCU 2001. LNCS, vol. 2055, pp. 1–10. Springer, Heidelberg (2001)
2. Cassaigne, J., Karhumäki, J.: Examples of undecidable problems for 2-generator matrix semigroups. Theoretical Computer Science **204**(1–2), 29–34 (1998)
3. Cocke, J., Minsky, M.: Universality of tag systems with $P = 2$. Journal of the Association for Computing Machinery **11**(1), 15–20 (1964)
4. Cook, M.: Universality in elementary cellular automata. Complex Systems **15**(1), 1–40 (2004)
5. De Mol, L.: Tag systems and Collatz-like functions. Theoretical Computer Science **390**(1), 92–101 (2008)
6. De Mol, L.: On the complex behavior of simple tag systems - an experimental approach. Theoretical Computer Science **412**(1–2), 97–112 (2011)
7. Ehrenfeucht, A., Karhumäki, J., Rozenberg, G.: The (generalized) Post correspondence problem with lists consisting of two words is decidable. Theoretical Computer Science **21**(2), 119–144 (1982)
8. Halava, V., Harju, T.: Mortality in matrix semigroups. American Mathematical Monthly **108**(7), 649–653 (2001)

9. Halava, V., Harju, T., Hirvensalo, M.: Undecidability bounds for integer matrices using Claus instances. International Journal of Foundations of Computer Science **18**(5), 931–948 (2007)
10. Harju, T., Margenstern, M.: Splicing systems for universal Turing machines. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 149–158. Springer, Heidelberg (2005)
11. Hermann, G.T.: The uniform halting problem for generalized one state Turing machines. In: Proceedings of the Ninth Annual Symposium on Switching and Automata Theory (FOCS), pp. 368–372, IEEE Computer Society Press, Schenectady, New York, Oct. 1968
12. Hooper, P.: Some small, multitape universal Turing machines. Information Sciences **1**(2), 205–215 (1969)
13. Kudlek, M.: Small deterministic Turing machines. Theoretical Computer Science **168**(2), 241–255 (1996)
14. Kudlek, M., Rogozhin, Y.: A universal turing machine with 3 states and 9 symbols. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 149–158. Springer, Heidelberg (2002)
15. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems **4**(3), 299–318 (1990)
16. Margenstern, M.: Non-erasing Turing machines: A new frontier between a decidable halting problem and universality. In: Baeza-Yates, R.A., Poblete, P.V., Goles, E. (eds.) LATIN. LNCS, vol. 911, pp. 386–397. Springer, Heidelberg (1995)
17. Margenstern, M.: Frontier between decidability and undecidability: a survey. Theoretical Computer Science **231**(2), 217–251 (2000)
18. Matiyasevich, Y., Sénizergues, G.: Decision problems for semi-Thue systems with a few rules. Theoretical Computer Science **330**(1), 145–169 (2005)
19. Minsky, M.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Annals of Mathematics **74**(3), 437–455 (1961)
20. Minsky, M.: Size and structure of universal Turing machines using tag systems. In: Recursive Function Theory: Proceedings, Symposium in Pure Mathematics, vol. 5, pp. 229–238, AMS, Provelence (1962)
21. Neary, T.: Small universal Turing machines. Ph.D thesis, National University of Ireland, Maynooth (2008)
22. Neary, T., Woods, D.: P-completeness of Cellular Automaton Rule 110. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 132–143. Springer, Heidelberg (2006)
23. Neary, T., Woods, D.: Small fast universal Turing machines. Theoretical Computer Science **362**(1–3), 171–195 (2006)
24. Neary, T.: Undecidability in binary tag systems and the Post correspondence problem for five pairs of words. In: Mayr, Ernst W., Ollinger, Nicolas (eds.) 32nd International Symposium on Theoretical Aspects of Computer Science, (STACS 2015), vol. 30 of LIPIcs, pp. 649–661 (2015)
25. Neary, T., Woods, D.: Small weakly universal turing machines. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 262–273. Springer, Heidelberg (2009)
26. Neary, T., Woods, D.: Four small universal Turing machines. Fundamenta Informaticae **91**(1), 123–144 (2009)
27. Neary, T., Woods, D.: The complexity of small universal turing machines: a survey. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 385–405. Springer, Heidelberg (2012)

28. Pavlotskaya, L.: Solvability of the halting problem for certain classes of Turing machines. Mathematical Notes (Springer) **13**(6), 537–541 (1973). (Translated from Matematicheskie Zametki, Vol. 13, No. 6, pp. 899–909, June, 1973)
29. Pavlotskaya, L.: Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring. Avtomaty i Mashiny, pp. 91–118 (1978). (Sufficient conditions for the halting problem decidability of Turing machines. In Russian)
30. Post, E.L.: Formal reductions of the general combinatorial decision problem. American Journal of Mathmatics **65**(2), 197–215 (1943)
31. Post, E.L.: Absolutely unsolvable problems and relatively undecidable propositions - account of an anticipation. In: Davis, M. (ed.) The undecidable: basic papers on undecidable propositions, unsolvable problems and computable functions, pages 340–406. Raven Press, New York (1965). (Corrected republication, Dover publications, New York, 2004)
32. Rogozhin, Y.: Small universal Turing machines. Theoretical Computer Science **168**(2), 215–240 (1996)
33. Rogozhin, Y., Verlan, S.: On the rule complexity of universal tissue P systems. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 356–362. Springer, Heidelberg (2006)
34. Rothemund, P.W.K.: A DNA and restriction enzyme implementation of Turing Machines, In: Lipton, R.J., Baum, E.B. (eds.) DNA Based Computers: Proceeding of a DIMACS Workshop, vol. 2055 of DIMACS, pp. 75–119. AMS, Princeton University (1996)
35. Shannon, C.E.: A universal Turing machine with two internal states. Automata Studies, Annals of Mathematics Studies **34**, 157–165 (1956)
36. Siegelmann, H.T., Margenstern, M.: Nine switch-affine neurons suffice for Turing universality. Neural Networks **12**(4–5), 593–600 (1999)
37. Wang, H.: Tag systems and lag systems. Mathematical Annals **152**(4), 65–74 (1963)
38. Watanabe, S.: 4-symbol 5-state universal Turing machine. Information Processing Society of Japan Magazine **13**(9), 588–592 (1972)
39. Wolfram, S.: A new kind of science. Wolfram Media Inc (2002)
40. Woods, D., Neary, T.: On the time complexity of 2-tag systems and small universal Turing machines. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 439–446. IEEE, Berkeley, California, Oct. 2006
41. Woods, D., Neary, T.: Small semi-weakly universal Turing machines. Fundamenta Informaticae **91**(1), 179–195 (2009)

Groups and Monoids of Cellular Automata

Ville Salo(✉)

Center for Mathematical Modeling, University of Chile, Santiago, Chile
vosalo@utu.fi

Abstract. We discuss groups and monoids defined by cellular automata on full shifts, sofic shifts, minimal subshifts, countable subshifts and coded and synchronized systems. Both purely group-theoretic properties and issues of decidability are considered.

Keywords: Automorphism groups · Cellular automata · Subshifts

1 Motivation

1.1 From CA to Monoid Actions

Consider a system (X, f) with a (discrete) time-evolution rule $f : X \to X$. The set X is thought of as the set of all possible states of the system, and f tells us how the system evolves as time progresses. If the system is currently in state $x \in X$, it will be in state $f(x)$ after one time step. Systems that fit this general picture are studied in physics as models of the real world, in computer science as models of computation, and in mathematics for their intrinsic interest. In the world of cellular automata (CA), X is typically the set of bi-infinite sequences over a finite alphabet, and f is a cellular automaton on this space.[1]

Questions we ask about such systems can be both dynamical and computational (or a combination of the two). When X has a topological or measurable structure, often the motivating question on the side of dynamics is how 'chaotic' the system is. Trying to understand what is chaotic about a system leads to the study of properties such as transitivity, mixing and entropy. The study of what is not chaotic about it leads to the study of periodicity, almost periodicity, equicontinuous factors and spectral theory. For cellular automata, a range of different behaviors are known to be possible. Another common motivating question on the side of both dynamics and computation is whether the system is 'universal'. We wish to understand which other systems our system can simulate, in one sense or another.

A particularly interesting case in all these considerations is when f is reversible, that is, no information is lost when f is applied. Typically, we are interested in

The author was supported by FONDECYT Grant 3150552.

[1] Later, we will instead fix f to be the left shift map on X, and see cellular automata in another way.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 17–45, 2015.
DOI: 10.1007/978-3-662-47221-7_3

a strong, structural kind of reversibility where there is a concrete inverse time-evolution rule $g : X \to X$ such that $f \circ g$ and $g \circ f$ are identity maps on X. The importance of reversibility in physics is that the laws of nature seem to be reversible, and thus it makes sense to make such an assumption on our models. In computer science, reversible computation is an interesting programming paradigm promising challenges for theoreticians, but also allows more energy-efficient computation. In mathematics, reversibility often makes systems more malleable to the development of theory.

Now, what happens if we have two rules? Suppose $f : X \to X$ and $g : X \to X$ are two evolution rules on X, that is, we have two transformations on X which we be applied in any order. This can model many situations. For example, we can apply f and g at random, so that X has two ways to evolve at each time step, and we want to understand what the limit behavior is likely to be. This is a very simple example of a (discrete) random dynamical system. In parallel computing, we can think of f and g as two computations that are happening in parallel, and we need to understand their interaction to know the good or legal orders to apply them in. In this case, the order in which the maps are applied might be chosen by an adversary. The reversibility of a system with two evolution rules can be defined as simply f and g being reversible: it automatically follows that every finite composition of f and g is reversible, as one can apply the inverses of f and g in the reverse order to undo their actions.[2]

Abstracting from this, in the not necessarily reversible case we obtain the mathematical concept of a monoid action, where instead of a single function on X, we have a countable discrete monoid M acting on X by functions. More precisely, we associate to each $m \in M$ a function $\phi_m : X \to X$, and write $m \cdot x = \phi_m(x)$ for short. We require that the obvious compatibility conditions $m \cdot m' \cdot x = mm' \cdot x$ and $1 \cdot x = x$ hold. The systems (X, f) with a single time-evolution rule are modeled by setting $M = \mathbb{N}$, and $n \cdot x = f^n(x)$ for all $n \in \mathbb{N}$. The systems with two evolution rules are most generally modeled by setting M to be the free monoid[3] with two generators a, b and defining the action of M on X by $a \cdot x = f(x)$ and $b \cdot x = g(x)$.

When the operations are reversible, we usually consider group actions instead,[4] that is, we choose the monoid M to have multiplicative inverses (so that it is a group). A group action should of course have the additional compatibility condition that the inverse g^{-1} of $g \in G$ undoes the action of g on every element. This is automatic: if $a \in M$ and $a^{-1} \in M$ is its inverse (that is, $aa^{-1} = a^{-1}a = 1$),

[2] Note that if f and g have been applied to x, the resulting state does *not* carry information in which order, and how many times, f and g have been applied. We can only reverse their action if we know in which order they have been applied.

[3] This is just the set of all words over the alphabet $\{a, b\}$ with concatenation as the multiplication operation.

[4] However, no-one forces us to: for example, from a reversible CA, we obtain both an $\mathbb{N}$-action and a $\mathbb{Z}$-action. While these systems may look the same, their properties may be different. For example, many $\mathbb{N}$-actions of CA are expansive (that is, there are *positively expansive* CA). A reversible CA cannot be expansive as an $\mathbb{N}$-action, but $\mathbb{Z}$-actions by reversible CA can be expansive.

then the action of $a^{-1} \in M$ is automatically the inverse of the action $a \in M$ by the compatibility condition for the monoid action:

$$a \cdot (a^{-1} \cdot x) = aa^{-1} \cdot x = 1 \cdot x = x = a^{-1}a \cdot x = a^{-1} \cdot (a \cdot x).$$

Thus, the natural compatibility conditions for a group action are the same as for a monoid action. If we have a single reversible time-evolution rule f, we obtain a natural action of $\mathbb{Z}$ by the same formula as in the non-reversible case, but extending to negative powers in the obvious way. In the case of two reversible actions, we obtain an action of the free group on two generators (see Section 2), again by the same formula. Considering the element $1 \in \mathbb{N}$ or $1 \in \mathbb{Z}$, the same formula shows that actions of $\mathbb{N}$ and $\mathbb{Z}$ are in one-to-one correspondence with systems with a time-evolution rule, and ones with a reversible time-evolution rule, respectively.

For most of the notions for $\mathbb{N}$-actions and $\mathbb{Z}$-actions, such as expansivity, transitivity, entropy and almost equicontinuity, there exist one or more corresponding notions that apply more generally to monoid or group actions. For example, one-dimensional expansivity for $\mathbb{Z}$-actions is defined by the formula $\exists \epsilon > 0 : \forall x, y \in X : \exists n \in \mathbb{Z} : d(f^n(x), f^n(y)) > \epsilon$. The formula defining expansivity for a general monoid is obtained by replacing $\mathbb{Z}$ by the monoid and f^n by the action of n. Sometimes additional conditions are needed on the monoid for the definition to work. For example, the entropy of a group action is measured along a Følner sequence, and thus requires amenability.[5]

The generalization allows us to ask what the dynamics of a finite (or even infinite) set of cellular automata looks like. We can choose a set of cellular automata F, let them generate a free monoid or group, and investigate the properties of the corresponding action. We can also fix the monoid to be M, and ask what M-actions of cellular automata look like. For example, it is known that the entropies of $\mathbb{N}$-actions by cellular automata on one-dimensional full shifts are precisely the class Π_1 of positive real numbers [GZ12]. What is the corresponding class for $\mathbb{N}^2$-actions of cellular automata (given by two CA f and g satisfying $f \circ g = g \circ f$)? For $\mathbb{Z}$-actions?[6]

1.2 From Monoid Actions to the Endomorphism Monoid

Now, let us return to the simple systems (X, σ) with a single time-evolution rule and suppose now that X is a compact Hausdorff space, and $\sigma : X \to X$ is continuous. It turns out that there is a monoid that one can attach to any such system: let $\mathrm{End}(X)$[7] be the set of all continuous functions $f : X \to X$ such that $\sigma \circ f = f \circ \sigma$ (as functions). Then $\mathrm{End}(X)$ becomes a monoid, called the *endomorphism monoid*, under function composition. We define $\mathrm{Aut}(X)$, the *automorphism group*, as the restriction of $\mathrm{End}(X)$ to the elements which are

[5] *Sofic entropy* applies more generally to sofic groups.[Bow10]

[6] The entropies of $\mathbb{Z}$-actions are just the entropies of reversible cellular automata. Characterizing the set of entropies in this case (on full shifts) seems to be essentially harder than in the non-reversible case.

[7] We omit σ from the notation $\mathrm{End}(X)$ since we consider it an intrinsic part of X.

bijective. From the assumption that X is compact and Hausdorff, it follows that every bijective continuous function on X has a continuous inverse. Since the inverse is easily seen to also commute with σ, $\mathrm{Aut}(X)$ is indeed a group under function composition. The endomorphism monoid and the automorphism group act on X in the obvious way: $f \cdot x = f(x)$.

This gives another way to see the set of cellular automata on a full shift: Let $X = S^{\mathbb{Z}}$ and let $\sigma : X \to X$ be the left shift map defined by $\sigma(x)_i = x_{i+1}$. Then $\mathrm{End}(X)$ is precisely the set of cellular automata on X: the continuous shift-commuting maps are precisely the ones admitting a spatially uniform local rule. Thinking of cellular automata as elements of the endomorphism monoid, and considering its action on X, we get a more global way to look at cellular automata, as this point of view encompasses not only the possible dynamics and computational power of individual cellular automata, but also their possible interactions.

In this paper we take a very simplified approach to this interaction, by forgetting the action of $\mathrm{End}(X)$ on X. Thus, we are interested in the abstract structure of the monoid $\mathrm{End}(X)$, and in particular the group $\mathrm{Aut}(X)$. This omits many interesting questions – for example, chaoticity or universality of a cellular automaton or a family of cellular automata does not (to our knowledge) in any way differentiate it among the other elements of $\mathrm{End}(X)$.[8] All that matters is which equalities $f_n \circ f_{n-1} \circ \cdots \circ f_1 = g_m \circ g_{m-1} \circ \cdots \circ g_1$ hold, when f_i and g_i are cellular automata. In the case of groups, it only matters which compositions of cellular automata are equal to the identity map on X.

While dynamical properties are out of the question, there are many things we can ask: For any property of groups (of which there are many), we can ask if $\mathrm{Aut}(X)$ has this property. Is it abelian? Is it amenable? For many group-theoretic decidability questions (of which there are many) we can ask if the question is decidable for $\mathrm{Aut}(X)$. Is the word problem decidable? What about the torsion problem? We can also ask if there are alternative descriptions of $\mathrm{Aut}(X)$, or connections with previously known groups.

It turns out that many interesting things can be said when X is a full shift over an alphabet S of size at least 2: For example, every finite group can be embedded in $\mathrm{Aut}(X)$, as can $\mathbb{Z}$ and free groups with a countable number of generators [Hed69,BLR88]. On the other hand, this group is residually finite and has a decidable word problem.

Fixing the action σ on $S^{\mathbb{Z}}$ makes it natural to consider also subshifts $X \subset S^{\mathbb{Z}}$, where we forbid a possibly infinite set of finite words from appearing in points of $S^{\mathbb{Z}}$. There are uncountably many subshifts X, and to each we associate the monoid $\mathrm{End}(X)$ and the group $\mathrm{Aut}(X)$, which act on X by function application. On each subshift X, $\mathrm{End}(X)$ still corresponds to the usual cellular automata,

[8] It is a very interesting question which properties have such algebraic definitions. Equicontinuity corresponds to eventual periodicity and by *Ryan's theorem* the shift maps are precisely the center of $\mathrm{Aut}(X)$ [Rya72] and more generally of $\mathrm{End}(X)$ [Sal14b] on mixing SFTs, but we don't know much more.

defined by a local rule; of course, it may be hard or impossible to tell which local rules give a well-defined map on X.

It turns out that the known constructions on full shifts can be carried out in many subshift X under some chaoticity assumptions on the action of the shift map on X. In fact, we show in Section 3 that we can embed the automorphism group of a full shift in many subshifts, such as all positive entropy sofic shifts [BLR88]. In Section 6, we show that similar constructions, and much more, can be carried out on the larger classes of synchronized and coded systems – in particular on coded systems we obtain a large set of groups as automorphism groups [FF96].

We also show examples of subshifts on which the automorphism group is essentially smaller, and some ways to control this. In Section 5 we discuss some interesting recent results in the case where either the word complexity grows slowly or recurrence times are short. In particular, in the case of subshifts with linear word complexity (for example, ones generated by primitive substitutions), we seem to be very close to a full understanding of the set of automorphism groups. In the case of countable sofic shifts, the author is working on the characterization of the automorphism groups, and we give an example of such a computation in Section 4.

Remark 1. Since, on the full shift, Ryan's theorem guarantees that there is an algebraic way to separate the shift map from the others (as it is the center of the group), we have no need to carry it in the structure $\mathrm{Aut}(X)$ explicitly. However, this is not true in general, as for example in minimal and coded systems we can have large abelian automorphism groups (so the group is its own center). In this case, it makes sense to think of σ as part of the algebraic structure, and consider for example the group $\mathrm{Aut}(X)/\langle\sigma\rangle$ instead of $\mathrm{Aut}(X)$. This simplifies many problems, as for example the characterization of these groups is known in the linear word complexity case.

2 Definitions and Basic Results

We give basic definitions of dynamical systems. In particular for the case of subshifts with $\mathbb{Z}$-actions, some standard references are [Kůr03, LM95, Kit98].

By a *dynamical (M-)system* we mean a pair (X, M, ϕ) where X is a compact metric zero-dimensional space, M is a countable discrete monoid, and M acts on X by $\phi_m : X \to X$ for $m \in M$, that is,

$$\phi_1(x) = x \text{ and } \forall m, m' \in M : \phi_{m\cdot m'}(x) = \phi_m(\phi_{m'}(x)).$$

Usually, the action is left implicit, and we write simply (X, M) for the system and $m \cdot x$ for $\phi_m(x)$. Of particular interest to us are the *subshifts* $(X, \mathbb{Z}^d, \sigma)$, topologically closed sets $X \subset S^{\mathbb{Z}^d}$ which are invariant under the *shifts* $\sigma_{\boldsymbol{v}}$ defined by $\sigma_{\boldsymbol{v}}(x)_{\boldsymbol{w}} = x_{\boldsymbol{w}+\boldsymbol{v}}$, where S is some finite alphabet. The subshift $S^{\mathbb{Z}^d}$ is called

the *full shift (over S)*. A subshift $X \subset S^{\mathbb{Z}^d}$ is a dynamical system with a $\mathbb{Z}^d$-action given by the shifts. If d is not specified, we by default study the *one-dimensional* setting where $d = 1$, that is, the set $S^{\mathbb{Z}}$ of two-way infinite sequences, and explicitly state when studying the *multidimensional* case $d > 1$.

In the one-dimensional case,[9] subshifts are characterized as sets of sequences in a full shift where none of a (possibly infinity) set of *forbidden words* occurs. Yet another characterization is the following: Let $L \subset S^*$ be a set of words over S. We say L is *extendable* if $\forall w \in L : \exists a, b \in S : awb \in L$. The *factor-closure* of L is the set $F(L) = \{u \in S^* \mid \exists v, v' \in S^* : vuv' \in L\}$. Subshifts are precisely the sets of infinite words whose finite subwords belong to the factor-closure of a fixed extendable language. Thus, if L is extendable, we define

$$\mathcal{L}^{-1}(L) = \{x \in S^{\mathbb{Z}} \mid \forall a, b : x_{[a,b]} \in F(L)\}.$$

We write $\mathcal{L}(X)$ for the *language* of X, that is, the set of words occurring in X. It is always factor closed, and $\mathcal{L}(\mathcal{L}^{-1}(L)) = F(L)$ and $\mathcal{L}^{-1}(\mathcal{L}(X)) = X$ for an extendable language L and a subshift X. We write $\mathcal{L}_n(X) = \mathcal{L}(X) \cap S^n$.

A *sofic shift* is a subshift that can be defined by a regular language of forbidden words, or alternatively as $\mathcal{L}^{-1}(L)$ for an extendable regular language L. An *SFT* is a subshift that can be defined by a finite set of forbidden words.

The *endomorphism monoid* of a subshift consists of the continuous functions on X which commute with the translations:

$$\mathrm{End}(X) = \{f : X \to X \mid f \text{ continuous and } \forall \boldsymbol{v} \in \mathbb{Z}^k : \sigma_{\boldsymbol{v}} \circ f = f \circ \sigma_{\boldsymbol{v}}\}.$$

We give $\mathrm{End}(X)$ the structure of a monoid by function composition $(f, g) \mapsto f \circ g$. The monoid $\mathrm{End}(X)$ acts on X from the left by $f \cdot x = f(x)$, and thus $(X, \mathrm{End}(X))$ is itself a dynamical system. We are interested in the following family of questions:

Question 1. What can we say about $\mathrm{End}(X)$ as a monoid and $\mathrm{Aut}(X)$ as a group by looking at properties of X? What can we say about X by looking at properties of $\mathrm{End}(X)$ (or $\mathrm{Aut}(X)$)?

We emphasize in particular the automorphism group, since it is often easier to understand than the endomorphism monoid, and since there is more literature on it. For any property of groups, and any subshift X, we can ask if $\mathrm{Aut}(X)$ has the property.

Some notions we need, mainly for $\mathbb{Z}$-subshifts, are the following: (X, σ) is *transitive* if there exists a *transitive point*, that is, a point $x \in X$ such that $\bigcup_{n \in \mathbb{Z}} \sigma^n(x) = X$. If every point is transitive, the system is *minimal*, equivalently, it has no proper subsystems except the empty one.

In terms of words, a minimal subshift is one where every word occurs with bounded gaps, that is, in every long enough word that occurs in a point of

[9] And in more dimensions with an obvious generalization.

the subshift. Transitivity means $\forall u, v \in \mathcal{L}(X) : \exists w : uwv \in \mathcal{L}(X)$. A subshift $X \subset S^{\mathbb{Z}}$ is *mixing* if

$$\forall u, v \in \mathcal{L}(X) : \exists n : \forall m \geq n : \exists w \in \mathcal{L}_m(X) : uwv \in \mathcal{L}(X).$$

The *entropy* of a subshift X is $\lim_{n\to\infty} \frac{\log |\mathcal{L}_n(X)|}{n}$.

2.1 Properties and Examples of Groups

For a group G, we can ask which kind of subshifts (if any) can have it as an automorphism group, or can have an embedded copy of it in the automorphism group. Similarly, each property of groups gives a family of questions about automorphism groups subshifts: Does there exist a subshift whose automorphism group has that property? Does one exist in a particular class such as the class of SFTs or minimal subshifts? What closure properties do automorphism groups have when restricting to particular classes of subshifts? Can we characterize the automorphism groups of some families of subshifts? A group is not determined by its subgroups, and thus it is also interesting to ask what kind of subgroups automorphism groups have. In this section, we give basic group theoretical definitions needed in later sections. The notation and definitions in this section are mostly standard, but we give them for completeness. For more details, the reader may consult any standard reference [Rot95].

In this section, and usually also in other sections a 'group' is a countable (discrete) group. Thus, a group is a countable set of objects called *elements*, where a mapping $(\cdot) : G \times G \to G$ satisfying a particular set of axioms is defined. The axioms are associativity $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, the existence of an identity element $\exists 1 \in G : a \cdot 1 = 1 \cdot a = a$ (which is automatically unique) and the existence of inverses $\forall a : \exists a^{-1} : a \cdot a^{-1} = a^{-1} \cdot a = 1$ (which are automatically unique), where a, b, c are arbitrary elements of the group. We often drop the symbol '$\cdot$' and write $ab = a \cdot b$. Groups are usually denoted by G and H. If $g_1, g_2, \cdots, g_k \in G$, the products of g_i and g_i^{-1} generate a subgroup of G, and we write this group as $\langle g_1, g_2, \cdots, g_k \rangle$. This is the *subgroup of G generated by* $g_1, g_2, \cdots, g_k$. This naturally generalizes to infinite sets of generators.

A group that has a finite set of generators is *finitely generated*. A group is *cyclic* if it has only one generator, that is, $G = \langle g \rangle$ for some $g \in G$. The only infinite cyclic group is the additive group of integers $\mathbb{Z}$ with the operation $a \cdot b = a + b$.[10] The finite cyclic groups are the groups $\mathbb{Z}_n$ with elements $[0, n-1]$ and group operation $a \cdot b = ((a + b) \bmod n)$.[11] A group is *locally finite* if its finitely generated subgroups are finite.

There are many ways to build new groups from existing ones. If G is a group, a *subgroup* $H \leq G$ of G is a subset of G which is closed under products and

[10] Of course, $\mathbb{Z}$ has a natural multiplication operation as well, the multiplication of integers, but it does not yield a group.

[11] Again, multiplication could be defined on $\mathbb{Z}_n$ by integer multiplication modulo n, but this does not form a group. However, if 0 is omitted, we *do* obtain a group when n is a prime.

inverses, that is, $g, h \in H \implies gh \in H \wedge g^{-1} \in H$. When H is a subgroup, the sets $gH = \{gh \mid h \in H\}$ for $g \in G$ are called *cosets*, and they form a partition of G (that is, $gH \cap g'H \neq \emptyset \implies gH = g'H$). The set of cosets is denoted by G/H. We say H is *normal* and write $H \trianglelefteq G$ if $gH = Hg$ for all $g \in G$, and then the cosets G/H have a natural group structure given by $gH \cdot g'H = gg'H$. The cardinality of G/H is denoted by $[G : H]$, and is called the *index* of H in G. If the index is finite, H is called a *finite-index subgroup*. If H has a particular group property P and H is a finite-index subgroup of G, then we say G is *virtually P*.

A function $\pi : G \to H$ satisfying $\pi(gh) = \pi(g)\pi(h)$ is called a *group homomorphism* from G to H. We say H is a *quotient* of the group G if there is a surjective group homomorphism $\pi : G \to H$. A bijective group homomorphism is called an *isomorphism*, and we write $G \cong H$ if there is an isomorphism between G and H. We say G and H are *isomorphic*, and consider them the same group for most purposes. The *kernel* $\ker(\pi) \subset G$ of a homomorphism π consists of the elements $g \in G$ such that $\pi(g) = 1_H$. It is always a normal subgroup, and $G/\ker(f) \cong \pi(G)$. The group G is *residually finite* if for all $g \in G$ with $g \neq 1$ there exists a finite group H and a homomorphism $\phi : G \to H$ such that $\phi(g) \neq 1_H$.

Given two groups G, H, one can construct larger groups in multiple ways. The *direct product* of G and H is the group $G \times H$ whose elements are the elements of the Cartesian product $G \times H$ and the operation is $(g, h) \cdot (g', h') = (gg', hh')$. Generalizing this, we define the *semidirect product* of G and H as follows. Write $\mathrm{Aut}(G)$ for the group of bijective group homomorphisms from G to itself. Let $\psi : H \to \mathrm{Aut}(G)$ be a group homomorphism, and define $G \rtimes_\psi H$ (or just $G \rtimes H$) as the group with the Cartesian product $G \times H$ as elements, and

$$(g, h)(g', h') = (g\psi_h(g'), hh')$$

as the operation. The idea is that H is 'acting' on G. We show only associativity:

$$\begin{aligned}(a, b)((c, d)(e, f)) &= (a, b)(c\psi_d(e), df) \\ &= (a\psi_b(c\psi_d(e)), bdf) \\ &= (a\psi_b(c)\psi_{bd}(e)), bdf)) \\ &= (a\psi_b(c), bd)(e, f) \\ &= ((a, b)(c, d))(e, f).\end{aligned}$$

Both the direct and semidirect product have G and H as subgroups in a natural way: $G \cong G \times \{1\}$ and $H \cong \{1\} \times H$. Both subgroups are normal in $G \times H$, but (a priori) only G is normal in $G \rtimes H$.

A more general 'construction' is the following: suppose $N \trianglelefteq G$, and $\phi : G \to H$ is a homomorphism with $\ker(\phi) = N$. Then G is a *group extension* of H by N. If G is a direct product of N and H, then it can be seen as a group extension of N by H, and of H by N. The semidirect product $N \rtimes H$ is a group extension of H by N. However, there are not the only possible extensions, and even among finite groups, there is no full understanding of group extensions: the problem of characterizing them is called the *extension problem*, and it is still a major open

problem in group theory. Nevertheless, many properties of groups are closed under group extensions, in the sense that if N and H have the property, then also every group extension of H by N has this property.

We can also construct groups by combining infinitely many smaller groups. We present (simplified versions of) two of the most basic constructions. If $G_1 \subset G_2 \subset G_3 \subset \cdots$ is an increasing sequence of groups (by which we mean that the elements of G_i are elements of G_{i+1} for all i and the group operations are compatible), then $\bigcup G_i$ has an obvious group structure by $g \cdot h = g \cdot_i h$, where $\cdot_i$ is the group operation of any G_i with $g, h \in G_i$. This is called the *direct union* of the groups G_i. For an arbitrary countable family of groups $G_1, G_2, G_3, \cdots$, their *direct sum* is the group $\bigoplus_i G_i$ whose elements are functions $f : \mathbb{N} \to \bigcup_i G_i$ with $f(i) \in G_i$ for all i and $f(i) = 1_{G_i}$ for all but finitely many i, and whose group operation is pointwise product: $(f \cdot g)(i) = f(i) \cdot_i g(i)$.

We say G is *abelian* if $ab = ba$ for all $a, b \in G$. The finitely generated abelian groups are of the form $\mathbb{Z}^d \times \mathbb{Z}_{k_1} \times \cdots \times \mathbb{Z}_{k_n}$ for some $k_1, \ldots, k_n \in \mathbb{N}$. General countable abelian groups are *not* obtained by extending this to infinite products of this – in fact there is no full characterization of them. The best-known example of a non-finitely generated abelian group is probably the additive group of rational numbers $\mathbb{Q}$ with operation $a \cdot b = a + b$.[12]

A notion generalizing abelianity is nilpotency. For a group G and $g, h \in G$, define the *commutator* of g and h as $[g, h] = ghg^{-1}h^{-1}$ and the *commutator subgroup* $[A, B]$ generated by $A, B \subset G$ as the one genenerated by $[a, b]$ where $a \in A, b \in B$. The *lower central series* of G is defined inductively by $G_1 = G$ and $G_{i+1} = [G_i, G]$. If G_i is the trivial group $\{1\}$ for some $i \in \mathbb{N}$, then G is said to be *nilpotent*, and the smallest i such that $G_i = \{1\}$ is the *step* of G. Every abelian group is nilpotent, but the converse does not hold.

For a set S, the *free group* F_S generated by S is defined as the group whose elements are all words over the alphabet $\{s, s^{-1} \mid s \in S\}$ where the subwords of the form ss^{-1} and $s^{-1}s$ do not occur, and $u \cdot v$ is the word obtained from uv by repeatedly erasing subwords of the form ss^{-1} and $s^{-1}s$ until none occur. For $n \in \mathbb{N}$, we write F_n for the free group with any n generators, as they are all isomorphic. The free group F_∞ with countably many generators is defined in the obvious way, as a direct union of the F_n.

Generalizing the definition of the free group, a group presentation is a combinatorial way to describe a group. Given a list of generators $g_1, g_2, g_3, \ldots$ (considered as formal symbols) and a list of words

$$w_1, w_2, w_3, \ldots \in \{g_1, g_1^{-1}, g_2, g_2^{-1}, g_3, g_3^{-1}, \cdots\}^*$$

(either list of may also be finite), we define the group $\langle g_1, g_2, g_3 \ldots \mid w_1, w_2, w_3, \ldots \rangle$ as the group whose elements are equivalence classes of words over the symbols g_i and g_i^{-1} under the equivalence relation $\sim$ generated as follows: $\lambda \sim 1$, $gg^{-1} \sim g^{-1}g = \lambda$ where λ is the empty word, and $uw_iv \sim uv$ for all $i \in \mathbb{N}$. If $G \cong$

[12] The usual multiplication of $\mathbb{Q}$ is again not a group, but if 0 is omitted we obtain another non-finitely generated group, namely the free abelian group on countably many generators, by the unique factorization theorem of rational numbers.

$\{g_1, g_2, \dots \mid w_1, w_2, \dots\}$, then $\{g_1, g_2, \dots \mid w_1, w_2, \dots\}$ is a *group presentation* of G. A group presentation for F_n is $\langle a_1, a_2, \cdots, a_n \mid \emptyset \rangle$ and one for $\mathbb{Z}^d$ is

$$\langle a_1, a_2, \cdots, a_d \mid \{a_i a_j a_i^{-1} a_j^{-1} \mid 1 \leq i, j \leq d\} \rangle.$$

Every (countable) group G has a group presentation $\{g_1, g_2, \dots \mid w_1, w_2, \dots\}$ where the g_i are an enumeration of elements of G and w_i are all words in $\{g_1, g^{-1}, g_2, g^{-2}, g_3, g_3^{-1}, \cdots\}^*$ which present the identity element of G.

Using group presentations, we can define another product of G and H, namely their *free product*. Choose presentations for the groups G and H, $G \cong \langle g_1, g_2, \dots \mid w_1, w_2, \dots \rangle$ and $H \cong \langle h_1, h_2, \dots \mid u_1, u_2, \dots \rangle$, and define

$$G * H = \langle g_1, h_1, g_2, h_2, g_3, h_3, \cdots \mid w_1, u_1, w_2, u_2, w_3, u_3, \dots \rangle.$$

This group is defined by G and H up to isomorphism, no matter which presentations are chosen. Its elements can be thought of as words $w \in (G \cup H)^*$ such that $w_i \in G \iff w_{i+1} \in H$, that is, elements of G and H alternate, where neither 1_G nor 1_H occurs in w. The product of $u, v \in G * H$ is obtained from the word uv by repeatedly combining two adjancent elements of G into a single element of G using the group operation of G, and symmetrically for H, and removing 1_G and 1_H whenever they occur. The free product of finite groups can be infinite. For example, the $\mathbb{Z}_2 * \mathbb{Z}_2$ is virtually $\mathbb{Z}$ and $\mathbb{Z}_2 * \mathbb{Z}_2 * \mathbb{Z}_2$ is virtually F_2.

A group G *acts* on a set X as explained already in the introduction, for each $g \in G$, $x \mapsto g \cdot x$ is a bijection on X satisfying $1 \cdot x = x$ and $g \cdot h \cdot x = gh \cdot x$. The *orbit* of x under the action is the set $G \cdot x = \{g \cdot x \mid g \in G\}$, and the *stabilizer* of x is the subgroup $G_x \leq G$ of elements $g \in G$ such that $g \cdot x = x$.

The group S_n is the one acting maximally transitively on the set $[1, n]$, that is, it contains exactly the permutations of $[1, n]$. For $f, g \in S_n$, we define $f \circ g$ by $(f \circ g)(a) = f(g(a))$ where $a \in [1, n]$. The direct union of all such groups is called S_∞. Every finite group embeds in S_n for some $n \in \mathbb{N}$ by Cayley's theorem.

2.2 Amenability, Cayley Graphs and Growth

Amenability is a notion which is extremely important in group theory.[13] We say G is *amenable* if it admits a *Følner sequence*, that is, a sequence $A_1 \subset A_2 \subset A_3 \subset \cdots$ such that each $A_i \subset G$ is finite, $G = \bigcup_i A_i$ and

$$\forall g \in G : \frac{|gA_i \Delta A_i|}{|A_i|} \underset{i \to \infty}{\longrightarrow} 0.$$

Amenability is equivalent to (and often defined as) the existence of a *left-invariant mean* on G, that is, a functional $\nu : \ell^\infty(G) \to \mathbb{R}$ (where $\ell^\infty(G)$ are the bounded real-valued functions on G) satisfying $\nu(1_G) = 1$ (where 1_G is the constant-1 function on G) and

$$\forall f \in \ell^\infty(G) : (\forall g \in G : f(g) \geq 0) \implies \nu(f) \geq 0.$$

[13] It is also very important in dynamics: as noted in the introduction, it is needed in the direct generalization of entropy to group actions.

All finite groups are amenable, as we can choose $A_i = G$ for all i, and $\mathbb{Z}^d$ is amenable because we can choose $A_i = [-i, i]^d$ as a Følner sequence.[14] More generally, every abelian group is amenable. Amenable groups also have various closure properties: they are closed under subgroups, quotients, group extensions and direct unions. Thus, it is natural to define *elementary amenable groups* as the smallest class of groups which contains the finite and abelian ones, and is closed under subgroups, quotients, group extensions and direct unions (and isomorphism, naturally).

We usually explicitly discuss neither Følner sequences nor means: the following theorem summarizes our typical way to prove amenability or non-amenability.

Theorem 1. *An elementary amenability group is amenable, and a group containing a free group on two or more generators is not amenable.*

In general, an amenable group need not be elementary amenable [Gri85], and a non-amenable group need not contain a free group [Ols83].

Let $A = \{g_1, g_2, \ldots, g_k\}$ be a finite set of elements generating a group G, where $A^{-1} = A$. The *Cayley graph* of G with respect to the generators A is the directed edge-labeled countable graph with nodes G and for each $a \in A$ an edge (g, ga, a), where by (a, b, c) we mean an edge from a to b with label c. In a graph, a natural notion of distance between nodes is the length of the shortest path between them. We write $d_{G,A}$ for the distance function of G in its Cayley graph with respect to generators A. We write $B_{G,A}(n)$ for the corresponding *Cayley ball* of radius n, defined as

$$B_{G,A}(n) = \{g \in G \mid d_{G,A}(1, g) \leq n\}.$$

Changing the generators of G only changes distances by a multiplicative constant, that is, if $\langle A \rangle = \langle B \rangle = G$ then

$$\exists C > 1 : \forall g, h \in G : d_{G,B}(g, h)/C \leq d_{G,A}(g, h) \leq C d_{G,B}(g, h).$$

Thus, the growth rate of balls in all Cayley graphs of G is similar. For finitely generated groups, we define some notions giving a rough classification of this growth rate. We say a (finitely generated) group G has *polynomial growth rate* if

$$\exists k \in \mathbb{N} : \forall n : |B_{G,A}(n)| \leq n^k + k$$

for some – and thus any – set of generators A. We say it has *exponential growth rate* if

$$\exists a > 1 : \forall n : |B_{G,A}(n)| \geq a^n$$

and say it is *subexponential growth rate* if it does not have exponential growth rate. By Gromov's theorem, a group with polynomial growth is virtually nilpotent. A group of subexponential growth is amenable.

[14] That is, the Cayley balls are a Følner sequence in the abelian case – we note that this is *not* how Følner sequences look in general. For example, if the size of balls grows exponentially, then the balls are never a Følner sequence.

2.3 Decidability

Definitional Issues. In this section, we discuss mainly decidability questions regarding the relationship between a CA and its local rule, for example the decidability of reversibility. See [Kar12] for a survey of decidability in the theory of cellular automata.

To ask decidability questions about a group or a monoid, we need to fix a computational presentation of the elements. Let us choose such presentations for endomorphisms of subshifts: A *cellular automaton* on a subshift $X \subset S^{\mathbb{Z}^k}$ is a function $f : X \to X$ which has a *radius* $r \in \mathbb{N}$ and a *local rule* $F : S^{[-r,r]^k} \to S$ such that $f(x)_{\boldsymbol{v}} = F(x_{\boldsymbol{v}+[-r,r]^k})$. It is well-known that cellular automata are precisely the functions $\mathrm{End}(X)$. The local rule of a cellular automaton gives a computational presentation of the function, and thus a way to give a finite list of elements of the endomorphism monoid to an algorithm. This allows us to ask algorithmic questions about the monoid $\mathrm{End}(X)$, *on any subshift* X, even a highly uncomputable one. Note that from the local rules of $f, g : X \to X$, we can easily form *a* local rule for the composition of two cellular automata by composing the local rules in an obvious way, again no matter what the subshift is.

We make a few (mostly obvious) remarks about the canonicality and caveats of this presentation of cellular automata. Every cellular automaton, on every subshift, has an infinite family of presentations by local rules, as we can always increase its radius. However, there is always a smallest possible radius.[15] By also using a local rule $F : \mathcal{L}_{2r+1}(X) \to S$ instead of $F : S^{[-r,r]^k} \to S$ (so that we only define the cellular automaton on the words it actually sees), we obtain a unique presentation. If the language of X is decidable, that is, given $w \in S^*$ we can algorithmically check whether $w \in \mathcal{L}(X)$, then this minimization process can be done algorithmically:

Lemma 1. *Let $X \subset S^{\mathbb{Z}}$ be a subshift such that $\mathcal{L}(X)$ is a decidable language. Then, given $F : S^{[-r,r]} \to S$ defining a CA $f : X \to X$, we can compute $r' \leq r$ and $G : L \to S$ where $L \subset S^{[-r',r']}$ such that G defines the same CA f and both r' and L are minimal.*

We mainly study decidability questions on subshifts with decidable languages, so the presentations can be thought of as unique, but when the subshift is not a priori decidable, we feel it is more natural to assume the local rule given as input is not minimized.

A more subtle issue is which local rules are actually endomorphisms or automorphisms of a particular subshift X. More precisely, given a rule $F : S^{[-r,r]^k} \to S$, a CA $f : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ is defined, and we call the *well-definedness problem* the question of whether $f(X) \subset X$ holds (that is, whether f restricts to a CA on X), and the *reversibility problem* the question of whether $f|X \in \mathrm{Aut}(X)$, where

[15] We note that, except on full shifts, there is in general no minimal neighborhood, as easy examples show – nevertheless, there must be a minimal radius simply because the natural numbers are well-ordered.

$f|X$ is the restriction of f to X. The following simple example shows that neither problem is in general decidable even if the language of X is decidable, for somewhat uninteresting reasons.

Example 1. Define the CA f_n on the full shift $\{0, 1, 1', 2, 3\}^{\mathbb{Z}}$ which exhanges 10^n2 and $1'0^n2$ and otherwise behaves as the identity map. Take the subshift X of $\mathcal{L}^{-1}(0^*10^*20^*30^* + 0^*1'0^*20^*30^*)$ where additionally $1'0^n20^k3$ is forbidden if the nth Turing machine halts on the kth step. Then f_n restricts to an automorphism on X if the nth Turing machine never halts, and otherwise it is not well-defined on X – thus, both the well-definedness problem and the reversibility problem are undecidable. The language of X is clearly decidable. △

We mainly study decidability questions for cellular automata on sofic shifts in dimension one, and in this case both the well-definedness problem and the reversibility problem are decidable by basic automata theory [LM95, HMU06]. We note that the picture is different in the case of multidimensional full shifts, as checking whether a local rule corresponds to a reversible cellular automaton is undecidable [Kar90]. Since every effective subshift – in particular the example above – is a sofic shift in the next dimension [AS13, DRS12], checking well-definedness of a CA on a multidimensional sofic shift with a decidable language is undecidable. On multidimensional SFTs, a simple construction based on the undecidability of the domino problem with a seed tile, or an application of the result of [Gui11], shows that the well-definedness problem is undecidable on general SFTs. We do not know if the language can be made decidable.

Question 2. Is there a two-dimensional SFT with a decidable language where the well-definedness problem is undecidable?

Another basic issue is whether two given local rules are the same. As mentioned above, if the language of X is decidable, we can compute a canonical local rule for each CA from any given local rule. Thus, to check whether two CA are the same, we can simply compute their minimal local rules and compare them. In particular the *word problem* of $\mathrm{Aut}(X)$ – the problem of checking whether the product of a given list of automorphisms is the identity map, is decidable for all subshifts X with a decidable language.

Again, this applies in particular to one-dimensional sofic shifts. In two dimensions, SFTs need not have a decidable language, and indeed, using the undecidability of the domino problem with a seed tile, one can easily build a two-dimensional SFT X where the problem of deciding whether a given local rule represents the identity element is undecidable, even when restricted to local rules that represent reversible CA. Nevertheless we do not know whether there are two-dimensional SFTs on which the word problem of a finitely generated subgroup of the automorphism group is undecidable. See [Hoc10] for a related discussion.

Group-Related Issues. Sofar, we have mainly discussed definitional issues, which, while necessary background information, have little to do with the actual automorphism groups. The list of possible group-theoretical questions about

automorphism groups is pretty much endless: for any subshift X and any family of groups $\mathcal{G}$, we can ask whether a given finitely generated subgroup G of $\mathrm{Aut}(X)$ is in $\mathcal{G}$. For example, we are interested in the decidability of abelianity, cyclicity, finiteness and freeness of such groups. We can also ask whether the subgroup generated by them has a particular property in the larger group, such as normality or centrality.

There are also many well-known computational problems in pure group theory, which we also discuss. We already discussed the word problem.

Remark 2. Note that we defined the word problem of the automorphism group of a subshift in terms of local rules. For a finitely generated group, one can define the word problem can be defined without actually knowing what the elements of the group are: We choose a set of generators, and ask for the decidability of the word problem over those generators. The decidability of the word problem will be independent of the generators. Usually, it will be clear from context which problem we mean, and in fact they are equivalent in the situations we consider.

A question related to the word problem is the *geodesics problem*, where given a word w over a finite set of generators, we want to find the minimal word u over the same generators which represents the same group element. While the geodesics problem may take an exponentially longer time to solve than the word problem, in the sense of decidability the questions are equivalent. There are many variants of this problem, but as we do not address computational complexity in this paper, all of these questions may be thought of as restatements of the word problem. The *conjugacy problem* is the problem of, given two elements g, g' of the group, deciding whether $g = hg'h^{-1}$ for some group element h.

A common question to ask about a group, and one that turns out quite interesting in the case of automorphism groups, is the *torsion problem* of deciding, given an element of the group, whether it generates an infinite group. For example, it is known that the two-dimensional generalization $2V$ of Thompson's group V has an undecidable torsion problem [BB14], while Thompson's group V itself has a decidable torsion problem [BB14,BM14].

As noted in the introduction, many of the existing algorithmic questions about cellular automata do not fit into our framework. Namely, the usual approach to cellular automata is the study of the dynamics of the $\mathbb{N}$-action (or $\mathbb{Z}$-action) of a CA f given by $n \cdot x = f^n(x)$. A variety of dynamical and computational behaviors is observed in these systems, yet the groups and monoids generated by a single cellular automaton are not particularly interesting: they are cyclic, and thus the groups obtained are isomorphic to either $\mathbb{Z}$ or $\mathbb{Z}_n$ for some $n \in \mathbb{N}$ (and the monoids are equally simple).

2.4 Showing Finiteness of a Group of CA

It is useful to note that a group of cellular automata is finite if and only if the orbits under its action are finite (even bounded). We will use this result in Section 3 to prove some undecidability results for the automorphism group of a full shift.

Lemma 2. *Let X be a transitive subshift, and let $x \in X$ be a transitive point. Then the stabilizer of x in the action of $\mathrm{Aut}(X)$ is trivial, that is, if $f, g \in \mathrm{Aut}(X)$ and $f(x) = g(x)$, then $f = g$.*

Proof. If $f \neq g$, then $f(y)_0 \neq g(y)_0$ for some $y \in X$. Since x is transitive, $f(y)_0 = f(\sigma^n(x))_0 = g(\sigma^n(x))_0 = g(y)_0$ for some $n \in \mathbb{Z}$. □

In other words:

Theorem 2. *Let X be a transitive subshift and let G be any subgroup of $\mathrm{Aut}(X)$. Then the following are equivalent:*

- *G is infinite,*
- *$G \cdot x$ is infinite for some $x \in X$,*
- *$G \cdot x$ can be arbitrarily large for $x \in X$.*

In particular, it follows that if G is an infinite subgroup of $\mathrm{Aut}(X)$ for a transitive subshift X, then G is infinite if and only if it has an infinite orbit. In our main application, X is a full shift, and thus certainly transitive. The result is true in much more generality, but we omit the discussion of this.

3 Full Shifts and Transitive Sofic Shifts

We now look at full shifts and transitive sofic shifts, perhaps the most natural habitat of cellular automata. If the alphabet S is not explicitly specified, we assume $|S| > 1$.

One of the main tools in the positive entropy case is Lemma 3 below. It shows that it is usually enough to prove undecidability results and to perform constructions of subgroups on full shifts. A proof in the mixing SFT case, and restricted to automorphisms, appeared in [KR90]. The general proof for X a positive entropy (equivalently, uncountable) sofic shift is also easy, and outlines the idea of *marker constructions* (our version of the marker being the word u in the proof). We sketch a proof.

An *unbordered word* u is one that does not overlap itself, that is, $uw = vu \implies |v| \geq |u|$.

Lemma 3 (essentially [KR90]). *If X is a positive entropy sofic shift, then $\mathrm{End}(X)$ contains a copy of $\mathrm{End}(S^{\mathbb{Z}})$ for any alphabet S.*

Proof (Proof sketch). Every infinite aperiodic word contains arbitrarily long unbordered words, by Theorem 8.3.9 in [Lot02].[16] Choose a positive entropy transitive sofic Y subshift inside the X, and a long unbordered word u that occurs in Y. If u is taken long enough, there are many words uvu where v is of length $\lfloor |u|/2 \rfloor$,[17] and by the pigeonhole principle (and long enough u), many

[16] A stronger version of this is shown in Lemma 2.2 of [BLR88].

[17] This follows because every transitive sofic shift has a uniform distance k such that if u and v occur in the language, then uwv occurs for some w of length at most k. This is a direct application of the pigeonhole principle.

such words which correspond to the same element of the *syntactic monoid*[18] of $\mathcal{L}(X)$. Since u is unbordered, a local rule can safely change the word v between two occurrences of u, that is, compute a local transformation $uvu \mapsto uv'u$.

More precisely, let V with $|V| = |S|^2$ be the a set of words v of length $\lfloor |u|/2 \rfloor$ such that uvu is a word of X, and the words uvu for $v \in V$ correspond to the same element of the syntactic monoid. Choose a bijection $\pi : V \to S^2$. Now, we map each $f \in \mathrm{End}(S^{\mathbb{Z}})$ to a CA $g \in \mathrm{End}(X)$ with the following behavior: on the points $\cdots uv_{-2}uv_{-1}uv_0uv_1uv_2u \cdots$ where $v_i \in V$ for all i, we apply $f \times f^R$ to the point $\cdots \pi(v_{-2})\pi(v_{-1})\pi(v_0)\pi(v_1)\pi(v_2) \cdots \in (S^2)^{\mathbb{Z}}$ (which we think of as two separate tracks each containing a point over the full shift $S^{\mathbb{Z}}$), where $y^R = \cdots y_2y_1.y_0y_{-1}y_{-2} \cdots$ and $f^R(y) = f(y^R)^R$. On points where no such sequence appears, g is the identity map. When a partial such sequence occurs, we glue the two tracks together like a conveyor belt at the end of the sequence, and think of the first track turning 180 degrees and continuing backwards on the second track. It is easy to check that this gives a consistent embedding of the endomorphism monoid $\mathrm{End}(S^{\mathbb{Z}})$ to $\mathrm{End}(X)$. □

We note that we do *not* claim that $\mathrm{Aut}(X)$ embeds in $\mathrm{Aut}(Y)$ when X and Y are general mixing SFTs. We suspect this to be the case, but subtle problems seem to arise when attempting to generalize the proof above. See [KR90] for an example of this.

3.1 Subgroups of Aut(X) for a Transitive Sofic Shift X

In this section, we give some examples of what kind of subgroups can be created with the marker construction, and end the section with a complete characterization of the locally finite subgroups that appear in the automorphism group [KR90]. Most of the results in this section were essentially proved in [BLR88] or [KR90].

While we know no restrictions on the groups $\mathrm{Aut}(X)$ for general subshifts X, there are some restrictions when X has suitable dynamical properties.

Lemma 4 ([BLR88]). *Let X be a subshift where periodic points are dense (for example, a transitive sofic shift). Then* $\mathrm{Aut}(X)$ *is residually finite.*

Since the class of residually finite groups is closed under taking subgroups, we obtain nontrivial restrictions on the possible subgroups that can occur in $\mathrm{Aut}(X)$ when X has periodic points dense. For example, $\mathrm{Aut}(X)$ cannot contain a nontrivial divisible subgroup such as $(\mathbb{Q}, +)$, and cannot contain the infinite permutation group S_∞.

The decidability of the word problem restricts the possible subgroups further, as we have already seen:

[18] We omit the definition, but u and v corresponding to the same element of this monoid means exactly that they occur in the same contexts, so we may exchange them. See [HMU06].

Lemma 5. *If X is a subshift with a decidable language (for example, a sofic shift), then every finitely generated subgroup of* $\mathrm{Aut}(X)$ *has a decidable word problem.*

Note that in the proof of Lemma 3, the CA in the image of the embedding only change the points in subwords of the form $\cdots uv_{-1}uv_0uv_1u\cdots$. Since u can be taken arbitrarily long, the CA thus change only points in a subshift of strictly smaller entropy. Using standard techniques, we can make sure that the complement of this subshift still contains a positive entropy sofic shift. Using this idea, we obtain the following generalization.

Lemma 6 (Essentially Theorem 2.6 in [BLR88]). *If X is a positive entropy sofic shift (equivalently, uncountable), then* $\mathrm{End}(X)$ *contains a copy of the countable direct sum* $\bigoplus_{i\in\mathbb{N}} \mathrm{End}(S^{\mathbb{Z}})$ *for any alphabet S.*

Equivalently, we can have countably many distinct alphabets, by Lemma 3. As a direct corollary, we obtain two closure properties for subgroups of automorphism groups of full shifts.

Theorem 3. *The set of subgroups of* $\mathrm{Aut}(S^{\mathbb{Z}})$ *is closed under countable direct sums for any alphabet S.*

Of course, for this to be interesting, we need to have some subgroups to begin with. A trivial observation is that σ generates a copy of $\mathbb{Z}$ on any infinite subshift. Another observation, essentially Theorem 6.13 in [Hed69], is that every finite group embeds in $\mathrm{Aut}(X)$ for a positive entropy sofic shift. Using Lemma 3, this is very easy to show: a finite group G embeds in the permutation group S_k for some k, and thus into $\mathrm{Aut}([1,k]^{\mathbb{Z}})$ by symbol permutations.

Proposition 1. *If X is a positive entropy sofic shift, then* $\mathrm{Aut}(X)$ *contains every countable direct sum of finite groups and copies of* $\mathbb{Z}$.

In [BLR88], it is shown that the free group with infinitely many generators embeds in the automorphism group of a mixing SFT. Applying Lemma 3, we obtain the same result for the automorphism group of a positive entropy sofic shift.

Theorem 4. *If X is a positive entropy sofic shift, then* $\mathrm{Aut}(X)$ *contains* F_∞.

Corollary 1. *If X is a positive entropy sofic shift, then* $\mathrm{Aut}(X)$ *is not amenable.*

To prove Theorem 4, one embeds the free group $\mathbb{Z}_2 * \mathbb{Z}_2 * \mathbb{Z}_2$ in the automorphism group using a marker construction. In [KR90], it is attributed to R. C. Alperin that more generally any free product of finitely many finite groups embeds in the automorphism groups. We make a slightly bolder conjecture:

Conjecture 1. If X is a positive entropy sofic shift, $\mathrm{End}(X)$ contains the copy of the free product of countably many copies of $\mathrm{End}(S^{\mathbb{Z}})$ for any alphabet S.

Restricted to locally finite groups G, a full characterization of the subgroups of $\mathrm{Aut}(X)$ is known. As observed in the beginning of this section, the automorphism group of a transitive sofic shift is residually finite. This is the only requirement:

Theorem 5 ([KR90]). *Let X be a positive entropy sofic shift. Then a locally finite group G is isomorphic to a subgroup of the automorphism group of X if and only if G is residually finite and countable.*

The paper [KR90] contains many more examples of subgroups that can be embedded (for example, fundamental groups of 2-manifolds), and proves that the set of subgroups of $\mathrm{Aut}(X)$ is closed under finite extensions when X is a full shift. That is, if $H \leq \mathrm{Aut}(X)$ where X is a full shift, and $[G : H] < \infty$, then $G \leq \mathrm{Aut}(X)$.

Note that by Lemma 3, the groups $\mathrm{Aut}(\{0,1\}^{\mathbb{Z}})$ and $\mathrm{Aut}(\{0,1,2\}^{\mathbb{Z}})$ embed into each other, and thus have the same subgroups. However, we do not know whether there is an isomorphism between them.[19] This is one of the open problems in symbolic dynamics listed in [Boy08].

Question 3. Are $\mathrm{Aut}(\{0,1\}^{\mathbb{Z}})$ and $\mathrm{Aut}(\{0,1,2\}^{\mathbb{Z}})$ isomorphic?

It follows from Ryan's theorem that $\mathrm{Aut}(\{0,1\}^{\mathbb{Z}})$ and $\mathrm{Aut}(\{0,1,2,3\}^{\mathbb{Z}})$ are not isomorphic, because in $\mathrm{Aut}(\{0,1,2,3\}^{\mathbb{Z}})$ that shift map has a square root, while it does not have one in $\mathrm{Aut}(\{0,1\}^{\mathbb{Z}})$.

3.2 Decidability on Positive Entropy Sofic Shifts

Most decidability problems for cellular automata are about their dynamical properties, such as mixing, transitivity, sensitivity and expansivity. These questions are, at least a priori, outside our scope, as there is no known algebraic property satisfied by, for example, the mixing CA, but not the rest.

However, the dynamical notion of *equicontinuity* turns out to be equivalent to eventual periodicity on all subshifts. For automorphisms $f \in \mathrm{Aut}(S^{\mathbb{Z}})$, this is the question of whether $f^k = \mathrm{id}_X$ for some $k \geq 1$, that is, the torsion problem. Though the result is not stated in group-theoretic terms, in [KO08], this problem is shown undecidable on full shifts. Using Lemma 3,[20] we obtain the result for all positive entropy sofic shifts.

Theorem 6 ([KO08]). *Let X be a positive entropy sofic shift. Then the group $\mathrm{Aut}(X)$ has an undecidable torsion problem.*

This trivially implies many undecidability problems, such as whether a given finite set of elements generates an infinite group, or whether it generates a torsion group. We now prove some slightly more interesting corollaries.

[19] There certainly are non-isomorphic groups that embed into each other, for example the free groups F_2 and F_3 are such a pair.

[20] We also need to note that the embedding is explicitly computable – clearly it is.

The notion of time-symmetry was introduced for cellular automata in [GKM12]. We give this definition for an arbitrary subshift: a CA is *time-symmetric* if it is the composition of two *involutions*, where an involution is a CA $g \in \mathrm{Aut}(X)$ satisfying $g^2 = \mathrm{id}_X$. That is, $f \in \mathrm{Aut}(X)$ is time-symmetric $f = g \circ h$ for some $g, h \in \mathrm{Aut}(X)$ satisfying $f^2 = g^2 = \mathrm{id}_X$. The name comes from the equivalent condition that $g \circ f \circ g = f^{-1}$ for some involution $g \in \mathrm{Aut}(X)$. It is shown in [GKM12] that time-symmetric CA are *intrinsically universal* among the reversible cellular automata, that is, every reversible automaton can be simulated by a time-symmetric one. The proof of this claim gives the following theorem:[21]

Theorem 7. *Given a finite set of finite order automorphisms of a positive entropy sofic shift X, it is undecidable whether they generate a finite group.*

Proof. Again, it is enough to prove the result for full shifts by Lemma 3. We show that an algorithm for this problem would give an algorithm for the torsion problem as well. Let $f \in \mathrm{Aut}(S^{\mathbb{Z}})$ be given. Consider the full shift $(S \times S)^{\mathbb{Z}}$, and define $g, h \in \mathrm{Aut}((S \times S)^{\mathbb{Z}})$ by $g(x, y) = (f(y), f^{-1}(x))$ and $h(x, y) = (y, x)$. Then $(g \circ h)(x, y) = (f(x), f^{-1}(y))$. If $\langle f \rangle$ is infinite, then the orbit of some point x is infinite by Theorem 2, and thus the orbit of (x, y) is of infinite order by Theorem 2 for any $y \in S^{\mathbb{Z}}$. Let then $|\langle f \rangle| = k$, and consider a point (x, y). Clearly the orbit of (x, y) under the action of $\langle g, h \rangle$ is contained in the finite set $\{(f^i(x), f^j(y)), (f^i(y), f^j(x)) \mid i, j \in \mathbb{Z}\}$. Since every orbit is finite, $\langle g, h \rangle$ is finite, again by Theorem 2. □

The proof shows more precisely that given two automorphisms of order 2, it is undecidable whether they generate a finite or an infinite group. More precisely, in the construction, depending on whether f halts, we obtain either a finite dihedral group D_n, or the infinite dihedral group $D_\infty = \mathbb{Z}_2 * \mathbb{Z}_2$. One can easily perform a similar proof to for example show that the finiteness of a group generated by automorphisms of order 3 is undecidable. We prove a more general result of this form.

Theorem 8. *Let X be a positive entropy sofic, and let G be an arbitrary nonempty finite group. Then, given two finite subgroups $F, F' \leq \mathrm{Aut}(X)$ with $F \cong F' \cong G$, it is undecidable whether $\langle F \cup F' \rangle$ is finite.*

Proof. Again, we only need to prove the claim on full shifts. Let $f \in \mathrm{Aut}(S^{\mathbb{Z}})$ be arbitrary. We again show that an algorithm for the problem in the statement also decides whether $\langle f \rangle$ is finite. Without loss of generality, we may assume G is a subgroup of a symmetric group S_k, so that G acts nontrivially on $[1, k]$. Let $F = \{f_g \mid g \in G\}$ be the subgroup of $\mathrm{Aut}((S^k)^{\mathbb{Z}})$ permuting the tracks according to this embedding:

$$f_g(x_1, \ldots, x_k) = (x_{g^{-1}(1)}, x_{g^{-1}(2)}, \ldots, x_{g^{-1}(k)}).$$

[21] In fact, the CA constructed in [KO08] are already time-symmetric, but the idea on permuting tracks illustrates Theorem 2 better.

We now give another embedding, F', where in addition to permuting the tracks, we apply a power of f to the tracks when they are moved. For this, choose a function $c : [1, k] \to \mathbb{Z}$. The idea is that if track i is moved to track j by a permutation $g \in G$, the corresponding CA f'_g will apply $f^{c(j)-c(i)}$ to the ith track x_i before moving it. Thus, the jth track will always be $c(j) - c(i)$ steps ahead in time compared to the track i. This gives another embedding of G to $\mathrm{Aut}((S^k)^{\mathbb{Z}})$, since if $g_1 \cdot g_2 \cdots g_\ell = 1$ for $g_i \in G$, then $f'_{g_1} \circ f'_{g_2} \circ \cdots \circ f'_{g_\ell}$ permutes every track to its starting position, and naturally the movements in time cancel out, so that $f'_{g_1} \circ f'_{g_2} \circ \cdots \circ f'_{g_k} = \mathrm{id}_{(S^k)^{\mathbb{Z}}}$.

More precisely, take a function $c : G \to \mathbb{Z}$ and define $C : G \times [1, k] \to \mathbb{Z}$ by

$$C(g, i) = c(g \cdot i) - c(i).$$

Define $F' = \{f'_g \mid g \in G\}$ by

$$f'_g(x_1, \dots, x_k) = (f^{C(g,g^{-1}\cdot 1)}(x_{g^{-1}\cdot 1}), \dots, f^{C(g,g^{-1}\cdot k)}(x_{g^{-1}\cdot k})).$$

It is easy to check that $g \mapsto f'_g$ gives an embedding of G into $\mathrm{Aut}((S^k)^{\mathbb{Z}})$.[22]

We claim that if c is injective, then $\langle F \cup F' \rangle$ is finite if and only if $\langle f \rangle$ is. Consider thus an arbitrary point $(x_1, x_2, \dots, x_k) \in (S^k)^{\mathbb{Z}}$. First, if $\langle f \rangle$ is finite, we are done: every point in the orbit of $(x_1, x_2, \dots, x_k)$ has a point from the orbit of one of the points x_i on each track, which gives a finite upper bound on the size of orbits. Conversely, suppose x has an infinite orbit in the action of f. Let $g \cdot i = j$ for $g \in G$ and $i, j \in [1, k]$ with $i \neq j$ (the action of G is nontrivial on $[1, k]$). Since c is injective, $C(g, i) = c(g \cdot i) - c(i) = n \neq 0$. For notational simplicity, suppose $i = 1$ and $j = 2$. Then

$$(x, y, \cdots) \overset{f'_g}{\mapsto} (y', f^n(x), \cdots) \overset{f_{g^{-1}}}{\mapsto} (f^n(x), y'', \cdots)$$

for some points $y, y', y'' \in S^{\mathbb{Z}}$. Clearly, this shows that the orbit of any point $(x, \cdots)$ is infinite.

Again, the alphabet S^k can be changed by applying Lemma 3. □

There are many open questions about automorphism groups, even on full shifts. For example, the decidability of time-symmetry is open in one dimension (in two dimensions, it is undecidable [GKM12]).

Question 4. Is it decidable whether a given CA $f \in \mathrm{Aut}(S^{\mathbb{Z}})$ is time-symmetric?

More generally, we do not know whether it is decidable if a given CA is generated by involutions, or elements of finite order. For a more general question in the same spirit, see the FOG conjecture (not true in general [KR91]) and virtual FOG conjecture in [Boy08]. Another question whose solution we do not

[22] Readers familiar with cohomology will notice that c is just an arbitrary 0-cochain $c \in \mathrm{Hom}(C(G, [1, k]), \mathbb{Z})$ for the action of G on $[1, k]$ and C is the corresponding 1-coboundary – in particular, C is a 1-cocycle, from which it follows that the action of F' is well-defined.

know is the conjugacy problem: is it decidable in $\mathrm{Aut}(S^{\mathbb{Z}})$ whether two given elements f, g are conjugate?

More in line with Theorem 8 and our constructions in this section, we state the following conjecture.

Conjecture 2. It is undecidable whether two given automata generate a (non-abelian) free group.

If Conjecture 1 is true, and the embedding is computable, then the previous conjecture is true as well: given $f \in \mathrm{Aut}(S^{\mathbb{Z}})$, consider the CA $g, h \in \mathrm{Aut}(S^{\mathbb{Z}})$ given by the embedding of $\mathrm{Aut}(S^{\mathbb{Z}}) * \mathrm{Aut}(S^{\mathbb{Z}})$ into $\mathrm{Aut}(S^{\mathbb{Z}})$, so that $\langle f \rangle \cong \langle g \rangle \cong \langle h \rangle$ and g and h generate the product $\langle g \rangle * \langle h \rangle$ in $\mathrm{Aut}(S^{\mathbb{Z}})$. Then clearly g and h generate a group isomorphic to F_2 if and only if f has infinite order.

A simple decidable property is abelianness: to check whether a finite set F of cellular automata generate an abelian group, we only need to check whether the identity $f \circ g = g \circ f$ holds for all pairs $f, g \in F$.

4 Countable Subshifts

The simplest countable subshifts (and the simplest subshifts in general) are probably the finite ones. A finite subshift is always an SFT, and the automorphism groups of such SFTs are simply the centralizers of permutations on finite sets. The following characterization was given in [CK14]: Let S_n act on $\mathbb{Z}_m^n$ by $\phi(g)(i_1, i_2, \ldots, i_n) = (i_{g^{-1}(1)}, i_{g^{-1}(2)}, \ldots, i_{g^{-1}(n)})$ for $g \in S_n$.[23] Define the semidirect product $S(m, n) = \mathbb{Z}_m^n \rtimes S_n$ with respect to the action ϕ. For future purposes, similarly define $S(\infty, n) = \mathbb{Z}^n \rtimes S_n$.

Theorem 9. *A group G is the automorphism group of some finite subshift if and only if*

$$G \cong S(m_1, n_1) \times S(m_2, n_2) \times \cdots \times S(m_s, n_s)$$

for some $m_1 < m_2 < \cdots < m_s$ and $n_1 < n_2 < \cdots < n_s$.

Equivalently, these are precisely the finite groups that occur as automorphism groups of subshifts, since σ generates an infinite subgroup of $\mathrm{Aut}(X)$ if X is infinite.

In [ST12], cellular automata on countable sofic shifts were discussed from the point of view of computability. Unlike in the case of full shifts, many dynamical behaviors (though not all!) of such automata are decidable. In particular, the following is proved:

Theorem 10 ([ST12]). *Let X be a countable sofic shift. Then the torsion problem is decidable for* $\mathrm{Aut}(X)$.

[23] In [CK14], the dual definition is used. Our definition must be used to obtain a homomorphism when the composition of permutations is defined by $(\pi \circ \pi')(a) = \pi(\pi'(a))$ (but also the dual definition is used by some authors).

While undecidability results about the *dynamics* of cellular automata on countable sofic shifts are shown in [ST12], we do not know any interesting undecidability results about the automorphism group in the countable case.

This suggests that the automorphism group might be essentially simpler in the countable case, and in a way it is: While Corollary 1 shows that the automorphism group of an uncountable sofic shift is never amenable, the author and Michael Schraudner are working on the proof that the automorphism group of a countable sofic shift always is. We prove this for a simple example.

Proposition 2. *For $k \in \mathbb{N}$, let $X_k \subset \{0,1\}^{\mathbb{Z}}$ be the (countable sofic) subshift whose forbidden words form the regular language $(10^*)^k 1$. Then $\mathrm{Aut}(X_k)$ is elementarily amenable.*

Proof. We prove this by induction on k. The base case is the one-point subshift X_0 whose automorphism group is the trivial group, which is certainly elementary amenable. Now, let $k > 0$. The isolated points of X_k are exactly the ones containing k 1-symbols. A homeomorphism must map isolated points to isolated points, so if $f \in \mathrm{Aut}(X_k)$, then the restriction $f|X_{k-1}$ is well-defined. The map $\phi : f \mapsto f|X_{k-1}$ is a homomorphism from $\mathrm{Aut}(X_k)$ to $\mathrm{Aut}(X_{k-1})$. Its image is elementary amenable by induction, so we only need to show that $\ker(\phi)$ is as well.

For this, let r be the common radius of $f, f^{-1} \in \ker(\phi)$. Let Y_r be the set of points y in X that contain k 1-symbols, which all occur in a single subword of y of length $2r+1$. We claim that if $x \notin Y_r$, then $f(x) = x$. Otherwise, $x_i \neq f(x)_i$ for some i. If x contains less than k 1-symbols, then f is not in the kernel of ϕ. Otherwise, because $x \notin Y_r$, $x_{[i-r,i+r]}$ cannot contain all the 1-symbols. In particular,the point y with $y_{[i-r,i+r]} = x_{[i-r,i+r]}$ and $y_j = 0$ for $j \notin [i-r, i+r]$ is in X_{k-1}. But $f(y)_i \neq y_i$, so again f cannot be in the kernel. This contradiction shows that only points in Y_r can be changed. The same reasoning applies to f^{-1}, so the set Y_r is invariant under the action of f. In other words, the action of f permutes Y_r and leaves every point in $X \setminus Y_r$ invariant.[24]

Now, let F_r be the subgroup of $\ker(\phi)$ that only permutes the points in Y_r. This is clearly a subgroup, and $\ker(\phi) = \bigcup_{r \in \mathbb{N}} F_r$. Thus, it is enough to show that F_r is elementary amenable. For this, observe that the set Y_r consists of finitely many orbits: $Y_r = \mathcal{O}(x_1) \cup \mathcal{O}(x_2) \cup \cdots \mathcal{O}(x_n)$ for some n and $x_i \in X$. A permutation of Y_r can, by shift-commutation, only permute the tracks and shift them around. It is then easy to show that F_r embeds in the group $\mathbb{Z}^n \rtimes S_n$, which is elementary amenable as a semidirect product of amenable groups. □

On the other hand, unlike the automorphism group of a transitive sofic shift, the automorphism group of a countable one need not be residually finite:

Proposition 3. *There exists a countable sofic shift X with $S_\infty \leq \mathrm{Aut}(X)$. In particular, $\mathrm{Aut}(X)$ is not be residually finite.*

[24] It is a general fact that if a group action on A maps $B \subset A$ to $C \subset B$, then it maps B exactly onto itself, and also $A \setminus B$ onto itself.

Proof. An example is $X_2 = \mathcal{L}^{-1}(0^*10^*10^*)$. If $g \in S_\infty$, define $f_g : X_2 \to X_2$ by

$$f_g(^\infty 0.10^n 10^\infty) = {}^\infty 0.10^{g(n)}10^\infty.$$

Since g has finite support, f_g simply permutes finitely many (orbits of) isolated points and leaves everything else invariant. Continuity and shift-commutation are clear, and it is easily verified that $g \mapsto f_g$ embeds S_∞ into $\mathrm{Aut}(X_2)$. □

Every countable subshift has zero entropy, and it seems likely that this puts severe restrictions on the automorphism group. For example, the only ways to embed free groups into automorphism groups that the author is aware of generate entropy. Nevertheless, from just the assumption that X is countable, we are not able to prove any properties for $\mathrm{Aut}(X)$.

Question 5. If G is the automorphism group of a subshift, is it also the automorphism group of a countable subshift? Are automorphism groups of countable subshifts amenable? Can they contain a copy of F_2?

5 Minimal Subshifts and Subshifts of Low Complexity

A *substitution* is a function $\tau : S \to S^+$. We can apply such a map τ also to finite words by $\tau(w) = \tau(w_0)\tau(w_1)\cdots\tau(w_{|w|-1})$. Suppose τ is *primitive*, that is, $\exists n : \forall a, b \in S : \exists i : \tau^n(b)_i = a$. For $a \in S$ let $L_a = \{\tau^n(a) \mid n \in \mathbb{N}\}$. Then

$$X_\tau = \mathcal{L}^{-1}(L_a),$$

for any a, is the subshift generated by τ. The substitution τ is *binary* if $S = \{0, 1\}$ and *constant-length* if $\exists n : \forall a \in S : |\tau(a)| = n$. We say τ has a *coincidence* if $\exists i : \tau(a)_i = \tau(b)_i$ for all $a, b \in S$.

The subshifts X_τ for primitive τ are always minimal. Due to their rigid self-similar structure, one can often precisely compute their automorphism groups. To our knowledge, the first explicit result was the following.

Theorem 11 ([Cov71]). *Let τ be a binary primitive constant-length substitution. If τ has a coincidence, then $\mathrm{Aut}(X_\tau) = \langle\sigma\rangle$. Otherwise, $\mathrm{Aut}(X_\tau) = \langle\sigma\rangle \times f$, where f is the* binary flip CA *defined by $f(x)_i = 1 - x_i$.*

This was generalized in [HP89] to the non-binary case. We state only a weak form of the theorem.

Theorem 12 ([HP89]). *Let τ be a primitive constant-length substitution. Then $\mathrm{Aut}(X_\tau)$ is virtually $\mathbb{Z}$.*

It is also shown in [HP89] that this is close to optimal, as $\mathrm{Aut}(X_\tau)$ can be of the form $G \times \mathbb{Z}$ for any finite group G. (This construction can also be found in [DDMP14].) In [ST13], we generalized this result to all balanced substitutions – ones where $|\tau^n(a)| = a^n + d(n)$ where $a > 1$ and d is a bounded function.

A further generalization is the class of linearly recurrent minimal subshifts, and even more general are the minimal subshifts with *linear (factor) complexity*: the *word complexity* of X is the function $n \mapsto p_n(X) = |\mathcal{L}_n(X)|$, and X has linear factor complexity if $\exists C : \forall n : p_n(X) \leq Cn + C$. We asked in [ST14] whether it is true in general that linear factor complexity on a minimal subshift implies virtually $\mathbb{Z}$. It quickly turned out that the answer is 'yes':

Theorem 13 ([CY14, CK14, DDMP14]). *The automorphism group of an infinite minimal subshift with linear factor complexity is virtually* $\mathbb{Z}$.

In fact the result turned out to be, in some sense, folklore, though not explicitly published before. It can be proved quite quickly by using known properties of asymptotic points in the linear complexity case.

The result of [CK14] is stated more generally for transitive subshifts, and all papers show more general results, in different directions. For cellular automata, we obtain in particular that for some k, every reversible CA $f : X \to X$ on an infinite minimal subshift with linear factor complexity satisfies $f^k = \sigma^n$ for some $n \in \mathbb{Z}$. It is quite easy to see that a virtually $\mathbb{Z}$ group has a decidable word problem and a decidable torsion problem in the purely group-theoretical sense. Using the folklore result that a Π_1^0 minimal subshift has a decidable language (see [Sal14b]), we see that these problems are even uniformly decidable in the following sense.

Theorem 14. *Given a Turing machine T enumerating the forbidden patterns of an infinite minimal subshift X and a cellular automaton $f : X \to X$, we can decide whether $f = \mathrm{id}_X$. If X has linear factor complexity, whether $\exists n \geq 1 : f^n = \mathrm{id}_X$ is decidable as well.*

Proof. An algorithm for checking $f = \mathrm{id}_X$ follows directly from the decidability of the language of x, so in particular the word problem is decidable. As for the torsion problem, iterating f, we obtain local rules for f^n for all n. By the assumption, $f^n = \sigma^m$ for some n, m. Since the word problem is decidable, it is easy to find such n and m algorithmically.[25] If $m = 0$, the answer is 'yes'. Otherwise it is 'no'. □

A result analogous to Theorem 13 can be shown for the endomorphism monoid when the subshift is also *linearly recurrent*, that is, there exists n such that every word $u \in \mathcal{L}(X)$ that appears in every word of $\mathcal{L}_{n|u|}(X)$ (see [DHS99] for more on this concept). Namely, it is known that in this case the subshift is *coalescent*, that is, $\mathrm{Aut}(X) = \mathrm{End}(X)$ [Dur00].

We note that while it is known that the automorphism group is virtually $\mathbb{Z}$ for a linear growth minimal subshift (in fact [CK14] shows the subtly stronger result, that it is a semidirect product of a finite group and $\mathbb{Z}$), and groups of the form $G \times \mathbb{Z}$ all occur as automorphism groups, we do not know what the precise class of automorphism groups is even in the linear growth case. The answer is presumably right around the corner, but we don't know it:

[25] For this, the assumption that f is indeed a cellular automaton on X is crucial. Given a local rule not defining such a CA, it is not clear what can be said.

Question 6. Which groups appear as $\mathrm{Aut}(X)$ for minimal subshifts X with linear factor complexity?

Another class of subshifts that has been studied are the ones with subquadratic factor complexity (for all $C > 0$, we have $p_n(X) < Cn^2$ for large enough n):

Theorem 15 ([CK14]). *Every cellular automaton on a transitive subshift of subquadratic complexity is a root of a shift map.*

In other words, if $f : X \to X$ is a CA and X is transitive and of subquadratic complexity, then $f^k = \sigma^n$ for some $k > 0, n \in \mathbb{Z}$. Unlike in the case of linear complexity, there is no uniform bound on the k, that is, for a transitive subshift X of subquadratic complexity, if $k(f)$ is the least positive integer such that $f^{k(f)} \in \{\sigma^n \mid n \in \mathbb{Z}\}$, then $k : \mathrm{End}(X) \to \mathbb{N}$ may be unbounded. An explicit example of such a subshift is given in [Sal14a].

Theorem 16 ([Sal14a]). *There exists a minimal Toeplitz subshift X with subquadratic complexity whose automorphism group is not finitely generated:*

$$\mathrm{Aut}(X) \cong \left\langle \left(\frac{2}{5}\right)^i \mid i \in \mathbb{N} \right\rangle \leq (\mathbb{Q}, +).$$

Of course, Theorem 14 extends to the subquadratic minimal case by the same proof. Presumably it does not generalize to all minimal subshifts, but we have no examples.

Question 7. Is there a minimal subshift with a decidable language whose automorphism group does not have a decidable word problem? Can the automorphism group have a finitely generated subgroup whose word problem is undecidable?

In [BLR88], a minimal subshift is constructed whose automorphism group contains a copy of $\mathbb{Q}$. There is much freedom in the construction, and they explain how the group could be made precisely $\mathbb{Q}$.

Theorem 17 ([BLR88]). *There is a minimal subshift X with $\mathrm{Aut}(X) \cong \mathbb{Q}$.*

It seems likely that one can also modify the construction so that the subshift has subquadratic factor complexity. With a similar construction, it seems that one can also obtain for example S_∞ as the automorphism group of a subquadratic growth. Nevertheless, we have no conjecture what the characterization is.

All the examples above are locally virtually cyclic: every finitely generated subgroup is virtually cyclic. This is not always the case on minimal subshifts, as shown by the following example (although the group is still locally virtually abelian):

Proposition 4. *[DDMP14] For any $d \in \mathbb{N}$, there exists a minimal subshift X with $\lim_{n\to\infty} p_X(n)/n^{d+1} = 0$ and $\mathrm{Aut}(X) \cong \mathbb{Z}^d$.*

The following limitation is shown in [DDMP14] in the case that recurrence times of words are polynomial (which is a stronger assumption than polynomial complexity). For a subshift X, define

$$N_X(n) = \inf\{m \mid w \in \mathcal{L}_m(X) \implies \forall u \in \mathcal{L}_n(X) : u \text{ occurs in } w\}.$$

Theorem 18 ([DDMP14]). *Let X be a transitive subshift such that*

$$\sup_{n \geq 1} N_X(n)/n^d < \infty$$

for $d \geq 1$. Then, there is a constant C depending only on d, such that any finitely generated subgroup of $\mathrm{Aut}(X)$ is virtually nilpotent of step at most C.

Like in the case of countable subshifts, we are not aware of any general results about the possible automorphism groups of minimal subshifts.

Question 8. Which groups occur as automorphism groups of minimal subshifts? In particular, if G is the automorphism group of a subshift, is it also the automorphism group of a minimal subshift? Can F_2 occur as a subgroup? Are the automorphism groups of minimal subshifts amenable?

Let $\mathcal{G}_M$ (resp. $\mathcal{G}'_M$) be the family of groups $\mathrm{Aut}(X)$ (resp. $\mathrm{Aut}(X)/\langle\sigma\rangle$) for minimal subshifts X. Especially in conjunction with the case of coded subshifts, the following question is particularly interesting:

Question 9. Is $\mathcal{G}_M$ closed under subgroups? Is $\mathcal{G}'_M$? More generally, what closure properties do they have?

6 Coded and Synchronized Systems

We briefly describe two of the results shown in [FF96] about automorphism groups of two families of transitive subshifts, namely the coded and synchronized systems. A word $w \sqsubset X$ is *synchronizing* if $uw, wv \sqsubset X \implies uwv \sqsubset X$. This means, in some sense, that no information travels over w. A *synchronized system* is a transitive subshift with a synchronizing word. The following construction of synchronized systems is shown in [FF96]:

Theorem 19 ([FF96]). *Given any subshift X with periodic points dense, there is a synchronized system Y such that $\mathrm{Aut}(Y)$ contains a copy of $\mathrm{Aut}(X)$.*

A *coded system* is a subshift X of the form $X = \mathcal{L}^{-1}(W^*)$, where W is any countable set of words over a finite alphabet. In other words, points of X are the limit points of infinite concatenations of words in W. Every synchronized system is coded, but the converse does not hold.

There is much freedom in the construction of automorphism groups of coded systems, as shown by the following strong result.

Theorem 20 ([FF96]). *If X has dense periodic points and $G \leq \mathrm{Aut}(X)$, then there is a coded system Y with $\mathrm{Aut}(Y) \cong G \times \mathbb{Z}$, where the isomorphism maps σ to $(1_G, 1)$.*

We note that this allows the exact construction of automorphism groups, not only subgroups of them, which separates coded systems from the families discussed in the previous sections (at least when it comes to known results). In particular, all finitely generated abelian groups can be obtained exactly as automorphism groups of coded systems. Since coded systems themselves have periodic points dense, the theorem also gives a kind of closure property for their automorphism groups.

Corollary 2. *Let $\mathcal{G}'_C$ be the set of groups G such that $\mathrm{Aut}(X) \cong G \times \mathbb{Z}$ for some coded subshift X. Then $\mathcal{G}'_C$ is closed under taking subgroups.*

We are not aware of a similar closure property for any other natural class of subshifts.

The family of coded systems contains only subshifts with periodic points dense, and thus they have only residually finite automorphism groups. There are some additional restrictions:

Lemma 7 ([FF96]). *The residually finite group $\mathbb{Z}[1/2]$ is not the automorphism group of any coded system.*

Again, we do not know what the precise class of groups that occur is.

References

[AS13] Aubrun, N., Sablik, M.: Simulation of effective subshifts by two-dimensional subshifts of finite type. Acta Appl. Math. **126**(1), 35–63 (2013)

[BB14] Belk, J., Bleak, C.: Some undecidability results for asynchronous transducers and the Brin-Thompson group 2V. ArXiv e-prints, May 2014

[BLR88] Boyle, M., Lind, D., Rudolph, D.: The automorphism group of a shift of finite type. Transactions of the American Mathematical Society **306**(1), 71–114 (1988)

[BM14] Belk, J., Matucci, F.: Conjugacy and dynamics in thompson's groups. Geometriae Dedicata **169**(1), 239–261 (2014)

[Bow10] Bowen, L.: Measure conjugacy invariants for actions of countable sofic groups. Journal of the American Mathematical Society **23**(1), 217–245 (2010)

[Boy08] Boyle, M.: Open problems in symbolic dynamics. In: Geometric and Probabilistic Structures in Dynamics. Contemp. Math., vol. 469, pp. 69–118. Amer. Math. Soc., Providence (2008)

[CK14] Cyr, V., Kra, B.: The automorphism group of a shift of subquadratic growth. ArXiv e-prints, March 2014

[Cov71] Coven, E.M.: Endomorphisms of substitution minimal sets. Probability Theory and Related Fields **20**(2), 129–133 (1971)

[CY14] Coven, E., Yassawi, R.: Endomorphisms and automorphisms of minimal symbolic systems with sublinear complexity. ArXiv e-prints, November 2014

[DDMP14] Donoso, S., Durand, F., Maass, A., Petite, S.: On automorphism groups of low complexity minimal subshifts (2014)

[DHS99] Durand, F., Host, B., Skau, C.: Substitutional dynamical systems, Bratteli diagrams and dimension groups. Ergodic Theory Dynam. Systems **19**(4), 953–993 (1999)

[DRS12] Durand, B., Romashchenko, A., Shen, A.: Fixed-point tile sets and their applications. J. Comput. System Sci. **78**(3), 731–764 (2012)

[Dur00] Durand, F.: Linearly recurrent subshifts have a finite number of non-periodic subshift factors. Ergodic Theory and Dynamical Systems **20**, 1061–1078 (2000)

[FF96] Fiebig, D., Fiebig, U.-R.: The automorphism group of a coded system. Transactions of the American Mathematical Society **348**(8), 3173–3191 (1996)

[GKM12] Gajardo, A., Kari, J., Moreira, A.: On time-symmetry in cellular automata. Journal of Computer and System Sciences **78**(4), 1115–1126 (2012)

[Gri85] Grigorchuk, R.I.: Degrees of growth of finitely generated groups, and the theory of invariant means. Mathematics of the USSR-Izvestiya **25**(2), 259 (1985)

[Gui11] Guillon, P.: Projective subdynamics and universal shifts. In: 17th International Workshop on Cellular Automata and Discrete Complex Systems, Automata 2011, Center for Mathematical Modeling, University of Chile, Santiago, November 21–23, 2011, pp. 123–134 (2011)

[GZ12] Guillon, P., Zinoviadis, C.: Densities and entropies in cellular automata. ArXiv e-prints, April 2012

[Hed69] Hedlund, G.A.: Endomorphisms and automorphisms of the shift dynamical system. Math. Systems Theory **3**, 320–375 (1969)

[HMU06] Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2006)

[Hoc10] Hochman, M.: On the automorphism groups of multidimensional shifts of finite type. Ergodic Theory Dynam. Systems **30**(3), 809–840 (2010)

[HP89] Host, B., Parreau, F.: Homomorphismes entre systèmes dynamiques définis par substitutions. Ergodic Theory and Dynamical Systems **9**, 469–477 (1989)

[Kar90] Kari, J.: Reversibility of 2d cellular automata is undecidable. Physica D: Nonlinear Phenomena **45**(1–3), 379–385 (1990)

[Kar12] Kari, J.: Decidability and undecidability in cellular automata. Int. J. General Systems **41**(6), 539–554 (2012)

[Kit98] Kitchens, B.P.: Symbolic dynamics - One-sided, two-sided and countable state Markov shifts. Universitext. Springer-Verlag, Berlin (1998)

[KO08] Kari, J., Ollinger, N.: Periodicity and immortality in reversible computing. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 419–430. Springer, Heidelberg (2008)

[KR90] Kim, K.H., Roush, F.W.: On the automorphism groups of subshifts. Pure Mathematics and Applications **1**(4), 203–230 (1990)

[KR91] Kim, K.H., Roush, F.W.: Solution of two conjectures in symbolic dynamics. Proceedings of the American Mathematical Society **112**(4), 1163–1168 (1991)

[Kůr03] Kůrka, P.: Topological and symbolic dynamics. Cours Spécialisés [Specialized Courses], vol. 11. Société Mathématique de France, Paris (2003)
[LM95] Lind, D., Marcus, B.: An introduction to symbolic dynamics and coding. Cambridge University Press, Cambridge (1995)
[Lot02] Lothaire, M.: Algebraic combinatorics on words. Encyclopedia of Mathematics and its Applications, vol. 90. Cambridge University Press, Cambridge (2002)
[Ols83] Olshanskii, A.Y.: On a geometric method in the combinatorial group theory. In: Proceedings of the International Congress of Mathematicians, pp. 415–424 (1983)
[Rot95] Rotman, J.J.: An introduction to the theory of groups, vol. 148. Springer Science & Business Media (1995)
[Rya72] Patrick Ryan, J.: The shift and commutativity. Mathematical systems theory **6**(1–2), 82–85 (1972)
[Sal14a] Salo, V.: Toeplitz subshift whose automorphism group is not finitely generated. ArXiv e-prints, November 2014
[Sal14b] Salo, V.: Subshifts with Simple Cellular Automata. Ph.D. thesis (2014)
[ST12] Salo, V., Törmä, I.: Computational aspects of cellular automata on countable sofic shifts. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 777–788. Springer, Heidelberg (2012)
[ST13] Salo, V., Törmä, I.: Constructions with countable subshifts of finite type. Fundam. Inf. **126**(2–3), 263–300 (2013)
[ST14] Salo, V., Törmä, I.: Block maps between primitive uniform and pisot substitutions. Ergodic Theory and Dynamical Systems **FirstView**, 1–19 (2014)

A Physically Universal Quantum Cellular Automaton

Luke Schaeffer(✉)

Massachusetts Institute of Technology, Cambridge, Massachusetts, England
lrschaeffer@gmail.com

Abstract. We explore a quantum version of Janzing's "physical universality", a notion of computational universality for cellular automata which requires computations to be done directly on the cells. We discuss physical universality in general, the issues specific to the quantum setting, and give an example of a quantum cellular automaton achieving a quantum definition of physical universality.

1 Introduction

Many cellular automata are known to be *computationally universal*, in the sense that they can simulate Turing machines, and hence any other classical model of computation. Shortly after Bernstein and Vazirani introduced quantum Turing machines, Watrous [8] gave a definition for quantum cellular automata (QCA), and showed that QCA can simulate arbitrary quantum Turing machines. Raussendorf [4], van Dam [7], and others give QCA that achieve stricter notions of universality for quantum circuits.

Recently, Janzing [2] defined *physical universality* for cellular automata. A cellular automaton is *physically universal* if for any finite set of cells X, and any transformation f on the region X, there is some way to initialize the complement of X such that, whatever initial configuration x is in the region X, after some number of t timesteps, X contains $f(x)$. In other words, it is possible to perform any transformation on a region by initializing the surrounding cells and waiting for some prespecified time. We will discuss physical universality in general, the issues specific to the quantum setting, and finally give an example of a quantum cellular automaton which is physically universal in the quantum sense.

2 Cellular Automata

We will consider only layered cellular automata in the classical setting.

Definition 1. *A* layered cellular automaton *(LCA) is a 4-tuple* (L, V, Σ, ρ) *consisting of*

- *a finite dimensional lattice* $L = \mathbb{Z}^d$,
- *a list of shifts* $V = [v_1, \ldots, v_k]$ *for* k *different layers, where* $v_i \in \mathbb{Z}^d$,

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 46–58, 2015.
DOI: 10.1007/978-3-662-47221-7_4

– *a finite set of states, Σ, and*
– *a rule $\rho\colon \Sigma^k \to \Sigma^k$ for updating a cell.*

The points of in the lattice are called *cells*. Each *cell* in the lattice has k *layers*, and each layer of a cell has some state in Σ. Each time step, we apply the cell update rule ρ to every cell in the lattice, then shift each layer by the corresponding shift vector v_i. That is, the ith component of the state of a cell $x \in \mathbb{Z}^d$ is moved to the cell $x + v_i \in \mathbb{Z}^d$.

A *region* of cells is any finite subset of the lattice. A *configuration* of a region $X \subseteq L$ is mapping from cells in X to states in Σ^k, and we let $\mathcal{C}(X)$ denote the set of all configurations of X. Naturally, we can combine configurations $x \in \mathcal{C}(X)$ and $y \in \mathcal{C}(Y)$ of disjoint regions X and Y into a configuration $x \times y \in \mathcal{C}(X \cup Y)$ of $X \cup Y$.

A cellular automaton is *reversible* if every configuration of the automaton has a unique predecessor (according to the update rule of the automaton). One advantage of layered cellular automata is that it is trivial to check reversibility – a LCA is reversible if and only if the cell update rule ρ is bijective.

3 Quantum Cellular Automata

A reader unfamiliar with quantum computation will find quantum cellular automata analogous to probabilistic cellular automata (PCA). Probabilistic cellular automata (PCA) generalize (deterministic) cellular automata by letting the configuration of the lattice be a probability distribution over deterministic configurations, and letting the update rule map each classical state to a distribution of states.

If we think of quantum mechanics as a theory of probability with complex numbers [9], then quantum cellular automata (QCA) are very much like probabilistic cellular automata, except the probability distribution is replaced with a *quantum superposition* of classical configurations. That is, a quantum configuration is a map $\psi\colon \mathcal{C}(L) \to \mathbb{C}$ from classical configurations to complex number *amplitudes* such that

$$\ell_2(\psi) = \sum_{x \in L} |\psi(x)|^2 = 1.$$

The analogy between PCA and QCA does not end there. Given a distribution of deterministic configurations, each cell has a *marginal distribution*, but the marginal distributions for all cells do not give us a complete picture of a configuration because the states of cells may be *correlated*. Similarly, each cell of QCA has a quantum state, but cells may be *entangled*.

Let us define quantum cellular automata more formally.

Definition 2. *A* layered quantum cellular automaton *(LQCA) is a 5-tuple (L, V, Σ, q, ρ) consisting of*

– *a finite dimensional lattice $L = \mathbb{Z}^d$,*
– *a list of shifts $V = [v_1, \ldots, v_k]$ for k different layers, where $v_i \in \mathbb{Z}^d$,*

- *a finite set of states,* Σ,
- *a special* quiescent state $q \in \Sigma^k$, *and*
- *a unitary transformation* ρ *on the Hilbert space*

$$\mathcal{H} = \{f : \Sigma^k \to \mathbb{C}\}$$

of functions from Σ^k *to* $\mathbb{C}$.

In addition, we require that ρ *fixes the quiescent state. That is, if* $\psi \in H$ *is the map that sends* q *to* 1 *and all other inputs to* 0, *then* $\rho(\psi) = \psi$.

Intuitively, the state of a cell is a vector x in $\mathcal{H}$ such that $\|x\|_2 = 1$. The state of a region, X, containing k cells is a vector x in the tensor product $\mathcal{H}^{\otimes k}$ such that $\|x\|_2 = 1$, which makes sense because $H^{\otimes k}$ is isomorphic to the Hilbert space of functions from $\mathcal{C}(X)$ to $\mathbb{C}$. We would like a quantum configuration for the entire lattice to be a vector of

$$\bigotimes_{x \in L} \mathcal{H},$$

but this infinite tensor product is not well-defined. Instead, we say a *finite configuration* is a classical configuration $c \in \mathcal{C}(L)$ such that all but finitely many cells in the quiescent state, q. Let $\mathcal{C}^*(L)$ be the set of all finite (classical) configurations the lattice L. Then the *quantum configurations* of the LQCA are defined to be

$$\mathcal{Q}(L) := \{\psi : \mathcal{C}^*(L) \to \mathbb{C} \mid \ell_2(\psi(x)) = 1\}.$$

That is, functions from finite configurations to amplitudes such that the ℓ_2-norm is 1.

The evolution of the LQCA is defined by ρ, a unitary transformation on the quantum state of a cell, and V, the list of shifts. As before, we apply ρ to each cell, then shift each layer by the corresponding vector. Each step is *linear* in the sense that if $x, y \in \mathcal{Q}(L)$ are quantum configurations which evolve to $x', y' \in \mathcal{Q}(L)$ when we apply ρ to every cell, then $\alpha x + \beta y$ evolves to $\alpha x' + \beta y'$ (when we apply ρ) for all $\alpha, \beta \in \mathbb{C}$ such that $\ell_2(\alpha x + \beta y) = |\alpha|^2 + |\beta|^2 = 1$. Therefore it suffices to define the two steps for quantum configurations where for classical configurations (actually quantum configurations where one classical configuration has amplitude 1).

Recall that a finite configuration has only finitely many cells which are not in state q. Since ρ fixes state q, we can ignore all those cells (they remain in state q), and consider a finite quantum system composed of the remaining cells. The set of quantum states for the finite set of cells is in $H^{\otimes k}$ for some k, and we apply ρ to each cell in this finite dimensional space, i.e., apply $\rho \otimes \rho \otimes \cdots \otimes \rho$. We have already seen how to shift the layers of a classical configuration; it is the same in the quantum setting as it was in the classical setting.

4 Physical Universality

Computational universality is well studied in cellular automata. There are cellular automata which can simulate a wide variety of (formal) computational devices: circuits, Turing machines, quantum circuits, other cellular automata, etc. Almost all of these cellular automata require the "data" to be written in a special form, usually distinct from the "program".

- Conway's Life encodes information as gliders, but the program must be laid out as glider guns.
- Margolus' billiard ball machine uses balls to represent data, and the configuration of the "table" determines the computation.
- Raussendorf's universal quantum CA [4] puts quantum bits in even columns (moving left), and the description of quantum gates in odd columns (moving right). Computation occurs as the interleaving columns pass each other.
- Wim van Dam's CA operates directly on qubits, but the program cells are over a larger state space.

Janzing [2] defined physical universality as a stronger notion of universality for cellular automata. Informally, a cellular automaton is *physically universal* if one can implement any transformation on any finite set of cells by "programming" the other cells. To state it more formally, we first need the following definition.

Definition 3. *Let M be a CA on a lattice L. Let X be a region of the lattice. We say a configuration $y \in \mathcal{C}(L\backslash X)$* implements a transformation $f\colon \mathcal{C}(X) \to \mathcal{C}(X)$ in t time steps *if for every configuration $x \in \mathcal{C}(X)$, there exists a configuration $y' \in \mathcal{C}(L\backslash X)$ such that $x \times y$ evolves to $f(x) \times y'$ in t timesteps.*

Then physical universality (in the classical setting) is defined as follows.

Definition 4. *Let M be a CA on a lattice L. Then M is* physically universal *if for every finite set of cells $X \subseteq L$ and for every function $f\colon \mathcal{C}(X) \to \mathcal{C}(X)$, there exists a configuration y of $L\backslash X$ and a time $t \in \mathbb{Z}$ such that y implements the transformation f on X in t timesteps.*

There were no examples of physically universal CAs in Janzing's original paper. We now know that (classical) physically universal CAs exist, with relatively simple examples due to Schaeffer [6], Salo and Törmä [5]. These examples are all layered cellular automata, and the construction used to show physical universality has the same general structure in each case:

1. First, show that any finite configuration eventually becomes inactive.
2. Allow the input configuration to become inactive, and collect whatever information remains.
3. Use the reversibility and computational universality of the automaton to forensically reconstruct the original configuration of the input region.
4. Use computational universality again to apply the given transformation on the input.

5. Find a way to put the desired output configuration in the output region, usually by appealing to reversibility and computational universality yet again.

We will follow exactly the same approach to show that a layered quantum cellular automaton is physically universal, but first let us define quantum physical universality.

5 Quantum Physical Universality

Before we introduce quantum physical universality in cellular automata, let us discuss the inherent limitations of programmable quantum devices in the context of quantum circuits. Nielsen and Chuang [3] call such circuits *programmable quantum gate arrays* (PQGA).

Definition 5. *Let G be a quantum circuit with a* program register $\mathcal{P}$*, and a* data register $\mathcal{D}$*, each consisting of many qubits. Then G is a* programmable quantum gate array *if there exist program states $\{P_i\}_{i\in I}$, and unitary transformations $\{U_i\}_{i\in I}$ of the data register such that for all i and d,*

$$G(P_i \otimes d) = P_i'(d) \otimes U_i(d)$$

where $P_i'(d)$, the garbage left in the program register, may depend on i and d.

In other words, there are $|I|$ settings of the program register, which effect unitary transformations $\{U_i\}_{i\in I}$ on a data register. Nielsen and Chuang make a number of observations about PQGAs in [3], which we summarize in the following theorem.

Theorem 1. *Let G be a PQGA, with P_i, P_i' and U_i as above. Then*

1. *the garbage in the program register, $P_i'(d) = P_i'$, does not depend on d, and*
2. *if U_i and U_j are not the same (up to multiplication by $e^{i\theta}$) then P_i and P_j are orthogonal.*

This has several interesting consequences for quantum physical universality.

1. Since distinct programs have orthogonal program states, the number of unitary operations is bounded by $|\mathcal{C}^*(\mathcal{P})|$, the number of classical configurations of the program register. It is natural to use classical program states (i.e., configurations in $\mathcal{C}^*(\mathcal{P})$), because there is apparently nothing to gain by making them quantum superpositions.
2. The program register *after* the computation cannot depend on the input in the data register. This is purely a side-effect of unitary evolution. Compare this to the notion of reversible physical universality in the classical setting [6], where the final value of the program register needs to be *defined* to be independent of the data.

3. If the program register is finite then the PQGA has finitely many distinct programs. In our cellular automaton, the program register is technically infinite, but since only finitely many bits can interact with the data in any given time, there are still only finitely many distinct programs. Hence, we must abandon the idea of implementing all unitary transformations, and confine ourselves to *approximations* of arbitary unitaries. Fortunately, this problem is shared by quantum circuits, so there are procedures [1] for approximating unitary transformations with a finite set of quantum gates.

This informs our definition of quantum physical universality.

Definition 6. *Let M be a QCA on a lattice L, and let $X \subseteq L$ be a region. We say a configuration $y \in \mathcal{Q}(L\backslash X)$* implements a transformation $U\colon \mathcal{Q}(X) \to \mathcal{Q}(X)$ in t timesteps *if for every configuration $x \in \mathcal{Q}(X)$, there exists a configuration $y' \in \mathcal{Q}(L\backslash X)$ such that $x \otimes y$ evolves to $U(x) \otimes y'$ in t timesteps.*

Similarly, we say a configuration y of Y ϵ-approximately implements a transformation U in t timesteps *if y implements some transformation U' such that $\|U - U'\|_{tr} \leq \epsilon$, where*

$$\|\cdot\|_{tr} = \mathrm{trace}(\sqrt{A^*A})$$

is the trace norm.

Definition 7. *Let M be a QCA on a lattice L. Then M is* physically universal *if for every finite set of cells $X \subseteq L$, every unitary transformation $U\colon \mathcal{Q}(X) \to \mathcal{Q}(X)$ and every $\epsilon > 0$, there exists a configuration $y \in \mathcal{Q}(L\backslash X)$ and a time $t \in \mathbb{Z}$ such that y ϵ-approximately implements the transformation U on X in time t.*

We will see an alternative definition later, once we have an example of a physically universal quantum cellular automaton.

6 A Physically Universal LQCA

Our physically universal LQCA is on the lattice $L = \mathbb{Z}$, and has six layers of qubits (i.e., classical state 0 or 1) with speeds $-3, -2, -1, 1, 2, 3$. Like the reversibly physically universal CA of Salo and Törmä, we program a universal set of gates into the update rule (see below) for automaton. Specifically, we use the controlled-NOT CNOT, the $\pi/8$ gate T and the Hadamard gate H, described briefly below.

CNOT**:** A two-bit classical gate common in reversible computation. If the control bit is 1 then the other bit is negated, otherwise neither bit changes.

T**:** The $pi/8$ gate is a single qubit gate which changes the *phase* if the input is 1, but does nothing.

H**:** The Hadamard gate is a single qubit represented by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

In other words, $H(0)$ is the superposition where 0 and 1 have weight $\frac{1}{\sqrt{2}}$, and $H(1)$ has the sign of 1 reversed.

These three gates are known to be universal, and there exist algorithms for approximating arbitrary unitary transformations [1] with this gate set .

We define ρ by defining it for classical inputs and extending linearly to quantum superpositions thereof. Given a classical cell $(x_{-3}, x_{-2}, x_{-1}, x_1, x_2, x_3)$, we define $\rho(x_{-3}, x_{-2}, x_{-1}, x_1, x_2, x_3)$ by the following list of rules. Use the first rule that applies.

- If $x_{-3} = x_{-2} = x_{-1} = 1$ then cyclically permute $x_1 \to x_2 \to x_3 \to x_1$. Likewise, if $x_1 = x_2 = x_3 = 1$ then cyclically permute $x_{-1} \to x_{-3} \to x_{-2} \to x_{-1}$.
- If $x_2 + x_3 = 1 = x_{-2} + x_{-3}$ then either swap x_1 and x_{-1}, or perform a controlled-NOT on x_1 and x_{-1} as follows.
  - If $x_2 = x_{-2} = 1$ or $x_3 = x_{-3} = 1$ then swap x_1 and x_{-1}.
  - If $x_2 = x_{-3} = 1$ then apply CNOT to x_1 and x_{-1} with x_1 as the control bit.
  - If $x_{-2} = x_3 = 1$ then apply CNOT to x_1 and x_{-1} with x_{-1} as the control bit.
- If $x_2 = x_3 = x_{-3} = 1$ and $x_{-2} = 0$ then apply H to x_{-1}. Similarly, if $x_3 = x_{-2} = x_{-3} = 1$ and $x_2 = 0$ then apply H to x_1.
- If $x_2 = x_3 = x_{-2} = 1$ and $x_{-3} = 0$ then apply T to x_{-1}. Similarly, if $x_2 = x_{-2} = x_{-3} = 1$ and $x_3 = 0$ then apply T to x_1.
- Otherwise, leave the cell unchanged.

Observe that the CA is almost entirely classical. If a cell is in a classical state initially then, in most cases, ρ maps it to another classical state. The only exceptions are a handful of cases where change the phase (for T) or introduce a superposition (for H).

Our goal is to show that this LQCA is physically universal. Assume we are given a finite region X, and a unitary transformation U on the cells in X. By the Solovay-Kitaev theorem [1], for any $\epsilon > 0$ there exists a circuit C (of H, T and CNOT gates) which implements a unitary within ϵ of U. The problem is then to implement the circuit C.

We follow the pattern used in Schaeffer [6], and Salo and Törmä [5], so we start by showing that information contained in a bounded region will escape in a recoverable format. To this end, we define the notion of a depleted configuration.

Definition 8. *A quantum configuration $x \in \mathcal{Q}(\mathbb{Z})$ is* depleted *if, for every classical configuration with nonzero amplitude in x, there is at most one particle per cell and no particle of speed v_i occurs to the right of a particle of speed $v_j > v_i$, for all i and j.*

We show that any finite configuration quickly becomes depleted.

Theorem 2. *Let $X \subseteq \mathbb{Z}$ be the region $X = \{1, \ldots, n\}$ and let Y be the complement. Suppose $0_Y \in \mathcal{Q}(Y)$ is the configuration with all cells in the 0 state. Then there is a time $t = \frac{5}{2}n + O(1)$ such that for any $x \in \mathcal{Q}(X)$, the configuration $x \otimes 0_Y$ evolves to a depleted configuration within time t.*

Proof. The update rule does nothing to a cell if either $x_{-3} = x_{-2} = x_{-1} = 0$ or $x_3 = x_2 = x_1 = 0$. In particular, if there are no right-moving particles in cells $(-\infty, j]$ at some time, then there are no right-moving particles in the range $(-\infty, j+1]$ on the following step, because all the right-moving particles have moved right by at least one cell. Similarly for left-moving particles in $[j, \infty)$.

There are initially no right-moving particles (or particles of any kind) $(-\infty, 0]$, and no left-moving particles in $[n+1, \infty)$. By the observation above, there will be no right-moving particles in $(-\infty, \lceil \frac{n+1}{2} \rceil)$ in $\lceil \frac{n+1}{2} \rceil$ steps, nor left-moving particles in $[\lfloor \frac{n+1}{2} \rfloor, \infty)$. It follows that every cell contains only left-moving or right-moving particles (or neither), and therefore particles move at constant speed without interacting.

In $\lceil \frac{n+1}{2} \rceil$ time steps, a particle could move as far as $\frac{3}{2}n+1$ cells from its initial position. The right-moving particles, for instance, could be spread over a range of $2n + O(1)$ cells between $\frac{n}{2}$ and $\frac{5}{2}n + 2$. So it will take at most $2n + O(1)$ steps for the left-most speed 2 particles to overtake the right-most speed 1 particles, or for the left-most speed 3 particles to overtake the right-most speed 2 particles. Hence, the right-moving particles will be ordered by speed in at most $\frac{5}{2}n + O(1)$ time steps. Similarly for the left-moving particles, so the configuration becomes depleted in $t = \frac{5}{2}n + O(1)$ time steps. □

Suppose we start with a finite configuration. Over time, it becomes depleted according to the theorem above. At that point, whatever computation the configuration may have performed is over, since no interactions can occur in a depleted configuration. Furthermore, the remains of the computation are readily available in six groups of particles. Remember that each of these particles is present or absent in each classical configuration, representing a bit, but since we are in a quantum configuration, each particle is a qubit, and the qubits may be entangled. We will collect these quantum particles, gather them together, perform a quantum computation, and then place the result back into X.

Let us discuss how to manipulate these quantum particles. The way we manipulate these particles is by placing purely classical particles, which we call *manipulators* to distinguish them from the particles that come out of X, in the complement of X. The manipulators will interact with the quantum particles in such a way that quantum operations (H, T and CNOT) are performed on the particles, yet the manipulators remain purely classical particles, and do not change speed or direction.

By inspection of the cell update rule, we need at least three particles in a cell for an interaction to occur. Two manipulators are required to perform a swap or apply CNOT, and three are required for T and H. In the cases where only two manipulators are required, the operation does nothing unless there is a third particle present (usually in the speed 1 or speed -1 layer). We rely on the fact that two particles or manipulators have at most one point of intersection. This ensures that any pair of manipulators can only affect one particle, at a time and place predetermined by the initial locations of the manipulators.

We need to show how to do three things with manipulators:

1. Take the six groups of particles in a depleted configuration and redirect them such that they all move in the same direction at speed 1, in a format suitable for computation.
2. Approximate an arbitrary quantum computation. We do this by implementing an arbitrary quantum circuit (with gates H, T, and CNOT) *exactly*.
3. Convert a single group of speed 1 particles back into six groups of different speed, aimed towards the output region X.

The first step is a poor introduction to the manipulation of particles, so we begin with the second step – computation – and prove Theorem 3. We will then return to the problem of redirecting particles and prove Theorem 4. Finally we argue that the third step is the reverse of the first step, and therefore follows from the Theorem 4.

Theorem 3. *Suppose X is a region of size $2n$, containing particles $x_1, \ldots, x_n$ of speed 1 in the even cells. Let C be a circuit on n inputs, composed of* CNOT*, H, and T gates. Then there exists a time t (polynomial in n and the size of C), such that we can implement the transformation defined by C on $x_1, \ldots, x_n$ in time t, leaving the result in the even cells of a region Y of size $2n$.*

Proof. Let us call the area containing the particles the *workspace*. The workspace is initially X, but moves right as the particles move right, and may grow as we move particles around.

We need to show how to implement four operations: we must be able to apply T, H and CNOT to quantum particles, and be able to move or swap particles around. It suffices to be able to move a particle left relative to its peers, since this allows us to completely reorder the particles if we need to. Given these four operations, it is clear we can implement an arbitrary circuit C.

T and H: The easiest operation is T. We arrange three manipulators (of speed 2, -2, -3) to intercept the desired particle x_i at some time. The update rule causes an T gate to be applied to the speed 1 layer, containing x_i. Similarly, three manipulators (of speed 3, -2, -3) will apply a Hadamard gate H to a speed 1 particle. Since none of the manipulators have the same speed as the particles, they spend at most $O(n)$ time steps in the workspace. After that, the workspace is clear for the next operation.

Move left: If we meet a particle with two manipulators of speed 3 and -3, it swaps the speed 1 and speed -1 particles. The speed -1 layer is kept empty, so this effectively reverses the direction of the particle. After the particle has travelled in the opposite direction for some time, we may reverse it again, as long as there is no speed 1 particle already in this cell. This allows us to move particles to the left. There are, however, a few limitations:

- A particle with speed -1 moves 2 cells per timestep relative to the particles of speed 1. Hence, the distance between the initial and final position is a multiple of two. This is why the particles are assumed to be in even cells.

– We must be careful not to let the manipulators from the two swaps meet. If they do, they would perform another swap, potentially sending one of the x_i's off in the wrong direction. The manipulators will meet if and only if the time between the two reversals is a multiple of three, which we can easily avoid. If we need to move a particle by a multiple of three cells, we simply split the move into two parts.

As before, we wait until the manipulators have cleared the workspace before performing another operation.

CNOT: For CNOT, the idea is to reverse the direction of one input, and then at the moment it meets the other input, have two manipulators (speed -2 and 3) induce a CNOT operation (where the speed -1 particle controls the speed 1 particle). We have already seen how to move particles, and how to apply gates, so the only new problem is how to avoid interference between the manipulators of the two swaps and the manipulators which implement the CNOT.

If a speed -2 manipulator meets a speed 3 manipulator, it will implement a CNOT on the speed 1 and speed -1 layers of that cell. However, the CNOT does nothing unless there is a speed -1 particle, and the only speed -1 particle is the one we intend to use in the CNOT operation. Hence, we can ignore speed -2 manipulators.

The speed 3 manipulators, as we have already discussed, will intersect if the operations they implement are separated in time by a multiple of three. Fortunately, we only have three operations here: two particle reversals and a CNOT, so we can schedule these operations such that they do not interfere. In particular, this means that the initial position (relative to the other particles) of the control particle (of the CNOT), the final position of the control particle, and the position of the target particle must be distinct modulo 3. We may be required to move some of the particles around to accommodate this condition, but we have already seen how to do that.

We separate all operations by $O(n)$ time steps to ensure that manipulators from one operation leave the workspace entirely before the next operation, to avoid collisions within the workspace. Manipulators will inevitably meet outside the workspace, but there is no interaction unless there are at least three. It is difficult, if not impossible, to avoid having two manipulators intersect, but whenever *three* manipulators intersect, we can always postpone the last of the three operations (corresponding to the manipulators) to avoid the collision. □

Next we consider the problem of capturing the remains of X, once it has reached a depleted configuration.

Theorem 4. *Let $X \subseteq \mathbb{Z}$ be the region $X = \{1, \ldots, n\}$ and let Y be the complement. There is some configuration $y \in \mathcal{Q}(Y)$ and some time t such that if we let $x \otimes y$ evolve for t time steps, only the speed 1 layer, in even cells, depends on x. In other words, the information from X is contained in the speed 1 layer of even cells.*

Proof. First, Theorem 2 tells us the particles in X will separate out into six groups by speed, with the fastest left-moving particles on the far left and the fastest right-moving particles on the far right, in time $O(n)$. We will show how to change the speed of each group to 1.

Consider the group of speed 3 particles. The way we change the speed (but not direction) of a particle is by intercepting it with three manipulators (one of each speed) moving in the opposite direction. Applied to a speed 3 particle, these manipulators will reduce its to speed 1. We manipulate particles from back to front, so that the speed 1 particles fall behind the speed 3 particles, instead of being overtaken. We also leave plenty of time between manipulations to ensure that two manipulators ever intercept a particle, and three manipulators never intersect, except at the planned times and locations of manipulations.

For speed 2 particles, we manipulate each particle to have speed 3, reducing the problem to one we have already seen. This time we order the manipulations from front to back so that the new speed 3 particles do not overtake the old speed 2 particles. Similarly, we can convert speed -3 or speed -2 particles to speed -1.

The final step is to reverse the speed -1 particles. We saw how to do this in Theorem 3: two manipulators (speed 3 and -3) collide to swap the speed -1 layer with the speed 1 layer. As before, we can avoid unintended collisions between manipulators if we perform the manipulations in the right order, and with sufficient time between them.

Now if some particles of speed 1 lie in odd cells, then use further manipulations to increase their speed back to 2, breaking parity, and allowing us to maneuver the particle to an even cell (more accurately, an even cell in even time steps, an odd cell in odd time steps). Then we increase its speed to 3, and back to 1 again, but in an even cell. □

To finish the proof of physical univesrality, we need to show how to output the computed configuration. This means taking a collection of particles of speed 1, and dividing them into six groups. Then we accelerate each group to a different speed, and aim them at the region X. We assume that the particles output by the computational phase (i.e., the collection of particles of speed 1) were computed to account for the interactions that occur when the particles come together in region X, so they produce the desired output, namely U (or a close approximation) applied to the initial contents of X.

Observe that our LQCA is reversible, so we can run it backwards. Furthermore, it is close to being the same automaton in reverse – the case which allows us to change speed 1 to speed 2 to speed 3 cycles in the opposite direction, gates are inverted (of course, H and CNOT are their own inverses), and particles move backwards, but the automaton is close enough that Theorem 2 and Theorem 4 go through. Hence, we can construct a configuration (in the reverse QCA) which takes the contents of the region X, reduces their speed, and collects them together. The manipulators do not depend on the contents of X, so we can compute their positions at the end (again, in the reverse QCA) of this construction. Now let us run these manipulators forward in the (forward) LQCA.

The manipulators take a group of speed 1 particles and force them into an output region, which is exactly what we want!

In summary, we claim that the LQCA defined earlier is physically universal. Given a region X and a unitary transformation U on X, we construct a circuit (of T, H and CNOT gates) to implement some U' such that $\|U - U'\|_{tr} \leq \epsilon$. Then we implement U' in the LQCA in three steps:

- We extract the data initially in X by letting it reach a depleted configuration, and rounding up the particles that escape.
- We decode the initial configuration from the particles, apply the circuit for U', and encode the desired configuration as a collection of particles.
- We aim the particles at region X, and wait for them to interact in X and produce the transformed output.

With sufficient time between these three phases, we can avoid collisions between the manipulators. This concludes the proof of our main result.

Theorem 5. *The LQCA described at the beginning of the section is physically universal.*

In fact, the LQCA achieves a stronger definition of physical universality where the program configuration is not allowed to depend on ϵ. In other words, a single configuration implements arbitrarily good approximations of U if we let it run longer.

Definition 9. *Let M be a QCA on a lattice L. Then M is* strongly physically universal *if for every finite set of cells $X \subseteq L$, and every unitary transformation $U \colon \mathcal{Q}(X) \to \mathcal{Q}(X)$ there exists $y \in \mathcal{Q}(L \backslash X)$ such that for any $\epsilon > 0$ there exists a time $t \in \mathbb{Z}$ such that y ϵ-approximately implements the transformation U on X in time t.*

Corollary 1. *The LQCA described at the beginning of the section is strongly physically universal.*

Proof. Suppose the input region is X_0 and the unitary is U_0. Let $(\epsilon_i)_{i=0}^{\infty}$ be a sequence of positive real numbers tending to zero. By physical universality, there is a configuration y_0 in $\mathcal{Q}(L \backslash X_0)$ and time t_0 such that y ϵ_0-approximately implements the transformation U_0 on X_0 in time t_0.

Now iteratively build programs on larger and larger regions. In general, let X_{i+1} be a region containing X_i plus $3t_i$ cells on either side, and all non-quiescent cells in y_i. This region is large enough that no particle outside it can possibly affect X_0 in t_0 steps. Apply physical universality to X_{i+1} with error $\epsilon_{i+1} > 0$ and unitary U_{i+1}, where U_{i+1} applies U_i to the region X_i, and the identity transformation to the cells in $X_{i+1} \backslash X_i$. Physical universality gives us a program $y_{i+1} \in \mathcal{Q}(L \backslash X_{i+1})$ and time t_{i+1}.

Finally, combine the y_i configurations into a single large configuration $y \in \mathcal{Q}(L \backslash X_0)$. Given an $\epsilon > 0$, find some $\epsilon_i < \epsilon$ and let y run for t_i time steps. This is just enough time for the program y_i to execute, but not enough time for the later programs to interfere, so we get an $\epsilon_i < \epsilon$ approximation of U applied to X_0. □

7 Future Work

- We leave the time and space complexity of this cellular automaton open for analysis. Can we quantify the performance of the construction in the proof of strong physical universality?
- The automaton is not as simple or aesthetically pleasing as its classical counterparts. Can we construct a less obviously artificial LQCA in more dimensions?
- Is there a notion of physical universality for unbounded computations on a quantum Turing machine?

References

1. Dawson, C.M., Nielsen, M.A.: The Solovay-Kitaev algorithm. Quantum Info. Comput. **6**(1), 81–95 (2006)
2. Janzing, D.: Is there a physically universal cellular automaton or Hamiltonian? (2010). http://arxiv.org/abs/1009.1720
3. Nielsen, M.A., Chuang, I.L.: Programmable quantum gate arrays. Phys. Rev. Lett. **79**, 321 (1997)
4. Raussendorf, R.: Quantum cellular automaton for universal quantum computation. Phys. Rev. A **72**, 022301 (2005)
5. Salo, V., Törmä, I.: A one-dimensional physically universal cellular automaton. Personal Communication (2014)
6. Schaeffer, L.: A physically universal cellular automaton. In: Roughgarden, T. (ed.) Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, pp. 237–246. ACM, Rehovot (2015)
7. van Dam, W.: A universal quantum cellular automaton. In: Proceedings of PhysComp96, pp. 323–331. InterJournal (1996)
8. Watrous, J.: On one-dimensional quantum cellular automata. In: 36th Annual Symposium on Foundations of Computer Science, pp. 528–537. Society Press (1995)
9. Youssef, S.: Quantum Mechanics as Bayesian Complex Probability Theory. Modern Physics Letters A **9**, 2571–2586 (1994)

Effect of Graph Structure on the Limit Sets of Threshold Dynamical Systems

Abhijin Adiga[1], Chris J. Kuhlman[1], Henning S. Mortveit[1,2(✉)], and Sichao Wu[1]

[1] Network Dynamics and Simulation Science Laboratory, VBI, Virginia Tech, Blacksburg, USA
{abhijin,ckuhlman,hmortvei,sichao}@vbi.vt.edu
[2] Department of Mathematics, Virginia Tech, Blacksburg, USA

Abstract. We study the attractor structure of standard block-sequential threshold dynamical systems. In a block-sequential update, the vertex set of the graph is partitioned into blocks, and the blocks are updated sequentially while the vertices within each block are updated in parallel. There are several notable previous results concerning the two extreme cases of block-sequential update: (i) sequential and (ii) parallel. While parallel threshold systems can have limit cycles of length at most two, sequential systems can have only fixed points. However, Goles and Montealegre [5] showed the existence of block-sequential threshold systems that have arbitrarily long limit cycles. Motivated by this result, we study how the underlying graph structure influences the limit cycle structure of block-sequential systems. We derive a sufficient condition on the graph structure so that the system has only fixed points as limit cycles. We also identify several well-known graph families that satisfy this condition.

Keywords: Graph dynamical systems · Generalized cellular automata · Threshold functions · Block decomposition

1 Introduction

In this paper, we study Boolean graph dynamical systems or automata networks induced by threshold functions. Such systems are a natural choice to model various biological and sociological phenomena (see [6–8,11] for example). We consider *standard Boolean threshold functions* where, each vertex v is associated with a *threshold* T_v, and the vertex function of v evaluates to 1 if and only if at least T_v vertices in its closed neighborhood (v and its distance-1 neighbors) are in state 1. These systems have been extensively studied [1,3,4]. Several generalizations of standard threshold functions have been considered in the past. For example in [10], *bi-threshold systems* were studied where the thresholds for the 0 to 1 and 1 to 0 transitions can be different. Multi-threshold systems with more than 2 possible vertex states were considered in [4,9]. In [14], systems with *dynamic thresholds* were considered where the vertex thresholds vary with time. Another popular variant of the standard threshold function are Hopfield

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 59–70, 2015.
DOI: 10.1007/978-3-662-47221-7_5

networks, where only the open neighborhood is considered while evaluating the next state, i.e., the state of the vertex itself is ignored.

Our focus is on the *limit set* or the *attractor* structure of these systems which captures their "long-term" behavior. The *update scheme*, i.e., the order in which the vertex functions are evaluated, influences the attractor structure and in general the phase space. Two update schemes (i) *synchronous* or parallel and (ii) *sequential* are well-studied. In the synchronous update scheme, every vertex function is applied simultaneously, while in a sequential update scheme, vertices are updated one by one according to a total order defined on the vertex set. A generalization of these schemes is the *block-sequential update*. Here, the vertices are partitioned into blocks, and vertices within the blocks are updated synchronously while the blocks themselves are updated sequentially. A more general sequential scheme is the *word update* which is a generalization of the sequential systems. Here, a vertex can be updated more than once in a single time step [12].

Two interesting questions which have been repeatedly addressed in the past are: given a graph dynamical system, (i) what is the maximum possible length of a limit cycle? (ii) what conditions lead to only fixed points as limit sets? There are some notable results in the case of standard threshold systems. Goles and Olivos [3] and Barrett et al. [1] independently, using different methods, showed that sequential threshold systems exhibit only fixed points as limit cycles. In [3,4], it was shown that for synchronous update there can be limit cycles of length at most two. Their result is applicable for the more general case of weighted threshold functions. Kuhlman et al. [10] considered these questions regarding bi-threshold systems. They showed that, while synchronous systems can have limit cycles of length at most two, sequential systems can have arbitrarily long limit cycles.

In this paper, we consider standard threshold systems with block-sequential update. Mortveit [13] showed that if the blocks are of size at most 3, then there will be only fixed points. The author also conjectured that the limit cycle length can be at most two for arbitrary block size. However, this was disproved recently by Goles and Montealegre [5]. Unlike the sequential or synchronous cases, these systems can have arbitrarily long limit cycles. In [4], the more general setting of weighted threshold functions was studied. They gave a sufficient condition for the system to have only fixed points. In this work, we examine standard threshold systems systems from a structural perspective. Our main objective was to identify conditions on the underlying graph structure which lead to only fixed points. Our main result is given below.

Theorem 1. *Let X be a simple graph with vertex set $V[X]$ and edge set $E[X]$. Let $\mathcal{B}$ be a block partition of $V[X]$. If every block $B \in \mathcal{B}$ satisfies Condition (1) below, then, any block-sequential standard threshold system induced by $\mathcal{B}$ for any update order on the blocks has only fixed points as limit sets. Also, the transient length is at most $(|E[X]| + |V[X]| + 1)/2$.*

$$\text{For any non-empty } B' \subseteq B \text{ and any assignment } y \text{ of vertex states for } B',\ \|B'\| - 2|\Lambda_{B'}(y)| - |B'| < 0, \text{ where, } \|B'\| \text{ is the number of edges in the subgraph induced by } B' \text{ and } \Lambda_{B'}(y) = \big\{\{u,v\} \in E[X] \mid u, v \in B',\ \text{and } y_u = y_v\big\}. \tag{1}$$

An interesting feature of Theorem 1 is that Condition (1) only applies to the individual blocks and is independent of the connections between the blocks. The proof uses the potential function argument introduced in [1]. We build on the framework provided by [13] and extend the results of that paper. We also show that simple graph classes such as trees and complete graphs satisfy Condition (1). In addition, we show that any graph which can be *block-decomposed* into subgraphs which satisfy Condition (1), also satisfies this condition.

We note that Condition (1) is not a necessary condition. Consider any graph with arbitrary block partition where each vertex has threshold 1. This is a progressive threshold system, i.e., a vertex will never transition from 1 to 0. Hence, it has only fixed points even though the blocks may not satisfy Condition (1).

The organization of the paper is as follows. We introduce the notation and basic definitions in the next section. In Section 3, we prove Theorem 1. In Section 4, we derive the block-decomposition result. In Section 5, we demonstrate some graph classes which satisfy Condition (1) before we conclude in Section 6.

2 Preliminaries

Let X be a simple undirected graph on n vertices with vertex set $V[X]$ and edge set $E[X]$. Let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a block partition of $V[X]$. For $S \subseteq V[X]$, let $\deg_S(v)$ denote the number of neighbors of v in the graph induced by S, and let $\deg(v)$ be its degree in X. x_v denotes the *vertex state* of v. Since we are considering Boolean systems, $x_v \in \{0,1\}$. Let $x = (x_1, x_2, \ldots, x_n)$ be the system state. Let $n[v]$ denote the sorted sequence of the closed neighborhood of v, and let $x[v]$ denote the restriction of x to $n[v]$.

Every vertex is assigned a *threshold function* $f_v : \{0,1\}^{\deg(v)+1} \longrightarrow \{0,1\}$ defined as follows:

$$f_v\big(x[v]\big) = \begin{cases} 1, & \text{if } \sum_{w \in n[v]} x_w \geq T_v, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $T_v \in \mathbb{N}$ with $T_v \geq 1$ is the *threshold* of v. For a block B and system state x, the map $F_B(x) : \{0,1\}^n \longrightarrow \{0,1\}^n$ is given by

$$\big(F_B(x)\big)_v = \begin{cases} f_v\big(x[v]\big), & \text{if } v \in B, \\ x_v, & \text{otherwise.} \end{cases} \tag{3}$$

The block-sequential map $F : \{0,1\}^n \longrightarrow \{0,1\}^n$ is defined as

$$F = F_{B_m} \circ F_{B_{m-1}} \circ \cdots \circ F_{B_1}\,. \tag{4}$$

The two special cases of the block-sequential update scheme are sequential and parallel update schemes. The sequential update corresponds to each vertex belonging to a distinct block, i.e., for $i = 1, \ldots, n$, $|B_i| = 1$ and therefore, $m = n$. In parallel update, there is only one block, i.e., $m = 1$ and $B_1 = V[X]$.

3 Sufficient Condition for Fixed Points

3.1 Potential Function Method

For $v \in V[X]$, let $T_0(v)$ denote the smallest number of vertices in $n[v]$ that must be in state 0 for x_v to be mapped to zero. By the definition of threshold function in (2), we have $T_0(v) + T(v) = \deg(v) + 2$. With each vertex and edge, we associate a potential. The vertex potential for vertex v and system state x is

$$P(x, v) = \begin{cases} T(v), & x_v = 1, \\ T_0(v), & x_v = 0\,. \end{cases} \tag{5}$$

The edge potential for edge $e = \{v, v'\}$ is

$$P(x, e) = \begin{cases} 1, & x_v \neq x_{v'}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The system potential function $P : \{0, 1\}^n \to \mathbb{N}$ for state x is defined as

$$P(x) = \sum_{v \in V[X]} P(x, v) + \sum_{e \in E[X]} P(x, e)\,. \tag{7}$$

For sequential threshold systems, Barrett et al. [1] showed that for any $x' = F(x)$ where $x \neq x'$, $P(x') < P(x)$. Since $P(x) \geq 0$ for all x, it follows that the limit set is comprised of only fixed points. They also showed that the transient length is at most $(|E[X]| + |V[X]| + 1)/2$ as a consequence of this result. The same argument was applied by Mortveit [13] for block-sequential systems with blocks of size at most three.

3.2 Proof of Theorem 1

Suppose the system transitions from state x to x' when a block B is updated, i.e. $x' = F_B(x)$. Let $\Delta P = P(x') - P(x)$. Let $P_v(x) = P(x, v) + \sum_{e \in E_v[X]} P(x, e)$, where $E_v[X]$ is the set of edges incident with v. Let $\Delta P_v = P_v(x') - P_v(x)$ denote the change in potential at vertex v. Note that since only block B is updated, $\forall v \notin B$, $x_v = x'_v$. Let $B(x, x') \subseteq B$ denote the set of vertices such that $x_v \neq x'_v$. We use the following result from Mortveit [13].

Lemma 1 (Mortveit [13]). $\Delta P = \sum_{v \in B(x,x')} \Delta P_v$.

The lemma below gives an upper bound for ΔP_v.

Lemma 2. *For any $v \in B(x, x')$, let γ_v denote the number of neighbors of v in $B(x, x')$ which have the same state as v in x (and therefore, in x'). Applying F_B, $\Delta P_v \leq \deg_{B(x,x')}(v) - 2\gamma_v - 2$.*

Proof. In [13], ΔP_v is bounded as a function of $\deg_B(v)$ and the number of vertices in state 1 in x. We apply the same approach here, but obtain a more compact result. Let $n_1(x, v)$ and $n_0(x, v)$ denote the number of neighbors of v in state 1 and 0 respectively. There are two possible cases: v transitions either from 0 to 1 or 1 to 0. We consider these cases separately.

Transition 0 → 1. Note that $n_1(x) \geq T(v)$. We recall that $T(v) + T_0(v) = \deg(v) + 2$. Also, since only vertices in $B(x, x')$ change state,

$$\begin{aligned} n_0(x', v) &= n_0(x, v) + (\text{number of neighbors of } v \text{ in } B(x, x') \text{ in state 1 in } x) \\ &\qquad - (\text{number of neighbors of } v \text{ in } B(x, x') \text{ in state 0 in } x) \\ &= n_0(x, v) + (\deg_{B(x,x')}(v) - \gamma_v) - \gamma_v \\ &= \deg(v) - n_1(x, v) + \deg_{B(x,x')}(v) - 2\gamma_v \,. \end{aligned} \tag{8}$$

Now we compute ΔP_v.

$$\begin{aligned} \Delta P_v &= \big(T(v) + n_0(x', v)\big) - \big(T_0(v) + n_1(x, v)\big) \\ &= T(v) + \big(\deg(v) - n_1(x, v) + \deg_{B(x,x')}(v) - 2\gamma_v\big) \\ &\qquad - \big(\deg(v) + 2 - T(v) + n_1(x, v)\big) \\ &= 2\big(T(v) - n_1(x, v)\big) + \big(\deg_{B(x,x')}(v) - 2\gamma_v - 2\big) \\ &\leq \deg_{B(x,x')}(v) - 2\gamma_v - 2 \,. \end{aligned}$$

Transition 1 → 0. In this case, we have $n_1(x) \leq T(v) - 2$. Following a similar approach as in (8), it can be shown that $n_1(x', v) = n_1(x, v) + \deg_{B(x,x')}(v) - 2\gamma_v$.

$$\begin{aligned} \Delta P_v &= \big(T_0(v) + n_1(x', v)\big) - \big(T(v) + n_0(x, v)\big) \\ &= \big(\deg(v) + 2 - T(v)\big) + \big(n_1(x, v) + \deg_{B(x,x')}(v) - 2\gamma_v\big) \\ &\qquad - T(v) - \big(\deg(v) - n_1(x, v)\big) \\ &= 2\big(n_1(x, v) - T(v)\big) + 2 + \big(\deg_{B(x,x')}(v) - 2\gamma_v\big) \\ &\leq \deg_{B(x,x')}(v) - 2\gamma_v - 2 \,. \end{aligned}$$

□

For a set of vertices $S \subseteq V[X]$, let $\|S\|$ denote the number of edges in $X[S]$, the subgraph induced by S. For a state vector x, let $\Lambda_S(x) = \big\{(u, v) \mid u, v \in S,\ x_u = x_v \text{ and } \{u, v\} \in E[X]\big\}$, i.e., the set of all pairs of adjacent vertices in S which have the same state.

Lemma 3. $\Delta P = P(x') - P(x) \leq 2\big(\|B(x, x')\| - 2|\Lambda_{B(x,x')}(x)| - |B(x, x')|\big)$.

Proof. From Lemma 1, $P(x') - P(x) = \sum_{v \in B(x,x')} \Delta P_v$. Applying Lemma 2, $P(x') - P(x) \leq 2(\|B(x, x')\| - \sum_{v \in B(x,x')} \gamma_v - |B(x, x')|)$. Note that for each $(u, v) \in \Lambda_{B(x,x')}(x)$, v contributes 1 to γ_u and u contributes 1 to γ_v. Moreover, if $(u, v) \notin \Lambda_{B(x,x')}(x)$, then, it does not contribute to the sum. Therefore, $\sum_{v \in B(x,x')} \gamma_v = \sum_{(u,v) \in \Lambda_{B(x,x')}(x)} 2 = 2|\Lambda_{B(x,x')}(x)|$. Hence proved. □

Lemma 4. *Let block B satisfy Condition (1). Then, for any $x' = F_B(x)$ such that $x' \neq x$, $P(x') < P(x)$.*

Proof. From Lemma 3, $\Delta P \leq 2(\|B(x,x')\| - 2|\Lambda_{B(x,x')}(x)| - |B(x,x')|)$. Let $B' = B(x,x')$ and y be the state vector restricted to $B(x,x')$. Since $x' \neq x$, B' is not empty. Therefore, by Condition (1), $\|B'\| - 2|\Lambda_{B'}(y)| - |B'| < 0$. □

From Lemma 4, we note that if every block satisfies Condition (1), then, whenever a block is updated and some vertices change states, the system potential decreases. Since the potential cannot be negative by definition, it follows that there can be only fixed points as limit sets. Hence, we have proved Theorem 1.

4 Block Decomposition

In the graph theory literature, a *block* is a maximal connected subgraph without a cut vertex [2]. Every block can either be a maximal 2-connected subgraph, an edge, or an isolated vertex. Since the term "block" has already been used to mean something else in this paper, we will henceforth refer to maximal connected subgraphs as *subblocks*. Every graph can be decomposed into subblocks. Since they satisfy maximality, any two subblocks overlap in at most one vertex, which, if it exists, is a cut vertex of the graph. This is illustrated with an example in Figure 1(a). Let C be the set of cut vertices and $\mathcal{S}$ be the set of subblocks. The *block graph* is the bipartite graph on the vertex set $C \cup \mathcal{S}$ where for $c \in C$ and $S \in \mathcal{S}$, $\{c, S\}$ is an edge if and only if $c \in S$. It can be easily shown that the block graph is a tree. See Figure 1(b) for the block graph of the example.

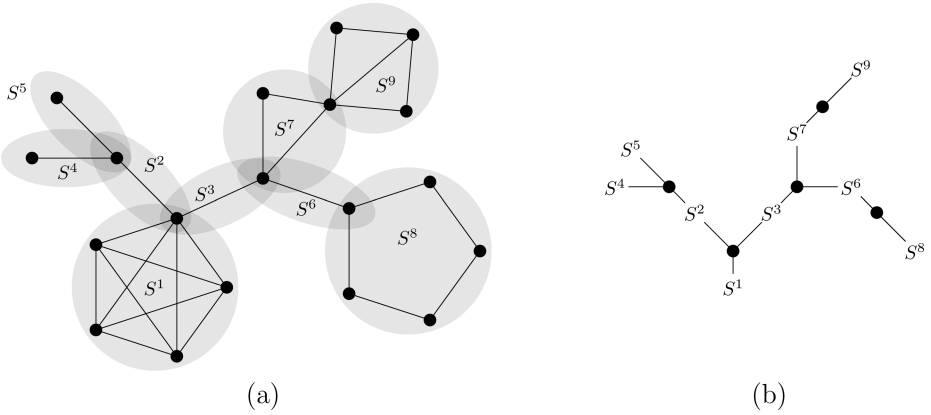


Fig. 1. An example of (a) a block decomposition and (b) the corresponding block graph

Theorem 2. *Let block B be such that all of its subblocks satisfy Condition* (1)*. Then, B satisfies Condition* (1) *too.*

Proof. Let $\{S^1, \ldots, S^k\}$ be the vertex partition of B where each S^i induces a subblock of B. From the block graph representation, it is easy to see that there exists an ordering of the subblocks such that every subblock has at most one cut

vertex in common with its previous subblocks. We will assume that the current ordering satisfies this property, i.e.,

$$\left|\left(\cup_{j<i} S^j\right) \cap S^i\right| = 1,\ \forall i = 1, \ldots, k\,. \tag{9}$$

Let $D \subseteq B$ and $D^i = D \cap S^i$. From (9), we have $|D| \geq |D^1| + \sum_{i=2}^{k} \left(|D^i| - 1\right)$. For a subset of vertices S, let E_S denote the edge set of the graph induced by S. We observe that the edge sets E_{S^i} partition E_B. This implies that E_{D^i} are mutually disjoint. Since $\Lambda_{D^i}(x) \subseteq E_{D_i}$, they are mutually disjoint, too. We have

$$\begin{aligned}\|D\| - 2\Lambda_D(x) - |D| &\leq \sum_{i=1}^{k} \left(\|D^i\| - 2\Lambda_{D^i}(x)\right) - \left(|D^1| + \sum_{i=2}^{k} \left(|D^i| - 1\right)\right) \\ &= \left(\|D^1\| - 2\Lambda_{D^1}(x) - |D^1|\right) + \sum_{i=2}^{k} \left(\|D^i\| - 2\Lambda_{D^i}(x) - |D^i| + 1\right).\end{aligned}$$

Since all the subblocks satisfy the Condition (1), the first term in the above expression is negative while the second term is at most 0. Hence, $\|D\| - 2\Lambda_D(x) - |D| < 0$. □

5 Simple Graph Classes which Satisfy Condition (1)

We will show that some graph classes such as trees, odd cycles, and complete graphs satisfy Condition (1). Even though these are very simple graphs, to the best of our knowledge, these results have not been obtained before using any other method. Throughout this section, B corresponds to a block in X and $B' \subseteq B$.

Proposition 1. *If B induces a tree in X, then it satisfies Condition (1).*

Proof. Suppose $B' \subseteq B$. If the graph induced by B' is connected, then it still corresponds to a tree. If not, then, each connected component (which is also a tree) in the graph can be considered independent of the rest of the block. In that case, effectively we are working with a smaller tree. Hence, without loss of generality, we will assume that B' is connected (and can be the same as B). Since $\|B'\| = |B'| - 1$, it implies that B satisfies Condition (1). □

Alternatively, we could have used Theorem 2 to prove the above proposition.

Proposition 2. *If B induces an odd cycle in X, then it satisfies Condition (1).*

Proof. If $B' \subset B$, then it corresponds to a collection of disconnected paths. Then, we can apply Proposition 1 to show that B' satisfies the condition. Therefore, we will assume that $B' = B$. We first note that $\|B\| = |B|$. Since the cycle is odd, there exists by the pigeonhole principle, at least one pair of vertices in $\Lambda_{B'}(y)$ for any state vector y. Hence, for all $B' \subseteq B$, $\|B\| - 2\Lambda_B(y) - |B| < 0$. □

Proposition 3. *If B induces a clique in X, then it satisfies Condition* (1).

Proof. If $B' \subset B$, then it still induces a clique. Hence, we will assume that $B' = B$. Let y be the state vector restricted to B. Let n_0 denote the number of vertices of B in state 0 and let $n = |B|$. Since B is a clique, $|\Lambda_B(x)| = \binom{n_0}{2} + \binom{n-n_0}{2}$, which attains a minimum value of $\frac{1}{2}\left[\lfloor \frac{n}{2} \rfloor \left(\lfloor \frac{n}{2} \rfloor - 1\right) + \lceil \frac{n}{2} \rceil \left(\lceil \frac{n}{2} \rceil - 1\right)\right]$ at $n_0 = \lfloor \frac{n}{2} \rfloor$. Therefore,

$$\begin{aligned} \|B\| - 2|\Lambda_B(x)| - |B| &= \binom{n}{2} - 2|\Lambda_B(x)| - n \\ &< \left\lfloor \frac{n}{2} \right\rfloor - n = -\left\lceil \frac{n}{2} \right\rceil . \end{aligned}$$

□

The next result concerns systems with block size at most 4. This is an extension of the result by Mortveit [13].

Proposition 4. *Any block B of size* 4*, other than the* 4*-cycle, satisfies Condition* (1).

Proof. If $B' \subset B$, then it corresponds to a block of size 3 or less, for which the result follows from [13]. Hence, we will assume that $B' = B$. If B is a tree or clique, then, by Propositions 1 and 3, the statement is true. The remaining possibilities excluding the 4-cycle are isomorphic to one of the graphs illustrated in Figure 2. We consider them one by one and in each case show that $\|B\| - 2|\Lambda_B(x)| - |B| < 0$ and the rest follows from Lemma 3.

Graph (a) In this case, $|B| = \|B\| = 4$ and since $\{2, 3, 4\}$ induces an odd cycle, it implies that $\Lambda_{B(x,x')}(x)$ is not empty and therefore $|\Lambda_{B(x,x')}(x)| \geq 1$.

Graph (b) Here, $\|B\| = 5$ and again, since $\{2, 3, 4\}$ (or $\{1, 3, 4\}$) induces an odd cycle, $|\Lambda_{B(x,x')}(x)| \geq 1$.

Graph (c) The argument is similar to the previous case. □

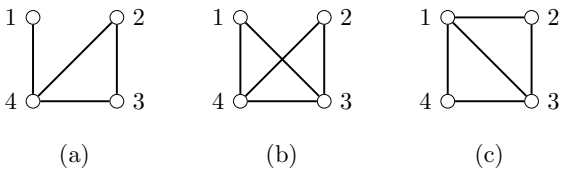


Fig. 2. Possible connected graphs (up to isomorphism) of size 4 excluding trees and 4-cycle

Remark 1. Note that one can configure an example corresponding to a 4-cycle (C_4) which does not satisfy Condition (1) (see Figure 3(a)). Moreover, this configuration corresponds to a limit cycle of length 2. So, a natural question to ask is whether graphs which do not have a C_4 as a vertex-induced subgraph satisfy Condition (1) for all x. Unfortunately, the answer is no. Figure 3(b) is an example where an induced-C_4-free graph has a limit cycle of length two.

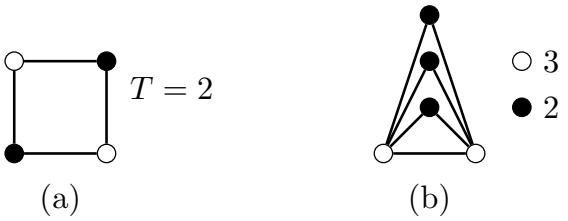


Fig. 3. Configurations which lead to a limit cycles of length two: if the black vertices are in state 0, then the white are in 1, and vice versa. (a) Block C_4 where all vertices have threshold 2 and (b) an induced-C_4-free graph with the black vertices having threshold 2 and white vertices 3.

Proposition 5. *If B is a wheel graph with odd cycle, then it satisfies Condition (1).*

Proof. A wheel graph is formed by connecting a single vertex to all vertices of a cycle. See Figure 4 (a) as an illustration. Let $B' \subseteq B$ and y be the state vector restricted to B'. There are three cases that need to be considered.

(a) B' does not contain the center vertex. In this case, B' induces either an odd cycle or a collection of paths. Then, from Propositions 1 and 2, $\|B'\| - 2|\Lambda_{B'}(y) - |B'| < 0$.

(b) $B' \subset B$ and contains the center vertex. In this case, the central vertex corresponds to a cut vertex (see Figure 4 (b)). Therefore, block decomposition of B' yields subblocks, all of which have the following structure: a path graph where each vertex is connected to a central vertex. Let B'' be such a subblock (illustrated in Figure 4 (b)). We only need to show that $\|B''\| - 2\Lambda_{B''}(y) - |B''| < 0$. The rest follows from Theorem 2. Let $n = |B''|$. We have $\|B''\| = 2n - 3$.

Without loss of generality, we will assume that the state of the center vertex is 0. Let Q denote the remaining set of vertices and of these, let k be in state 0. It is clear that the rest $n - 1 - k$ vertices are in state 1. We have

$$|\Lambda_{B''}(y)| = k + |\Lambda_Q(y)|\,. \tag{10}$$

Now we claim that

$$|\Lambda_{B''}(y)| \geq \frac{n-2}{2}\,. \tag{11}$$

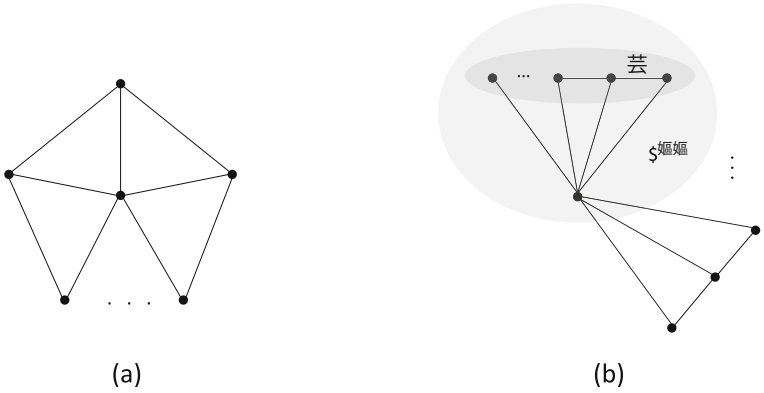


Fig. 4. (a) An exemplary wheel graph. (b) Induced sub-graph B' of a wheel graph by removing multiple vertices from the cycle.

If $k \geq \frac{n-2}{2}$, (11) trivially holds. Thus, we will assume that $k < \frac{n-2}{2}$. Note that each edge on Q contributes 1 to $|\Lambda_Q(y)|$ if and only if the two vertices associated with the edge are in the same state. Therefore, $|\Lambda_Q(y)|$ will achieve its minimum value when all vertices in state 0 have their neighbors in state 1. And since $k < \frac{n-2}{2}$, it is guaranteed that such a configuration exists. In this case, $n-2-2k$ edges will contribute to $|\Lambda_Q(y)|$, i.e. we have, $|\Lambda_Q(y)| \geq n-2-2k$. This yields, $|\Lambda_{B''}(y)| = k + |\Lambda_Q(y)| \geq n-2-k > \frac{n-2}{2}$. Hence, (11) holds, which in turn implies that

$$\|B''\| - 2|\Lambda_{B''}(y)| - |B''| \leq 2n - 3 - 2 \cdot \frac{n-2}{2} - n \leq -1\,.$$

Hence, proved.

(c) $B' = B$, i.e. we consider the whole wheel graph. In this case, $|B| = n$ and $\|B\| = 2n-2$. Now, we will show that $\|B\| - 2|\Lambda_B(y)| - |B| < 0$. The argument is similar to the previous case. Let Q denote the set of vertices in the cycle. Now, we will claim that

$$|\Lambda_B(y)| \geq \frac{n-1}{2}\,. \tag{12}$$

Since $|\Lambda_B(y)| = k + |\Lambda_Q(y)|$, if $k \geq \frac{n-1}{2}$, the above inequality holds. We can thus assume $k < \frac{n-1}{2}$. There are $n-1$ edges on the cycle. Using the same arguments as in the previous case, at most $2k$ edges do not contribute to the value of $|\Lambda_Q(y)|$, which means $|\Lambda_Q(y)| \geq n-1-2k$. It follows that

$$|\Lambda_B(y)| = k + |\Lambda_Q(y)| \geq n - 1 - k > \frac{n-1}{2}\,.$$

Hence, (12) holds. We have,

$$\|B\| - 2|\Lambda_B(y)| - |B| \leq 2n - 2 - 2 \cdot \frac{n-1}{2} - n \leq -1\,.$$

Hence, proved. □

Remark 2. Note that there exists a wheel graph with an even cycle corresponding to a limit cycle of length 2, see Figure 5. In this example, suppose the threshold value of the central vertex is 3 and all other vertices have threshold value 2. Then, one can verify that the central vertex will remain 0 and the other vertices will change states alternatively in pairs, i.e. this configuration leads to a length 2 limit cycle.

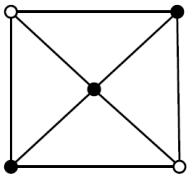


Fig. 5. Configuration over a wheel graph with an even cycle which leads to a limit cycle of length 2. Black vertices are in state 0 and white are in state 1. The threshold value for the central vertex is 3, and 2 for other vertices.

6 Conclusion

In this paper, we studied the limit cycle structure of standard threshold dynamical systems with block-sequential update. We identified a sufficient condition for the system to have only fixed points as limit sets. There are several possibilities to consider for the future. Even though the condition depends only on the blocks and not the graph as a whole, it seems to be restrictive. One direction to explore is to find more general conditions which take into account edges between the blocks too. Another direction would be to study bi-threshold block-sequential systems.

References

1. Barrett, C.L., Hunt, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: Complexity of reachability problems for finite discrete dynamical systems. Journal of Computer and System Sciences **72**(8), 1317–1345 (2006)
2. Diestel, R.: Graph Theory. Springer (2 edn.) (2000)
3. Goles, E., Olivos, J.: Comportement périodique des fonctions à seuil binaires et applications. Discrete Applied Mathematics **3**(2), 93–105 (1981)
4. Goles, E., Martínez, S.: Neural and automata networks. Kluwer (1990)
5. Goles, E., Montealegre, P.: Computational complexity of threshold automata networks under different updating schemes. Theoretical Computer Science **559**, 3–19 (2014). http://www.sciencedirect.com/science/article/pii/S0304397514006756. non-uniform Cellular Automata
6. Granovetter, M.: Threshold models of collective behavior. American journal of sociology, 1420–1443 (1978)

7. Karaoz, U., Murali, T., Letovsky, S., Zheng, Y., Ding, C., Cantor, C.R., Kasif, S.: Whole-genome annotation by using evidence integration in functional-linkage networks. Proceedings of the National Academy of Sciences of the United States of America **101**(9), 2888–2893 (2004)
8. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of theoretical biology **22**(3), 437–467 (1969)
9. Kuhlman, C.J., Mortveit, H.S.: Limit sets of generalized, multi-threshold networks. Journal of Cellular Automata (2015)
10. Kuhlman, C.J., Mortveit, H.S., Murrugarra, D., Kumar, V.S.A.: Bifurcations in boolean networks. In: Automata 2011, pp. 29–46 (2011)
11. Macy, M.W.: Chains of cooperation: Threshold effects in collective action. American Sociological Review, 730–747 (1991)
12. Mortveit, H., Reidys, C.: An introduction to sequential dynamical systems. Springer Science & Business Media (2007)
13. Mortveit, H.S.: Limit cycle structure for block-sequential threshold systems. In: Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2012. LNCS, vol. 7495, pp. 672–678. Springer, Heidelberg (2012)
14. Wu, S., Adiga, A., Mortveit, H.S.: Limit cycle structure for dynamic bi-threshold systems. Theoretical Computer Science (2014). http://www.sciencedirect.com/science/article/pii/S0304397514005088

A Cellular Automaton for Blocking Queen Games

Matthew Cook[1], Urban Larsson[2], and Turlough Neary[1(✉)]

[1] Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland
tneary@ini.phys.ethz.ch

[2] Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada
urban031@gmail.com

Abstract. We show that the winning positions of a certain type of two-player game form interesting patterns which often defy analysis, yet can be computed by a cellular automaton. The game, known as *Blocking Wythoff Nim*, consists of moving a queen as in chess, but always towards (0,0), and it may not be moved to any of $k-1$ temporarily "blocked" positions specified on the previous turn by the other player. The game ends when a player wins by blocking all possible moves of the other player. The value of k is a parameter that defines the game, and the pattern of winning positions can be very sensitive to k. As k becomes large, parts of the pattern of winning positions converge to recurring chaotic patterns that are independent of k. The patterns for large k display an unprecedented amount of self-organization at many scales, and here we attempt to describe the self-organized structure that appears.

1 Blocking Queen Games (k-Blocking Wythoff Nim)

In the paper [Lar11], the game of k-Blocking Wythoff Nim was introduced, with rules as follows.

Formulation 1: As in Wythoff Nim [Wyt07], two players alternate in removing counters from two heaps: any number may be removed from just one of the heaps, or the same number may be removed from both heaps. However, a player is allowed to reject the opponent's move (so the opponent must go back and choose a different, non-rejected move), up to $k-1$ times, where k is a parameter that is fixed for the game. The k^{th} distinct attempted move must be allowed. Thus, if there are at least k winning moves among the options from a given position, then one of these winning moves can be played.

Formulation 2: There are k chess pieces on an infinite (single quadrant) chess board: one queen, and $k-1$ pawns. On your turn you move the queen towards the origin. (The first player who cannot do this loses.) The queen cannot be

Urban Larsson is supported by the Killam Trust.
Turlough Neary is supported by Swiss National Science Foundation grants 200021-141029 and 200021-153295.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 71–84, 2015.
DOI: 10.1007/978-3-662-47221-7_6

moved to a position with a pawn, but it can move over pawns to an empty position. After moving the queen, you complete your turn by moving the $k-1$ pawns to wherever you like. The pawns serve to block up to $k-1$ of the queen's possible next moves.

Example Game: Consider a game with $k = 5$, where the queen is now at $(3,3)$ (yellow in Figure 1). It is player A's turn, and player B is blocking the four positions $\{(0,0),(1,1),(0,3),(3,0)\}$ (dark brown and light olive). This leaves A with the options $\{(3,1),(3,2),(2,2),(2,3),(1,3)\}$ (each is black or blue). Regardless of which of these A chooses, B will then have at least five winning moves to choose from (ones marked yellow, or light, medium, or dark olive). These are winning moves because it is possible when moving there to block all possible moves of the other player and thereby immediately win. Therefore player B will win.

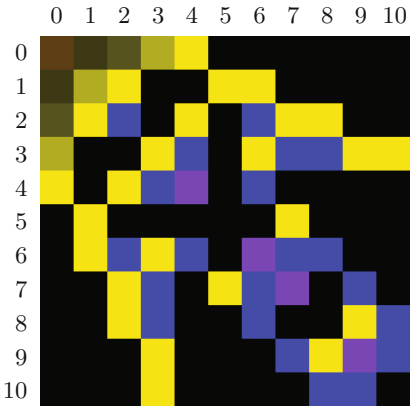


Fig. 1. Game 5: $(0,0)$ is the upper left position on this 11×11 chessboard, and the queen is allowed to move north, west, or north-west. The colors represent the number of winning moves available (ignoring blocking) from each position. 0: Dark Brown, 1: Dark Olive, 2: Olive, 3: Light Olive, 4: Yellow, 5: Black, 6: Blue, 7: Indigo. Since $k = 5$, up to four moves can be blocked. This means that by moving to a position with four or fewer winning moves available, it is possible to block all those winning moves, thus preventing the other player from making a winning move. Therefore the positions with four or fewer winning moves available (yellow, brown, and the olives) are themselves winning moves. The remaining positions, with five or more winning moves available (black, blue, and indigo), are losing moves, because if you move there, it is not possible to prevent the other player from making a winning move. Thus the color at any given position (the *palace number*, see text) can be computed by considering the colors above and to the left of it.

As shown in Figure 1, there is a simple algorithm to compute the winning positions for game k. These are known as *P-positions* in combinatorial game theory, and we will refer to them as *palace positions*, the idea being that the queen wants to move to a palace: if you move her to a palace, you can win,

while if you move her to a non-palace, your opponent can win. To win, you must always block (with pawns) all of the palaces your opponent might move to.

The idea is simply that a palace is built on any site that can see fewer than k other palaces when looking due north, west, or north-west. In this way, the pattern of palaces can be constructed, starting at (0,0) and going outward. For efficiency, a dynamic programming approach can be used, storing three numbers at each position, for the number of palaces visible in each of the three directions. With this technique, each line can be computed just from the information in the previous line, allowing significant savings in memory usage.

The case $k = 1$ corresponds to classical Wythoff Nim, solved in [Wyt07]. In [Lar11], the game was solved for $k = 2$ and 3. When we say a game is solved we mean it is possible to give closed-form expressions for the P-positions, or at least that a winning move, if it exists, can be found in log-polynomial time in the heap sizes. For example, the set $\{(\lfloor n\phi \rfloor, \lfloor n\phi^2 \rfloor), (\lfloor n\phi^2 \rfloor, \lfloor n\phi \rfloor)\}$, where n runs over the nonnegative integers and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio, provides a solution for classical Wythoff Nim. Combinatorial games with a blocking maneuver appeared in [GaSt04], [HoRe01], [HoRe] and [SmSt02], and specifically for Wythoff Nim in [Gur10], [HeLa06], [Lar09] and [Lar15].

The new idea that we use in this paper is to look directly at the number of palaces that the queen can see from each position on the game board, and to focus on this *palace number* rather than just on the palace positions (P-positions). (The palace positions are exactly the locations with palace numbers less than k.) In previous explorations, the palace number was in fact computed but then only used for finding the new palace positions, which when viewed on their own give the appearance of involving long-range information transfer. By observing the palace number directly, however, we can see that in almost all positions the palace number is surprisingly close to k, and if we look at its deviation from k then we are led to discover that these deviations follow a local rule that does not even depend on k. The next section will present this local rule as a cellular automaton (CA). A finite automaton for Wythoff Nim has been studied in a different context in [Lan02]. Other recent results for cellular automata and combinatorial games can be found in [Fin12], [Lar13] and [LaWa13].

The rest of the paper will present the rich structure visible in the patterns of palace numbers for games with large k. A surprising number regions of self organization appear, largely independent of the particular value of k, and some of the self-organized patterns are quite complex, involving multiple layers of self-organization. This is the first CA we are aware of that exhibits so many levels of self-organization. So far, these patterns offers many more questions than answers, so for now we will simply try to catalog our initial observations.

2 A Cellular Automaton Perspective

In this section we describe a cellular automaton that computes the palace numbers for blocking queen games.

2.1 Definition of the CA

The CA we present is one-dimensional and operates on a diamond space-time grid as shown in Figure 2, so that at each time step, each cell that is present at that time step derives from two parents at the previous time step, one being half a unit to the right, the other being half a unit to the left. On the diamond grid, there is no cell from the previous step that is 'the same' as a given cell on the current step. However, the three grandparents of a cell, from two steps ago, do include a cell which is at the same spatial position as the grandchild cell.

Our CA rule is based not only on the states of the two parent cells, but also on the state of the central grandparent cell, as well as *its* parents and central grandparent, all shown in blue in Figure 2. As such, this CA depends on the previous four time steps, *i.e.* it is a fourth-order CA. It is the first naturally occurring fourth-order CA that we are aware of. We say it is "naturally occurring" simply because we discovered these pictures by analyzing the blocking queen game, and only later realized that these pictures can also be computed by the fourth-order diamond-grid CA we present here.[1]

The states in our CA are integers. In general they are close to 0, but the exact bounds depend on the initial conditions. The formula for computing a cell's value, given its neighborhood, is described in Figure 2.

2.2 The Connection Between the CA and the Game

Since our definition of the CA appears to be completely different from the definition of the blocking queen game, we need to explain the correspondence.

The idea is that the states of this CA correspond to palace numbers minus k, which are generally integers close to zero. One can easily prove a bound of $3k+1$ for the number of states needed for the game with blocking number k, since each row, column, and diagonal can have at most k palaces, so palace numbers are always in the range $[0, 3k]$. However, in practice the number of states needed (after the initial ramping-up region near the origin) appears to be far smaller, more like $\log k$. For example, when $k = 500$, only eight states are needed, ranging between -4 and 3.

Surprisingly, this single CA is capable of computing the pattern of palace numbers regardless of the value of k. Different values of k simply require different initial conditions: The initial condition for a given value of k is that every site in the quadrant opposite the game quadrant should be k, and every site in the other two quadrants should be 0.

[1] Using the dynamic programming approach described in Section 1, the information for each cell can be computed from just its parents and central grandparent, so that approach is also clearly a CA, but it needs a large number of states for the cells and each different value of k requires a different rule. The CA we describe in this section is much more surprising, being a single rule that is completely independent of k and using very few states to produce all of the interesting parts of the pictures.

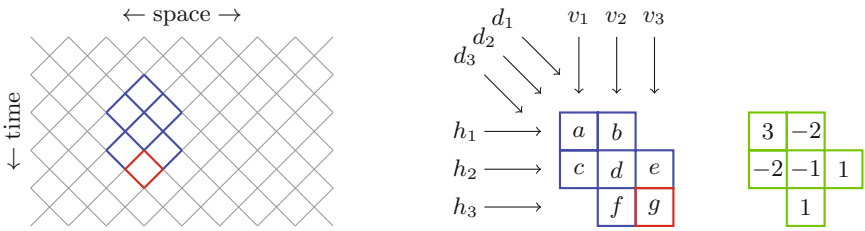


Fig. 2. The CA rule. **(left)** The diamond grid on which the CA operates. The value of the red cell is determined by the values of the blue cells. This neighborhood stretches four steps back in time (i.e., four levels above the red cell), making it a 4^{th} order CA. **(center)** This diagram is rotated 45°, so time flows in the direction of the diagonal arrows. This is the orientation used in all the figures in this paper. The red cell's value is computed according to the formula $g = a - b - c + e + f + p$ **(right)** These green squares correspond to the blue cells and show the *palace compensation terms*. For any blue cell containing a negative value (and therefore a palace, see Section 2.2), the corresponding palace compensation term must be added. In the formula, p represents the total contribution of these palace compensation terms. Note that location d only affects g via its palace compensation term, so only its sign matters.

Theorem 1. *The k-Blocking Wythoff Nim position (x, y) is a P-position if and only if the CA given in Figure 2 gives a negative value at that position, when the CA is started from an initial condition defined by*

$$CA(x,y) = \begin{cases} k & x < 0 \text{ and } y < 0 \\ 0 & x < 0 \text{ and } y \geqslant 0 \\ 0 & x \geqslant 0 \text{ and } y < 0 \end{cases}$$

Proof. irst we will consider the case of no P-positions occurring within the CA neighborhood, so compensation terms can be ignored.

As shown in Figure 2, we will let v_1 be the number of P-positions directly above a and c, and similarly for v_2 and v_3, as well as for the diagonals d_i and horizontal rows h_i.

This gives us $a = v_1 + d_2 + h_1 - k$, and so on: when adding the k-value to each cell this represents the sum of the numbers of P-positions in the three directions.

We would like to express g in terms of the other values. Notice that

$$a + e + f \;=\; \sum_{i=1}^{3} v_i \;+\; \sum_{i=1}^{3} d_i \;+\; \sum_{i=1}^{3} h_i \;- 3k = \; b + c + g$$

and therefore $a + e + f = b + c + g$, allowing us to express g in terms of the other values as $g = a - b - c + e + f$.

All that remains is to take any P-positions in the CA neighborhood into account, so as to understand the compensation terms.

If there is a P-position at a, then b, c, d, and g (i.e. the positions in a line to the right, down, or right-down) will all be one higher than they were before

taking that palace into account. Since the equation $a+e+f = b+c+g$ was true when ignoring the palace at a, it becomes wrong when the palace at a produces its effect of incrementing b, c, d, and g, because that makes the right hand side go up by 3 while the left hand side is untouched. To compensate for this, we can add a term p_a to the left hand side, which is 3 if there is a palace at a, and 0 otherwise.

Similarly, if b is a P-position, then this increments d, e, and f, so to compensate, we will need to subtract 2 from the left hand side of $a + e + f = b + c + g$ if b is a P-position.

We can see that we are computing exactly the compensation terms shown in the green squares of Figure 2. Once we include all the compensation terms, the formula for g becomes correct even in the presence of local P-positions, and it corresponds exactly to the rule given in Figure 2.

The initial condition can be confirmed to produce (via the CA rule) the correct values in the first two rows and columns of the game quadrant, and from that point onwards the reasoning given above shows that the correct palace numbers, and therefore the correct P-positions, are being computed by the CA rule.

2.3 Notes on Reversability

The reversed version of this CA computes a, given b, c, d, e, f, and g. This is done with the equation $a = g - f - e + c + b - p$, which is equivalent to the equation in the caption of Figure 2. However, the palace compensation term p can depend on a, so this equation has not fully isolated a on the left hand side. (The forward direction did not have this problem, since p does not depend on g.) Writing $p = p_a + p_{bcdef}$ to separate the palace compensation term from a from the other palace compensation terms, we get $a + p_a = g - f - e + c + b - p_{bcdef}$. Since p_a is 3 when a is negative, and 0 otherwise, this equation always yields either one or two solutions for a. If the right hand side is 3 or more, then a must be equal to it. If the right hand side is negative, then a must be 3 less than it. And if the right hand side is 0, 1, or 2, then a can *either* be equal to it *or* be 3 less than it—we are free to choose. The reversed rule is non-deterministic, but it can always find a compatible value. In other words, there is no "Garden of Eden" pattern for this rule, if we assume that all integers are permissible states.

3 Self-Organization

The top row of Figure 3 shows the palace number patterns for games 100 and 1000. The patterns are strikingly similar, given that the value of k differs by an order of magnitude. The pattern for game 1000 has the appearance of being "the same, but ten times bigger" than the pattern for game 100. The middle and lower rows of Figure 3 zoom in on subregions where the two patterns are in fact identical, without any scaling factor.

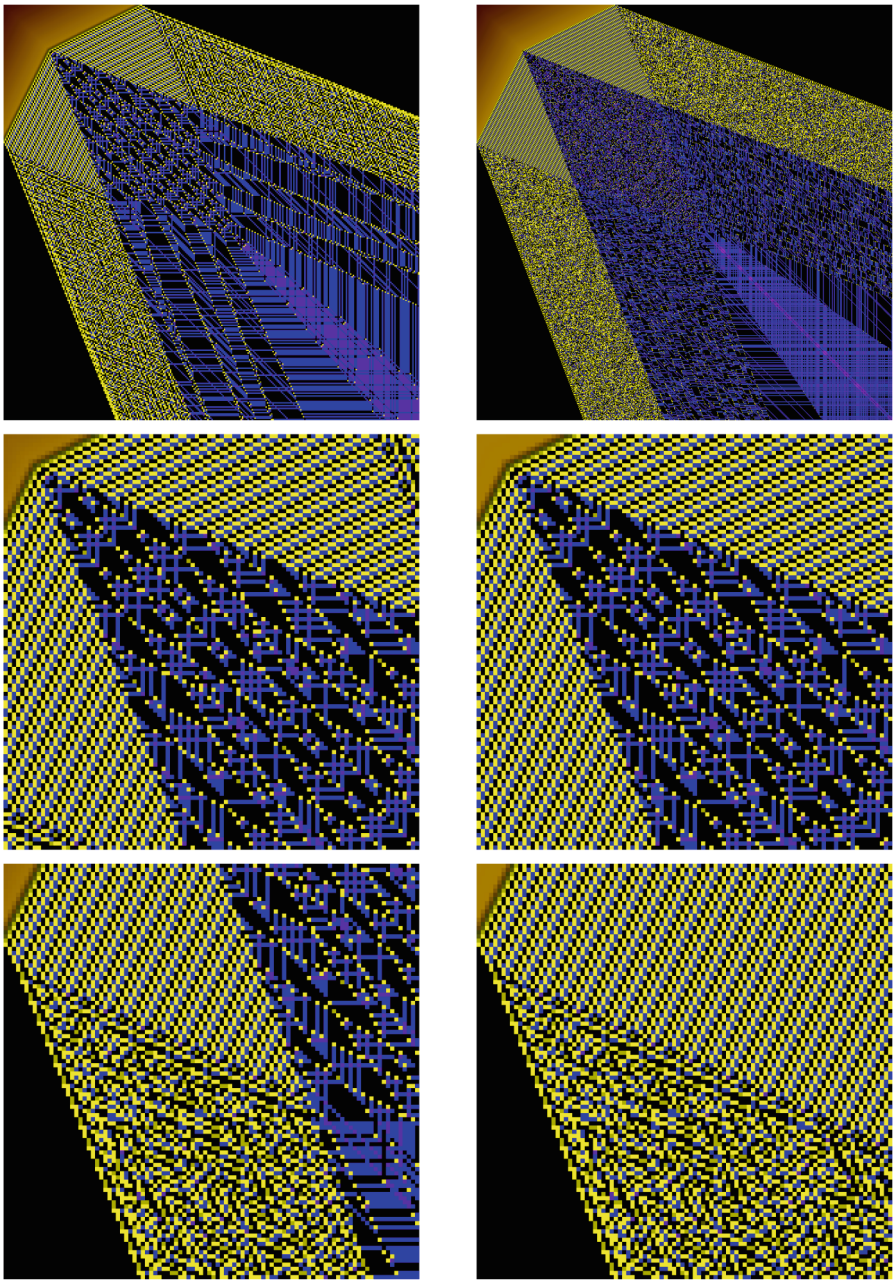


Fig. 3. Self-organized regions in game 100 (left, 300×300 region shown in top row) and game 1000 (right, 3000×3000 region shown in top row). The lower images are close-ups of the upper images, showing regions of identical behavior. The middle row shows a 100×100 region by the nose, and the bottom row shows a 100×100 region by the shoulder.

3.1 Terminology

As an aid to our discussion of these complex images, we will give names to the prominent features in them.

We can see that this system self-organizes itself into 11 regions with 14 borders and 6 junctions. Ignoring duplicates due to the mirror symmetry, there are 7 regions, 7 borders, and 4 junction points.

We will start by naming the 7 regions and some features inside them. The region at the upper left (the game's terminal region, and the CA's starting region), in the shape of a dented triangle, is the *hood*. The triangular regions adjacent to the hood, with a periodic interior (visible in all panels of Figure 3), are the *épaulettes*. The rhomboid region that emanates from between the pair of épaulettes is the *fabric*. The solid black regions at the top and at the left constitute the *outer space*. Between the outer space and the épaulettes we find the *arms* (irregular yellow regions in Figure 3), which extend indefinitely. Extending next to the arms, and of similar width, we have the *warps*. Each warp contains a number of *threads* (strings of yellow dots, clearly visible in the top left panel of Figure 3) which come out of the fabric. Between the warps lies the *inner sector*, and the blue stripes in the warps and in the inner sector are the *weft*.

Next, we will name the 4 junction points. The hood, épaulettes, and fabric all meet at the *nose*. The hood, épaulette, arm, and outer space all meet at the *shoulder*. The warp, fabric, épaulette, and arm all meet at the *armpit*, which is often a hotspot of highly positive palace numbers. And the fabric, warps, and inner sector meet at the *prism*. The inner sector often contains slightly higher palace numbers than the warps, especially near the main diagonal, giving the impression of light being emitted from the prism.

Finally, we come to the 7 borders. The hood and épaulette meet cleanly at the *casing*. The hood contains all the states from $-k$ to 0, but after the casing, the CA uses very few states. The épaulette and arm meet at the *seam*. The épaulette and fabric meet at the *rift*, a narrow, relatively empty space which appears to get wider very slowly. The fabric and warp meet at the *fray*, where threads almost parallel to the fray unravel from the fabric, and threads in the other direction exit the fabric and start merging to form thick threads in the warp. There is no clear boundary where the warp meets the inner sector, since the warp simply runs out of threads. The warp also meets the arm cleanly, at the inside of the arm. At the boundary between the warp and the arm, it appears that the yellow nature of the arm is due to being packed full of threads, and the warp simply has a much lower density of threads. Threads that bend into the inner sector, and stop being parallel to the rest of the warp, are often called *beams*. The often-occurring slightly-separated periodic part of the arm, bordering the outer space, is the *skin*.

The fray, warp, central sector, armpits, and prism are all very sensitive to k, but all the other regions are not, with the exception of the fabric and the rift, which are sensitive only to $k \bmod 3$. The fabric, fray, warp, prism, and central sector are all full of weft. Often the centermost beams will communicate with

each other via the weft, and this process can usually be analyzed to calculate the slopes of these beams, which are generally quadratic irrationals.

This is the greatest complexity of self-organization that we have seen for a system that has no structured input.

3.2 Structure Within the Regions

The hood and the épaulets have a very regular structure. The palace numbers in the hood increase steadily in each row and column, increasing by one when the higher coordinate is incremented, and by two when the lower coordinate is incremented. Thus the hood contains all palace numbers from 0, at the upper left corner, to k, where the hood meets the épaulettes at the casing, a line with slope -1/2 (and -2) that connects the nose at $(k/3, k/3)$ to the shoulder at $(0, k)$ (and $(k, 0)$). The hood is exactly the region where every move is a winning move, because all possible further moves can be blocked, thereby immediately winning the game. When players are not making mistakes, the game ends when (and only when) somebody moves into the hood, thereby winning the game.

The palace numbers in the épaulettes form a two-dimensional periodic pattern, with periods $(5, 1)$ and $(4, 3)$ (and all integral linear combinations of those base periods). Only the palace numbers $k - 1$, k, and $k + 1$ appear in the

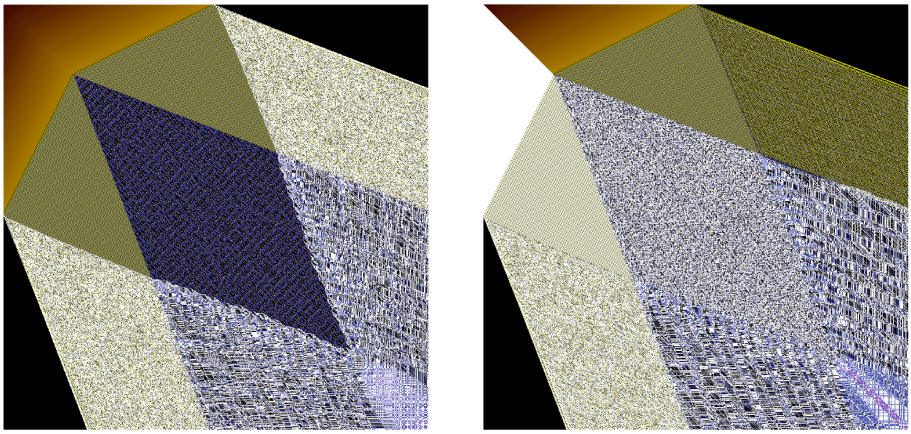


Fig. 4. **(left)** A comparison between $k = 497$ and $k = 500$. Positions where the two images differ are masked in white. All other colors show places where the two images match, meaning that if the palace number at position (x, y) is p in the image for $k = 497$, then the palace number at position $(x + 1, y + 1)$ is $p + 3$ in the image for $k = 500$. P-positions are shown in yellow (and brown, in the hood), and N-positions are shown in black and blue. Note that the hood, épaulettes, and fabric match perfectly. **(right)** A comparison between $k = 499$ and $k = 500$. In this comparison, "matching" means that if the palace number at position (x, y) is p in the image for $k = 499$, then the palace number at position $(x + 1, y)$ is $p + 1$ in the image for $k = 500$. Note that the épaulette and arm match perfectly, as does half of the hood.

épaulettes. In the épaulette's periodic region of size 11, $k-1$ (a P-position) appears 5 times, and k and $k+1$ (both N-positions (*no palace*)) each appear 3 times. The rift has a periodicity of (81,31), which is also a periodicity of the épaulette.

The arms, shown in the bottom row of Figure 3 have a random appearance, although they often contain temporary black stripes at one of the two angles parallel to their sides. Despite this initial appearance of disorder, the arms have many interesting properties, discussed in Section 3.4.

The fabric exhibits further self-organization. The larger black regions visible in the middle row of Figure 3 form a rough grid, and in much larger pictures ($k \gg 1000$) the grid morphs into a larger-scale grid, which is at a slightly different angle and has cells about 3.5 times larger. Regions of the small-grid pattern appear to travel through the large grid like meta-gliders, visible in the top right of Figure 5. These grid cells are separated by threads of P-positions, which are able to split and merge to form smaller and larger threads, and sometimes seem to disappear.

Threads typically have a measurable (vertical, horizontal, and/or diagonal) thickness, which is added when they merge, as happens frequently just after the fray, as in Figure 5. For example, in the left-hand warp the threads have integer vertical thicknesses, that is, each thread has a fixed number of P-positions that occur in every column. Furthermore, this fixed number is always a Fibonacci number.

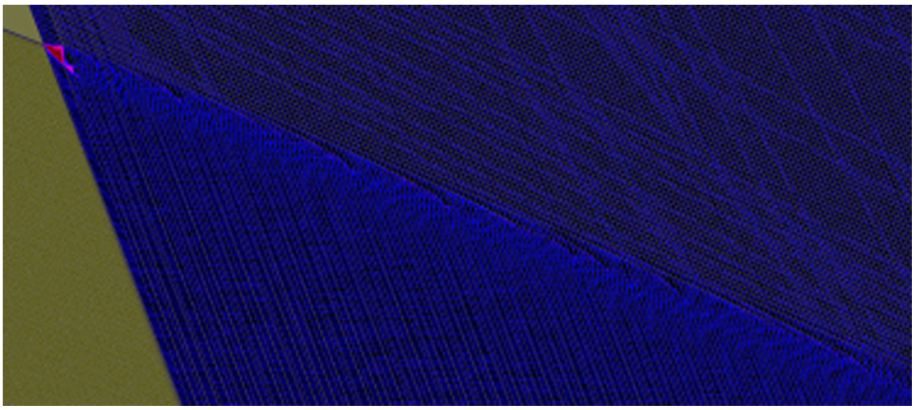


Fig. 5. Game 9999. The armpit in the upper left corner is generated when the rift hits the seam. This starts the fray, which separates the meta-glider behavior in the fabric from the merging threads of the warp. If viewing this document electronically, zoom in for detail.

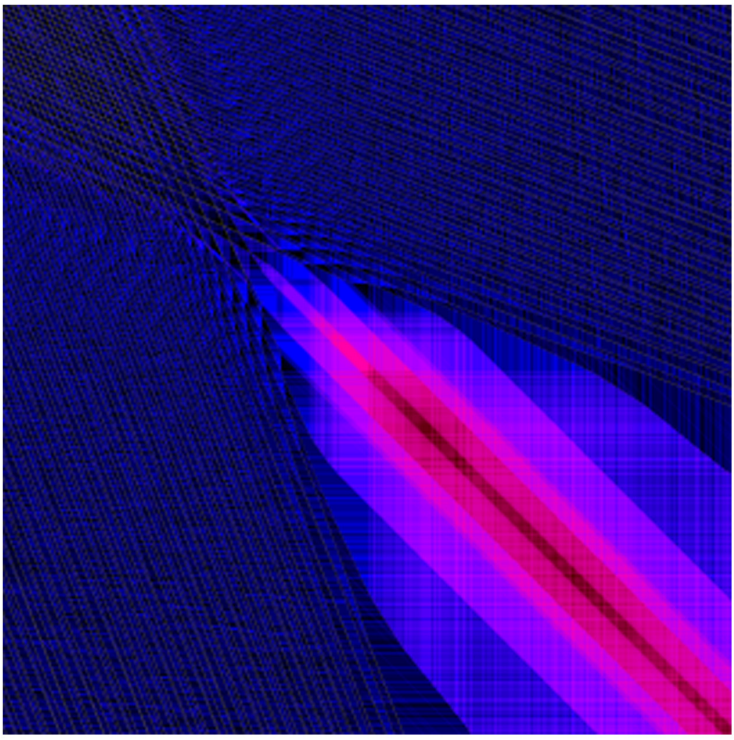


Fig. 6. Game 40003: A prism and its light beam. The slightly higher palace numbers near the main diagonal can give the impression of a beam of light being emitted by the prism. When the two fraying edges of the fabric meet at the prism, such higher palace numbers are often produced. If viewing this document electronically, zoom in for detail.

3.3 Region Prefix Properties

If we look around the nose, we see one of three pictures, depending on the value of k mod 3, because this determines the precise pixel arrangement of the casing at the nose. These three shapes are shown in Figure 7. Since there are only three possibilities for the hood boundary shape at the nose, and the CA rule can then be used to produce the épaulettes and fabric without knowing k, we see that despite the chaotic nature of the fabric, there are in fact only three fabric patterns that can be produced. Figures 3 and 4 both show examples of how the full fabric area matches between different games with k congruent modulo 3.

Using the CA, starting from an infinite casing pattern, we can make any of the three fabric patterns in an infinitely large version. The fabric patterns that we see in practice are simply prefixes of one of these three infinite fabrics.

The arms similarly can be formed by the CA rule from the shoulder, and in this case there is only one possible pattern. Figure 4 shows how this pattern remains very stable as k increases.

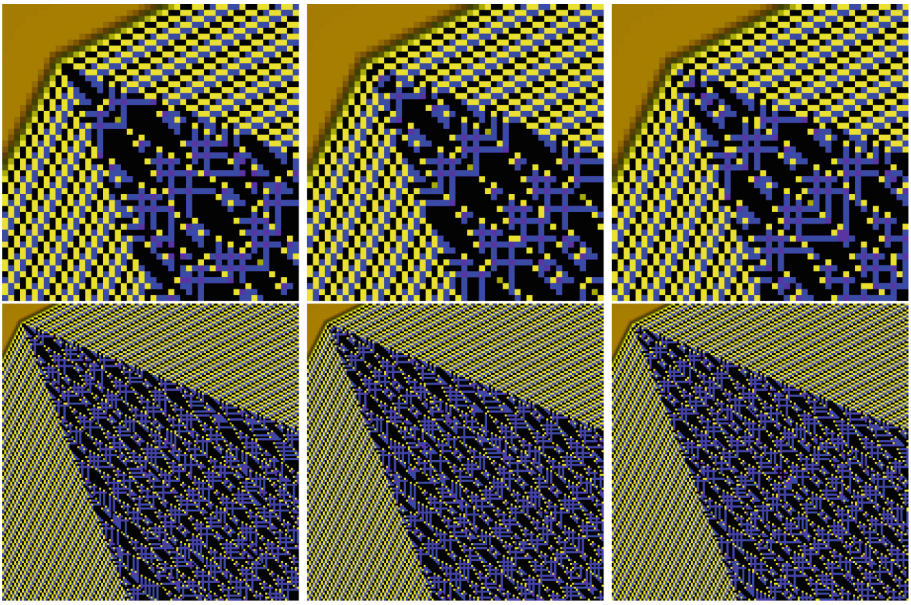


Fig. 7. The 3 fabrics. **(left)** $k = 0 \bmod 3$. **(center)** $k = 1 \bmod 3$. **(right)** $k = 2 \bmod 3$. The upper row is the same as the lower row, but more zoomed in.

3.4 Properties of the Skin

Where the arm borders the outer space, the arm grows a periodic skin (see Figure 8), which is slightly separated from the rest of the arm, except that it emits a vertical (in the case of the upper arm) line once per period. This skin consists of solidly packed P-positions, with a vertical thickness of f_{2n} (or $f_{2n}+1$ where a line is emitted), a diagonal thickness of f_{2n+1}, a horizontal thickness of f_{2n+2}, a horizontal period of f_{2n+3}, and a vertical period of f_{2n+1}, where f_{2n} is the $2n^{\text{th}}$ Fibonacci number. The skin originally forms at the top of the arm with $n = 1$, and after about 200 pixels (horizontally, about 80 vertically) it changes to the form with $n = 2$, then after about 7700 pixels it changes to the form with $n = 3$. We conjecture that for large k, it will continue to thicken in this manner, with n increasing by one each time, approaching a slope of ϕ^2. In the other direction, one can even see the $n = 0$ stage of this pattern as part of the épaulette at the start of the arm for the first 10 pixels or so, although it is not clearly visible due to the lack of black pixels between this skin and the rest of the arm.

The boundary between the skin (which has palace numbers $k-2$ and $k-1$) and the outer space (which has constant palace number k) consists of steps of width 2 or 3. Each time the skin thickens, the pattern of steps expands according to the rule $\{2 \to 23, 3 \to 233\}$, starting from the pattern of just 2 for the skin $n = 0$. The skin pattern of positions of palace numbers $k-2$ and $k-1$ can be computed from this pattern of steps and from the assumption that there are

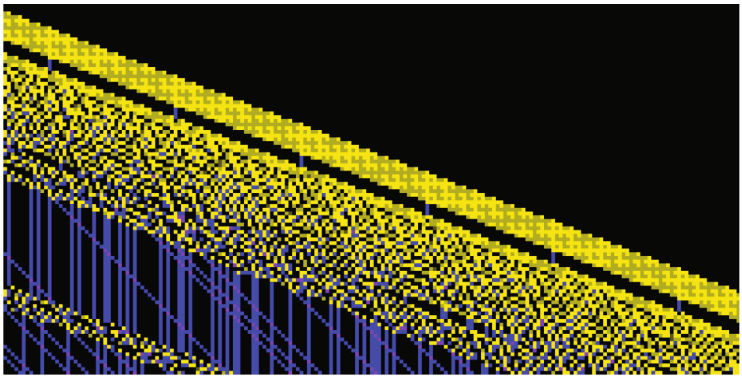


Fig. 8. Skin pattern

$k - f_{2n+3}$ palaces to the left of the skin in each row. This pattern is rotationally symmetric within each period (between the columns that produce the vertical lines), and with each thickening follows the expansion rule (writing -2 for $k-2$, etc.)

$$-2 \to \begin{pmatrix} -2 & -1 \\ -2 & -2 \end{pmatrix}, \begin{pmatrix} -2 & -2 & -1 \\ -2 & -2 & -2 \end{pmatrix}, \begin{pmatrix} -1 & -1 \\ -2 & -1 \\ -2 & -2 \end{pmatrix}, \begin{pmatrix} -2 & -2 & -1 \\ -2 & -2 & -2 \\ -2 & -1 & -1 \end{pmatrix},$$

$$-1 \to \begin{pmatrix} -1 & -1 \\ -2 & -1 \end{pmatrix}, \begin{pmatrix} -2 & -2 & -1 \\ -2 & -1 & -1 \end{pmatrix}, \begin{pmatrix} -1 & -1 \\ -2 & -1 \\ -1 & -1 \end{pmatrix}, \begin{pmatrix} -2 & -2 & -1 \\ -1 & -1 & -1 \\ -2 & -1 & -1 \end{pmatrix},$$

where one of the four expansion matrices is chosen based on the required size. The widths are determined by the step pattern, and the heights are partially defined by the property that all matrices produced by a given row should have a row in common.

4 Conjectures and Questions

Most of the observations in Section 3 may be viewed as open problems. Here we list of a few.

- Is it the case that a sufficiently thick arm will grow thicker and thicker skin over time?
- The arm's skin (as observed for $n = 1$ and $n = 2$) has a vertical thickness of f_{2n} (except where a line is emitted), a diagonal thickness of f_{2n+1}, a horizontal thickness of f_{2n+2}, a horizontal period of f_{2n+3}, and a vertical period of f_{2n+1}, where n is the thickness level (here f_i is the i^{th} Fibonacci number). Can this pattern be explained, and does it continue?

- Are the inner sectors beyond the fabric essentially different for different k, starting with distinct armpits (see Figure 5), and eventually producing ever different and irregular innermost P-threads?
- Are the armpits the unique regions from which the queen views the most palaces (for large k)?
- Do the innermost P-threads always have slopes corresponding to algebraic numbers? If there is only one innermost upper P-thread, is the slope a root of a second degree polynomial with rational coefficients?
- How many non-periodic threads can there be for a given k? We conjecture at most two upper ones, which must be the inner ones; with all others eventually becoming periodic.
- Why is it that the threads of the warp merge in such a way that their thickness is always a Fibonacci number?

References

CoLaNe. Cook, M., Larsson, U., Neary, T.: Generalized cyclic tag systems, with an application to the blocking queen game (in preparation)

Fin12. Fink, A.: Lattice games without rational strategies. J. of Combin. Theory, Ser. A **119**(2), 450–459 (2012)

Fra73. Fraenkel, A.S.: Complementing and exactly covering sequences. J. of Combin. Theory, Ser. A **14**(1), 8–20 (1973)

Gur10. Gurvich, V.: Further generalizations of Wythoff's game and minimum excludant function, RUTCOR Research Report, 16–2010, Rutgers University

GaSt04. Gavel, H., Strimling, P.: Nim with a modular Muller twist. Integers **4**(G4) (2004)

HeLa06. Hegarty, P., Larsson, U.: Permutations of the natural numbers with prescribed difference multisets. Integers **6**, Paper A3 (2006)

HoRe01. Holshouser, A., Reiter, H.: Problems and solutions: Problem 714 (Blocking Nim). The College Mathematics Journal **32**(5), 382 (2001)

HoRe. Holshouser, A., Reiter, H.: Three pile Nim with move blocking. http://citeseer.ist.psu.edu/470020.html

Lan02. Landman, H.A.: A simple FSM-based proof of the additive periodicity of the Sprague-Grundy function of Wythoff's game. In: Nowakowski, R.J. (Ed.) More Games of No Chance, vol. 42. MSRI Publications (Cambridge University Press), pp. 383–386 (2002)

Lar09. Larsson, U.: 2-pile Nim with a restricted number of move-size imitations. Integers **9**(G4), 671–690 (2009)

Lar11. Larsson, U.: Blocking Wythoff Nim. Electron. J. Combin. **18**(1), P120 (2011)

Lar13. Larsson, U.: Impartial games emulating one-dimensional cellular automata and undecidability. J. of Combin. Theory, Ser. A **120**(5), 1116–1130 (2013)

Lar15. Larsson, U.: Restrictions of m-Wythoff Nim and p-complementary Beatty sequences. In: Games of no Chance 4. MSRI Publications (Cambridge University Press)

LaWa13. Larsson, U., Wästlund, J.: From heaps of matches to the limits of computability. Electron. J. Combin. **20**(3), P41 (2013)

SmSt02. Smith, F., Stănică, P.: Comply/constrain games or games with a Muller twist. Integers **2**(G4) (2002)

Wyt07. Wythoff, W.A.: A modification of the game of Nim. Nieuw Arch. Wisk. **7**, 199–202 (1907)

Hard Core via PCA: Entropy Bounds

Kari Eloranta(✉)

Institute of Mathematics, Aalto University, Espoo, Finland
kari.v.eloranta@gmail.com

Abstract. We establish bounds for the entropy of the Hard Core Model/ Independent Sets on a few 2-d lattices. Our PCA-based sequential fill-in method yields an increasing sequence of lower bounds for the topological entropy. Additionally the procedure gives some insight on the support of the measure of maximal entropy. The method also applies to other lattices and models with appropriate sublattice splitting.

Keywords: Hard core model · Independent sets · Topological entropy

1 Introduction

The **Hard Core Model/Golden Mean Subshift/Independent Sets** is a highly useful model in various disciplines as witnessed by its many appearances under distinct names in fields like Statistical Mechanics/Symbolic Dynamics/Theoretical Computer Science respectively. Simply described, one distributes 0's and 1's on each vertex of a given graph and requires an **exclusion rule** to hold everywhere: no two 1's can be nearest graph neighbors ([1]).

On a k-partite graph it is natural to associate with it a probabilistic cellular automaton (PCA) which updates each of the k subgraphs from the others with a local rule as follows: in an all-0 neighborhood update the center vertex to 1 with probability $u \in (0,1)$, otherwise update to 0. Running a sequential update with the PCA (from any input) through the subgraphs yields eventually any Hard Core configuration with positive probability (and no others).

The PCA behavior is of interest for different u-values. If u is near 1, we are in the **high density regime** and the characterization of the allowed Hard Core configurations is essentially a packing problem (studied e.g. in an earlier paper [6]). In the **dilute case** i.e. u is near 0, the configurations are essentially Bernoulli($\tilde{u}$)-fields (sometimes called Hard Square Gas), $0 < \tilde{u} < u$.

Here we concentrate on the intermediate **entropic regime**. The outstanding open question near $u = 1/2$ is the exponential size of the configuration space. In almost all two or higher dimensional set-ups the exact answer is unknown. We try to alleviate the situation a little bit by establishing a procedure to estimate the entropy from below and to characterize the typical configurations. For simplicity we restrict here the graphs to be 2-d lattices. This facilitates comparison to entropy approximations elsewhere via different methods (e.g. [2], [4], [7], [8], [9]). Our ideas also generalize to more complicated and higher dimensional set-ups. For a general percolation approach to Hard Core see e.g. [3].

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 85–98, 2015.
DOI: 10.1007/978-3-662-47221-7_7

1.1 Set-up

Let $\mathbf{L}$ be a 2-dimensional lattice. Subsequently we will consider mostly one of the regular lattices (square ($\mathbf{Z}^2$), honeycomb ($\mathbf{H}$) or triangular ($\mathbf{T}$)). In a few cases we illustrate the principles developed on more exotic stages like square lattice with the Moore neighborhood ($\mathbf{Z}^2$M, eight nearest Euclidean neighbors, a non-planar graph) or the Kagomé lattice ($\mathbf{K}$). Our method is not intrinsically 2-d but for understanding the clear geometry of 2-d serves best.

A configuration on $\mathbf{L}$ satisfying the **Hard Core Rule** is an element in $X = \{0,1\}^{\mathbf{L}}$ where no two 1's can be nearest neighbors. This rule can naturally be viewed as a zero range infinite repulsive potential i.e. a hard exclusion rule not unlike that in hard sphere packing. Call the collection of configurations $X_{\mathbf{L}}^{hc}$.

The exclusion rule naturally imposes a sublattice split on $\mathbf{L}$ if it is a k-partite graph. For example on $\mathbf{Z}^2$, a bipartite graph, one can man all sites on $\mathbf{2Z}^2$ ($\mathbf{Z}^2$ rescaled by $\sqrt{2}$ and rotated by 45°) with 1's and the rest of $\mathbf{Z}^2$ must then be all 0's. Call the former the **even sublattice**, $\mathbf{L}_e$ and the latter the **odd sublattice**, $\mathbf{L}_o$ (it is a $(1/2, 1/2)$-shifted copy of the former). In rendering these we will present the even/odd sublattices as **circle/dot sublattices**. In a similar fashion $\mathbf{H}$ splits into two identical sublattices and $\mathbf{T}$ (a tripartite graph) into three "thinned" copies of $\mathbf{T}$. Both in the dense packing regime of [6] and in the the entropic regime of this paper, this splitting will be highly relevant.

Let $X_0 \subset X = \{0,1\}^{\mathbf{L}}$. $|\cdot|$ means the cardinality and $x|_A$ means the restriction of x to the set A. The standard measure of richness of the configuration set is

Definition 1. *The* **topological entropy** *of the set X_0 is*

$$h_{X_0}^{top} = \lim_{n\to\infty} \frac{1}{n} \ln |\{x|_{A_n} \mid x \in X_0\}|$$

where $|A_n| = n$ and the sequence $\{A_n\}$ grows in a sufficiently regular fashion.

Remark. For the full shift on any lattice $h_X^{top} = \ln 2$ (indicating two independent choices per lattice site). If $L = \mathbf{Z}$ the Hard Core model is explicitly solvable and a standard transfer matrix argument implies that $h^{top} = \ln\left(\frac{1+\sqrt{5}}{2}\right)$ (e.g. [11]). For two and higher dimensional lattices the matrix argument breaks down and the exact value of the Hard Core topological entropy remains an unsolved problem except for $\mathbf{T}$ (see [1]). In this paper we try to approach and in particular approximate it in a novel way.

From the general theory of lattice dynamical systems ([11]) it is known that shift invariant probability measures on a space of configurations, $\mathcal{M}$, satisfy the maximum principle, $h^{top} = \sup_{\mathcal{M}} h_\mu$, where h_μ is the measure-entropy. The special measures yielding the equality are **measures of maximal entropy**. For two and higher dimensional systems they are in general not unique. In all our cases they are believed to be so, but we do not actually need to know this.

For the idea of our approach recall that given the entropy function $H(\mathcal{P}) = -\sum \mu(P_i) \ln \mu(P_i)$ for a partition $\mathcal{P}$, it holds $H(\mathcal{P} \vee \mathcal{P}') = H(\mathcal{P}) + H(\mathcal{P}'|\mathcal{P})$.

In the case of $\mathbf{Z}$ let $\mathcal{P}^{(e)} = \{\{x|x_0 = i\}\}$ and $\mathcal{P}^{(o)} = \{\{x|x_1 = i\}\}$ be the (generating) partitions for the even/odd sublattice for the left shift σ. $\mathcal{P} = \mathcal{P}^{(e)} \vee \mathcal{P}^{(o)}$ generates on $\mathbf{Z}$ and the conditional entropy formula above implies

$$h_\mu(\sigma) = \frac{1}{2} h_\mu(\sigma^2|\mathcal{P}) = \frac{1}{2} h_\mu^{(e)} + \inf_n \frac{1}{2n} H_\mu \left(\bigvee_{k=0}^{n-1} \sigma^{-2k} \mathcal{P}^{(o)} \Big| \bigvee_{k=0}^{n-1} \sigma^{-2k} \mathcal{P}^{(e)} \right). \quad (1.1)$$

The formula readily generalizes to $\mathbf{Z}^d$-actions, to multiple sublattices etc. The last expression is trivial only in the independent case (equals $\frac{1}{2} h_\mu^{(o)}$). It codes the additional entropy to be gained from the odd sublattice, *given the even*. In the case of Hard Core it is manageable since 1. the rule is of range one and 2. the hard exclusion greatly simplifies the computation of the conditional probabilities involved. This will become much more transparent in the subsequent analysis.

2 Lower Bounds

We now proceed to establish lower bounds for the topological entropy using the sublattice partition representation (1.1) and a sequential fill-in scheme to circumvent the dependencies. To keep the ideas clear we first present them in the bipartite the case and then comment on the k-partite cases.

Let N_e denote an all-0 nearest neighbor neighborhood of a site on the odd lattice in the even lattice. In the case of $\mathbf{Z}^2$ lattice the sites in N_e form the vertices of an **even unit diamond**, $\diamondsuit_e$ ($\diamondsuit_o$). On the honeycomb and triangular lattices these sites form triangular arrangements, $\triangle$ or $\triangledown$ or a hexagon.

It will become quite useful to think the fill-in in terms of forming a tiling. The pieces are 0/1-**tiles** which in $\mathbf{Z}^2$ case are either 0/1-diamonds (as above) depending on whether the center site carries 0 or 1. On the hexagonal and triangular lattices the tiles are 0/1-**(unit) hexes**. Once a sublattice is chosen, one can tile the plane using any combination of 0/1-tiles centered on the sublattice.

Recall that the Bernoulli measure with parameter p, $B(p)$, assigns 1's independently with probability p to each (sub)lattice site and 0's otherwise. Its entropy, denoted by $h_B(p)$, is $-p \ln p - (1-p) \ln (1-p)$.

Proposition 1. *The topological entropy of the hard core model on a lattice with a two-way sublattice split is given by*

$$h^{top} = \frac{1}{2} \Big\{ h^{(e)} + \mathbf{P}(N_e) \ln 2 \Big\}, \quad (2.1)$$

where $h^{(e)}$ is the entropy of the measure of maximal entropy computed from the even sublattice alone.

Proof. We attain the maximum entropy by first assigning the marginal of the measure of maximal entropy to the even lattice and then filling in the non-blocked sites on the odd lattice. These are centered at the even unit diamonds, which have probability $\mathbf{P}(N_e)$. $B(1/2)$ is the maximal entropy measure among $B(p)$, hence this on the allowed odd sites and the factor $h_B(1/2) = \ln 2$. □

The principle in the Proposition can be directly applied to square and honeycomb lattices. A further argument is required to cover all regular lattices. In the following result we present these arguments and further extend to Kagomé lattice, $\mathbf{K}$ (a tripartite graph), as well as to the square lattice with Moore neighborhood, $\mathbf{Z}^2\mathrm{M}$ (eight nearest neighbors in Euclidean metric, a 4-partite graph).

Theorem 1. *The topological entropy of the hard core model is bounded from below on the square ($m = 4$) and honeycomb ($m = 3$) lattices by*

$$\underline{h}_{\mathbf{Z}^2/\mathbf{H}}(p) = \frac{1}{2}\Big\{h_B(p) + (1-p)^m \ln 2\Big\}\ , \tag{2.2}$$

on triangular ($m' = 3$) and Kagomé ($m' = 2$) lattices by

$$\underline{h}_{\mathbf{T}/\mathbf{K}}(p,q) = \frac{1}{3}\Big\{h_B(p) + (1-p)^{m'}\big[\ h_B(q) + [1-(1-p)q]^2 \ln 2\ \big]\Big\} \tag{2.3}$$

and on $\mathbf{Z}^2\mathrm{M}$ lattice by

$$\begin{aligned}\underline{h}_{\mathbf{Z}^2\mathrm{M}}(p,q,r) = \frac{1}{4}\Big\{h_B(p) + (1-p)^2\Big[\ h_B(q) + [1-(1-p)q]^4\ h_B(r)\\ + (1-p)^2(1-q)^2\big[1-(1-(1-p)q)^2 r\big]^2 \ln 2\ \Big]\Big\}\end{aligned} \tag{2.4}$$

where $p,\ q$ and $r \in (0,1)$.

Proof. The lower bounds (2.2) follow simply from (2.1) by assigning $B(p)$ to the even sublattice since then $\mathbf{P}\,(N_e) = (1-p)^{|N_e|}$ where the exponent is the number of elements in N_e in $\mathbf{Z}^2$ and $\mathbf{H}$ respectively.

On the triangular lattice the sublattice partition is three-way. We call the parts the dot, circle and triangle sublattices. They are filled in three stages in the order $\circ \to \bullet \to \triangleright$. See Figure 1a and b for the notation and arrangement of the sublattices in a neighborhood of a triangle site.

Suppose the three sublattices are initially all empty. First fill-in the circle lattice with $B(p)$, hence the entropy contribution $\frac{1}{3}h_{B(p)}$. Then fill-in all dot sites centered at $\bigtriangledown$ with $B(q)$, this implies the entropy increase $\frac{1}{3}(1-p)^3 h_{B(p)}$ from the dot lattice.

To update the center site which is a triangle we need to know that its value is not forced. Hence

$$\begin{aligned}&\mathbf{P}(\text{center triangle not forced by nearest neighbor circle or dot})\\ &= \mathbf{P}(\text{no } 1'\text{s in the hexagon of nearest neighbors of the triangle})\\ &= \mathbf{P}(\triangle = \underline{0} \text{ and } \bigtriangledown = \underline{0}) = \mathbf{P}(c_2 = c_4 = c_5 = 0 \text{ and } d_1 = d_2 = d_3 = 0)\\ &= \mathbf{P}(d_1 = d_2 = d_3 = 0 \mid c_2 = c_4 = c_5 = 0)\ \mathbf{P}(c_2 = c_4 = c_5 = 0)\\ &= \mathbf{P}(d_1 = d_2 = d_3 = 0 \mid c_2 = c_4 = c_5 = 0)\ (1-p)^3\\ &= [\ \mathbf{P}(d_1 = 0 \mid c_2 = c_4 = c_5 = 0)\ ]^3\ \ (1-p)^3\\ &= [\ \mathbf{P}\,(c_1 = 1 \text{ or } \{c_1 = 0 \text{ and } d_1 = 0\} \mid c_2 = c_4 = c_5 = 0)\ ]^3\ \ (1-p)^3\\ &= [p + (1-p)(1-q)]^3\ (1-p)^3\end{aligned} \tag{2.5}$$

which together with the choice $B(1/2)$ on the non-blocked dots gives (2.3).

The Kagomé lattice argument is similar to the triangular one. There are three sublattices involved, all identical copies of the Kagomé, only thinned and reoriented. For the nearest neighbors of a triangle-site see Figure 1c. Again we fill in the order $\circ \to \bullet \to \triangleright$. In the last stage the probability of the triangle site being unforced is now

$$\begin{aligned}
&\mathbf{P}(c_2 = c_4 = 0 \text{ and } d_1 = d_2 = 0) \\
&= [\ \mathbf{P}\,(c_1 = 1 \text{ or } \{d_1 = 0 \text{ and } c_1 = 0\} \mid c_2 = c_4 = 0)\ ]^2\ (1-p)^2 \\
&= [p + (1-p)(1-q)]^2\ (1-p)^2
\end{aligned}$$

In the case of the square lattice with Moore neighborhood there is a four-way sublattice partitioning. We denote and fill them as follows: $\circ \to \bullet \to \triangleright \to \diamond$ (see Fig. 1d).

The two first terms of the formula (2.4) are straightforward since circles are laid independently and each dot has exactly two circle neighbors. Furthermore as above we can show that $\mathbf{P}(\triangleright \text{ unforced}) = (1-p)^2\,[p + (1-p)(1-q)]^4$.

For the diamond site at the center of Fig. 1d to contribute to the entropy we need to know the probability that it is unforced i.e. all entries in the punctured square S rendered with dotted line in Fig 1d. are 0's:

$$\begin{aligned}
\mathbf{P}\big(S = \underline{0}\big) &= \mathbf{P}\big(\text{all } \triangleright, \bullet \in S \text{ are } 0 \mid \text{all } \circ \in S \text{ are } 0\big)\ (1-p)^4 \\
&= \mathbf{P}\big(\triangleright \in S \text{ are } 0 \mid \circ, \bullet \in S \text{ are } 0\big)\ (1-p)^4(1-q)^2 \\
&= \mathbf{P}\big(d_1 = 1 \text{ or } \{d_1 = d_2 = 0 \text{ and } t_1 = 0\} \mid \circ, \bullet \in S \text{ are } 0\big)(1-p)^4(1-q)^2 \\
&= \big[\ \mathbf{P}\big(d_1 = 1 \mid \circ, \bullet \in S \text{ are } 0\big) \\
&\quad + \mathbf{P}\big(d_1 = d_2 = 0 \text{ and } t_1 = 0) \mid \circ, \bullet \in S \text{ are } 0\big)\big]^2\ (1-p)^4(1-q)^2
\end{aligned} \tag{2.6}$$

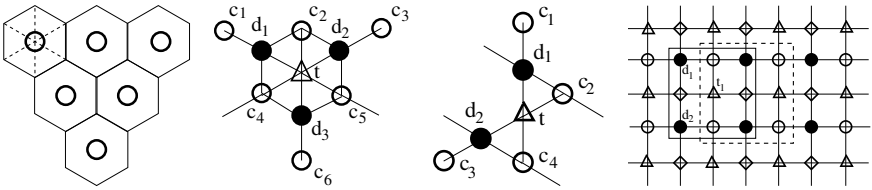
One can compute the two probabilities in the last expression to be

$$\begin{aligned}
&2p(1-p)q + (1-p)^2(2-q)q \quad \text{and} \\
&\big[p^2 + 2p(1-p)(1-q) + (1-p)^2(1-q)^2\big]\,(1-r)
\end{aligned}$$

respectively. From these the formula in the square brackets in (2.6) can finally be simplified to the form $1 - [1-(1-p)q]^2 r$. QED

The entropy bounds in (2.2) - (2.4) can be maximized with respect to the parameters using standard optimization routines in a desktop machine.

In the square lattice case the topological entropy has been computed to a great accuracy (e.g. in [2] to some 40 decimal places) using the corner transfer matrix methods. These numerical studies attack the problem in a very different way. Our aim is not to compete in decimal count but rather present an alternative method applicable in many lattice set-ups to estimate the entropy which simultaneously yields some explicit information on the generic configurations/the measure of maximal entropy.



(a) (b) (c) (d)

Fig. 1. Tiling of the triangular lattice with hexagonal 1-tiles and neighborhoods in triangular, Kagomé and $\mathbf{Z}^2$M cases

Table 1. First lower bounds for Hard Core topological entropy and the corresponding sublattice densities for some 2-d lattices. To the right we have indicated the best numerical estimates for the entropy and corresponding density (in parenthesis) found in the literature.

L	max $\underline{h}_{\mathbf{L}}$	sublattice densities	best estimates
$\mathbf{Z}^2$	0.3924	$(0.1702, 0.2370)$	$0.4075\ (0.2266)_{[9],[2]}$
H	0.4279	$(0.2202, 0.2371)$	$0.4360\ (0.2424)_{[2]}$
T	0.3253	$(0.1457, 0.1559, 0.1517)$	$0.3332\ (0.1624)_{[2]}$
K	0.3826	$(0.1944, 0.1948, 0.1866)$	
$\mathbf{Z}^2$M	0.2858	$(0.119, 0.127, 0.130, 0.126)$	

The measure of maximal entropy doesn't need to be unique for a 2-d lattice model but in the case of hard square gas it is. This follows from the Dobrushin criterion ([5], [10]). Using this knowledge and the results above we now establish bounds for the density of 1's in the generic configurations. The exact value of the upper bound in the following result is in the Proof but we prefer to give the statement in this more explicit form.

Proposition 2. *In the square lattice case the density of 1's at the equilibrium is in the interval* $(0.21367, 0.25806)$.

Proof. Let ρ_e be the density of 1's on the even lattice and let c denote the expected number of 0's that a 1 forces on the odd lattice. Since exactly half of the non-forced sites will be 1's it must by the uniqueness of the measure of maximal entropy hold that $(2+c)\rho_e = 1$. Hence under it

$$\mathbf{P}(x_i = 0) = \frac{1+c}{2+c}, \qquad \mathbf{P}(x_i = 1) = \frac{1}{2+c} \quad \text{and} \quad \mathbf{P}\left(\diamondsuit_e\right) = \frac{2}{2+c}$$

on both lattices. The last one is due to exactly half of $\diamondsuit_e$ giving rise to all the 1's on the other sublattice. (0-diamond as defined in the beginning of the section).

The entropy of any distribution on the even lattice with 1-density ρ_e is bounded from above by the entropy of the Bernoulli distribution with parameter

ρ_e. Hence the total entropy at that 1-density is bounded from above by

$$\frac{1}{2}\left(h_B\left(\frac{1}{2+c}\right)+\frac{2}{2+c}\ln 2\right).$$

This expression bounded by h^{top} of Table 1 ([2] or [9]) yields an upper bound for c, 2.6801 which in turn gives the lower bound for ρ_e.

The upper bound for ρ_e follows from a lower bound for c which we establish using a monotonicity argument. The 1's on say the even lattice are $B(1/2)$-distributed on the non-forced sites. Call this set F. Pick a site in it which has symbol 1. How many sites will this entry block? Let F' be a superset of F. Then clearly $\mathbf{E}(c|\ F) \geq \mathbf{E}(c|\ F')$ as in a bigger domain the 1 is more likely to share the blocking with a nearest neighbor 1 on the same sublattice. Hence a lower bound is obtained by calculating the blocking for a 1 with its eight nearest neighbors also in F. Enumerating the 2^8 possible neighborhood configurations and weighting them uniformly according to the $B(1/2)$-distribution we get the lower bound for c: 15/8. This in turn implies the upper bound for ρ_e, 8/31. QED

Since our first estimate for the lower bound on $\mathbf{Z}^2$ is associated with densities incompatible with Proposition 2 we will try out a symmetric variant of the theme. The (near) equality of the densities on the sublattices should be a natural property of a measure corresponding to a good lower bound since the measure of maximal density is believed to be unique in all our cases. In the last three cases in Table 1. the non-equality of the densities isn't far off but for the first two we present an "equalization".

Proposition 3. *To achieve equal densities of 1's on each of the sublattices one needs to replace the $B(1/2)$ distribution in the last stage of the measure construction by $B(p')$ and thereby $\ln 2$ in (2.2) by $h_{B(p')}$, where $p' = p(1-p)^{-|N_e|}$.*

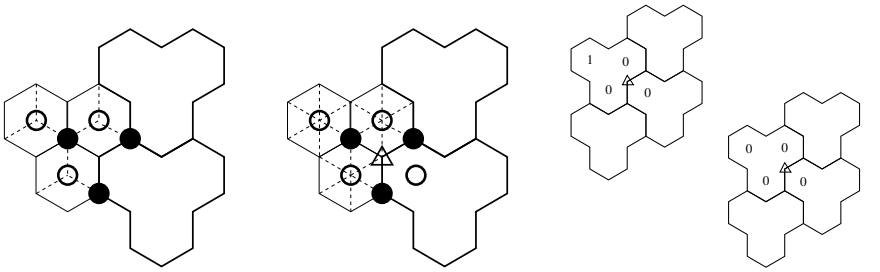
Proof. In the case of two sublattices after $B(p)$ distribution of 1's on the even lattice there are a density of $(1-p)^{|N_e|}$ unforcing neighborhoods on this sublattice. These have to produce the correct density of 1's on the odd lattice, hence we need the even lattice flip probability p' to satisfy $p'(1-p)^{|N_e|} = p$. QED

Using Proposition 3 one can optimize the square lattice topological entropy bound to (a slightly worse value) 0.3921 at common density level 0.2015. In view of Proposition 2 this indicates that the entropy generating 1's are not yet packed in densely enough. In the case of the honeycomb lattice the corresponding values are 0.427875 at 0.2284.

3 Higher Order Blocks

To improve the entropy bounds and more importantly to get some insight into the character of the measure of maximal entropy we now consider more complicated optimization schemes involving Bernoulli-distributed blocks on sublattices. We first illustrate the ideas on hexagonal and triangular lattices.

A **three-hex** is obtained by gluing together three unit hexes so that each has two joint sides. Figure 2a. illustrates three such three-hexes next to each other (for reference lattice edges are indicated as thin dotted lines in one of the unit hexes). Note that the unit tiles on each of them are all centered on the same sublattice, the circle lattice in this case (call the tile a **circle three-hex**). The dots of the other sublattice are all in the centers of the three-hexes or on their extremities (three of them are indicated). Three-hexes of the same orientation obviously tile the plane.



(a) a (b) b (c) c

Fig. 2. 3-hex arrangements in hexagonal and triangular cases

Let $B(\mathbf{p}),\ \mathbf{p} = (p_0, p_1, p_2, p_3)$ be the Bernoulli distribution on circle three-hexes with the probability that the three-hex has exactly k 1-tiles in it in a given orientation being p_k (so $p_0 + 3p_1 + 3p_2 + p_3 = 1$). Its entropy is then $h_B^{(3)}(\mathbf{p}) = -p_0 \ln p_0 - 3p_1 \ln p_1 - 3p_2 \ln p_2 - p_3 \ln p_3$.

Theorem 2. *Let $a(\mathbf{p}) = p_0 + 2p_1 + p_2$. For the hexagonal lattice the Hard Core entropy is bounded from below by*

$$\underline{h}_{\mathbf{H}}^{(3)}(\mathbf{p}) = \frac{1}{6}\Big\{h_B^{(3)}(\mathbf{p}) + \big[p_0 + 2a(\mathbf{p})^3\big] \ln 2\Big\} \tag{3.1}$$

and for the triangular lattice a corresponding bound is

$$\begin{aligned}\underline{h}_{\mathbf{T}}^{(3)}(\mathbf{p}, q) = \frac{1}{9}\Big\{&h_B^{(3)}(\mathbf{p}) + \big[p_0 + 2a(\mathbf{p})^3\big]\, h_B(q) \\ &+ 3\,[p_1 + p_0(1-q)]\, a(\mathbf{p})^3 (2-q)^2 \ln 2\Big\}\end{aligned} \tag{3.2}$$

where $p_i,\ q \in (0, 1)$.

Proof. For the construction of the measure we will fill in the lattice in the order $\circ \to \bullet$. If the circle three-hexes are distributed Bernoulli with parameter $\mathbf{p}$ the entropy contribution from the circle lattice will be $\frac{1}{2}\frac{1}{3}h_B(\mathbf{p})$ where the factors result from the sublattice density and the fact that we distribute triples. As in Proposition 1. in the next stage the maximal entropy choice for the unforced

sites on the dot lattice is the $B(1/2)$ distribution. The total density of sites available is computed at two different types of dot sites (as in Fig. 2a, the three dots indicated) and is $\frac{1}{3}\left[p_0 + 2\left(p_0 + 2p_1 + p_2\right)^3\right]$ where the coefficient 2 and the power 3 follow from the fact that at two of the three dot sites three adjacent three-hexes coincide. These formulas combined and simplified yield (3.1).

On the triangular lattice a third sublattice enters and the fill-in order is then $\circ \to \bullet \to \triangleright$. The entropy contribution from the Bernoulli circle three-hexes is now $\frac{1}{3}\frac{1}{3}h_B(\mathbf{p})$ since each sublattice is identical, hence of density 1/3.

In the second stage the unforced dot sites are filled with $B(q)$ distribution. Their density is computed as above to be $\frac{1}{3}\left[p_0 + 2\left(p_0 + 2p_1 + p_2\right)^3\right]$, hence the entropy contribution from dot lattice will be this expression multiplied by $\frac{1}{3}B(q)$.

In the final stage the unforced triangle sites are filled by $B(1/2)$. Their density in the full lattice is

$$\begin{aligned}&\frac{1}{3}\mathbf{P}(\text{nearest neighbor } \circ \text{ and } \bullet \text{ sites all } 0'\text{s})\\ &\quad= \frac{1}{3}\Big\{p_1(p_0+2p_1+p_2)\left[(p_1+2p_2+p_3)+(p_0+2p_1+p_2)(1-q)\right]^2 \qquad (3.3)\\ &+p_0(p_0+2p_1+p_2)(1-q)\left[(p_1+2p_2+p_3)+(p_0+2p_1+p_2)(1-q)\right]^2\Big\},\end{aligned}$$

which results from considering the two different arrangements of four neighboring three-hexes as shown in Fig. 2c. (top and bottom cases for the top and bottom expressions in (3.3)). The formulas merged and simplified result in (3.2). QED

Table 2. Optimized lower bounds and densities for three-hex Bernoulli blocks

L	max $\underline{h}_{\mathbf{L}}$	(p_0, p_1, p_2, p_3), q	sublattice densities
H	0.4304	$(0.504, 0.110, 0.048, 0.021)$	$(0.2276, 0.2376)$
T	0.3265	$(0.64, 0.092, 0, 025, 0.010)$, 0.25	$(0.153, 0.155, 0.151)$

Remarks. 1. The Kagomé lattice case is treated in an analogous fashion to **T**. 2. Note that apart from improvements in the entropy bounds, almost all of the sublattice densities have increased (in comparison to values in Table 1) indicating a better packing of the 1's on the sublattices. Moreover they have significantly less variation which is to be expected since the densities are equal for the measure of maximal entropy.

Let us now return to our original motivation, the Hard Core on the square lattice. Compounding the principles above and some further ideas we will implement an increasing sequence of lower bounds converging to the topological entropy. Along the way we'll get more explicit information on the configurations favored by the measure of maximal entropy.

1-tiles in the $\mathbf{Z}^2$ case are diamonds of side length $\sqrt{2}$ centered on either of the two sublattices. ***k*-omino** is formed by gluing together k such 1-tiles along

edges. If $k = n^2$ and the 1-tiles are in a diamond formation we call them a $n \times n$ **-blocks**. There are 2^{n^2} of them. The optimization results in Section 2 were for the 1×1-blocks.

Consider next 2×2 -blocks. There are 16 of them, but after assuming isotropy for them i.e. that blocks that are rotations of each other are distributed with equal probability (inevitable when measure of maximal entropy is unique), there are only five free parameters for Bernoulli distribution $B(\mathbf{p})$ on them ($\mathbf{p} = (p_0, p_1, p_{21}, p_{22}, p_3, p_4)$, $p_0 + 4p_1 + 4p_{21} + 2p_{22} + 4p_3 + p_4 = 1$. Here the first subindex of p refers to the number of 1's in the block and p_{22} and p_{21} denote the two different arrangement of two 1's in the block (side by side and across)).

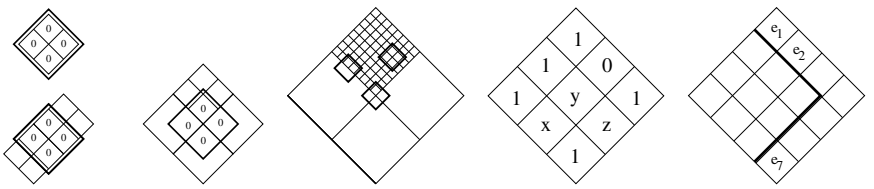
The entropy contribution from the even lattice (on which we distribute first the 1's using $B(\mathbf{p})$ is now

$$-\frac{1}{4}\Big\{p_0 \ln p_0 + 4p_1 \ln p_1 + 4p_{21} \ln p_{21} + 2p_{22} \ln p_{22} + 4p_3 \ln p_3 + p_4 \ln p_4\Big\} \ . \quad (3.4)$$

The density of the unforced sites on the odd lattice can be computed from the three cases indicated in Figure 3a. and results in

$$\frac{1}{4}\left\{p_0 + 2\left(p_0 + 2p_1 + p_{21}\right)^2 + \left(p_0 + 3p_1 + 2p_{21} + p_{22} + p_3\right)^4\right\} \ . \quad (3.5)$$

These combined yield a lower bound for h^{top}, which is optimized in Table 3 (second row).



(a) a (b) b (c) c (d) d

Fig. 3. $n \times n$ -blocks and update window in $\mathbf{Z}^2$ case. Reductions in a 3×3 -block and the extension.

In the block size 3×3 there are initially 511 free block probabilities to optimize. When rotational invariance is imposed the variable number is reduced and additionally we will expect blocks that are reflections of each other to have equal probabilities at the optimum. After these two types of symmetries are accounted the number of free variables will be 101.

In this size and in larger blocks another feature appears which enables further variable weeding. Consider the block in Figure 3c. The symbol assignments in sites x, y and z are irrelevant in the sense that the existing 1's in the 3×3 -block already force all the odd sites (to carry 0's) that x, y and z might force if any of them were 1's. Hence there are 2^3 blocks of equal probability. This combined with the symmetry assumptions above yield the total of 64 blocks with identical

Table 3. Optimized lower bounds and densities for a Bernoulli blocks on $\mathbf{Z}^2$

block size	max $\underline{h}_{\mathbf{Z}^2}$	sublattice densities	final/initial variables
1×1	0.392421	$(0.1702, 0.2370)$	1/1
2×2	0.39877	$(0.1993, 0.2254)$	5/15
3×3	0.4014	$(0.2073, 0.2254)$	46/511

probabilities at the optimum (this is actually the maximum reduction achievable in this block size). Combing through the set of all blocks for this feature will result in reduction by a factor about 11 to the final set of 46 variables. Their optimal values have been computed and the results are in Table 3.

Subsequently we call sites like x, y and z above **weak** with respect to the rest of the given block. Only the corner sites of a block cannot ever be weak.

The procedure of variable reduction is highly useful since the above rotational and reflection symmetry search as well as the weak site identification can be automated. Moreover the reduction improves significantly at every stage: for example in the next block size of 4×4 the initial variable number of 65.536 shrinks 66-fold to 991 final free variables.

Note also that the optima in block size $n \times n$ can be utilized as indicated in Figure 3d to initiate the search in the next larger block size. Once e.g. the 3×3 subblock optimum probability is known, the added half frame $(e_1, \ldots, e_7)$ should be assigned $B(p)$ entries with p computed from 3×3 blocks. With tailored optimization routines one should be able to deal with several thousands of variables in the larger block sizes. All the optimizations here were done with non-specialized code using *Mathematica*.

The optimal block probabilities satisfy a useful monotonicity property, that we establish next. For this let $B_i,\ i = 1, 2$ be $n \times n$ -blocks, whose subsets of 1's we refer to as $B_i^{(1)}$. There is a partial order on the blocks via $B_i^{(1)}$ using the ordinary set inclusion: $B_i \prec B_j$ if $B_i^{(1)} \subset B_j^{(1)}$. Let the optimal probabilities for the blocks be $\mathbf{p} = (p_0, p_1, p_2, p_3, \ldots, p_l),\ l = 2^{n^2}$ (no reductions done yet and no particular order in the coordinates).

Theorem 3. *Given two blocks B_1 and B_2 with optimal lower bound probabilities p_1 and p_2, if $B_1^{(1)} \subset B_2^{(1)}$ then $p_1 \geq p_2$. If $B_2^{(1)} \setminus B_1^{(1)}$ contains only weak sites with respect to $B_1^{(1)}$ then $p_1 = p_2$, otherwise $p_1 > p_2$.*

Proof. The optimal lower bound is given by $\underline{h}(\mathbf{p}) = \frac{1}{n^2}\{-\sum_i p_i \ln p_i + \mathbf{P}(N_e) \ln 2\}$ where N_e is the even 2×2 -diamond of all 0's as in Section 2. Let B_i be such that $B_1^{(1)} \subset B_2^{(1)}$ and let $p_1 = p + \epsilon$, $p_2 = p - \epsilon$, $0 \leq |\epsilon| < p$. Denote by $h_\epsilon(\mathbf{p})$ the lower bound with the given p_1 and p_2. To prove the result we will consider the entropy variation under the probability change of the two blocks: $\Delta h_\epsilon(\mathbf{p}) = h_\epsilon(\mathbf{p}) - h_0(\mathbf{p})$. More explicitly

$$\begin{aligned}\Delta h_\epsilon(\mathbf{p}) = \frac{1}{n^2}\Big\{ \big[&-(p+\epsilon)\ln(p+\epsilon) - (p-\epsilon)\ln(p-\epsilon) + 2p\ln p\big] \\ &+ \big[\mathbf{P}_{1,\epsilon}(N_e) - \mathbf{P}_1(N_e) \\ &+ \mathbf{P}_{2,\epsilon}(N_e) - \mathbf{P}_2(N_e) \\ &+ \mathbf{P}_{4,\epsilon}(N_e) - \mathbf{P}_4(N_e)\big]\ln 2\Big\},\end{aligned} \tag{3.6}$$

where $\mathbf{P}_{k,\epsilon}(N_e)$ and $\mathbf{P}_k(N_e)$ are the N_e-diamond probabilities computed from the different arrangements involving $k = 1, 2$ or 4 $n \times n$ -blocks as in Figures 3a and b, for the block probability choices $p \pm \epsilon$ or p for both.

By $\ln(1+x) \approx x$ the first square bracket behaves for small ϵ like $c_1\epsilon^2,\ c_1 < 0$.

If $B_2^{(1)} \setminus B_1^{(1)}$ contains only weak sites with respect to $B_1^{(1)}$ then the blocks B_i allow exactly the same sites to flip on the odd lattice hence each of the three last lines in (3.6) vanishes. The sole contribution to $\Delta h_\epsilon(\mathbf{p})$ then comes from the first square bracket and since this is negative for small but nonzero ϵ, it must be that $p_1 = p_2$ at the optimum.

If $B_2^{(1)} \setminus B_1^{(1)}$ contains non-weak sites with respect to $B_1^{(1)}$ let us first assume that they force k odd interior sites (recall that the odd sites are the vertices of the grids in Figure 3. There are $(n-1)^2$ such interior sites in a $n \times n$ -block). Let m be the number non-forced odd interior sites over block B_1. Then

$$\begin{aligned}\mathbf{P}_{1,\epsilon}(N_e) - \mathbf{P}_1(N_e) &= \Big(\ldots + \frac{(p+\epsilon)m}{(n-1)^2} + \frac{(p-\epsilon)(m-k)}{(n-1)^2} + \ldots\Big) \\ &\quad - \Big(\ldots + \frac{pm}{(n-1)^2} + \frac{p(m-k)}{(n-1)^2} + \ldots\Big) = \frac{k\epsilon}{(n-1)^2},\end{aligned}$$

where the dots refer to the contributions from the other blocks. All these terms cancel out, since the other block probabilities are identical.

If non-weak sites only force odd interior sites then by geometry of the set-up the two last lines in (3.6) are immediately zero. If e extra odd edge, off-corner sites are forced, similar argument than above gives estimate $(c_2 + \frac{e\epsilon}{4(n-1)})^2 - c_2^2, c_2 > 0$ for $\mathbf{P}_{2,\epsilon}(N_e) - \mathbf{P}_2(N_e)$ so the next to last line in (3.6) has the first order behavior $c_3\epsilon,\ c_3 > 0$. Some added bookkeeping yields $\mathbf{P}_{4,\epsilon}(N_e) - \mathbf{P}_4(N_e) = (c_4 + l\epsilon/4)^4 - c_4^4 \approx c_5\epsilon,\ c_5 > 0$ (l is the number of odd corners forced).

The leading orders for the terms in the square brackets in (3.6) together yield $c_1\epsilon^2 + d\epsilon,\ c_1 < 0,\ d \geq 0$. If there are non-weak sites in $B_2^{(1)} \setminus B_1^{(1)}$ with respect to $B_1^{(1)}$, then $d > 0$. Hence $p_1 > p_2$ must prevail at the optimum. QED

Remarks. 1. Intuitively the result says that if neither of two even blocks gives more subsequent choice on the odd lattice, for maximum entropy one should weight them equally. Otherwise one should favor the one giving more choice on the odd lattice.

2. One can readily see some chains imposed by the order in Figure 4: $0 \prec 12 \prec 23 \prec 31$ or $0 \prec 11 \prec 21/22 \prec 33$ etc. The monotonicity can be utilized in limiting the number of $n \times n$ -blocks optimized for larger values of n (dropping blocks

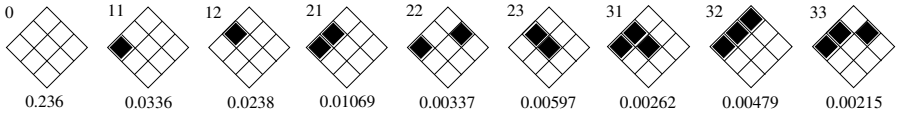


Fig. 4. Prevalent 3×3 -blocks with optimal probabilities without multiplicities

with least probability as dictated by the Theorem and with least multiplicity (most symmetric)).
The correlation structure inside the measure of maximal entropy gradually presents itself in the Bernoulli approximations when we consider higher order blocks. Correlations between the blocks are zero because of independence, but within the blocks it is worth making comparisons.

By adding the optimum probabilities of all 3×3 blocks at a given density level $k/9 = 0, 1/9, \ldots, 1$ we obtain the "density profile" of this measure (here k is the number of 1's in the block).

Suppose next that we generate the 3×3 blocks from 1×1 Bernoulli entries with the appropriate optimal p for 1's (as found above). By adding these up we again obtain a density profile, this time for the 1×1 optimal Bernoulli measure at the resolution level of the block size 3×3. The 3×3 blocks can of course be generated using the optimal 2×2 blocks as well and yet another density profile results. These three discrete plots are rendered as curves in Figure 5.

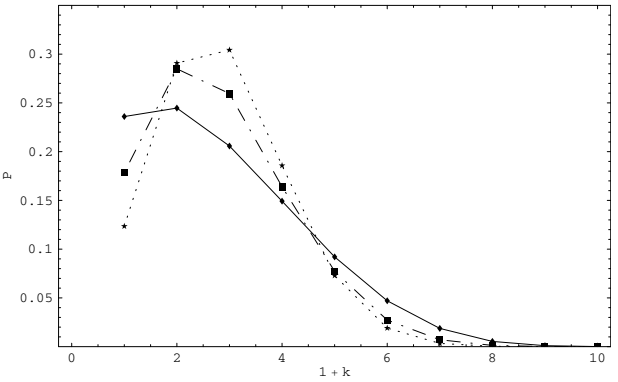


Fig. 5. 3×3 -block occupation probabilities from Bernoulli blocks of size 3×3 (diamond), 2×2 (square) and 1×1 (star). $k \in \{0, 1, \ldots, 9\}$ is the number of 1's in the block.

Perhaps the most notable feature here is the flattening of the distributions, as the block size increases i.e. the total block probabilities move towards the tails (while their means stay constant around 0.22). The curves cross between density levels $1/3 - 4/9$: below this cross over the shorter range Bernoulli measures favor light 3×3 blocks, above it they discount heavier blocks in comparison to the optimal 3×3 Bernoulli measure.

When examined closer one will see that the total probability of 3×3 blocks at a given density level essentially comes from at most three different kinds of local configurations (up to reductions above that is). These seem to be "grown": when moving from density level d to level $d + 1/9$ the high probability blocks are generated by adding a (contiguous) 1 into an existing high probability block. This mechanism cannot prevail when the 3×3 blocks are generated independently from smaller blocks. Consequently the small block curves in Figure 5. have suppressed tails. We expect this phenomenon to prevail in the higher order Bernoulli blocks as well and thereby to be a significant feature in the long range correlations of the measure of maximal entropy.

Acknowledgments. The author would like to thank the referees for valuable suggestions that led to the improvement of the exposition.

References

1. Baxter, R.J.: Exactly solvable models in statistical mechanics. Academic Press (1982)
2. Baxter, R.J.: Planar lattice gasses with nearest-neighbor exclusion. Annals of Combinatorics **3**(2–4), 191–203 (1999). MR1772345 (2001h:82010)
3. van den Berg, J., Steif, J.: Percolation and the hard-core lattice gas model. Stoc. Proc. and Appl. **49**, 179–197 (1994)
4. Calkin, N., Wilf, H.: The number of independent sets in a grid graph. SIAM J. Discr. Math. **11**, 54–60 (1998)
5. Dobrushin, R., Shlosman, S.: In: Fritz, J., Jaffe, A., Szasz, D. (eds.) Statistical Physics and Dynamical System, pp. 347–370. Birkhäuser (1985)
6. Eloranta, K.: Dense packing on uniform lattices. Journal of Stat. Phys. **130**(4), 741–755 (2008). arXiv:math-ph/0907.4247, MR2387564 (2009a:52016)
7. Engel, K.: On the Fibonacci number of an $M \times N$ lattice. Fibonacci Quarterly **28**(1) (1990)
8. Forchhammer, S., Justesen, J.: Entropy bounds for constrained two-dimensonal random fields. IEEE Tr. Info. Theory **45**(1), 118–127 (1999)
9. Milosevic, S., Stosic, B., Stosic, T.: Towards finding exact residual entropies of the Ising ferromagnets. Physica A **157**, 899–906 (1989)
10. Radulescu, D., Styer, D.: The Dobrushin-Shlosman Phase Uniqueness Criterion and Applications to Hard Squares. Journal of Stat. Phys. **49**, 281–295 (1987). MR0923859 (89d:82010)
11. Walters, P.: Ergodic theory. Springer (1982)

Classification of Elementary Cellular Automata Up to Topological Conjugacy

Jeremias Epperlein(✉)

Institute for Analysis and Center for Dynamics, TU Dresden, Dresden, Germany
jeremias.epperlein@tu-dresden.de

Abstract. Topological conjugacy is the natural notion of isomorphism in topological dynamics. It can be used as a very fine grained classification scheme for cellular automata. In this article, we investigate different invariants for topological conjugacy in order to distinguish between non-conjugate systems. In particular we show how to compute the cardinality of the set of points with minimal period n for one-dimensional CA. Applying these methods to the 256 elementary one-dimensional CA, we show that up to topological conjugacy there are exactly 83 of them.

1 Introduction

One-dimensional cellular automata can be topologically characterized as the continuous σ-commuting endomorphisms of the space $A^{\mathbb{Z}}$. Topological dynamics is therefore a natural framework to study their dynamics and has shown to be rather fruitful [6].

Topological dynamics in our sense is the study of compact metrizable space X together with a continuous map $F\colon X \to X$. The classical notion of isomorphism in this setting is that of a *topological conjugacy*. Two topological dynamical systems $F\colon X \to X$ and $G\colon Y \to Y$ are called conjugate, if there is a homeomorphism $\varphi\colon X \to Y$ such that $\varphi \circ F = G \circ \varphi$ [7]. It is easily seen that this defines an equivalence relation on topological dynamical systems. A natural problem now is to classify a certain class of such systems up to conjugacy.

This problem received a lot of attention for the case of subshifts of finite type. While there has been substantial progress and some powerful invariants have been found, there still remain many questions, ranging from the question if conjugacy is decidable for SFTs, to the question of deciding conjugacy for two concretely given edge shifts [2].

The corresponding problem of classifying CA up to topological conjugacy has up to now seen very little activity, although many classification schemes for CA have been proposed (see [8] for a survey). As a starting point we will classify the elementary one-dimensional cellular automata, mainly using the cardinality of the set of points with minimal period n, the Cantor-Bendixson derivative of the periodic points and various ad-hoc arguments.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 99–112, 2015.
DOI: 10.1007/978-3-662-47221-7_8

2 Definitions

Let A be a finite set with $|A| \geq 2$, which we call our *alphabet*. The set of bi-infinite sequences in A is denoted by $A^{\mathbb{Z}}$ and set of words over A is denoted by A^*. We endow $A^{\mathbb{Z}}$ with the product topology turning it into a Cantor space. On $A^{\mathbb{Z}}$ we define the shift map σ by $\sigma(x)_k = x_{k+1}$. The dynamical system $(A^{\mathbb{Z}}, \sigma)$ is called the *full shift over* A. Replacing $\mathbb{Z}$ by $\mathbb{N} = \{1, 2 \dots\}$ we get the dynamical system $(A^{\mathbb{N}}, \sigma)$, called the *one-sided full shift* over A. A *subshift* is a closed σ-invariant subset of $A^{\mathbb{Z}}$. The subshift X is a *subshift of finite type (SFT)* if there is a finite list of words such that X consists of all configurations not containing one of these words. For further information concerning shift spaces we refer to the standard reference [7].

Denote by $\mathcal{H}_A$ the set of all homeomorphisms from $A^{\mathbb{Z}}$ to itself, and denote by $\mathcal{CA}_A$ the set of all *cellular automata* (CA), that is, the set of all continuous maps $F\colon A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ with $\sigma \circ F = F \circ \sigma$. By the Curtis-Lyndon-Hedlund Theorem (see [7]) for each cellular automaton there is $r \in \mathbb{N}$, called its *radius*, and a *block map* $f\colon A^{2r+1} \to A$ with $F(x)_i = f(x_{i-r}, \dots, x_{i+r})$. The block map also induces a map $f\colon A^* \to A^*$ by $f(x_1, \dots, x_\ell) = (f(x_{1,\dots,2r+1}), \dots, f(x_{\ell-2r,\dots,\ell}))$.

Let (X, F) be a dynamical system. A point $x \in X$ is called *periodic with period* $n \in \mathbb{N}$, if $F^n(x) = x$. The minimal n, for which this equality holds is called its *minimal period*. We denote by $\mathrm{Per}_n(F)$ the set of all n-periodic points with respect to F and by $\widetilde{Per}_n(F)$ the set of all points with minimal period n. Thus $\mathrm{Per}_n(F)$ is the disjoint union of all sets in $\{\widetilde{Per}_k(F)\,;\, k \mid n\}$. We also write $\mathrm{Per}(F) = \bigcup_{n\in\mathbb{N}} \mathrm{Per}_n(F)$ for the set of all periodic points.

When counting periodic points we will encounter sets of countable cardinality and of cardinality equal to that of the continuum. We write these cardinalities with the help of the Hebrew letter $\beth$ (this notation is similar to the better known notion of the $\aleph$ cardinal numbers), so we define $|\mathbb{N}| =: \beth_0$ and $|\mathbb{R}| = |2^{\mathbb{N}}| =: \beth_1$.

For us *digraphs* are tuples $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}), t, h)$ with $V(\mathcal{G})$ and $E(\mathcal{G})$ being finite sets and $t, h\colon E(\mathcal{G}) \to V(\mathcal{G})$ being the *tail* resp. the *head* of an edge. Thus our edges are directed and we allow multiple edges as well as loops. A *path* $(\gamma_i)_{i\in I}$ with $I = \{1, \dots, k\}$ or $I = \mathbb{Z}$ is sequence of edges in $\mathcal{G}$ with $h(\gamma_i) = t(\gamma_{i+1})$. We denote by $\mathrm{Path}(\mathcal{G})$ the set of all bi-infinite pathes in $\mathcal{G}$. They form a SFT contained in $E(\mathcal{G})^{\mathbb{Z}}$, the *edge shift* of the graph. A *vertex path* in $\mathcal{G}$ is a sequence of vertices $(v_i)_{i\in\{1,\dots,k\}}$ such that for each $i \in \{1, \dots, k-1\}$ there is an edge $e_i \in E(\mathcal{G})$ with $t(e_i) = v_i$ and $h(e_i) = v_{i+1}$.

3 Topological Conjugacies

Since the composition of cellular automata gives another cellular automaton, the conjugation of a CA by an invertible one is again a cellular automaton. The simplest instance of this is conjugacy by a symbol permutation ("exchaning black and white"). Another way of getting a conjugate CA from a given one, is to reflect the rule ("exchanging left and right"). This is equivalent to conjugation by the reflection map $\tau\colon A^{\mathbb{Z}} \to A^{\mathbb{Z}}, \quad \tau(x)_k := x_{-k}$. See [3] for further properties

of these conjugacies. The next theorem will show, that these are in a sense the only general methods to get a conjugate CA from another.

Theorem 1. *Let* $\varphi\colon A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ *be a homeomorphism. Then the following are equivalent.*

(a) $\varphi \circ \mathcal{CA}_A \circ \varphi^{-1} \subseteq \mathcal{CA}_A$,
(b) $\varphi \circ \mathcal{CA}_A \circ \varphi^{-1} = \mathcal{CA}_A$,
(c) $\exists H \in \mathcal{CA}_A\colon \varphi = H$ *or* $\varphi = H \circ \tau$.

Proof (of Theorem 1). $(c) \Rightarrow (b)$ and $(b) \Rightarrow (a)$ are trivial.
$(a) \Rightarrow (c)$ Let F be an arbitrary CA. Then $G := \varphi \circ F \circ \varphi^{-1}$ is again a CA by the assumption and therefore commutes with σ. Hence

$$G = \sigma \circ G \circ \sigma^{-1} = \sigma \circ \varphi \circ F \circ \varphi^{-1} \circ \sigma^{-1},$$
$$F = \varphi^{-1} \circ \sigma \circ \varphi \circ F \circ \varphi^{-1} \circ \sigma^{-1} \circ \varphi$$

By setting $F = \sigma$, we see that $\varphi^{-1} \circ \sigma \circ \varphi$ is a CA. Now Ryan's theorem [9] tells us that the center of the group $\mathcal{H}_A \cap \mathcal{CA}_A$ consists only of powers of the shift, i.e. if an invertible CA commutes with all other invertible CA, it must be a power of the shift. Hence $\varphi^{-1} \circ \sigma \circ \varphi = \sigma^k$ for some $k \in \mathbb{Z}$ or equivalently $\sigma \circ \varphi = \varphi \circ \sigma^k$. This first of all implies that $k \neq 0$. Now take any point $y \in \mathrm{Per}_1(\sigma^k)$. Then $(\sigma \circ \varphi)(y) = (\varphi \circ \sigma^k)(y) = \varphi(y)$. Hence $\varphi(y) \in \mathrm{Per}_1(\sigma)$ and therefore φ defines an injective mapping from $\mathrm{Per}_1(\sigma^k)$ into $\mathrm{Per}_1(\sigma)$. Having a look at the cardinalities we see that $|A|^{|k|} = |\mathrm{Per}_1(\sigma^k)| \leq |\mathrm{Per}_1(\sigma)| = |A|$, implying $k = \pm 1$. In the case of $k = 1$ we are done. In the other case

$$\tau \circ \varphi^{-1} \circ \sigma \circ \varphi \circ \tau^{-1} = \tau \circ \sigma^{-1} \circ \tau^{-1} = \sigma,$$

hence $\varphi \circ \tau^{-1}$ is a CA. □

In the light of Theorem 1, we call a conjugacy $\varphi \in \mathcal{CA} \cup \mathcal{CA} \circ \tau$ a *strong conjugacy*. In Section 7 we will see conjugate cellular automata, that are not strongly conjugate.

4 Periodic Points and the Cantor-Bendixson Derivative

Consider two conjugate cellular automata F and $G := \varphi \circ F \circ \varphi^{-1}$ with $\varphi \in \mathcal{H}_A$. The first invariant of topological conjugacy normally considered is the number of periodic points, for if $F^k(x) = x$ then $(\varphi \circ F \circ \varphi^{-1})^k(\varphi(x)) = (\varphi \circ F^k)(x) = \varphi(x)$. Hence φ maps $\mathrm{Per}_k(F)$ bijectively onto $\mathrm{Per}_k(\varphi^{-1} \circ F \circ \varphi)$. While for shifts have only finitely many periodic points of a given period, this is in general not true any more for cellular automata.

To deal with this, we use standard cardinal arithmetic in order to extend the addition on $\mathbb{N}$ to $\mathcal{C} := \mathbb{N} \cup \{\beth_0, \beth_1\}$ by defining

$$\beth_1 + k := k + \beth_1 := \beth_1 \text{ for } k \in \mathcal{C}$$
$$\beth_0 + k := k + \beth_0 := \beth_0 \text{ for } k \in \mathbb{N} \cup \{\beth_0\}.$$

This turns $\mathcal{C}$ into a commutative monoid. The justification for this definition is given by the fact, that for $A_1, \dots, A_\ell$ pairwise disjoint sets with $|A_i| \in \mathcal{C}$ we have $|\bigcup_{i=1}^{\ell} A_i| = \sum_{i=1}^{\ell} |A_i|$. Notice however, that for two disjoint sets A, B with $|A|, |B| \in \mathcal{C}$ it is no longer possible to recover the cardinality of B from the knowledge of $|A|$ and $|A \cup B|$.

In settings where only a finite number of periodic points can occur, one can reconstruct $\widetilde{Per}_n(F)$ from the knowledge of $(\mathrm{Per}_k(F))_{k \le p}$ by $\widetilde{Per}_n(F) = \sum_{d|n} \mu(\frac{n}{d}) \mathrm{Per}_d(F)$, where μ is the Möbius function. This no longer works in our case where $(\widetilde{Per}_\ell(F))_{\ell \in \{1,\dots,n\}}$ carries more information then $(\mathrm{Per}_\ell(F))_{\ell \in \{1,\dots,n\}}$. As an easy example consider a cellular automaton F with $|\mathrm{Per}_1(F)| = \beth_1$. Then $|\mathrm{Per}_k(F)| = \beth_1$ for all $k \in \mathbb{N}$. Therefore we are interested in determining $(|\widetilde{Per}_\ell(F)|)_{\ell \in \{1,\dots,n\}}$, which is harder to calculate than $(|\mathrm{Per}_\ell(F)|)_{\ell \in \{1,\dots,n\}}$, though.

While these are already nice invariants they do not use the fact that φ is continuous at all but only its bijectivity. However, two spaces with cardinality $\beth_1$ might look rather different from a topological point of view. We therefore look at the set of all limit points $D(\mathrm{Per}_n(F))$ of $\mathrm{Per}_n(F)$ defined as follows.

Definition 2. *Let $B \subseteq A^{\mathbb{Z}}$. The set of* limit points *of B, also called its* Cantor-Bendixson derivative*, is defined by*

$$D(B) := \{x \in A^{\mathbb{Z}} \,;\, \exists (y_n)_{n\in\mathbb{N}} \text{ in } B \setminus \{x\} \colon y_n \xrightarrow{n\to\infty} x\} = \bigcap_{x \in B} \overline{B \setminus \{x\}}.$$

It is well known and easy to proof that $\varphi(D(B)) = D(\varphi(B))$ for any homeomorphism $\varphi \colon A^{\mathbb{Z}} \to A^{\mathbb{Z}}$. For a subshift (X, σ) and a subset $B \subseteq X$ we can characterize the set of limit points as follows. A configuration $(x_i)_{i\in\mathbb{Z}}$ is a limit point of B if for all $k \in \mathbb{N}$ the word $x_{-k,\dots,k}$ can be extended to a configuration in B different from X. We will use this characterization at the end of Section 5 to compute $D(\mathrm{Per}_n(F))$.

Now we fix $n \in \mathbb{N}$ and a cellular automaton $F \colon A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ with radius $r \ge 1$ and local rule $f \colon A^{2r+1} \to A$, and try to determine quantities $|\widetilde{Per}_n(F)|$ and $|D(\mathrm{Per}_n(F))|$.

We define the De Bruijn graph $\mathcal{D} = (V, E, t, h)$ by

$$\begin{aligned} V &:= A^{2nr}, \\ E &:= A^{2nr+1}, \\ t(x_1, \dots, x_{2nr+1}) &= (x_1, \dots, x_{2nr}), \\ h(x_1, \dots, x_{2nr+1}) &= (x_2, \dots, x_{2nr+1}), \end{aligned}$$

together with a homeomorphism

$$\Psi \colon A^{\mathbb{Z}} \to \mathrm{Path}(\mathcal{D}),\ \Psi(x) = (x_{i-nr}, \dots, x_{i+nr})_{i\in\mathbb{Z}}.$$

Next we annotate the edges of $\mathcal{D}$ by the function $p \colon E(\mathcal{D}) \to \{1, \dots, n\}$ with $p(e_1, \dots, e_{2nr+1}) = \{t \in \{1, \dots, n\} \,;\, f^t(e_1, \dots, e_{2nr+1})_{nr-tr+1} = e_{nr+1}\}$.

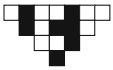


Fig. 1. Successive application of $f := w_{28}(x_{-1}, x_0, x_1) \mapsto x_{-1}(1 \oplus x_1 \oplus x_0 x_1) \oplus x_0$ (see Sec. 6 for notation) to 1011011

A direct calculation (see Fig. 1 for an illustration) shows, that $F^\ell(x) = x$ iff $\ell \in \bigcap_{i\in\mathbb{Z}} p(\psi(x)_i)$. Now we take the subgraph of $\mathcal{D}$ containing only those edges e with $n \in p(e)$ and then remove all edges not contained in any infinite path and call the result $\mathcal{G}$. By this construction $\Psi(\mathrm{Per}_n(F)) = \mathrm{Path}(\mathcal{G}) =: \mathrm{Per}_n(\mathcal{G})$ and $\Psi(\widetilde{Per}_n(F)) = \{\gamma \in \mathrm{Path}(\mathcal{G})\,;\, \bigcap_{i\in\mathbb{Z}} p(\gamma_i) = \{n\}\} =: \widetilde{Per}_n(\mathcal{G})$. See Fig. 2 for an example.

5 Computing the Invariants

In this section we show how to compute $\widetilde{Per}_n(\mathcal{G})$ and $D(\mathrm{Per}_n(\mathcal{G}))$. Let $S_\mathcal{G}$ be the set of strongly connected components of $\mathcal{G}$, that is the maximal strongly connected subgraphs of $\mathcal{G}$. Define the *strong component digraph* $\mathcal{S}_\mathcal{G}$ (see [1]) of G as the acyclic digraph with vertex set $S_\mathcal{G}$, edge set $E(\mathcal{S}_\mathcal{G}) := \{(s_1, s_2)\,;\, \exists e \in E(\mathcal{G})\colon t(e) \in s_1 \text{ and } h(e) \in s_2\}$ and tail resp. head being the first resp. second entry of the edge. For each vertex $i \in V(\mathcal{G})$ there is a unique component $s(i) \in \mathcal{S}_\mathcal{G}$ such that $i \in V(s(i))$. Each bi-infinite path $(\gamma_i)_{i\in\mathbb{Z}}$ in $\mathcal{G}$ induces a unique finite vertex-path $s(\gamma) = (s(\gamma)_1, \ldots, s(\gamma)_\ell)$ in $\mathcal{S}_\mathcal{G}$ (since $\mathcal{S}_\mathcal{G}$ is a finite acyclic digraph, it contains only finite paths) such that

$$\{s(\gamma)_1, \ldots, s(\gamma)_\ell\} = \{s(h(\gamma_i))\,;\, i \in \mathbb{Z}\}.$$

Thus $s(\gamma)$ is the path in $\mathcal{S}_\mathcal{G}$ traversed by the vertices on γ.

For components $s_1, \ldots, s_k \in S_\mathcal{G}$ we define $\mathrm{Path}(s_1, \ldots, s_k)$ as the set of all bi-infinite paths in G that traverse the components $s_1, ..., s_k$ in that order, i.e.

$$\mathrm{Path}(s_1, \ldots, s_k) = \{\gamma \in \mathrm{Path}(G)\,;\, s(\gamma) = (s_1, \ldots, s_k)\}$$

We now annotate the vertices and edges of $\mathcal{S}_\mathcal{G}$ by three functions defined as follows (remember that the vertices of $\mathcal{S}_\mathcal{G}$ are subgraphs of $\mathcal{G}$).

$$c\colon V(\mathcal{S}_\mathcal{G}) \to \mathbb{N} \cup \{\sqsupset_1\} \quad c(s) := |\mathrm{Path}(s)| = \begin{cases} |E(s)| & \text{if } s \text{ is a directed cycle} \\ & \text{or a single vertex} \\ \sqsupset_1 & \text{otherwise} \end{cases}$$

$$\rho\colon V(\mathcal{S}_\mathcal{G}) \to \{1, \ldots, n\} \qquad \rho(s) := \bigcap \{p(e)\,;\, e \in E(\mathcal{G}), t(e) \in V(s), h(e) \in V(s)\}$$

$$\mathcal{P}\colon E(\mathcal{S}_\mathcal{G}) \to 2^{\{1,\ldots,n\}} \quad \mathcal{P}(s_1, s_2) := \{p(e)\,;\, e \in E(\mathcal{G}), t(e) \in V(s_1), h(e) \in V(s_2)\}$$

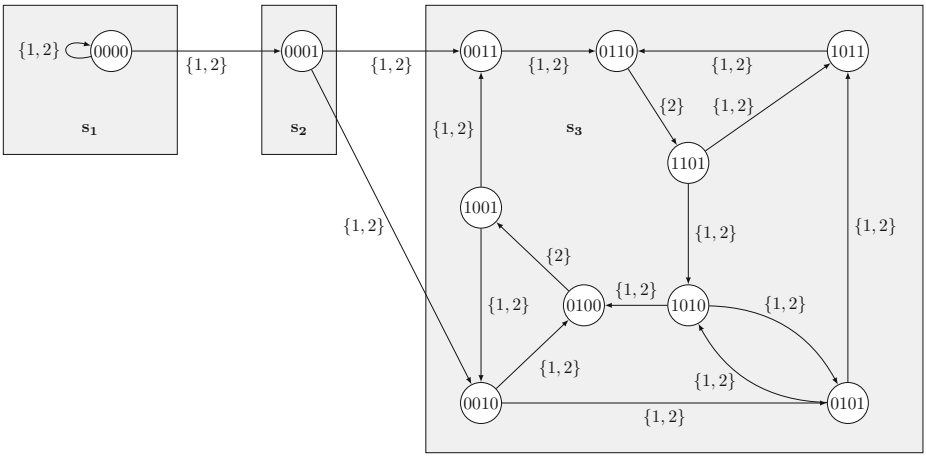


Fig. 2. The subgraph $\mathcal{G}$ of the De Bruijn graph for the CA W_{28} generated by w_{28} with $n = 2$. Its strong component digraph $\mathcal{S}_\mathcal{G}$ is a directed line with vertices s_1, s_2, s_3. The edges are labelled by p.

With these annotations, we can calculate the cardinality of $\mathrm{Path}(s_1, \ldots, s_k)$ as follows:

$$|\operatorname{Path}(s_1,\ldots,s_k)| = \begin{cases} c(s_1) & \text{if } k = 1 \\ 0 & \text{if } c(s_1) = 0 \text{ or } c(s_2) = 0 \\ \beth_1 & \text{if } c(s_1) \neq \emptyset, c(s_2) \neq \emptyset, \\ & \exists \ell \in \{1,\ldots,k\} \colon c(s_\ell) = \beth_1 \\ \beth_0 & \text{otherwise} \end{cases} . \tag{1}$$

Together with the following theorem this gives an algorithm for computing $|\widetilde{Per}_n(F)| = |\widetilde{Per}_n(\mathcal{G})|$.

Theorem 3. *Let m be the length of the longest vertex path in $\mathcal{S}_\mathcal{G}$ and let M_k be the set of all vertex paths $(s_1, \ldots, s_k)$ in $\mathcal{S}_\mathcal{G}$ with $c(s_1) \neq 0, c(s_k) \neq 0$ and $\{n\} \in \{p(s_1) \cap \cdots \cap p(s_k) \cap z_1 \cap \cdots \cap z_{k-1};\ z_1 \in P(s_1, s_2), \ldots, z_{k-1} \in P(s_{k-1}, s_k)\}$. Then $|\widetilde{Per}_n(\mathcal{G})| = \sum_{k=1}^{m} \sum_{(s_1,\ldots,s_k)\in M_k} |\operatorname{Path}(s_1, \ldots, s_k)| \in \mathcal{C}$.*

Proof. We first show that a vertex path $(s_1, \ldots, s_k) \in \mathcal{S}_\mathcal{G}$ is in M_k if and only if $\operatorname{Path}(s_1, \ldots, s_k) \cap \widetilde{Per}_n(\mathcal{G}) \neq \emptyset$.

Let $\gamma \in \operatorname{Path}(s_1, \ldots, s_k) \cap \widetilde{Per}_n(\mathcal{G})$. Let $\ell_1, \ldots, \ell_{k-1} \in \mathbb{Z}$ be the indices where γ goes from one strongly connected component to another, that is, $t(\gamma_{\ell_i}) \in V(s_i), h(\gamma_{\ell_{i+1}}) \in V(s_{i+1})$ for $i \in \{1, \ldots, k-1\}$. Then $p(\gamma_{\ell_i}) \in P(s_i, s_{i+1})$ for $i \in \{1, \ldots, k-1\}$. This implies

$$\{n\} = \bigcap_{i\in\mathbb{Z}} p(\gamma_i) \supseteq \bigcap_{j=1}^{k-1} p(\gamma_{\ell_j}) \cap p(s_1) \cap \cdots \cap p(s_k) \supseteq \{n\},$$

and thus $(s_1, \ldots, s_k) \in M_k$.

On the other hand let $(s_1, \dots, s_k) \in M_k$. There are edges $e_1, \dots, e_{k-1} \in E(\mathcal{G})$ with $t(e_i) \in V(s_i), h(e_i) \in V(s_{i+1})$ and $p(s_1) \cap \dots \cap p(s_k) \cap p(e_1) \cap \dots \cap p(e_k) = \{n\}$. Let $L \subseteq \mathrm{Path}(\mathcal{G})$ be the set of all bi-infinite paths containing all of the edges in $E(s_1) \cup \dots \cup E(s_k) \cup \{e_1, \dots, e_{k-1}\}$ and no other edges. Then $L \subseteq \mathrm{Path}(s_1, \dots, s_k)$ and for $\gamma \in L$ we have

$$\{n\} \subseteq \bigcap_{i \in \mathbb{Z}} p(\gamma_i) \subseteq p(s_1) \cap \dots \cap p(s_k) \cap p(e_1) \cap \dots \cap p(e_k) \subseteq \{n\}.$$

Hence $\gamma \in \widetilde{Per}_n(\mathcal{G})$ and $\emptyset \neq L \subseteq \mathrm{Path}(s_1, \dots, s_k) \cap \widetilde{Per}(\mathcal{G})$.

The set L contains $\beth_1$ elements iff one of the components $s_1, \dots, s_k$ is not a directed circle or a single vertex. If this is not the case and there are at least two components, then $|L| = \beth_0$. If $k = 1$ and s_1 is a directed circle or a single vertex, then $L = \mathrm{Path}(s_1, \dots, s_k)$. Therefore by (1) $|\mathrm{Path}(s_1, \dots, s_k)| = |\mathrm{Path}(s_1, \dots, s_k) \cap \widetilde{Per}_n(\mathcal{G})|$ for $(s_1, \dots, s_k) \in M_k$. The result follows with $|\widetilde{Per}_n(\mathcal{G})| = \sum_{k=1}^{m} \sum_{(s_1,\dots,s_k) \in M_k} |\mathrm{Path}(s_1, \dots, s_k) \cap \widetilde{Per}_n(\mathcal{G})|$. □

Determining the derived set of $\mathrm{Per}_n(\mathcal{G})$ is simpler. By the definition of the topology on $E(\mathcal{G})^{\mathbb{Z}}$ we have that $\mathrm{Path}(s_1, \dots, s_k) \neq \emptyset$ is either contained in $D(\mathrm{Per}_n(\mathcal{G}))$ or its complement $D(\mathrm{Per}_n(\mathcal{G}))^c$. The first case happens if and only if at least one of the following conditions is met

(i) $c(s_1) = \beth_1$ or $c(s_k) = \beth_1$,
(ii) $\exists t \in S_{\mathcal{G}}$ with $(t, s_1) \in E(\mathcal{S}_{\mathcal{G}})$ or $\exists t \in S_{\mathcal{G}}$ with $(s_k, t) \in E(\mathcal{S}_{\mathcal{G}})$.

6 Data for the 256 Elementary CA

Armed with the algorithm to compute the number of minimally p-periodic points of a CA F we can now set forth and apply this to the classification of the 256 *elementary CA*, the CA with alphabet $\{0, 1\}$ and radius 1. We enumerate them according to their Wolfram code [10], so W_k is the CA with Wolfram code k.

There remains one issue. All periodic points of F lie in its eventual image $\omega(F) := \bigcap_{t \in \mathbb{N}} F^t(A^{\mathbb{Z}})$. If two CA are conjugate when restricted to their eventual image but differ in their transient behaviour, we have no possibility to detect this up to now. As a very simple invariant capturing some transient behaviour we therefore check

(a) if F resp. F^2 is idempotent, that is, if $F^2 = F$ resp. $F^4 = F^2$,
(b) if F is an involution, that is, if $F^2 = \mathrm{id}$ and
(c) if $F^3 = F$.

We already know from Section 3, that we can always get an conjugate elementary CA by conjugation with the homeomorphisms of $\{0, 1\}^{\mathbb{Z}}$ induced by

$$\upsilon \colon \{0,1\} \to \{0,1\}, \quad \upsilon(a) = 1 - a,$$
$$\tau \colon \mathbb{Z} \to \mathbb{Z}, \quad \tau(k) = -k.$$

Each equivalence class of CA up to conjugation with these two homeomorphisms contains at most four elements (it contains less if e.g. $F = \upsilon F \upsilon^{-1}$). It is well known that 88 of these equivalence classes remain [8]. We represent each of them by the member with the smallest Wolfram code. For each equivalence class we compute the invariants and group them by this data. The results are shown in Table 1.

7 The Special Cases

We still have 10 classes of elementary cellular automata left, that we could not distinguish with the invariants considered up to now. We start with the non-trivially conjugate CA.

The following pairs of cellular automata are conjugate by

$$\vartheta\colon \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}} \quad \vartheta(x)_k := \begin{cases} 1 - x_k & \text{if } k \equiv 0 \pmod 2 \\ x_k & \text{if } k \equiv 1 \pmod 2 \end{cases}$$

(a) $(15, 170), W_{15} = \sigma \circ \upsilon$, $W_{170} = \sigma$. Notice that W_{15} and W_{170} can not be strongly conjugate since any cellular automaton commutes with σ and therefore the only other CA strongly conjugate to σ is σ^{-1}.
(b) $(77, 232)$,
(c) $(23, 178)$.

Next we have the three rules $90, 105, 150$ with

$$\begin{aligned} w_{90}(x_{-1}, x_0, x_1) &= x_{-1} \oplus x_1, \\ w_{105}(x_{-1}, x_0, x_1) &= 1 \oplus x_{-1} \oplus x_0 \oplus x_1, \\ w_{150}(x_{-1}, x_0, x_1) &= x_{-1} \oplus x_0 \oplus x_1. \end{aligned}$$

These (together with their conjugates with respect to υ) are exactly the left- and right-permutive elementary CA. Therefore by a result of Kurka and Nasu [5] they are conjugate to the one-sided full shift with alphabet $\{1, \ldots, 4\}$ and in particular they are conjugate to each other.

We will show on a case by case basis, that all CA in the remaining classes are pairwise non-conjugate. For this we use two new invariants, again only using the bijectivity of the conjugation φ. Let $\mathrm{Fix}_k(F)$ be the set of all fixed points of F with k preimages, that is,

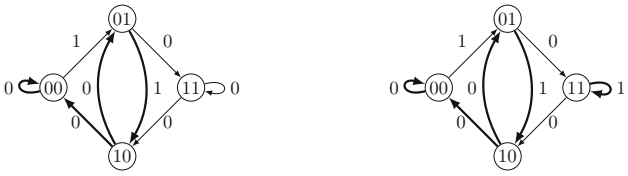
$$\mathrm{Fix}_k(F) := \{x \in \mathrm{Per}_1(F)\,;\, |F^{-1}(x)| = k\}.$$

It is straightforward to see, that $|F^{-1}(\mathrm{Per}_1(F))|$ and $|\mathrm{Fix}_k(F)|$ both remain invariant under conjugation.

For each CA F with local rule $f\colon \{0,1\}^3 \to \{0,1\}$ the De Bruijn graph for $n = 1$ with edges annotated by f is shown. A edge is drawn thickly if $f(x_{-1}x_0x_1) = x_0$, therefore the edge shift of the subgraph defined by the thick edges is $\Psi(\mathrm{Per}_1(F)) = \mathrm{Per}_1(\mathcal{G})$.

Table 1. Invariants for the 88 equivalence classes of elementary CA (using N for $\beth_0$ and C for $\beth_1$)

CA	$F^2 = id$	$F^2 = F$	$F^3 = F$	$F^4 = F^2$	$\lvert\widetilde{Per_1}(F)\rvert$	$\lvert D(\mathrm{Per}_1(F))\rvert$	$\lvert D(\mathrm{Per}_1(F)^c)\rvert$	$\lvert\widetilde{Per_2}(F)\rvert$	$\lvert D(\mathrm{Per}_2(F))\rvert$	$\lvert D(\mathrm{Per}_2(F)^c)\rvert$	$\lvert\widetilde{Per_3}(F)\rvert$	$\lvert D(\mathrm{Per}_3(F))\rvert$	$\lvert D(\mathrm{Per}_3(F)^c)\rvert$	$\lvert\widetilde{Per_4}(F)\rvert$	$\lvert D(\mathrm{Per}_4(F))\rvert$	$\lvert D(\mathrm{Per}_4(F)^c)\rvert$	$\lvert\widetilde{Per_5}(F)\rvert$	$\lvert D(\mathrm{Per}_5(F))\rvert$	$\lvert D(\mathrm{Per}_5(F)^c)\rvert$	$\lvert\widetilde{Per_6}(F)\rvert$	$\lvert D(\mathrm{Per}_6(F))\rvert$	$\lvert D(\mathrm{Per}_6(F)^c)\rvert$
0	T	T	T	T	1	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	F	F	T	T	0	0	0	C	C	0	0	0	0	0	C	0	0	0	0	0	C	0
3	F	F	F	F	0	0	0	2	0	2	3	0	3	0	0	2	5	0	5	0	0	5
5	F	F	T	T	C	C	0	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0
7	F	F	F	F	2	0	2	N	4	N	0	0	2	0	4	N	5	0	7	0	4	N
8	T	F	F	T	1	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
9	F	F	F	F	0	0	0	N	6	N	9	0	9	0	6	N	C	C	0	0	6	N
10	F	F	F	F	1	0	1	0	0	1	3	0	4	8	0	9	10	0	11	12	0	16
11	F	F	F	F	0	0	0	2	0	2	N	9	N	4	0	6	10	0	10	N	21	N
13	F	F	F	F	C	C	0	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0
14	F	F	F	F	N	3	N	0	3	N	0	3	N	4	3	N	N	18	N	N	27	N
19	F	F	F	T	0	0	0	C	C	0	0	0	0	0	C	0	0	0	0	0	C	0
22	F	F	F	F	3	0	3	4	0	7	0	0	3	C	C	7	C	C	3	C	C	31
25	F	F	F	F	0	0	0	2	0	2	3	0	3	N	14	N	0	0	0	0	0	5
26	F	F	F	F	1	0	1	6	0	7	6	0	7	24	0	31	N	25	N	126	0	139
27	F	F	F	F	0	0	0	2	0	2	6	0	6	0	0	2	5	0	5	6	0	14
28	F	F	F	F	N	3	N	C	C	0	0	3	N	0	C	0	0	3	N	0	C	0
29	F	F	T	T	2	0	2	C	C	0	0	0	2	0	C	0	0	0	2	0	C	0
30	F	F	F	F	3	0	3	0	0	3	12	0	15	28	0	31	45	0	48	84	0	99
32	F	F	F	F	1	0	1	2	0	3	0	0	1	0	0	3	0	0	1	0	0	3
33	F	F	F	F	0	0	0	C	C	2	0	0	0	0	C	2	0	0	0	0	C	2
34	F	F	F	F	1	0	1	2	0	3	3	0	4	4	0	7	10	0	11	12	0	18
35	F	F	F	F	0	0	0	N	4	N	3	0	3	0	4	N	N	10	N	0	4	N
37	F	F	F	F	3	0	3	C	C	3	0	0	3	0	C	3	0	0	3	18	C	21
38	F	F	F	F	1	0	1	0	0	1	6	0	7	8	0	9	10	0	11	18	0	25
40	F	F	F	F	1	0	1	2	0	3	3	0	4	0	0	3	5	0	6	0	0	6
41	F	F	F	F	0	0	0	N	6	N	N	12	N	0	6	N	5	0	5	30	18	N
42	F	F	F	F	1	0	1	2	0	3	6	0	7	8	0	11	20	0	21	30	0	39
43	F	F	F	F	0	0	0	N	4	N	N	12	N	4	4	N	N	20	N	N	40	N
44	F	F	F	F	C	C	0	0	C	0	3	C	3	0	C	0	0	C	0	0	C	3
45	F	F	F	F	3	0	3	2	0	5	3	0	6	8	0	13	55	0	58	0	0	8
50	F	F	F	F	1	0	1	C	C	1	0	0	1	0	C	1	0	0	1	0	C	1
51	T	F	T	T	0	0	0	C	C	0	0	0	0	0	C	0	0	0	0	0	C	0
54	F	F	F	F	1	0	1	4	0	5	0	0	1	C	C	5	0	0	1	24	0	29
56	F	F	F	F	1	0	1	2	0	3	N	6	N	4	0	7	N	10	N	N	12	N
57	F	F	F	F	0	0	0	4	0	4	N	6	N	0	0	4	N	10	N	0	6	N
58	F	F	F	F	1	0	1	2	0	3	3	0	4	0	0	3	N	10	N	0	0	6
60	F	F	F	F	1	0	1	0	0	1	3	0	4	0	0	1	15	0	16	12	0	16
62	F	F	F	F	1	0	1	0	0	1	C	C	1	0	0	1	5	0	6	0	C	1
73	F	F	F	F	C	C	0	C	C	2	C	C	0	0	C	2	C	C	0	C	C	26
74	F	F	F	F	N	4	N	0	4	N	N	7	N	N	8	N	N	9	N	N	25	N
94	F	F	F	F	C	C	1	C	C	1	C	C	1	24	C	25	0	C	1	C	C	121
104	F	F	F	F	C	C	0	6	C	6	0	C	0	0	C	6	0	C	0	0	C	6
106	F	F	F	F	1	0	1	6	0	7	12	0	13	4	0	11	10	0	11	12	0	31
108	F	F	F	T	C	C	0	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0
110	F	F	F	F	1	0	1	N	5	N	N	10	N	0	5	N	C	C	0	0	14	N
122	F	F	F	F	1	0	1	C	C	3	0	0	1	C	C	3	0	0	1	C	C	3
128	F	F	F	F	2	0	2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2
130	F	F	F	F	2	0	2	0	0	2	3	0	5	4	0	6	5	0	7	6	0	11
132	F	F	F	F	C	C	1	0	C	1	0	C	1	0	C	1	0	C	1	0	C	1
136	F	F	F	F	N	2	N	0	2	N	0	2	N	0	2	N	0	2	N	0	2	N
138	F	F	F	F	2	0	2	0	0	2	3	0	5	8	0	10	15	0	17	24	0	29
142	F	F	F	F	N	4	N	0	4	N	0	4	N	4	4	N	N	24	N	N	40	N
146	F	F	F	F	2	0	2	C	C	2	0	0	2	C	C	2	0	0	2	C	C	2
152	F	F	F	F	N	2	N	0	2	N	N	5	N	N	6	N	N	7	N	N	11	N
154	F	F	F	F	2	0	2	6	0	8	6	0	8	32	0	40	30	0	32	162	0	176
156	F	F	F	F	N	N	N	C	C	C	0	N	N	0	C	C	0	N	N	0	C	C
160	F	F	F	F	2	0	2	2	0	4	0	0	2	0	0	4	0	0	2	0	0	4
162	F	F	F	F	2	0	2	2	0	4	3	0	5	4	0	8	10	0	12	12	0	19
164	F	F	F	F	C	C	1	6	C	7	0	C	1	24	C	31	0	C	1	120	C	127
168	F	F	F	F	N	2	N	2	2	N	3	2	N	4	2	N	10	2	N	12	2	N
172	F	F	F	F	C	C	0	0	C	0	C	C	0	C	C	0	C	C	0	C	C	0
184	F	F	F	F	N	2	N	2	2	N	N	8	N	N	10	N	N	22	N	N	32	N
204	T	T	T	T	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0	0	C	0
6;134	F	F	F	F	N	3	N	0	3	N	0	3	N	N	11	N	N	23	N	N	27	N
15;170	F	F	F	F	2	0	2	2	0	4	6	0	8	12	0	16	30	0	32	54	0	64
18;126	F	F	F	F	1	0	1	C	C	1	0	0	1	C	C	1	0	0	1	C	C	1
23;178	F	F	F	F	2	0	2	C	C	2	0	0	2	0	C	2	0	0	2	0	C	2
36;72	F	F	F	T	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0	0	C	0
77;232	F	F	F	F	C	C	0	2	C	2	0	C	0	0	C	2	0	C	0	0	C	2
78;140	F	F	F	F	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0	0	C	0
2;24;46	F	F	F	F	1	0	1	0	0	1	3	0	4	4	0	5	5	0	6	6	0	10
90;105;150	F	F	F	F	4	0	4	12	0	16	60	0	64	240	0	256	1020	0	1024	4020	0	4096
4;12;76;200	F	T	T	T	C	C	0	0	C	0	0	C	0	0	C	0	0	C	0	0	C	0



(a) W_6 (b) W_{134}

Fig. 3. De Bruijn graphs for W_6 and W_{134}

Rules 6 and 134

We have that

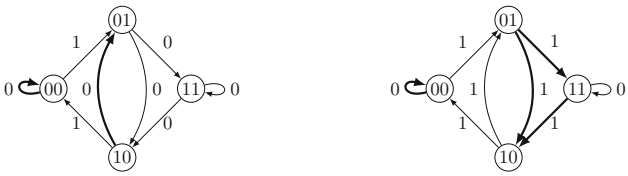
$$|W_6^{-1}(^\infty 0^\infty)| = \beth_1, \qquad |W_6^{-1}(^\infty(01)^\infty)| = 1,$$
$$|W_6^{-1}(^\infty(01).0^\infty) = \beth_1.$$

Hence $|W_6^{-1}(\mathrm{Per}_1(W_6))| = \beth_1$. On the other hand

$$|W_{134}^{-1}(^\infty 0^\infty)| = \beth_0, \qquad |W_{134}^{-1}(^\infty 1^\infty)| = 1,$$
$$|W_{134}^{-1}(^\infty(01)^\infty)| = 1, \qquad |W_{134}^{-1}(^\infty(01).0^\infty) = \beth_0,$$

and thus $|W_{134}^{-1}(\mathrm{Per}_1(W_{134}))| = \beth_0$. Therefore W_{134} and W_6 are not conjugate.

Rules 18 and 126



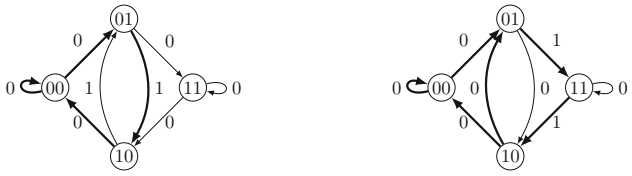
(a) W_{18} (b) W_{126}

Fig. 4. De Bruijn graphs for W_{18} and W_{126}

Both of them have only one fixed point $^\infty 0^\infty$. From the De Bruijn graphs in Fig. 4, we see that $|W_{18}^{-1}(\mathrm{Per}_1(W_{18}))| = \beth_1$ and $|W_{126}^{-1}(\mathrm{Per}_1(W_{126}))| = 2$, hence these CA are not conjugate.

Rules 36 and 72

Because of the horizontal symmetry of the annotated De Bruijn graph in Fig. 5a we see that $\mathrm{Fix}_1(W_{36}) = \emptyset$. On the other hand $^\infty(011).(011)^\infty \in \mathrm{Fix}_1(W_{72})$.

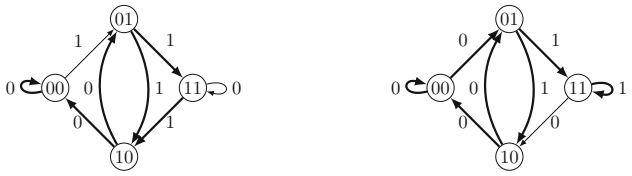


(a) W_{36} (b) W_{72}

Fig. 5. De Bruijn graphs for W_{36} and W_{72}

Rules 78 and 140

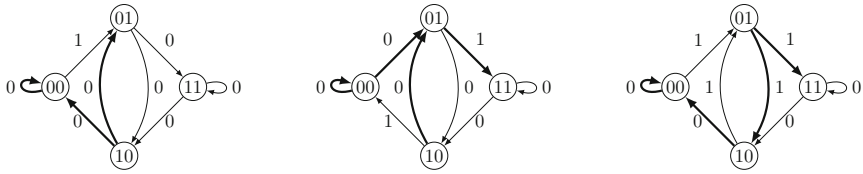
From Fig. 6 we derive that ${}^{\infty}1^{\infty} \in \mathrm{Fix}_1(W_{140})$, while $\mathrm{Fix}_1(W_{78}) = \emptyset$ since $W_{78}^{-1}({}^{\infty}0^{\infty}) = \{{}^{\infty}0^{\infty}, {}^{\infty}1^{\infty}\}$ and each occurrence of 01010 resp. 10110 might be replaced by 01110 resp. 10010 in fixed points of W_{78} without changing the image.



(a) W_{78} (b) W_{140}

Fig. 6. De Bruijn graphs for W_{78} and W_{140}

Rules 2, 24 and 46



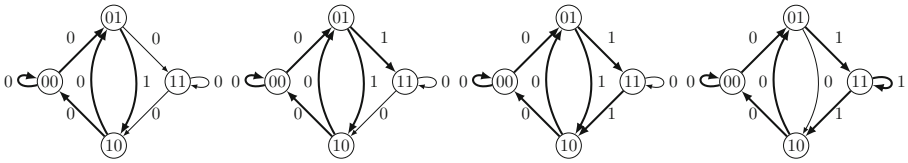
(a) W_2 (b) W_{24} (c) W_{46}

Fig. 7. De Bruijn graphs for W_2, W_{24} and W_{46}

These CA are equivalent to the shift, either σ or σ^{-1}, on their eventual image. For W_2 the eventual image is reached in one time step, that is, $W_2(\{0,1\}^{\mathbb{Z}}) = \omega(W_2)$, while the same is not the case for W_{24} and W_{46}.

Now we have a look at the sets $M_{24} := W_{24}^{-1}(\mathrm{Per}_1(W_{24}))$ and $M_{46} := W_{46}^{-1}(\mathrm{Per}_1(W_{46}))$. Both are countable SFTs. M_{24} is generated by ${}^{\infty}0.(10)^{\infty}$ and ${}^{\infty}1.(01)^{\infty}$, while M_{46} is generated by ${}^{\infty}1.0^{\infty}$. Therefore M_{24} has four accumulation points, while M_{46} has only two of them.

Rules 4, 12, 76 and 200



(a) W_4 (b) W_{12} (c) W_{76} (d) W_{200}

Fig. 8. De Bruijn graphs for W_4, W_{12}, W_{76} and W_{200}

These CA are all equal to the identity on their eventual image, or more specifically $\mathrm{Per}_1(F) = \omega(F) = F(A^{\mathbb{Z}})$ for $F \in \{W_4, W_{12}, W_{76}, W_{200}\}$. Their eventual images are all homeomorphic to the Cantor set. Notice that $\mathrm{Per}_1(W_4) = \mathrm{Per}_1(W_{12})$.

As a last invariant we look at the possible cardinalities of the preimage of a point and define $\mathrm{PF}(F) := \{|F^{-1}(x)| \; ; \; x \in A^{\mathbb{Z}}\} \subseteq \mathcal{C}$. Let Fib be the set of *Fibonacci numbers*, defined by $a_1 = 1, a_2 = 2, a_{k+2} = a_{k+1} + a_k$ for $k \in \mathbb{N}$. We will show that

$$\begin{aligned}\mathrm{PF}(W_{200}) &= \mathrm{PF}(W_{12}) \\ &= \beth_1 \cup \{b_1 b_2 \dots b_k \; ; \; k \in \mathbb{N}, b_i \in \mathrm{Fib} \text{ for } i \in \{1, \dots, k\}\}.\end{aligned}$$

In the case of W_{200} the ambiguity in forming the preimage comes from blocks of the form 110^k11, see Fig. 9b. Since isolated 1s are erased by W_{200}, the number of preimages of ${}^\infty 1.0^k 1^\infty$ equals the number of words of length $k-2$ containing no two consecutive 1s, which equals $a_{k-1} \in \mathrm{Fib}$. If more then one block of the form 110^k11 occurs, one can independently put isolated 1s in these blocks without changing the image, hence the number of the preimages is the product of those for the single blocks. The same is true for W_{76} but here we look at blocks terminated by 11 on each side and containing only isolated 1s, e.g. 11001001010001011. We can replace 010^k10 by $01^{k+2}0$ without changing the image. But since we can not do this for adjacent occurrences of 010^k10, again the number of preimages of ${}^\infty 10w01^\infty$ with w containing ℓ isolated 1s is a_ℓ.

(a) W_{76} (b) W_{200}

Fig. 9. Space-Time-Diagrams of W_{76}, W_{200} with random initial condition and periodic boundary, black represents 0 and grey represents 1

On the other hand

$$W_{12}^{-1}({}^{\infty}(01).0^{\infty}) = \{{}^{\infty}(01).1^{k}0^{\infty} \; ; \; k \in \mathbb{N}_0\} \cup \{{}^{\infty}(01).1^{\infty}\},$$

so $\beth_0 \in \mathrm{PF}(W_{12})$. But $\beth_0 \notin \mathrm{PF}(W_4)$, since any point having infinitely many preimages wrt. W_4 must contain infinitely many occurrences of blocks of the form 10^k1 with $k \geq 2$ or start resp. end in ${}^{\infty}0$ resp. 0^{∞}, thus already having uncountably many predecessors. Consequently W_{12} is not conjugate to any of W_4, W_{76} and W_{200}.

This leaves us with these three cellular automata. Next we look at $W_4^{-1}(x)$ for

$$x = {}^{\infty}(01).0\underline{0000}0(10)^{\infty}).$$

Each element of this set has to coincide with x everywhere except for the underlined block of four consecutive zeros. In this block we only have to ensure that no isolated 1s occur. So we have to determine the number of $0,1$ blocks of length 4 where ones only occur in blocks of length at least two. Therefore there can be only either zero or one block of ones, of length from 2 to 4. This gives $1 + 3 + 2 + 1 = 7$ possibilities. But 7 is not a product of Fibonacci numbers, hence W_4 is not conjugate to either W_{76} or W_{200}.

Finally we differentiate between these two CA. Notice that $\mathrm{Fix}_3(W_{200})$ consists of all configuration in $\mathrm{Per}_1(W_{200})$ containing the block 11000011 but no other block of zeros of length greater then two. Hence the closure of $\mathrm{Fix}_3(W_{200})$ is contained in $\mathrm{Fix}_3(W_{200}) \cup \mathrm{Fix}_1(W_{200})$. On the other hand we have $({}^{\infty}0.10^{\infty}) \in \mathrm{Fix}_3(W_{76})$, hence there is $(x_n)_{n\in\mathbb{N}}$ in $\mathrm{Fix}_3(W_{76})$ with $x_n \to {}^{\infty}0^{\infty} \in \mathrm{Fix}_2(W_{76})$. With that we have finally shown that W_{200} and W_{76} are not topologically conjugate.

Notice however, that $|\mathrm{Fix}_k(W_{76})| = |\mathrm{Fix}_k(W_{200})|$ for all $k \in \mathcal{C}$. Therefore W_{76} and W_{200} are conjugate when $\{0,1\}^{\mathbb{Z}}$ is endowed with the discrete topology.

8 Conclusion

We showed that there are exactly 83 equivalence classes of topologically conjugate elementary CA. Among them we saw examples of pairs of CA that are

(a) conjugate, but not strongly conjugate, e.g. $W_{170} = \sigma$ and $W_{15} = \sigma \circ \nu$,
(b) not conjugate, but conjugate if one neglects the topology, e.g. W_{200} and W_{76},
(c) not conjugate, but conjugate when restricted to their eventual image, e.g. W_4 and W_{12}.

Our main tool in differentiating non-conjugate CA was the number of minimally n-periodic points. In higher dimensions this is in general not computable, as already being able to decide if $|\mathrm{Per}_1(F)| = 0$ is equivalent to deciding the tiling problem. Therefore it would be interesting how far one can get in deciding conjugacy of higher-dimensional CA with small radius and alphabet size.

A cellular automaton is nilpotent, iff restricted to its eventual image it is conjugate to the dynamical system whose state space consists of a single point.

This implies that all nilpotent CA are conjugate when restricted to their eventual image. Nilpotency is undecidable already in dimension one [4]. Hence it is undecidable if two CA are topological conjugacy when restricted to their eventual image. But this does not immediately imply that topological conjugacy is undecidable. Therefore we finish with the following conjecture.

Conjecture 4. Topological conjugacy of one-dimensional cellular automata is undecidable.

References

1. Bang-Jensen, J., Gutin, G.Z.: Digraphs: Theory, Algorithms and Applications, 2nd edn. Springer (2008)
2. Boyle, M.: Open problems in symbolic dynamics. In: Geometric and Probabilistic Structures in Dynamics, March 15–18, 2008, University of Maryland, College Park, MD, vol. 469, p. 69. American Mathematical Soc. (2008)
3. Cattaneo, G., Formenti, E., Margara, L., Mauri, G.: Transformations of the one-dimensional cellular automata rule space. Parallel Computing **23**(11), 1593–1611 (1997)
4. Kari, J.: The Nilpotency Problem of One-Dimensional Cellular Automata. SIAM Journal on Computing **21**(3), 571–586 (1992)
5. Kůrka, P.: Topological and symbolic dynamics, vol. 11. Société Mathématique de France (2003)
6. Kůrka, P.: Topological dynamics of cellular automata. In: Encyclopedia of Complexity and Systems Science, pp. 9246–9268 (2009)
7. Lind, D., Marcus, B.: An Introduction to Symbolic Dynamics and Coding. Cambridge University Press (1995)
8. Martinez, G.J.: A Note on Elementary Cellular Automata Classification, June 2013. http://arxiv.org/abs/1306.5577
9. Ryan, J.P.: The shift and commutativity. Mathematical Systems Theory **6**(1–2), 82–85 (1972)
10. Wolfram, S.: Statistical mechanics of cellular automata. Reviews of Modern Physics **55**(3), 601–644 (1983)

Remarks on the Cellular Automaton Global Synchronisation Problem

Nazim Fatès(✉)

Inria Nancy Grand-Est, LORIA UMR 7503, 54 600 Villers-lès-Nancy, France
nazim.fates@inria.fr

Abstract. The global synchronisation problem consists in making a cellular automaton converge to a homogeneous blinking state from any initial condition. We here study this inverse problem for one-dimensional binary systems with periodic boundary conditions (i.e., rings). For small neighbourhoods, we present results obtained with the formulation of the problem as a SAT problem and the use of SAT solvers. Our observations suggest that it is not possible to solve this problem perfectly with deterministic systems. In contrast, the problem can easily be solved with stochastic rules.

Keywords: Inverse problems · SAT solving · Stochastic vs. deterministic solutions

1 Introduction

The study of inverse problems is becoming a fertile field in the research on cellular automata (CA). Among the recent achievements, we mention the construction of an exact solution to the parity problem [1], the construction of exact or approached solutions on the density classification problem [2,3,5,10] or the re-interpretation of the solution to the Firing squad problem with fields [7].

A common feature of these problems is the need to reach a global consensus: there exists a moment where all cells, or a large fraction, must agree on a given state that is interpreted as the output of the algorithm. The difficulty is related to the propagation of this information from a local to a global scale: in a decentralised framework such as cellular automata, how do the cells "agree" on a common state while they have only a local view of the system?

We here study the *global synchronisation problem*: in its original form, the question is to find a CA rule such that, from any initial condition, the system reaches a "blinking state" in which the two homogeneous configurations alternate. This problem can be generalised to more states but we will here restrict our study to the binary case.

Since its formulation by Das et al. in 1994 [4], the problem has received only a limited attention. This lack of interest is probably due to the fact that it is much easier to solve than other inverse problems such as the density classification problem. In fact, solutions with "100% success rates" were presented in the

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 113–126, 2015.
DOI: 10.1007/978-3-662-47221-7_9

very paper where the problem was formulated [4]. The authors used genetically engineered solutions to show that it was possible to obtain a performance of "100%" for ring sizes going up to 999. Their interest was to find rules which attain a consensus by removing the "defects" that separate the non-synchronised regions of the system.

Our purpose in this note is to go one step forward and to ask if perfect solutions do exist. We will thus request that *all* initial conditions lead to the blinking cycle and not only a sample of configurations, drawn at random. After presenting the formal definitions of the problem (Sec. 2), we will present a simple "manual proof" that no ECA solves the problem (Sec. 3). This construction will guide us to formulate the problem as a SAT problem (Sec. 4) and to obtain a first set of results for larger neighbourhoods (Sec. 5). We then show that, in contrast, perfect solutions can easily been constructed (Sec. 6). We conclude by formulating a few questions.

2 Fundamentals

2.1 CA Definitions

We here consider *finite* binary cellular automata with periodic boundary conditions. The basic components of our systems are the *cells*; each cell can hold one of the two states: `0` or `1`. The variable n is used to denote the number of cells that compose the system, the cells are arranged in a ring and the set of cells is denoted by $\mathcal{L} = \mathbb{Z}/n\mathbb{Z}$.

A *configuration* $x = (x_i)_{i\in\mathcal{L}}$ represents the state of the system at a given time. The set of configurations is denoted by $\mathcal{E}_n = \{\mathtt{0},\mathtt{1}\}^{\mathcal{L}}$. The interactions between the cells are local, that is, each cell can only "see" a finite subset of the cells of the system, the *neighbourhood*. Without loss of generality, we can consider that the neighbourhood $\mathcal{N}$ of our cellular automata are formed of discrete intervals: $\mathcal{N} = \{-l,\dots,r\}$, with $l \in \mathbb{N}$ and $r \in \mathbb{N}^* = \{1,2,\dots\}$. The width of this interval is called the *size* of the neighbourhood and is denoted by k, with $k = l + r + 1$.

The evolution of a cell follows a function $f : \{\mathtt{0},\mathtt{1}\}^k \to \{\mathtt{0},\mathtt{1}\}$, called the *local rule*. Following Wolfram's notation, a local rule of size k, that is, defined on a neighbourhood of size k, is assigned a code W which is an integer between 0 and $2^k - 1$. This code is given by the formula: $W = \sum_{i=0}^{2^k-1} f(\mathfrak{b}_k(i),\dots,\mathfrak{b}_1(i)) \cdot 2^i$ where $\mathfrak{b}_j(i)$ is the value of the j-th bit of the binary representation of i.

For a given ring size n, the global transition function $F : \mathcal{E}_n \to \mathcal{E}_n$ associated to the ring size n is the function that maps a configuration x^t to a configuration $x^{t+1} = F(x^t)$ such that $x^0 = x$ and:

$$\forall i \in \mathcal{L},\ x_i^{t+1} = f(x_{i-l},\dots,x_{i+r}). \tag{1}$$

Note that to be perfectly rigorous, we should denote F with indices showing that it depends on f and n. We however drop these elements for the sake of clarity since f and n will be made clear from the context.

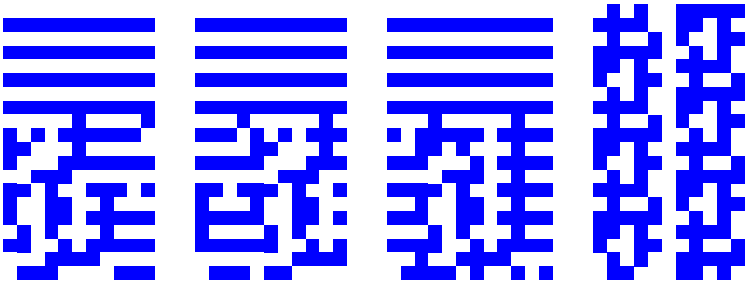


Fig. 1. Four space-time diagrams for rule 1078270911 with $k = 5$ (see page 122). Time goes from bottom to top, white and blue squares represent cells in state 0 and 1, respectively. (a), (b): synchronised initial conditions with $n = 11$, (c) and (d): initial conditions with $n = 12$, the last one is not synchronised.

2.2 Formulation of the Problem

We denote by $\mathbf{0} = 0^{\mathcal{L}}$ and $\mathbf{1} = 1^{\mathcal{L}}$ the two *uniform* configurations. In the following, we will require that 0 and 1 are two non-quiescent states, that is, $f(0, \ldots, 0) = 1$ and $f(1, \ldots, 1) = 0$. We call this condition the *blinking condition*.

We define the *height* h of a configuration $x \in \mathcal{E}_n$ as the time needed to reach one of the two uniform configurations:

$$h(x) = \min\{t \in \mathbb{N}, x^t = F^t(x) \in \{\mathbf{0}, \mathbf{1}\}\}. \tag{2}$$

with the convention that $h(x) = \infty$ if x^t does not reach $\mathbf{0}$ or $\mathbf{1}$. Similarly, the *height* of a given configuration space $\mathcal{E}_n$ is the maximum height of the configurations of $\mathcal{E}_n$.

We say that F *synchronises* a configuration $x \in \mathcal{E}_n$ if $h(x)$ is finite. We also say that x is synchronised on $\mathbf{0}$ (resp. on $\mathbf{1}$) if the first homogeneous configuration that is met is $\mathbf{0}$ (resp. $\mathbf{1}$). Similarly, we say that F synchronises the size n if it synchronises all the configurations of $\mathcal{E}_n$. We can now formulate the global synchronisation problem:

> Does there exist a local rule f such that for all $n \in \mathbb{N}^*$, the associated global function F synchronises the size n?

2.3 Elementary Properties

Let f be a local rule of size k. The *reflexion* $R(f)$ and the *conjugate* $C(f)$ are the local rules respectively obtained by the exchange of the left and right directions and by the exchange of the 0 and 1 states. Formally, $\forall (q_1, \ldots, q_k) \in \{0, 1\}^k$,

$$R(f)(q_1, \ldots, q_k) = f(q_k, \ldots, q_1),$$

and

$$C(f)(q_1, \ldots, q_k) = \overline{f(\overline{q_1}, \ldots, \overline{q_k})},$$

where $\overline{q} = 1 - q$ denotes the inversion of states. Similarly, the reflexion-conjugate rule $RC(f)$ is the local rule obtained by composing the two previous symmetries: $RC(f) = R \circ C(f) = C \circ R(f)$.

Proposition 1 (rule symmetries). *f is a solution to the global synchronisation problem if and only if $R(f), C(f), RC(f)$ are also solutions.*

Proof. Clearly, the property of synchronising a given size is preserved by the reflection and conjugation symmetries: for a given size n, if $F, F_{\mathrm{r}}, F_{\mathrm{c}}, F_{\mathrm{rc}}$ are the global functions respectively associated to $f, R(f), C(f)$ and $RC(f)$, then F synchronises the size n if and only if $F_{\mathrm{r}}, F_{\mathrm{c}}, F_{\mathrm{rc}}$ synchronise the size n. □

Let σ denote the (left) shift operator, that is, a function $\sigma : \mathcal{E}_n \to \mathcal{E}_n$ such that $\forall i \in E, \sigma(x)_i = x_{i+1}$. We call the *rotations* of x the set of configurations that are obtained by applying a positive number of shifts on x; this set is denoted by $[x] = \{\sigma^k(x), k \in \mathbb{N}\}$.

Proposition 2 (configuration symmetries). *A global rule F synchronises a configuration x if and only if it synchronises all the configurations of $[x]$.*

This simply results from the fact that: (a) F commutes with the shift and (b) $[\mathbf{0}] = \{\mathbf{0}\}$ and $[\mathbf{1}] = \{\mathbf{1}\}$. Note that the rotation $[\cdot]$ defines an equivalence class: we say that a configuration y is equivalent to x if y is a rotation of x. It can be easily verified that this is an equivalence relation.

The next proposition states that the iterates of a configuration can not be contained in the rotations of this configuration.

Proposition 3 (images of a configuration). *If F synchronises a configuration x, then $\left(\cup_{k\geq 1} F^k([x])\right) \bigcap [x] = \emptyset$, that is, $\forall x' \in [x], \forall k \in \mathbb{N}^*, F^k(x') \notin [x]$.*

Proof. By contradiction, let us assume that there exists $k \in \mathbb{N}^*$ and $i \in \mathbb{N}$ such that $F^k(x) = \sigma^i(x)$. By recurrence, using the commutation of the shift with F, we have: $F^{kn}(x) = \sigma^{in}(x)$. Since the space is a ring of size n, we have $\sigma^{in}(x) = x$, which implies $x = F^{kn}(x)$. The configuration x thus evolves on a cycle of length k. The two homogeneous states $\mathbf{0}$ and $\mathbf{1}$ are excluded from this cycle – otherwise x would be reachable from these two states – and x can not be synchronised, which contradicts the hypothesis. □

An immediate consequence of this proposition is that if F synchronises x, then x can not be a *fixed point* ($F(x) = x$) or a *blinking point* ($F(x) = \overline{x}$ and $F(\overline{x}) = x$) or a *translating point* ($F(x) = \sigma^i x$ with $i \in \mathbb{N}^*$).

Proposition 4 (color-discernation). *If a local rule f is a solution to the problem, then it is not color-blind, that is, $C(f) \neq f$.*

Proof. By contradiction, let us assume that we have $C(f) = f$. Let us take an even size $n = 2m$ with $m \in \mathbb{N}$ and consider the configuration $x = (01)^m$. Without loss of generality we can assume that x is synchronised on $\mathbf{0}$. Formally, if we denote by $T = h(x)$ the height of x, this reads: $F^T(x) = \mathbf{0}$.

Then, if we consider $\overline{x} = (\overline{x_i})_{i \in \mathcal{L}}$, we have: $F_c^T(\overline{x}) = \overline{F^T(x)} = \overline{\mathbf{0}} = \mathbf{1}$.

On the other hand: $F^T(\sigma x) = \sigma F^T(x) = \sigma \mathbf{0} = \mathbf{0}$. Since $\overline{x} = \sigma x$, we remark that these two equations are contradicting. □

3 Direct Inspection of the ECA Space

As a starting point, let us consider Elementary Cellular Automata (ECA), that is, binary CA with $\mathcal{N} = \{-1, 0, 1\}$.

Proposition 5. *There exists no ECA which solves the synchronisation problem.*

Proof. We have 256 rules to consider, these rules are defined with:

$\mathsf{a} = f(0,0,0)$ $\quad \mathsf{b} = f(0,0,1)$ $\quad \mathsf{c} = f(1,0,0)$ $\quad \mathsf{d} = f(1,0,1)$
$\mathsf{e} = f(0,1,0)$ $\quad \mathsf{f} = f(0,1,1)$ $\quad \mathsf{g} = f(1,1,0)$ $\quad \mathsf{h} = f(1,1,1)$.

We now have to determine whether there exists an assignment of these eight boolean variables $\mathsf{a}, \ldots, \mathsf{h}$ which satisfies the problem.

Case $n = 1$**:** Given the specification of the problem, the states 0 and 1 are not quiescent: $\mathsf{a} = 1$ and $\mathsf{h} = 0$. We are thus left with 64 rules to search.

Case $n = 2$**:** Proposition 3 implies $\mathsf{d} = \mathsf{e}$. Indeed, if the two variables are not equal, 01 would be either a fixed point or a blinking point. We are now left with 32 rules.

Case $n = 3$**:** By noting that the configuration 001 can not be a fixed point and can not be translated (by Prop. 3), we obtain: $s_1 = \mathsf{b} + \mathsf{c} + \mathsf{e} \neq 1$ (CondA). By symmetry, we have $s_2 = \mathsf{d} + \mathsf{f} + \mathsf{g} \neq 2$ (CondB).

The case where a configuration of [001] is transformed into a configuration of [011] and vice-versa is also impossible, otherwise the uniform configurations would never be reached (Prop. 3). We thus have $(s_1, s_2) \neq (2, 1)$ (CondC). It is then easy to verify that these conditions are sufficient to achieve the synchronisation of size 3.

We find the following set of remaining 8 rules, divided into three sets of rules: $(1, 127)$, $(9, 65, 111, 125)$, $(19, 55)$. The rules are grouped by their equivalence with the conjugation and reflexion symmetries (see Prop. 1).

Case $n = 4$**:** It is easy to check that none of the remaining rules allows a synchronisation of size 4. For instance, we remark that 0001 is a translating point for rule 1 and 9 and that 0110 is a translating point for rule 19. □

4 The Synchronisation Problem as a SAT Problem

The previous method required a methodical inspection of the space but this approach can not easily been generalised to larger neighbourhoods. We now propose to do an "automated filtering" of the rules by transforming the problem

into a SAT problem. The idea is to model the transition table as sequence of boolean variables and to express a boolean formula to state that a configuration is synchronised. We want to examine more and more initial conditions until we reach a point where we find that the problem is not satisfiable. If, on the contrary, we find that there exists a rule which synchronises all the initial conditions considered, then we would have a good "candidate" for solving the problem.

4.1 General SAT Formulation

A SAT problem consists in finding a assignment to boolean variables that satisfies a given boolean formula. A *clause* is a conjunction of literals, that is, a boolean formula defined only with the `or` operator (e.g., $a \vee b \vee c$). The convention is that a SAT problem is formulated in a conjunctive normal form (CNF): it is a disjunction of clauses. In the sequel, we call a *CNF formula* any of such disjunction of clauses.

Let k be the size of the neighbourhood and $M = 2^k$ the number of transitions[1] of this neighbourhood. We introduce M boolean variables $t_0, \ldots, t_{M-1}$ to encode the transitions of a rule f; the convention is that t_i is true if and only if $f\big(\mathfrak{b}_k(i), \ldots, \mathfrak{b}_1(i)\big) = 1$. For an initial condition $x \in \mathcal{E}_n$, we also introduce $(\tau + 1) \cdot n$ additional variables, denoted by $(\xi(t,i))_{t\in\{0,\ldots,\tau\},i\in\{0,\ldots,n-1\}}$, which correspond to the values $(x_i^t) \in \{\mathtt{0},\mathtt{1}\}$ taken by the cells in the evolution of x.

4.2 Blinking Condition

The blinking condition $f(q,\ldots,q) = \overline{q}$ is simply expressed by a CNF formula with two atomic clauses: $\mathrm{F}_{\mathrm{bl}} = t_0 \wedge \neg t_{M-1}$.

4.3 Initial State

Let us now see how to encode the states of an initial condition $x \in \mathcal{E}_n$ in a formula. The operation simply consists in "translating" the initial condition x into the CNF formula:

$$\mathrm{F}_{\mathrm{ic}}(x) = \bigwedge_{i\in\{0,\ldots,n-1\}} \mathbb{1}\{\xi(0,i), x_i^0\}. \tag{3}$$

where $\mathbb{1}\{V,q\}$ is a function which associates to the boolean variable V and to a cell state $q \in \{\mathtt{0},\mathtt{1}\}$ the variable V if $q = \mathtt{1}$ and the variable $\neg V$ otherwise.

For instance if we have $x = \mathtt{0011}$ as an initial condition, the associated formula will be:

$$\mathrm{F}_{\mathrm{ic}}(x) = \neg\xi(0,0) \wedge \neg\xi(0,1) \wedge \xi(0,2) \wedge \xi(0,3). \tag{4}$$

[1] A transition is a tuple of size k which is given as an input to the local rule.

4.4 Synchronisation Condition

Let τ be the maximum number of time steps to achieve the synchronisation. To express the condition that x is synchronised in at most τ steps, we can write $x^\tau \in \{\mathbf{0}, \mathbf{1}\}$. Unfortunately, this can not be translated in a straightforward way into a CNF formula. Indeed, if we write:

$$\mathrm{F}_{\mathrm{sc}}(x,\tau) = \left(\bigwedge_{i\in\{0,\ldots,n-1\}} \xi(\tau,i) \right) \bigvee \left(\bigwedge_{i\in\{0,\ldots,n-1\}} \neg\xi(\tau,i) \right), \tag{5}$$

we need to distribute the `and` operator over the `or`to obtain a CNF. This is why we prefer to formulate this condition as:

$$\mathrm{F}_{\mathrm{sc}}(x,\tau) = \bigwedge_{i\in\{0,\ldots,n-2\}} \xi(\tau,i) = \xi(\tau,i+1), \tag{6}$$

which simply represents the fact that all the states of x^τ are equal. By noting that $a = b$ is equivalent to $(a \vee \neg b) \wedge (\neg a \vee b)$, F_{sc} becomes:

$$\mathrm{F}_{\mathrm{sc}}(x,\tau) = \bigwedge_{i\in\{0,\ldots,n-2\}} (\xi(\tau,i) \vee \neg\xi(\tau,i+1)) \wedge (\neg\xi(\tau,i) \vee \xi(\tau,i+1)). \tag{7}$$

4.5 Consistency Conditions

We now need to write a CNF formula for the condition: $x^{t+1} = F(x^t)$ for $t \in \{0,\ldots,\tau-1\}$. We call this formulation the "*consistency condition*", as it expresses the fact a given boolean formula in consistent with the evolution of the cellular automaton. We now give a precise description of this CNF formula. In order to ease the notations, let us detail this operation for the specific case of $\mathcal{N} = \{-1, 0, 1\}$; it is easy to generalise it to other neighbourhoods. Locally, our condition is expressed by

$$\forall i \in \mathcal{L},\ x_i^{t+1} = f(x_{i-1}^t, x_i^t, x_{i+1}^t) \tag{8}$$

which is translated as:

$$\forall i \in \{0,\ldots,n-1\}, \varphi(\xi(t+1,i), \xi(t,i^-), \xi(t,i), \xi(t,i^+)) \tag{9}$$

where $i^- = (i-1) \mod n$ and $i^+ = (i+1) \mod n$, and where φ is a function that remains to be found.

Our goal is to find φ such that $\varphi(y', x, y, z)$ is a CNF formula that expresses that y' is the result of transition $f(x, y, z)$. In a usual programming environment, one would need simply to calculate $i = x + 2y + 4z$ and then to read the value of t_i and assign it to y'. Unfortunately, there is no direct way of "coding" these operations in a SAT formula. We thus need to enumerate all the possible values for the variables y', x, y, z and then write a consistency condition

that expresses that y' equals t_i where i is the index which corresponds to the transition (x, y, z).

For example, if we take transition $f(0, 1, 1) = 1$, we have $i = 3$ and we write the formula with five clauses:

$$\varphi_3 = \neg x \wedge y \wedge z \wedge t_3 \wedge (y' \vee \neg t_3) \wedge (\neg y' \vee t_3), \tag{10}$$

where the two last clauses stand for $y' = t_3$.

Formally, we write: $\varphi(y', x, y, z) = \bigvee_{\lambda \in \{0,\dots,7\}} \varphi_\lambda$ with:

$$\begin{aligned} \varphi_\lambda = {} & \mathbb{1}\{x, \lambda_1\} \wedge \mathbb{1}\{y, \lambda_2\} \wedge \mathbb{1}\{z, \lambda_3\} \wedge \\ & \mathbb{1}\{t_\lambda, f(x, y, z)\} \wedge (y' \vee \neg t_\lambda) \wedge (\neg y' \vee t_\lambda), \end{aligned} \tag{11}$$

where $\lambda_i = \mathfrak{b}_i(\lambda)$ is the value of the i-th bit of the binary representation of λ. By distributing the `and` operator over the `or`, φ_λ becomes:

$$\begin{aligned} \varphi_\lambda = {} & (\mathbb{1}\{x, \overline{\lambda_1}\} \vee \mathbb{1}\{y, \overline{\lambda_2}\} \vee \mathbb{1}\{z, \overline{\lambda_3}\} \vee y' \vee \neg t_\lambda) \bigwedge \\ & (\mathbb{1}\{x, \overline{\lambda_1}\} \vee \mathbb{1}\{y, \overline{\lambda_2}\} \vee \mathbb{1}\{z, \overline{\lambda_3}\} \vee \neg y' \vee t_\lambda). \end{aligned} \tag{12}$$

Each elementary transition of a given cell at a given time step is thus encoded with a formula φ which contains $2M = 2^{k+1} = 16$ clauses. As there are $n\tau$ such elementary conditions, the consistency condition is given by $\mathrm{F_e}$ with $2^{k+1} \cdot n\tau$ clauses:

$$\mathrm{F_e}(x, \tau) = \bigwedge_{\substack{t \in \{0,\dots,\tau-1\}, \\ i \in \{0,\dots,n\}}} \varphi\big(\xi(t+1, i), \xi(t, i^-), \xi(t, i), \xi(t, i^+)\big). \tag{13}$$

4.6 Combining Initial Conditions

The last step that remains is to combine various initial conditions in order to: (a) either find out that the problem is not solvable for a given setting or (b) exhibit a good candidate to solve the problem.

We proceed iteratively by increasing the size of the initial conditions to synchronise. From Prop. 2, we know that we do not need to consider all the initial conditions: for each size n, it is sufficient to select only *one* initial condition in each possible set of rotations. Formally, we say that a set of configurations is *representative* if the rotations of its members form a partition of the configuration space. Formally, let us denote by $\chi(n)$ the set of representative conditions; we write:

$$\chi(n) = \{X \subset \mathcal{E}_n, \forall x, y \in X, [x] \cap [y] = \emptyset, \bigcup_{x \in X} [x] = \mathcal{E}_n\}. \tag{14}$$

The following table shows the growth of these sets[2]:

n	1	2	3	4	5	6	7	8	9
$\lvert\chi(n)\rvert$	2	3	4	6	8	14	20	36	60

[2] It corresponds to sequence `A000031` in the *Online Encyclopedia of integer sequences*. One reads: "In music, $|\chi(n)|$ is the number of distinct classes of scales and chords in an n-note equal-tempered tuning system", see: https://oeis.org/A000031

To sum up, if we fix a set of ring sizes $\mathcal{S} = \{n_1, \ldots, n_s\}$, and a maximum synchronisation time τ, we construct a sequence of *sets* of configurations: $X_1, \ldots, X_s$ such that $X_i \in \chi(n_i)$ and build the *general formula*:

$$\mathrm{F}_{\mathrm{synch}}(\mathcal{S}, \tau) = \mathrm{F}_{\mathrm{bl}} \bigwedge_{i \in \{1,\ldots,s\}} \bigwedge_{x \in X_i} \mathrm{F}_{\mathrm{ic}}(x, \tau) \wedge \mathrm{F}_{\mathrm{sc}}(x, \tau) \wedge \mathrm{F}_{\mathrm{e}}(x, \tau). \qquad (15)$$

This is the final formula; it expresses the fact that all the configurations of size $n \in \mathcal{S}$ are synchronised in at most τ time steps.

5 First Experimental Results

The use of SAT solvers is a well-explored field of research in Computer science. As we are not a specialist of these questions, we did not endeavour to optimise the search for a solution by any means. We simply used the `minisat` solver[3] and generated the formulae with our cellular automata simulation program `FiatLux`[4].

5.1 ECA Space

We take $k = 3$. The results with the SAT solver confirm the results of Sec. 3: there exists no rule which is a solution for $\mathcal{S} = \{2, 3, 4\}$. For $\mathcal{S} = \{2, 3\}$, we find that:

- $(1, 127)$ have height of 1,
- $(19, 55)$ and $(9, 65, 111, 125)$ have height of 2, and
- no rule has height of 3.

We can also explore the synchronisation for other sets of ring sizes $\mathcal{S}$. For $\mathcal{S} = \{4\}$, we find that:

- $(37, 91)$ have a height of 3,
- $(25, 67, 103, 61)$ and $(45, 101, 75, 89)$ have height of 4 and,
- no rule has a height of 5.

For ECA 61, the synchronisation process is presented on Fig. 2, p. 122. Surprisingly, for $\mathcal{S} = \{5\}$, we find that $(9, 65, 111, 125)$ synchronises with a height of 5.

5.2 The $k = 4$ Space

We now examine a neighbourhood with one more cell: we take $k = 4$ and $\mathcal{N} = \{-1, 0, 1, 2\}$. The space contains $2^{2^4} = 2^{16} = 65536$ rules.

By testing increasing values of n and setting the maximum synchronisation time τ equal to $\chi(n)$ (see Prop. 3), we found that the maximum synchronisation length of this neighbourhood is 6, that is, for $\mathcal{S} = \{2, \ldots, 7\}$, no solution is found.

For $\mathcal{S} = \{2, \ldots, 6\}$, we find that:

[3] see: http://minisat.se/
[4] see http://fiatlux.loria.fr

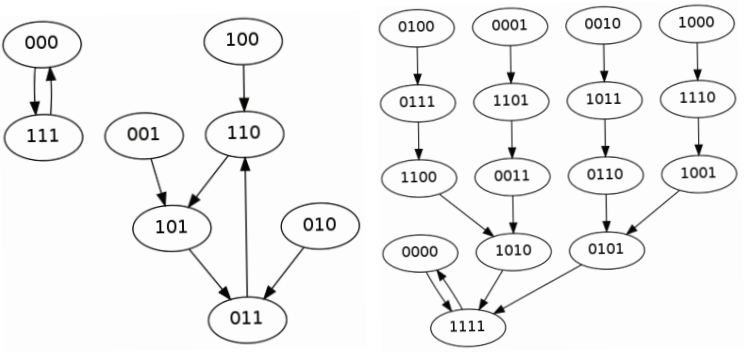


Fig. 2. Transition graphs of ECA 61 for $n = 3$ and $n = 4$. An oriented link between a configuration x and y represents the relationship $y = F(x)$.

– 6 rules have a height of 4: (1077,21471), (4427,11639), (11893,20875),
– 6 rules have a height of 5: (1205,17461,21215,21469), (5419,11095),
– 2 rules have a height of 6: (4363,12151).

For rule 5419, the synchronisation process is presented on Fig. 3, p. 122.

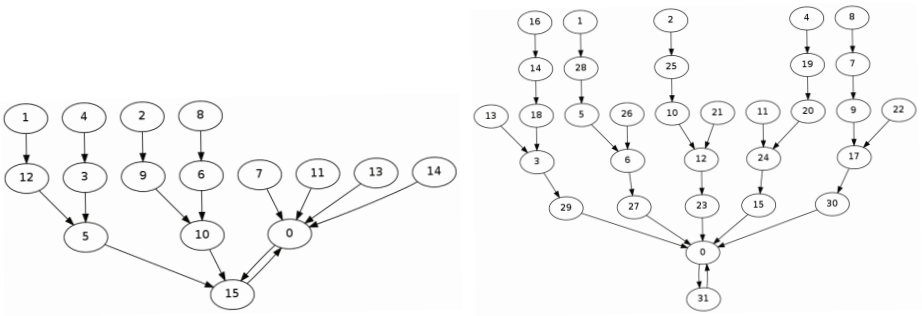


Fig. 3. Neighbourhood of size $k = 4$: transition graphs of rule 5419 for $n = 4$ and $n = 5$. For the sake of readability configurations have been represented by a number which corresponds to the decimal conversion of their bits.

5.3 The $k = 5$ Space

We now examine a neighbourhood with one more cell: we take $k = 5$ and $\mathcal{N} = \{-2, -1, 0, 1, 2\}$. The space contains $2^{2^5} = 2^{32} \sim 4.10^{10}$ rules. At this point, we reach the limits of our approach: The CNF formula of the problem has 74768 variables and 4563060 clauses. By progressively increasing the values of n and τ, our best result was to find a rule which synchronises the size interval $\mathcal{S} = \{2, \ldots, 11\}$: rule 1078270911. This rule has a height of 18 (see Fig. 1, p. 115).

There are probably other rules which solve the problem for τ higher than 18, but we leave this exploration for future work. For $\mathcal{S} = \{2, \ldots, 12\}$, no solution was found for $\tau = 30$. Surprisingly, the non-satisfiability of the formula is given rapidly, which suggests that the maximum synchronisation length for $k = 5$ is equal to 12.

We also tested the SAT solver for $k = 6$, but $\tau = 12$ is sufficient to generate SAT problems that are not solved after more than two hours of computation. We thus leave the exploration of these greater spaces for future work.

6 "Perfect" Stochastic Solutions

We now propose to examine what is the situation of the stochastic rules. In fact, if we allow randomness in the transitions of the rule, it becomes difficult *not* to solve the problem! For the sake of simplicity we restrict our study to the probabilistic ECA case. We thus take $\mathcal{N} = \{-1, 0, 1\}$ and define a local transition function $\phi : Q^3 \to [0, 1]$, which associates to each neighbourhood state its probability to be updated to $\mathtt{1}$.

Formally, starting from a configuration x, the system can be described by a *stochastic process* $(x^t)_{t\in\mathbb{N}}$. The sequence (x^t) now denotes a sequence of *random variables*, which is constructed recursively with: $x^0 = x$ (with probability 1) and

$$\forall t \in \mathbb{N},\ \forall i \in \mathcal{L},\ x_i^{t+1} = \mathcal{B}(\phi(x_{i-1}^t, x_i^t, x_{i+1}^t)), \tag{16}$$

where $\mathcal{B}(p)$ is the Bernoulli random variables, which equals $\mathtt{1}$ with probability p and $\mathtt{0}$ with probability $1 - p$. Note that strictly speaking, the definition above is more a characterisation than a definition and that a "proper" definition would require the use of tools from measure theory (see e.g. [8]).

We now need to redefine what it means to solve the problem perfectly. The blinking condition is easily translated to $\phi(\mathtt{0}, \mathtt{0}, \mathtt{0}) = 1$ and $\phi(\mathtt{1}, \mathtt{1}, \mathtt{1}) = 0$. For a rule which verifies the blinking condition and a given configuration $x \in \mathcal{E}_n$, we define the synchronisation time $T(x)$ as the random variable which corresponds to the number of steps needed to attain one of the two homogeneous configurations: $T(x) = \min\{t, x^t \in \{\mathbf{0}, \mathbf{1}\}\}$. The average synchronisation time of x is the expectancy of $T(x)$, denoted by $\mathbb{E}\{T(x)\}$. For a given size n, we define the worst expected synchronisation time (WEST) of x and the expected average synchronisation time (EAST) of x as:

$$\mathrm{WEST}(n) = \max_{x\in\mathcal{E}_n} \mathbb{E}\{T(x)\}, \tag{17}$$

$$\mathrm{EAST}(n) = \tfrac{1}{2^n} \textstyle\sum_{x\in\mathcal{E}_n} \mathbb{E}\{T(x)\}. \tag{18}$$

We say that f *synchronises* the size n if $\mathrm{WEST}(n)$ is finite. Clearly, this is equivalent as having a finite $\mathrm{EAST}(n)$. By extension, f is a solution to the global synchronisation problem if f synchronises all sizes $n \in \mathbb{N}$.

Let us now examine how to build a solution. For a function ϕ, we introduce the variables: $p_0 = \phi(\mathtt{0}, \mathtt{0}, \mathtt{0})$, $p_1 = \phi(\mathtt{0}, \mathtt{0}, \mathtt{1})$, $\ldots$, $p_7 = \phi(\mathtt{1}, \mathtt{1}, \mathtt{1})$.

Proposition 6. *Let ϕ be a probabilistic ECA such that $p_0 = 1$, $p_7 = 0$ and $\forall i \in \{1, \ldots, 6\}, p_i \in]0, 1[$, then ϕ is a solution to the global synchronisation problem.*

Proof (Sketch). To show that $T(x)$ is finite for every $x \in E$, it is sufficient to show that there is a non-zero probability to reach $\mathbf{0}$ from x in a finite number of steps. Let $Z(x)$ be the function such that $Z(x)_i = \begin{cases} 1 & \text{if } (x_{i-1}, x_i, x_{i+1}) = (0, 0, 0), \\ 0 & \text{otherwise .} \end{cases}$
The effect of Z is to change a cell to a 0 whenever there is a non-zero probability that this cells updates to 0. Let us denote by m the greatest time needed to shrink the regions of ones; clearly, $m = \lfloor n/2 \rfloor$. It can then easily be checked that:

(a) $\forall x \neq \mathbf{1}, Z^m(x) \in \{\mathbf{0}, \mathbf{1}\}$, and

(b) there is a non-zero probability to have $\forall t \in \{1, \ldots, m\}, x^t = Z^t(x)$.

In other words, there is a non-zero probability that x is absorbed in the **0**-**1** cycle in at most m time steps. Viewing the system as a Markov chain, this corresponds to the fact that the only two recurrent states are **0** and **1** and all the other states are transient[5]. □

Note that the proposition above only gives *sufficient* conditions to solve the problem; moreover, it does not give any idea on the time needed to converge to a fixed point. This means in particular that the function WEST(n) may scale exponentially with n, which is not what is expected for an efficient solution to the problem.

Proposition 7. *Consider the probabilistic ECA ϕ defined with the relationship: $\phi(x, y, z) = (\overline{y} + \overline{z})/2$; this rule is such that: $WEST(n) = \Theta(n^2)$.*

Proof (Sketch). By construction, the rule is the composition of two (commutative) operations: (a) an α-asynchronous left shift with $\alpha = 1/2$ and (b) a global inversion ($x \to \overline{x}$). In the α-asynchronous updating each cell independently applies the local rule with probability α and keeps its state with probability $1 - \alpha$. This rule was analysed in a previous work [6] and it was shown that its WECT scales quadratically with n.

It can be verified that the global inversion preserves the dynamics of the asynchronous shift. Indeed, the effect of ϕ is simply to shift the interfaces between regions of 0s and regions of 1s. To formalise this, one could use a coupling between the original process and the asynchronous shift. □

The same construction can be applied to the rules proposed by Fukś and Schüle to solve the density classification problem with stochastic rules. These rules were also analysed and were shown to have a quadratic scaling of their convergence time [5].

[5] The definitions of recurrent and transient can be found in the introductions to Markov chain theory. Informally, a recurrent state corresponds to a state who is returned to an infinite number of times with probability 1 and a transient state is a state which is not recurrent: it will then be "leaved" definitively with probability 1.

It is however interesting to note that the Traffic-Majority rule [5], whose convergence is conjectured to be linear with the ring size, *does not* obey the invariance by global inversion. Intuitively, the reason of this non-symmetry is that the Traffic rule (i.e., ECA 184) treats the 0s and 1s differently and that a global inversion also reverses the direction in which each state is "translated". It is an open question to find a rule with a synchronisation time that has a linear scaling.

7 Questions

We studied the global synchronisation problem and listed a few simple properties the potential solutions. By formulating the problem as a SAT problem, we could perform a first systematic exploration of the existence of perfect solutions and give a different point of view than the techniques that have been used so far (see e.g. Ref. [9]).

We also noted the existence of a huge gap between deterministic and stochastic systems: the use of randomness allows one to easily obtain a "perfect" solution in the sense that any configuration will be almost surely synchronised in finite time. The precise estimation of the average time of convergence is a delicate operation in all generality but insights could be given for some precise rules.

We end this note with a list of questions and indications:

Question 1. Does there a exist a deterministic rule which synchronises all sizes?

As many researchers do, we believe that the answer to this question is negative.

Question 2. If the answer is no, given a neighbourhood of size k, what is the maximum ring size that can be synchronised? Is this function computable?

We have absolutely no hint on how to answer this question.

Question 3. Given a neighbourhood of size k and maximum synchronisation time τ, what is a good algorithm to find all the rules with a height less or equal to τ? What is the complexity of this problem?

The work presented here with the use of SAT solvers can largely be improved. This is only a first attempt to use such techniques. Research could be continued by looking for other symmetries of the problem or other ways to add the constraints expressed in Prop. 3. Naturally, other paths have also to be searched.

Question 4. Is there a stochastic solution the global synchronisation problem whose WEST scales linearly with the ring size?

At the moment, we do not see how such a rule could be constructed.

Question 5. To which general context is it worth to generalise the global synchronisation problem?

One may think of higher dimensions, more states, non-homogeneous rules[6], Boolean networks, etc.

Acknowledgments. The author is grateful to Irène Marcovici, Jean Mairesse and Sukanta Das for stimulating discussions on the topic and to the anonymous reviewers for their precious remarks and suggestions.

References

1. Betel, H., de Oliveira, P.P.B., Flocchini, P.: Solving the parity problem in one-dimensional cellular automata. Natural Computing **12**(3), 323–337 (2013)
2. Briceño, R., Moisset de Espanés, P., Osses, A., Rapaport, I.: Solving the density classification problem with a large diffusion and small amplification cellular automaton. Physica D: Nonlinear Phenomena **261**, 70–80 (2013)
3. Bušić, A., Fatès, N., Mairesse, J., Marcovici, I.: Density classification on infinite lattices and trees. Electronic Journal of Probability **18**(51), 1–22 (2013). http://ejp.ejpecp.org/article/view/2325
4. Das, R., Crutchfield, J.P., Mitchell, M., Hanson, J.E.: Evolving globally synchronized cellular automata. In: Proceedings of 6th ICGA, pp. 336–343. Morgan Kaufmann, San Francisco (1995)
5. Fatès, N.: Stochastic cellular automata solutions to the density classification problem - when randomness helps computing. Theory of Computing Systems **53**(2), 223–242 (2013)
6. Fatès, N., Regnault, D., Schabanel, N., Thierry, É.: Asynchronous behavior of double-quiescent elementary cellular automata. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 455–466. Springer, Heidelberg (2006)
7. Maignan, L., Yunès, J.-B.: Experimental finitization of infinite field-based generalized FSSP solution. In: Wąs, J., Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 136–145. Springer, Heidelberg (2014)
8. Mairesse, J., Marcovici, I.: Around probabilistic cellular automata. Theoretical Computer Science **559**, 42–72 (2014)
9. Oliveira, G.M., Martins, L.G., de Carvalho, L.B., Fynn, E.: Some investigations about synchronization and density classification tasks in one-dimensional and two-dimensional cellular automata rule spaces. Electronic Notes in Theoretical Computer Science **252**, 121–142 (2009)
10. de Oliveira, P.P.B.: On density determination with cellular automata: Results, constructions and directions. Journal of Cellular Automata **9**(5–6), 357–385 (2014)

[6] Readers may test ECA 3 with two boundary cells that have a state 0.

L-Convex Polyominoes Are Recognizable in Real Time by 2D Cellular Automata

Anaël Grandjean and Victor Poupet(✉)

LIRMM, Université Montpellier 2, 161 rue Ada, 34392 Montpellier, France
{anael.grandjean,victor.poupet}@lirmm.fr

Abstract. A polyomino is said to be L-convex if any two of its cells are connected by a 4-connected inner path that changes direction at most once. The 2-dimensional language representing such polyominoes has been recently proved to be recognizable by tiling systems by S. Brocchi, A. Frosini, R. Pinzani and S. Rinaldi. In an attempt to compare recognition power of tiling systems and cellular automata, we have proved that this language can be recognized by 2-dimensional cellular automata working on the von Neumann neighborhood in real time.

Although the construction uses a characterization of L-convex polyominoes that is similar to the one used for tiling systems, the real time constraint which has no equivalent in terms of tilings requires the use of techniques that are specific to cellular automata.

Introduction

Two-dimensional cellular automata and tiling systems are two different models that can be considered to recognize classes of two-dimensional languages (or picture languages). Although they share some similarities such as locality and uniformity, the two models are fundamentally different.

Tiling systems as language recognizers were introduced by D. Giammarresi and A. Restivo in 1992 [3] and are based on the model of tile sets introduced by H. Wang [7]. The strength of the model lies in its inherent non-determinism. The system itself is a set of local rules describing valid image patterns and a picture language is recognized by the system if it is the image by a projection of the set of configurations that verify all local rules.

Cellular automata on the contrary are deterministic dynamical models. Introduced in the 1940s by S. Ulam and J. von Neumann [6] to study self replication in complex systems they were rapidly considered as computation models and language recognizers [4]. Contrary to some other classical computation models that inherently work on words, they can be considered naturally in any dimension (the original cellular automata studied by Ulam and von Neumann were 2-dimensional) and are therefore particularly well suited to picture languages. Language recognition is performed by encoding the input in an initial configuration and studying the (deterministic) evolution of the automaton from that configuration. Time and space complexities can be defined in the usual way.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 127–140, 2015.
DOI: 10.1007/978-3-662-47221-7_10

Because tiling systems lack dynamic behavior, some picture languages that can be recognized by cellular automata with minimal space and time complexity (in real time) cannot be recognized by tiling systems, such as the language of square pictures with vertical symmetry.

Conversely, the non-determinism of tiling systems should allow the recognition of languages that cannot be recognized by cellular automata in low time complexities. It is straightforward for instance to verify that the language considered in [5] as an example of language that cannot be recognized in real time by a cellular automaton working on the Moore neighborhood but can be recognized on the von Neumann neighborhood can be recognized by a tiling system, thus proving that tiling systems and real time cellular automata on the Moore neighborhood are incomparable.

Because the language of L-convex polyominoes was recently proved to be recognizable by tiling systems when it was previously though not to be, we decided to investigate its recognizability by real time von Neumann neighborhood cellular automata. Although the language was also recognized by cellular automata, the construction turned out to be quite different from the case of tiling systems and used some techniques specific to cellular automata (and possibly von Neumann neighborhood cellular automata). This article describes said construction.

1 Definitions

1.1 Cellular Automata

Definition 1 (Cellular Automaton). *A* cellular automaton *(CA) is a quadruple* $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$ *where*

- $d \in \mathbb{N}$ *is the dimension of the automaton ;*
- $\mathcal{Q}$ *is a finite set whose elements are called* states *;*
- $\mathcal{N}$ *is a finite subset of* $\mathbb{Z}^d$ *called* neighborhood *of the automaton ;*
- $\delta : \mathcal{Q}^{\mathcal{N}} \to \mathcal{Q}$ *is the* local transition function *of the automaton.*

Definition 2 (Configuration). *A* d-dimensional configuration $\mathfrak{C}$ *over the set of states* $\mathcal{Q}$ *is a mapping from* $\mathbb{Z}^d$ *to* $\mathcal{Q}$.

The elements of $\mathbb{Z}^d$ *will be referred to as* cells *and the set of all* d*-dimensional configurations over* $\mathcal{Q}$ *will be denoted as* $\mathrm{Conf}_d(\mathcal{Q})$.

Given a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, a configuration $\mathfrak{C} \in \mathrm{Conf}_d(Q)$ and a cell $c \in \mathbb{Z}^d$, we denote by $\mathcal{N}_{\mathfrak{C}}(c)$ the neighborhood of c in $\mathfrak{C}$:

$$\mathcal{N}_{\mathfrak{C}}(c) : \begin{cases} \mathcal{N} \to \mathcal{Q} \\ n \mapsto \mathfrak{C}(c+n) \end{cases}$$

From the local transition function δ of a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, we can define the *global transition function of the automaton* $\Delta : \mathrm{Conf}_d(\mathcal{Q}) \to \mathrm{Conf}_d(\mathcal{Q})$ obtained by applying the local rule on all cells :

$$\Delta(\mathfrak{C}) = \begin{cases} \mathbb{Z}^d \to \mathcal{Q} \\ c \mapsto \delta(\mathcal{N}_{\mathfrak{C}}(c)) \end{cases}$$

The action of the global transition rule makes $\mathcal{A}$ a dynamical system over the set $\mathrm{Conf}_d(\mathcal{Q})$. Because of this dynamic, in the following we will identify the CA $\mathcal{A}$ with its global rule so that $\mathcal{A}(\mathfrak{C})$ is the image of a configuration $\mathfrak{C}$ by the action of the CA $\mathcal{A}$, and more generally $\mathcal{A}^t(\mathfrak{C})$ is the configuration resulting from applying t times the global rule of the automaton from the initial configuration $\mathfrak{C}$.

Definition 3 (Von Neumann and Moore Neighborhoods). *In d dimensions, the most commonly considered neighborhoods are the von Neumann neighborhood $\mathcal{N}_{\mathrm{vN}} = \{c \in \mathbb{Z}^d,\ ||c||_1 \leq 1\}$ and the Moore neighborhood $\mathcal{N}_{\mathrm{M}} = \{c \in \mathbb{Z}^d, ||c||_\infty \leq 1\}$. Figure 1 illustrates these two neighborhoods in 2 dimensions.*

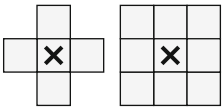


Fig. 1. The von Neumann (left) and Moore (right) neighborhoods in 2 dimensions

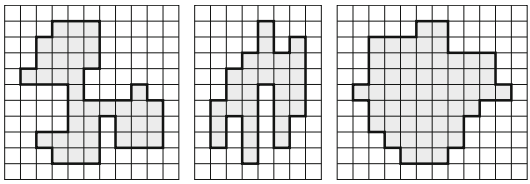


Fig. 2. Three polyominoes. The center and right ones are vertically convex, the right one is HV-convex.

1.2 Picture Recognition

From now on we will only consider 2-dimensional cellular automata (2DCA), and the set of cells will always be $\mathbb{Z}^2$.

Definition 4 (Picture). *For $n, m \in \mathbb{N}$ and Σ a finite alphabet, an (n,m)-*picture *(picture of width n and height m) over Σ is a mapping*

$$p : [\![0, n-1]\!] \times [\![0, m-1]\!] \to \Sigma$$

$\Sigma^{n,m}$ denotes the set of all (n,m)-pictures over Σ and $\Sigma^{*,*} = \bigcup_{n,m\in\mathbb{N}} \Sigma^{n,m}$ the set of all pictures over Σ. A picture language *over Σ is a set of pictures over Σ.*

Definition 5 (Picture Configuration). *Given an (n,m)-picture p over Σ, we define the* picture configuration *associated to p with quiescent state $q_0 \notin \Sigma$ as*

$$\mathfrak{C}_{p,q_0} : \begin{cases} \mathbb{Z}^2 \to \Sigma \cup \{q_0\} \\ x, y \mapsto \begin{cases} p(x,y) & \text{if } (x,y) \in [\![0, n-1]\!] \times [\![0, m-1]\!] \\ q_0 & \text{otherwise} \end{cases} \end{cases}$$

Definition 6 (Picture Recognizer). *Given a picture language L over an alphabet Σ, we say that a 2DCA $\mathcal{A} = (2, \mathcal{Q}, \mathcal{N}, \delta)$ such that $\Sigma \subseteq \mathcal{Q}$ recognizes L with quiescent state $q_0 \in \mathcal{Q} \setminus \Sigma$ and accepting states $\mathcal{Q}_a \subseteq \mathcal{Q}$ in time*

$\tau : \mathbb{N}^2 \to \mathbb{N}$ if, for any picture p (of size $n \times m$), starting from the picture configuration $\mathfrak{C}_{p,q_0}$ at time 0, the origin cell of the automaton is in an accepting state at time $\tau(n,m)$ if and only if $p \in L$. Formally,

$$\forall n, m \in \mathbb{N}, \forall p \in \Sigma^{n,m}, \quad \mathcal{A}^{\tau(n,m)}(\mathfrak{C}_{p,q_0})(0,0) \in \mathcal{Q}_a \Leftrightarrow p \in L$$

Because cellular automata work with a finite neighborhood, the state of the origin cell at time t (after t actions of the global rule) only depends on the initial states on the cells in $\mathcal{N}^t$, where $\mathcal{N}^0 = \{0\}$ and for all n, $\mathcal{N}^{n+1} = \{x + y,\ x \in \mathcal{N}^n, y \in \mathcal{N}\}$. The real time function is informally defined as the smallest time such that the state of the origin may depend on all letters of the input :

Definition 7 (Real Time). *Given a neighborhood $\mathcal{N} \subset \mathbb{Z}^d$ in d dimensions, the real time function $\tau_{\mathcal{N}} : \mathbb{N}^d \to \mathbb{N}$ associated to $\mathcal{N}$ is defined as*

$$\tau_{\mathcal{N}}(n_1, n_2, \ldots, n_d) = \min\{t, [\![0, n_1 - 1]\!] \times [\![0, n_2 - 1]\!] \times \ldots \times [\![0, n_d - 1]\!] \subseteq \mathcal{N}^t\}$$

When considering the specific case of the 2-dimensional von Neumann neighborhood, the real time is defined by $\tau_{\mathcal{N}_{\text{vN}}}(n,m) = n + m - 2$. There is however a well known constant speed-up result :

Proposition 1 (folklore). *For any $k \in \mathbb{N}$, any language that can be recognized in time $(\tau_{\mathcal{N}_{\text{vN}}} + k)$ by a 2DCA working on the von Neumann neighborhood can also be recognized in real time by a 2DCA working on the von Neumann neighborhood.*

So it will be enough to prove that a language is recognized in time $(n,m) \mapsto n + m + k$ for some constant k to prove that it is recognized in real time.

1.3 Polyominoes

Definition 8 (Polyomino). *A* placed polyomino *is a finite and 4-connected subset of $\mathbb{Z}^2$. A* polyomino *is the equivalence class of a placed polyomino up to translation.*

Definition 9 (HV-Convexity). *A polyomino p is said to be* horizontally (resp. vertically) convex *if any cell between two cells of the polyomino on a same horizontal (resp. vertical) line is also a cell of the polyomino :*

$$\forall x_1, x_2, x_3, y \in \mathbb{Z}, \quad x_1 \leq x_2 \leq x_3 \wedge (x_1, y) \in p \wedge (x_3, y) \in p \Rightarrow (x_2, y) \in p$$

A polyomino is HV-convex *if it is both horizontally and vertically convex (see Figure 2).*

We will now present the notion of L-convex polyomino, first introduced in [2] to classify HV-convex polyominoes. Informally, an L-convex polyomino p is such that for any two of its cells there exists a 4-connected path of cells of p that connects them such that the path changes direction at most once (see Figure 3).

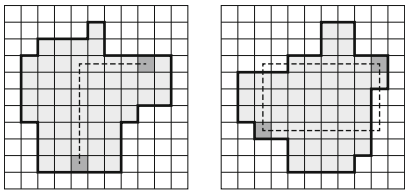


Fig. 3. The polyomino on the left is L-convex (the figure shows an inner path connecting two cells with at most one direction change, and there is such a path for any pair of cells). The polyomino on the right is HV-convex but not L-convex as illustrated by the pair of highlighted cells for which there is no inner connecting path that changes direction at most once.

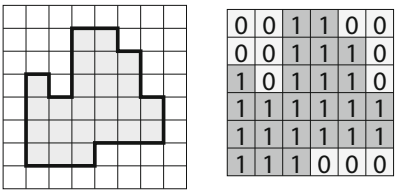


Fig. 4. A polyomino (left) and its corresponding picture over $\{0, 1\}$ (right). When this picture is encoded as a configuration of a cellular automaton, the origin of the automaton is on the lower left corner of the picture.

The following remarks will lead to a formal definition of L-convex polyominoes:

- a path that changes direction at most once connecting two cells of a polyomino p on the same row (resp. column) is fully horizontal (resp. vertical) therefore L-convex polyominoes are HV-convex ;
- if $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ are two cells in an L-convex polyomino p, either $a_1 = (x_1, y_2)$ or $a_2 = (x_2, y_1)$ is a cell of p because a_1 and a_2 are the angles of the only two paths connecting c_1 and c_2 that change direction at most once ;
- if a polyomino p is HV-convex and such that for any two of its cells $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ either $a_1 = (x_1, y_2)$ or $a_2 = (x_2, y_1)$ is a cell of p, then p is L-convex since by HV-convexity, the whole path connecting c_1 to c_2 going through a_1 or a_2 is in p.

Definition 10 (L-Convexity). *A polyomino p is L-convex if it is HV-convex and verifies*

$$\forall x_1, x_2, y_1, y_2 \in \mathbb{Z}, \quad (x_1, y_1) \in p \wedge (x_2, y_2) \in p \Rightarrow (x_1, y_2) \in p \vee (x_2, y_1) \in p$$

Given a polyomino p, the picture over the alphabet $\{0, 1\}$ associated to p is the picture whose dimensions are the dimensions of the minimal bounding rectangle of p, where the cell has state 1 if the corresponding cell is in the polyomino and 0 otherwise (see Figure 4). We define the language $L_{\text{L-convex}}$ as the picture language of all L-convex polyomino pictures.

2 Main Result

This section will be entirely devoted to the proof of the following result

Theorem 1. *The picture language $L_{\text{L-convex}}$ of L-convex polyomino pictures is recognizable in real time by a 2DCA working on the von Neumann neighborhood.*

The proof will be done by describing the behavior of a 2DCA working on the von Neumann neighborhood that recognizes $L_{\text{L-convex}}$ in real time. In this description we will use cardinal directions north, south, east and west to denote the different directions on the configuration as follows :

- north is towards the increasing y axis ;
- south is towards the decreasing y axis ;
- east is towards the increasing x axis ;
- west is towards the decreasing x axis.

With such conventions, the origin of the automaton is located at the south-west (SW) angle of the picture in the initial configuration and the picture therefore extends from the origin eastward and northward.

2.1 Preliminary Check

First of all, the automaton must check that the input is the picture of a HV-convex polyomino.

To do so, during the first step of the computation, each cell containing a 1 considers its neighbors and remembers which of them also contains a 1. Then a signal moves westward from the eastmost point of each row and southward from the northmost point on each column. These signals check that each row and each column contains exactly one segment of connected 1 symbols. Moreover, the signals check that the segment of 1 on each line and column is connected to that of the neighbor rows and columns using the neighboring information gathered during the first step.

These two properties guarantee that the polyomino is connected, HV-convex and that the picture's dimensions are that of the minimal bounding rectangle (no empty row or column). If an error is found on a row or column, the signal is directed towards the origin and the input is not accepted.

We can now assume that the input corresponds to a HV-convex polyomino picture, and must determine whether it is also L-convex.

2.2 Characterization of L-Convex Polyominoes

We will now present the characterization of L-convex polyominoes that will be used by the automaton. It is a slighly rephrased version of the characterization presented in [1] (Theorem 2).

Given a polyomino p, we say that a cell of p is a *corner* if it has two consecutive neighbors that are not in p. We classify corners depending on the directions in which such neighbors not in p are located : a north-east (NE) corner is one such that the northern and eastern neighbors are not in p, and we similarly have NW, SW and SE corners (see Figure 5 for an illustration of NE corners). Note that corner types are not exclusive : a cell can for instance be both a NE and NW corner.

Proposition 2 (Characterization of L-convex polyominoes [1]). *A HV-convex polyomino p is L-convex if and only if for every NE corner $c = (x, y)$, denote by (x, y') the southest cell of p on the same column as c, and (x', y) the westmost cell of p on the same row as c, there is no cell (x'', y'') of p verifying any of the following three conditions*

(a). $x'' > x$ *(resp.* $x'' > x'$*) and* $y'' < y'$
(b). $x'' < x'$ *(resp.* $x'' < x'$*) and* $y'' > y'$
(c). $x'' < x'$ *(resp.* $x'' < x'$*) and* $y'' < y'$

and the symmetric conditions holds for all NW corners (in the South and East directions).

Figure 5 illustrates this characterization.

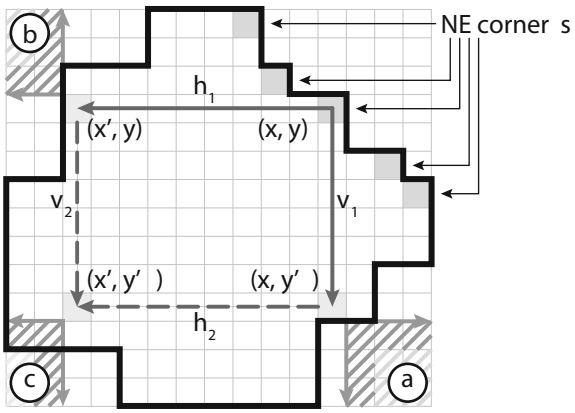


Fig. 5. A HV-convex polyomino is L-convex is for any of its NE corners (represented as dark grey cells), no cell of the polyomino lies in any of the three zones represented in hatched light grey, and symmetrically for all of its NW corners. The illustrated polyomino is not L-convex because there are two cells in the lower left hatched area (these cells cannot be connected to the represented NE corner by an inner path with at most one direction change)

Proof (sketch). It is enough to verify that all pairs of corners of a HV-convex polyomino are connected by an inner path with at most one change of direction. Moreover by symmetry we can consider only NW and NE corners.

The cells (x', y), (x, y') and (x', y') in the characterization represent the farthest points that can be reached from a given corner in their respective directions. Cells of the three restricted areas cannot be connected to the corner and conversely all cells not in these areas can be connected to the corner.

Note that because the polyomino is assumed to be HV-convex it is enough to check that there is no polyomino cell on the two lines extending from the starting

check point (represented in dark hatched grey in Figure 5). For instance, for the condition (a), it is enough to check that there is no cell $(x'', y' - 1)$ with $x'' > x$ and no cell $(x + 1, y'')$ with $y'' < y'$ in the polyomino. This follows from the 4-connectedness of the polyomino.

Although the conditions to verify are perfectly symmetric for NE and NW corners, when implementing it on a real time cellular automaton the case of NE corners is significantly simpler because all signals move towards the origin at maximum speed so the result of the verification easily arrives on time. On the other hand, for NW corners, some signals move eastward (away from the origin) so it would take too much time to send the signal all the way to the east side and back to the origin. We will therefore now focus on implementing the characterization for NE corners and come back to the NW corners at the end of the proof.

2.3 Compression and Marking

The characterization from Proposition 2 depends on cells being able to tell if there is a polyomino cell in a given direction from them. To make sure that each cell knows this information, consider signals going eastward from the west side of each row. If the initial configuration is the picture configuration of a HV-convex polyomino, there is exactly one segment of 1 symbols on each row. Before the signal meets the first 1, cells can be notified that there is no 1 westward and that there is at least one 1 eastward. On the segment of 1, cells are notified of whether they are a border cell or an inner cell and, after the segment of 1, all cells are notified that there is a 1 westward and none eastward. Of course the same thing can be done on columns with northward signals.

Now consider a horizontal compression of the input as illustrated by Figure 6. To compress the input, consider that each cell can now hold two initial states instead of one (this can be done by increasing the number of states of the automaton) and move all states westward unless the column in which they should go is full (contains two states) or is out of the boudaries of the initial configuration (ignore the darker dots from Figure 6 for the moment).

Such a compression takes $\lceil \frac{n}{2} \rceil$ time steps where n is the width of the input. During these steps, no signal can propagate westward because the initial data is already moving west at maximum speed but the time lost performing the compression can be recovered afterwards because each cell now sees twice as many states horizontally, which means that relatively to the original states, horizontal signals can perform two steps at a time.

During the compression, signals can however be propagated eastward (as illustrated by the darker dots in Figure 6). This means that while the compression is taking place, the signal indicating to each cell if it has 1 symbols east or west can propagate, so that at the end of the compression, cells have access to this information.

By performing a vertical compression after the horizontal one we can otain in half of the real time a compressed copy of the initial configuration on which every cell now has the added information of whether there is a 1 in any of the

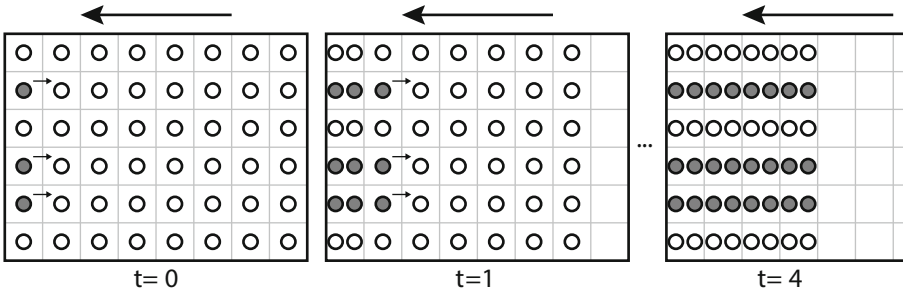


Fig. 6. Horizontal compression of the input, with eastward transmission of information (dark dots)

four directions. Moreover, later in the construction we will need to know which columns correspond to the same horizontal segment on the southern border of the polyomino, so we also propagate northward signals during the vertical compression from the borders of all horizontal segments of the southern border of the polyomino (see dashed northward arrows in Figure 8).

After both compressions, the computation of the automaton can properly start and in this computation horizontal and vertical signals can propagate twice as fast and all information is twice closer to the origin. This means that the compressed run of the automaton can behave exactly as if the configuration was not compressed but was given the extra information propagated by the eastward and northward signals from the beginning[1]. We will now ignore the compression in the following explanations, and simply consider that the information propagated by the northward and eastward signals is readily available to each cell.

Remark: As it is described, it looks as if cells should know when the compression is finished to start performing the next task (be it the second compression or the accelerated simulation of the uncompressed automaton). However, one can show that cells can *asynchronously* start the next task as soon as they have the necessary information to do so. It is sufficient to detect when all cells in their neighborhood have finished the compression to perform one step of the next task. From there, we can show that if each cell advances the following task as soon as it has enough information to do so, cells that have completed the compression early will be slowed down progressively to wait for the further cells to catch up. However, the last cells to finish the compression will never be slowed down as all other cells have the necessary information available to them. This means that by continuing the computation after the compression as soon as the information is available, all cells are at least as advanced as if all had started their computation at the time when the compression is finished, thus negating the need to synchronize all cells after the compression.

[1] This compression technique works in our case because the automaton (as it will be described later) only uses horizontal and vertical signals that change directions a bounded number of times. It is only possible to simulate two steps of the uncompressed automaton if they only involve horizontal or vertical movement, not both.

2.4 First Conditions of the Characterization

With the informations we have, checking conditions (a) and (b) of Proposition 2 is very easy as it is only a matter of sending a westward and a southward signal from each NE corner. When these signals reach the border of the polyomino, they check that there are no 1 in the corresponding area by using the information that was transmitted to each cell during the compressions. If a 1 is found where it should not be, a signal is directed towards the origin to indicate that the input should not be accepted.

There are no conflicting signals during this step because there can be at most one NE corner per column and one at most per row.

2.5 The Third Condition

The third condition from Proposition 2 is much more complex to implement. It requires sending a westward signal h_1 and a southward signal v_1 from each NE corner and having these signals generate secondary signals h_2 (westward, from the collision of v_1 with the border of the polyomino) and and v_2 (southward from the collision of h_1 and the border). The intersection of h_2 and v_2 indicate the cell on which condition (c) should be checked, as illustrated by Figure 5.

Two problems arise when implementing this behavior :

- although v_1 and h_1 signals originating from different NE corners will never overlap, if two signals arrive on the same row or column they will produce v_2 or h_2 signals that might overlap ;
- h_2 signals might intersect with many v_2 signals, but only one of them originates from the same NE corner. It is therefore necessary to ensure that the verification of condition (c) is not performed on cells that do not correspond to a valid intersection of h_2 and v_2 signals.

Priority Rule. To solve the first problem, we use a simple priority rule : if two NE corners c_1 and c_2 are north of the same horizontal segment on the southern border of the polyomino, we can ignore the easternmost one. There are two cases to consider (illustrated by Figure 7). Assume c_1 lies north-west of c_2 :

- if the westward h_1 signal from c_1 reaches the border east of that from c_2 (left of Figure 7), then the area that would be checked by considering the intersection of the signals v_2 and h_2 from c_1 (dark grey area in the Figure) is east of the one that would be considered by the intersection from c_2 (light grey area) and therefore contains it entirely, which means that it is not necessary to check the area indicated from c_2 ;
- if on the contrary the h_1 signal from c_1 arrives west of that from c_2 (right of Figure 7), the HV-convexity of the polyomino guarantees that there can be no 1 in either of the two areas considered by the intersections from c_1 and c_2 since there is at least one 1 north west of where the h_1 signal from c_2 arrives, no 1 west of that point so there cannot be any 1 west and south of it. In this case, it doesn't matter which intersection is considered since neither will find a contradiction with the (c) condition from Proposition 2.

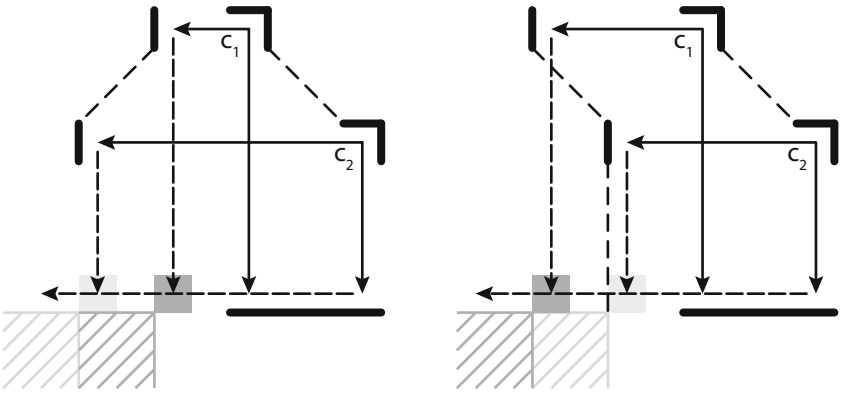


Fig. 7. When two v_1 signals arrive on the same row we can always safely ignore the one originating from the eastmost NE corner

A symmetrical argument shows that it is sufficient to consider signals originating from the southernmost of two NE corners whose h_1 signals arrive on the same vertical segment of the western border of the polyomino. Horizontal and vertical signals are however handled differently because the last part of the construction is not symmetrical.

We want to make sure that there are as many v_1 signals as there are distinct (non-overlapping) h_2 signals. To do so, v_1 signals are not sent directly by NE corners but rather sent by the h_1 signal when the h_1 signal knows that the corner it originated from is the westmost of the corresponding horizontal segment in the southern border (see Figure 8). When an h_1 signal finds a cell of the polyomino north before reaching the border of the southern segment (dashed line in the figure), it knows there is another NE corner west for that segment and therefore disappears. On the contrary, if such a signal reaches the border of the southern segment it sends the v_1 signal southward.

Counters. For v_2 signals, we need to solve the second problem that was described previously which is to determine which of the possibly many h_2 signals intersected is the one that originated from the same NE corner. To do so, h_1 signals produced by NE corners will count how many v_2 signals they cross while going west. If a h_1 signal crossed n v_2 signals, then the v_2 signal it produces will consider that its corresponding h_2 signal is the $(n+1)$-th to last one (the last n are not the one that should be considered).

Figure 9 illustrates why the result of such a behavior is correct. Consider an NE corner c_2 such that its h_1 signal crossed the v_1 signal produced by an NE corner c_1 (top circled intersection)

– if the v_1 signal produced by c_2 arrives north of the one produced by c_1 (left part of the figure) then the real intersection of the v_2 and h_2 signals from c_2

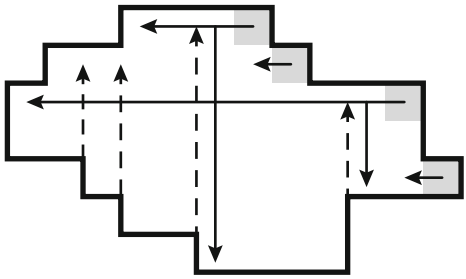


Fig. 8. h_1 signals from an NE corner are interrupted if they detect that there is another NE corner west whose v_1 signal would arrive on the same horizontal segment on the southern border of the polyomino. v_1 signals are sent by h_1 signals on the westmost column corresponding to the southern horizontal segment.

is the first that the v_2 signal from c_2 encounters, and the later one should be ignored (lower circled intersection) ;

– if on the contrary the v_1 signal from c_2 arrives south of that of c_1 (right part of the figure), the real intersection that should be considered is the last one but by considering the first the automaton will not find any contradiction to condition (c) since by HV-convexity of the polyomino there are no 1 south and west of either of the two intersections (so it will pick the wrong intersection but that will not change the final result).

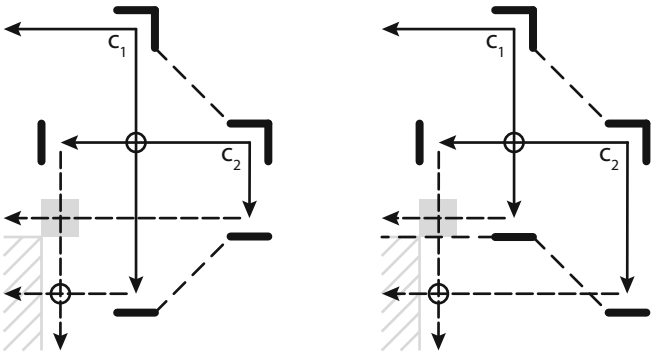


Fig. 9. Considering that for each v_2 signal crossed by the h_1 signal from c_2, one of the last intersections with an h_2 signal should be ignored by the subsequent v_2 signal leads to a correct characterization

In order to implement this rule, signals need to carry a binary counter. This counter should follow the signal at maximal speed, and will be incremented by the h_1 signal for each v_1 signal encountered (which can be easily done as incremental binary counters can be implemented on one-way one dimensional CA). For technical reasons, the counter has an initial value of 1.

As for the v_2 signal, as it crosses h_2 signals it checks if the area south-west of said intersection contains a 1 (which can be done instantly because of the information gathered during the initial compressions), and if so decrements the counter[2]. If the counter is equal to 0 (decreasing a 0 counter leaves it at 0) when the v_2 signal reaches the southernmost border of the picture, no error is detected, but if it is positive then a message is sent to the origin to indicate that the input is not L-convex.

This works because we know that if there is no 1 south-west of a cell c, there is no 1 south-west of a cell south of c. If the h_1 signal crosses n v_1 signals the counter indicates $(n+1)$ when the v_2 signal starts moving south and if the counter is at 0 it means that the $(n+1)$ last intersections were correct according to condition (c) and therefore the $(n+1)$-th to last was correct.

Checking that the counter is 0 takes $\log(n)$ steps where n is the maximal value of the counter ($\log(n)$ is the maximal length of the counter). If the counter is incremented to n it means that there are at least n NE corners north of the one from which the signal originated. This means that if the signal moves from this corner towards the origin (south or west) at maximal speed, it would reach the origin at least n steps before the real time, and therefore it can spend $\log(n)$ steps checking the value of the counter and still arrive in real time.

Moreover, conflicts of overlapping counters can be resolved by the priority rule described previously. Precedence must always be given to the counter corresponding to the southernmost NE corner :

- when an h_1 signal reaches the west border of the polyomino, it marks the cell on which the v_2 signal is produced ;
- if a v_2 signal moves through such a marked cell, it is erased ;
- if the counter following a v_2 signal is on a cell where a new v_2 signal is created, the counter is invalidated (the end symbol is erased) so that the new v_2 signal has precedence over it. An invalidated counter will ignore decrementations and will ignore the test to 0 at the end.

2.6 The North-West Corners

The previous subsections describe how NE corners can properly implement the characterization of L-convex polyominoes from Proposition 2. NW corners will behave in a very similar fashion, but special care must be taken to prove that the result of their verification can reach the origin in real time.

On a regular configuration, a signal issued from a NW corner needs to go east through most of the polyomino, then south and then the result of the verification should travel back west to the origin. In doing so the signal goes twice through the width of the input which cannot be done in real time. To solve this problem, we consider the path of the signal during a horizontal compression of the configuration:

[2] Decrementation can also be performed on a binary counter moving at maximal speed but in that case the length of the counter is not reduced when going from an 2^n to (2^n-1), but instead a leading 0 is added, which is not a problem for our construction.

- the signal starts from the NW corner on the cell $c = (x, y)$;
- during the $\frac{x}{2}$ first steps the signal moves west with the compression, and when the cell is compressed, the h_1 signal is sent eastward ;
- meanwhile, the cell (x', y) that should have been the target of the h_1 signal moves left with the compression. The h_1 signal and the cell arrive at the cell $(\frac{x'}{2}, y)$ at time $(\frac{x'}{2}, y)$;
- the signal v_2 from c moves south until it reaches the southern border of the input after y steps. At this point a delay of at most $\log(h - y)$ steps is incurred to check the value of the counter (h is the total height of the input) ;
- the result of the verification is directed towards the origin, it arrives at time $\frac{x'}{2} + y + \log(h - y) + \frac{x'}{2} < x' + h$ which is before real time.

NW corners can therefore properly perform the necessary verifications to implement the characterization from Proposition 2, which concludes the proof of Theorem 1.

References

1. Brocchi, S., Frosini, A., Pinzani, R., Rinaldi, S.: A tiling system for the class of L-convex polyominoes. Theor. Comput. Sci. **475**, 73–81 (2013)
2. Castiglione, G., Restivo, A.: Reconstruction of l-convex polyominoes. Electronic Notes in Discrete Mathematics **12**, 290–301 (2003). 9th International Workshop on Combinatorial Image Analysis
3. Giammarresi, D., Restivo, A.: Recognizable picture languages. International Journal of Pattern Recognition and Artificial Intelligence **6**(02n03), 241–256 (1992)
4. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. Journal of the ACM **6**, 233–253 (1972)
5. Terrier, V.: Two-dimensional cellular automata recognizer. Theor. Comput. Sci. **218**(2), 325–346 (1999)
6. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
7. Wang, H.: Proving theorems by pattern recognition II. Bell System Technical Journal **40**, 1–42 (1961)

Shrinking One-Way Cellular Automata

Martin Kutrib(✉), Andreas Malcher, and Matthias Wendlandt

Institut Für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
{kutrib,malcher,matthias.wendlandt}@informatik.uni-giessen.de

Abstract. We investigate cellular automata as acceptors for formal languages. In particular, we consider real-time one-way cellular automata (OCA) with the additional property that during a computation any cell of the OCA has the ability to dissolve itself and we call this model shrinking one-way cellular automata (SOCA). It turns out that real-time SOCA are more powerful than real-time OCA, since they can accept certain linear-time OCA languages. Additionally, a construction is provided that enables real-time SOCA to accept the reversal of real-time iterative array languages. Finally, restricted real-time SOCA are investigated which are allowed to dissolve only a number of cells that is bounded by some function. In this case, an infinite strict hierarchy of language classes is obtained.

1 Introduction

Devices of homogeneous, interconnected, parallel acting automata have widely been investigated from a computational capacity point of view. In particular, many results are known about *cellular automata* (see, for example, the surveys [10,11]) which are linear arrays of identical copies of deterministic finite automata, where the single nodes, which are sometimes called cells, are homogeneously connected to their both immediate neighbors. Additionally, they work synchronously at discrete time steps. The computational power of cellular automata (CA) can be measured by their ability to accept formal languages. Initially, each word is written symbolwise into the cells. Then, the transition function is synchronously applied to each cell at discrete time steps, and the input is accepted if there is a time step at which the leftmost cell enters an accepting state. The in a way simplest model of CA is that of real-time one-way cellular automata (OCA) [4]. In this model, every cell is connected with its right neighbor only which restricts the flow of information from right to left. Additionally, the available time to accept an input is restricted to the length of the input. The class of languages accepted by real-time OCA is properly included both in the class of languages accepted by real-time CA and by linear-time OCA. Furthermore, it is incomparable with the class of context-free languages. More results on real-time OCA such as, for example, closure properties and decidability questions and references to the literature may be found in [10,11].

The first notions of cellular automata date back to Ulam and von Neumann [14] whose work was biologically motivated by finding a way to design

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 141–154, 2015.
DOI: 10.1007/978-3-662-47221-7_11

a self-replicating machine. While the birth of new cells and the death of existing cells are natural processes in biology, these features are not reflected, at least to our knowledge and in connection with formal language recognition, in the existing literature. In this paper, we introduce *shrinking* one-way cellular automata (SOCA). Here, the transition function of each cell is extended so that each cell has the ability to dissolve itself. Dissolving of a cell means that the cell itself and all information stored in the cell is deleted. Moreover, the cell to the left of a dissolved cell is directly connected with the right neighbor of the dissolved cell. In this way, the number of cells available is shrinking. Nevertheless, real-time is still defined by the initial length of the input and the input is accepted whenever the leftmost cell at some computation step enters an accepting state. A related concept is that of *fault tolerant* CA (see, for example, [6,12,15,18]). These are CA where cells may become defective, that is, they cannot process the information from its neighbors any longer, but only can forward the unchanged information to its neighbors. Hence, defective cells can no longer take part in the computation, but they are still existing.

The paper is organized as follows. In Section 2 we provide a formal definition of SOCA and an illustrating example. Moreover, real-time SOCA are compared to conventional real-time OCA. To this end, a technique is developed how to embed languages. It turns out that such embeddings of languages are still acceptable by conventional OCA as long as the unembedded language is accepted in time $n + r(n)$, where r is some sublinear function. If the unembedded language is accepted in time $2n$, but not in real time, then it is shown that the embedded language is accepted by a real-time SOCA, but not by any real-time OCA. In Section 3 we investigate an interesting relation of real-time SOCA to real-time iterative arrays (IA) which are similar to real-time CA, but process their input sequentially. It is known due to Cole [3] that the language classes induced by real-time IA and real-time OCA are incomparable. Here, we obtain that real-time IA are very close to real-time SOCA. It is shown that for every language L accepted by a real-time IA, a real-time SOCA can effectively be constructed which accepts all words of even or odd length from L, or the language $\$L^R$, that is, the reversal of L headed by a new symbol $\$$. In Section 4 we study the relation between the dissolving of cells and additional time. It turns out that real-time SOCA where the number of dissolved cells in accepting computations is bounded by some function r depending on the length of the input, can be simulated by conventional OCA which work in time $n+r(n)$. This enables us to utilize results known for the time hierarchy between real-time and linear-time OCA and to obtain an infinite hierarchy with regard to the number of dissolved cells.

2 Preliminaries and Definitions

We denote the set of non-negative integers by $\mathbb{N}$. Let A denote a finite set of letters. Then we write A^* for the set of all finite words (strings) built with letters from A. The empty word is denoted by λ, the reversal of a word w by w^R, and

for the length of w we write $|w|$. For the number of occurrences of a subword x in w we use the notation $|w|_x$. A subset of A^* is called a language over A. We use $\subseteq$ for set inclusion and $\subset$ for strict set inclusion. For a set S and a symbol a we abbreviatory write S_a for $S \cup \{a\}$. We use complexity functions $f : \mathbb{N} \to \mathbb{R}^+$ tacitly assuming that the range are integers by setting $f'(n) = \lceil f(n) \rceil$ and denote f' also by f. The i-fold composition of a function f is denoted by $f^{[i]}$, $i \geq 1$. Function f is said to be *increasing* if $m < n$ implies $f(m) \leq f(n)$. The *inverse* of an increasing and unbounded function $f : \mathbb{N} \to \mathbb{N}$ is defined as $f^{-1}(n) = \min\{\, m \in \mathbb{N} \mid f(m) \geq n \,\}$. The notation $f(O(n)) \leq O(f(n))$ means $\forall\, c \geq 1 : \exists\, n_0, c' \geq 1 : \forall\, n \geq n_0 : f(c \cdot n) \leq c' \cdot f(n)$. In order to avoid technical overloading in writing, two languages L and L' are considered to be equal, if they differ at most by the empty word, that is, $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$. Throughout the article two devices are said to be *equivalent* if and only if they accept the same language.

2.1 One-Way (Shrinking) Cellular Automata

A one-way cellular automaton is a linear array of identical deterministic finite state machines, called cells. Except for the rightmost cell each one is connected to its nearest neighbor to the right. The state transition depends on the current state of a cell itself and the current state of its neighbor, where the rightmost cell receives information associated with a boundary symbol on its free input line. The state changes take place simultaneously at discrete time steps. A one-way cellular automaton is said to be shrinking if the cells may dissolve themselves. If a cell dissolves itself the next cell to its left is connected to the next cell to its right. The input mode for cellular automata is called parallel. One can suppose that all cells fetch their input symbol during a pre-initial step.

Definition 1. *A* shrinking one-way cellular automaton (*SOCA*) *is a system* $\langle S, F, A, \#, \delta \rangle$*, where* S *is the finite, nonempty set of* cell states*,* $F \subseteq S$ *is the set of* accepting states*,* $A \subseteq S$ *is the nonempty set of* input symbols*,* $\# \notin S$ *is the permanent* boundary symbol*, and* $\delta : S \times S_\# \to S \cup \{\mathsf{dissolve}\}$ *is the* local transition function*.*

A configuration c_t *of* M *at time* $t \geq 0$ *is a string of the form* $S^*\#$*, that reflects the cell states from left to right. The computation starts at time 0 in a so-called* initial configuration*, which is defined by the input* $w = a_1 a_2 \cdots a_n \in A^+$*. We set* $c_0 = a_1 a_2 \cdots a_n \#$*. Successor configurations are computed according to the global transition function* Δ*. Let* $c_t = x_1 x_2 \cdots x_n \#$*,* $t \geq 0$*, be a configuration, then its successor* $c_{t+1} = y_1 y_2 \cdots y_n \#$ *is defined as follows:*

$$c_{t+1} = \Delta(c_t) \iff \begin{cases} y_i = h(\delta(x_i, x_{i+1})), i \in \{1, 2, \ldots, n-1\} \\ y_n = h(\delta(x_n, \#)) \end{cases}$$

where $h : S \cup \{\mathsf{dissolve}\} \to S \cup \{\lambda\}$ *is the homomorphism that maps states to states and erases* dissolve*, that is,* $h(p) = p$ *for* $p \in S$*, and* $h(\mathsf{dissolve}) = \lambda$*. Thus,* Δ *is induced by* δ*.*

Fig. 1. A (shrinking) one-way cellular automaton

An SOCA is *non-shrinking* if $\delta(s_1, s_2) \neq \mathsf{dissolve}$ for all $s_1, s_2 \in S_{\#}$. Non-shrinking SOCA are referred to as one-way cellular automata. They are denoted by OCA.

An input w is accepted by an SOCA M if at some time step during the course of its computation the leftmost cell enters an accepting state, that is, the leftmost symbol of the configuration is an accepting state. The *language accepted by* M is denoted by $L(M)$. Let $t : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(M)$ are accepted with at most $t(|w|)$ time steps, then M is said to be of time complexity t.

In general, the family of languages accepted by some device X with time complexity t is denoted by $\mathscr{L}_t(X)$. The index is omitted for arbitrary time. Actually, arbitrary time is exponential time due to the space bound. If t is the identity function n, acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(X)$. The *linear-time* languages $\mathscr{L}_{lt}(X)$ are defined according to $\mathscr{L}_{lt}(X) = \bigcup_{k\in\mathbb{Q},\, k\geq 1} \mathscr{L}_{k\cdot n}(X)$.

Example 1. The language $L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\}$ is accepted by a real-time SOCA.

The basic idea to accept language L is that every cell x with $x \in \{a,b\}$ having the border cell as right neighbor sends a signal to the left that eventually dissolves a cell y with $y \in \{a,b\}$ and $y \neq x$. Subsequently, this cell dissolves itself in the following step. Meanwhile the signal is forwarded to the left by cells carrying the same input symbol x. Thus, the transition function δ of an SOCA accepting L may be sketched as follows with $x \in \{a,b\}$. We set $\delta(x, \#) = \delta(x', \#) = \overline{x}$, $\delta(\overline{x}, \#) = \mathsf{dissolve}$, and $\delta(x, \overline{x}) = \delta(x, x') = x'$. If an x-signal reaches a cell carrying an input symbol $y \in \{a,b\}$ with $y \neq x$, the cell dissolves itself as well. We set $\delta(y, \overline{x}) = \delta(y, x') = \mathsf{dissolve}$. If the input belongs to L, then all cells carrying a's and b's have been dissolved up to the next to last computation step. Thus, the only \$ cell has the border symbol as right neighbor and can enter an accepting state *acc*. If the only \$ cell sees a state x' or $\overline{x}$ to its right, it enters a rejecting state *rej*. If the input contains no symbol \$ at all, then the automaton can never enter an accepting state. If the input contains more than one \$, there is a cell having a \$-cell as right neighbor. This cell sends a signal to the left that prevents all cells passed through from dissolving and accepting.

To show that the SOCA works in real time, notice that in every second computation step the cell next to the border cell is dissolved. Furthermore, such a cell initiates a signal to the left that will dissolve another cell somewhere to the left. So, at some time step $2i$ there are i cells dissolved next to the border cell and i signals for dissolving a cell are sent. In particular, for an input word $\$w$ with $w \in L$, we have $|w|$ is even and at time $2\frac{|w|}{2}$ exactly $\frac{|w|}{2}$ cells have been dissolved next to the border cell and exactly $\frac{|w|}{2}$ cells have been dissolved somewhere to the left. So, only the \$-cell at the left border remains which accepts in the next,

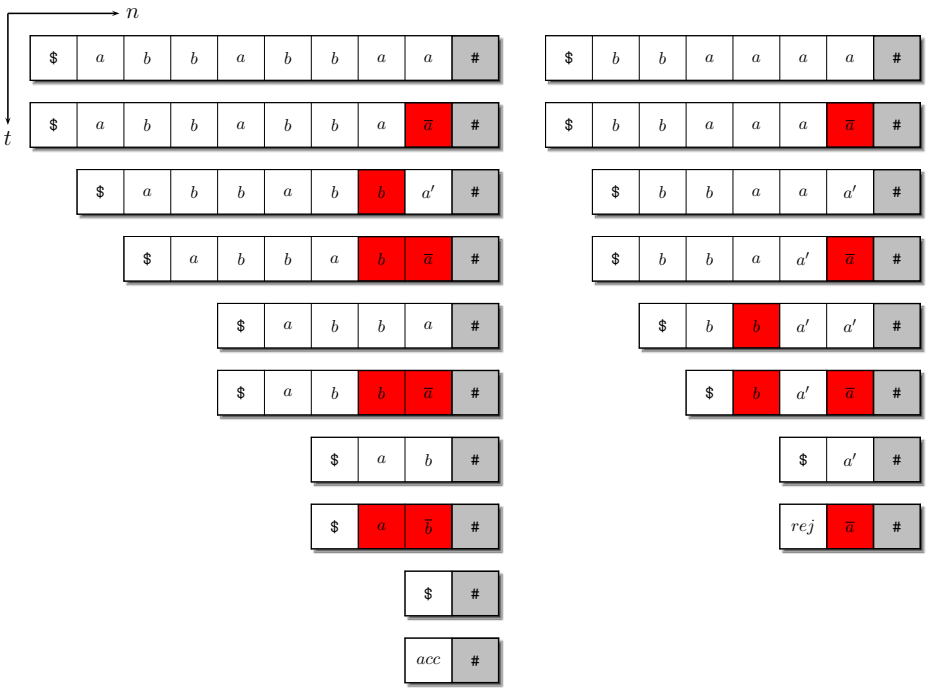


Fig. 2. The computation on the left accepts the input $\$abbabbaa$ and the computation on the right rejects the input $\$bbaaaa$. Cells to be dissolved in the next time step are depicted with background.

that is, $(|w| + 1)$st step, hence, in real time. Two example computations can be found in Figure 2. □

It is worth mentioning that it is still an open problem whether or not the deterministic context-free language L of Example 1 can be accepted by a real-time OCA.

2.2 Does Shrinking Really Help?

It is well known that the family of real-time OCA languages is a proper subfamily of the linear-time OCA languages [11]. This section is devoted to comparing the power of real-time shrinking one-way cellular automata to the power of classical real-time OCA. It turns out that the former are more powerful.

Let $L \subseteq A^*$ be a language and $\$ \notin A$ be a letter. The embedding of a $\$$ after every symbol from A is given by the homomorphism $\text{emb} : A^* \to A^*_{\$}$, where $\text{emb}(a) = a\$$ for $a \in A$. Next we show that this embedding does not help OCA. We consider time complexities from real time to linear time of the form $n + r(n)$, where $r : \mathbb{N} \to \mathbb{N}$ is a linear or sublinear function. In order to avoid functions with strange behavior, we require that r meets the weak properties increasing and

$r(O(n)) \leq O(r(n))$. Recall that the latter means $\forall\, c \geq 1 : \exists\, n_0, c' \geq 1 : \forall\, n \geq n_0 : r(c \cdot n) \leq c' \cdot r(n)$. It can easily be verified that many of the commonly considered linear and sublinear time complexity functions have this property.

Lemma 1. *Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function so that $r(O(n)) \leq O(r(n))$. A language L belongs to the family $\mathscr{L}_{n+r(n)}(OCA)$ if and only if* emb(L) *belongs to $\mathscr{L}_{n+r(n)}(OCA)$.*

Proof. Let $M = \langle S, F, A_{\$}, \#, \delta \rangle$ be an OCA accepting emb$(L) \subseteq A_{\$}^*$ in $n + r(n)$ time. We construct an OCA $M' = \langle S', F', A, \#, \delta' \rangle$ accepting $L \subseteq A^*$ as follows. The simple basic idea is to let each cell of M' simulate two adjacent cells of M, that is, a cell receiving an input from A together with its neighboring cell receiving a $\$$. In each step of M' two steps of M are simulated, which is possible since the simulated input of M appears compressed in M'. A formal construction is:

$$S' = A \cup S^2, \qquad F' = \{\, (q_1, q_2) \in S' \mid q_1 \in F \,\},$$

$$\begin{aligned}
\delta'(p_1, \#) &= (\delta(\delta(p_1, \$), \delta(\$, \#)), \delta(\delta(\$, \#), \#)) \\
&\quad \text{for all } p_1 \in A, \\
\delta'(p_1, q_1) &= (\delta(\delta(p_1, \$), \delta(\$, q_1)), \delta(\delta(\$, q_1), \delta(q_1, \$))) \\
&\quad \text{for all } p_1, q_1 \in A, \\
\delta'((p_1, p_2), (q_1, q_2)) &= (\delta(\delta(p_1, p_2), \delta(p_2, q_1)), \delta(\delta(p_2, q_1), \delta(q_1, q_2))) \\
&\quad \text{for all } (p_1, p_2), (q_1, q_2) \in S'^2,
\end{aligned}$$

in all other cases δ' does not change the current state of a cell. Let w be some word in L. So, M may use $2|w| + r(2|w|)$ time steps to accept emb(w). In order to complete the simulation on input w, the OCA M' takes at most $|w| + \frac{1}{2} r(2|w|)$ steps. Since $r(O(n)) \leq O(r(n))$, there is a constant $c' \geq 1$ so that $c' \cdot r(|w|) \geq r(2|w|)$. Therefore, M' takes at most $|w| + \frac{c'}{2} \cdot r(|w|)$ steps. Strong speed-up theorems for several devices are obtained in [7,8]. In particular, it is possible to speed up the time beyond real time linearly. Here we choose a speed-up constant $\frac{2}{c'}$ and apply this technique to M'. The resulting OCA obeys the time complexity $n + r(n)$ and accepts L.

For the converse simulation, now let M be an $n + r(n)$-time OCA accepting $L \subseteq A^*$. First, M is sped up to $n + \frac{1}{2} r(n)$ time. The further construction idea for an OCA M' that accepts emb(L) in $n + r(n)$ time is as follows.

At initial time a signal is sent from the right border to the left. It checks the correct format of the input. If the format is incorrect the left border cell is prevented from accepting. So we safely may assume the format to be correct. All cells remember whether their input symbol is a $\$$ or a symbol from A (the leftmost cell must be an A-cell). Now, in each other time step an A-cell does nothing, while a $\$$-cell copies the current state of its right neighbor as part of its own state. In the following step, a $\$$-cells does nothing, while an A-cell simulates one transition of M.

Let w be some word in L. So, M may use $|w|+\frac{1}{2}r(|w|)$ time steps to accept w. In order to complete the simulation on input $\mathrm{emb}(w)$, the OCA M' takes at most $2|w| + r(|w|)$ steps. Since $|\mathrm{emb}(w)| = 2|w|$ and r is increasing, this is less than $2|w| + r(2|w|)$ and M' obeys the time complexity $n + r(n)$. □

On the other hand, the embedding together with the possibility of cells to dissolve themselves strengthens the power of real-time OCA significantly. As mentioned before, the proper inclusion $\mathscr{L}_{rt}(\mathrm{OCA}) \subset \mathscr{L}_{lt}(\mathrm{OCA})$ is known.

Lemma 2. *Let L be a language from $\mathscr{L}_{lt}(OCA) \setminus \mathscr{L}_{rt}(OCA)$. Then* $\mathrm{emb}(L)$ *belongs to $\mathscr{L}_{rt}(SOCA)$.*

Proof. As in the proof of Lemma 1, we utilize the speed-up techniques obtained in [7,8]. So, we safely may assume that for any linear-time OCA language there exists a $(2n-2)$-time OCA M that accepts it.

Now let $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. A real-time SOCA M' accepting $\mathrm{emb}(L)$ works as follows. During the first transition every cell checks whether its right neighbor carries a correct input symbol. That is, a cell with input symbol belonging to the alphabet of L must have a `$` neighbor and vice versa (except for the rightmost cell). If the input format is incorrect a failure signal is sent to the left that prevents all cells passed through from dissolving and accepting. During the second transition all cells with `$` input dissolve themselves, while the others remain in their states. Finally, the OCA M is simulated on the remaining cells. On input of length n every other cell is dissolved during the first two time steps. Then the simulation of M takes $2\frac{n}{2} - 2$ time steps. Altogether the SOCA M' works in n time steps, that is, in real time. □

Lemma 1 and Lemma 2 have shown the next theorem.

Theorem 1. *Let L be a language from $\mathscr{L}_{lt}(OCA) \setminus \mathscr{L}_{rt}(OCA)$. Then* $\mathrm{emb}(L)$ *belongs to $\mathscr{L}_{rt}(SOCA)$ but does not belong to $\mathscr{L}_{rt}(OCA)$. In particular, the family $\mathscr{L}_{rt}(OCA)$ is properly included in $\mathscr{L}_{rt}(SOCA)$.*

3 Real-Time Shrinking OCA and Iterative Arrays

Iterative arrays (IA) are another simple model for massively parallel computations, which sometimes are called cellular automata with sequential input. In connection with formal language processing iterative arrays have been introduced in [3]. What makes these devices interesting in the current context is the incomparability of the families of languages accepted by OCA and IA in real time. Moreover, the latter devices accept non-semilinear unary languages while the former devices accept only regular unary languages. Conversely, for example, all linear context-free languages belong to $\mathscr{L}_{rt}(\mathrm{OCA})$ but there is a deterministic linear context-free language not accepted by any real-time iterative array. In the following, we investigate to what extent real-time shrinking OCA can simulate real-time IA.

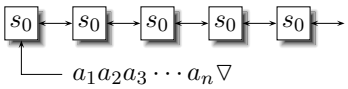


Fig. 3. An iterative array

For the formal constructions we need to introduce IA in more detail. Basically, they are semi-infinite linear arrays of finite automata, again called cells. But now the information flow is two-way, that is, each cell except the leftmost one is connected to its both nearest neighbors (one to the left and one to the right). The input is supplied sequentially to the distinguished communication cell at the origin which is connected to its immediate neighbor to the right only. For this reason, we have two different local transition functions. The state transition of all cells but the communication cell depends on the current state of the cell itself and the current states of its both neighbors. The state transition of the communication cell additionally depends on the current input symbol (or if the whole input has been consumed on a special end-of-input symbol). The finite automata work synchronously at discrete time steps. Initially they are in the so-called quiescent state.

Definition 2. *An* iterative array *(IA) is a system* $\langle S, F, A, \triangledown, s_0, \delta, \delta_0\rangle$*, where* S *is the finite, nonempty set of* cell states*,* $F \subseteq S$ *is the set of* accepting states*,* A *is the finite, nonempty set of* input symbols*,* $\triangledown \notin A$ *is the* end-of-input symbol*,* $s_0 \in S$ *is the* quiescent state*,* $\delta : S^3 \to S$ *is the* local transition function for non-communication cells *satisfying* $\delta(s_0, s_0, s_0) = s_0$*, and* $\delta_0 : A_\triangledown \times S^2 \to S$ *is the* local transition function for the communication cell.

The notions of configurations are a straightforward adaption from OCA. An input is accepted if at some time step during the course of its computation the communication cell enters an accepting state. Acceptance is said to be in real time if all w in the accepted language are accepted within at most $|w| + 1$ time steps.

The next result reveals an interesting relation between real-time IA and SOCA. The proof shows a basic construction that will later be modified for further relations.

Theorem 2. *Let* L *belong to* $\mathscr{L}_{rt}(IA)$*. Then* $\{\, w \mid w^R \in L$ *and* $|w|$ *is even* $\}$ *is accepted by a real-time SOCA.*

Proof. Given a language L accepted by some IA M in real time, the claimed real-time SOCA M' is constructed in three steps.

The first step is to embed the computation of M into a one-way device M'. To overcome the problem to simulate two-way information flow by one-way information flow, the cells of M' collect the information necessary to simulate one transition of M in an intermediate step. So, the first step of M is simulated in the second step of M' and so on. Moreover, the configuration flows with speed

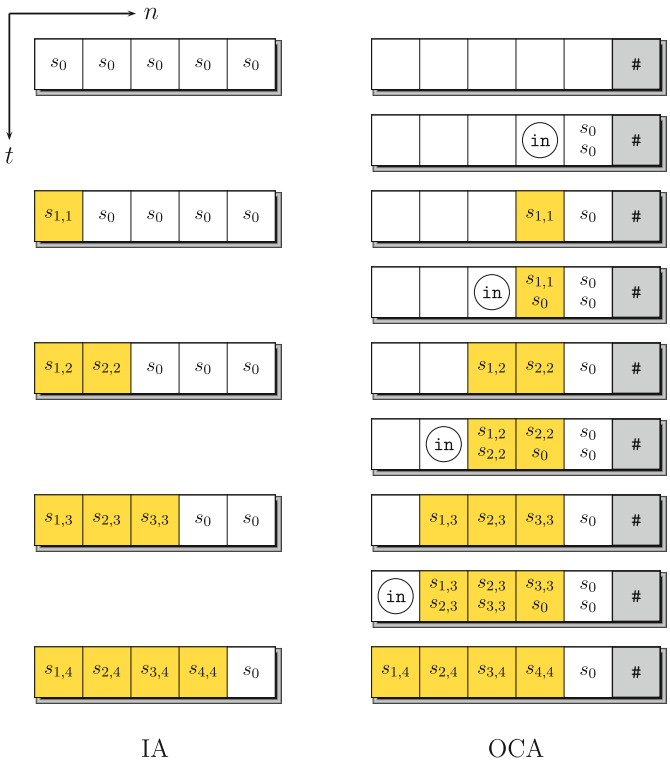


Fig. 4. Simulation of two-way information flow by one-way information flow. The input to the OCA is provided at the cells marked by the circled `in`.

1/2 to the left. The behavior is depicted in Figure 4, where for this step it is assumed that the input is available where it is consumed.

The second step is to speed up the computation of M'. Assume that in every computation step two input symbols are fed to M'. Since all cells of an IA are initially in the same, that is, quiescent state, the computation can be set up such that each cell simulates two cells of M. Hence, this together with the compressed input allows some OCA M'' to simulate two steps of M' at once at every time step.

The third step is to construct the desired real-time SOCA M'''. Based on M'' we first explain how cells dissolve themselves. This happens at every other time step beginning at time step two. More precisely, all cells maintain an internal counter modulo two. So, they can detect even time steps. At these time steps, precisely the cell which carries two input symbols and whose right neighbor simulates a state of M dissolves itself. Since the simulated configuration flows to the left with speed 1/2 and at every other time step one cell dissolves itself, the cell simulating the communication cell of M arrives at the leftmost cell at real time as desired (see Figure 5). It remains to be shown how the input is provided

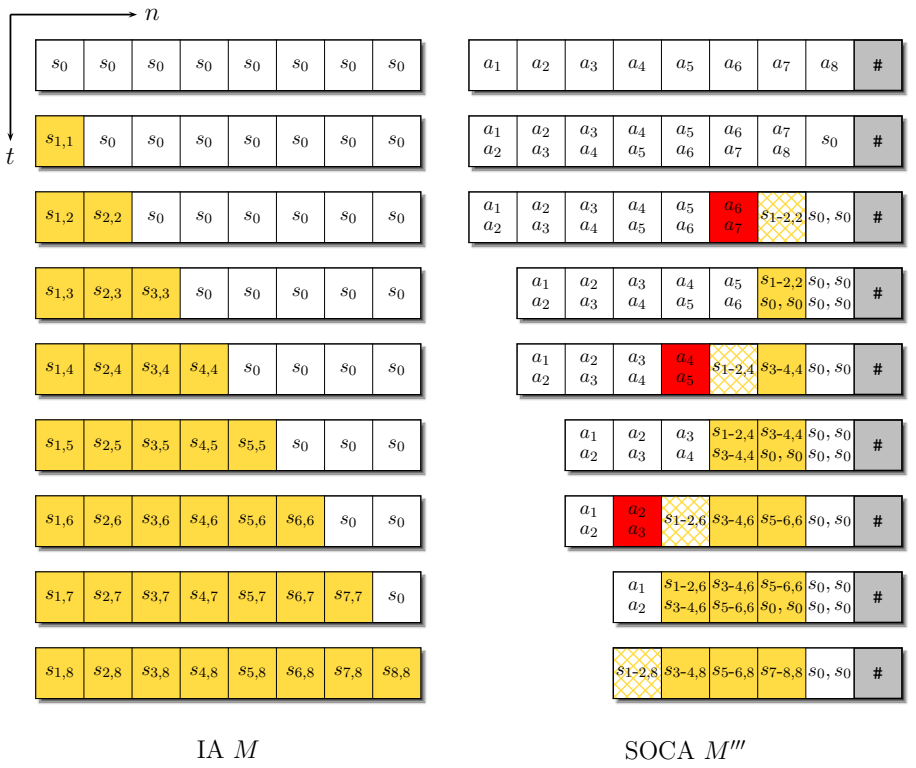


Fig. 5. Simulation of a real-time IA by a real-time SOCA on even length input. The internal modulo two counters are not depicted. The red cells dissolve themselves. The state of cell i at time k is denoted by $s_{i,k}$. We write $s_{i\text{-}j,k}$ for $s_{i,k}, s_{i+1,k}, \ldots, s_{j,k}$. The crosshatched cells are those who simulate a further transition under the assumption that they receive the last input symbol.

as requested. To this end, it is sufficient that during the first transition each cell copies the state of its right neighbor, that is its input, as part of its own state.

The SOCA M''' simulates the given IA M on even length inputs. However, the last step of M depends on the end-of-input symbol. So far, M''' does not simulate this last step of M. Therefore, the construction has slightly to be extended. Whenever, a cell of M''' storing input symbols changes to a state that simulates states of M, it simulates one more transition of M under the assumption that the cell has received the last input symbols and, thus, is the leftmost one. The result is stored in an extra register. Finally, a state is accepting if this extra register contains an accepting state of M. The transition that fills the extra register of the real leftmost cell corresponds to the missing last transition of M. □

The proof of Theorem 2 can be modified to accept inputs of odd length instead of even length in a straightforward way. Even more complicated cycles of move and dissolve operations can be realized.

Theorem 3. *Let L belong to $\mathscr{L}_{rt}(IA)$. Then $\{ w \mid w^R \in L$ and $|w|$ is odd$\}$ is accepted by a real-time SOCA.*

Example 2. Basically, a function $f : \mathbb{N} \to \mathbb{N}$ is said to be IA-constructible if there is an IA which indicates by states of the communication cell the time steps $f(n)$, for $n \geq 1$. For details and further results on IA-constructibility we refer to [1,5,13]. However, the family of such functions is rich. For example, the functions $f(n) = 2^n$ and $f(n) = p_n$, where p_n is the nth prime number, are IA-constructible. Therefore, the non-semilinear unary languages $\{ a^{2^n} \mid n \geq 0 \}$ and $\{ a^{p_n} \mid n \geq 2 \}$ belong to the family $\mathscr{L}_{rt}$(IA). By Theorems 2 and 3 they are accepted by an SOCA in real time as well. On the other hand, real-time OCA accept only regular, that is, semilinear unary languages [17]. □

A further relation between real-time IA and SOCA has been foreshadowed by Example 1. The words of the language of the example have the property that the leftmost cell can identify itself. Clearly this is realized by concatenating a fixed new symbol to the left. In general, we have the following result.

Theorem 4. *Let $L \subseteq A^*$ be a language from $\mathscr{L}_{rt}(IA)$ and $\$ \notin A$ be a letter. Then $\{ \$w \mid w^R \in L \}$ is accepted by a real-time SOCA.*

Example 3. It is shown in [16] that the real-time deterministic context-free language $L = \{ c^m a^{k_0} b a^{k_1} b \cdots b a^{k_m} b \cdots b a^{k_\ell} b d^n \mid m, n, k_i \geq 0, \ell \geq 1, k_m = n \}$ does not belong to $\mathscr{L}_{rt}$(OCA). Since the family $\mathscr{L}_{rt}$(OCA) is closed under reversal and left quotient with a fixed new symbol, the language $\$L^R$ does not belong to $\mathscr{L}_{rt}$(OCA), neither. On the other hand, all real-time deterministic context-free languages belong to $\mathscr{L}_{rt}$(IA) [11]. Now Theorem 4 shows that $\$L^R$ is accepted by some SOCA in real time. □

4 Dissolving Versus Time

In this section, we investigate to what extent the possibility of an SOCA to dissolve cells can be captured by providing additional time. We will obtain that this can always be achieved. This result enables us together with a suitable embedding of symbols to translate the time hierarchy known to exist in between real-time and linear-time conventional OCA ([9]) into a hierarchy for SOCA with regard to the number of cells dissolved. We start by defining classes of SOCA where the maximum number of dissolved cells in accepting computations is put in relation to the length of the input processed.

Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function. If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be *dissolving bounded* by f. The class of SOCA that are dissolving bounded by f is denoted by f-SOCA.

The next result says that every dissolving bounded SOCA can be simulated by a conventional OCA with additional time steps depending only on the number of dissolved cells.

Theorem 5. *Let M be an f-SOCA working in real time. Then an equivalent conventional OCA M' working in time $n + f(n)$ can effectively be constructed.*

Next, we utilize results on a time hierarchy in between real-time and linear-time known for OCA to obtain a hierarchy for SOCA according to the number of dissolved cells. The hierarchy result uses the notion of OCA-constructible functions. A function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$, is *OCA-constructible* if there exists a length-preserving homomorphism h and a language $L \in \mathscr{L}_{rt}(\text{OCA})$ such that $h(L) = \{\, a^{f(n)-n}b^n \mid n \geq 1 \,\}$. More details on OCA-constructibility may be found in [2]. Further, the language $L_d \subset \{0, 1, (,), |\}^+$ is used whose words are of the form $x\,(\,x_1 \,|\, y_1\,) \cdots (\,x_n \,|\, y_n\,)\,y$, where $x, x_i, y, y_i \in \{0, 1\}^*$ for $1 \leq i \leq n$, and $(\,x \,|\, y\,) = (\,x_i \,|\, y_i\,)$ for at least one $i \in \{1, \ldots, n\}$.

Now, the witness languages for the time hierarchy defined in [9] are as follows. Let $f : \mathbb{N} \to \mathbb{N}$ be an increasing OCA-constructible function and language $\hat{L}_f \subseteq B^+$ be a witness for the constructibility, that is, $\hat{L}_f \in \mathscr{L}_{rt}(\text{OCA})$ such that $h(\hat{L}_f) = \{\, a^{f(n)-n}b^n \mid n \geq 1 \,\}$ for a length-preserving homomorphism h. From $\hat{L}_f$ and L_d the language $L_f \subseteq ((A \cup \{\sqcup\}) \times B)^+$ is constructed as follows:

1. The second component of each word w in L_f is a word of $\hat{L}_f$ that implies that w is of length $f(m)$ for some $m \geq 1$.
2. The first component of w contains exactly $f(m) - m$ blank symbols and m non-blank symbols.
3. The non-blank symbols in the first component of w form a word in L_d.

Theorem 6. *Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions. If r_1^{-1} is OCA-constructible, $r_2(O(n)) \leq O(r_2(n))$, and $r_2 \cdot \log(r_2) \in o(r_1)$, then*

$$\mathscr{L}_{rt}(r_2\text{-}SOCA) \subset \mathscr{L}_{rt}(r_1\text{-}SOCA).$$

Proof. In [9] the proper inclusion $\mathscr{L}_{n+r_2(n)}(\text{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\text{OCA})$ has been shown with $L_{r_1^{-1}}$ as witness language. Let A' be the alphabet of $L_{r_1^{-1}}$ and M_1 be an $(n + r_1(n))$-time OCA accepting $L_{r_1^{-1}}$. By applying speed-up techniques we obtain an equivalent OCA M_1' working in time $n + r_1(n) - 2$.

Here, we will consider language $L_{\$,r_1^{-1}}$ instead, where a word w belongs to $L_{\$,r_1^{-1}}$ if and only if it is of the form $A'^*(A'\$)^*$, $|w|_\$ = r_1(|w|)$, and $h(w) \in L_{r_1^{-1}}$, for the homomorphism $h(a) = a$, for $a \in A$, and $h(\$) = \lambda$. That is, we take the words from $L_{r_1^{-1}}$ and insert exactly one $\$$ after each of the rightmost $r_1(|w|)$ symbols.

A real-time r_1-SOCA M accepting $L_{\$,r_1^{-1}}$ is constructed as follows. During the first transition, every cell with input from A' applies the homomorphism that witnesses the time constructibility of r_1 to the content of its second register and checks whether its right neighbor carries a correct input symbol. That is, if the homomorphic image is a b the cell must have a $\$$ neighbor, if it is an a the cell must not have a $\$$ neighbor. In addition, every cell with input $\$$ must have a neighbor whose homomorphic image is b or the border symbol. If the check fails, a failure signal is sent to the left that prevents all cells passed through

from dissolving and accepting. During the second transition all cells with \$ input dissolve themselves, while the others remain in their states. Finally, the $(n+r_1(n)-2)$-time OCA M_1' accepting $L_{r_1^{-1}}$ is simulated on the remaining cells.

So, the SOCA M accepts an input w if and only if it is of the form $A'^*(A'\$)^*$, the input without \$ symbols belongs to $L_{r_1^{-1}}$, that is $h(w) \in L_{r_1^{-1}}$, and the number of \$, that is the number of b, is m for input length $|w| = r_1^{-1}(m)$. The latter means $|w|_\$ = r_1(|w|)$. We conclude that M is an r_1-SOCA that accepts $L_{\$,r_1^{-1}}$.

In order to derive the time complexity of M, we recall that $r_1(n)$ cells are dissolved during the second time step on inputs of length n. Then, the simulation of M_1' on inputs of length $n - r_1(n)$ takes $n - r_1(n) + r_1(n - r_1(n)) - 2 \leq n - 2$ time steps, since r_1 is increasing. Altogether the r_1-SOCA M works in n time steps, that is, in real time.

To show a proper inclusion between language classes $\mathscr{L}_{rt}(r_2\text{-SOCA})$ and $\mathscr{L}_{rt}(r_1\text{-SOCA})$, we first notice that the first language class is included in the latter since $r_2 \cdot \log(r_2) \in o(r_1)$ holds. Now, we assume by way of contradiction that the language classes $\mathscr{L}_{rt}(r_2\text{-SOCA})$ and $\mathscr{L}_{rt}(r_1\text{-SOCA})$ are identical. Then we obtain that $L_{\$,r_1^{-1}}$ is accepted by some real-time r_2-SOCA as well. An application of Theorem 5 gives that then $L_{\$,r_1^{-1}}$ also belongs to $\mathscr{L}_{n+r_2(n)}(\text{OCA})$. So, it remains to be shown that the latter is a contradiction.

The contradiction is derived along the lines of the proof of Lemma 1. Given an $(n + r_2(n))$-time OCA N accepting $L_{\$,r_1^{-1}}$, an OCA N' accepting $L_{r_1^{-1}}$ is constructed, such that each cell receiving an input symbol whose homomorphic image is a b simulates two adjacent cells, that is, the cell itself together with its neighboring cell receiving a \$. In each such step two original steps are simulated. The rightmost a-cell starts to simulate two steps at time step 1, the next a-cell at time step 2 and so on. In this way, the leftmost cell starts to simulate two steps from time step $|w| - r_1(|w|)$ on.

Let w be some word in $L_{r_1^{-1}}$. So, N may use $|w| + r_1(|w|) + r_2(|w| + r_1(|w|))$ time steps to accept the associated padded word from $L_{\$,r_1^{-1}}$. In order to complete the simulation on input w, the OCA N' takes at most

$$|w| - r_1(|w|) + \frac{1}{2}(2r_1(|w|) + r_2(|w| + r_1(|w|))) \\ = |w| + \frac{1}{2}r_2(|w| + r_1(|w|)) \leq |w| + \frac{1}{2}r_2(2|w|)$$

time steps. Since $r_2(O(n)) \leq O(r_2(n))$, there is a constant $c' \geq 1$ so that $c' \cdot r_2(|w|) \geq r_2(2|w|)$. Therefore, N' takes at most $|w| + \frac{c'}{2} \cdot r_2(|w|)$ steps. Now, the time beyond real time is sped-up linearly by a constant $\frac{2}{c'}$. The resulting OCA obeys the time complexity $n + r_2(n)$ and accepts $L_{r_1^{-1}}$, which is a contradiction. □

Example 4. Let $0 \leq p < q \leq 1$ be two rational numbers. Then there is the following proper inclusion

$$\mathscr{L}_{rt}(n^p\text{-SOCA}) \subset \mathscr{L}_{rt}(n^q\text{-SOCA}).$$

Another example is given by the i-fold iterated logarithms $\log^{[i]}$. Let $i < j$ be two positive integers. Then there is the following proper inclusion

$$\mathscr{L}_{rt}(\log^{[j]}\text{-SOCA}) \subset \mathscr{L}_{rt}(\log^{[i]}\text{-SOCA}).$$

□

References

1. Buchholz, T., Kutrib, M.: Some relations between massively parallel arrays. Parallel Comput. **23**, 1643–1662 (1997)
2. Buchholz, T., Kutrib, M.: On time computability of functions in one-way cellular automata. Acta Inform. **35**, 329–352 (1998)
3. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans. Comput. **C–18**, 349–365 (1969)
4. Dyer, C.R.: One-way bounded cellular automata. Inform. Control **44**, 261–281 (1980)
5. Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM **12**, 388–394 (1965)
6. Harao, M., Noguchi, S.: Fault tolerant cellular automata. J. Comput. System Sci. **11**, 171–185 (1975)
7. Ibarra, O.H., Kim, S.M., Moran, S.: Sequential machine characterizations of trellis and cellular automata and applications. SIAM J. Comput. **14**, 426–447 (1985)
8. Ibarra, O.H., Palis, M.A.: Some results concerning linear iterative (systolic) arrays. J. Parallel Distributed Comput. **2**, 182–218 (1985)
9. Klein, A., Kutrib, M.: Fast one-way cellular automata. Theoret. Comput. Sci. **295**, 233–250 (2003)
10. Kutrib, M.: Cellular automata - a computational point of view. In: Bel-Enguix, G., Jiménez-López, M.D., Martin-Vide, C. (eds.) New Developments in Formal Languages and Applications. SCI, vol. 113, pp. 183–227. Springer, Heidelberg (2008)
11. Kutrib, M.: Cellular automata and language theory. In: Encyclopedia of Complexity and System Science, pp. 800–823. Springer (2009)
12. Kutrib, M., Löwe, J.T.: Space- and time-bounded nondeterminism for cellular automata. Fund. Inform. **58**, 273–293 (2003)
13. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comput. Sci. **217**, 53–80 (1999)
14. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press (1966), edited and completed by A.W. Burks
15. Nishio, H., Kobuchi, Y.: Fault tolerant cellular spaces. J. Comput. System Sci. **11**, 150–170 (1975)
16. Okhotin, A.: Comparing linear conjunctive languages to subfamilies of the context-free languages. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 431–443. Springer, Heidelberg (2011)
17. Seidel, S.R.: Language recognition and the synchronization of cellular automata. Tech. Rep. 79–02, Department of Computer Science, University of Iowa (1979)
18. Umeo, H.: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. IEICE Trans. Inf. Syst. **E87–D**, 733–739 (2004)

Universal Time-Symmetric Number-Conserving Cellular Automaton

Diego Maldonado[1(✉)], Andrés Moreira[2], and Anahí Gajardo[1]

[1] Departamento de Ingeniería Matemática and Centro de Investigación en Ingeniería Matemática, Universidad de Concepción, Casilla 160-C, Concepción, Chile
diegomaldonado@udec.cl

[2] Departamento de Informática, Universidad Técnica Federico Santa María, Vicuña Mackenna, 3939 Santiago, Chile

Abstract. We show the existence of Turing-universal and intrinsically universal cellular automata exhibiting both time symmetry and number conservation; this is achieved by providing a way to simulate reversible CA with time-symmetric CA, which preserves the number-conserving property. We also provide some additional results and observations concerning the simulation relations between reversible, time-symmetric and number-conserving CA in the context of partitioned CA.

1 Introduction

Reversibility is an important property in the realm of cellular automata, both because of its obvious theoretical consequences and because of the most practical applications for massive distributed computing models where heat dissipation would be unwanted. It has therefore been well studied (see reviews in [5,9]), and some of this effort has been directed towards showing computational capabilities (by showing the existence of universal reversible CA, under different notions of universality) and towards the easy construction of reversible CA.

A more specific kind of reversibility, which is common in physical theories, is that of time symmetry, and it has been only recently applied to cellular automata [4]. In time-symmetric systems there is an involution (i.e., a root-of-identity transformation) that reverses the flux of time, so that further application of the system's dynamics has the same effect as going back with the inverse transformation, if the involution had not been applied. Time symmetry is a natural property for study, because of its ubiquity in physical theories, and also because of potential practical concerns: one of the advantages of using reversible CA for distributed computation, besides heat minimization, is that we may actually reverse time ocassionally, if we want to find the source of a phenomenon or debug a computation mistake. In that case, using the same CA (and, in a practical setting, the same circuits) to move backwards in time would be certainly handy.

This work was partially supported by CONICYT-Chile through BASAL project CMM, Universidad de Chile, and FONDECYT project# 1140833.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 155–168, 2015.
DOI: 10.1007/978-3-662-47221-7_12

A further CA property that has attracted some attention is number conservation: here the CA states are assumed to be numbers, and their sum over finite -or periodic- configurations is preserved. The inspiration comes again from physics, with its conservation laws, and from models of simple interacting agents, when there is a fixed number of them. Number conservation can be seen as the most simple case of more general additive conservation laws in cellular automata (moreover, whatever is true for number-conserving CA usually carries on nicely to the more general settings).

Despite apparently being a rather restrictive property, number conservation allows the existence of intrinsically universal CA (roughly speaking, CA that can simulate any other) and hence also of CA which are Turing-universal [7]. This is also the case for time symmetry [4], with the caveat that intrinsic universality is restricted to the simulation of all other *reversible* CA. This implies the existence of full computational capabilities and of arbitrarily complex phenomena within these classes. In this manuscript we show that this is also true for the CA that share both properties. This is done in Section 3, by giving a general simulation procedure that turns arbitrary reversible CA into time-symmetric ones while preserving number conservation. The existence of reversible and number-conserving Turing-universal CA was already proved by Morita [10].

While mapping the territory we came across a possible path to the same result, which was not ultimately fruitful, but which yielded a few results and observations that are presented here in Section 4. The main result is a generalization of one from Morita [10], showing that any reversible partitioned CA can be simulated by a reversible number-conserving CA. We also define "intrinsic time symmetry" (ITS) in a natural way for partitioned CA, characterize ITS partitioned CA in terms of their local function, and show that the number-conserving CA simulating them (obtained through the generalized Morita construction) are time-symmetric. This could have been a path to the result of Section 3 *if* we had an example of intrinsic universality in ITS partitioned CA, which we currently do not. Nevertheless, the generalization does bridge two CA classes through a simulation relation. As a byproduct, it also gives a way to construct time-symmetric (number-conserving) CA by starting with ITS partitioned CA, which are rather easy to construct, and where ITS is a decidable property.

2 Definitions

Definition 1 (Cellular Automata). *A* 1-dimensional cellular automaton *(CA) A is a 4-tuple* $A = (\mathbb{Z}, Q, N, f)$ *where* Q *is a finite set called the* state set, $N = (n_1, ...n_m) \in \mathbb{Z}^m$ *is the* neighborhood, *and* $f : Q^m \to Q$ *is the* local transition function.

A CA defines a dynamics on the space $Q^{\mathbb{Z}}$ *of configurations through the* global transition function

$$F : Q^{\mathbb{Z}} \longrightarrow Q^{\mathbb{Z}},$$
$$F(\alpha)_i = f(\alpha_{i+n_1}, \alpha_{i+n_2}, ..., \alpha_{i+n_m}).$$

The size of the neighborhood is m and when $N = (-r, -r+1, .., 0, .., r)$*, we say that it is a symmetric neighborhood of* radius r*. If* $r = 0$*, the CA is* autarkic*. When* $n_j \geq 0$ *for every* j*, the CA is* one way*. CA with the asymmetric neighborhood* $(0, 1)$ *are said to have radius 1/2.*

Additive conservation laws occur in CA when some locally computed numerical attribute, added over periodic configurations, is preserved through time. The simplest version happens when the states themselves are numbers, and their sum is conserved. Here we give the definition based on finite configurations, which are configurations where all the cells, except for a finite number, are equal to 0. This has been shown to be equivalent to the definition based on periodic configurations.

Definition 2 (Number Conservation). *Given a CA* $A=(\mathbb{Z}, \{0, 1, ..., s\}, N, f)$*, we say that it is* number-conserving (NCCA) *if for every finite configuration* c

$$\sum_{i \in \mathbb{Z}} F(c)_i = \sum_{i \in \mathbb{Z}} c_i$$

Remark 1. The class of number-conserving CA is closed for composition and inverse.

Remark 2. In NCCA all states are quiescent, *i.e.*, all homogeneous configurations are fixed points.

Definition 3 (Time Symmetry). *Given a reversible CA* $A = (\mathbb{Z}, Q, N, f)$*, we say that it is* time-symmetric (TS) *if there exists a CA with global transition function* H *such that*

$$H \circ F \circ H = F^{-1} \qquad \wedge \qquad H \circ H = Id.$$

H *in this case is an* involution associated to *the time-symmetric CA* A*.*

Remark 3. A direct consequence of time symmetry is that

$$\forall n \in \mathbb{Z},\ H \circ F^n \circ H = F^{-n}.$$

Remark 4. The class of time symmetry CA is closed for inverse.

Working with reversible CA, and in particular constructing them, is a bit cumbersome since identifying reversibility from the local transition function is not direct, and only an exponential algorithm exists. This has led a number of researchers to prefer the more convenient formalism provided by *partitioned cellular automata*. They are a particular case of CA, but they are general in the sense that every CA can be simulated by a partitioned CA.

Definition 4 (Partitioned Cellular Automaton). *A* partitioned cellular automaton (PCA) *is a 4-tuple* $P = (\mathbb{Z}, Q = Q_{n_1} \times ... \times Q_{n_m}, (n_1, ..., n_m), f)$*, where* Q_i *are finite sets and* $f : Q \to Q$ *is the* local transition function *of* P*.*

The global transition function *of* P *is* $F : Q^{\mathbb{Z}} \to Q^{\mathbb{Z}}$ *given by*

$$F(\alpha)_i = f(\alpha_{i+n_1}(n_1), ..., \alpha_{i+n_m}(n_m))$$

where $\alpha \in Q^{\mathbb{Z}}$*, and* $\alpha_i(j)$ *is the coordinate indexed by* j *of the* i*-th cell in* α*.*

It is practical to view PCA as the composition of two CA: a product of shifts and an autarkic CA.

Proposition 1. *Given a PCA* $P = (\mathbb{Z}, Q = Q_{n_1} \times ... \times Q_{n_m}, (n_1, ..., n_m), f)$, *then its transition function* F *is equal to* $A_f \circ I_{(n_1,...,n_m)}$, *where*

- A_f *is the autarkic CA defined by* f: $A_f(\alpha)_i = f(\alpha_i)$,
- $I_{(n_1,...,n_m)} = \sigma^{n_1} \times \sigma^{n_2} \times ... \times \sigma^{n_m} : (Q_{n_1} \times ... \times Q_{n_m})^{\mathbb{Z}} \to (Q_{n_1} \times ... \times Q_{n_m})^{\mathbb{Z}}$,
- σ *is the shift of configurations and* $\forall n \in \mathbb{Z}, \forall i \in \mathbb{Z},\ \sigma^n(x)_i = x_{i+n}$.

Definition 5 (Universality). *In order to prevent unnecessary clutter here, we will dispense with the formal definitions of universality. Turing-universality is common lore by now, and as for intrinsic universality, we refer the reader to [2] for definitions as well as excelent discussion. However, intrinsic universally depends on the precise simulation relation being used, and it is therefore important to remark that the simulations presented here comply with the definition of "injective bulking" introduced in [2], and that is the sense in which intrinsic universality must be understood.*

3 Universality

In [4] arbitrary reversible CA are transformed into time-symmetric CA by using the fact that the product CA $F \times F^{-1}$ simulates F and is time-symmetric. If the original CA is number-conserving, the product CA does not have to be, since there is no natural relabeling of its states (which are ordered pairs) that may ensure number conservation. On the other hand, one of the ideas used by Morita [10] to transform a reversible PCA into a number-conserving CA is to represent vectors through numbers over a larger set. Here we combine these two ideas to transform a reversible number-conserving CA into a time-symmetric number-conserving CA.

Theorem 1. *Given a reversible number-conserving CA there exists a time-symmetric number-conserving CA that simulates it.*

Proof. Let $A = (\mathbb{Z}, Q = \{0, 1, ..., s-1\}, N, f)$ be a reversible and number conserving CA. Let us define first the automaton $\overline{A} = (\mathbb{Z}, \overline{Q}, N, \overline{f})$ where $\overline{Q} = \{ks : k = 0, 1, ..., s-1\}$ and $\overline{f}(\overline{q}_1, ..., \overline{q}_{|N|}) = sf(\overline{q}_1/s, ..., \overline{q}_{|N|}/s)$. It is just A, but with the states multiplied by s. Now let Q' be $Q' = \{0, 1, ..., s^2 - 1\}$; clearly, for every $q' \in Q'$ there is a unique pair $(q, \overline{q}) \in Q \times \overline{Q}$ such that $q' = q + \overline{q}$. We can thus define two projections $p : Q' \to Q$ and $\overline{p} : Q' \to \overline{Q}$ such that for every $q' \in Q'$, $q' = p(q') + \overline{p}(q')$. Let P and $\overline{P}$ be the natural generalizations of these functions to $Q'^{\mathbb{Z}}$. Our time-symmetric number-conserving CA $A' = (\mathbb{Z}, Q', N, f')$ is defined through the global transition function

$$F'(\alpha) = F(P(\alpha)) + \overline{F}^{-1}(\overline{P}(\alpha))$$

Such a cellular automaton exists because F is reversible, $\overline{F}$ is conjugated to F, and because the addition of CAs is also a CA, by the Hedlund-Lyndon-Curtis theorem. Clearly, F' simulates F, because $Q \subset Q'$, and if $\alpha \in Q^{\mathbb{Z}}$ then $F'(\alpha) = F(\alpha) + \overline{F}^{-1}(0) = F(\alpha)$. Let us prove now that F' has the requested properties.

Reversibility. The inverse of F' is given by $F'^{-1}(\alpha) = F^{-1}(P(\alpha)) + \overline{F}(\overline{P}(\alpha))$:

$$\begin{aligned} F'^{-1}(F'(\alpha)) &= F'^{-1}\big(\underbrace{F(P(\alpha))}_{\in Q} + \underbrace{\overline{F}^{-1}(\overline{P}(\alpha))}_{\in \overline{Q}}\big) \\ &= F^{-1}(F(P(\alpha))) + \overline{F}(\overline{F}^{-1}(\overline{P}(\alpha)) \\ &= P(\alpha) + \overline{P}(\alpha) \\ &= \alpha. \end{aligned}$$

Number conservation. Let $\alpha \in Q'^{\mathbb{Z}}$ be a finite configuration. Since both F and $\overline{F}$ are number-conserving,

$$\begin{aligned} \sum_{i\in\mathbb{Z}} F'(\alpha)_i &= \sum_{i\in\mathbb{Z}} \Big(F(P(\alpha))_i + \overline{F}^{-1}(\overline{P}(\alpha))_i\Big) \\ &= \sum_{i\in\mathbb{Z}} F(P(\alpha))_i + \sum_{i\in\mathbb{Z}} \overline{F}^{-1}(\overline{P}(\alpha))_i \\ &= \sum_{i\in\mathbb{Z}} P(\alpha)_i + \sum_{i\in\mathbb{Z}} \overline{P}(\alpha)_i \\ &= \sum_{i\in\mathbb{Z}} \big(P(\alpha)_i + \overline{P}(\alpha)_i\big) \\ &= \sum_{i\in\mathbb{Z}} \alpha_i \end{aligned}$$

Time symmetry. The involution H that makes F' time-symmetric is defined by $H(\alpha) = \underbrace{s \cdot P(\alpha)}_{\in\overline{Q}} + \underbrace{\frac{1}{s} \cdot \overline{P}(\alpha)}_{\in Q}$. It is an autarkic CA, and $H \circ H = I$. Let us prove that $H \circ F' \circ H = F'^{-1}$,

$$\begin{aligned} H(F'(H(\alpha))) &= H\left(F\left(\frac{1}{s} \cdot \overline{P}(\alpha)\right) + \overline{F}^{-1}(s \cdot P(\alpha))\right) \\ &= sF\left(\frac{1}{s} \cdot \overline{P}(\alpha)\right) + \frac{1}{s}\overline{F}^{-1}(s \cdot P(\alpha)) \\ &= s\frac{1}{s}\overline{F}\left(\overline{P}(\alpha)\right) + \frac{1}{s}sF^{-1}(P(\alpha)) \\ &= F'^{-1}(\alpha) \end{aligned}$$

□

Corollary 1. *There exists a Turing-universal time-symmetric and number-conserving CA.*

Proof. This follows from the existence of a Turing-universal reversible number-conserving CA, established by Morita [10] based on several of his previous results.

Corollary 2. *There exists an intrinsically-universal time-symmetric and number-conserving CA within the class of reversible CA.*

Proof. Durand-Lose [3] proves the existence of a reversible intrinsically universal PCA of radius 1. On the other hand, Morita shows in [8] that any RPCA of radius 1 can be simulated by a one-way RPCA of radius 1/2, and in [10] that any one-way RPCA of radius 1/2 can be simulated by a reversible NCCA. Along with the present theorem, these results imply the existence of a time-symmetric intrinsically universal NCCA.

4 The Road Not Taken

In [10], in order to obtain a Turing-universal reversible NCCA with a small neighborhood, Morita developed a way to transform a one way PCA of radius 1/2 into a number-conserving CA of radius 3/2 (*i.e.* neighborhood $(-1, 0, 1, 2)$). His CA is not time-symmetric and it *could not* have been, since time symmetry in a one way PCA implies equicontinuity (see Proposition in the Appendix), which in turns implies periodicity. Nevertheless, his construction can be generalized to PCA of arbitrary neighborhood, and as we shall see, it does preserve time symmetry if it is present (in a sense to be defined) in the original PCA.

Proposition 2. *Given a reversible PCA, there exists a reversible and number-conserving CA that simulates it.*

Proof. Let P be a reversible PCA with state sets $Q_i = \{q_{i0}, q_{i1}, ..., q_{i(|Q_i|-1)}\}$; without loss of generality, and only for the sake of simplifying the proof, its neighborhood can be assumed to have the form $(-n, ..., n)$.

We follow the idea of Morita's construction in [10], which is to duplicate each set of states Q_i into two sets that we call here $\hat{S}_i \subseteq \mathbb{N} \cup \{0\}$ and $\check{S}_i \subseteq \mathbb{N} \cup \{0\}$, so that for each q_{ij} there exist $\hat{s}_{ij} \in \hat{S}_i$ and $\check{s}_{ij} \in \check{S}_i$ satisfying that $\hat{s}_{ij} + \check{s}_{ij}$ is a constant that depends only on i. In order to compute this numbers we define two bijective funcions for each i: $\hat{\phi}_i$ and $\check{\phi}_i$, which also ensure that the $\check{s}_i$ and $\hat{s}_i$ of different i are on different scales and can be added into a single number wihout losing information.

The k-th cell of a configuration is represented by two cells in the number conserving CA, indexed by $2k$ and $2k+1$. In this way, the sum of the $2k$-th and $2k+1$-th cells in the representing configuration will be equal to a constant K, independent from the content of the original cell. Now we formally define all of the needed sets and functions, for each $i \in \{-n, .., n\}$:

- $K_{-n-1} = 1$, $K_i = \prod_{j=-n}^{i} 2|Q_j|$
- $\hat{S}_i = \{kK_{i-1} : k = 0, ..., |Q_i| - 1\}$
- $\check{S}_i = \{(k + |Q_i|)K_{i-1} : k = 0, ..., |Q_i| - 1\}$
- $\hat{\phi}_i : Q_i \to \hat{S}_i$, $\hat{\phi}_i(q_{ik}) = kK_{i-1}$
- $\check{\phi}_i : Q_i \to \check{S}_i$, $\check{\phi}_i(q_{ik}) = (2|Q_i| - k - 1)K_{i-1}$
- $\hat{S} = \sum_{i=-n}^{n} \hat{S}_i$, $\check{S} = \sum_{i=-n}^{n} \check{S}_i$
- $S_i = \hat{S}_i \cup \check{S}_i$, $S = \sum_{i=-n}^{n} S_i$
- $\hat{\phi} : Q \to \hat{S}$, $\hat{\phi}(a_{-n}, ..., a_n) = \sum_{j=-n}^{n} \hat{\phi}_j(a_j)$
- $\check{\phi} : Q \to \check{S}$, $\check{\phi}(a_{-n}, ..., a_n) = \sum_{j=-n}^{n} \check{\phi}_j(a_j)$

With these definitions it is easy to see that

- there are projections $p_i : S \to S_i$ such that for $s \in S$, $s = \sum_{i=-n}^{n} p_i(s)$;
- $|Q_i| = |\hat{S}_i| = |\check{S}_i| = \frac{|S_i|}{2}$;
- $\forall i, \forall j, \hat{\phi}_i(q_{ij}) + \check{\phi}_i(q_{ij}) = K_i - K_{i-1}$; and
- $K_n = |S|$.

Each cell of P is transformed into two cells of the new automaton by an injective but not surjective function $\Psi : Q^{\mathbb{Z}} \to S^{\mathbb{Z}}$, defined as follows.

$$\Psi(\alpha)_i = \begin{cases} \hat{\phi}(\alpha_{i/2}) & \text{if } i \text{ is even} \\ \check{\phi}(\alpha_{(i-1)/2}) & \text{if } i \text{ is odd} \end{cases}$$

It is not difficult to define an $\overline{F} : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ such that $\Psi \circ F = \overline{F} \circ \Psi$ over the set $\Psi(Q^{\mathbb{Z}})$; however, we need it to be number-conserving and reversible over the whole set $S^{\mathbb{Z}}$. This will be done by considering a decomposition $\overline{F} = \bar{A}_f \circ \bar{I}$ (akin to that of Proposition 1) which makes the diagram below commute, and choosing $\bar{A}_f$ and $\bar{I}$ in a way that ensures reversibility and number conservation in each of them.

$$\begin{array}{ccccc} Q^{\mathbb{Z}} & \xrightarrow{\quad I_{(-n,..,n)} \quad} & Q^{\mathbb{Z}} & \xrightarrow{\quad A_f \quad} & Q^{\mathbb{Z}} \\ \Psi \downarrow & & \Psi \downarrow & & \Psi \downarrow \\ S^{\mathbb{Z}} & \xrightarrow{\quad \bar{I} \quad} & S^{\mathbb{Z}} & \xrightarrow{\quad \bar{A}_f \quad} & S^{\mathbb{Z}} \end{array}$$

$\bar{I} : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ is just the equivalent of $I_{(-n,..,n)}$:

$$\bar{I}(\alpha)_i = \sum_{j=-n}^{n} p_j(\alpha_{i+2j}).$$

When Ψ duplicates information into pairs of cells, we can distinguish between the *left* member (L) and the *right* member (R) of each pair, and moreover, these can be readily identified in configurations obtained through Ψ. In arbitrary

configurations there can be *wrong* cells which are neither left nor right. Formally, given a configuration $\alpha \in S^{\mathbb{Z}}$, we define $L(\alpha), R(\alpha), W(\alpha) \subseteq \mathbb{Z}$ as follows:

$$i \in L(\alpha) \Leftrightarrow \forall j \in \{-n, .., n\}, p_j(\alpha_i) \in \hat{S}_j \wedge p_j(\alpha_i) + p_j(\alpha_{i+1}) = K_j - K_{j-1},$$
$$i \in R(\alpha) \Leftrightarrow \forall j \in \{-n, .., n\}, p_j(\alpha_i) \in \check{S}_j \wedge p_j(\alpha_{i-1}) + p_j(\alpha_i) = K_j - K_{j-1},$$
$$i \in W(\alpha) \Leftrightarrow i \notin L(\alpha) \wedge i \notin R(\alpha).$$

The function $\bar{A}_f : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ is then defined by

$$\bar{A}_f(\alpha)_i = \begin{cases} \hat{\phi} \circ f \circ \hat{\phi}^{-1}(\alpha_i) & \text{if } i \in L(\alpha) \\ \check{\phi} \circ f \circ \check{\phi}^{-1}(\alpha_i) & \text{if } i \in R(\alpha) \\ \alpha_i & \text{if } i \in W(\alpha) \end{cases}$$

Let us remark that $\bar{A}_f$ is not autarkic, but of radius 1, since the neighbors of a cell must be known in order to determine its class. It is clear that $\bar{I}$ is reversible and number-conserving. To show that $\bar{A}_f$ has the same properties, we need to follow Morita and prove first that it preserves the sets L, R and W.

To see this, first we notice that L and R cells come in pairs: if $i \in L(\alpha)$, then for every j we have $p_j(\alpha_{i+1}) = K_j - K_{j-1} - p_j(\alpha_i)$ and $p_j(\alpha_{i+1}) \in \check{S}_j$, and hence $i + 1 \in R(\alpha)$; the converse is analogous. Any configuration consists therefore of blocks of L-R pairs, possibly separated by cells in W. Moreover, when $i \in L(\alpha)$, we have $\hat{\phi}^{-1}(\alpha_i) = \check{\phi}^{-1}(\alpha_{i+1})$. Therefore, when $\bar{A}_f$ is applied, $f(\hat{\phi}^{-1}(\alpha_i)) = f(\check{\phi}^{-1}(\alpha_{i+1}))$, $\bar{A}_f(\alpha)_i \in \hat{S}$ and $\bar{A}_f(\alpha)_{i+1} \in \check{S}$. This implies that $p_j(\bar{A}_f(\alpha)_i) + p_j(\bar{A}_f(\alpha)_{i+1}) = K_j - K_{j-1}$, for each j, and thus $i \in L(\bar{A}_f(\alpha))$ and $i+1 \in R(\bar{A}_f(\alpha))$. In other words, pairs in α are still pairs in $\bar{A}_f(\alpha)$, and neither cell can form a pair with a cell that was in W; so, W is preserved too.

From this fact we can see that $\bar{A}_f$ is number-conserving, since cells in W are not modified and the sum of the pairs is constant, equal to $|S| - 1$. Reversibility is deduced from this too, since a bijective function is applied to each cell. Its inverse is expressed as follows.

$$\bar{A}_f{}^{-1}(\alpha)_i = \begin{cases} \hat{\phi} \circ f^{-1} \circ \hat{\phi}^{-1}(\alpha_i) & \text{if } i \in L(\alpha) \\ \check{\phi} \circ f^{-1} \circ \check{\phi}^{-1}(\alpha_i) & \text{if } i \in R(\alpha) \\ \alpha_i & \text{if } i \in W(\alpha) \end{cases}$$

We conclude that $\bar{F} = \bar{A}_f \circ \bar{I}$ is also reversible and number-conserving. □

Since PCA are CA, the general definition of time symmetry applies to them: a PCA F is time-symmetric if there is an involution H such that $F \circ H = H \circ F^{-1}$. Notice, however, that H does not need to be a PCA; in fact, neither does F^{-1}. We introduce a more convenient restricted notion of time symmetry, which requires H to be a PCA with the same neighborhood of F.

Definition 6. *Given a reversible PCA* $P = (\mathbb{Z}, Q = Q_{n_1} \times ... \times Q_{n_m}, (n_1, ..., n_m), f)$ *we say that it is* intrinsically time-symmetric *(ITS) if there exists a PCA* $H = (\mathbb{Z}, Q = Q_{n_1} \times ... \times Q_{n_m}, (n_1, ..., n_m), h)$ *such that*

$$H \circ F \circ H = F^{-1} \qquad \wedge \qquad H \circ H = Id.$$

Proposition 3. *A PCA P is intrinsically time-symmetric if and only if its neighborhood $(n_1, ..., n_m)$ is symmetric, and there exists a family of functions $h_i : Q_i \to Q_{-i}$ such that*

- $h_j = h^{-1}_{m+1-j}$, *and*
- $h = (h_1, ..., h_m)$ *satisfies* $f \circ h = h \circ f^{-1}$.

Proof. As a first and also main step, we will prove that a PCA $H = (\mathbb{Z}, Q = Q_{n_1} \times ... \times Q_{n_m}, (n_1, ..., n_m), h)$ is an involution if and only if its neighborhood is symmetric, the components h_j of $h = (h_1, ..., h_m)$ depend only of their $(m+1-j)$-th coordinate, and they verify $h_j = h^{-1}_{m+1-j}$.

Suppose that H is an involution and let h_j be the components of h, so that $h = (h_1, ..., h_m)$. Notice that, a priori, each h_j is a function with the whole set Q as its domain. Let $(g_j : Q \to Q_j)_{j=1}^m$ be a family of functions such that $h^{-1} = g = (g_1, ..., g_m)$. Let $H = A_h \circ I$ be the decomposition of H as per Proposition 1; recall that here $I = I_{(n_1,...,n_m)}$ is a product of shifts. Since H is an involution we have

$$I \circ A_h \circ I = A_h^{-1} = A_{h^{-1}}$$

$$\forall i \in \mathbb{Z}, \quad (I \circ A_h \circ I(\alpha))_i = (g_1(\alpha_i), ..., g_j(\alpha_i), ..., g_m(\alpha_i))$$

$$(h_1(I(\alpha)_{i+n_1}), ...h_j(I(\alpha)_{i+n_j}), ...h_m(I(\alpha)_{i+n_m})) = (g_1(\alpha_i), ..., g_j(\alpha_i), ..., g_m(\alpha_i))$$

We obtain that $g_j(\alpha_i) = h_j(I(\alpha)_{i+n_j})$, for each $j \in \{1, ..., m\}$. However, if we consider the decomposition of α_i as $(\alpha_i(1), ..., \alpha_i(m))$, what we actually have is

$$g_j(\alpha_i(1), ..., \alpha_i(k), ..., \alpha_i(m)) = h_j(\alpha_{i+n_j+n_1}(1), ..., \alpha_{i+n_j+n_k}(k), ..., \alpha_{i+n_j+n_m}(m))$$

These two maps are equal and they depend on different sets of variables. They are not constant (since the global function is bijective), and must therefore really depend only on the variables they have in common. There is at most one of these for each j, and requires that for some k we have $n_j + n_k = 0$. This implies that the neighborhood must be symmetric, $g_j = h_j$ depends only on the $(m+1-j)$-th variable, and $|Q_j| = |Q_{m+1-j}|$.

Now using that $g = h^{-1}$, we obtain that $h_j^{-1} = g_{m+1-j} = h_{m+1-j}$.

The converse is trivial: if h verifies the properties described above, it is easy to show that indeed $h = h^{-1}$, and $H = H^{-1}$.

The only part of the proposition which remains to be proved is the time symmetry relation verified at the local level. By using the decomposition of Proposition 1, let us write the global transition functions of the PCA P and of its PCA involution as $F = A_f \circ I_{(n_1,...,n_m)}$ and $H = A_h \circ I_{(n_1,...,n_m)}$ respectively. The time symmetry of F by way of H is equivalent to the next equations:

$$H \circ F \circ H = F^{-1}$$

$$(A_h \circ I) \circ (A_f \circ I) \circ (A_h \circ I) = I^{-1} \circ A_f^{-1}$$

$$(I \circ A_h \circ I) \circ A_f \circ (I \circ A_h \circ I) = A_f^{-1}$$

$$A_h \circ A_f \circ A_h = A_f^{-1} = A_{f^{-1}}$$

Since A_f, $A_{f^{-1}}$ and A_h are autarkic, this last equation is equivalent to

$$h \circ f \circ h = f^{-1}.$$

□

It was shown in [4] that every reversible autarkic CA is time-symmetric, that is, for every bijective function $f : Q \to Q$ there exists some involution $h : Q \to Q$ such that $h \circ f \circ h = f^{-1}$. Proposition 3 imposes additional conditions to h, which will not be always verified. Thus, not every reversible f defines an ITS PCA. For example, no involution h on $Q = \{0,1\} \times \{0\} \times \{0,1\}$ satisfies $h \circ f \circ h = f^{-1}$ for the function f that traverses the cycle $((0,0,1)(1,0,0)(1,0,1)(0,0,0))$.

Proposition 4. *If F is the transition function of an ITS PCA, then the number-conserving function $\overline{F}$ produced by Proposition 2 is time-symmetric.*

Proof. Let H be the involution that makes F time-symmetric; we can suppose without loss of generality that the neighborhood is $\{-n, ..., n\}$. The involution that will work for $\bar{F}$ acts just like H on S, ignoring whether the coordinates are in $\hat{S}_j$ or $\check{S}_j$. For this we define the signature of a number $s \in S$ as the sets to which its projections belong, as follows.

$$\begin{aligned} \mathcal{C} : S &\to \{\wedge, \vee\}^{\{-n,\dots,n\}} \\ \mathcal{C}(s) &= (c_{-n}(s), ..., c_n(s)) \\ \forall j \in \{-n, ..., n\}, c_j(s) &= \begin{cases} \wedge \text{ if } p_j(s) \in \hat{S}_j \\ \vee \text{ if } p_j(s) \in \check{S}_j \end{cases} \end{aligned}$$

For convenience we define $S_j^{\vee} = \check{S}_j$, $S_j^{\wedge} = \hat{S}_j$, $\phi_j^{\vee} = \check{\phi}_j$ and $\phi_j^{\wedge} = \hat{\phi}_j$. Now $\tilde{A}_h$ is defined through the function h by using the functions $\phi^c : Q \to \sum_{j=-n}^{n} S_j^{c_j}$, defined by $\phi^c(q) = \sum_{j=-n}^{n} \phi_j^{c_j}(q_j)$ to transform states to numbers with signature c and vice versa. We will use the notation $\overleftarrow{c} = (c_n, .., c_{-n})$.

$$\begin{aligned} \tilde{A}_h &: S^{\mathbb{Z}} \to S^{\mathbb{Z}} \\ \tilde{A}_h(\alpha)_i &= \phi^{\overleftarrow{\mathcal{C}(\alpha_i)}}(h((\phi^{\mathcal{C}(\alpha_i)})^{-1}(\alpha_i))) \end{aligned}$$

$\tilde{A}_h$ is autarkic and it is an involution. Let us see that the associated automaton $\bar{H} = \tilde{A}_h \circ \bar{I}$ is also an involution, with $\overline{I}$ taken from Proposition 2.

$$\begin{aligned} (\tilde{A}_h \circ \bar{I}(\alpha))_i &= \phi^{\overleftarrow{\mathcal{C}(\overline{I}(\alpha_i))}} \left(h \left(\left(\phi^{\mathcal{C}(\overline{I}(\alpha_i))} \right)^{-1} \left(\overline{I}(\alpha_i) \right) \right) \right) \\ &= \sum_{j=-n}^{n} \phi_j^{c_{-j}(\overline{I}(\alpha_i))} \left(h_j \left(\left(\phi^{\mathcal{C}(\overline{I}(\alpha_i))} \right)^{-1} \left(\sum_{j=-n}^{n} p_j(\alpha_{i+2j}) \right) \right) \right) \end{aligned}$$

We use Proposition 3.

$$= \sum_{j=-n}^{n} \phi_j^{c_{-j}(\alpha_{i-2j})} \left(h_j \left(\left(\left(\phi^{c_{-j}(\alpha_{i-2j})} \right)_{-j}^{-1} (p_{-j}(\alpha_{i-2j})) \right) \right) \right)$$

We apply $\bar{I}$ from the left.

$$\begin{aligned}
\left(\bar{I} \circ \tilde{A}_h \circ \bar{I}(\alpha) \right)_i &= \sum_{j=-n}^{n} p_j \left(\tilde{A}_h \circ \bar{I}(\alpha)_{i+2j} \right) \\
&= \sum_{j=-n}^{n} \phi_j^{c_{-j}(\alpha_{i+2j-2j})} \left(h_j \left(\left(\phi^{c_{-j}(\alpha_{i+2j-2j})} \right)_{-j}^{-1} (p_{-j}(\alpha_{i+2j-2j})) \right) \right) \\
&= \sum_{j=-n}^{n} \phi_j^{c_{-j}(\alpha_i)} \left(h_j \left(\left(\phi^{c_{-j}(\alpha_i)} \right)_{-j}^{-1} (p_{-j}(\alpha_i)) \right) \right) \\
&= \sum_{j=-n}^{n} \phi_j^{c_{-j}(\alpha_i)} \left(h_j \left(\left(\phi^{\mathcal{C}(\alpha_i)} \right)_{-j}^{-1} (\alpha_i) \right) \right) \\
&= \tilde{A}_h(\alpha)_i
\end{aligned}$$

Now we just need to prove that $\bar{H} \circ \bar{F} \circ \bar{H} = \bar{F}^{-1}$. As we saw in the proof of the previous proposition, all we need to show is that $\tilde{A}_h \circ \bar{A}_f \circ \tilde{A}_h = (\bar{A}_f)^{-1}$. We analyze by cases depending on the class of cell i, and using the fact that if $s \in \hat{S}$ (or $\check{S}$) then $\phi^{\mathcal{C}(s)}(s) = \hat{\phi}(s)$ ($\phi^{\mathcal{C}(s)}(s) = \check{\phi}(s)$).

Case 1: $i \in L(\alpha)$.

$$\begin{aligned}
\tilde{A}_h \circ \bar{A}_f \circ \tilde{A}_h(\alpha)_i &= \hat{\phi} \circ h \circ \hat{\phi}^{-1} \circ \hat{\phi} \circ f \circ \hat{\phi}^{-1} \circ \hat{\phi} \circ h \circ \hat{\phi}^{-1}(\alpha_i) \\
&= \hat{\phi} \circ h \circ f \circ h \circ \hat{\phi}^{-1}(\alpha_i) \\
&= \hat{\phi} \circ f^{-1} \circ \hat{\phi}^{-1}(\alpha_i) \\
&= \bar{A}_{f^{-1}}(\alpha)_i \\
&= \bar{A}_f^{-1}(\alpha)_i
\end{aligned}$$

Case 2: $i \in R(\alpha)$. It is analogous to Case 1.

Case 3: $i \in W(\alpha)$. We remark first that $\tilde{A}_h(\alpha)_i \in W$. Thus $\bar{A}_f(\tilde{A}_h(\alpha)_i) = \tilde{A}_h(\alpha)_i$. Since $\tilde{A}_h$ is autarkic, $(\tilde{A}_h(\bar{A}_f(\tilde{A}_h(\alpha))))_i = (\tilde{A}_h(\tilde{A}_h(\alpha)))_i = \alpha_i = (\bar{A}_f)^{-1}(\alpha)_i$.

□

5 Final Remarks

The road described in Section 4 could have led to the corollaries of Section 3, provided that universal ITS PCA had been known to exist. Let us briefly discuss the situation for each kind of universality.

Is there an ITS Turing-universal PCA? Probably. In 1989 Morita and Harao [11] showed a way to simulate any given reversible Turing machine with a PCA of radius 1. We currently conjecture that the construction (perhaps with slight modifications) yields an ITS PCA whenever the Turing machine is itself "time-symmetric". Since there has been little discussion of time symmetry in Turing machines, the notion used in this case is the simple one given in [1], where the involution consists of an autarkic CA involution on the tape, along with an independent permutation of the machine's state set. The Turing machine needs to be expressed in quadruples in order to give sense to this notion, since only in this case does its inverse transition function correspond to a Turing machine. With this definition, universal time-symmetric Turing machines do exist: following the technique of Kari and Ollinger [6], in order to obtain one, it is enough to join the inverse and the direct machine into one.

Is there an ITS intrinsically universal PCA? The intrinsically universal PCA built by Durand-Lose [3] is *not* ITS, but we believe that it could be modified to make it comply. In any case, its construction is quite complex and requires many states. Even if the modification worked, the existence of a simple intrinsically universal ITS PCA (or indeed, time-symmetric CA) remains an open question.

Finally, please notice that Section 4 provides a path for the construction of time-symmetric CA (possibly with the added bonus of number conservation), of which there are not that many. Besides this, the alternatives previously known to us were 1) to find involutions and blindly compose them, or 2) to design reversible PCA with a prescribed behavior (something for which PCA are quite useful), turn it into a reversible CA, and convert this into a time-symmetric CA. The new option is to construct an ITS PCA and then use Proposition 2 to get a time-symmetric CA. The ITS PCA could be designed by hand, or we could construct reversible PCA and test for the existence of an appropriate involution using Proposition 3, which is an improvement with respect to the general 1D CA case, where time symmetry is conjectured to be undecidable.

6 Appendix

Definition 7 (Equicontinuity). *A CA with transition function F is* equicontinous *if for every N there exists $M \in \mathbb{N}$ such that $\alpha_{[-M,M]} = \beta_{[-M,M]}$ implies that for every $t \in \mathbb{N}$, $F^t(\alpha)_{[-N,N]} = F^t(\beta)_{[-N,N]}$.*

Definition 8 (N-sensitivity). *A configuration α is said to be* N-sensitive *if for every $M \in \mathbb{N}$ there exists β and $t \in \mathbb{N}$ such that $\alpha_{[-M,M]} = \beta_{[-M,M]}$ and $F^t(\alpha)_{[-N,N]} \neq F^t(\beta)_{[-N,N]}$.*

It is known that a CA is equicontinous if and only if none of its configurations is N-sensitive, for any $N \in \mathbb{N}$.

Proposition 5. *A one way time-symmetric PCA must be equicontinuous.*

Proof. Let us first remark that one way CA can be time-symmetric without being periodic, even if the inverse is also one way in the same direction; therefore, our proof strongly relies on the partitioned nature of the CA.

Let P be a time-symmetric PCA which is one way in the right direction, *i.e.*, its neighborhood $(n_1, .., n_m)$ contains only non-negative coordinates, with n_m being the biggest one.

Remark 5. A particular feature of reversible PCA is that when computing the preimage of a configuration, the content of a given cell determines, without ambiguity, part of the contents of the m cells in its neighborhood. Which conversely implies that if one cell i is perturbated, at least one cell in $\{i - n_1, .., i - n_m\}$ will 'see' the difference in the next iteration of P.

Let us suppose that P is time-symmetric with an involution H of radius r, and that it is *not* equicontinuous. Let α be an N-sensitive configuration, and β a configuration given by the definition for $M = N + 2r$. Furthermore, let t be the least one for which $F^t(\alpha)_{[-N,N]} \neq F^t(\beta)_{[-N,N]}$. Since P is one way, we can assure that the difference between these two last configurations lies between $N - n_m$ and N. Moreover, we can assume that $\alpha_{[-\infty,M]} = \beta_{[-\infty,M]}$, because differences to the left of $-M$ will not perturbate cells in $[-N, N]$.

From time symmetry, we have that $H(\beta) = F^t(H(F^t(\beta)))$. But

$$H(\beta)_{[-\infty,N+r]} = H(\alpha)_{[-\infty,N+r]}, \text{ and}$$
$$H(F^t(\beta))_{[N-n_m-r,N+r]} \neq H(F^t(\alpha))_{[N-n_m-r,N+r]}$$

and using Remark 5, we obtain that

$$F^t(H(F^t(\beta)))_{[N-n_m-r-tn_m,N+r]} \neq F^t(H(F^t(\alpha)))_{[N-n_m-r-tn_m,N+r]}, \text{ i.e.}$$
$$H(\beta)_{[N-n_m-r-tn_m,N+r]} \neq H(\alpha)_{[N-n_m-r-tn_m,N+r]},$$

which is a contradiction. □

References

1. Cassaigne, J., Ollinger, N., Torres-Avilés, R.: A Small Minimal Aperiodic Reversible Turing Machine (2014). https://hal.inria.fr/hal-00975244
2. Delorme, M., Mazoyer, J., Ollinger, N., Theyssier, G.: Bulking II: classifications of cellular automata. Theoretical Computer Science **412**(30), 3881–3095 (2011)
3. Durand-Lose, J.: Intrinsic universality of a 1-dimensional reversible cellular automaton. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 439–450. Springer, Heidelberg (1997)
4. Gajardo, A., Kari, J., Moreira, A.: On time-symmetry in cellular automata. Journal of Computer and System Sciences **78**(4), 1115–1126 (2012)
5. Kari, J.: Reversible cellular automata. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 57–68. Springer, Heidelberg (2005)

6. Kari, J., Ollinger, N.: Periodicity and immortality in reversible computing. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 419–430. Springer, Heidelberg (2008)
7. Moreira, A.: Universality and decidability of number-conserving cellular automata. Theoretical Computer Science **292**(3), 711–721 (2003)
8. Morita, K.: Computation-universality of one-dimensional one-way reversible cellular automata. Information Processing Letters **42**(6), 325–329 (1992)
9. Morita, K.: Reversible computing and cellular automata: A survey. Theoretical Computer Science **395**(1), 101–131 (2008)
10. Morita, K.: Universality of one-dimensional reversible and number-conserving cellular automata. In: Formenti, E. (ed.) AUTOMATA JAC. EPTCS, vol. 90, pp. 142–150 (2012)
11. Morita, K., Harao, M.: Computation-universality of one-dimensional reversible (injective) cellular automata. The Transactions of the IEICE **72**(6), 758–762 (1989)

The Ideal Energy of Classical Lattice Dynamics

Norman Margolus(✉)

Massachusetts Institute of Technology, Cambridge, MA, USA
nhm@mit.edu

Abstract. We define, as local quantities, the least energy and momentum allowed by quantum mechanics and special relativity for physical realizations of some classical lattice dynamics. These definitions depend on local rates of finite-state change. In two example dynamics, we see that these rates evolve like classical mechanical energy and momentum.

1 Introduction

Despite appearances to the contrary, we live in a finite-resolution world. A finite-sized physical system with finite energy has only a finite amount of distinct detail, and this detail changes at only a finite rate [1–3]. Conversely, given a physical system's finite rates of distinct change in time and space, general principles of quantum mechanics define its minimum possible average energy and momentum. We apply these definitions to classical finite-state lattice dynamics.

1.1 Ideal Energy

It was finiteness of distinct state, first observed in thermodynamic systems, that necessitated the introduction of Planck's constant h into physics [4]. Quantum mechanics manages to express this finiteness using the same continuous coordinates that are natural to the macroscopic world. Describing reality as superpositions of waves in space and time, finite momentum and energy correspond to effectively finite bandwidth; hence finite distinctness. For example [3], the average rate ν at which an isolated physical system can traverse a long sequence of distinct states is bounded by the average (classical) energy E:

$$\nu \leq 2\,E/h\,, \tag{1}$$

taking the minimum possible energy to be zero. Here E/h is the average frequency of the state, which defines a half-width for the energy frequency distribution. If we compare (1) in two frames, we can bound the average rate μ of changes not visible in the rest frame, and hence attributable to overall motion:

$$\mu \leq 2\,pv/h\,. \tag{2}$$

Here p is the magnitude of a system's average (classical) momentum, which is also a half-width for a (spatial) frequency distribution; v is the system's speed.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 169–180, 2015.
DOI: 10.1007/978-3-662-47221-7_13

These kinds of constraints are sometimes referred to as uncertainty bounds, but they in no way preclude precise finite-state evolution. Given rates of change, these bounds define ideal (minimum achievable) average energy and momentum for finite state systems, emulated as efficiently as possible (with no wasted motion or state) by perfectly-tailored quantum hamiltonians [3,5].

Clearly there can never be more *overall spatial change* μ than *total change* ν in a physical evolution: this is reflected in $pv/E = (v/c)^2$. From this and (2),

$$E \geq (h\mu/2)/(v/c)^2 \,. \tag{3}$$

Thus for a given rate μ of overall motional change, E can only attain its minimum possible value if the motion is at the speed of light; then no energy is invested in rest-frame dynamics (rest energy). In a finite-state dynamics with several geometrically related signal speeds, to minimize all energies (3) the fastest signals must move at the speed of light. If we then want to realize the dynamics running faster, we must put the pieces of the system closer together: we can increase p in (2), but not v. Of course in finite-state models of particular physical systems, realistic constraints on speeds and separations may require higher energies.

These bounds can be used to define ideal local energies and momenta for some invertible lattice dynamics, determined by rates of distinct change.

1.2 Local Change

We restrict our attention to finite-state lattice dynamics that emulate the locality, uniformity and microscopic invertibility of physical law: invertible cellular automata (CA). We assume the dynamics is defined as a regular arrangement of invertible interactions (logic gates), repeated in space and time, each of which independently transforms a localized set of state variables.

This kind of CA format, where the state variables are always updated in independent groups, has sometimes been called partitioning CA, and encompasses a variety of lattice formats that have been used to model physical dynamics [6–13]. It is interesting that all globally invertible CA can be recast in this physically realistic format, as a composition of independent invertible interactions, even if the CA was originally defined as a composition of non-invertible operations on overlapping neighborhoods [14–16]. Historically, CA originated as physics-like dynamics *without* invertibility [17–19].

Now, in the energy bounds above, only rates of change matter, not the amount of state updated in a single operation. This is unrealistic. We can define a large-scale synchronous dynamics, where the global rate of state change is independent of the size of the system. Physically, total energy must be bounded by the total rate of local changes, since each independent local update also obeys an energy bound. We resolve this conflict by allowing synchronous definition, but counting the global average rate of distinct change as if local updates were non-synchronous—which would in fact be true in most relativistic frames.

There is also an issue of what not to count. For a dynamics defined by a set of gate operations, it might seem natural to include, in the minimum,

energy required to construct the gates and to turn them on and off. This is the energy needed to construct a perfectly-tailored hamiltonian. Here we ignore this construction energy, and discuss the ideal case where the hamiltonian is given for free (as part of nature), and we only need to account for energy required by state change within the dynamics.

1.3 Two Examples

In the remainder of this paper, we introduce and discuss two 2×2 block partitioning CA (*cf.* [20]). These dynamics are isomorphic to classical mechanical systems, and are simple enough that it is easy to compare energetic quantities, defined by local rates of state change, with classical ones.

The first example is a scalable CA version of the Soft Sphere Model [21], which is similar to Fredkin's classical mechanical Billiard Ball Model [22]. This digital system emulates the integer time behavior of an idealized classical mechanical system of elastically colliding balls, and is computation universal. The CA is scalable in that square blocks of ones (balls) of any size can be collided to simulate a billiard ball computation. This model has not been published before.

The second example is a CA model of an elastic string that exhibits simple-harmonic motion and exactly emulates the continuum wave equation at integer times, averaged over pairs of adjacent sites. This model has been discussed before [7,23–25], but the analysis of overall translational motion, ideal energy, and their relativistic interpretation, have not been previously published.

2 Scalable Soft Sphere CA

Many CA dynamics can be interpreted as the integer-time behavior of a continuous classical mechanical system, started from an exactly specified initial state. This is true, for example, for lattice gas models of fluids. Such *stroboscopic* classical mechanical CA inherit, from their continuous counterparts, conserved quantities such as energy and momentum that we can compare to ideal quantities determined by local rates of state change. Of course the continuum models we have in mind would be numerically unstable if actually run as continuous dynamics, but this issue is not inherited by the finite-state CA [26].

A famous stroboscopic dynamics of this sort is Fredkin's billiard ball model of computation, in which hard spheres moving in a plane, each with four possible initial velocities, are restricted to a square lattice of initial positions. At each integer time, the system is again in such a configuration. To guarantee this property without additional restrictions on initial states, we let billiard balls pass through each other in some kinds of collisions, without interacting.

Figure 1 shows a variant of this model in which the balls are much more compressible, so collisions deflect paths inward rather than outward. This variant has the advantage that it is more directly related to a simple partitioning CA (*cf.* [6]). In the collision illustrated in Fig. 1a balls enter from the left with a

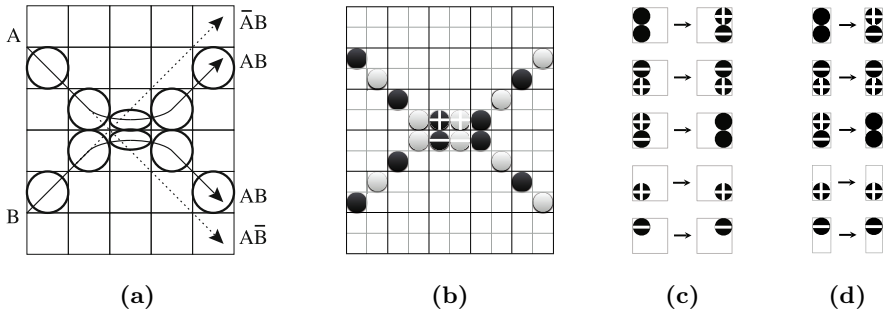


Fig. 1. Scalable Soft Sphere dynamics. (a) Stroboscopic view of continuous classical collision, one time step per column. (b) Finite state collision, with particles drawn lighter at odd time-steps. (c) 2D partitioning rule. Only these cases (and their rotations) interact. Otherwise, all particles move diagonally, unchanged. (d) 1D version of the rule.

horizontal component of velocity of one column per time unit, so consecutive moments of the history of a collision occur in consecutive columns.

The collision shown is energy and momentum conserving, and compression and rebound take exactly the time needed to displace the colliding balls from their original paths onto the paths labeled AB. If a ball had come in only at A with no ball at B, it would have left along the path labeled $A\bar{B}$: the collision acts as a universal logic gate.

Figure 1b shows a realization of the collision as a simple partitioning dynamics. Each time step in (a) corresponds to two in (b), and again particles are shown at each integer time—drawn dark at even times and light at odd. The rule (c) is inferred from (b), interpreting that diagram as showing the positions of two streams of colliding particles at one moment (dark), and their positions at the next moment (light). All particles move diagonally across a block, unchanged, in the cases not defined explicitly in (c). If this rule is applied to just the dark particles in each of the dark-bordered 2×2 blocks in (b), ignoring the light particles, it moves them to the light positions; applied to just the light particles in each of the light-bordered blocks, it moves them to the dark positions. The dynamics alternately applies the rule to these two partitions. To also allow collisions like (b) for streams of balls arriving from the right, top, or bottom, we define the rule (c) to have discrete rotational symmetry: in each of the cases shown in (c), each of the four 90° rotations of the pattern on the left turns into the corresponding rotation of the pattern on the right.

Note that (b) can also be interpreted as showing a time history of a collision of two particles in a one-dimensional partitioning dynamics (the center of mass dynamics). Then we get the rule (d), with the cases not explicitly shown interchanging the two cell values. Three dimensional versions of the dynamics can be constructed as in [21].

It is not surprising that a time-independent continuous dynamics turns into a time-dependent discrete partitioning. In the continuous model, balls approach

a locus of possible collision, interact independently of the rest of the system, and then move away toward a new set of loci. The partitions in the continuous case are just imaginary boxes we can draw around places within which what happens next doesn't depend on anything outside, for some period of time. Thus it is also not surprising that we can assign a conserved energy to partitioning dynamics.

2.1 Ideal Energy and Momentum

For a physical realization of the SSS dynamics, let τ be the time needed for gate operations to update all blocks of one partition, and let v_0 be the average speed at which the physical representation of a fastest-moving particle travels within the physical realization of the CA lattice (assuming discrete isotropy).

Equations (2) and (3) define an ideal (minimum) momentum and energy for a block in which there is a distinct overall spatial change and direction of motion. Clearly these ideal quantities are conserved overall in collisions, since freely moving particles move diagonally at v_0 before and after. Are they also conserved in detail during collisions?

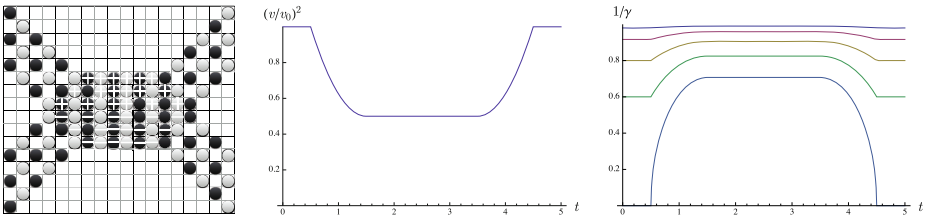
When two freely moving particles enter a single block in the collision of Fig. 1b, the number of block changes is reduced: one instead of two. The ideal magnitude of momentum for each freely moving particle before the collision is $p_1 = (h/2\tau)/v_0$. For two colliding particles moving horizontally within a block the ideal is $p_2 = (h/2\tau)/(v_0/\sqrt{2}) = 2p_1/\sqrt{2}$, which is the same as the net horizontal momentum before the collision. Ideal energy is similarly conserved.

Note, however, that the separate horizontal motions of the + and − particles during the next step of the collision of Fig. 1b imply an increase in the minimum energy and momentum for that step. This effect becomes negligible as we enlarge the scale of the objects colliding.

2.2 Rescaling the Collision

If two columns of k particles are collided in the SSS dynamics, then the resulting collision just shifts the output paths by k positions along the axis of the collision. This is illustrated in Fig. 2a for $k = 3$. Thus $k \times k$ blocks of particles collide exactly as in the classical collision of Fig. 1a: the SSS CA can perform logic with diagonally-moving square "balls" of any size. When two balls of equal size meet "squarely," moving together along a horizontal axis, each pair of columns evolves independently of the rest; colliding along a vertical axis, pairs of rows evolve independently. Square balls can participate in both kinds of collisions.

During such a collision, from the blocks that change we can infer a net momentum and hence a velocity for the motion of each colliding ball: Fig. 2b illustrates this for $k = 100$, with the time unit being the time for a freely moving $k \times k$ ball to travel its length (and width). Looking at just the changes in the top half of (a), we determine the magnitude and direction of minimum average momentum for each block that changes using (2), and hence determine a total momentum. Half of the conserved total energy is associated with each ball, so



(a) **(b)** **(c)**

Fig. 2. Multiple collisions in the SSS dynamics. (a) Colliding columns of particles are displaced horizontally by the height of the column. (b) Each column is slowed down by the collision. (c) The fraction of energy that is mass during a collision decreases with increasing initial particle speed v_0 (from top, 20% of c, 40%, 60%, 80%, 100%).

$v/v_0 = vE_{\text{ball}}/v_0E_{\text{ball}} = p/p_0$ gives the magnitude of velocity of a ball as a function of time, as number and type of changes evolve. This is plotted in (b).

The fraction $1/\gamma$ of the total energy E that is mass energy depends on $(v/c)^2 = (v/v_0)^2(v_0/c)^2$. Thus given (b), it depends on an assumption about the value of v_0/c. The fraction $1/\gamma$, as a function of time in the $k = 100$ collision, is plotted in Fig. 2c under different assumptions. The bottom case, $v_0 = c$, has the greatest range but the smallest value at all times. The top case is $v_0 = .2c$. As expected, the faster the speed of the fastest signals, the less the energy tied up in mass, hence the smaller the total energy. Ideally, $v_0 = c$.

3 Elastic String CA

In this second example we discuss a classical finite state model of wave motion in an elastic string. This stroboscopic classical mechanical model exactly reproduces the behavior of the time-independent one-dimensional wave equation sampled at integer-times and locations. As in the SSS example, a continuous model is turned into a finite state one by restricting the initial state (in this case the initial wave shape) to a perfect discrete set of possible initial configurations, and this constraint reappears at each integer time. In the continuum limit the discrete constraint on the wave shape disappears; the exactness of the wave dynamics itself (at discrete times) is independent of this limit.

The elastic string CA uses partitioning, but in a different way than the SSS CA: here the partitioning actually constrains the continuous classical dynamics used to define the CA, but in a way that never affects the classical energy. In the SSS case, the time dependence associated with the partitioning completely disappears in the continuous classical-mechanical version of the dynamics.

3.1 Discrete Wave Model

Consider an ideal continuous string for which transverse displacements exactly obey the wave equation. In Figure 3a we show an initial configuration with the

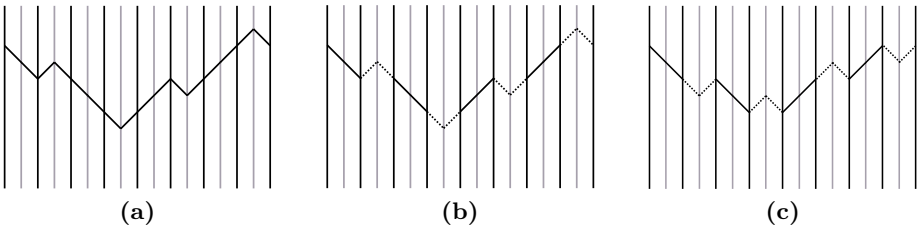


Fig. 3. Discrete wave dynamics. Elastic string is held fixed where it crosses black bars.

string stretched between equally spaced vertical bars. The set of initial configurations we're allowing are periodic, so the two endpoints must be at the same height.[1] Any configuration is allowed as long as each segment running between vertical bars is straight and lies at an angle of $\pm 45°$ to the horizontal.

Initially the string is attached at a fixed position wherever it crosses a vertical bar. We start the dynamics by releasing the attachment constraint at all of the gray bars. The attachment to the black bars remains fixed. In Figure 3b the segments that are about to move are shown with dotted lines: the straight segments have no tendency to move. Under continuum wave dynamics, the dotted segments all invert after some time interval τ. This will be our unit of time for the discrete dynamics. The new configuration at the end of this interval is shown in Figure 3c, with segments that have just moved dotted. At this instant in time all points of the string are again at rest and we are again in an allowed initial configuration. Now we interchange the roles of the black and gray bars and allow the segments between adjacent gray bars to move for a time interval τ. The dynamics proceeds like this, interchanging the roles of the black and gray bars after each interval of length τ. Since attachments are always changed at instants when all energy is potential and the string is not moving, the explicit time dependence of the system doesn't affect classical energy conservation.

We express this dynamics as a purely digital rule in Figure 4. In Figure 4a we show a wave with the black bars marking the attachments for the next step. To simplify the figure we have suppressed the gray bars—they are always situated midway between the black bars and so don't need to be shown. We have also added a grid of 45° dotted lines that shows all of the segments that the string could possibly follow. In Figure 4b we add in horizontal black bars, in order to partition the space into a set of 2×2 blocks that can be updated independently. Note that in all cases the segments that are allowed to change during this update step, as well as the cells that they will occupy after the update, are enclosed in a single block. The long box below Figure 4b contains just the slope information from the string. This array of gradients is clearly sufficient to recreate the wave pattern if the height at one position is known. This is not part of the 2D dynamics: it will be discussed as a related 1D dynamics.

[1] Unless the right and left edges of the space itself are joined with a vertical offset.

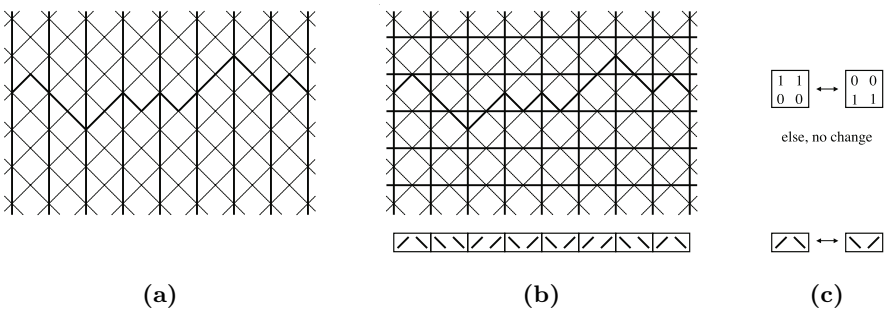


Fig. 4. Discrete wave dynamics. (a) A wave configuration. Possible wave paths are indicated by dotted lines. (b) Horizontal and vertical lines indicate one of two partitions used for discrete update rule. A 1D array summarizing wave gradients is shown below (not part of the 2D dynamics). (c) Top, dynamical rule for 2D wave. Presence of wave-path segments is indicated by 1's. Bottom, equivalent 1D dynamical rule for gradients.

Figure 4c shows the dynamical rule for a block. Since the dotted lines indicate the direction in which segments must run if they appear in any cell, the state information for each segment is only whether it is there or not: this is indicated with a 1 or a 0. The only segments that change are peaks /\ or valleys \/, and these are represented by two 1's at the top of a block or at the bottom of a block respectively. The rule is that peaks and valleys turn into each other, and nothing else changes. We apply the rule alternately to the blocks shown, and to a complementary partition shifted half a block horizontally and vertically.

3.2 Exact Wave Behavior

At the bottom of Figure 4c we've presented a dynamics for the *gradients* of the wave. The full 2D dynamics just turns peaks into valleys and vice versa, leaving straight segments unchanged: we can do that equally well on the array of gradients. As the 2D dynamics interchanges which blocking to use, the dynamics on the gradients also alternates which pairs of gradients to update together. In all cases, the dynamics on the gradients duplicates what happens on the string: if the two dynamics are both performed in parallel, the gradient listed below a column will always match the slope of the string in that column.

The dynamics on the gradients has an interesting property. Turning a peak into a valley and vice versa is exactly the same as swapping the left and right elements of a block. Leaving a // or \\ unchanged is also exactly the same as swapping the left and right elements of a block. In all cases, the dynamics on the gradients is equivalent to a swap.

This means that the left element of a block will get swapped into the right position, and at the next update it will be the left element of a new block and will again get swapped into the right position, and so on. Thus all of the gradients that start off in the left side of a block will travel uniformly to the right, and all that start in the right side of a block will travel uniformly to the left.

This shows that the system obeys a discrete version of the wave equation. Half of the gradients constitute a right-going wave, and half constitute a left-going wave. At any step of the dynamics, the 2D wave in the original dynamics is just the sum of the two waves: it is reproduced by laying gradients end to end.

If we refine the lattice, using more and more cells to represent a wave of given width, smoother and smoother waves can be represented. Of course even without going to a large-scale limit, the CA dynamics is already exactly equivalent to a continuous wave equation with constrained initial wave shapes, sampled at integer times: simply stretch the rightgoing and leftgoing waves constructed out of gradients to the full width of the lattice. This just amounts to drawing the wave shape corresponding to each block of the current partition a little differently.

3.3 Overall Transverse Motion

Assume the string carrying a discrete wave wraps around the space. We've discussed the horizontal motion of waves along such a string, but the string itself can move vertically. For example, a pattern such as \/\/\/...\/ all the way around the space reproduces itself after two partition update steps, but shifted vertically by two lattice units. This is clearly the maximum rate of travel for a string: one position vertically per update step. Call this v_0.

We can express the net velocity of the string in terms of the populations of rightgoing and leftgoing gradient segments. Let R_+ be the number of rightgoing segments with slope +1 (rightgoing /'s), and similarly for R_-, L_+ and L_-. If the width of the space is B blocks, then there are $B = R_+ + R_-$ segments forming the rightgoing wave, and $B = L_+ + L_-$ forming the leftgoing one.

For the rightgoing or leftgoing wave, periodically repeating its sequence of gradients corresponds to an unbounded wave with the same average slope. When both waves have shifted horizontally the width of one period (after $2B$ partition update steps), the net vertical shift is the sum of the slopes of the leftgoing gradients, minus the sum for the rightgoing ones: $(L_+ - L_-) - (R_+ - R_-)$. We can compute this by summing the differences for each pair of slopes grouped together in the columns of one partition. Only columns containing \/ or /\ contribute a non-zero difference, and so we only need to count the numbers of blocks $B_{\backslash/}$ and $B_{/\backslash}$ that are about to change, to compute the constant velocity

$$\frac{v}{v_0} = \frac{(L_+ - L_-) - (R_+ - R_-)}{2B} = \frac{B_{\backslash/} - B_{/\backslash}}{B} \,. \tag{4}$$

3.4 Ideal Energy and Momentum

Only blocks that change have overall motion, and with τ the time taken to update one partition, the frequencies of positive and negative motion are $B_{\backslash/}/\tau$ and $B_{/\backslash}/\tau$. Thus from (2), attributing a momentum to each changing block, the total ideal momentum up is $hB_{\backslash/}/2\tau v_0$, and down is $hB_{/\backslash}/2\tau v_0$, so the net ideal momentum $p = (h/2\tau v_0)(B_{\backslash/} - B_{/\backslash})$. From (4), the corresponding relativistic

energy is $E = c^2 p/v = (hB/2\tau)/(v_0/c)^2$. Letting $v_0 \to c$ to minimize energy, and choosing units with $h = 2$ and $c = 1$ and $\tau = 1$, this becomes

$$E = B \quad \text{and} \quad p = B_{\backslash/} - B_{/\backslash}\,. \tag{5}$$

Energy is the constant width (in blocks) of the string, and momentum is the constant net number of blocks moving up.

There is an interesting subtlety involved in letting $v_0 \to c$ in the 2D dynamics. We interpret all gradient segments as always moving, swapping in pairs in each update in order to recover the wave equation—even though some paired segments are in different blocks when they "swap" identical values. If all block motion forward or backward is at the speed c, each segment must be interpreted as traveling at the speed $c\sqrt{2}$ as it swaps diagonally. If instead we interpret segments as moving up and down (or not moving), none travel faster than light, but the interaction is non-local at the scale of an individual block.

3.5 Rest Frame Energy

For the transverse motion of the string to approach the maximum speed, almost all of the block updates must contribute to overall motion, and almost none to just internally changing the string. This slowdown of internal dynamics is a kind of time dilation, which is reflected in the rest frame energy $\sqrt{E^2 - p^2}$. From (5),

$$E_r = \sqrt{B^2 - (B_{\backslash/} - B_{/\backslash})^2}\,. \tag{6}$$

The energy E_r available for rest-frame state-change decreases as more blocks move in the same direction. In this model total energy E is independent of v, hence rest energy $E_r = E/\gamma$ must approach 0 as $1/\gamma \to 0$. This contrasts with a normal relativistic system that can never attain the speed of light, which has a constant rest energy E_r and a total energy E that changes with v.

The analysis up to here applies equally well to both the 1D and 2D versions of the dynamics of Fig. 4. In 2D, however, there is an additional constraint: there must be an equal number B of positive and negative slopes, so that the string meets itself at the periodic boundary. Since there are also an equal number B of right and left going gradients, $R_+ = L_-$ and $R_- = L_+$. Thus from (4) and (6),

$$E_r = 2\sqrt{R_+ L_+}\,. \tag{7}$$

If R_+/B were the probability for a walker to take a step to the right, and L_+/B the probability to the left, then (7) would be the standard deviation for a $2B$-step random walk. Related models of diffusive behavior that make contact with relativity are discussed in [24,27,28]. None of these define relativistic objects that have an internal dynamics, however.

4 Discussion

Given the definition of a finite-state dynamics, we could try to assign intrinsic properties to it based on the best possible implementation. For example, programming it on an ordinary computer, a basic property is the minimum time

needed, on average, to simulate a step of the dynamics. It would be hard, though, to be sure we've found the most efficient mapping onto the computer's architecture, and the minimum time would change if we used a different computer, or built custom hardware using various technologies. The true minimum time would correspond to the fastest possible implementation allowed by nature! Such a definition seems vacuous, though, since we don't know the ultimate laws of nature, and even if we did, how would we find the best possible way to use them?

Surprisingly, a fundamental-physics based definition of intrinsic properties is not in fact vacuous, if we base it on general principles. Assuming the universe is fundamentally quantum mechanical, we couldn't do better than to simply *define* a hamiltonian that exactly implements the classical finite-state dynamics desired at discrete times, with no extra distinct states or distinct state change. This ideal hamiltonian identifies the fastest implementation that is *mathematically possible*, with given average energy.

This procedure assigns to every invertible finite-state dynamics an ideal energy that depends only on the average rate of distinct state change. This is generally not much like a physical energy, though, since we haven't yet included any realistic constraints on the dynamics. For example, each state change might correspond to a complete update of an entire spatial lattice, as in the synchronous definition of a CA. Then the energy would be independent of the size of the system. We can fix this by constraining the finite-state dynamics to be local and not *require* synchrony: defining it in terms of gates that are applied independently.

We expect the ideal energy, and distinct portions of it, to become more realistic with additional realistic constraints. For this reason, we studied invertible lattice dynamics derived from the integer-time behavior of idealized classical mechanical systems. In the examples we looked at, ideal energies and momenta defined by local rates of state change evolve like classical relativistic quantities.

It seems interesting and novel to introduce intrinsic definitions of energy and other physical quantities into classical finite-state systems, and to use these definitions in constructing and analyzing finite-state models of physical dynamics. Since all finite-energy systems in the classical world actually have finite state, and since classical mechanics doesn't, this may be a productive line of inquiry for better modeling and understanding that world. Moreover, inasmuch as all physical dynamics can be regarded as finite-dimensional quantum computation, finite-state models of classical mechanics may play the role of ordinary computation in understanding the more general quantum case.

Acknowledgments. I thank Micah Brodsky and Gerald Sussman for helpful discussions.

References

1. Bekenstein, J.D.: Universal upper bound on the entropy-to-energy ratio for bounded systems. Phys. Rev. D **23**, 287 (1981)
2. Margolus, N., Levitin, L.B.: The maximum speed of dynamical evolution. Physica D **120**, 188 (1998)

3. Margolus, N.: The maximum average rate of state change. arXiv:1109.4994
4. Planck, M.: On the law of distribution of energy in the normal spectrum. Ann. Phys. (Berlin) **309**, 553 (1901)
5. Margolus, N.: Quantum emulation of classical dynamics. arXiv:1109.4995
6. Margolus, N.: Physics like models of computation. Physica D **10**, 81 (1984)
7. Margolus, N.: Crystalline computation. In: Hey, A. (ed.) Feynman and Computation, p. 267. Perseus Books (1998). arXiv:comp-gas/9811002
8. Toffoli, T., Margolus, N.: Cellular automata machines: a new environment for modeling. MIT Press (1987)
9. Chopard, B., Droz, M.: Cellular Automata Modeling of Physical Systems. Cambridge University Press (2005)
10. Rothman, D., Zaleski, S.: Lattice Gas Cellular Automata: Simple Models of Complex Hydrodynamics. Cambridge University Press (2004)
11. Rivet, J.P., Boon, J.P.: Lattice Gas Hydrodynamics. Cambridge University Press (2005)
12. Fredkin, E.: A computing architecture for physics. In: CF 2005 Proceedings of the 2nd Conference on Computing Frontiers, p. 273. ACM (2005)
13. Wolfram, S.: A new kind of science. Wolfram Media (2002)
14. Toffoli, T., Margolus, N.: Invertible cellular automata: a review. Physica D **45**, 229 (1990)
15. Kari, J.: Representation of reversible cellular automata with block permutations. Math. Systems Theory **29**, 47 (1996)
16. Durand-Lose, J.: Representing reversible cellular automata with reversible block cellular automata. Discrete Math. Theor. Comp. Sci. Proc. AA, 145 (2001)
17. Ulam, S.: Random processes and transformations. In: Proceedings of the International Congress on Mathematics, 1950, vol. 2, p. 264 (1952)
18. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press (1966)
19. Zuse, K.: Calculating Space. MIT Tech. Transl. AZT-70-164-GEMIT (1970)
20. Margolus, N.: Mechanical Systems that are both Classical and Quantum. arXiv:0805.3357
21. Margolus, N.: Universal cellular automata based on the collisions of soft spheres. In: Griffeath, D., Moore, C. (eds.) New Constructions in Cellular Automata, p. 231. Oxford University Press (2003). arXiv:0806.0127
22. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21**, 219 (1982)
23. Hrgovčić, H.: Discrete representations of the n-dimensional wave equation. J. Phys. A: Math. Gen. **25**, 1329 (1992)
24. Toffoli, T.: Action, or the fungibility of computation. In: Hey, A. (ed.) Feynman and Computation, p. 349. Perseus Books (1998)
25. Margolus, N.: Physics and computation. Massachusetts Institute of Technology Ph.D. Thesis (1987)
26. Toffoli, T.: Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. Physica D **10**, 117 (1984)
27. Smith, M.: Representation of geometrical and topological quantities in cellular automata. Physica D **45**, 271 (1990)
28. Ben-Abraham, S.I.: Curious properties of simple random walks. J. Stat. Phys. **73**, 441 (1993)

On the Periods of Spatially Periodic Preimages in Linear Bipermutive Cellular Automata

Luca Mariot and Alberto Leporati(✉)

Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi Milano - Bicocca, Viale Sarca 336/14, 20126 Milano, Italy
l.mariot@campus.unimib.it, alberto.leporati@unimib.it

Abstract. In this paper, we investigate the periods of preimages of spatially periodic configurations in linear bipermutive cellular automata (LBCA). We first show that when the CA is only bipermutive and y is a spatially periodic configuration of period p, the periods of all preimages of y are multiples of p. We then present a connection between preimages of spatially periodic configurations of LBCA and concatenated linear recurring sequences, finding a characteristic polynomial for the latter which depends on the local rule and on the configurations. We finally devise a procedure to compute the period of a single preimage of a spatially periodic configuration y of a given LBCA, and characterise the periods of all preimages of y when the corresponding characteristic polynomial is the product of two distinct irreducible polynomials.

Keywords: Linear bipermutive cellular automata · Spatially periodic configurations · Preimages · Surjectivity · Linear recurring sequences · Linear feedback shift registers

1 Introduction

It is known that if $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ is a surjective cellular automaton (CA) and $y \in A^{\mathbb{Z}}$ is a spatially periodic configuration, then all preimages $x \in F^{-1}(y)$ are spatially periodic as well [2]. However, to our knowledge there are no works in the literature addressing the problem of actually finding the periods of such preimages.

The aim of this paper is to study the relation between the periods of spatially periodic configurations and the periods of their preimages in the case of *linear bipermutive cellular automata* (LBCA). Given a spatially periodic configuration $y \in A^{\mathbb{Z}}$ of period p, we first prove that in generic bipermutive cellular automata (BCA) the period of a preimage $x \in F^{-1}(y)$ is a multiple of p, where the multiplier h ranges in $\{1, \cdots, q^{2r}\}$, with q being the size of the alphabet and r the radius of the BCA. We then show that, in the case of LBCA, a preimage $x \in F^{-1}(y)$ can be described as a *concatenated linear recurring sequence* (LRS) whose characteristic polynomial is the product of the characteristic polynomials respectively induced by the local rule f of the CA and by configuration y. Finally, we present a

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 181–195, 2015.
DOI: 10.1007/978-3-662-47221-7_14

procedure which given a block $x_{[0,2r-1]}$ of a preimage $x \in F^{-1}(y)$ determines the period of x, and we characterise the periods of all q^{2r} preimages of y when their characteristic polynomial is the product of two irreducible polynomials.

This research was inspired from the problem of determining the maximum number of players allowed in a BCA-based secret sharing scheme presented in [10].

The rest of this paper is organised as follows. Section 2 recalls some basic definitions and facts about cellular automata, linear recurring sequences and linear feedback shift registers. Section 3 shows that the periods of spatially periodic preimages are multiples of the periods of their respective images, and characterises preimages of LBCA as concatenated linear recurring sequences. Section 4 focuses on the characteristic polynomial of concatenated LRS, while Section 5 presents an algorithm to compute the period of a single LBCA preimage and characterises the periods of all preimages of a spatially periodic configuration y in the particular case of irreducible characteristic polynomials. Finally, Section 6 summarises the results presented throughout the paper and points out some possible future developments on the subject.

2 Basic Definitions

2.1 Cellular Automata

Let A be a finite alphabet having q symbols, and let $A^{\mathbb{Z}}$ be the *full shift space* consisting of all biinfinite configurations over A. Given $x \in A^{\mathbb{Z}}$ and $i, j \in \mathbb{Z}$ with $i \leq j$, by $x_{[i,j]}$ we denote the finite block $(x_i, \cdots, x_j)$. In what follows, we focus our attention on *one-dimensional cellular automata*, formally defined below:

Definition 1. *A* one-dimensional cellular automaton *is a function $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ defined for all $x \in A^{\mathbb{Z}}$ and $i \in \mathbb{Z}$ as:*

$$F(x)_i = f(x_{[i-r,i+r]}) \ ,$$

where $f : A^{2r+1} \to A$ is the local rule *of the CA and $r \in \mathbb{N}$ is its* radius.

From a dynamical point of view, a CA can be considered as a biinfinite array of *cells* where, at each time step $t \in \mathbb{N}$, all cells $i \in \mathbb{Z}$ simultaneously change their *state* $s_i \in A$ by applying the local rule f on the *neighbourhood* $\{i-r, \cdots, i+r\}$.

The main class of CA studied in this paper consists of *bipermutive* CA, defined as follows:

Definition 2. *A CA $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ induced by a local rule $f : A^{2r+1} \to A$ is called* left permutive *(respectively,* right permutive*) if, for all $z \in A^{2r}$, the restriction $f_{R,z} : A \to A$ (respectively, $f_{L,z} : A \to A$) obtained by fixing the first (respectively, the last) $2r$ coordinates of f to the values specified in z is a permutation on A. A CA which is both left and right permutive is said to be a* bipermutive cellular automaton *(BCA).*

Another class of CA which can be defined by endowing the alphabet with a group structure is that of *linear* (or *additive*) cellular automata. We give the definition for the particular case in which A is a finite field. Thus, we have $A = \mathbb{F}_q$ with $q = \rho^\alpha$, where $\rho \in \mathbb{N}$ is a prime number (called the *characteristic* of $\mathbb{F}_q$) and $\alpha \in \mathbb{N}$.

Definition 3. *A CA $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ with local rule $f : \mathbb{F}_q^{2r+1} \to \mathbb{F}_q$ is* linear *if there exists $(c_0, \cdots, c_{2r}) \in \mathbb{F}_q^{2r+1}$ such that f can be defined for all $(x_0, \cdots, x_{2r}) \in \mathbb{F}_q^{2r+1}$ as:*

$$f(x_0, \cdots, x_{2r}) = c_0 \cdot x_0 + \cdots + c_{2r} \cdot x_{2r} \ ,$$

where $+$ and $\cdot$ respectively denote sum and product over $\mathbb{F}_q$.

One easily checks that if both c_0 and c_{2r} in Definition 3 are nonzero then a linear CA is bipermutive as well. Most of the results proved in this paper concern cellular automata which are both linear and bipermutive.

A configuration $x \in A^{\mathbb{Z}}$ is called *spatially periodic* if there exists $p \in \mathbb{N}$ such that $x_{n+p} = x_n$ for all $n \in \mathbb{Z}$, and the least p for which this equation holds is called the *period* of x. In this case, x is generated by the *biinfinite concatenation* of a string $u \in A^p$ with itself, denoted by ${}^{\omega}u^{\omega}$. A proof of the following result about preimages of spatially periodic configurations in surjective CA can be found in [2].

Lemma 1. *Let $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be a surjective CA. Then, given a spatially periodic configuration $y \in A^{\mathbb{Z}}$, each preimage $x \in F^{-1}(y)$ is also spatially periodic.*

This lemma is a consequence of a theorem proved by Hedlund [7], which states that every configuration $x \in A^{\mathbb{Z}}$ has a finite number of preimages under a surjective CA. In the same work, Hedlund showed that bipermutive CA are also surjective. Indeed, given a BCA $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ induced by a local rule $f : A^{2r+1} \to A$ and a configuration $y \in A^{\mathbb{Z}}$, a preimage $x \in F^{-1}(y)$ is determined by first setting in x a block of $2r$ cells $x_{[i,i+2r-1]} \in A^{2r}$, with $i \in \mathbb{Z}$. Then, denoting by $f_{R,z}^{-1} : A \to A$ and $f_{L,z}^{-1} : A \to A$ the inverses of the permutations obtained by respectively fixing the first and the last $2r$ coordinates of f to $z \in \mathbb{A}^{2r}$, for all $n \geq i + 2r$ and $n < i$ the value of x_n is determined through the following recurrence equation:

$$x_n = \begin{cases} f_{R,z(n)}^{-1}(y_{n-r}), \text{ where } z(n) = x_{[n-2r,n-1]}, & \text{if } n \geq i + 2r \quad \text{(a)} \\ f_{L,z(n)}^{-1}(y_{n+r}), \text{ where } z(n) = x_{[n+1,n+2r]}, & \text{if } n < i \quad \text{(b)} \end{cases} \tag{1}$$

As a consequence, by Lemma 1 the preimages of spatially periodic configurations under a BCA are spatially periodic as well. Moreover, since a preimage of y is uniquely determined by a $2r$-cell block using Equation (1), it follows that y has exactly q^{2r} possible preimages in $F^{-1}(y)$.

We now formally state the problem analysed in the remainder of this paper:

Problem. Let $y \in A^{\mathbb{Z}}$ be a spatially periodic configuration of period $p \in \mathbb{N}$. Given a BCA $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$, find the relation between p and the spatial periods of the preimages $x \in F^{-1}(y)$.

2.2 Linear Recurring Sequences and Linear Feedback Shift Registers

We now recall some basic definitions and results about the theory of linear recurring sequences and linear feedback shift registers, which will be useful to characterise the periods of preimages in LBCA. All the proofs of the theorems mentioned in this section may be found in the book by Lidl and Niederreiter [9].

Definition 4. *Given $k \in \mathbb{N}$ and $a, a_0, a_1, \cdots, a_{k-1} \in \mathbb{F}_q$, a* linear recurring sequence *(LRS) of order k is a sequence $s = s_0, s_1, \cdots$ of elements in $\mathbb{F}_q$ which satisfies the following relation:*

$$s_{n+k} = a + a_0 s_n + a_1 s_{n+1} + \cdots + a_{k-1} s_{n+k-1} \quad \forall n \in \mathbb{N} \ . \tag{2}$$

The terms $s_0, s_1, \cdots, s_{k-1}$ which uniquely determine the rest of the LRS are called the *initial values* of the sequence. If $a = 0$ the sequence is called *homogeneous*, otherwise it is called *inhomogeneous*. In what follows, we will only deal with homogeneous LRS.

A linear recurring sequence can be generated by a device called *linear feedback shift register* (LFSR), depicted in Figure 1. Basically, a LFSR of order k is

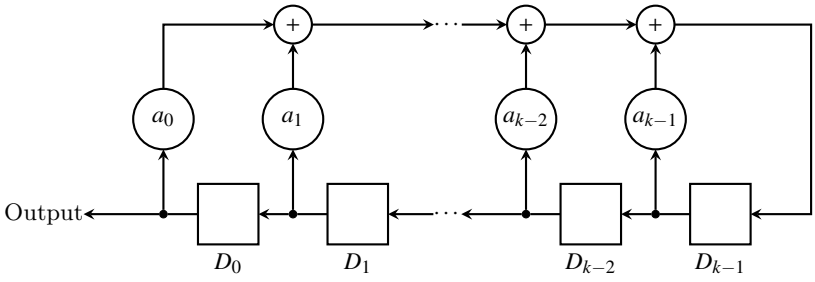


Fig. 1. Diagram of a linear feedback shift register of length k

composed of k *delayed flip-flops* $D_0, D_1, \cdots, D_{k-1}$, each containing an element of $\mathbb{F}_q$. At each time step $n \in \mathbb{N}$, the elements $s_n, s_{n+1}, \cdots, s_{n+k-1}$ in the flip-flops are shifted one place to the left, and D_{k-1} is updated by the linear combination $a_0 \cdot s_n + \cdots + a_{k-1} \cdot s_{n+k-1}$, which corresponds to the linear recurrence defined in Equation (2).

It is straightforward to observe that the output produced by the LFSR (that is, the LRS $s = s_0, s_1, \cdots$) must be *ultimately periodic*, that is, there exist $p, n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $s_{n+p} = s_n$. In fact, for all $n \in \mathbb{N}$ the state of the LFSR is completely described by the vector $(s_n, s_{n+1}, \cdots, s_{n+k-1})$. Since all the components of such vector take values in $\mathbb{F}_q$, which is a finite set of q elements, after at most q^k shifts the initial value of the vector will be repeated. In particular, in [9] it is proved that if $a_0 \neq 0$, then the sequence produced by the LFSR (or, equivalently,

the corresponding LRS) is *periodic*, i.e., it is ultimately periodic with preperiod $n_0 = 0$.

An important parameter of a k-th order homogeneous LRS $s = s_0, s_1, \cdots$ is its *characteristic polynomial* $a(x) \in \mathbb{F}_q[x]$, defined as:

$$a(x) = x^k - a_{k-1}x^{k-1} - a_{k-2}x^{k-2} - \cdots - a_0 \ . \tag{3}$$

The *multiplicative order* of the characteristic polynomial, denoted by $ord(a(x))$, is the least integer e such that $a(x)$ divides $x^e - 1$, and it can be used to characterise the period of s. In fact, in [9] it is shown that if $a(x)$ is irreducible over $\mathbb{F}_q$ and $a(0) \neq 0$, then the period p of s equals $ord(a(x))$, while in the general case where $a(x)$ is reducible $ord(a(x))$ divides p.

A common way of representing a LRS $s = s_0, s_1, \cdots$ is by means of its *generating function* $G(x)$, which is the formal power series defined as:

$$G(x) = s_0 + s_1 x + s_2 x^2 + \cdots = \sum_{n=0}^{\infty} s_n x^n \tag{4}$$

In this case, the terms $s_0, s_1, \cdots$ are called the *coefficients* of $G(x)$. The set of all generating functions over $\mathbb{F}_q$ can be endowed with a ring structure in which sum and product are respectively pointwise addition and convolution of coefficients. The *fundamental identity of formal power series* states that the generating function $G(x)$ of a k-th order homogeneous LRS s can be expressed as a rational function:

$$G(x) = \frac{g(x)}{a^*(x)} = \frac{-\sum_{j=0}^{k-1} \sum_{i=0}^{j} a_{i+k-j} s_i x^j}{x^k a(1/x)} \ . \tag{5}$$

where $g(x)$ is the *initialisation polynomial*, which depends on the k initial terms of sequence s (in which we set $a_k = -1$), while $a^*(x) = x^k a(1/x)$ is the *reciprocal characteristic polynomial* of s.

It is easy to see that a given LRS $s = s_0, s_1, \cdots$ over $\mathbb{F}_q$ satisfies several linear recurrence equations. Hence, several characteristic polynomials can be associated to s, one for each recurrence equation which s satisfies. The *minimal polynomial* $m(x)$ associated to s is the characteristic polynomial which divides all other characteristic polynomials of s, and it can be computed as follows:

$$m(x) = \frac{a(x)}{\gcd(a(x), h(x))} \ , \tag{6}$$

where $a(x)$ is a characteristic polynomial of s and $h(x) = -g^*(x)$ is the reciprocal of the initialisation polynomial $g(x)$ appearing in Equation (5), with the sign changed. In [9] it is proved that the period of s equals the order of its minimal polynomial $m(x)$.

In order to study the periods of preimages of LBCA, we also need some results about the sum of linear recurring sequences. Let $s = s_0, s_1, \cdots$ and $t = t_0, t_1, \cdots$ be homogeneous LRS over $\mathbb{F}_q$. The *sum sequence* $\sigma = s + t$ is defined as $\sigma_n = s_n + t_n$, for all $n \in \mathbb{N}$.

Theorem 1. *Let σ_1 and σ_2 be two homogeneous LRS having minimal polynomials $m_1(x), m_2(x) \in \mathbb{F}_q[x]$ and periods $p_1, p_2 \in \mathbb{N}$, respectively. If $m_1(x)$ and $m_2(x)$ are relatively prime, then the minimal polynomial $m(x) \in \mathbb{F}_q[x]$ of the sum $\sigma = s + t$ is equal to $m_1(x) \cdot m_2(x)$, while the period of σ is the least common multiple of p_1 and p_2.*

The following theorem gives a characterisation of the periods of LRS associated to an irreducible characteristic polynomial.

Theorem 2. *Let $S(a(x))$ be the set of all homogeneous linear recurring sequences over $\mathbb{F}_q$ with irreducible characteristic polynomial $a(x) \in \mathbb{F}_q[x]$, and let e be the multiplicative order of $a(x)$. Then, $S(a(x))$ contains one sequence of period 1 and $q^k - 1$ sequences of period e.*

3 Preliminary Results

3.1 Preimages Periods in Generic BCA

We begin our analysis of Problem 2.1 by considering the general case where only bipermutivity holds. To this end, we first show a relation between finite blocks in the preimages of BCA.

Lemma 2. *Let $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be a BCA with local rule $f : A^{2r+1} \to A$. Then, given a configuration $y \in A^{\mathbb{Z}}$ and $i, j \in \mathbb{Z}$, for all $x \in F^{-1}(y)$ there exists a permutation between the blocks $x_{[i,i+2r-1]}$ and $x_{[j,j+2r-1]}$.*

Proof. Without loss of generality, let us assume $i < j$. Since y is fixed and F is bipermutive, for all $x_{[i,i+2r-1]} \in A^{2r}$ define $\varphi_y : A^{2r} \to A^{2r}$ as $\varphi_y(x_{[i,i+2r-1]}) = x_{[j,j+2r-1]}$, where for each $n \in \{j, \cdots, j+2r-1\}$ the value of x_n is computed by applying case (a) of Equation (1). We have to show that φ_y is a permutation on A^{2r} (Figure 2).

For all possible values of block $x_{[j,j+2r-1]}$, the value of $x_{[i,i+2r-1]}$ is uniquely determined by applying case (b) of Equation (1). As a consequence, under φ_y each image has a unique preimage, and thus φ_y is bijective. □

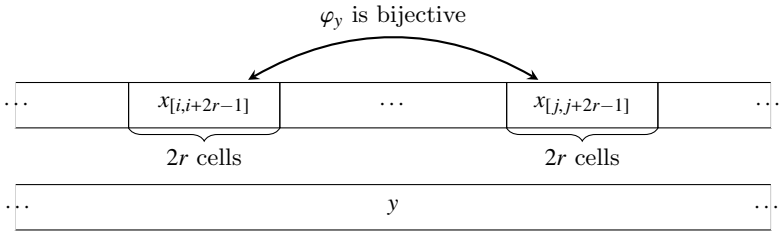


Fig. 2. By fixing y, function φ_y is a A^{2r}-permutation

Using Lemma 2, the following useful information about the periods of spatially periodic preimages in BCA can be deduced:

Proposition 1. *Let $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be a BCA with local rule $f : A^{2r+1} \to A$ and let $y \in A^{\mathbb{Z}}$ be a spatially periodic configuration of period $p \in \mathbb{N}$. Given a preimage $x \in F^{-1}(y)$, the period $m \in \mathbb{N}$ of x is a multiple of p. In particular, it holds that $m = p \cdot h$, where $h \in \{1, \cdots, q^{2r}\}$.*

Proof. Since y is spatially periodic of period p, we have that $y = {}^{\omega}u^{\omega}$ for a certain $u \in A^p$. Given a preimage $x \in F^{-1}(y)$, denote by $w_1 \in A^{2r}$ the block $x_{[i-r,i+r-1]}$, where $i \in \mathbb{Z}$ is such that $y_i = y_{i+p} = u_1$. In other words, w_1 is a $2r$-cell block of x placed across the boundary between two copies of u in y (see Figure 3). By Lemma 2 we know that block u fixes a permutation $\varphi_u : A^{2r} \to A^{2r}$ which maps block w_1 to $w_2 = x_{[i+p-r,i+p+r-1]}$. More in general, observe that for all $j \geq 2$ the permutation which associates block $w_j = x_{[i+pj-r,i+pj+r-1]}$ to $w_{j+1} = x_{[i+p(j+1)-r,i+p(j+1)+r-1]}$ is always φ_u, the reason being that the block below w_j and w_{j+1} is a repetition of u. Since $|A| = q$, the permutation φ_u can be composed by at most one cycle of length q^{2r}. This means that, after at most $h \leq q^{2r}$ applications of φ_u, block $w_h = x_{[i+ph-r,i+ph+r-1]}$ will be equal to w_1, and from then on the preimage will periodically repeat itself. Thus, it results that $x_n = x_{n+ph}$ for all $n \in \mathbb{Z}$, from which we deduce that the period of x is $p \cdot h$. □

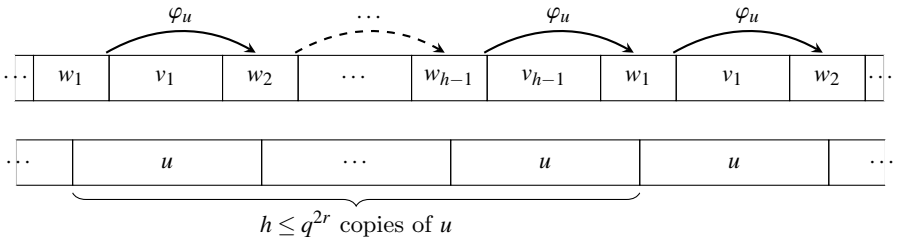


Fig. 3. After at most $h \leq q^{2r}$ applications of φ_u, the $2r$-cell block w_1 will be repeated. At this point, the subsequent p-cell block in the preimage will be a copy of $v_1 w_2$

3.2 Characterising LBCA Preimages by LRS Concatenation

Proposition 1 limits the possible values of the periods attained by preimages of spatially periodic configurations in BCA. In what follows we show that, by narrowing the analysis to the class of LBCA, further information about the periods of preimages can be obtained.

Let $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ be a LBCA of radius r with local rule $f : \mathbb{F}_q^{2r+1} \to \mathbb{F}_q$ defined by a vector $(c_0, \cdots, c_{2r}) \in \mathbb{F}_q^{2r+1}$, where $c_0 \neq 0$ and $c_{2r} \neq 0$. Given $x \in \mathbb{F}_q^{2r+1}$ and $y = f(x)$, the following equalities hold:

$$y = c_0 x_0 + c_1 x_1 + \cdots + c_{2r-1} x_{2r-1} + c_{2r} x_{2r}$$

$$x_{2r} = c_{2r}^{-1}(-c_0 x_0 - c_1 x_1 - \cdots - c_{2r-1} x_{2r-1} + y) \ .$$

Setting $d = c_{2r}^{-1}$ and $a_i = -d \cdot c_i$ for all $i \in \{0, \cdots, 2r-1\}$, we obtain

$$x_{2r} = a_0 x_0 + a_1 x_1 + \cdots + a_{2r-1} x_{2r-1} + dy \ . \tag{7}$$

Equation (7) defines the inverse $f_{R,z}^{-1}$ of the permutation $f_{R,z} : \mathbb{F}_q \to \mathbb{F}_q$ obtained by fixing the first $2r$ coordinates of f to the values of $z = (x_0, \cdots, x_{2r-1})$. Hence, given a configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$ and the $2r$-cell block $x_{[0,2r-1]} \in \mathbb{F}_q^{2r}$ in a preimage $x \in F^{-1}(y)$, case (a) of Equation (1) yields

$$x_n = a_0 x_{n-2r} + a_1 x_{n-2r+1} + \cdots + a_{2r-1} x_{n-1} + dy_{n-r} \quad \forall n \geq 2r \ , \tag{8}$$

and by setting $k = 2r$ and $v_n = y_{n+r}$ for all $n \in \mathbb{N}$, Equation (8) can be rewritten as

$$x_{n+k} = a_0 x_n + a_1 x_{n+1} + \cdots + a_{k-1} x_{n+k-1} + dv_n \quad \forall n \geq 2r \ . \tag{9}$$

Equation (9) reminds the definition of a linear recurring sequence of order $k = 2r$, with the exception of term dv_n. However, if y is a spatially periodic configuration of period p then it is possible to describe the sequence $v = v_0, v_1, \cdots$ as a linear recurring sequence of order $l \leq p$ defined by

$$v_{n+l} = b_0 v_n + b_1 v_{n+1} + \cdots + b_{l-1} v_{n+l-1} \ , \tag{10}$$

where $b_i \in \mathbb{F}_q$ for all $i \in \{0, \cdots, l-1\}$, and the initial terms of the sequence are $v_0 = y_r$, $v_1 = y_{r+1}$, $\cdots$, $v_{l-1} = y_{r+l-1}$. In the worst case, the LRS v will have order $l = p$, and it will be generated by the trivial LFSR which cyclically shifts a word of length p.

As a consequence, preimage $x \in F^{-1}(y)$ is a linear recurring sequence of a special kind, where x_{n+k} is determined not only by the previous $k = 2r$ terms, but it is also "disturbed" by the LRS v. In particular, we define x as the *concatenation* of sequences s and v, which we denote by $s \leftsquigarrow v$, where $s = s_0, s_1, \cdots$ is the k-th order LRS satisfying the recurrence equation

$$s_{n+k} = a_0 s_n + a_1 s_{n+1} + \cdots + a_{k-1} s_{n+k-1} \ , \tag{11}$$

and whose initial values are $s_0 = x_0$, $s_1 = x_1$, $\cdots$, $s_{k-1} = x_{k-1}$.

Equivalently, a preimage $x \in F^{-1}(y)$ is generated by a LFSR of order $k = 2r$ where the feedback is summed with the output of an l-th order LFSR multiplied by $d = c_{2r}^{-1}$, which produces sequence v. Similarly to concatenated LRS, we call this system a *concatenation* of LFSR. Figure 4 depicts the block diagram of this concatenation.

In conclusion, we have shown that the periods of the preimages $x \in F^{-1}(y)$ are equivalent to the periods of the concatenated LRS generated by the LFSR in Figure 4, where the disturbing LFSR is initialised with the values y_r, $\cdots$, y_{r+l-1}. In particular, since multiplying the terms of a LRS by a constant does not change its period, in what follows we will assume $d = 1$.

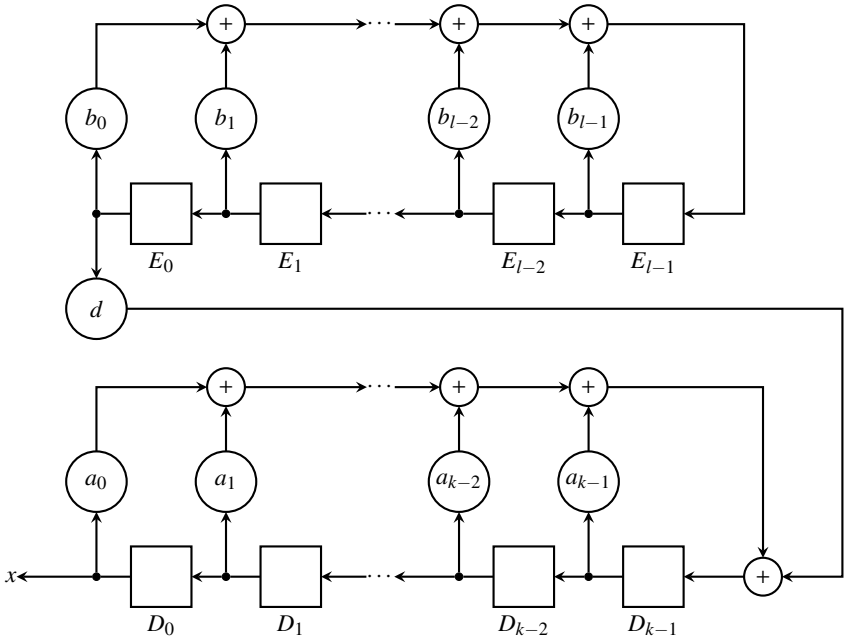


Fig. 4. Diagram of two concatenated LFSR

4 Analysis of Concatenated LRS

4.1 Sum Decomposition of Concatenated LRS

In order to study the period of the concatenated linear recurring sequence $s \leftsquigarrow v$ giving rise to preimage $x \in F^{-1}(y)$, we first prove that it can be decomposed into the *sum* of two LRS: namely, sequence s and the 0-*concatenation* $u = s \leftsquigarrow_0 v$ satisfying the same recurrence Equation (9) of x, but whose k initial terms $u_0, \cdots, u_{k-1}$ are set to 0.

Theorem 3. *Let $s = s_0, s_1, \cdots$ and $v = v_0, v_1, \cdots$ be the LRS respectively satisfying Equations (11) and (10), whose initial terms are respectively $s_0 = x_0, \cdots, s_{k-1} = x_{k-1}$ and $v_0 = y_r, \cdots, v_{l-1} = y_{r+l-1}$, and let $x = s \leftsquigarrow v$ be the concatenation of s and v defined by Equation (9), where $d = 1$. Additionally, let $u = s \leftsquigarrow_0 v$ be the 0-concatenation of sequences s and v, where $u_0 = u_1 = \cdots = u_{k-1} = 0$. Then, $x_n = s_n + u_n$ for all $n \in \mathbb{N}$.*

Proof. Since $u_0 = u_1 = \cdots = u_{k-1} = 0$, for all $n \in \{0, \cdots, k-1\}$ it holds

$$s_n + u_n = s_n + 0 = x_n \quad .$$

Therefore, it remains to prove $x_n = s_n + u_n$ for all $n \geq k$. We proceed by induction on n. For $n = k$, we have

$$\begin{aligned} s_k + u_k &= a_0 s_0 + \cdots + a_{k-1} s_{k-1} + a_0 u_0 + \cdots + a_{k-1} u_{k-1} + v_0 = \\ &= a_0 x_0 + \cdots + a_{k-1} x_{k-1} + v_0 = x_k \quad . \end{aligned}$$

For the induction step we assume $s_n + u_n = x_n$ *for* $n \leq k$*. The sum* $s_{n+1} + u_{n+1}$ *is equal to:*

$$\begin{aligned} s_{n+1} + u_{n+1} &= a_0 s_{n-k+1} + \cdots + a_{k-1} s_n + a_0 u_{n-k+1} + \cdots + a_{k-1} u_n + v_{n-k+1} = \\ &= a_0(s_{n-k+1} + u_{n-k+1}) + \cdots + a_{k-1}(s_n + u_n) + v_{n-k+1} \ . \end{aligned} \tag{12}$$

By induction hypothesis, $s_{n-k+i} + u_{n-k+i} = x_{n-k+i}$ *for all* $i \in \{1, \cdots, k\}$*. Hence, Equation* (12) *can be rewritten as*

$$s_{n+1} + u_{n+1} = a_0 x_{n-k+1} + \cdots + a_{k-1} x_n + v_{n-k+1} = x_{n+1} \ .$$

□

4.2 Characteristic Polynomial of Concatenated LRS

Theorem 3 tells us that a preimage $x \in F^{-1}(y)$ can be generated by the sum of two LRS: the LRS generated by the concatenated LFSR of Figure 4, where the disturbed LFSR is initialised to zero, and the LRS produced by the *non-disturbed* LFSR, that is, the lower LFSR in Figure 4 without the external feedback, initialised to the values $x_0, \cdots, x_{k-1}$.

We now show that this sum decomposition allows one to determine a characteristic polynomial of the concatenated sequence $x = s \leftsquigarrow v$. To this end, we first need a result proved by Chassé in [3] which concerns the generating function of the 0-concatenation $u = s \leftsquigarrow_0 v$. The proof stands on the observation that for all $n \in \mathbb{N}$, the n-th term of u is given by the linear combination $\sum_{i=0}^{n-1} A_n^{(i)} \cdot v_i$, where the terms $A_n^{(i)}$ depend only on the coefficients a_j which define Equation (11). In particular, we will need the values of $A_n^{(0)}$ for $n \geq 0$, which can be computed by the following recurrence equation:

$$A_n^{(0)} = \begin{cases} \sum_{j=0}^{k-1} a_j A_{n-k+j}^{(0)} \ , \text{ if } n > 1 \\ 1 \ , \text{ if } n = 1 \\ 0 \ , \text{ if } n = 0 \end{cases} \tag{13}$$

where $k = 2r$ and $A_{n-k+j}^{(0)} = 0$ if $n - k + j < 0$. Using our notation and terminology, Chassé's result can thus be stated as follows:

Proposition 2. *Let* $u = s \leftsquigarrow_0 v$ *be the* 0*-concatenation of the LRS* s *and* v *defined in Theorem 3, and let* $V(x)$ *be the generating function of* v*. Denoting by* $\mathcal{A}(x)$ *the generating function of the sequence* $A = \{A_{n+1}^{(0)}\}_{n \in \mathbb{N}}$*, the generating function of* u *is*

$$U(x) = x \cdot \mathcal{A}(x) \cdot V(x) \ . \tag{14}$$

Moreover, if $a(x) \in \mathbb{F}_q[x]$ *is the characteristic polynomial of the sequence* s *associated to the recurrence equation* (11)*, then* $a(x)$ *is also a characteristic polynomial of* A*.*

We now prove that the characteristic polynomial of the concatenation $s \leftsquigarrow v$ is the product of the characteristic polynomials of s and v.

Theorem 4. *Let $s \leftsquigarrow v$ be the concatenation of LRS s and v defined by Equation (9) with $d = 1$, and let $a(x), b(x) \in \mathbb{F}_q[x]$ be the characteristic polynomials of s and v, respectively associated to the linear recurring equations (11) and (10). Then, $a(x) \cdot b(x)$ is a characteristic polynomial of $s \leftsquigarrow v$.*

Proof. *By Theorem 3 the concatenation of LRS s and v can be written as $s \leftsquigarrow v = s + u$, where $u = s \leftsquigarrow_0 v$ is the 0-concatenation associated to $s \leftsquigarrow v$. By applying the fundamental identity of formal power series (Equation (5)) and Proposition 2, the following equalities hold:*

$$S(x) = \frac{g_s(x)}{a^*(x)} \tag{15}$$

$$U(x) = \frac{x \cdot g_A(x) \cdot g_v(x)}{a^*(x) \cdot b^*(x)} \ , \tag{16}$$

where $g_s(x)$, $g_A(x)$ and $g_v(x)$ are polynomials whose coefficients are computed according to the numerator in the RHS of Equation (5). Hence, the generating function of $s \leftsquigarrow v$ is:

$$G(x) = \frac{g_s(x)}{a^*(x)} + \frac{x \cdot g_A(x) \cdot g_v(x)}{a^*(x) \cdot b^*(x)} = \frac{g_s(x) \cdot b^*(x) + x \cdot g_A(x) \cdot g_v(x)}{a^*(x) \cdot b^*(x)} \ . \tag{17}$$

By applying again the fundamental identity of formal power series to Equation (17), we deduce that the reciprocal of $c(x) = a^*(x) \cdot b^*(x)$ is a characteristic polynomial of $s \leftsquigarrow v$. Denoting by k and l the degrees of $a(x)$ and $b(x)$ respectively, it follows that $c(x) = x^{k+l} \cdot a(1/x) \cdot b(1/x)$, and thus the reciprocal of $c(x)$ is

$$c^*(x) = x^{k+l} \cdot \frac{1}{x^{k+l}} \cdot a(x) \cdot b(x) = a(x) \cdot b(x) \ . \tag{18}$$

Therefore, $a(x) \cdot b(x)$ is a characteristic polynomial of $s \leftsquigarrow v$. □

Theorem (4) thus gives a characteristic polynomial for all preimages $x \in F^{-1}(y)$ of a spatially periodic configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$. As a matter of fact, the polynomials $a(x)$ and $b(x)$ do not depend on the particular value of the block $x_{[0,2r-1]}$, but only on the local rule f and on configuration y, respectively. From the LFSR point of view, this means that a preimage $x \in F^{-1}(y)$ can be generated by a single LFSR implementing the $(k+l)$-th order recurrence equation

$$\sigma_{n+k+l} = c_0\sigma_n + c_1\sigma_{n+1} + \cdots + c_{k+l-1}\sigma_{n+k+l-1} \ , \tag{19}$$

where for all $\mu \in \{0, \cdots, k+l-1\}$ the term c_μ is the μ-th convolution coefficient in the multiplication $a(x) \cdot b(x)$ given by

$$c_\mu = \sum_{i+j=\mu} a_i b_j, \text{ for } i \in \{0, \cdots, k\} \text{ and } j \in \{0, \cdots, l\} \ . \tag{20}$$

Additionally, the first $k = 2r$ initial terms $\sigma_0, \cdots, \sigma_{k-1}$ in Equation (19) are initialised to the values in $x_{[0,2r-1]}$, while the remaining l ones are obtained using the

recurrence equation (9). Hence, by applying the fundamental identity of formal power series, the numerator of Equation (17) can also be expressed as:

$$g(x) = -\sum_{j=0}^{k-1} \sum_{i=0}^{j} c_{i+k-j} \sigma_i x^j \quad . \tag{21}$$

5 Further Results

5.1 Computing the Period of a Single Preimage

To summarise the results discussed so far, we now present a practical procedure to compute the spatial period of a single preimage. Given a LBCA $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ with local rule $f : \mathbb{F}_q^{2r+1} \to \mathbb{F}_q$ of radius $r \in \mathbb{N}$, a spatially periodic configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$ and a $2r-$cell block $x_{[0,2r-1]} \in \mathbb{F}_q^{2r}$ of a preimage $x \in F^{-1}(y)$, the procedure can be described as follows:

1. Find the minimal polynomial $b(x) = x^l - b_{l-1}x^{l-1} \cdots - b_0$ of the linear recurring sequence v, where $v_n = y_{n+r}$ for all $n \in \mathbb{N}$.
2. Set the characteristic polynomial $a(x)$ associated to the inverse permutation $f_{R,z}^{-1}$ to $a(x) = x^k - a_{k-1}x^{k-1} - \cdots - a_0$, where $k = 2r$ and the coefficients a_i are those appearing in the recurrence equation (11).
3. Compute the polynomial $g(x)$ given by Equation (21), and set $h(x) = -g^*(x)$.
4. Determine the minimal polynomial of the preimage by computing

$$m(x) = \frac{a(x) \cdot b(x)}{gcd(a(x) \cdot b(x), h(x))} \quad . \tag{22}$$

5. Compute the order of $m(x)$, and output it as the period of preimage x.

For step 1, the minimal polynomial of v can be found using the *Berlekamp-Massey algorithm* [11], by giving as input to it the string composed by the first $2p$ elements of v, where p is the period of y (and hence the period of v as well). The time complexity of this algorithm is $O(p^2)$. Step 4 requires the computation of a greatest common divisor, which can be performed using the standard Euclidean division algorithm in $O(n^2)$ steps, where $n = max\{deg(a(x)b(x)), deg(h(x))\}$. Finally, the order of $m(x)$ in step 5 can be determined by first factorizing the polynomial, for example by using *Berlekamp's algorithm* [1] which has a time complexity of $O(D^3)$, where D is the degree of $m(x)$, if the characteristic ρ of $\mathbb{F}_q$ is sufficiently small. Once the factorization of $m(x)$ is known, $ord(m(x))$ can be computed using the following theorem proved in [9]:

Theorem 5. *Let $m(x) \in \mathbb{F}_q[x]$ be a polynomial having positive degree and such that $m(0) \neq 0$. Let $m(x) = a \cdot \prod_{i=0}^{n} f_i(x)^{b_i}$ be the canonical factorization of $m(x)$, where $a \in \mathbb{F}_q$, $b_1, \cdots, b_n \in \mathbb{N}$ and $f_1(x), \cdots, f_n(x) \in \mathbb{F}_q[x]$ are distinct monic irreducible polynomials. Then $ord(m(x)) = e\rho^t$, where ρ is the characteristic of $\mathbb{F}_q$, e is the least common multiple of $ord(f_1(x)), \cdots, ord(f_n(x))$ and t is the smallest integer such that $\rho^t \geq \max(b_1, \cdots, b_n)$.*

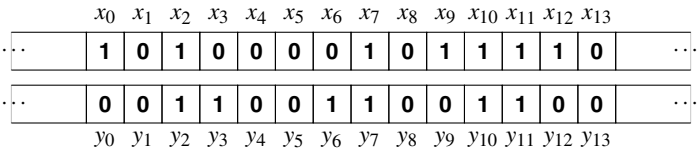


Fig. 5. Block $x_{[0,11]}$ which generates preimage $x \in F^{-1}(y)$ under rule 150, computed using case (a) of Equation (1). Notice that $(x_{12}, x_{13}) = (x_0, x_1)$ and $(y_{12}, y_{13}) = (y_0, y_1)$. Hence, for $n \geq 12$ and $n < 0$ the preimage will periodically repeat itself.

Notice that Theorem 5 depends on the knowledge of the orders of the irreducible polynomials involved in the factorization of $m(x)$. A method to determine the order of an irreducible polynomial is also described in [9], which relies on the factorization of $q^D - 1$. There exist several factorization tables for numbers in this form, especially for small values of q (see for example [4]).

We now present a practical application of the procedure described above. The computations in the following example have been carried out with the computer algebra system MAGMA.

Example 1. Let $F : \mathbb{F}_2^{\mathbb{Z}} \to \mathbb{F}_2^{\mathbb{Z}}$ be the LBCA with local rule $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ of radius $r = 1$, defined as $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$ for all $(x_1, x_2, x_3) \in \mathbb{F}_2^3$, which is the elementary rule 150. Let $y \in \mathbb{F}_2^{\mathbb{Z}}$ be a spatially periodic configuration of period $p = 4$ generated by the block $y_{[0,3]} = (0, 0, 1, 1)$, and let $x_{[0,1]} = (1, 0)$ be the initial 2-cell block of a preimage $x \in F^{-1}(y)$. Since $r = 1$, sequence v is generated by block $v_{[0,3]} = (0, 1, 1, 0)$. Feeding the string $(0, 1, 1, 0, 0, 1, 1, 0)$ to the Berlekamp-Massey algorithm yields the polynomial $b(x) = x^3 + x^2 + x + 1$, while the characteristic polynomial associated to rule 150 is $a(x) = x^2 + x + 1$. Hence, it follows that $c(x) = a(x) \cdot b(x) = x^5 + x^3 + x^2 + 1$ is a characteristic polynomial of the preimage. Since the first 5 elements of preimage x are $1, 0, 1, 0, 0$, the initialisation polynomial of Equation (21) is $g(x) = x^4 + x^3 + 1$, from which we deduce that $h(x) = x^4 + x + 1$. Considering that $h(x)$ is irreducible, the greatest common divisor of $c(x)$ and $f(x)$ is 1, and thus by Equation (22) $c(x)$ is also the minimal polynomial of the preimage. The factorization of $c(x)$ is $(x+1)^3(x^2+x+1)$, and the orders of $x+1$ and $x^2 + x + 1$ are respectively 1 and 3, from which it follows that the least common multiple e is 3. Finally, the smallest integer t such that $2^t \geq 3$ is $t = 2$. Therefore, by applying Theorem 5 the period of preimage x is $e2^t = 12$. Figure 5 shows the actual value of the block $x_{[0,11]}$ which generates preimage x.

5.2 Characterisation of Periods When $a(x)$ and $b(x)$ Are Irreducible

As a further application of Theorem 4, we now show a complete characterisation of the periods of $x \in F^{-1}(y)$ in the special case where the characteristic polynomials $a(x)$ and $b(x)$ are irreducible. To this end, we first report an additional theorem proved in [9] which concerns the sum of families of LRS.

Theorem 6. *Let $f_1(x), f_2(x) \in \mathbb{F}_q$ be non-constant monic polynomials, and let $S(f_1(x))$ and $S(f_2(x))$ be the families of LRS whose characteristic polynomials are respectively $f_1(x)$ and $f_2(x)$. Denoting by $S(f_1(x)) + S(f_2(x))$ the family of all LRS $\sigma + \tau$ where $\sigma \in S(f_1(x))$ and $\tau \in S(f_2(x))$, it follows that $S(f_1(x)) + S(f_2(x)) = S(c(x))$, where $c(x)$ is the least common multiple of $f_1(x)$ and $f_2(x)$.*

Our characterisation result, which is analogous to Theorem 2, is the following:

Theorem 7. *Let $F : \mathbb{F}_q^{\mathbb{Z}} \to F_q^{\mathbb{Z}}$ be an LBCA having local rule $f : \mathbb{F}_q^{2r+1} \to \mathbb{F}_q$, and let $a(x) = x^k - a_{k-1}x^{k-1} - \cdots - a_0 \in \mathbb{F}_q[x]$ be the characteristic polynomial associated to the inverse permutation $f_{R,z}^{-1}$, where $k = 2r$, $a_0, \cdots, a_{k-1}$ are the coefficients appearing in Equation (11) and $ord(a(x)) = e$. Further, let $y \in \mathbb{F}_q^{\mathbb{Z}}$ be a spatially periodic configuration of period $p > 1$, and let $b(x)$ be the minimal polynomial of sequence v, where $v_n = y_{n+r}$ for all $n \in \mathbb{N}$. If $a(x)$ and $b(x)$ are both irreducible and $a(x) \neq b(x)$, then $F^{-1}(y)$ contains one configuration of period p and $q^k - 1$ configurations of period m, where m is the least common multiple of e and p.*

Proof. By Theorem (4), $a(x) \cdot b(x)$ is a characteristic polynomial of the q^k preimages in $F^{-1}(y)$. Denote by $S(a(x))$ and $S(b(x))$ the sets of LRS having characteristic polynomials $a(x)$ and $b(x)$, respectively. Since $a(x)$ and $b(x)$ are both irreducible and $a(x) \neq b(x)$, by Theorem 6 it follows that $S(a(x) \cdot b(x)) = S(a(x)) + S(b(x))$. Hence, $F^{-1}(y)$ is a subset of $S(a(x)) + S(b(x))$, and as a consequence every preimage $x \in F^{-1}(y)$ can be written as $x = \sigma + \tau$, where $\sigma \in S(a(x))$ and $\tau \in S(b(x))$. In particular, by applying Theorem 2 it results that $S(a(x))$ is composed by one sequence of period 1 and $q^k - 1$ sequences of period e, while since $p > 1$ the sequence τ is necessarily one of the $q^l - 1$ sequences of period p of $S(b(x))$, where l is the degree of $b(x)$. Therefore, by making all possible sums for σ ranging in $S(a(x))$, Theorem 1 yields that $F^{-1}(y)$ is composed by one configuration having period p, which is the preimage $x = \sigma + \tau$ where σ has period 1, while the period of all the remaining $q^k - 1$ configurations is the least common multiple of e and p. □

6 Conclusions

In this work, we studied the relation between the periods of spatially periodic configurations of LBCA and the periods of their preimages, characterising the latter as concatenations of linear recurring sequences. We remark that Theorem 4 can be straightforwardly generalised to the case $x^{(t)} \in F^{-t}(y)$, i.e. preimages of y with respect to the *t-th iterate* of the CA, where $t \in \mathbb{N}$. Indeed, it can be shown that $a(x)^t \cdot b(x)$ is a characteristic polynomial of $x^{(t)}$, which is thus generated by a "cascade" of concatenated LFSR where each LFSR is initialised to a block $x^{(i)}_{[0,2r-1]}$ of an intermediate preimage $x^{(i)} \in F^{-i}(y)$, for $i \in \{1, \cdots, t\}$. Of course in this case we have to take into account the fact that the running time of the procedure described in Section 5.1 grows exponentially in the degree D of the minimal polynomial $m(x)$, since it depends on the factorization of $q^D - 1$.

We conclude by discussing some possible future directions of research on the subject. A first idea is to generalise the results presented in this paper to

nonlinear BCA, where the preimages are generated by a *Nonlinear Feedback Shift Register* (NFSR) disturbed by the LFSR which generates configuration y. We remark that this concatenation is also the main primitive upon which the stream cipher Grain is based [8]. Hence, finding a general method to study the periods of preimages of nonlinear BCA could also be useful to cryptanalyse this cipher. This study could be further generalised to generic surjective CA. In this regard, a possible starting point could be a result reported in [5], which implies that if $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ is a surjective linear CA, then there exists $t \in \mathbb{N}$ such that the t-th iterate F^t is bipermutive. Finally, a further extension of this research would be to analyse the periods of spatially periodic configurations in the case of multi-dimensional cellular automata, by considering suitable notions of bipermutivity such as the ones introduced in [6].

References

1. Berlekamp, E.R.: Factoring polynomials over finite fields. Bell Syst. Tech. J. **46**, 1853–1859 (1967)
2. Cattaneo, G., Finelli, M., Margara, L.: Investigating topological chaos by elementary cellular automata dynamics. Theor. Comp. Sci. **244**, 219–241 (2000)
3. Chassé, G.: Some remarks on a LFSR "disturbed" by other sequences. In: Cohen, G., Charpin, P. (eds.) EUROCODE 1990. LNCS, vol. 514, pp. 215–221. Springer, Heidelberg (1991)
4. The Cunningham Project. http://homes.cerias.purdue.edu/~ssw/cun/index.html
5. Dennunzio, A., Di Lena, P., Formenti, E., Margara, L.: On the directional dynamics of additive cellular automata. Theor. Comput. Sci. **410**, 4823–4833 (2009)
6. Dennunzio, A., Formenti, E., Weiss, M.: Multidimensional cellular automata: closing property, quasi-expansivity, and (un)decidability issues. Theor. Comput. Sci. **516**, 40–59 (2014)
7. Hedlund, G.A.: Endomorphisms and Automorphisms of the Shift Dynamical Systems. Mathematical Systems Theory **7**(2), 138–153 (1973)
8. Hell, M., Johansson, T., Maximov, A., Meier, W.: The grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
9. Lidl, R., Niederreiter, H.: Introduction to finite fields and their applications. Cambridge University Press, Cambridge (1994)
10. Mariot, L., Leporati, A.: Sharing secrets by computing preimages of bipermutive cellular automata. In: Wąs, J., Sirakoulis, GCh., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 417–426. Springer, Heidelberg (2014)
11. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory **15**, 122–127 (1969)

Merging Cellular Automata Rules to Optimise a Solution to the Modulo-*n* Problem

Claudio L.M. Martins[2(✉)] and Pedro P.B. de Oliveira[1,2]

[1] Faculdade de Computação e Informática, Universidade Presbiteriana Mackenzie, São Paulo, Brazil
[2] Pós-Graduação em Engenharia Elétrica e Computação, Rua da Consolação 896, Consolação 01302-907, São Paulo, SP, Brazil
claudio.luis.martins@terra.com.br, pedrob@mackenzie.br

Abstract. Understanding how the composition of cellular automata rules can perform predefined computations can contribute to the general notion of emerging computing by means of locally processing components. In this context, a solution has been recently proposed to the Modulo-*n* Problem, which is the determination of whether the number of 1-bits in a binary string is perfectly divisible by the positive integer *n*. Here, we show how to optimise that solution in terms of a reduction of the number of rules required, by means of a *merging* operation involving of the rules´ active state transitions. The potential for a more general usage of the merging operation is also addressed.

Keywords: Cellular automata · Emergent computation · Rule composition · Modulo-*n* problem · MOD*n* problem · Merging · Active state transitions · Parity problem

1 Introduction: The Modulo-*n* Problem Solution to be Optimised

Cellular automata (CAs) are discrete dynamical systems with a grid-like regular lattice of identical finite automata cells, each cell having an identical pattern of connections to its neighbours. The next state of each cell is given by the transition rule of the automaton, according to the current cell state and those of its neighbouring cells. CAs may perform arbitrary computations, even out of the action of simple local rules [6].

One of these computations consists in solving the parity problem, herein denoted the MOD2 problem, which consists of determining the parity of the number of 1s in a binary string: even parity, when the number of 1s modulo-2 is 0, or odd parity, when the number of 1s modulo-2 is 1.

In its formulation for cellular automata, this computational problem is considered solved when any odd-sized (N) binary string initialising a cyclic lattice, is converted, after some time steps, into 0^N or 1^N, if the initial amount of 1-bits is even, or odd, respectively. The MOD2 problem is ill-defined for even-sized lattices because the initial configuration 1^N would have an even number of 1s, which would be a contradiction because 1^N should be the final configuration for lattices with odd number of

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 196–209, 2015.
DOI: 10.1007/978-3-662-47221-7_15

1s. Although it has been proved in [5] that a one-dimensional rule with radius at least 4 is required to solve MOD2, [2] and [4] showed how to solve the problem with a composition of only two elementary rules. This makes it evident the power of rule compositions.

As a general case, we refer to the MOD*n* problem, which consists of determining whether the number of 1-bits in a cyclic binary string is multiple of *n*, with the constraint that it is always ill-defined for lattice sizes multiple of *n*.

In [1], we described a generalised solution to the MOD*n* problem, based upon the application of a set of one-dimensional CA rules, in a pre-determined order, which amounts to composing the individual rules employed. This solution is only constrained in that the lattice size *N* cannot be a multiple of *n* nor a multiple of any factor of *n*. Such a general solution (S_n) for the MOD*n* problem, for any binary string σ with size *N*, is given below, where we name rules R_n^0 and R_n^1 as the *Replacement* rules, and G_1^0, G_1^1, G_2^0, G_2^1 and so on, as the *Grouping* rules, whose meanings are defined in the next section.

$$S_n = E_{254}^{\left\lfloor \frac{N}{2} \right\rfloor} \left(G_{n-2}^{1\left\lfloor \frac{N}{2} \right\rfloor} \ldots G_1^{1\left\lfloor \frac{N}{2} \right\rfloor} R_n^{1\left\lfloor \frac{N}{n} \right\rfloor} G_{n-2}^{0\left\lfloor \frac{N}{2} \right\rfloor} \ldots G_1^{0\left\lfloor \frac{N}{2} \right\rfloor} R_n^{0\left\lfloor \frac{N}{n} \right\rfloor} \right)^{\left\lceil \frac{N}{n} \right\rceil} \sigma. \quad (1)$$

The solution means that starting with the size-*N* initial configuration σ, the following sequence of rule applications should be performed:

1. Apply rule R_n^0 for $\lfloor \frac{N}{n} \rfloor$ time steps, followed by rules G_1^0, G_2^0 and so on, up to G_{n-2}^0, for $\lfloor \frac{N}{2} \rfloor$ time steps each.
2. Apply rule R_n^1 for $\lfloor \frac{N}{n} \rfloor$ time steps, followed by rules G_1^1, G_2^1 and so on, up to G_{n-2}^1, for $\lfloor \frac{N}{2} \rfloor$ time steps each.
3. Repeat the two previous procedures $\lceil \frac{N}{n} \rceil$ times.
4. Finalise the process, by applying elementary CA rule 254 for $\lfloor \frac{N}{2} \rfloor$ time steps.

Since the sequence of rules superscripted with 0 operate on the 0-bits and the ones superscripted with 1 operate on the 1-bits, the stages 1 and 2 above may be inverted, with the same global outcome.

In this paper, we propose a simplification of the solution above, by performing a *merging* procedure of the rules´ *active* state transitions, that is, those that replace the state of the centre cell in the neighbourhood.

In the next section, we present the *Replacement* rules, the *Grouping* rules and the result of their composition. In Section 3, we discuss the active state transitions of the rules present in S_n, as well as how they can be used to simplify their representations. The merging of these rules is then discussed in Section 4, as well as how the active transitions should be modified so as to render viable mergings. We conclude in Section 5 with various remarks, in particular addressing some conditions we must respect in the choice of rules to be merged and the active transitions to be modified.

2 Replacement and Grouping Rules

Two key roles are required for rules to solve the MOD*n* problem: to transform certain blocks of states of a configuration, and to group together specific kinds of blocks; these roles are achieved by the *Replacement* rules (*R*) and the *Grouping* rules (*G*), respectively. Elementary rule 254 just finalises the problem, by preserving the configuration 0^N, and transforming all the others to 1^N.

Replacement rules R_n^0 can replace *n* end 0s, of a sequence of *n* or more consecutives 0s, with *n* 1s, while, analogously, *Replacement* rules R_n^1 can replace *n* end 1s, of a sequence of *n* or more consecutives 1s, with *n* 0s. Both are, therefore, MOD*n*-conserving rules. Considering the *n* end bits, 0s or 1s, that need to be replaced of a sequence of *n* or more consecutive identical bits, the following cases are possible: *n* bits from the left, *n*-1 from the left and 1 from the right, *n*-2 from the left and 2 from the right, …, 2 from the left and *n*-2 from the right, 1 from the left and *n*-1 from the right, and *n* bits from the right.

Knowing that a one-dimensional CA rule that changes *n* end bits from one side of the string must have at least radius *n*, and that it suffices radius *n*-1 for a rule that is to change *n*-1 bits from one side and 1 from the other side, or *n*-2 bits from one side and 2 from the other side, and so on, we can consider only the smallest possible radius of these rules.

So, there are *n*-1 rules that replace *n* end 0s, of a sequence of *n* or more 0s, with *n* 1s, namely, rule $R_{n-1,1}^0$ (*n*-1 bits from the left and 1 from the right), that transforms the strings $10^{n-1}0^x01$ into $11^{n-1}0^x11$, rule $R_{n-2,2}^0$ (*n*-2 bits from the left and 2 from the right), that transforms the strings $10^{n-2}0^x001$ into $11^{n-2}0^x111$, and so on, up to rule $R_{1,n-1}^0$ (1 bit from the left and *n*-1 from the right), that transforms the strings $100^x0^{n-1}1$ into $110^x1^{n-1}1$; in all cases, for any integer $x \geq 0$.

By applying *Replacement* rules for $\lfloor \frac{N}{n} \rfloor$ iterations, no sequence with *n* or more consecutive identical bits is left in the lattice, except if its configuration is 0^N or 1^N.

In order to eliminate simultaneous occurrence of different blocks of the same bit left by the *Replacement* rules, the smaller blocks will be grouped into larger ones, moving themselves through the lattice, according to the *Grouping* rules.

Grouping rules are those that can shift to the left or to the right, strings of *m* identical bits, in order to group them with larger strings of the same bit.

We refer to a *Grouping* rule that can shift *m* 0s as G_m^0; but since this movement may be possible to the left or to the right, we denote it $\overleftarrow{G}_m^0$ when the movement is to the left, and $\vec{G}_m^0$ when the movement is to the right. Analogously, in order to shift and group *m* 1s, rules $\overleftarrow{G}_m^1$ and $\vec{G}_m^1$ are employed, or just G_m^1, indistinctly.

Rules that can only move an isolated bit, 0 or 1 (G_1^0 or G_1^1, respectively), should have, at least, radius 2. Rules that can only move an isolated pair of bits should have at least radius 3, and so on. These rules are also MOD*n*-conserving rules.

For either 0 or 1, *n*-2 *Grouping* rules will be used, because, after the application of the *Replacement* rules, no strings of consecutive identical bits larger than *n*-1 will remain in the lattice. So, we just have to move strings smaller than *n*-1 consecutive identical bits to group them into the larger blocks of the same bit.

In order to solve the MOD2 problem, since $n = 2$, no *Grouping* rules are necessary, as demonstrated in [4].

By composing only *Replacement* and *Grouping* rules, with no application of the elementary rule 254, any initial configuration is transformed into another that belongs to a reduced group of final configurations. We disregard differences due to rotational symmetry, which means that final configurations as 1100000000, 0110000000, ..., 0000000110 are considered the same as 0000000011.

Simplifying the possible relationships between initial and final configurations before applying rule 254, we have the following, where $|\sigma|_1$ stands for the number of 1s in string σ:

If MOD$n(N)$ = MOD$n(|\sigma|_1)$ and both $\neq 0$ (ill-defined problem):

— MOD$n(|\sigma|_1) \neq 0$: 1^N (meaning that, when MODn $(|\sigma|_1)=1$, convergence is to 1^N)

If MOD$n(N) \neq$ MOD$n(|\sigma|_1)$:

— MOD$n(|\sigma|_1) = 0$: 0^N

— MOD$n(|\sigma|_1) \neq 0$: $0^{N\text{-MOD}n(|\sigma|1)}1^{\text{MOD}n(|\sigma|1)}$ or some *necklaces*

We have already disregarded the lattices with size N multiple of n (ill-defined problem), where MOD$n(N) = 0$.

The predominance of 0s instead of 1s is because, at the end, we use the *Replacement* rules that operate on the 1s, transforming them into 0s.

Necklaces are configurations of the form $(0^A1^B)^C$, for integers A, B and C, where $A < n$ and $B < n$, or $A < n$ and $B > n$, but B is not multiple of n, or $B < n$ and $A > n$ but A is not multiple of n. For necklace configurations, the *Grouping* rules just cause shifts on the lattice, with no further effect; also, the *Replacement* rules cause no effect when $A < n$ and $B < n$, or may lead to periodic regimes only when $A > n$ or $B > n$, by continuously transforming $(0^A1^B)^C$ into $(0^{A'}1^{B'})^C$, back and forth.

If N is a prime number, no *necklace* will remain in the lattice.

Therefore, for any initial configurations where MOD$n(|\sigma|_1) = 0$, the problem is already solved, as defined. However, for all the other configurations where MOD$n(|\sigma|_1) \neq 0$ should be converted into 1^N, elementary rule 254 can be used, without affecting the configuration 0^N.

All the details, lemmas and their proofs, and further explanations regarding this section can be found in [1].

3 Active State Transitions and the Simplified CA Representations

A solution for the MOD3 problem was reported in [3] and further optimised and generalised in [1]. In order to solve the MOD3 problem we need radius 2 rules as *Replacement* rules and *Grouping* rules. A radius 2 rule has 32 state transitions that can be active or not: by active transition, we mean those that change the state value of the centre cell of the neighbourhood.

$R^0_{2,1}$ Rule 4.059.296.252

Transition	Neighbourhood					Output bit	Note
31	1	1	1	1	1	1	
30	1	1	1	1	0	1	
29	1	1	1	0	1	1	
28	1	1	1	0	0	1	
27	1	1	0	1	1	0	
26	1	1	0	1	0	0	
25	1	1	0	0	1	0	
24	1	1	0	0	0	1	1st 0 from the left
23	1	0	1	1	1	1	
22	1	0	1	1	0	1	
21	1	0	1	0	1	1	
20	1	0	1	0	0	1	
19	1	0	0	1	1	0	
18	1	0	0	1	0	0	
17	1	0	0	0	1	1	2nd 0 from the left
16	1	0	0	0	0	1	2nd 0 from the left
15	0	1	1	1	1	1	
14	0	1	1	1	0	1	
13	0	1	1	0	1	1	
12	0	1	1	0	0	1	
11	0	1	0	1	1	0	
10	0	1	0	1	0	0	
9	0	1	0	0	1	0	
8	0	1	0	0	0	1	1st 0 from the left
7	0	0	1	1	1	1	
6	0	0	1	1	0	1	
5	0	0	1	0	1	1	
4	0	0	1	0	0	1	
3	0	0	0	1	1	1	1st 0 from the right
2	0	0	0	1	0	1	1st 0 from the right
1	0	0	0	0	1	0	
0	0	0	0	0	0	0	

Active Transition	Neighbourhood					Output bit	Note
24	1	1	0	0	0	1	1st 0 from the left
17	1	0	0	0	1	1	2nd 0 from the left
16	1	0	0	0	0	1	2nd 0 from the left
8	0	1	0	0	0	1	1st 0 from the left
3	0	0	0	1	1	1	1st 0 from the right
2	0	0	0	1	0	1	1st 0 from the right

Active Transitions	Neighbourhood					Output bit	Note
24 & 8	*	1	0	0	0	1	1st 0 from the left
17 & 16	1	0	0	0	*	1	2nd 0 from the left
3 & 2	0	0	0	1	*	1	1st 0 from the right

$\overleftarrow{G}^0_1$ Rule 3.704.675.536

Transition	Neighbourhood					Output bit	Note
31	1	1	1	1	1	1	
30	1	1	1	1	0	1	
29	1	1	1	0	1	0	1 on the left
28	1	1	1	0	0	1	
27	1	1	0	1	1	1	Isolated 0
26	1	1	0	1	0	1	Isolated 0
25	1	1	0	0	1	0	
24	1	1	0	0	0	0	
23	1	0	1	1	1	1	
22	1	0	1	1	0	1	
21	1	0	1	0	1	0	1 on the left
20	1	0	1	0	0	1	
19	1	0	0	1	1	0	
18	1	0	0	1	0	0	
17	1	0	0	0	1	0	
16	1	0	0	0	0	0	
15	0	1	1	1	1	1	
14	0	1	1	1	0	1	
13	0	1	1	0	1	0	1 on the left
12	0	1	1	0	0	1	
11	0	1	0	1	1	1	Isolated 0
10	0	1	0	1	0	1	Isolated 0
9	0	1	0	0	1	0	
8	0	1	0	0	0	0	
7	0	0	1	1	1	1	
6	0	0	1	1	0	1	
5	0	0	1	0	1	0	1 on the left
4	0	0	1	0	0	1	
3	0	0	0	1	1	0	
2	0	0	0	1	0	0	
1	0	0	0	0	1	0	
0	0	0	0	0	0	0	

Active Transition	Neighbourhood					Output bit	Note
29	1	1	1	0	1	0	1 on the left
27	1	1	0	1	1	1	Isolated 0
26	1	1	0	1	0	1	Isolated 0
21	1	0	1	0	1	0	1 on the left
13	0	1	1	0	1	0	1 on the left
11	0	1	0	1	1	1	Isolated 0
10	0	1	0	1	0	1	Isolated 0
5	0	0	1	0	1	0	1 on the left

Active Transitions	Neighbourhood					Output bit	Note
27, 26, 11 & 10	*	1	0	1	*	1	Isolated 0
29, 21, 13 & 5	*	*	1	0	1	0	1 on the left

Fig. 1. State transitions of rules $R^0_{2,1}$ and $\overleftarrow{G}^0_1$, with their simplified representations through their active transitions

CA rules can be represented just through their active transitions, and grouped whenever possible, i.e., placed together at the same row of the state transition table, using symbol * to stand for either possibilities, 0 or 1 (see Figure 1).

This figure shows all the state transitions of the rules $R^0_{2,1}$ and $\overleftarrow{G}^0_1$, separating and placing together only the active transitions as shown further down in the figure. The *Replacement* rule $R^0_{2,1}$ has only 6 active transitions, highlighted out of the 32 possibilities, and can be

represented by the three rows and the end of the figure. The *Grouping* rule $\overleftarrow{G}_1^0$ has 8 active transitions, and can be represented just by two rows. The written notes indicate which bit is been replaced by the *Replacement* rule or changed to make the shift by the *Grouping* rule.

The minimum required radius for a rule to perform as expected depends on the number of cells (excluding those with the * character), to the right or to the left, of the centre cell of the neighbourhood of all grouped active transitions. The highest value is the required radius. At the end of Figure 1 we can see that the minimum radius required to both rules is 2.

4 Merging *Replacement* and *Grouping* Rules

4.1 The Merging Operation

The *merging* process should, in fact, be generally regarded as a two-stage process, consisting of *joining* together the active transitions of the rules involved, with subsequent *editing* of some of them, if required. Details are given throughout this section.

The *Replacement* and *Grouping* rules to be joined in just one *Merged* rule do not operate on the same strings simultaneously, because these strings have different sizes for any value of *n*, i.e., while *Replacement* rules replace *n* bits (from strings with *n* or more consecutive identical bits), *Grouping* rules move *m* bits (from strings with just an isolated string of *m* bits), and *m* is always equal or smaller than *n*-2.

Therefore, all effects of the *Merged* rule – such as the possibility of partition reduction, the formation of strings with only 0s or only 1s, or the formation of some necklaces – are the same when applying the separated rules. One exception is the formation of *partial necklaces* (described in Section 4.2), that occurs because of the impossibility to join some isolated strings to their larger blocks (created by the *Grouping* rules), due to changes on the lattice, through the simultaneous action of the *Replacement* and *Grouping* rules.

As a consequence, the same lemmas and proofs described in [1] should be considered, but the formation of *partial necklaces* must now be added to the possibilities of final configuration after applying the *Merged* rule. In the next subsections we address some specific cases, such as the merging of the rules that solve the problem MOD3 and the problem MOD4, so as to convey the underlying issues of the process more clearly.

4.2 The MOD3 Case

As is the case here, we may consider that a task of a CA rule can be performed by one or more *active state transitions*. Accordingly, the task performed by rule $R_{2,1}^0$, for example, is to eliminate any string with three or more consecutive 0s in the lattice, conserving the Modulo-3 property, while the task performed by the rule $\overleftarrow{G}_1^0$ is to shift isolated 0s to the left in order to group them with larger strings of the same bit, also conserving the Modulo-3 property. In what follows, we go about joining these functions into a single rule.

Figure 1 shows that *Replacement* rule $R_{2,1}^0$ and *Grouping* rule $\overleftarrow{G}_1^0$ have different active state transitions. The resulting rule of the merging of a *Replacement* rule and one or more

Grouping rules is generically termed in this work as M_n^0 or M_n^1, according to the specific bit it manipulates. In the case of the MOD3 problem, for instance, merging rules $R_{2,1}^0$ and $\overleftarrow{G}_1^0$ results the rule $\overleftarrow{M}_{2,1}^0$.

Figure 2 shows the simplified representation of rule $\overleftarrow{M}_{2,1}^0$ from the perspective of its active transitions.

$\overleftarrow{M}_{2,1}^0$ **Rule 3.721.649.628**

Active Transitions	Neighbourhood					Output bit	Note
27, 26, 11 & 10	*	1	0	1	*	1	Isolated 0
29, 21, 13 & 5	*	*	1	0	1	0	1 on the left
24 & 8	*	1	0	0	0	1	1st 0 from the left
17 & 16	1	0	0	0	*	1	2nd 0 from the left
3 & 2	0	0	0	1	*	1	1st 0 from the right

Fig. 2. Simplified representation of the rule $\overleftarrow{M}_{2,1}^0$

Our goal is to ensure that these 3 alternatives can perform the same tasks: $\left(\overleftarrow{G}_1^{0\,N}\, R_{2,1}^{0\,N}\right)^N \sigma$, $\left(R_{2,1}^{0\,N}\, \overleftarrow{G}_1^{0\,N}\right)^N \sigma$, or $\left(\overleftarrow{M}_{2,1}^{0\,N}\right)^N \sigma$.

The application to all possible initial configurations of a given size of the possible rule sequences – that is, the *Replacement* rule followed by the *Grouping* rule, or vice versa, or yet the *Merged* rule only – allows us to compare all resulting final configurations; this is what is summarised in Figure 3, for N = 13 (therefore, with respect to all 2^{13} different initial configurations).

Analysis of the data in Figure 3 indicates that the Modulo-3 property is preserved for all initial configurations, thus increasing their number of 1s, up to 11, 12 or 13 occurrences, correspondingly to the initial values of Modulo-3 equal to 2, 0 or 1, respectively. This occurs because the *Replacement* rule can transform three 0s into three 1s. The alternative that begins with the *Grouping* rule has an advantage in this replacement task, in that it shifts isolated 0s and groups them into larger chains before replacing the 0s. This increases the amount of strings with three or more consecutive 0s, therefore improving the effectiveness of the *Replacement* rule.

Hence, the final configurations of the different alternatives are not necessarily the same, even disregarding differences due to rotational symmetries, because the number of 1s may be different. Even with the same number of 1s, the spaces between isolated 0s may be also different.

By a superficial analysis, we would say that there is no equivalence among the outcomes of the three possible processing alternatives; but, if we go back to our initial goals, it is possible to observe that, for all possible initial conditions, the application of any of the three alternatives

transforms the lattice into a final configuration as desired. In other words, all final configurations have at most two consecutive 0s (because of the *Replacement* rule) and there are no isolated 0s and isolated pairs of 0s occurring simultaneously (because of the *Grouping* rule).

$\left(\overleftarrow{G_1^0}^{\,N} R_{2,1}^{0\,N} \right)^N$

Amount of 1s		Amount of
Initial	Final	Configurations
0	0	1
1	13	13
2	11	78
3	9	78
	12	208
4	7	52
	10	130
	13	533
5	5	13
	8	13
	11	1261
6	9	585
	12	1131
7	7	78
	10	455
	13	1183
8	8	91
	11	1196
9	9	234
	12	481
10	10	156
	13	130
11	11	78
12	12	13
13	13	1
Total		8192

$\left(R_{2,1}^{0\,N} \overleftarrow{G_1^0}^{\,N} \right)^N$

Amount of 1s		Amount of
Initial	Final	Configurations
0	0	1
1	13	13
2	11	78
3	9	104
	12	182
4	7	65
	13	650
5	5	13
	11	1274
6	9	650
	12	1066
7	7	78
	13	1638
8	8	91
	11	1196
9	9	234
	12	481
10	10	156
	13	130
11	11	78
12	12	13
13	13	1
Total		8192

$\left(\overleftarrow{M}_{2,1}^{0\,N} \right)^N$

Amount of 1s		Amount of
Initial	Final	Configurations
0	0	1
1	13	13
2	11	78
3	9	78
	12	208
4	7	52
	10	130
	13	533
5	5	13
	8	13
	11	1261
6	9	533
	12	1183
7	7	78
	10	650
	13	988
8	8	91
	11	1196
9	9	234
	12	481
10	10	156
	13	130
11	11	78
12	12	13
13	13	1
Total		8192

Fig. 3. Summary after the application to all possible initial configurations of a given size (N=13) of the different rule sequences

So, if the desired task is exactly the latter (lattice with at most two consecutive 0s, and without isolated 0s and pairs of 0s simultaneously), the goal has been achieved, demonstrating the equivalence of these alternatives.

We should remember that the solution to the MOD3 alternates the sequence of rules that operate on the 0s with the sequence of rules that operate on the 1s, alternating the size of the bit strings with only 0s and 1s, until achieving the required condition for elementary rule 254 to finalise the solution.

Therefore, applying rule M_3^0 instead of R_3^0 and G_1^0, and rule M_3^1 instead of R_3^1 and G_1^1, the solution S_3 is simplified to Ss_3, using only three rules, instead of five, as originally (according to [1]):

$$S_3 = E_{254}^{\lfloor \frac{N}{2} \rfloor} \left(G_1^{1\lfloor \frac{N}{2} \rfloor} R_3^{1\lfloor \frac{N}{3} \rfloor} G_1^{0\lfloor \frac{N}{2} \rfloor} R_3^{0\lfloor \frac{N}{3} \rfloor} \right)^{\lceil \frac{N}{3} \rceil} \sigma \quad (2)$$

$$Ss_3 = E_{254}^{\lfloor \frac{N}{2} \rfloor} \left(M_3^{1N} M_3^{0N} \right)^{\lfloor \frac{N}{3} \rfloor} \sigma. \quad (3)$$

Disregarding the action of elementary rule 254 in the previous expressions, and letting S_3' and Ss_3' denote the remaining (partial) and simplified (partial) solutions, respectively, Figure 4 compares them, displaying a summary of all final configurations left after applying these two solutions to all possible initial configurations with N=16. The final configuration 1100000000000000 and all its equivalent configurations due to rotational symmetry are considered the same.

Original Sequence S_3'

Amount of 1s	Final Configuration with some Necklaces	No. Of Configs.
0	0000000000000000	21845
2	0000000000000011	19232
	0000000100000001	2608
8	0011001100110011	4
	0101010101010101	2
16	1111111111111111	21845
TOTAL		65536

Simplified Sequence Ss_3'

Amount of 1s	Final Configuration with some Necklaces	No. Of Configs.
0	0000000000000000	21845
2	0000000000000011	14576
	0000000100000001	656
8	0011001100110011	4
	0101010101010101	2
16	1111111111111111	21845
TOTAL		65536

Amount of 1s	Final Configuration with some Partial Necklaces	No. Of Configs.
2	0000000000000101	4768
	0000000000100001	1568
5	0000000101010101	144
	0000100001010101	64
	0000101000010101	64

Fig. 4. Summary after applying both partial solutions (original and simplified) to the MOD3 problem on all possible initial configurations of size 16

The simplified partial solution transforms some initial configurations into *partial necklaces* (i.e, configurations where only a part of it is a *necklace*), further to the *necklaces* that already had been transformed by the original partial solution.

Necklaces and *partial necklaces* appear only when MOD$n(N) \neq$ MOD$n(|\sigma|_1)$; otherwise, the lattice converges to 0^N or 1^N.

For the MOD3 problem, *necklaces* have the forms $(01)^8$ or $(0^21^2)^4$, and *partial necklaces* belong to $((0^3)^+(01)^+)^+$; they are trapped in these forms because after M_3^0 has transformed the three 0s into three 1s, only isolated 0s remain, that move synchronously, keeping the form $((1^3)^+(10)^+)^+$, until M_3^1 is applied, thus reversing the configuration to $((0^3)^+(01)^+)^+$.

For both alternatives, original and simplified partial solutions, any initial configurations where MOD$n(|\sigma|_1) = 0$ converges to 0^N and the problem is already solved, as defined. However, in order for all the other configurations with MOD$n(|\sigma|_1) \neq 0$ to end up in 1^N, elementary rule 254 is used, with no effect on the configuration 0^N.

4.3 The MOD4 Case and the Problem of Synchronised Displacements

Following the same rationale for merging rules in the optimised MOD3 solution (i.e., merging *Replacement* and *Grouping* rules that work on the same bit), in order to optimise the solution to MOD4 we have to merge one *Replacement* rule with two *Grouping* rules. For the 0-bit, M_4^0

results from merging rules R_4^0, G_1^0 and G_2^0, while for the 1-bit, M_4^1 results from merging rules R_4^1, G_1^1 and G_2^1. Rules R_4 and G_2 should have radius 3, at least. Therefore, the resulting M_4 rules should also have radius 3. We employ the '+' symbol for the joining of active transitions of the rules.

The problem in this merging is related to the *Grouping* rules. For instance, applying rules G_1^0 and G_2^0, separately or joined, without editing some active transitions, does not lead to the same outcome because of possible synchronised movements of isolated 0s and isolated pairs of 0s.

Figure 5 shows the joining of rules $R_{2,2}^0$, $\overleftarrow{G}_1^0$ and $\overleftarrow{G}_2^0$, and also shows the temporal evolution of an initial configuration where the problem occurs.

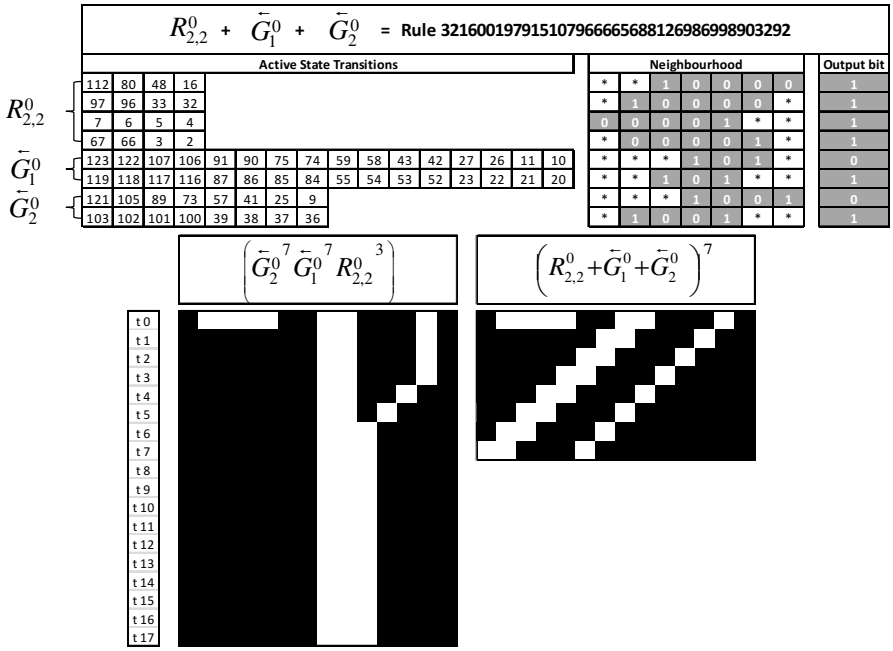


Fig. 5. Joining of rules $R_{2,2}^0$, $\overleftarrow{G}_1^0$ and $\overleftarrow{G}_2^0$, and the temporal evolution of the configuration 10000110011101 comparing two processing alternatives (joined or separate rules)

4.4 Removing and Inserting Active Transitions to Change the Displacement Step of the Rule

Hence, for the efficient merging of these three rules, we have to fix these synchronised movements caused by the *Grouping* rules. A first idea would be to exchange one of them by a shift to the right, compensating for the current shift to the left. However, looking ahead to the context

of the MOD5 problem, in order to optimise its solution with 3 *Grouping* rules, this method would no longer be sufficient.

Since we are handling larger radius than in the MOD3 problem, a more robust solution is to change the displacement step of rule $\overleftarrow{G}_1^0$: instead of moving the isolated 0 just one position to the left, we had better move it two positions, whenever possible (which is not always the case). The new rule with this feature is denoted $\overleftarrow{G}_1^{0,2}$. Figure 6 shows its 8 active transitions (125, 109, 93, 77, 61, 45, 29 and 13), but no longer the 8 others (123, 122, 91, 90, 59, 58, 27 and 26) that the previous rule had.

$\overleftarrow{G}_1^{0,2}$ = **Rule 297668273963817264613187722719825810176**

Active State Transitions															
119	118	117	116	87	86	85	84	55	54	53	52	23	22	21	20
		107	106			75	74			43	42			11	10
125	109			93	77			61	45			29	13		

Neighbourhood							Output bit
*	*	1	0	1	*	*	1
*	*	0	1	0	1	*	0
*	*	*	1	1	0	1	0

Fig. 6. Rule $\overleftarrow{G}_1^{0,2}$ represented by its active transitions

4.5 Removing Active State Transitions to Eliminate Remaining Synchronised Displacements

Finally, there is a further problem yet to be solved: when the isolated 0 cannot be moved 2 positions because of an isolated pair of 0s close to its left (as happens in the string *100101*), a synchronised displacement will continue to occur. Therefore, instead of transforming the string 100101 into 001011 (as would happen due to the modification just proposed), we had rather transform it into 000111.

To accomplish this some active state transitions should be turned off, as shown in Figure 7, indicated by two arrows. For clarity purposes of this process, notice that the string **0101* in Figure 6 has been instantiated into strings 000101*, 010101*, 100101* and 110101*, and that the string *1001** in Figure 5 gave rise to *100100, *100101, *100110 and *100111.

The active transitions that should be deactivated are 75 and 74 from rule $\overleftarrow{G}_1^{0,2}$, and 101 and 37 from rule $\overleftarrow{G}_2^0$. Rules $\overleftarrow{G}_1^{0,2}$ and $\overleftarrow{G}_2^0$ become $\overleftarrow{G}_{1*}^{0,2}$ and $\overleftarrow{G}_{2*}^0$, respectively, after some of their original active transitions have been deactivated (see Figure 7). Rule $\overleftarrow{M}_{2,2}^0$ is derived from joining $R_{2,2}^0$, $\overleftarrow{G}_{1*}^{0,2}$ and $\overleftarrow{G}_{2*}^0$.

The ideal merging of active state transitions to solve these conflicts is presented in Figure 8.

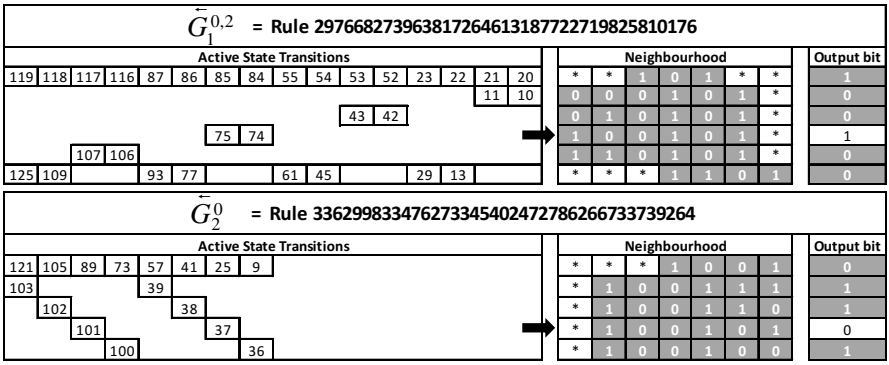


Fig. 7. Rules $\overleftarrow{G}_1^{0,2}$ and $\overleftarrow{G}_2^{0}$ represented by their active state transitions and pointing out those which have to be disabled (not highlighted output bit)

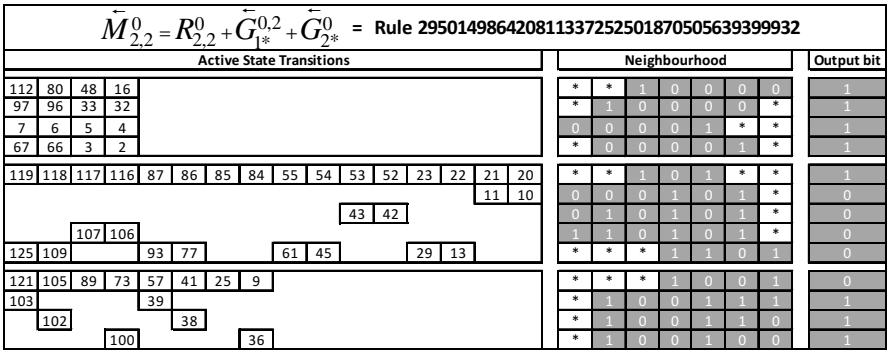


Fig. 8. Rule $\overleftarrow{M}_{2,2}^{0}$ represented by its active state transitions

The same rationale lead us to rule $\overleftarrow{M}_{2,2}^{1}$, composed by $R_{2,2}^{1}$, $\overleftarrow{G}_{1*}^{1,2}$ and $\overleftarrow{G}_{2*}^{1}$, whose Wolfram number is 255816358659918533639648036274123862084.

So, the solution S_4 is simplified to Ss_4, using only 3 rules instead of the 7 original rules:

$$S_4 = E_{254}^{\left\lfloor \frac{N}{2} \right\rfloor} \left(G_2^{1\left\lfloor \frac{N}{2} \right\rfloor} G_1^{1\left\lfloor \frac{N}{2} \right\rfloor} R_4^{1\left\lfloor \frac{N}{4} \right\rfloor} G_2^{0\left\lfloor \frac{N}{2} \right\rfloor} G_1^{0\left\lfloor \frac{N}{2} \right\rfloor} R_4^{0\left\lfloor \frac{N}{4} \right\rfloor} \right)^{\left\lceil \frac{N}{4} \right\rceil} \sigma \tag{4}$$

$$Ss_4 = E_{254}^{\left\lfloor \frac{N}{2} \right\rfloor} \left(M_4^{1N} M_4^{0N} \right)^{\left\lfloor \frac{N}{4} \right\rfloor} \sigma. \tag{5}$$

Testing the same procedure for MOD5 and obtaining similar results, we conclude that the simplified solution that can solve the MOD*n* problem can have only 3 rules, one elementary

(radius 1), and two others with radius n-1, as long as the size N of the binary string σ is not multiple of n neither multiple of any factor of n. This simplified solution then becomes:

$$\mathrm{Ss}_n = E_{254}^{\left\lfloor \frac{N}{2} \right\rfloor} \left(M_n^{1^N} M_n^{0^N} \right)^{\left\lfloor \frac{N}{n} \right\rfloor} \sigma. \tag{6}$$

5 Concluding Remarks

We demonstrated how a simple action, locally coordinated, represented here by one-dimensional cellular automata rules, can perform a solution to a complex global computation such as the MODn problem.

The design of rules to compute some task may start from analysing these tasks, and maybe dividing them into smaller instances. For each minor task, we can select a single or a group of active state transitions to execute it.

Different tasks performed by different CA rules, can be merged in a single rule, as long as there is no conflict among the individual tasks; this was the case of the task performed by the *Replacement* rules and the one performed by the *Grouping* rules in the MOD3 problem.

Much study is still necessary for the understanding and the generalisation of these mergings. However, in our work we could realise some evidence which might help further studies.

Rules classified with fixed point or null dynamical regimes according to [7] yield a predictable behaviour and render themselves, in some cases, to performing specific tasks. These rules are good candidates to have their active transitions joined or even separated, generating other rules that perform the same task. In contrast, rules with complex, chaotic or periodic dynamical behaviours are not candidates for these compositions or decompositions of their active transitions.

Even for the candidate rules, it is not always possible to merge their tasks into a single rule, especially when there is an overlap among the active transitions, or when two tasks performed simultaneously do not perform as required; this is what occurred, for instance, in our attempt to join the two *Grouping* rules, where one of them grouped isolated bits and the other grouped isolated pairs of bits in the MOD4 problem.

In order to merge rules for performing equivalent operation, we modified tasks and solved conflicts, following a standard that allowed us to generalise the MODn solution. As n increases, in order to solve the MOD5 problem, for instance, the number of synchronised displacements caused by merging *Grouping* rules also increases. So, in order to merge satisfactorily rules $G_{1*}^{0,3}$, $G_{2*}^{0,2}$ and G_{3*}^{0}, or rules $G_{1*}^{1,3}$, $G_{2*}^{1,2}$ and G_{3*}^{1}, we have to disable active transitions in the rule pairs involved in synchronised displacements of strings, as shown below (where G^{-} stands for either G^0 or G^1):

- 10001001 (or 01110110): G_{3*}^{-} and $G_{2*}^{-,2}$ (a single step for each rule);
- 1000101 (or 0111010): G_{3*}^{-} and $G_{1*}^{-,3}$ (a single step for each rule);
- 11001101 (or 00110010): $G_{2*}^{-,2}$ and $G_{1*}^{-,3}$ (2 steps for each rule);
- 0100101 (or 1011010): $G_{2*}^{-,2}$ and $G_{1*}^{-,3}$ (a single step for each rule); and
- 1100101 (or 0011010): $G_{2*}^{-,2}$ and $G_{1*}^{-,3}$ (2 steps for $G_{2*}^{-,2}$ and 1 for $G_{1*}^{-,3}$).

For the suitability of M_5^0 and M_5^1, we then have to disable 5 pairs of groups of active transitions. In general, this number depends on *n*, as exemplified in Figure 9.

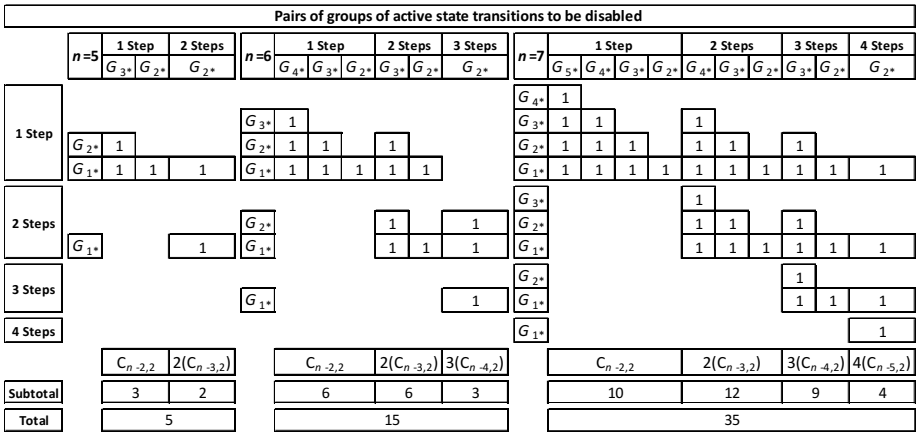


Fig. 9. Number of groups of active state transitions to be disabled according to *n*

The number of pairs of groups of active transitions to be disabled for each merging rule is $C_{n\text{-}2,2} + 2C_{n\text{-}3,2} + 3C_{n\text{-}4,2} + \ldots + (n\text{-}3)C_{2,2}$ which can be calculated through the 4th degree polynomial $((n\text{-}3)^4 + 6(n\text{-}3)^3 + 11(n\text{-}3)^2 + 6(n\text{-}3))/24$. All these conflicts can be eliminated, and the merging process can always be carried out.

Acknowledgements. We are grateful to CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, and IPM – Instituto Presbiteriano Mackenzie.

References

1. Martins, C.L.M., de Oliveira, P.P.B.: Computing Modulo-n by Composing Cellular Automata Rules. Under submission to Fundamenta Informaticae (2015)
2. Martins, C.L.M., de Oliveira, P.P.B.: Improvement of a result on sequencing elementary cellular automata rules for solving the parity problem. Electronic Notes Theoretical Computer Science **252**, 103–119 (2009)
3. Xu, H., Lee, K.M., Chau, H.F.: Modulo three problem with a cellular automaton solution. International Journal of Modern Physics C **14**(03), 249–256 (2003)
4. Lee, K.M., Xu, H., Chau, H.F.: Parity problem with a cellular automaton solution. Physical Review E **64**, 026702/1–026702/4 (2001)
5. Betel, H., de Oliveira, P.P.B., Flocchini, P.: Solving the parity problem in one-dimensional cellular automata. Natural Computing **12**(3), 323–337 (2013)
6. Wolfram, S.: A New Kind of Science, Wolfram Media (2002)
7. Li, W.: Parameterizations of Cellular Automata Rule Space. SFI Technical Report: Preprints, Santa Fe, NM, USA (1991)

Network Structure and Activity in Boolean Networks

Abhijin Adiga[1], Hilton Galyean[1,3], Chris J. Kuhlman[1], Michael Levet[1,2], Henning S. Mortveit[1,2(✉)], and Sichao Wu[1]

[1] Network Dynamics and Simulation Science Laboratory, VBI, Virginia Tech, Blacksburg, USA
{abhijin,ckuhlman,henning.mortveit,sichao}@vbi.vt.edu, {hgalyean,mlevet}@vt.edu
[2] Department of Mathematics, Virginia Tech, Blacksburg, USA
[3] Department of Physics, Virginia Tech, Blacksburg, USA

Abstract. In this paper we extend the notion of activity for Boolean networks introduced by Shmulevich and Kauffman (2004). Unlike the existing notion, we take into account the actual graph structure of the Boolean network. As illustrations, we determine the activity of all elementary cellular automata, and d-regular trees and square lattices where the vertex functions are bi-threshold and logical nor functions.

The notion of activity measures the probability that a perturbation in an initial state produces a different successor state than that of the original unperturbed state. Apart from capturing sensitive dependence on initial conditions, activity provides a possible measure for the significance or influence of a variable. Identifying the most active variables may offer insight into design of, for example, biological experiments of systems modeled by Boolean networks. We conclude with some open questions and thoughts on directions for future research related to activity.

Keywords: Boolean networks · Finite dynamical system · Activity · Sensitivity · Network · Sensitive dependence on initial conditions

1 Introduction

A Boolean network (BN) is a map of the form

$$F = (f_1, \ldots, f_n) \colon \{0,1\}^n \longrightarrow \{0,1\}^n \,. \tag{1}$$

BNs were originally proposed as a model for many biological phenomena [9,10], but have now been used to capture and analyze a range of complex systems and their dynamics [2]. Associated to F we have the *dependency graph of* F whose vertex set is $\{1, 2, \ldots, n\}$ and with edges all (i, j) for which the function f_i depends non-trivially on the variable x_j. See [8,14,17] for more general discussions of maps F.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 210–223, 2015.
DOI: 10.1007/978-3-662-47221-7_16

The study of stability and the response to perturbations of Boolean networks is central to increased understanding of their dynamical properties. Perturbations may take many forms, with examples including perturbations of the dependency graph [1,13,15], the vertex states [7,18,19], the vertex functions [20,21], or combinations of these.

This paper is concerned with noise applied to vertex states. Specifically, we want to know the following:

What is the probability that $F(x)$ and $F(x + e_i)$ are different?

Here i is a vertex while e_i is the i^{th} unit vector with the usual addition modulo 2. In [19], Shmulevich and Kauffman considered Boolean networks over regular graphs with $f_j = f$ for all vertices j, that is, a common vertex function. They defined the notion of *activity* of f with respect to its i^{th} argument as the expected value of the Boolean derivative of f with respect to its i^{th} variable. Under their assumptions, this may give a reasonable indication of the expected impact of perturbations to the i^{th} variable under the evolution of F. However, this approach does not consider the impact of the dependency graph structure. We remark that Layne et al. compute the activities of nested canalyzing functions given their canalyzing depth, extending results in [19] on canalyzing functions, see [12].

This question of sensitivity has also been studied when x is restricted to attractors in order to assess stability of long-term dynamics under state noise. The notion of threshold ergodic sets (TESs) is introduced in [16] and studied further in [11,13]. The structure of TESs capture long-term stability under state perturbations of periodic orbits and the resulting mixing between attractors that may happen as a result.

Other tools for analyzing sensitivity of vertex noise includes Lyapunov exponents, see for example [3,4], although this is perhaps mostly relevant or suited for the infinite case such as cellular automata over (infinite) regular lattices. Also, the notion of *Derrida diagrams* have been used to quantify how *Hamming classes* of states separate on average [5,6] after one or k transitions under F. Derrida diagrams, however, are mainly analyzed through numerical experiments via sampling. Moreover, analyzing how Hamming classes of large distance separate under F may not be so insightful – it seems more relevant to limit oneself to the case of nearby classes of vertex states.

Returning to the original question, we note that $F(x)$ and $F(x + e_i)$ may only differ in the coordinates j for which f_j depends on x_i. The answer to the question therefore depends on the vertex functions in the 1-neighborhood of i in the dependency graph X, and therefore the structure of the induced subgraph of the 2-neighborhood of i in X with the omission of edges connecting pairs of vertices both of distance 2 from i. We denote this subgraph by $X(i;2)$, see Figure 1.

To determine the activity of vertex i one will in general have to evaluate F over all possible states of $X(i;2)$, a problem which, in the general case, is computationally intractable. We will address three cases in this paper: the first

case is when $X(i;2)$ is a tree, the second case is that of elementary cellular automata (a special case of the former case), and the case where X is a regular, square, 2-dimensional lattice. Work for other graph classes is in progress and we comment on some of the challenges in the Summary section. Here we remark that a major source of challenges for analytic computations is the introduction of *type-3* and *type-4* edges as illustrated in Figure 1. Lack of symmetry adds additional challenges. The activity of a vertex can be evaluated analytically using the inclusion-exclusion principle, and type-3 and type-4 edges impact the complexity of the combinatorics.

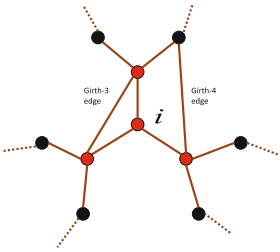


Fig. 1. The subgraph $X(i;2)$ of X induced by vertex i and its distance $\leqslant 2$ neighbors. Vertices belonging to the closed 1-neighborhood $n[i]$ of i are marked red. Type-3 edges (relative to i) connect neighbors of i, while type-4 edges connect neighbors of $j \in n'[i]$ through a common neighbor different from i. Here $n'[i]$ is $n[i]$ with i omitted. Edges connecting vertices of distance 2 from i in X do not belong to $X(i;2)$. For terminology we refer to Section 2.

A goal of the work on activity started here is as follows: when given a network X and vertex functions $(f_i)_i$, we would like to rank the vertices by activity in decreasing order. Just being able to identify for example the ten (say) vertices of highest activity would also be very useful. This information would allow one to identify the vertices for which state perturbations are most likely to produce different outcomes, at least in the short term. In a biological experiment for which there is a BN model of the form (1), one may then be able to allocate more resources to the measurement of such states, or perhaps ensure that these states are carefully controlled and locked at their intended values.

Paper Organization. After basic definitions and terminology in Section 2 we carefully define a new notion of activity denoted by $\bar{\alpha}_{F,i}$. Basic results to help in analytic evaluations of $\bar{\alpha}_{F,i}$ are given in Section 3 followed by specific results for the elementary cellular automata in Section 4, d-regular trees in Section 5, and then nor-BNs over square lattices in Section 6. A central goal is to relate the structure of $X(i;2)$ and the functions $(f_j)_j$ to $\bar{\alpha}_{F,i}$. We conclude with open questions and possible directions for followup work in Section 7.

2 Background, Definitions and Terminology

In this paper we consider the discrete dynamical systems of the form (1) where each map f_i is of the form $f_i\colon \{0,1\}^n \longrightarrow \{0,1\}$. However, f_i will in general depend nontrivially only on some subset of the variables $x_1, x_2, \ldots, x_n$, a fact that is captured by the dependency graph defined in the introduction and denoted by X_F or simply X when F is implied. The graph X is generally directed and will contain loops. We will, however, limit ourselves to undirected graphs.

Each vertex i has a *vertex state* $x_i \in \{0,1\}$ and a *vertex function* of the form $f_i\colon \{0,1\}^{d(i)+1} \longrightarrow \{0,1\}$ taking as arguments the states of vertices in the 1-neighborhood of i in X. Here $d(i)$ is the degree of vertex i. We write $n[i]$ for the ordered sequence of vertices contained in the 1-neighborhood of i (with i included) and $x[i]$ for the corresponding sequence of vertex states. We denote the *system state* by $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

We will write the evaluation of F in (1) as

$$F(x) = \big(f_1(x[1]), f_2(x[2]), \ldots, f_n(x[n])\big) \ .$$

The phase space of the map F in (1) is the directed graph $\Gamma(F)$ with vertex set $\{0,1\}^n$ and directed edges all pairs $(x, F(x))$. A state on a cycle in $\Gamma(F)$ is called a *periodic point* and a state on a cycle of length one is a *fixed point*. The sets of all such points are denoted by $\mathrm{Per}(F)$ and $\mathrm{Fix}(F)$ respectively. All other states are *transient states*. Since $\{0,1\}^n$ is finite, the phase space of F consists of a collection of oriented cycles (called periodic orbits), possibly with directed trees attached at states contained on cycles.

In this paper we analyze *short-term stability of dynamics* through the function $\alpha_{F,i}\colon K^n \longrightarrow \{0,1\}$ defined by

$$\alpha_{F,i}(x) = \mathbb{I}[F(x + e_i) \neq F(x)] \tag{2}$$

where $\mathbb{I}$ is the indicator function and e_i is the i^{th} unit vector. In other words, $\alpha_{F,i}(x)$ measures if perturbing x by e_i results in a different successor state under F than $F(x)$.

Definition 1. *The* activity of F with respect to vertex i *is the expectation value of* $\alpha_{F,i}$ *using the uniform measure on* K^n*:*

$$\bar{\alpha}_{F,i} = \mathbb{E}[\alpha_{F,i}] \ . \tag{3}$$

The activity of F *is the vector*

$$\bar{\alpha}_F = (\bar{\alpha}_{F,1}, \bar{\alpha}_{F,2}, \ldots, \bar{\alpha}_{F,n}) \ , \tag{4}$$

while the sensitivity of F *is the average activity* $\bar{\alpha} = \sum_{i=1}^{n} \bar{\alpha}_{F,i}/n$.

For a randomly chosen state $x \in K^n$, the value $\bar{\alpha}_{F,i}$ may be interpreted as the probability that perturbing x_i will cause $F(x+e_i) \neq F(x)$ to hold. This activity

notion may naturally be regarded as a measure of sensitivity with respect to initial conditions.

From (2) it is clear that $\bar{\alpha}_{F,i}$ depends on the functions f_j with $j \in n[i]$ and the structure of the distance-2 subgraph $X(i;2)$, see Figure 1. The literature (see, e.g. [12,19]) has focused on a very special case when considering activity. Rather than considering the general case and Equation (2), they have focused on the case where X is a regular graph where each vertex has degree d and all vertices share a common vertex function $f\colon K^d \longrightarrow K$. In this setting, activity is defined with respect to f and its i^{th} argument, that is, as the expectation value of the function $\mathbb{I}[f(x+e_i) \neq f(x)]$ where $x \in K^{d+1}$. Clearly, this measure of activity is always less than or equal to $\mathbb{E}[\alpha_{F,i}]$. Again, we note that this simpler notion of activity does not account for the network structure of $X(i;2)$.

3 Preliminary Results

For the evaluation of $\bar{\alpha}_{F,i}$ we introduce some notation. In the following we set $K = \{0,1\}$, write N_i for the size of $X(i;2)$, and $K(i) = K^{N_i}$ for the projection of K^n onto the set of vertex states associated to $X(i;2)$. For $j \in n[i]$, define the sets $A_j(i) \subset K(i)$ by

$$A_j(i) = \{x \in K(i) \mid F(x+e_i)_j \neq F(x)_j\} \,.$$

These sets appear in the evaluation of $\bar{\alpha}_{F,i}$, see Proposition 1. For convenience, we also set

$$A_j^m(i) = \{x \in A_j(i) \mid x_i = m\} \,,$$

for $m = 0, 1$. We write $\bar{A}_j(i) = A_j^0(i)$ and $\binom{n}{k}$ for binomial coefficients using the convention that it evaluates to zero if either $k < 0$ or $n - k < 0$.

The following proposition provides a somewhat simplified approach for evaluating $\bar{\alpha}_{F,i}$ in the general case.

Proposition 1. *Let X be a graph and F a map over X as in (1). The activity of F with respect to vertex i is*

$$\bar{\alpha}_{F,i} = \Pr\Big(\bigcup_{j\in n[i]} A_j \mid x_i = 0\Big) = \Pr\Big(\bigcup_{j\in n[i]} \bar{A}_j\Big) \,. \tag{5}$$

Proof. As stated earlier, we can write

$$\bar{\alpha}_{F,i} = \mathbb{E}[\alpha_{F,i}] = \sum_{x\in K^n} \alpha_{F,i}(x)\Pr(x) = \sum_{x\in K(i)} \alpha_{F,i}(x)\Pr(x) = \Pr\Big[\bigcup_{j\in n[i]} A_j\Big] \,,$$

where probabilities in the first and second sum are in K^n and $K(i)$, respectively. This follows by considering the relevant cylinder sets. Equation (5) follows by conditioning on the possible states for x_i. Since we are in the Boolean case, we have a bijection between each pair of sets A_j^0 and A_j^1 for $j \in n[i]$ which immediately allows us to deduce Equation (5).

As an example, consider the complete graph $X = K_n$ with threshold functions at every vertex. Recall that the standard Boolean *threshold function*, denoted by $\tau_{k,n}\colon K^n \longrightarrow K$, is defined by

$$\tau_{k,n}(x_1,\ldots,x_n) = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} x_j \geqslant k, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here we have

$$\bar{\alpha}_{F,i} = \binom{n-1}{k-1} \Big/ 2^{n-1} \ .$$

To see this, note first that all the sets $\bar{A}_j(i)$ are identical. For a state x with $x_i = 0$ to satisfy $\tau_k(x) \neq \tau_k(x+e_i)$ it is necessary and sufficient that x belong to Hamming class $k-1$. Since $x_i = 0$, it follows from Proposition 1 that $\Pr(\bar{A}_j(i)) = \frac{1}{2^{n-1}}\big|A_j\big| = \binom{n-1}{k-1}/2^{n-1}$ as stated.

Next, consider the Boolean nor-function $\mathrm{nor}_m\colon K^m \longrightarrow K$ defined by

$$\mathrm{nor}_m(x_1,\ldots,x_m) = (1+x_1)\cdot\cdots\cdot(1+x_m) \ , \tag{7}$$

with arithmetic operations modulo 2. If we use the nor-function over K_n we obtain

$$\bar{\alpha}_{F,i} = 1/2^{n-1} \ .$$

Again, $\bar{A}_j = \bar{A}_k$ for all $j,k \in n[i]$ and we have $F(x+e_i)_i \neq F(x)_i$ precisely when $x_j = 0$ for all $j \in n(i)$ leading to $\Pr\left(\bar{A}_i\right) = \frac{1}{2^{n-1}}$. We record the previous two results as a proposition:

Proposition 2. *If F is the GDS map induced by the* nor*-function over K_n, then*

$$\bar{\alpha}_{F,i} = \frac{1}{2^{n-1}},$$

and if F is induced by the k-threshold function over K_n, then

$$\bar{\alpha}_{F,i} = \binom{n-1}{k-1} \Big/ 2^{n-1} \ .$$

In the computations to follow, we will frequently need to evaluate the probability of the union of the A_j's. For this, let B denote the union of A_j's for all $j \neq i$. We then have

$$\Pr\left[\bigcup_{j\in n[i]} A_j\right] = \Pr(A_i \cup B) = \Pr(B) + \Pr(A_i \cap B^c) = 1 - \Pr(B^c) + \Pr(A_i \cap B^c) \ , \tag{8}$$

where B^c denotes the complement of B.

4 Activity of Elementary Cellular Automata

The evaluation of Equation (5) can often be done through the inclusion-exclusion principle. We demonstrate this in the context of elementary cellular automata (ECA). This also makes it clear how the structure of X comes into play for the evaluation of $\bar{\alpha}_{F,i}$.

Let F be the ECA map over $X = \text{Circle}_n$ with vertex functions given by $f\colon \{0,1\}^3 \longrightarrow \{0,1\}$. We will assume that $n \geqslant 5$; the case $n = 3$ corresponds to the complete graph K_3 and the case $n = 4$ can be done quite easily. Here we have

$$\bar{\alpha}_{F,i} = \Pr\big(\bar{A}_{i-1}(i) \cup \bar{A}_i(i) \cup \bar{A}_{i+1}(i)\big) \ .$$

Applying the definitions,

$$\bar{A}_j(i) = \big\{x = (x_{i-1}, x_{i-1}, x_i = 0, x_{i+1}, x_{i+2}) \mid \text{ and } f\big(x[j]\big) \neq f\big((x + e_i)[j]\big)\big\} \tag{9}$$

for $j \in n[i] = \{i-1, i, i+1\}$ with indices modulo n.

Proposition 3. *The activity for k-threshold ECA is*

$$\bar{\alpha}_{F,i} = \begin{cases} 0, & \text{if } k = 0 \text{ or } k > 3 \ , \\ 1/2, & \text{if } k = 1 \text{ or } k = 3 \ , \\ 7/8, & \text{if } k = 2 \ . \end{cases}$$

Proof. Clearly, for $k = 0$ and $k > 3$ we always have $F(x + e_i) = F(x)$ so in these cases it follows that $\bar{\alpha}_{F,i} = 0$. The cases $k = 1$ and $k = 3$ are symmetric, and, using $k = 1$, we have

$$\begin{aligned} \bar{A}_{i-1} &= \big\{(0,0,0,x_{i+1},x_{i+2})\big\}, \\ \bar{A}_i &= \big\{(x_{i-2},0,0,0,x_{i+2})\big\}, \text{ and} \\ \bar{A}_{i+1} &= \big\{(x_{i-1},x_{i-1},0,0,0)\big\} \ . \end{aligned}$$

By the inclusion-exclusion principle, it follows that

$$\begin{aligned} \big|\bar{A}_{i-1} \cup \bar{A}_i \cup \bar{A}_{i+1}\big| &= \big|\bar{A}_{i-1}\big| + \big|\bar{A}_i\big| + \big|\bar{A}_{i+1}\big| \\ &\quad - \big|\bar{A}_{i-1} \cap \bar{A}_i\big| - \big|\bar{A}_{i-1} \cap \bar{A}_{i+1}\big| - \big|\bar{A}_i \cap \bar{A}_{i+1}\big| \\ &\quad + \big|\bar{A}_{i-1} \cap \bar{A}_i \cap \bar{A}_{i+1}\big| \\ &= 3 \times 4 - 2 - 1 - 2 + 1 = 8 \ . \end{aligned}$$

This yields $\bar{\alpha}_{F,i} = 8/2^4 = 1/2$. The proof for the case $k = 2$ is similar to that of $k = 1$ so we leave this to the reader.

Remark 1. If we instead use the nor-function, then $\bar{\alpha}_{F,i} = 1/2$ when $n \geqslant 5$.

Remark 2. In [19], activity is defined with respect to f instead of F and its i^{th} argument. With this context, $\bar{\alpha}_{f,i}(x) = \mathbb{E}\Big[\mathbb{I}[f(x + e_i) \neq f(x)]\Big]$. Clearly, we always have $\bar{\alpha}_{f,i} \leqslant \bar{\alpha}_{F,i}$. As an example, for circle graph of girth $\geqslant 5$ and threshold-1 functions it can be verified that $\bar{\alpha}_{f,i} = 0.25$ and we have already shown that $\bar{\alpha}_{F,i} = 0.5$ in this case.

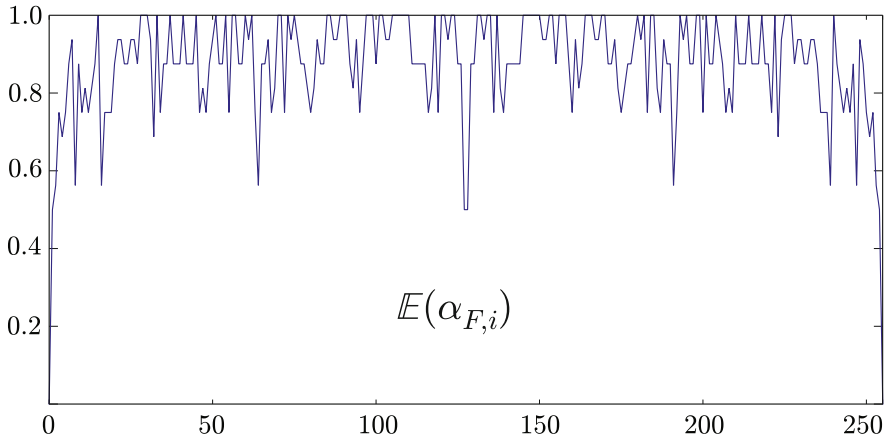


Fig. 2. Activity of ECA by rule number. With the exception of the constant rules of activity 0, all rules have activity $\geqslant 1/2$ with most rules exceeding 3/4.

5 Activity over *d*-Regular Trees

The case of d-regular trees is natural as a starting point. Here the sets A_j (or more precisely, the sets $n[j]$) with $j \neq i$ only overlap at vertex i. As a result, when we condition on $x_i = m$, the resulting sets are independent. More generally, this will hold if the girth of the graph is at least 5.

The Boolean bi-threshold function $\tau_{i,k_{01},k_{10},n} \colon K^n \longrightarrow K$ generalizes standard threshold functions and is defined by

$$\tau_{i,k_{01},k_{10},n}(x_1,\ldots,x_n) = \begin{cases} 1, & \text{if } x_i = 0 \text{ and } \sum_{j=1}^{n} x_j \geqslant k_{01} \\ 0, & \text{if } x_i = 1 \text{ and } \sum_{j=1}^{n} x_j < k_{10} \\ x_i, & \text{otherwise,} \end{cases} \tag{10}$$

where the integers k_{01} and k_{10} are the up- and down-thresholds, respectively. Here i is a designated vertex – it will be the index of a vertex function. If the up- and down-thresholds are the same we get standard threshold systems.

Proposition 4. *Let X be a d-regular graph of girth $\geqslant 5$, and F the GDS map over X with the bi-threshold vertex functions as in Equation (10). Then the activity of F with respect to vertex i is given by*

$$\begin{aligned} \bar{\alpha}_{F,i} =& 1 - \left[\frac{2^d - \binom{d-1}{k_{01}-1} - \binom{d-1}{k_{10}-2}}{2^d}\right]^d + \\ & \frac{1}{2} \sum_{k=k_{01},k_{10}} \frac{\binom{d}{k-1}}{2^d} \left[\frac{2^{d-1} - \binom{d-1}{k_{10}-2}}{2^{d-1}}\right]^{k-1} \left[\frac{2^{d-1} - \binom{d-1}{k_{01}-1}}{2^{d-1}}\right]^{d-(k-1)} . \end{aligned} \tag{11}$$

Proof. Let B denote the union of A_j's for all $j \neq i$ as in (8) and note that B^c is the event that none of the A_j's occur for $j \neq i$. Using girth $\geqslant 5$ and the resulting independence from conditioning on x_i we can write

$$\begin{aligned}\Pr(B^c) &= \frac{1}{2}\Pr(B^c \mid x_i = 0) + \frac{1}{2}\Pr(B^c \mid x_i = 1) \\ &= \frac{1}{2}\Big(\bigcap_{\substack{j\in n[i]\\ j\neq i}} A_j^c \mid x_i = 0\Big) + \frac{1}{2}\Big(\bigcap_{\substack{j\in n[i]\\ j\neq i}} A_j^c \mid x_i = 1\Big) = \Big(\bigcap_{\substack{j\in n[i]\\ j\neq i}} A_j^c \mid x_i = 0\Big) \\ &= \prod_{\substack{j\in n[i]\\ j\neq i}} \Pr\big(A_j^c \mid x_i = 0\big)\ .\end{aligned} \tag{12}$$

Next we have

$$\begin{aligned}\Pr\big(A_j^c \mid x_i = 0\big) &= \frac{1}{2}\Pr\big(A_j^c \mid x_i = 0 \text{ and } x_j = 0\big) + \frac{1}{2}\Pr\big(A_j^c \mid x_i = 0 \text{ and } x_j = 1\big) \\ &= \frac{1}{2}\cdot\frac{2^{d-1} - \binom{d-1}{k_{01}-1}}{2^{d-1}} + \frac{1}{2}\cdot\frac{2^{d-1} - \binom{d-1}{k_{10}-2}}{2^{d-1}} \\ &= [2^d - \binom{d-1}{k_{01}-1} - \binom{d-1}{k_{10}-2}]/2^d\ ,\end{aligned}$$

which substituted into Equation (12) yields

$$\Pr(B^c) = \left[\frac{2^d - \binom{d-1}{k_{01}-1} - \binom{d-1}{k_{10}-2}}{2^d}\right]^d . \tag{13}$$

Next, we compute the probability of $A_i \cap B^c$ as

$$\begin{aligned}\Pr(A_i \cap B^c) &= \frac{1}{2}\Pr(A_i \cap B^c \mid x_i = 0) + \frac{1}{2}\Pr(A_i \cap B^c \mid x_i = 1) \\ &= \frac{1}{2}\Pr(A_i \mid x_i = 0)\cdot\Pr(B^c \mid A_i \text{ and } x_i = 0) \\ &\quad + \frac{1}{2}\Pr(A_i \mid x_i = 1)\cdot\Pr(B^c \mid A_i \text{ and } x_i = 1) \\ &= \frac{\binom{d}{k_{01}-1}}{2^{d+1}}\left[\frac{2^{d-1} - \binom{d-1}{k_{10}-2}}{2^{d-1}}\right]^{k_{01}-1}\left[\frac{2^{d-1} - \binom{d-1}{k_{01}-1}}{2^{d-1}}\right]^{d-(k_{01}-1)} \\ &\quad + \frac{\binom{d}{k_{10}-1}}{2^{d+1}}\left[\frac{2^{d-1} - \binom{d-1}{k_{10}-2}}{2^{d-1}}\right]^{k_{10}-1}\left[\frac{2^{d-1} - \binom{d-1}{k_{01}-1}}{2^{d-1}}\right]^{d-(k_{10}-1)} \\ &= \frac{1}{2}\sum_{k=k_{01},k_{10}} \frac{\binom{d}{k-1}}{2^d}\left[\frac{2^{d-1} - \binom{d-1}{k_{10}-2}}{2^{d-1}}\right]^{k-1}\left[\frac{2^{d-1} - \binom{d-1}{k_{01}-1}}{2^{d-1}}\right]^{d-(k-1)} .\end{aligned} \tag{14}$$

Substituting Equation (13) and (14) into Equation (8) leads to Equation (11), which ends the proof.

Since the standard threshold function is a special case of the bi-threshold function when k_{01} and k_{10} coincide, it is straightforward to obtain following corollary.

Corollary 1. *If F is the GDS map with k-threshold vertex functions over a d-regular graph of girth $\geqslant 5$, then the activity of F with respect to vertex i is*

$$\bar{\alpha}_{F,i} = 1 - \left[\frac{2^d - \binom{d}{k-1}}{2^d}\right]^d + \frac{\binom{d}{k-1}}{2^d}\left[\frac{2^{d-1} - \binom{d-1}{k-2}}{2^{d-1}}\right]^{k-1}\left[\frac{2^{d-1} - \binom{d-1}{k-1}}{2^{d-1}}\right]^{d-(k-1)} . \tag{15}$$

We next consider the nor-function.

Proposition 5. *Let X be a d-regular graph of girth $\geqslant 5$ and F the GDS map over X induced by the* nor*-function. Then the activity of F with respect to i is given by*

$$\bar{\alpha}_{F,i} = 1 - \left(1 - \frac{1}{2^d}\right)^d + \left(\frac{1}{2} - \frac{1}{2^d}\right)^d . \tag{16}$$

Proof. Conditioning on $x_i = 0$ and using independence we have

$$\Pr\left(B^c\right) = \Pr\left(\bigcap_{j\in n(i)} A_j^c\right) = \prod_{j\in n(i)} \Pr\left(A_j\right) . \tag{17}$$

For a state x with $x_i = 0$ to be in $\bar{A}_j$ all remaining d states of $\bar{A}_j$ must be zero leading to $\Pr\left(\bar{A}_j^c\right) = 1 - \frac{1}{2^d}$ and

$$\Pr\left(\bar{B}^c\right) = \left(1 - \frac{1}{2^d}\right)^d . \tag{18}$$

In order to calculate $\Pr\left(A_i \cap B^c\right)$ we note that $\Pr\left(A_i \cap B^c\right) = \Pr\left(A_i\right)\Pr\left(B^c|A_i\right)$ and again use independence to obtain

$$\Pr\left(\bar{B}^c|\bar{A}_i\right) = \prod_{j\in n(i)} \Pr\left(\bar{A}_j^c|\bar{A}_i\right) . \tag{19}$$

Note that $\Pr\left(\bar{A}_j|\bar{A}_i\right) = 1/2^{d-1}$ so that $\Pr\left(\bar{A}_j^c|\bar{A}_i\right) = 1 - \Pr\left(\bar{A}_j|\bar{A}_i\right) = 1 - 1/2^{d-1}$ which substituted into (19) gives

$$\Pr\left(\bar{B}^c|\bar{A}_i\right) = \left(1 - \frac{1}{2^{d-1}}\right)^d .$$

Noting that $\Pr(\bar{A}_i) = 1/2^d$ we obtain the third term in (15), finishing the proof.

6 Square Lattices

In this section we consider graphs with type-4 edges or girth 4. Here the 1-neighborhoods $n[j]$ (with $j \neq i$) may intersect, the key aspect we want to

address here. As an example, the reader may verify that for threshold-2 functions and Circle$_4$, the activity of any vertex is 3/4 and not 7/8 as when $n \geqslant 5$ in Proposition 3.

As a specific case, take as graph X a regular, square 2-dimensional lattice. It may be either infinite or with periodic boundary conditions. In the latter case, we assume for simplicity that its two dimensions are at least 5. This graph differs from the 4-regular tree by introduction of type-4 edges: sets A_j and A_{j+1} of A_i, when conditioned on the state of vertex i, are no longer independent. The graph has cycles of size 4 containing i. We illustrate this case using nor-functions as these allow for a somewhat simplified evaluation. We discuss more general cases and girth-3 graphs in the Summary section.

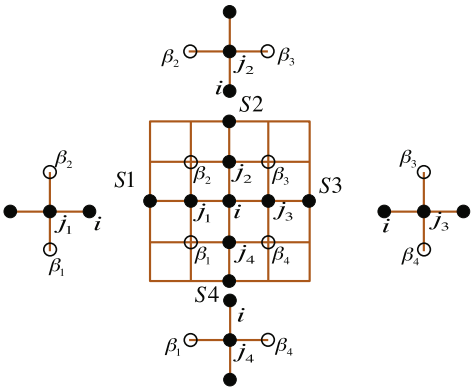


Fig. 3. The subgraph of a torus that includes $X(i;2)$ at center. The subgraphs S_l, each containing the vertices associated to an A_l, $l \in \{1,2,3,4\}$, are also shown separated from the center torus, with "center" vertex j_ℓ. The overlapping or common vertices, β_ℓ, are also denoted. We have vectors $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)$ and $\hat{x}_j = (x_{j_1}, x_{j_2}, x_{j_3}, x_{j_4})$, where x_{j_ℓ} is the state of the "center" vertex j_ℓ of A_ℓ, and where β_ℓ is a common vertex in A_ℓ and $A_{\ell+1}$ (ℓ is always modulo 4).

Proposition 6. *Let X be the 2-dimensional lattice as above where every vertex has degree 4, and let F be the GDS map over X induced by* nor*-functions. Then the activity of F for any vertex i is $\bar{\alpha}_{F,i} = 1040/2^{12}$.*

Proof. The neighborhoods of i involved are illustrated in Figure 3. We rewrite (8) as follows

$$\Pr\left(\bigcup_{j\in n[i]} A_j\right) = \Pr(B) + \Pr(A_i)\left(1 - \Pr(B|A_i)\right) \ . \tag{20}$$

To determine $|B|$, refer to Figure 3. With indices $p, q, r, s \in \{1, 2, 3, 4\}$ we have

$$|A_1 \cup A_2 \cup A_3 \cup A_4| = \sum_{p=1}^{4} A_p - \sum_{p<q} |A_p \cap A_q| + \sum_{p<q<r} |A_p \cap A_q \cap A_r| - \sum_{p<q<r<s} |A_p \cap A_q \cap A_r \cap A_s| \,, \tag{21}$$

which, along with Equation (20), is also valid if we condition on $x_i = 0$ in every set. We take $i \neq j_\ell$ with $\ell \in \{1, 2, 3, 4\}$ and let j_ℓ denote the vertex at the center of $n[j_\ell]$ and i the vertex at the center of $n[i]$. Here $|X(i; 2)| = 13$. Using the symmetry of the sets $\bar{A}_{j_\ell}$ and the fact that a state x is in the set $\bar{A}_{j_\ell}$ precisely when $x[j_\ell] = 0$ leads to

$$\sum_{\ell=1}^{4} |\bar{A}_{j_\ell}| = 4 \cdot 2^8 \,.$$

For the second term on the right in (21) we have two cases for intersections: (i) two adjacent vertices $j_\ell, j_{\ell+1} \in n'[i]$ with all indices modulo 4, of which there are four instances, and (ii) two non-adjacent vertices $j_\ell, j_{\ell+2} \in n'[i]$, of which there are two instances. In the first case we obtain $|\bar{A}_{j_\ell} \cap \bar{A}_{j_{\ell+1}}| = 2^5$, while in the second case we have $|\bar{A}_{j_\ell} \cap \bar{A}_{j_{\ell+2}}| = 2^4$, making the second term on the right in (21)

$$\sum_{p<q} |\bar{A}_{j_p} \cap \bar{A}_{j_q}| = 4 \cdot 2^5 + 2 \cdot 2^4 \,. \tag{22}$$

The third sum appearing on the right in (21) contains four terms all corresponding to an intersection of three sets A_{j_ℓ}, and we obtain

$$\sum_{p<q<r} |\bar{A}_{j_p} \cap \bar{A}_{j_q} \cap \bar{A}_{j_r}| = 4 \cdot 2^2 \,. \tag{23}$$

Finally, the intersection of all four sets $\bar{A}_{j_\ell}$ has size 1, and we get

$$\Pr(\bar{B}) = |\bar{B}|/2^{12} = \left[4 \cdot 2^8 - (4 \cdot 2^5 + 2 \cdot 2^4) + 4 \cdot 2^2 - 1\right]/2^{12} \,. \tag{24}$$

We next determine $\Pr(B|A_i)$. For a state x to be in A_i we must have $x[i] = 0$ so that $|A_i| = 2^8$. Conditioning on this, we can calculate $|\bigcup_{\ell=1}^{4} A'_{j_\ell}|$ as above. The reader can verify that

$$\begin{aligned} \sum_{\ell=1}^{4} |A'_{j_\ell}| &= 4 \cdot 2^5 \,, \\ \sum_{p<q} |A'_{j_p} \cap A'_{j_q}| &= 4 \cdot 2^3 + 2 \cdot 2^2 \,, \\ \sum_{p<q<r} |A'_{j_p} \cap A'_{j_q} \cap A'_{j_r}| &= 4 \cdot 2 \,, \\ |A'_{j_1} \cap A'_{j_2} \cap A'_{j_3} \cap A'_{j_4}| &= 1 \,, \end{aligned} \tag{25}$$

leading to the expression

$$\Pr\left(\bar{A}_i \cap \bar{B}^c\right) = \Pr\left(\bar{A}_i\right)\left(1 - \Pr\left(\bar{B}|\bar{A}_i\right)\right) = \frac{1}{2^4}\left(1 - \frac{1}{2^8}\left[2^7 - 2^5 - 2^3 + 2^3 - 1\right]\right) ,$$

which, together with (24) in (20) gives the stated result of $\bar{\alpha}_{F,i} = 1040/2^{12}$.

7 Summary and Research Directions

We have introduced an extension of the notion of activity proposed by Shmulevich and Kauffman [19]. This extension takes into account the impact of the network structure when studying $\mathbb{E}\Big[\mathbb{I}[F(x) \neq F(x + e_i)]\Big]$, which estimates how likely the perturbation e_i will cause successor states to diverge. Naturally, orbits that initially separate may later converge. Nonetheless, this notion of activity provides a measure for sensitivity with respect to initial conditions.

Possibly interesting avenues for further work includes studying asynchronous systems. If the vertex functions are applied sequentially according to for example a permutation update sequence, are there effective ways of relating activity and the permutation? If so, are there principles connecting network structure and update sequence that allows one to minimize or maximize the activity of one or more vertices?

Square grids have type-4 edges and cycles of length 4 involving the vertex i. If we permit more general graphs with type-3 edges we naturally obtain cycles of length 3 containing i. The notion of cluster coefficient quantifies the number of triangles incident to i. Is it possible to relate activity and cluster coefficient? Of course, cluster coefficient is solely a graph property and includes nothing about vertex functions. Can one still find such a relation for a specific choice of vertex functions?

Finally, we considered arbitrary initial states $x \in K^n$. What can be said about activity if we restrict x to be a periodic point? We invite the reader to explore this – some initial results on attractor activity are given in [11].

Acknowledgments. We thank our external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. This work has been partially supported by DTRA Grant HDTRA1-11-1-0016.

References

1. Adiga, A., Kuhlman, C., Mortveit, H.S., Vullikanti, A.K.S.: Sensitivity of diffusion dynamics to network uncertainty. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)
2. Aldana, M., Coppersmith, S., Kadanoff, L.P.: Boolean dynamics with random couplings. In: Perspectives and Problems in Nolinear Science, pp. 23–89. Springer (2003)

3. Baetens, J.M., Van der Weeën, P., De Baets, B.: Effect of asynchronous updating on the stability of cellular automata. Chaos, Solitons & Fractals **45**, 383–394 (2012)
4. Baetens, J.M., De Baets, B.: Phenomenological study of irregular cellular automata based on lyapunov exponents and jacobians. Chaos **20**, 1–15 (2010)
5. Derrida, B., Pomeau, Y.: Random networks of automata: A simple annealed approximation. Europhysics Letters **1**, 45–49 (1986)
6. Fretter, C., Szejka, A., Drossel, B.: Perturbation propagation in random and evolved boolean networks. New Journal of Physics **11**, 1–13 (2009)
7. Ghanbarnejad, F., Klemm, K.: Impact of individual nodes in boolean network dynamics. EPL (Europhysics Letters) **99**(5), 58006 (2012)
8. Goles, E., Martinez, S.: Neural and Automata Networks: Dynamical Behaviour and Applications. Kluwer Academic Publishers (1990)
9. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology **22**, 437–467 (1969)
10. Kauffman, S.A.: The origins of order: Self-organization and selection in evolution. Oxford University Press (1993)
11. Kuhlman, C.J., Mortveit, H.S.: Attractor stability in nonuniform boolean networks (in press, 2014). Published online August 23, 2014. Special volume of TCS: Nonuniform Cellular Automata
12. Layne, L., Dimitrova, E., Matthew, M.: Nested canalyzing depth and network stability. Bulletin of Mathematical Biology, 74 (2012)
13. Luo, J.X., Turner, M.S.: Evolving sensitivity balances boolean networks. PLoS One **7**, e36010 (2012)
14. Mortveit, H.S., Reidys, C.M.: An Introduction to Sequential Dynamical Systems. Universitext. Springer (2007)
15. Pomerance, A., Ott, E., Girvan, M., Losert, W.: The effect of network topology on the stability of discrete state models of genetic control. Proceedings of the National Academy of Sciences **106**(20), 8209–8214 (2009)
16. Ribeiro, A.S., Kauffman, S.A.: Noisy attractors and ergodic sets in models of gene regulatory networks. Journal of Theoretical Biology **247**, 743–755 (2007)
17. Robert, F.: Discrete Iterations. A Metric Study. No. 6 in Springer Series in Computational Mathematics. Springer (1986)
18. Serra, R., Villani, M., Barbieri, A., Kauffman, S., Colacci, A.: On the dynamics of random boolean networks subject to noise: attractors, ergodic sets and cell types. Journal of Theoretical Biology **265**(2), 185–193 (2010)
19. Shmulevich, I., Kauffman, S.A.: Activities and sensitivities in Boolean network models. Physical Review Letters **93**(4), 048701:1–048701:4 (2004)
20. Shmulevich, I., Lähdesmäki, H., Dougherty, E.R., Astola, J., Zhang, W.: The role of certain post classes in boolean network models of genetic networks. Proceedings of the National Academy of Sciences **100**(19), 10734–10739 (2003)
21. Xiao, Y., Dougherty, E.R.: The impact of function perturbations in boolean networks. Bioinformatics **23**(10), 1265–1273 (2007)

Group-Walking Automata

Ville Salo[1] and Ilkka Törmä[2(✉)]

[1] Center for Mathematical Modeling, University of Chile, Santiago, Chile
vsalo@dim.uchile.cl
[2] TUCS – Turku Centre for Computer Science, University of Turku, Turku, Finland
iatorm@utu.fi

Abstract. In the setting of symbolic dynamics on discrete finitely generated infinite groups, we define a model of multi-headed finite automata that walk on Cayley graphs, and use it to define subshifts. We characterize the torsion groups (also known as periodic groups) as those on which the group-walking automata are strictly weaker than Turing machines.

Keywords: Group-walking automaton · Torsion group · Periodic group · Multi-headed automaton · Subshift

1 Introduction

One of the central objects in symbolic dynamics is the dynamical system S^G (where G is a discrete group and S a finite alphabet), called the *full shift*, where G acts by translations. In particular, one studies its subsystems, usually called *subshifts*, and classes of such subsystems. Some of the important classes studied are the SFTs (subshifts defined by a finite set of forbidden patterns), sofic shifts (the factors of SFTs) and the effective, or Π_1^0 subshifts (defined by a recursively enumerable set of forbidden patterns). SFTs and sofic shifts are natural objects to study on all groups, and a robust notion of effectiveness of subshifts on arbitrary groups is given in [2] (see also Section 5).

In this paper, continuing the work in [9], we define some new families of subshifts on an arbitrary (discrete finitely generated infinite) group G. Namely, we discuss the class of subshifts defined by certain multi-headed automata that walk on the Cayley graph of the group G. We have studied the case $G = \mathbb{Z}^d$ in [9], the main result being that three-headed finite-state automata define the same subshifts as general Turing machines.[1] It turns out that up to notational complications and a few simple tricks, the same result can be shown on all groups containing a copy of $\mathbb{Z}$. We show this in Theorem 1.

Most finitely generated groups of practical interest contain a copy of $\mathbb{Z}$. For example, in addition to infinite (finitely generated) abelian groups, this is true for free groups, Baumslag-Solitar groups, the Heisenberg group, the Thompson groups F, T and V, and the general linear groups $GL(n, \mathbb{Z})$. In fact, infinite

[1] In the earlier article [4], essentially the same observation is made in a slightly different setting on the group $\mathbb{Z}^2$.

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 224–237, 2015.
DOI: 10.1007/978-3-662-47221-7_17

finitely generated groups without a copy of $\mathbb{Z}$, known as torsion groups, are quite rare and hard to construct. Nevertheless, many examples exist in the literature. The question is particularly hard in the case that the torsion is bounded, that is, there exists $n \in \mathbb{N}$ such that every element of the group generates a subgroup of order at most n. See [1] for a discussion of groups with bounded torsion. In the case of unbounded torsion, there are examples that are relatively simple to define, and simple to prove torsion. We mention in particular [5,6].

Given that such groups exist, an obvious question is whether we can extend Theorem 1 to this case. It turns out that we cannot: in Theorem 2 we show that a subshift on a torsion group accepted by a multi-headed automaton 'cannot be too sparse', and as a further result we obtain Theorem 6, which characterizes the torsion groups as those on which multi-headed automata are strictly weaker than Turing machines.

2 Definitions and Examples

2.1 Subshifts

In this section, we define some basic notions of of symbolic dynamics and computability. Some references on symbolic dynamics on general groups are [2,3], and a standard reference on $\mathbb{Z}$ is [8].

Let G be a group with identity element $1_G \in G$. Our groups are always infinite (the finite case being trivial) and finitely generated (since the notions we consider are local). For convenience, if G is finitely generated, we fix a symmetric finite set $s(G) \subset G$ of generators for it. The set $s(G)^*$ consists of all finite words over $s(G)$, and for $v, w \in s(G)^*$, we denote $v \sim w$ and $v \sim g$ if the words correspond to the same element $g \in G$. We denote by $B_G(n)$ the ball of radius n with respect to the fixed set of generators: $B_G(n) = \{g \in G \mid w \in s(G)^*, |w| \leq n, w \sim g\}$.

A *torsion element* of a group G is an element $g \in G$ that satisfies $g^n = 1_G$ for some $n \geq 1$. If all elements of G are torsions, then G is a *torsion group*. To each torsion element $g \in G$ we associate its *order* $t_G(g) = \min\{n \geq 1 \mid g^n = 1_G\}$, and to each finitely generated torsion group we associate the *torsion function* $T_G : \mathbb{N} \to \mathbb{N}$, defined by $T_G(n) = \max\{t_G(g) \mid g \in B_G(n)\}$. A non-torsion group, conversely, is one containing an isomorphic copy of $\mathbb{Z}$.

Both *alphabet* and *state set* mean any finite set. The symbol S always means an alphabet, and the set S^G is the *full G-shift over S*. Its elements, usually denoted by x, y, z, are called *configurations*. We define both a left and a right action of G on S^G, called the *left and right shifts*. The left action is given by $(g \cdot x)_h = x_{g^{-1}h}$. It is indeed an action because

$$(g_2 \cdot (g_1 \cdot x))_h = (g_1 \cdot x)_{g_2^{-1}h} = x_{g_1^{-1}g_2^{-1}h} = x_{(g_2g_1)^{-1}h} = (g_2g_1 \cdot x)_h.$$

The right action is given by $\sigma^R_g(x)_h = x_{hg}$. It is indeed an action because

$$\sigma^R_{g_2}(\sigma^R_{g_1}(x))_h = \sigma^R_{g_1}(x)_{hg_2} = x_{hg_2g_1} = \sigma^R_{g_2g_1}(x)_h.$$

We give S^G the product topology induced by the discrete topology on S, making it a compact metrizable space. It is easy to show that both actions are continuous in this topology. A *subshift* of S^G is a topologically closed subset of S^G closed under the left action of G. A *cellular automaton* on a subshift $X \subset S^G$ is a continuous map $f : X \to X$ that commutes with the left shifts in the sense that $g \cdot f(x) = f(g \cdot x)$ holds for all $x \in S^G$ and $g \in G$. We denote by $\mathrm{Aut}(X)$ the group of bijective cellular automata on X under composition.

For us, the main importance of the left action is that it allows for nice definitions of subshifts and cellular automata. On the other hand, when the right action of an element $g \in G$ is well-defined on a subshift, it is a cellular automaton. In particular, the right actions show that $\mathrm{Aut}(S^G)$ contains a copy of the group G, by the injective group homomorphism $g \mapsto \sigma_g^R$.

A *pattern (on G)* is a function $P \in S^D$, where $D = D(P)$ is a finite subset of G, called the *domain* of P. Each pattern P defines a *cylinder set* $[P] = \{x \in S^G \mid x|_D = P\}$. The clopen (topologically closed and open) sets in S^G are precisely the finite unions of cylinders, and form a basis for the topology. Subshifts can be characterized as sets $X \subset S^G$ for which there exists a set of *forbidden patterns* $\mathcal{F}$ such that

$$X = \{x \in S^G \mid \forall P \in \mathcal{F} : \forall g \in G : g \cdot x \notin [P]\}.$$

Each cellular automaton on X has a *radius* $r \in \mathbb{N}$ and a *local rule* $F : S^{B_G(r)} \to S$ such that $f(x)_g = F(g^{-1} \cdot x|_{B_G(r)})$ holds for all $x \in X$ and $g \in G$.

Example 1. Let G be the free group generated by $g, h \in G$, and $X \subset S^G$ the set

$$\{x \in S^G \mid \forall g \in G : \forall n \in \mathbb{Z} : x_{gh^n} = x_g\}.$$

We show that X is a subshift, and for that, let $x \in X$ and $g \in G$. We need to show $g \cdot x \in X$. Given $f \in G$ and $n \in \mathbb{Z}$, we have

$$(f \cdot x)_{gh^n} = x_{f^{-1}gh^n} = x_{f^{-1}g} = (f \cdot x)_g$$

by the definition of the left action, and the fact $x \in X$.

Definition 1. *If $S \ni 0$ is a finite alphabet, then the* one-S subshift *on a group G is the subshift $X_S^G \subset S^G$ where a finite pattern $P \in S^D$ is forbidden if and only if there exist $d \neq e \in D$ with $P_e \neq 0 \neq P_d$. If $0 \notin S$, we write $X_S^G = X_{S \cup \{0\}}^G$.*

The group G is usually clear from context, and we write X_S for X_S^G.

Definition 2. *Let $S \ni 0$ be a finite alphabet. A configuration $x \in S^G$ is* k-sparse *if it satisfies $|\{g \in G \mid x_g \neq 0\}| \leq k$. A subshift is* k-sparse *if each of its configurations is k-sparse, and* sparse *if it is k-sparse for some $k \in \mathbb{N}$.*

The one-S subshift X_S is of course a 1-sparse subshift on any group. Note that in a sparse subshift, there is a global bound on the number of nonzero

symbols. The *sum* $x+y$ of sparse configurations $x, y \in S^G$ with disjoint support (no $g \in G$ satisfies $x_g \neq 0$ and $y_g \neq 0$) is defined by

$$(x+y)_g = \begin{cases} x_g \text{ if } x_g \neq 0, \\ y_g \text{ otherwise.} \end{cases}$$

A finite pattern is represented computationally as a finite list of word-symbol pairs $(w, d) \in s(G)^* \times S$. Such a list is *inconsistent* if it contains two pairs (v, d) and (w, e) with $v \sim w$ and $d \neq e$ (in this case, it does not actually encode a pattern), and otherwise *consistent*.

Definition 3. *The* word problem of G *is the set* $E = \{w \in s(G)^* \mid w \sim 1_G\}$ *of words that represent the identity element of* G*. Whether the word problem is decidable is independent of the chosen generator set. We say* G *is* recursively presented *if* $G \cong \langle g_1, \ldots, g_k \mid w_1, w_2, \ldots \rangle$*, where* $(w_i)_{i \in \mathbb{N}}$ *is a computable sequence of relations.*[2] *This is equivalent to the set* E *being recursively enumerable.*

If G *has a decidable word problem, we say that a subshift* $X \subset S^G$ *is* Π_1^0 *if there exists a Turing machine that enumerates a list of consistent forbidden patterns defining it.*

A subshift X is Π_1^0 if and only if there exists an oracle Turing machine that, given an oracle for a configuration $x \in S^G$ (which returns the symbol $x_w \in S$ for a given word $w \in s(G)^*$), eventually halts if and only if $x \notin X$.

2.2 Automata

We now define group-walking automata and the subshifts they recognize. Here and henceforth, by π_i we mean the projection to the ith coordinate of a finite Cartesian product.

Definition 4. *A* k-headed group-walking automaton *on the full shift* S^G *is a tuple* $A = (\prod_{i=1}^k Q_i, f, I, F)$*, where* $Q_1, Q_2, \ldots, Q_k$ *are state sets not containing the symbol* 0*,* I *and* F *are finite clopen subsets of the product subshift* $Y = \prod_{i=1}^k X_{Q_i}$*, and* $f : S^G \times Y \to S^G \times Y$ *is a CA satisfying* $\pi_1 \circ f = \pi_1$*.*

We denote by $\mathcal{S}(G, k)$ *the class of subshifts* $X \subset S^G$ *for which there exists a* k*-headed automaton* A *as above such that*

$$X = \{x \in S^G \mid \forall g, h \in G, y \in I, n \in \mathbb{N} : h \cdot \pi_2(f^n(g \cdot x, y)) \notin F\}.$$

We also write $\mathcal{S}(G) = \bigcup_{k \geq 1} \mathcal{S}(G, k)$*.*

The intuition for these definitions is the following. A configuration $y \in Y = \prod_{i=1}^k X_{Q_i}$ consists of k layers $\pi_i(y)$, each of which contains at most one nonzero symbol $q_i \in Q_i$, representing the i'th head of the automaton in state q_i. The cellular automaton f is the update function of the heads: since f has a finite

[2] The term 'recursively presented' comes from the fact that one may always assume $\{w_i \mid i \in \mathbb{N}\}$ to be a recursive set of words.

radius, the heads can only move at a bounded speed, and interact over bounded distances. Also, the condition $\pi_1 \circ f = \pi_1$ ensures that the automaton cannot alter the configuration of S^G that it runs on. The clopen sets $I, F \subset Y$ are the *initial and final states* of the automaton. Each of them is a finite union of cylinder sets $[P]$, and since they are also finite as sets, each of the patterns P necessarily contains all k heads of the automaton. Thus, an initial or finite state specifies the position and internal state for each head, and we translate them by every element of G in the definition of $\mathcal{S}(G)$.

The definition is given in dynamical terms to make the connection with cellular automata clearer, and to facilitate the statement and proof of Lemma 3. With some work, one can show that this model is equivalent to the one we gave in [9] in terms of the classes of subshifts defined.

Example 2. Let G be again the free group generated by the elements $g, h \in G$, and let $S = \{0, 1\}$. We define a two-headed group-walking automaton $A = (Q_1 \times Q_2, f, I, F)$ on G as follows. The local state sets are $Q_1 = \{q_g, q_{g^{-1}}\}$ and $Q_2 = \{q_h, q_{h^{-1}}\}$, the set of initial states I contains only the cylinder set $\{x \in (Q_1 \times Q_2)^G \mid x_{1_G} = (q_g, q_h)\}$, and the set of final states F contains the cylinder $\{x \in (Q_1 \times Q_2)^G \mid x_{1_G} = (q_{g^{-1}}, q_{h^{-1}})\}$. This means that the heads of the automaton are initialized at the same coordinate in states q_g and q_h, and a configuration is rejected if they ever return to the same coordinate in states $q_{g^{-1}}$ and $q_{h^{-1}}$. The CA f moves each head by the step indicated in its state, and if a head encounters a symbol 1 in state q_g or q_h, it assumes the respective inverse state $q_{g^{-1}}$ or $q_{h^{-1}}$.

In a run of the automaton, the heads start moving in the directions g and h until they encounter symbols 1, and then turn back. If both of them turn at the same time, they will meet again where they started, in the states $q_{g^{-1}}$ and $q_{h^{-1}}$, so the configuration is rejected. If not, the configuration is not rejected. Thus the automaton A defines the subshift $X \subset S^G$ with the forbidden patterns

$$\{1_G \mapsto 0, g \mapsto 0, h \mapsto 0, \dots, g^{n-1} \mapsto 0, h^{n-1} \mapsto 0, g^n \mapsto 1, h^n \mapsto 1\}$$

for all $n \geq 1$. It is not an SFT.

Naturally, Turing machines are stronger than multi-headed finite automata.

Lemma 1. *If G has a decidable word problem and $X \in \mathcal{S}(G)$, then $X \in \Pi_1^0$.*

Proof. Let A be a group-walking automaton that defines X. We construct a Turing machine T_A that outputs its forbidden patterns. The machine T_A enumerates all consistent patterns over G (using the fact that G has a decidable word problem), and simulates a run of the automaton A on each of them, from every initial state. If one of the heads exits the pattern during such a simulation, or every head enters an infinite loop, that simulation is simply discarded. If one of the runs enters a rejecting state on the pattern P before exiting it (from any initial configuration and initial position on the domain $D(P)$), the machine T_A outputs the pattern P. It is clear that T_A defines the same subshift as A. □

3 Non-torsion Groups with a Decidable Word Problem

On non-torsion groups, there are essentially no restrictions on the types of computation a multi-headed finite state automaton can do, apart from the inherent limits of computation. In fact, we will implement all Π_1^0-subshifts on such groups, using just three heads. The construction is similar to that in [4] and [9].

Theorem 1. *If G is finitely generated, infinite and non-torsion, and has a decidable word problem, then $\mathcal{S}(G, 3)$ is exactly the class of Π_1^0-subshifts.*

Proof. By Lemma 1, all $\mathcal{S}(G, 3)$-subshifts are Π_1^0. To show that $\mathcal{S}(G, 3)$ contains all Π_1^0-subshifts, we repeat the proof of Theorem 5 in [9], where the same problem was considered for $G = \mathbb{Z}^d$, with one additional detail in the non-abelian case. Since there are not many changes, we refer to [9] for some of the details.

Let $X \subset S^G$ be a Π_1^0-subshift, and let $h \in G$ be an element of infinite order. Given a Turing machine T enumerating a list of forbidden patterns for X, we construct an automaton A_T with three heads, the *pointer head*, the *zig-zag head* and the *counter head*. The relative positions of these heads store a number, which we increment, decrement, multiply and divide by suitable constants, and test for equivalence and divisibility by constants, in order to perform arbitrary computation: such a model is Turing-complete by the results of [10].

More precisely, all heads are initialized on the same element of G, which we may assume to be 1_G. The run of the automaton proceeds in *sweeps*, each of which either corresponds to an arithmetical operation as described above, or moves the heads in some direction. Between these sweeps, the location of both the pointer head and the zig-zag head is some $g \in G$, and the position of the counter head is gh^p. The number $p \in \mathbb{N}$ is the *counter value*. Changes in the counter value are used to perform computation, and changes to the value g allow us to read the contents of every cell in the configuration.

The operations are implemented as in the case $G = Z^d$ (for example, see Proposition 3 in [9]). The only operations that are nontrivial to implement are multiplication and division, and they are dealt with by standard signaling techniques. The details of this are omitted in [9], so we outline the construction here: we explain how to multiply the counter value by a rational number $0 < \frac{m}{n} < 1$ assuming the counter value is divisible by n; to multiply by a rational number greater than 1, one essentially performs the same steps in reverse.

For this, let $g \in G$ be the position of the pointer head. The idea is that the zig-zag head moves to the counter head, which is at gh^p, along the progression $g, gh, gh^2, \ldots$. The two heads then perform a coordinated move along the path $g, gh, gh^2, \ldots, gh^c$, so that they meet exactly at $gh^{\frac{m}{n}p}$. The zig-zag head then returns to the pointer head, and computation continues. We have much freedom in performing these moves, but we fix a particular scheme that works: After the zig-zag head and the counter head meet, the counter head starts moving in steps of h towards the pointer head (so that from the cell gh^j, it moves to the cell gh^{j-1} in one step), until it meets the zig-zag head again. The zig-zag head moves towards the pointer head by h^n every step, until it meets the pointer

head. Note that n divides p, so that the zig-zag head indeed reaches exactly the cell g. After this, the zig-zag head starts moving back towards the counter head at speed $\frac{m}{n-m-1}$. More precisely, the zig-zag head carries a modular counter, starting at 0, and at each step it increments this counter. When the modular counter reaches $n-m-1$, the zig-zag head resets it to 0 and moves by h^m. When the zig-zag head reaches the counter head, it turns back, and returns to the pointer head. It is a simple calculation to check that the heads meet exactly at $gh^{\frac{m}{n}p}$, as required, so the counter value has been changed correctly.

Now that we can do arbitrary computation in the counter value, we give the algorithm we simulate in it. The algorithm is the same as in the proof of Theorem 5 of in [9], and we reproduce it in Algorithm 1 with trivial modifications. In the algorithm, objects related to the group are stored as they are output by the Turing machine: group elements are finite words over $s(G)$, and patterns $P \in S^D$ are lists of pairs $(w, s) \in s(G)^* \times S$ meaning $P_w = s$. We assume the Turing machine T outputs an infinite list of forbidden patterns, and enters the state q_{out} every time it outputs a new pattern.

Algorithm 1. The algorithm that the three-headed automaton A_T simulates.

$c \leftarrow c_0$ ▷ A configuration of T, set to the initial configuration
$u \leftarrow 1_G \in G$ ▷ The position of the pointer head relative to the initial position
$P : \emptyset \to S$ ▷ A finite pattern at the initial position
loop
  repeat
    $c \leftarrow \text{NextConf}_T(c)$ ▷ Simulate one step of T
  until $\text{State}(c) = q_{\text{out}}$ ▷ T outputs something
  $P' \leftarrow \text{OutputOf}(c)$ ▷ A forbidden pattern
  while $D(P') \not\subset D(P)$ **do**
    $w \leftarrow \text{LexMin}(D(P) \setminus D(P'))$ ▷ The lexicographically minimal element
    for $a = u_1^{-1}, u_2^{-1}, \ldots, u_{|u|}^{-1}, w_1, w_2, \ldots, w_{|w|}$ **do**
      $\text{MoveBy}(a)$ ▷ Move all heads of A_T by group element a
    $u \leftarrow w$ ▷ New position of the pointer is w
    $b \leftarrow \text{ReadSymbol}$ ▷ Read the symbol of x under the pointer head
    $P \leftarrow P \cup \{u \mapsto b\}$ ▷ Expand P by one coordinate
  if $P|_{D(P')} = P'$ **then halt** ▷ The forbidden pattern P' was found

The function ReadSymbol gives the symbol currently under the pointer head. The procedure MoveBy(a) causes the three heads to assume new positions: if the pointer head and zig-zag head are at g and the counter head is at gh^p, they are moved to ga and gah^p, respectively. This step is the main difference between the abelian and non-abelian cases, and we explain it below. We note that there are only finitely many different messages sent between the abstract computation and A_T, namely the exchange related to ReadSymbol and the commands MoveBy(a) for finitely many $a \in G$. This information exchange can easily be performed by storing the state of the Turing machine T directly in the finite state of the pointer head.

It is easy to see that this algorithm does what we want: whenever the Turing machine T enumerates a forbidden pattern P', we expand the stored pattern P by reading the configuration until its domain contains that of P'. If P' occurs in the configuration, it is eventually found by the algorithm from some starting position, and conversely, if the automaton halts, this is because it found a forbidden pattern.

To finish the proof, we explain how to perform MoveBy(a). If G is abelian, this can be done as in [9]: the zig-zag head moves to the counter head, informs it of the element of G by which it should move, and returns back. The counter head moves as instructed, and the pointer head does so as well. If the pointer head was previously at g and the counter head at gh^P, and both move by $a \in G$, then after this sequence of moves, the pointer head will be at ga, and the counter head at $gh^P a = gah^P$, as required. More generally, this works if h is in the center of G. Otherwise, we may have $gh^P a \neq gah^P$. Since we do not necessarily have $gah^P \in g\langle h \rangle a$, the counter head may not even encode a valid counter value.

However, using the same trick we used to perform multiplications, we can perform the movement in general. First, the zig-zag head moves to the counter head. Then, both heads start moving toward the pointer head. The counter head moves in steps of h^{-1}, computing the parity of p on the way, and the zig-zag head moves in steps of h^{-2}. If p is even, then the zig-zag head reaches the anchor head exactly, moves to ga, and starts moving along the sequence $ga, gah, gah^2, \ldots$ in steps of h. If p is odd, then the zig-zag head reaches the cell gh^{-1} instead, moves to gah, and starts moving in steps of h as before. The counter head performs the same task, but with the speeds reversed: after reaching the anchor head with speed h^{-1}, it starts moving from ga in steps of h^2 if p was even, and from gah in steps of h^2 if p was odd. When the counter head reaches the pointer head, the pointer head also moves to ga. It is easy to check that the counter head and the zig-zag head meet at the cell gah^p. The counter head stops, and the zig-zag head returns to the pointer head. □

4 Walking on Torsion Groups

A torsion group is one where every element generates a finite subgroup. In this section, we show that on such groups, non-trivial sparse subshifts cannot be recognized by multi-headed automata. We also show two results about cellular automata and automorphism groups of sparse subshifts on torsion groups. These follow from a curious property, Lemma 3, of CA on sparse subshifts on torsion groups. In its proof, we use the following lemma about finite metric spaces.

Lemma 2. *Let X be a finite metric space with $|X| = k \geq 2$. For all $c < \mathrm{diam}(X)/(k-1)$, there exists a nontrivial partition $X = Y \cup Z$ with $d(Y, Z) > c$.*

Proof. For a set $E \subset X$, write $B_E(r)$ for the closed ball of radius $r \geq 0$ around E. Let $\mathrm{diam}(X) = d(y, z)$ for some $y, z \in X$. Let $X_1 = \{y\}$, and inductively define $X_{i+1} = B_{X_i}(c)$. For all $i \geq 1$ we have either $|X_{i+1}| > |X_i|$ or $X_{i+1} = X_i$,

and in the latter case we have $X_j = X_i$ for all $j \geq i$. It follows that $X_i = X_{i+1}$ holds for some $i \leq k$.

If we have $X_i = X_{i+1} = X$, then $\operatorname{diam}(X) \leq (k-1)c$, since every element of X, including z, is in the ball $B_y((i-1)c) \subset B_y((k-1)c)$. This is a contradiction, so it must be the case that $X_i = X_{i+1} \neq X$. Then $Y = X_i$ and $Z = X \setminus X_i$ give the desired partition. □

Lemma 3. *For all torsion groups G, there exists a function $d : \mathbb{N}^3 \to \mathbb{N}$ with the following property. For all k-sparse subshifts $X \subset S^G$ over all alphabets $S \ni 0$ with $|S| = q + 1$, all cellular automata $f : X \to X$ with radius $r \in \mathbb{N}$, and all $x \in X$, we have*

$$(\exists n \in \mathbb{N} : f^n(x)_g \neq 0) \implies \exists h \in B_G(d(k, q, r)) : x_{gh} \neq 0.$$

Proof. We prove the existence of such a function d by induction. We define the function so that it is monotone in all the three parameters. Let t_G be the order function and T_G the torsion function of G.

Case 1: $k = 1$

First, let $k = 1$, and let $f : X \to X$ be a CA. It is easy to show that if $x_{gh} = 0$ for all $h \in B_G(r)$, then $f(x)_g = 0$. Intuitively, this means that nonzero symbols can 'spread' by at most r per time step, and one cannot appear from nowhere. Since X is a k-sparse subshift and $k = 1$, every point $x \in X$ contains at most one nonzero coordinate $x_g \neq 0$. Intuitively, we want to give an upper bound on how far the nonzero symbol can travel from its initial position g.

By shift-commutation, it is enough to analyze the case $x_{1_G} \neq 0$. Combining the previous observations and the fact $|S| = q+1$, it follows from the pigeonhole principle that $f^{n+m}(x) = \sigma_h^R(f^n(x))$ for some $0 \leq n < n + m \leq q + 1$ and $h \in B_G((q+1)r)$. Since f commutes with the shift, we have $f^{n+\ell m}(x) = \sigma_{h^k}^R(f^n(x))$ for all $\ell \in \mathbb{N}$. Since $h^{t_G(h)} = 1_G$, we have $f^{n+t_G(h)m}(x) = f^n(x)$. We have shown that $f^j(x)_{h'} \neq 0$ for some $j \in \mathbb{N}$ implies $h' \in B_G((q + 1)r(1 + t_G(h)))$. Since $h \in B_G((q + 1)r)$, we can define

$$d(1, q, r) = (q + 1)r(1 + T_G((q + 1)r)).$$

Next, consider the case $k > 1$. To each configuration $x \in X$, we associate the metric space $A(x)$ whose points are the nonzero coordinates of x, and whose distances are those induced by the natural (right) distance in G. We will split the analysis of the dynamics of f on the point x into two cases, depending on whether the diameter of $A(f^n(x))$ stays bounded (by an explicit constant) as n grows.

Intuitively, the idea is that as long as the diameter stays small, we can shrink all the information in x into a single symbol, reducing to the case $d(1, \cdot, \cdot)$, and if the configuration starts expanding, then it splits into two pieces that can never again communicate, and we apply induction to these smaller pieces.

More precisely, define $A(x) = (\{g \in G \mid x_g \neq 0\}, \delta)$ where

$$\delta(g, h) = \min\{\ell \in \mathbb{N} \mid \exists w \in s(G)^\ell : h = gw\}.$$

Define also $c = 2d(k-1, q, r) + r$, and note that since d is monotone, we in particular have

$$c \geq \max_{1 \leq \ell < k} d(\ell, q, r) + d(k - \ell, q, r) + r.$$

We say that a configuration $x \in X$ is *clustered* if $\operatorname{diam}(A(x)) \leq (k-1)c$ holds, and *scattered* otherwise.

Case 2: clustered configurations

First, suppose $x \in X$ and $N \in \mathbb{N}$ are such that $f^n(x)$ is clustered for all $n \leq N$. We will give an upper bound on how far nonzero symbols can travel from their original positions in these N steps. Let $Z \subset X$ be the subshift generated by the configurations $f^n(x)$ for $n \leq N$. It is easy to see that every configuration of Z is clustered. Note that that the subshift Z may not be closed under f.

Let $Y = X_{\{0\} \cup K}$, where $K \subset \mathcal{L}_{B_G((k-1)c)}(Z)$ is the set of patterns P of shape $B_G((k-1)c)$ occurring in Z such that $P_{1_G} \neq 0$. Clearly, Y is a 1-sparse subshift, and it should be thought of as a 'compressed' version of Z, where all the nonzero symbols have been encoded into a single coordinate. The idea is to simulate CA f on the compressed subshift Y, and reduce back to the $k = 1$ case. Let $\phi : Y \to S^G$ be the 'decompression function' defined by

$$\phi(y)_h = \begin{cases} (y_g)_{g^{-1}h}, & \text{if } \exists g \in G : y_g \neq 0 \wedge g^{-1}h \in B_G((k-1)c), \\ 0, & \text{otherwise.} \end{cases}$$

Let $Y' = \phi^{-1}(Z)$, so that $\phi : Y' \to Z$ is a surjective block map.[3] A visualization of ϕ is shown in Figure 1.

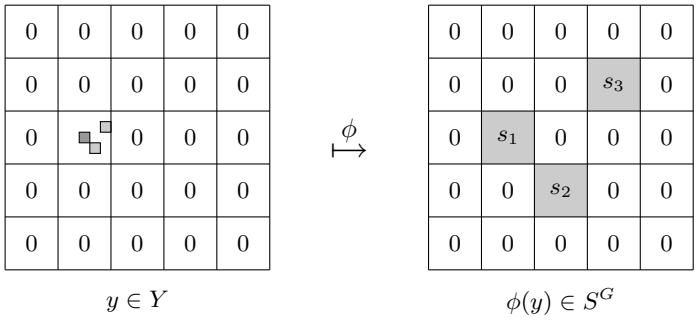


Fig. 1. The decompression function ϕ applied to a configuration $y \in Y$. We have chosen $G = \mathbb{Z}^2$ here for simplicity, even though it is not a torsion group. Note that the alphabet of Y consists of certain patterns of X and the symbol 0.

[3] The fact that G is torsion prevents us, in general, from defining a bijective version of ϕ. Also, the subshift Y' may be strictly smaller than Y.

Claim. There exists a (not necessarily unique) cellular automaton $f_\phi : Y' \to Y'$ such that $\phi(f_\phi(y)) = f(\phi(y))$ holds for all $y \in Y'$ such that $f(\phi(y)) \in Z$.

Intuitively, the CA f_ϕ simulates f on the compressed configurations of Y', as long as their ϕ-images are clustered.

Proof (of claim). Observe that for each $z \in Z$ and $g \in G$ there is at most one configuration $y \in Y'$ such that $y_g \neq 0$ and $\phi(y) = z$. Let then $1 = h_1 < h_2 < h_3 < \cdots$ be any total order on the group G, not necessarily in any way compatible with its algebraic structure. Then we can define a map f_ϕ with the desired properties as follows. First, for the all-0 configuration $0^G \in Y'$, we define $f_\phi(0^G) = 0^G$, and for all $y \in Y'$ such that $f(\phi(y)) \notin Z$, we also define $f'(y) = 0^G$. For all other $y \in Y'$, let $g \in G$ be the unique element with $y_g \neq 0$, and let $W \subset Y'$ be the set of configurations $y' \in Y'$ with $\phi(y') = f(\phi(y))$. The set W is nonempty since $\phi : Y' \to Z$ is surjective, and it is finite because the unique nonzero coordinate of each $y' \in W$ is among the coordinates gh where $h \in B_G((k-1)c + r)$, since we assumed $P_{1_G} \neq 0$ for each $P \in K$. Now, we choose $f_\phi(y)$ to be the unique configuration $y' \in W$ with $y'_{gh} \neq 0$, where $h \in G$ is minimal in the ordering $h_1 < h_2 < \cdots$. It is easy to check that f_ϕ is then continuous and shift-commuting. In fact, from the way we defined it, we see that its radius is at most $(k-1)c + r$. □

Recall the clustered configuration $x \in S^G$. We have $x \in Z$ by the definition of Z, so there exists a configuration $y \in Y'$ such that $\phi(y) = x$. By the above claim, we have $\phi(f_\phi^n(y)) = f^n(x)$ for all $n \leq N$. Since Y is a 1-sparse subshift with alphabet of size $|K|+1$ and f_ϕ is a CA on it with radius at most $(k-1)c+r$, we have

$$(\exists n : f_\phi^n(y)_g \neq 0) \implies \exists h \in B_G(d(1, |K|, (k-1)c + r)) : y_{gh} \neq 0 \qquad (1)$$

by Case 1 of this proof. We also remark that if we have $N > |K|$, then the configuration $f^n(x)$ is clustered for all $n \in \mathbb{N}$, since there exist $i < j \leq N$ such that $f_\phi^i(\phi(y))$ is a translated version of $f_\phi^j(\phi(y))$.

It remains to prove a variant of the above formula for f, and for that, let $f^n(x)_g \neq 0$ for some $g \in G$. Since the block map ϕ has radius $(k-1)c$, we have $\phi(f^n(x))_{gh'} = f_\phi^n(y)_{gh'} \neq 0$ for some $h' \in B_G((k-1)c)$. Equation (1) implies that $y_{gh'h} \neq 0$ for some $h \in B_G(d(1, |K|, (k-1)c + r))$, and from the definition of ϕ it follows that $x_{gh'h} \neq 0$ as well, since $(y_{gh'h})_{1_G} \neq 0$. We have shown that if $f^n(x)$ contains a nonzero symbol in some coordinate, then there is a nonzero coordinate in x at distance at most $d(1, |K|, (k-1)c + r) + (k-1)c$. Note that the cardinality of K is at most exponential in $(k-1)c$.

Case 3: scattered configurations

Suppose finally that the configuration $f^n(x)$ is scattered for some $n \in \mathbb{N}$, which we assume to be minimal. By the remark at the end of Case 2, we have $n \leq |K|$. We apply Lemma 2 to the metric space $A(f^n(x))$, and obtain a partition for it into sets $C, D \subset G$ with distance at least c.

Denote $y = f^n(x)$. We define a partition of the configuration y by $y = y_C + y_D$, where $(y_C)_g = y_g$ when $g \in C$ and $(y_C)_g = 0$ otherwise, and y_D is defined analogously. By the definition of c, we have $c \geq d(|C|, q, r) + d(|D|, q, r) + r$. It is then easy to see that $f^n(y) = f^n(y_C) + f^n(y_D)$ for all $n \in \mathbb{N}$. In particular, if we have $f^j(y)_g \neq 0$ for some $j \in \mathbb{N}$ and $g \in G$, then $y_{gh} \neq 0$ for some $h \in B_G(\max_{\ell<k} d(\ell, q, r)) \subset B_G(d(k-1, q, r))$ by the induction hypothesis. Since we have $n \leq |K|$ and the CA f has radius r, this implies that $x_{ghh'} \neq 0$ for some $h' \in B_G(r|K|)$, which implies $hh' \in B_G(r|K| + d(k-1, q, r))$.

Putting all three cases together, we can define the function d recursively by

$$d(k, q, r) = d(1, |K|, (k-1)c + r) + (k-1)c + r|K| + d(k-1, q, r)$$

for all $k > 1$. □

The bounds we give are not very strong, but at least one can check that if the torsion function T_G is primitive recursive, then so is the function d.

Theorem 2. *If G is finitely generated, infinite and torsion, and $X \subset S^G$ is sparse and nontrivial, then $X \notin \mathcal{S}(G)$.*

Proof. Let A be a group-walking automaton and Y its associated subshift, and let $X' = \{x + y \mid x, y \in X, \forall g \in G : x_g = 0 \vee y_g = 0\}$. Since $X' \times Y$ is sparse, Lemma 3 implies that any head of A can only travel a bounded distance on any configuration of $X' \times Y$. Then, for all $x \in X$ and all but finitely many $g \in G$, the configuration $x + (g \cdot x)$ is rejected by A if and only if x is. If the support of x is maximal, this configuration is not in X. Thus A does not define X. □

Lemma 3 also restricts the structure of the automorphism group of a sparse subshift on a torsion group.

Theorem 3. *If G is torsion and $X \subset S^G$ is sparse, then* $\mathrm{Aut}(X)$ *is also torsion.*

The last theorem has an obvious converse: if G is not torsion, then the shift along a copy of $\mathbb{Z}$ is a non-torsion element of $\mathrm{Aut}(X)$ whenever X is sparse and nontrivial. One can construct such examples even in the quotient group $\mathrm{Aut}(G)/\langle \sigma_g^R \mid g \in G \rangle$.

5 Undecidable Word Problem

If the word problem for G is not necessarily decidable, one can give multiple definitions of Π_1^0. We give two, both of which correspond to our previous definition of Π_1^0 when the word problem is decidable. Recall that finite patterns are represented computationally as lists of pairs drawn from $s(G)^* \times S$.

Definition 5. *A subshift on G is Π_1^0 if there exists a Turing machine enumerating a set of (possibly inconsistent) forbidden lists of word-symbol pairs for it.*

Definition 6. *A subshift X on G is* intrinsically Π_1^0 *if there exists an oracle Turing machine that, given an oracle for the word problem of G, enumerates a set of consistent forbidden lists of word-symbol pairs for X.*

In [2], what we call intrinsically Π_1^0 is called G-effective, and this notion was first defined and studied there. Its actual definition in [2] uses 'group-walking Turing machines', but it is also shown to be equivalent to Definition 6. The following results, the first of which is a direct corollary of Theorem 1, relate these classes of subshifts to the hierarchy of group-walking automata.

Theorem 4. *If G is finitely generated, infinite and non-torsion, then $\mathcal{S}(G, 3)$ contains the class of Π_1^0-subshifts.*

Theorem 5. *If G is finitely generated, infinite and non-torsion, then $\mathcal{S}(G, 4)$ is exactly the class of subshifts on G which are intrinsically Π_1^0.*

Proof. Clearly, all $\mathcal{S}(G, 4)$ subshifts are intrinsically Π_1^0, since a Turing machine with an oracle for the word problem of G can simulate a multi-headed finite state machine on the group. The proof that $\mathcal{S}(G, 4)$ contains the intrinsically Π_1^0 subshifts is similar to that of Theorem 1, except that we must simulate a Turing machine with access to an oracle for G. Thus, we only need to describe how one can use four heads to check whether the identity $1 \sim w$ holds for an arbitrary $w \in s(G)^*$. For this, we use three heads to move by the letters of w, and leave the fourth head as a marker in the cell we started from. We return back on top of the fourth head if and only if $1 \sim w$. We can then move back by w^{-1} and pick up the fourth head. □

From these results, we obtain a characterization of torsion groups.

Lemma 4. *The X_S subshift is intrinsically Π_1^0 on every group.*

Theorem 6. *Let G be a finitely generated infinite group. Then G is torsion if and only if $\mathcal{S}(G, 4)$ is not equal to the class of all intrinsically Π_1^0 subshifts.*

Proof. This follows from Lemma 4, Theorem 5 and Theorem 2. □

Finally, we note that Lemma 4 requires the intrinsic notion of computability, as shown by the following corollary of [2, Proposition 2.3] (also proved in [7]).

Proposition 1. *Let G be a recursively presented and finitely generated group, and S is a nontrivial finite alphabet. The subshift X_S^G is Π_1^0 if and only if G has a decidable word problem.*

6 Future Work and Open Questions

While we need four heads in the *proof* of Theorem 5, we are not able to separate the class $\mathcal{S}(G, 3)$ from $\mathcal{S}(G, 4)$ on any group G. We do have a general construction which separates these classes on all sufficiently complex torsion groups. Unfortunately, we do not know how to construct a group with the necessary properties, as the construction of torsion groups is quite complicated. Nevertheless, this leads us to believe that the classes are not always equal.

Conjecture 1. There exists an infinite finitely generated torsion group G such that $\mathcal{S}(G,3) \subsetneq \mathcal{S}(G,4)$. In particular, $\mathcal{S}(G,3)$ is not always equal to the class of intrinsically Π_1^0 subshifts.

We know that if G is not a torsion group, then the hierarchy $\mathcal{S}(G,k)_{k\geq 1}$ collapses to the fourth level (if not earlier), and $\mathcal{S}(G,4)$ is exactly the class of intrinsically Π_1^0 subshifts. On torsion groups, the hierarchy never reaches all intrinsically Π_1^0 subshifts, but we have not shown that it is infinite. We believe we have a general construction that proves exactly this, but it is relatively complicated, so for now we only state its conclusion as a conjecture.

Conjecture 2. If G is an infinite finitely generated torsion group, then the hierarchy $\mathcal{S}(G,k)_{k\geq 1}$ is infinite.

Some very basic questions about the abelian cases were left open in [9]. We have no progress on these questions.

Question 1. Do we have $\mathcal{S}(\mathbb{Z},2) = \mathcal{S}(\mathbb{Z},3)$ or $\mathcal{S}(\mathbb{Z}^2,2) = \mathcal{S}(\mathbb{Z}^2,3)$?

We note that in [4], a slightly different model of multi-headed group-walking automaton is studied on the group $\mathbb{Z}^2$, and it is shown that in this model, two-headed machines are strictly weaker than three-headed ones. It seems that the question is harder in our model. In [9], we only showed that $\mathcal{S}(\mathbb{Z}^d,2) \subsetneq \mathcal{S}(\mathbb{Z}^d,3)$ holds for $d \geq 3$.

References

1. Adian, S.I.: The burnside problem on periodic groups and related questions. Proceedings of the Steklov Institute of Mathematics **272**(2), 2–12 (2011)
2. Aubrun, N., Barbieri, S., Sablik, M.: A notion of effectiveness for subshifts on finitely generated groups. ArXiv e-prints, December 2014
3. Ceccherini-Silberstein, T., Coornaert, M.: Cellular Automata and Groups. Springer Monographs in Mathematics. Springer (2010)
4. Delorme, M., Mazoyer, J.: Pebble automata. figures families recognition and universality. Fundam. Inf. **52**(1–3), 81–132 (2002)
5. Grigorčuk, R.I.: On Burnside's problem on periodic groups. Funktsional. Anal. i Prilozhen. **14**(1), 53–54 (1980)
6. Gupta, N., Sidki, S.: On the burnside problem for periodic groups. Mathematische Zeitschrift **182**(3), 385–388 (1983)
7. Jeandel, E.: Some Notes about Subshifts on Groups. ArXiv e-prints, January 2015
8. Lind, D., Marcus, B.: An introduction to symbolic dynamics and coding. Cambridge University Press, Cambridge (1995)
9. Salo, V., Törmä, I.: Plane-walking automata. CoRR, abs/1408.6701 (2014)
10. Schroeppel, R.: A two counter machine cannot calculate 2^N (1972)

Restricted Density Classification in One Dimension

Siamak Taati(✉)

Mathematics Institute, Leiden University, P.O. Box 9512,
2300 RA Leiden, The Netherlands
siamak.taati@gmail.com

Abstract. The density classification task is to determine which of the symbols appearing in an array has the majority. A cellular automaton solving this task is required to converge to a uniform configuration with the majority symbol at each site. It is not known whether a one-dimensional cellular automaton with binary alphabet can classify all Bernoulli random configurations almost surely according to their densities. We show that any cellular automaton that washes out finite islands in linear time classifies all Bernoulli random configurations with parameters close to 0 or 1 almost surely correctly. The proof is a direct application of a "percolation" argument which goes back to Gács (1986).

Keywords: Cellular automata · Density classification · Phase transition · Spareness · Percolation

1 Introduction

An array containing symbols `0` and `1` is given. We would like to determine which of the two symbols `0` and `1` appears more often in this array. The challenge is to perform this task in a local, uniform and decentralized fashion, that is, by means of a cellular automaton. A cellular automaton solving this problem is to receive the input array as its initial configuration and to end by reaching a consensus, that is, by turning every symbol in the array into the majority symbol. All computations must be done on the same array with no additional symbols.

If we require the cellular automaton to solve the task for all odd-sized finite arrays with periodic boundary conditions (i.e., arrays indexed by a ring $\mathbb{Z}_n$ or a d-dimensional torus $\mathbb{Z}_n^d$, where n is odd), then no perfect solution exists [10] (see also [1]). Indeed, the effect of an isolated `1` deep inside a large region of `0`'s will soon disappear, hence its removal from the starting configuration should not affect the end result. However, removing such an isolated `1` could shift the balance of the majority from `1` to `0` in a borderline case.

Here, we consider a variant of the problem on infinite arrays, and focus on the one-dimensional case. We ask for a cellular automaton that classifies a randomly chosen configuration (say, using independent biased coin flips) according to its density *almost surely* (i.e., with probability 1). We relax the notion of classification to allow computations that take infinitely long: we only require that the

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 238–250, 2015.
DOI: 10.1007/978-3-662-47221-7_18

content of each site is eventually turned into the majority symbol and remains so forever, but we allow the fixation time to depend on the site.

Almost sure classification of random initial configurations is closely related to the question of stability of cellular automata trajectories against noise and the notion of ergodicity for probabilistic cellular automata. Constructing a cellular automaton with at least two distinct trajectories that remain distinguishable in presence of positive Bernoulli noise is far from trivial. Toom [12,13] produced a family of examples in two dimensions. Each of Toom's cellular automata has two or more distinct fixed points that are stable against noise: in presence of sufficiently small (but positive) Bernoulli noise, the cellular automaton starting from each of these fixed points remains close to that fixed point for an indefinite amount of time. The noisy version of each of these cellular automata is thus non-ergodic in that it has more than one invariant measure.

The most well-known of Toom's examples is the so-called *NEC* rule (NEC standing for North, East, Center). The NEC rule replaces the symbol at each site with the majority symbol among the site itself and its north and east neighbors. Combining the combinatorial properties of the NEC rule and well-known results from percolation theory, Bušić, Fatès, Mairesse and Marcovici [1] showed that the NEC cellular automaton also solves the classification problem: starting from a random Bernoulli configuration with parameter p on $\mathbb{Z}^2$ (i.e., using independent coin flips with probability p of having $\mathtt{1}$ at each site), the cellular automaton converges almost surely to the uniform configuration $\underline{\mathtt{0}}$ if $p < 1/2$ and to $\underline{\mathtt{1}}$ if $p > 1/2$.

The situation in dimension one is more complicated. No one-dimensional cellular automaton with binary alphabet is known to classify Bernoulli random configurations. Moreover, Toom's examples do not extend to one dimension; the only example of a one-dimensional cellular automaton with distinct stable trajectory in presence of noise is a sophisticated construction due to Gács [5,6] based on error-correction and self-simulation, which uses a huge number of symbols per site.

There are however candidate cellular automata in one dimension that are suspected to both classify Bernoulli configurations and to remain bi-stable in presence of noise. The oldest, most studied candidate is the *GKL* cellular automaton, introduced by Gács, Kurdyumov and Levin [4]. Another candidate with similar properties and same degree of simplicity is the *modified traffic* cellular automaton studied by Kůrka [9] and Kari and Le Gloannec [8]. Both of these two automata have the important property that they "wash out finite islands of errors" on either of the two uniform configurations $\underline{\mathtt{0}}$ and $\underline{\mathtt{1}}$ [7,8]. In other words, each of the two uniform configurations $\underline{\mathtt{0}}$ and $\underline{\mathtt{1}}$ is a fixed point that attracts all configurations that differ from it at no more than finitely many sites. Incidentally, this same property is also shared among Toom's cellular automata, and is crucial (but not sufficient) for its noise stability and density classification properties.

A cellular automaton that washes out finite islands of errors, also washes out infinite sets of errors that are sufficiently sparse. In this context, a set should be considered sparse if it can be covered with disjoint finite islands that are washed

out before sensing the effect of (or having an effect on) one another. It turns out that a Bernoulli random configuration with sufficiently small parameter is sparse with probability 1. The proof is via a beautiful and relatively simple argument that goes back to Gács [5,6], who used it to take care of the probabilistic part of his result. The author has learned this argument in a more streamlined form from a recent paper of Durand, Romashchenko and Shen [2], who used it in the context of aperiodic tilings. Given its simplicity and potential, we shall repeat this argument below.

An immediate consequence of the sparseness of low-density Bernoulli sets is that any cellular automaton that washes out finite islands of errors on $\underline{0}$ and $\underline{1}$ (e.g., GKL and modified traffic) almost surely classifies a Bernoulli random configuration correctly, as long as the Bernoulli parameter p is close to either 0 or 1. It remains open whether the same classification occurs for all values of p in $(0, 1/2) \cup (1/2, 1)$.

1.1 Terminology

Let us proceed by fixing the terminology and formulating the problem more precisely. By a *configuration*, we shall mean an infinite array of symbols x_i chosen from an alphabet S that are indexed by integers $i \in \mathbb{Z}$, or equivalently, a function $x : \mathbb{Z} \to S$. The evolution of a cellular automaton is obtained by iterating a transformation $\Phi : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ on a starting configuration $x : \mathbb{Z} \to S$. The transformation $x \mapsto \Phi x$ is carried out by applying a *local update rule* f simultaneously on every site so that the new symbol at site i reads $(\Phi x)_i \triangleq f(x_{i-r}, x_{i-r+1}, \ldots, x_{i+r})$. We call the sites $i-r, i-r+1, \ldots, i+r$ the *neighbors* of site i and refer to r as the neighborhood *radius* of the cellular automaton.

The *density* of a symbol a in a configuration x is not always well-defined or non-ambiguous. We take as the definition,

$$\rho_a(x) \triangleq \lim_{N \to \infty} \frac{|\{i \in [-N, N] : x_i = a\}|}{2N + 1} \tag{1}$$

when the limit exists. According to the law of large numbers, the density of a symbol a in a Bernoulli random configuration is almost surely the same as the probability of occurrence of a at each site. Formally, if X is a random configuration $\mathbb{Z} \to S$ in which the symbol at each site is chosen independently of the others, taking value a with probability $p(a)$, then $\mathbb{P}\{\rho_a(X) = p(a)\} = 1$.

When $S = \{\mathtt{0}, \mathtt{1}\}$, we simply write $\rho(x) \triangleq \rho_{\mathtt{1}}(x)$ for the density of $\mathtt{1}$'s in x. We say that a cellular automaton $\Phi : \{\mathtt{0}, \mathtt{1}\}^{\mathbb{Z}} \to \{\mathtt{0}, \mathtt{1}\}^{\mathbb{Z}}$ *classifies* a configuration $x : \mathbb{Z} \to \{\mathtt{0}, \mathtt{1}\}$ according to density if $\Phi^t x \to \underline{\mathtt{0}}$ or $\Phi^t x \to \underline{\mathtt{1}}$ as $t \to \infty$, depending on whether $\rho(x) < 1/2$ or $\rho(x) > 1/2$. The notation $\underline{a}$ is used to denote a uniform configuration with symbol a at each site. For us, the meaning of the convergence of a sequence of configurations $x^{(1)}, x^{(2)}, \ldots$ to another configuration x is *site-wise eventual agreement*: for each site i, there must be an index n_i after which all the following configurations in the sequence agree with x on the content of site i. (Formally, $x_i^{(n)} = x_i$ for all $n \geq n_i$.) This is the concept of convergence in the product topology of $S^{\mathbb{Z}}$, which is a compact and metric topology.

2 Eroder Property

Let us describe two candidates that are suspected to solve the density classification problem in one dimension: the cellular automaton of Gács, Kurdyumov and Levin and the modified traffic rule. Both cellular automata are defined on binary configurations $\mathbb{Z} \to \{0, 1\}$ and have neighborhood radius 3.

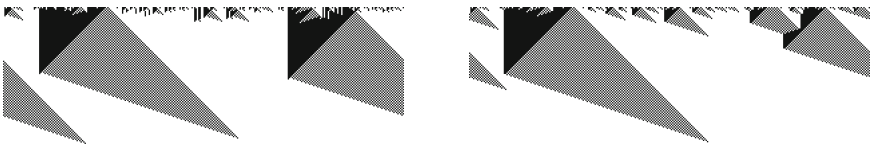
The cellular automaton of Gács, Kurdyumov and Levin [4] (*GKL* for short) is defined by the transformation

$$(\Phi x)_i \triangleq \begin{cases} \mathrm{maj}(x_{i-3}, x_{i-1}, x_i) & \text{if } x_i = 0, \\ \mathrm{maj}(x_i, x_{i+1}, x_{i+3}) & \text{if } x_i = 1, \end{cases} \tag{2}$$

where $\mathrm{maj}(a, b, c)$ denotes the majority symbol among a, b, c.

The *modified traffic* cellular automaton [8,9] is defined as a composition of two simpler automata: the traffic automaton followed by a smoothing filter. The *traffic* automaton transforms a configuration by replacing every occurrence of 10 with 01. The follow-up filter replaces the 1 in every occurrence of 0010 with 0, and symmetrically, turns the 0 in every occurrence of 1011 into a 1.

Sample space-time diagrams of the GKL and the modified traffic automata are depicted in Figure 1. Note that both GKL and modified traffic have the following symmetry: exchanging 0 with 1 and right with left leaves the cellular automaton unchanged.



(a) GKL (b) modified traffic

Fig. 1. Finding the majority in a biased coin-flip configuration. Time goes downwards.

The uniform configurations $\underline{0}$ and $\underline{1}$ are fixed points of both GKL and modified traffic automata. The following theorem states that both automata wash out finite islands of errors on either of the two uniform configurations $\underline{0}$ and $\underline{1}$. This is sometimes called the *eroder property*. For the GKL automaton, the eroder property was proved by Gonzaga de Sá and Maes [7]; for modified traffic, the result is due to Kari and Le Gloannec [8]. Let us write $\mathrm{diff}(x, y) \triangleq \{i \in \mathbb{Z} : x_i \neq y_i\}$ for the set of sites at which two configurations x and y differ. We call x a *finite perturbation* of z if $\mathrm{diff}(z, x)$ is a finite set.

Theorem 1 (Eroder property [7,8]). *Let Φ be either the GKL or the modified traffic cellular automaton. For every finite perturbation x of $\underline{0}$, there is a time t such that $\Phi^t x = \underline{0}$. If $\mathrm{diff}(\underline{0}, x)$ has diameter at most n (i.e., covered by an interval of length n), then $\Phi^{2n} x = \underline{0}$. The analogous statement about finite perturbations of $\underline{1}$ holds by symmetry.*

Let us emphasize that many simple cellular automata have the eroder property on some uniform configuration. For instance, the cellular automaton $\Phi : \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ defined by $(\Phi x)_i \triangleq x_{i-1} \wedge x_i \wedge x_{i+1}$ washes out finite islands on the uniform configuration $\underline{0}$. What is remarkable about GKL and modified traffic is the fact that they have the eroder property on *two distinct* configurations $\underline{0}$ and $\underline{1}$. This double eroder property may lead one to guess that these two cellular automata could indeed classify Bernoulli configurations according to density or that the trajectories of the fixed points $\underline{0}$ and $\underline{1}$ are stable in presence of small but positive noise.

3 Washing Out Sparse Sets

In this section, we consider a slightly more general setting. We assume that $\Phi : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ is a cellular automaton that washes out finite islands of errors on a configuration z in linear time; that is, there is a constant m such that $\Phi^{ml}x = \Phi^{ml}z$ for any finite perturbation of x for which $\mathrm{diff}(z,x)$ has diameter at most l. For GKL and modified traffic, z can be either $\underline{0}$ or $\underline{1}$, which are fixed points (hence $\Phi^{ml}z = z$), and the constant m can be chosen to be 2.

The above eroder property automatically implies that Φ also washes out (possibly infinite) sets of error that are "sparse enough". Indeed, an island of errors which is well separated from the rest of the errors will disappear before sensing or affecting the rest of the error set. We are interested in an appropriate notion of "sparseness" for $\mathrm{diff}(z,x)$ that guarantees the attraction of the trajectory of x towards the trajectory of z.

To elaborate this further, let us denote the neighborhood radius of Φ by r. Consider an arbitrary configuration x and think of it as a perturbation of z with errors occurring at sites in $\mathrm{diff}(z,x)$. Let $I \subseteq \mathbb{Z}$ be an interval of length l such that x agrees with z on a margin of width $2rml$ around I, that is, $x_j = z_j$ for $j \in \mathbb{Z} \setminus I$ within distance $2rml$ from I. We call such an interval an *isolated island* (of errors) on x. Let y be a configuration obtained from x by *erasing* the errors on I, that is, by replacing x_i with z_i for each $i \in I$. (Note on terminology: we shall use "erasure" to refer to this abstract construction of one configuration from another, and reserve the word "washing" for what the cellular automaton does.) Observe that within ml steps, the distinction between x and y disappears and we have $\Phi^{ml}x = \Phi^{ml}y$ (see Figure 2). Namely, the island I is washed out before time ml and the sites in $\mathrm{diff}(z,x) \setminus I = \mathrm{diff}(z,y) \setminus I$ do not get a chance to feel the distinction between x and y.

We find that erasing an isolated island of length at most l from x does not affect whether the trajectory of x is attracted towards the trajectory of z or not. Neither does erasing several (possibly infinitely many) isolated islands of length $\leq l$ at the same time. On the other hand, erasing some isolated islands from x makes the error set $\mathrm{diff}(z,x)$ sparser and possibly turns larger portions of $\mathrm{diff}(z,x)$ into isolated islands (see Figure 3). Hence, we can perform the erasure procedure recursively, by first erasing the isolated islands of length 1, then erasing the isolated islands of length 2, then erasing the isolated islands of length 3 and so

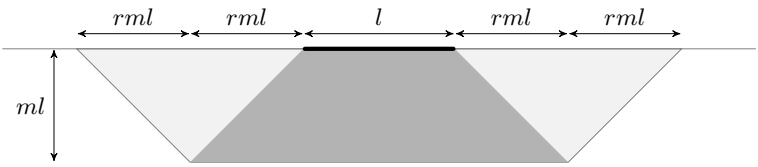


Fig. 2. Forgetting an isolated region of errors

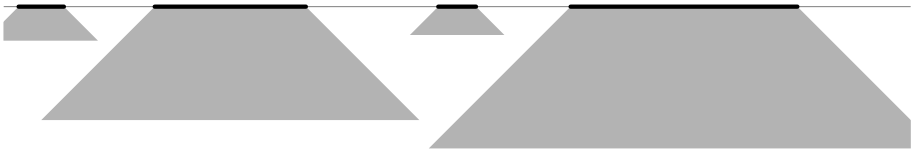


Fig. 3. Washing out a sparse set of errors

forth. In this fashion, we obtain a sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ with $x^{(0)} = x$ and $\mathrm{diff}(z, x^{(l)}) \supseteq \mathrm{diff}(z, x^{(l+1)})$ obtained by successive erasure of isolated islands. We say that the error set $\mathrm{diff}(z, x)$ is *sparse* if all errors are eventually erased, that is, if $\bigcap_l \mathrm{diff}(z, x^{(l)}) = \varnothing$.

However, this notion of sparseness still does not guarantee the attraction of the trajectory of x towards the trajectory of z. (The trajectory of x is considered to be *attracted* towards the trajectory of z if for each site i, there is a time t_i such that $(\Phi^t x)_i = (\Phi^t z)_i$ for all time steps $t \geq t_i$. If $\Phi z = z$, this attraction becomes equivalent to the convergence $\Phi^t x \to z$.) Note that it is well possible that all errors are eventually washed out from x (hence, their information is lost) but the washing out procedure for larger and larger islands affects a given site i indefinitely, so that $(\Phi^t x)_i \neq (\Phi^t z)_i$ for infinite many time steps t (see Figure 4).

To clarify this possibility, note that an isolated island of length l can affect the state of sites within distance rml up to time ml (see Figure 2). Let us denote by $A_l \triangleq \mathrm{diff}(z, x^{(l)}) \setminus \mathrm{diff}(z, x^{(l-1)})$ the union of isolated islands of length l that are erased from $x^{(l-1)}$ during the l'th stage of the erasure procedure. The only possibility for a site i to have a value other than $(\Phi^t z)_i$ at time t is that site i is within distance rml from A_l for some l satisfying $ml > t$. In this case, we say that i is within the *territory* of such A_l. A sufficient condition for the attraction of the trajectory of x towards the trajectory of z is that the error set $\mathrm{diff}(z, x)$ is sparse, and furthermore, each site i is within the territory of A_l for at most finitely many values of l. If this condition is satisfied, we say that the error set $\mathrm{diff}(z, x)$ is *strongly sparse*. In summary, the trajectory of x is attracted towards the trajectory of z if $\mathrm{diff}(z, x)$ is *strongly sparse*.

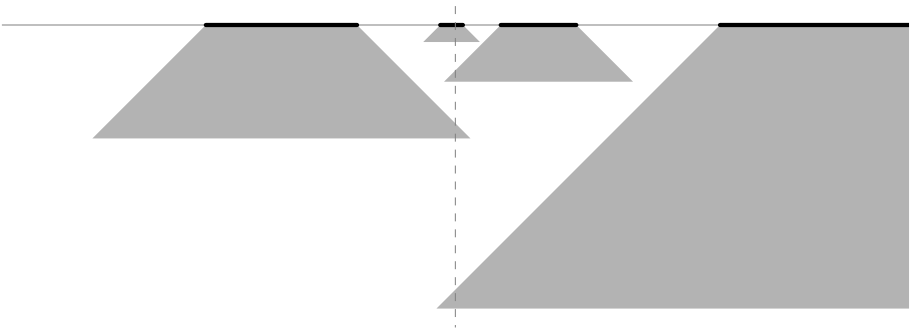


Fig. 4. Washing out but not attracting

4 Sparseness

The notion of (strong) sparseness described in the previous section can be formulated and studied without reference to cellular automata, and that is what we are going to do now. This notion is of independent interest, as it commonly arises in error correcting scenarios. More sophisticated applications appear in [5,6] and [2]. Our exposition is close to that of [2].

We refer to a finite interval $I \subseteq \mathbb{Z}$ as an *island*. Let k be a fixed positive integer. The *territory* (or the *interaction range*) of an island I of length l is the set of sites $i \in \mathbb{Z}$ that are within distance kl from I. We denote the territory of I by $R(I)$. Two disjoint islands I and I' of lengths l and l', where $l \leq l'$, are considered *well separated* if $I' \cap R(I) = \varnothing$, that is, if the larger island does not intrude the territory of the smaller one. A set $E \subseteq \mathbb{Z}$ is said to be *sparse* if it can be covered by a family $\mathcal{I}$ of (disjoint) pairwise well-separated islands. A sparse set is *strongly sparse* if the cover $\mathcal{I}$ can be chosen so that each site i is in the territory of at most finitely many elements of $\mathcal{I}$.

Note that for $k \triangleq 2rm$, we get essentially the same notion of sparseness as in the previous section. Indeed, let $\mathcal{I}_l$ be the sub-family of $\mathcal{I}$ containing all islands of length at most l, and denote by $E_l \triangleq E \setminus \bigcup_{I \in \mathcal{I}_l} I$ the subset of E obtained by erasing the islands of length at most l. Then, every island $I \in \mathcal{I}$ having length l is *isolated* in E_{l-1}, because its territory is not intruded by $E_{l-1} \setminus I$. The new notion of strong sparseness might be slightly more restrictive, as we define the territory by the constant $k = 2rm$ rather than $k/2 = rm$, but the arguments below are not sensitive to this distinction.

The most basic observation about sparseness is its monotonicity.

Proposition 1 (Monotonicity). *Any subset of a (strongly) sparse set is (strongly) sparse.*

One expects a "small" set to be sparse. The following theorem due to Levin [11] is an indication of this intuition.

Theorem 2 (Sparseness of small sets [11]). *There are constants $\varepsilon, c \in (0,1)$ depending on the sparseness parameter k such that every periodic set $E \subseteq \mathbb{Z}$ with period n and at most $c\,n^{\varepsilon}$ elements per period is strongly sparse.*

The reverse intuition is misleading: a sparse set does not need to be "small". In fact, there are sets with arbitrarily large densities that are sparse. The existence of such sets is demonstrated by Kari and Le Gloannec [8], and in special cases, was also noted by Levin [11] and Kůrka [9].

Theorem 3 (Large sparse sets [8]). *There are periodic subsets of $\mathbb{Z}$ with density arbitrarily close to 1 that are strongly sparse.*

It immediately follows that the set of possible densities of strongly sparse (periodic) subsets of $\mathbb{Z}$ is dense in $[0,1]$. A more important corollary is a strengthening of the impossibility result of Land and Belew [10] for cellular automata with *linear-time* eroder property: for any such automaton, there are configurations x with density $\rho(x)$ close to any number in $[0,1]$ that are incorrectly classified.

The main result of interest for us is the sparseness of sufficiently biased Bernoulli random sets.

Theorem 4 (Sparseness of Bernoulli sets [2,5,6]). *A Bernoulli random set $E \subseteq \mathbb{Z}$ with parameter p is almost surely strongly sparse as long as $p < (2k)^{-2}$, where k is the sparseness parameter.*

Proof. For a set $E \subseteq \mathbb{Z}$, we recursively construct a family $\mathcal{I}$ of pairwise well-separated islands as a candidate for covering E. The family $\mathcal{I}$ will be divided into sub-families $\mathcal{J}_l$ consisting of islands of length l, and E_l will be the set obtained by erasing the selected islands of length at most l from E. Let $E_0 \triangleq E$. For $l \geq 1$, recursively define $\mathcal{J}_l$ as the family of islands $I \subseteq \mathbb{Z}$ of length l that intersect E_{l-1} and are isolated in E_{l-1} (i.e., $E_{l-1} \setminus I$ does not intersect the territory of I), and set $E_l \triangleq E_{l-1} \setminus \bigcup_{I \in \mathcal{J}_l} I$. Let $\mathcal{I} \triangleq \bigcup_l \mathcal{J}_l$.

To see that the elements of $\mathcal{I}$ are pairwise well separated, let us first argue that every island $I \in \mathcal{J}_l$ is minimal, in that, it is the smallest interval containing $I \cap E_{l-1}$. Indeed, let $J \subseteq I$ be the smallest island containing $I \cap E_{l-1}$, and assume that $|J| < l$. Then, the endpoints of J must be in E_{l-1}. Therefore, every island $I' \in \mathcal{J}_{l'}$ with $l' < l$ must have been at distance more than kl' from J, for otherwise, I' would not have been isolated in $E_{l'-1}$. In particular, for l' satisfying $|J| \leq l' < l$, the island J has distance more than $k\,|J|$ from every $I' \in \mathcal{J}_{l'}$. Since the distance between J and E_{l-1} is also more than $kl \geq k\,|J|$, it follows that J is isolated in $E_{|J|-1}$. On the other hand, J intersects $E_{|J|-1}$, because it intersects E_{l-1} and $E_{l-1} \subseteq E_{|J|-1}$. We find that $J \in \mathcal{J}_{|J|-1}$, which is a contradiction. Therefore, I is minimal. The well-separation of two islands $I \in \mathcal{J}_l$ and $I' \in \mathcal{J}_{l'}$ with $l \leq l'$ follows from the minimality of I'. We conclude that the elements of $\mathcal{I}$ are also well separated.

Now, let E be a Bernoulli random configuration with parameter p. We choose an appropriate sequence $0 < l_1 < l_2 < l_3 < \cdots$ (to be specified more explicitly below) and observe whether a site u is in E_{l_n}. We will show that the probability

that site u is in E_{l_n} is double exponentially small, that is, $\mathbb{P}(u \in E_{l_n}) \leq \alpha^{2^n}$ for some $\alpha < 1$.

Let u be an arbitrary site. In order for u to be in E_{l_n}, it is necessary that u is also in E_{l_n-1}, and furthermore, u is not covered by any island in $\mathcal{J}_{l_n}$. Therefore, $E_{l_{n-1}}$ (which includes E_{l_n-1}) must contain two elements $u_0 \triangleq u$ and u_1 that are farther than $l_n/2$ from each other but no farther than $(k+1/2)l_n$ from each other (see Figure 5). In a similar fashion, in order for u_0 and u_1 to be in $E_{l_{n-1}}$,

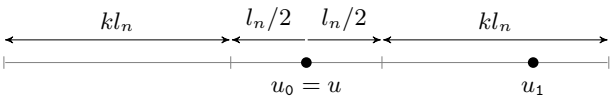


Fig. 5. An evidence for $u \in E_{l_n}$ in $E_{l_{n-1}}$ (see the proof of Theorem 4)

the set $E_{l_{n-2}}$ must contain elements $u_{00} \triangleq u_0$, u_{01}, $u_{10} \triangleq u_1$ and u_{11} such that

$$\frac{1}{2}l_{n-1} < d(u_{00}, u_{01}) \leq \Big(k + \frac{1}{2}\Big)l_{n-1} \,, \tag{3}$$

$$\frac{1}{2}l_{n-1} < d(u_{10}, u_{11}) \leq \Big(k + \frac{1}{2}\Big)l_{n-1} \,. \tag{4}$$

Repeating this procedure, we find a binary tree of depth n with roots in $E_0 = E$ that provides an evidence for the presence of u in E_{l_n}. We call such a tree an *explanation tree*. Thus, in order to have $u \in E_{l_n}$, there must be at least one explanation tree for it.

We estimate the probability of the existence of an explanation tree for $u \in E_{l_n}$. Let $T = (u, u_0, u_1, u_{00}, u_{01}, \ldots, u_{11\cdots0}, u_{11\cdots1})$ be a *candidate* explanation tree, that is, a tree with the right distances between the nodes. To simplify the estimation, we choose the lengths $l_1, l_2, \ldots$ in such a way to make sure that the leaves of T are distinct elements of $\mathbb{Z}$. A sufficient condition for the distinctness of the leaves of T is that for each m,

$$\frac{1}{2}l_m \geq 2\Big(k + \frac{1}{2}\Big)(l_{m-1} + l_{m-2} + \cdots + l_1) \,. \tag{5}$$

This would guarantee that the two subtrees descending from each node do not intersect. We choose $l_m \triangleq (4k+3)^{m-1}$, which is a solution of the above system of inequalities.

A candidate tree T is an explanation tree for $u \in E_{l_n}$ if and only if all its leaves are in E. Whether or not each leaf u_w of T is in E is determined by a biased coin flip with probability p of falling in E. With the above choice of l_m, the events $u_w \in E$ for different leaves of T are independent. It follows that T is an explanation tree for $u \in E_{l_n}$ with probability p^{2^n}.

Let us now estimate the number of candidate trees of depth m. Denote this number by f_m. Observe that f_m satisfies the recursive inequality

$$f_m \leq 2kl_m\, f_{m-1}^2 \tag{6}$$

with $f_0 \triangleq 1$. Indeed, $2kl_m$ counts for the number of possible positions for u_1 and f_{m-1}^2 counts the number of possibilities for each of the two subtrees. Letting $g_m \triangleq \log f_m$, we have

$$g_m \leq a\, m + b + 2g_{m-1}\ , \tag{7}$$

where $a \triangleq \log(4k+3)$ and $b \triangleq \log 2k - \log(4k+3)$. Expanding the last recursion we get

$$g_m \leq 2^m(2b + a\sum_{i=0}^{m} \frac{i}{2^i}) \tag{8}$$

$$\leq 2^m(2b + a\sum_{i=0}^{\infty} \frac{i}{2^i}) \tag{9}$$

$$= 2^{m+1}(a+b)\ . \tag{10}$$

Therefore,

$$f_m \leq (2k)^{2^{m+1}}\ . \tag{11}$$

By the sub-additivity of the probabilities, we find that the probability of the existence of at least one explanation tree for $u \in E_{l_n}$ satisfies

$$\mathbb{P}(u \in E_{l_n}) \leq p^{2^n} f_n \leq \alpha^{2^n}\ , \tag{12}$$

where $\alpha \triangleq p(2k)^2$. Since $p < (2k)^{-2}$, we get $\alpha < 1$.

The probability that a given site $u \in \mathbb{Z}$ is in E but is not covered by $\mathcal{I}$ (i.e., never erased) is

$$\mathbb{P}(u \in \bigcap_l E_l) = \mathbb{P}(u \in \bigcap_n E_{l_n}) = \lim_{n\to\infty} \mathbb{P}(u \in E_{l_n}) = \lim_{n\to\infty} \alpha^{2^n} = 0\ . \tag{13}$$

Since $\mathbb{Z}$ is countable, we find, by sub-additivity, that $\mathbb{P}(\bigcap_l E_l \neq \varnothing) = 0$, which means, E is sparse with probability 1.

That E is strongly sparse with probability 1 follows by the Borel-Cantelli argument. Namely, the event that a site u is in the territory of infinitely many islands $I \in \mathcal{I}$ can be expressed as $\bigcap_m \bigcup_{n\geq m} \{d(u, E_{l_n}) \leq kl_n\}$. (Note that an island covering a site in E_{l_n} has length greater than l_n.) The probability that u is within distance kl_n from E_{l_n} satisfies

$$\mathbb{P}\big(d(u, E_{l_n}) \leq kl_n\big) \leq (2kl_n + 1)\alpha^{2^n} = (2k(4k+3)^{n-1} + 1)\alpha^{2^n}\ . \tag{14}$$

Therefore,

$$\mathbb{P}\Big(\bigcup_{n\geq m} \{d(u, E_{l_n}) \leq kl_n\}\Big) \leq \sum_{n\geq m} (2k(4k+3)^{n-1} + 1)\alpha^{2^n} < \infty\ . \tag{15}$$

It follows that

$$\mathbb{P}\Big(\bigcap_m \bigcup_{n\geq m} \{d(u, E_{l_n}) \leq kl_n\}\Big) \leq \lim_{m\to\infty} \sum_{n\geq m} (2k(4k+3)^{n-1}+1)\alpha^{2^n} = 0\,. \quad (16)$$

Using again the countability of $\mathbb{Z}$, we find that, with probability 1, no site u is in the territory of more than finitely may islands $I \in \mathcal{I}$. That is, E is almost surely strongly sparse. □

Theorem 4, along with a standard application of monotonicity, shows that when the Bernoulli parameter is varied, a non-trivial phase transition occurs.

Corollary 1 (Phase transition). *There is a critical value $p_c \in (0,1]$ depending on the sparseness parameter k such that a Bernoulli random set $E \subseteq \mathbb{Z}$ with parameter p is almost surely strongly sparse if $p < p_c$ and is almost surely not strongly sparse if $p > p_c$.*

Proof. First, observe that the (strong) sparseness of E is a translation-invariant event (i.e., for $a \in \mathbb{Z}$, the sparseness of $a + E$ is equivalent to the sparseness of E). Therefore, by ergodicity, the probability that a Bernoulli random set is (strongly) sparse is either 0 or 1.

The presence of a threshold value $p_c \in [0,1]$ (possibly 0) is a standard consequence of monotonicity. Indeed, let $U_i, i \in \mathbb{Z}$ be a collection of independent random variables with uniform distribution on the real interval $[0,1]$. For $p \in [0,1]$, define a set $E^{(p)} \triangleq \{i \in \mathbb{Z} : U_i < p\}$. Then, $E^{(p)}$ is a Bernoulli random set with parameter p, and the collection of sets $E^{(p)}$ is increasing in p. Let $p_c \triangleq \sup\{p : E^{(p)}$ is almost surely (strongly) sparse$\}$. By monotonicity, the set $E^{(p)}$ is almost surely (strongly) sparse for $p < p_c$ and is almost surely not (strongly) sparse for $p > p_c$.

Finally, we know from Theorem 4 that $p_c > 0$. □

5 Restricted Classification

Let us state the claimed result of this paper explicitly as a corollary of Theorem 4 and the discussions in the previous sections.

Corollary 2 (Restricted classification). *Let $\Phi : \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ be a cellular automaton that washes out finite islands of errors on either of the two uniform configurations $\underline{0}$ and $\underline{1}$ in linear time. Namely, suppose that there is a constant m such that for every finite perturbation x of $\underline{0}$ for which $\mathrm{diff}(\underline{0}, x)$ has diameter at most l, we have $\Phi^{ml}x = \underline{0}$, and similarly for $\underline{1}$. Then, there is a constant $p_c \in (0, 1/2]$ such that Φ classifies a Bernoulli random configuration with parameter $p \in [0, p_c) \cup (1 - p_c, 1]$ almost surely correctly.*

For GKL and modified traffic, we have $k = 2rm = 12$. Therefore, Theorem 4 only guarantees correct classification if the Bernoulli parameter p is within distance $(2k)^{-2} = 24^{-2} \approx 0.0017$ from either 0 or 1.

6 Discussion

We conclude with few comments and questions.

Corollary 2 shows that the asymptotic behaviour of the GKL and modified traffic automata starting from a Bernoulli random configuration undergoes a phase transition: the cellular automaton converges to $\underline{0}$ for p close to 0 and to $\underline{1}$ for p close to 1. It remains open whether the transition occurs precisely at $p = 1/2$, or if there are other transitions in between. The result of Bušić et al. [1] shows that the transition in the NEC cellular automaton is unique and happens precisely at $p = 1/2$.

Another open issue is the behaviour of the GKL and modified traffic automata on random configurations with non-Bernoulli distributions. One might expect the sparseness argument to extend to measures that are sufficiently mixing. For instance, it should be possible to show the same kind of classification on a Markov random configuration that has density close to 0 or 1.

It would also be interesting to see if the sparseness method can be applied to probabilistic cellular automata that are suggested for the density classification task. Fatès [3] has introduced a parametric family of one-dimensional probabilistic cellular automaton with a density classification property: for every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is a setting of the parameter such that the automaton classifies a periodic configuration with period n with probability at least $1 - \varepsilon$. Does the majority-traffic rule of Fatès with a fixed parameter classify sufficiently biased Bernoulli random configurations? A two-dimensional candidate would be the noisy version of the nearest-neighbor majority rule, in which the noise occurs only when there is no consensus in the neighborhood.

Finally, given its various applications, one might try to study the notion of sparseness in a more systematic fashion, trying to capture more details about the transition. It is curious that the notion of sparseness of Bernoulli random sets supports a hierarchy of phase transitions, even in one dimension where the standard notion of percolation fails.

Acknowledgments. Research supported by ERC Advanced Grant 267356-VARIS of Frank den Hollander. I would like to thank Jarkko Kari for suggesting this problem and for discussions that lead to this paper.

References

1. Bušić, A., Fatès, N., Mairesse, J., Marcovici, I.: Density classification on infinite lattices and trees. Electronic Journal of Probability **18**(51), 1–22 (2013)
2. Durand, B., Romashchenko, A., Shen, A.: Fixed-point tile sets and their applications. Journal of Computer and System Sciences **78**, 731–764 (2012)
3. Fatès, N.: Stochastic cellular automata solutions to the density classification problem. Theory of Computing Systems **53**(2), 223–242 (2013)
4. Gach, P., Kurdyumov, G.L., Levin, L.A.: One-dimensional uniform arrays that wash out finite islands. Problems of Information Transmission **14**(3), 223–226 (1978)

5. Gács, P.: Reliable computation with cellular automata. Journal of Computer and System Sciences **32**, 15–78 (1986)
6. Gács, P.: Reliable cellular automata with self-organization. Journal of Statistical Physics **103**(1/2), 45–267 (2001)
7. de Sá, G.P., Maes, C.: The Gacs-Kurdyumov-Levin automaton revisited. Journal of Statistical Physics **67**(3/4), 507–522 (1992)
8. Kari, J., Le Gloannec, B.: Modified traffic cellular automaton for the density classification task. Fundamenta Informaticae **116**, 1–16 (2012)
9. Kůrka, P.: Cellular automata with vanishing particles. Fundamenta Informaticae **58**, 203–221 (2003)
10. Land, M., Belew, R.K.: No perfect two-state cellular automata for density classification exists. Physical Review Letters **74**(25), 5148–5150 (1995)
11. Levin, L.A.: Self-stabilization of circular arrays of automata. Theoretical Computer Science **235**(1), 143–144 (2000)
12. Toom, A.L.: Nonergodic multidimensional systems of automata. Problems of Information Transmission **10**(3), 239–246 (1974)
13. Toom, A.L.: Stable and attractive trajectories in multicomponent systems. In: Dobrushin, R.L., Sinai, Y.G. (eds.) Multicomponent Random Systems, pp. 549–575. Marcel Dekker (1980)

Recognition of Linear-Slender Context-Free Languages by Real Time One-Way Cellular Automata

Véronique Terrier(✉)

GREYC, UMR 6072, Université de Caen Basse-Normandie Campus Côte de Nacre, boulevard Maréchal Juin, 14032 Caen, France
veronique.terrier@unicaen.fr

Abstract. A linear-slender context-free language is a context-free language whose number of words of length n is linear in n. Its structure has been finely characterized in a work of Ilie, Rozenberg and Salomaa. Thanks to this characterization, we show that every linear-slender context-free language is recognizable by a real time one-way cellular automaton.

1 Introduction

One-way cellular automaton (OCA) is one of the simplest parallel recognizing devices. Since its introduction by Dyer [5], it has been the subject of much interest. Numerous studies have been directed towards the real time OCA, the class of languages decidable in minimal time, and have already provided a good understanding of its abilities and limitations.

In particular, it is known that the real time OCA and the family of context free languages are incomparable [12]. This calls into question which languages they have in common. That is not the case for deterministic CFL: there exits a LL(1) CFL which is not a real time OCA one [11]. On the other hand, linear CFL and visible pushdown languages have been proved to be real time OCA ones [3,11].

With the aim to identify more common languages, we will take into consideration the counting function that measures the number of words of length n in the language. Noticing that all CFL known to be not real time OCA have their counting function of exponential order, the best is to look at sparse languages. These are the poly-slender CFL whose counting functions are polynomial in n and, more specifically, the linear-slender CFL whose counting functions are linear. The purpose here is to show that linear-slender CFL are real time OCA ones.

The present work essentially relies on a paper by Ilie, Rozenberg and Salomaa which presents a characterization of poly-slender CFL in terms of Dyck loops [7]. Čulík's OCA which recognizes in real time the language $\{a^n b^{n+m} a^m : n, m \geq 0\}$ and Okhotin's characterization of real time OCA by linear conjunctive grammars will come also into play [2,10].

J. Kari (Ed.): AUTOMATA 2015, LNCS 9099, pp. 251–262, 2015.
DOI: 10.1007/978-3-662-47221-7_19

2 Preliminary

2.1 Basic Notions

We first recall some basic definitions and notations.

For any language L, the number of words in L of length n is denoted by $\sharp_n(L)$.
For an integer k, a language L is *k-poly-slender* if $\sharp_n(L)$ is in $\mathcal{O}(n^k)$.
A language L is *poly-slender* if L is k-poly-slender for some k. A language L is *linear-slender* if $\sharp_n(L)$ is in $\mathcal{O}(n)$.

A language L is *bounded* if there exists a finite number of words $u_1, u_2, \cdots, u_k$ such that $L \subseteq u_1^* u_2^* \cdots u_k^*$.
For any word $w \in \Sigma^+$, the primitive root of w is denoted by $\rho(w)$ and corresponds to the minimal word such that $w \in (\rho(w))^*$.

2.2 Poly-slender Context Free Languages

We review the definitions and results presented in [7] that will be fundamental ingredients throughout this paper.

Definition 1 (Dyck loop). *Let $z = z_1 z_2 \cdots z_{2k}$ be a Dyck word on $\{[,]\}$ of length $2k$. Let l_i and r_i denote the respective positions of the i-th opening parenthesis [and its corresponding closing one] in z. Given some words $y_0, \cdots, y_{2k}$, $x_1, \cdots, x_{2k}$, consider the map $h_{n_1,\cdots,n_k}(z_1 z_2 \cdots z_{2k}) = y_0 x_1^{e_1} y_1 x_2^{e_2} y_2 \cdots x_{2k}^{e_{2k}} y_{2k}$ where each pair of parentheses shares the same exponent: $e_{l_i} = e_{r_i} = n_i$.*
A k-Dyck loop *is any set $D = \{h_{n_1,\cdots,n_k}(z_1 z_2 \cdots z_{2k}) \colon n_i \geq 0\}$ for some underlying Dyck word $z_1 z_2 \cdots z_{2k}$ and words x_i, y_i.*

Example 1. $\{ab^{n_1}a(ba)^{n_2+1}a^{n_2+1}(ba)^{n_1+n_3+2}b^{n_3} \colon n_1, n_2, n_3 \geq 0\}$ is a 3-Dyck loop for the underlying Dyck word $[[]][]$ and words $y_0 = a, x_1 = b, y_1 = aba$, $x_2 = ba, y_2 = a, x_3 = a, y_3 = \varepsilon, x_4 = ba, y_4 = bab, x_5 = ab, y_5 = a, x_6 = b, y_6 = \varepsilon$.

Example 2. $\{a^{n_1}b^{n_1+n_2}a^{n_2} \colon n_1, n_2 \geq 0\}$ is a 2-Dyck loop for the underlying Dyck word $[][]$ and words $x_1 = x_4 = a, x_2 = x_3 = b, y_0 = y_1 = y_2 = y_4 = \varepsilon$.

The structure of poly-slender CFL finely corresponds to Dyck loops:

Theorem 1. *For any $k \geq 0$, a context-free language is k-poly-slender if and only if it is a finite union of $(k+1)$-Dyck loops.*

The following notion captures whether the position of some word w can be distinguished or not.

Definition 2 (Link). *Let $u, v \in \Sigma^+$ and $w \in \Sigma^*$. The word w* links *u with v (*`link`*(u, w, v)) if and only if $\rho(u)w = w\rho(v)$. That means $\rho(u) = pq$, $\rho(v) = qp$ and $w = (pq)^* p$ for some p, q. And so $u^m w v^n$ is a prefix of $(pq)^*$*

Example 3. With x_i and y_i as defined in *Example 1*, $\texttt{link}(x_4, y_4, x_5)$ holds but $\texttt{link}(x_i, y_i, x_{i+1})$ does not hold for $i = 1, 2, 3, 5$.
With x_i and y_i as defined in *Example 2*, $\texttt{link}(x_2, y_2, x_3)$ holds but neither $\texttt{link}(x_1, y_1, x_2)$ nor $\texttt{link}(x_3, y_3, x_4)$ holds.

We will make great use of the following lemma:

Lemma 1. *Consider some words $x_i \in \Sigma^+$ and $y_i \in \Sigma^*$, and some non-negative integers n_i, m_i. Denote $w = y_0 x_1^{n_1} y_1 x_2^{n_2} y_2 \cdots x_r^{n_r} y_r$, $w' = y_0 x_1^{m_1} y_1 x_2^{m_2} y_2 \cdots x_r^{m_r} y_r$ and assume that $\texttt{link}(x_i, y_i, x_{i+1})$ holds for no i.*
Then there is a constant N_0, depending only on the lengths of the words x_i and y_i, such that, if n_i, m_i are larger than N_0 and there is i with $n_i \neq m_i$, then $w \neq w'$.

For a general overview, we recall the result of Latteux and Thierrin [8] and the one of Ginsburg and Spanier [6]:

Theorem 2. *A context-free language is poly-slender if and only if it is bounded.*

Theorem 3. *The family of bounded languages is the smallest family which contains all finite languages and is closed under the following operations:*

1. *union*
2. *catenation*
3. $(x, y)^\star L = \bigcup_{n \geq 0} x^n L y^n$ *for any x, y words*

2.3 Real Time One-Way Cellular Automaton

A *one-way cellular automaton* is a one-dimensional array of finite automata (the cells) indexed by $\mathbb{N}$. The cells evolve synchronously at discrete time steps. Each cell takes one value from a finite set of states Q. At each step, the evolution of a cell is defined by its own state and the state of its right neighbor according to a transition function δ. Formally, denoting $\langle c, t \rangle$ the state of the cell c at time t, $\langle c, t+1 \rangle = \delta(\langle c, t \rangle, \langle c+1, t \rangle)$.

In order that an OCA acts as a language recognizer, we specify two subsets of Q: the alphabet Σ of input symbols and the set of accepting states Q_{accept}. The input mode is parallel. At initial time 0, the i-th bit of the input word $w \in \Sigma^*$ is fed to the cell i: $\langle i, 0 \rangle = w_i$. An OCA is said to accept (resp. reject) a word w in real time, if on input w the cell 0 enters an accepting (resp. non-accepting) state at time $|w| - 1$. It corresponds to the minimal time for the cell 0 to know the whole input. A language is a *real time OCA language* if there exists some OCA $(Q, \Sigma, Q_{\text{accept}}, \delta)$ which accepts in real time exactly the words $w \in L$.

The computation of an OCA is usually represented by a time-space diagram (see Fig 1). The t-row corresponds to the cellular array at time t. Only those sites involved in the real time computation are depicted. As a matter of fact, there are two topologically equivalent ways to display the time-space diagram. We will use here the bilateral symmetric layout, i.e., the right one.

The real time OCA class is robust: introducing the notion of conjunctive grammar, Okhotin has exhibited a characterization of real time OCA in terms

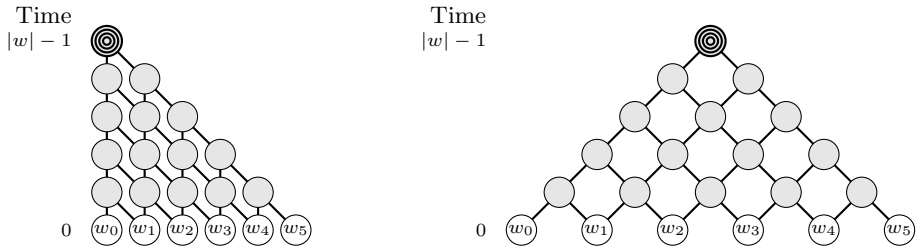


Fig. 1. Time-space diagram of a real time OCA

of a generating device [10]. Let us recall its statement, it will be one of the main ingredients in this paper. An alternating grammar is a grammar enhanced with a conjunctive operation symbolized by &. Denoting N the set of variables and Σ the set of terminals, each production is of the form $A \to \alpha_1 \,\&\, \cdots \,\&\, \alpha_k$ where $A \in N$ and $\alpha_1, \cdots, \alpha_k \in (A \cup \Sigma)^*$. Such a production denotes that the language generated by A is the intersection of the languages generated by $\alpha_1, \cdots, \alpha_k$. Analogously to linear context free grammars, a linear conjunctive grammar is defined as an alternating grammar with the restriction that for every production $A \to \alpha_1 \,\&\, \cdots \,\&\, \alpha_k$, no α_i has more than one instance of a variable. An algorithm describing how to recognize any linear CFL on a real time OCA was already known [3]. More radically, to extend linear context free grammars with the conjunctive operation & leads to a complete characterization of real time OCA [10]:

Theorem 4. *A language L is recognized in real time by an OCA if and only if L is generated by a linear conjunctive grammar.*

Let us recall the conversion from a real time OCA to a linear conjunctive grammar that we will need later. See Fig. 2 to get some insight about the translation.

Lemma 2. *Given any language L recognizable by some real time OCA $\mathcal{A} = (Q, \Sigma, Q_{accept}, \delta)$, L is generated by the linear conjunctive grammar $\mathcal{G} = (\Sigma, \{S\} \cup \{A_q : q \in Q\}, S, \mathcal{R})$ where $\mathcal{R}$ contains the following rules:*

$$S \to A_q \text{ for all } q \in Q_{accept}$$
$$A_a \to a \text{ for all } a \in \Sigma$$
$$A_{\delta(g,d)} \to A_g b \,\&\, c A_d \text{ for all } g, d \in Q \text{ and all } b, c \in \Sigma$$

At last, let us give some examples and properties of real time OCA to illustrate their abilities and limits. The Dyck language, the linear-slender CFL $\{a^n b^{n+m} a^m : n, m \geq 0\}$, the inherently exponentially ambiguous CFL L^* with $L = \{a^i b^j c^k : i = j \text{ or } j = k\}$ [9], the non CFL languages $\{a^n b^n c^n : n \geq 0\}$ and $\{a^n b^{2^n} : n \geq 0\}$ with a non semilinear Parikh image, are all real time OCA languages. In addition, real time OCA languages include all linear CFL and visible pushdown languages [3,11]. On the other hand, several languages are known not to be real time OCA: all non regular languages over a one letter alphabet,

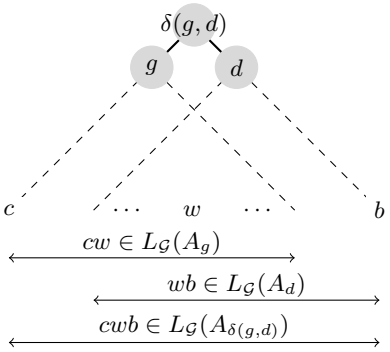


Fig. 2. The OCA's transition $\delta(g,d)$ is converted into the rule $A_{\delta(g,d)} \to A_g b \,\&\, cA_d$

the inherently ambiguous CFL L_1L_1 square of the linear CFL $L_1 = \{1^k 0 u 1 0^k : k > 0, u \in \{0,1\}^*\}$, the language $\{uvu : u,v \in \{0,1\}^*, |u| > 1\}$, the deterministic CFL (to be more precise: LL(1) language) $\{c^m a^{l_0} b a^{l_1} b \cdots a^{l_m} b \cdots a^{l_z} b d^n : m,n,l_i \geq 0, z \geq 1, l_m = n\}$ [1,11–13].

Further, the real time OCA class is closed under boolean operations, reverse, $\star$ operation (the one defined in Theorem 3), left and right concatenation with regular languages. The proofs are folklore. A contrario, the real time OCA class is not closed under morphism, concatenation, Kleene star and cycle [4,12,14].

2.4 Poly-slender Context Free Languages and Real Time One-Way Cellular Automata

The question behind this paper is whether the poly-slender CFL are real time OCA languages. According to Theorem 3, the poly-slender CFL are the smallest family which contains all finite languages and is closed under union, concatenation and $\star$ operation. Of course, real time OCA languages include all finite languages and are closed under union and $\star$ operation. This is easy to prove using the grammar characterization of real time OCA. The problematic point is concatenation: can we assert that the concatenation of two Dyck loops is a real time OCA language? We do not answer this question but in the simple case: 1-Dyck loops. Precisely we will show the following result:

Theorem 5. *Linear-slender context-free languages are recognizable in real time by one-way cellular automata.*

The rest of the paper will be devoted to the proof of this theorem. Concretely we have to show that 2-Dyck loops are recognized in real time by OCA.

3 Recognition of 2-Dyck Loops by Real Time OCA

As an introduction, we may observe that 1-Dyck loops are real time OCA languages. Indeed a 1-Dyck loop corresponds to a set $D = \{y_0 x_1^n y_1 x_2^n y_2 : n \in \mathbb{N}\}$

for some words y_0, x_1, y_1, x_2, y_2. It is a linear CFL and so it is a real time OCA language. As a consequence, all constant-slender languages are real time OCA languages.

Let us now consider 2-Dyck loops. According to whether the underlying Dyck word is [[]] or [][], they are of shape $\{y_0x_1^{n_1}y_1x_2^{n_2}y_2x_3^{n_2}y_3x_4^{n_1}y_4 \colon n_1, n_2 \geq 0\}$ or $\{y_0x_1^{n_1}y_1x_2^{n_1}y_2x_3^{n_2}y_3x_4^{n_2}y_4 \colon n_1, n_2 \geq 0\}$ A first simplification is to assume that the two ends y_0 and y_4 are empty knowing that if L is a real time OCA language then so is $\{y_0\}L\{y_4\}$ whatever the words y_0 and y_4 are. We will suppose also that the words x_i are non-empty, the degenerate cases being easy to handle.

3.1 The Underlying Dyck Word Is [[]].

That is the simple case. The corresponding Dyck loop $\{x_1^{n_1}y_1x_2^{n_2}y_2x_3^{n_2}y_3x_4^{n_1} \colon n_1, n_2 \geq 0\}$ is clearly a linear CFL and so it is real time OCA recognizable. Here it is basically the closure under the $\star$ operation which is involved.

3.2 The Underlying Dyck Word Is [][].

Now it is the closure under concatenation of 1-Dyck loops which is involved.

Case 1. $\texttt{link}(x_i, y_i, x_{i+1})$ holds for *no* $i = 1, 2, 3$

Let N_0 be a constant as defined in Lemma 1. The Dyck loop can be divided into three subsets D_1, D_2 and D_3, where

$$D_1 = \{x_1^{n_1}y_1x_2^{n_1}y_2x_3^{n_3}y_3x_4^{n_3} \colon n_1 < N_0, n_3 \in \mathbb{N}\},$$
$$D_2 = \{x_1^{n_1}y_1x_2^{n_1}y_2x_3^{n_3}y_3x_4^{n_3} \colon n_1 \in \mathbb{N}, n_3 < N_0\},$$
$$D_3 = \{x_1^{n_1}y_1x_2^{n_1}y_2x_3^{n_3}y_3x_4^{n_3} \colon n_1, n_3 \geq N_0\}.$$

Observe that $D_1 = \bigcup_{0 \leq i < N_0} \{x_1^i y_1 x_2^i y_2 x_3^n y_3 x_4^n \colon n \in \mathbb{N}\}$ is a finite union of linear CFL's and thus is alinear CFL. The subset D_2 is as well a linear CFL. Further, Lemma 1 ensures that D_3 can be specified as the intersection of two linear CFL's : $D_3 = \{x_1^{n_1}y_1x_2^{n_1}y_2x_3^{n_3}y_3x_4^{n_4} \colon n_1, n_3, n_4 \geq N_0\} \cap \{x_1^{n_1}y_1x_2^{n_2}y_2x_3^{n_3}y_3x_4^{n_3} \colon n_1, n_2, n_3 \geq N_0\}$ and so is real time recognizable by an OCA.

Case 2. $\texttt{link}(x_i, y_i, x_{i+1})$ holds for *one* $i = 1, 2, 3$

The case where y_1 links x_1 with x_2 (or in a symmetric way y_3 links x_3 with x_4) does not present any difficulty. By Definition 2, y_1 links x_1 with x_2 if $\rho(x_1) = pq$, $\rho(x_2) = qp$ and $y_1 = (pq)^\gamma p$ for some words p, q and integer γ. Setting $x_1 = (pq)^\alpha$, $x_2 = (qp)^\beta$, $x_1^{n_1}y_1x_2^{n_2}y_2$ can be rewritten as $(pq)^{(\alpha+\beta)n_1+\gamma}py_2$ and thus specifies a regular language. Moreover $x_3^{n_3}y_3x_4^{n_4} = x_3^{n_3}y_3x_4^{n_3}$ corresponds to a linear CFL. Their concatenation is linear CFL and so real time recognizable by an OCA.

It remains to examine the most technical case where y_2 links x_2 with x_3 but $\texttt{link}(x_i, y_i, x_{i+1})$ does not hold for $i = 1$ and $i = 3$. As y_2 links x_2 with x_3 there exists some p, q and α, β, γ such that $x_2 = (pq)^\alpha$, $x_3 = (qp)^\beta$ and $y_2 = (pq)^\gamma p$.

Then $x_1^{n_1}y_1x_2^{n_2}y_2x_3^{n_3}y_3x_4^{n_4}$ can be rewritten as $x_1^{n_1}y_1(pq)^{\alpha n_1+\beta n_3+\gamma}py_3x_4^{n_3}$. Setting $z_1 = x_1, z_2 = y_1, z_3 = pq, z_4 = py_3, z_5 = x_4$, such 2-Dyck loops can be reshaped into $z_1^{n_1}z_2z_3^{\alpha n_1+\beta n_3+\gamma}z_4z_5^{n_3}$ with the properties that z_2 does not link z_1 with z_3 and, readily verifiable, z_4 does not link z_3 with z_5. So our task is to show that $\{z_1^n z_2 z_3^{\alpha n+\beta m+\gamma} z_4 z_5^m : n, m \geq 0\}$ is a real time OCA language. It will be done in two steps:

1. Given a five letters alphabet $\mathcal{A} = \{a_1, \cdots, a_5\}$, whatever $\alpha, \beta \geq 1$ and $\gamma \geq 0$ are, we will present a real time OCA which recognizes the language $L_{\alpha,\beta,\gamma} = \{a_1^n a_2 a_3^{\alpha n+\beta m+\gamma} a_4 a_5^m : n, m \geq 0\}$.
2. For any homomorphism h on $\mathcal{A}$ such that neither $\texttt{link}(h(a_1), h(a_2), h(a_3))$ nor $\texttt{link}(h(a_3), h(a_4), h(a_5))$ holds, we will verify that $h(L_{\alpha,\beta,\gamma})$ is a real time OCA language.

Proposition 1. *The language* $L_{\alpha,\beta,\gamma} = \{a_1^n a_2 a_3^{\alpha n+\beta m+\gamma} a_4 a_5^m : n, m > 0\}$, *where the symbols* $a_1, \cdots, a_5$ *are distinct, is recognizable in real time by an OCA.*

Proof. The main ingredient is Čulík's OCA which recognizes in real time the language $\{a^n b^{n+m} a^m : n, m \geq 0\}$ [2]. His construction makes ingenious use of a firing squad synchronization.

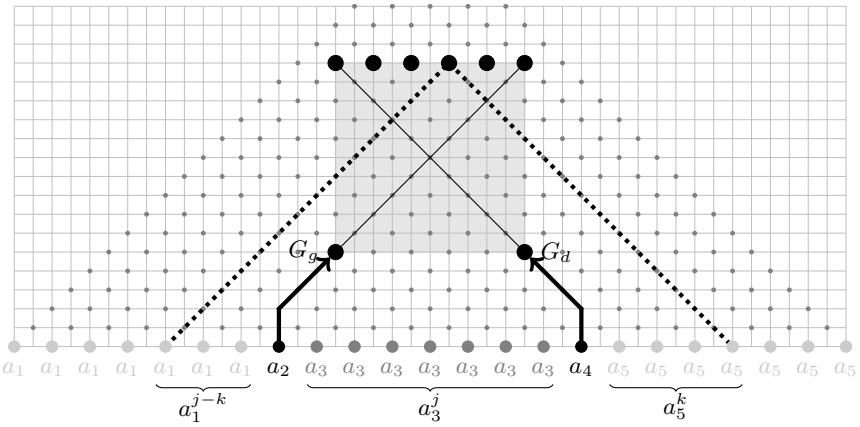
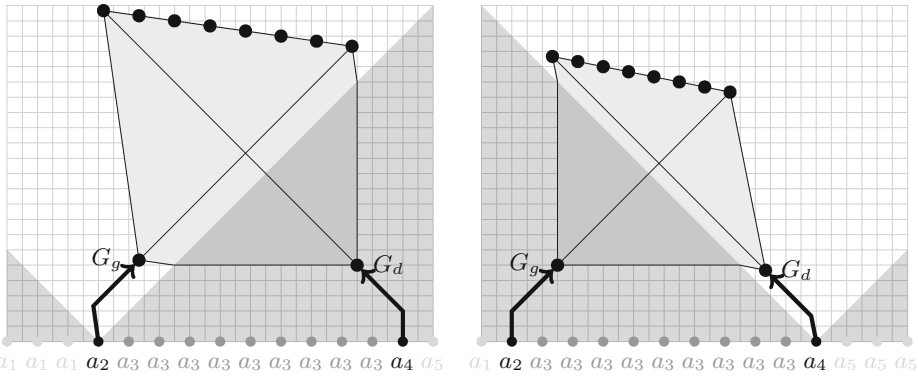


Fig. 3. Čulík's OCA

Let us begin by recalling the process (see Fig. 3). For convenience, we identify the sites of the OCA with integer coordinates (x, y) where $x + y$ is even. In this way, the input symbols are placed at positions $(2x, 0)$. The unique symbol a_2 is chosen to be at the origin $(0, 0)$ and, when the input is of shape $a_1^+ a_2 a_3^j a_4 a_5^+$, the unique symbol a_4 is at $(2j + 2, 0)$. The process is set up using a firing squad synchronization with two generals G_g and G_d located according to the unique symbols a_2 and a_4. Precisely, the left general G_g is at $(3, 5)$ and the right general

G_d at $(2j-1,5)$. Thus the synchronization occurs at points $(1+2k,1+2j)$ for all k with $0 < k < j$. We check that the k-th firing points $(1+2k,1+2j)$ corresponds to the output of the input subword which begins at $(2(k-j),0)$ and ends at $(2(1+j+k),0)$, namely the subword $a_1^{j-k}a_2a_3^ja_4a_5^k$.

Next, Čulík showed that the construction can be adapted to recognize the languages $\{a^ib^ja^k : i,k \geq 0, mj+c = i+k\}$ for all $m \geq 1$, $c \geq 0$. Actually, the same approach works in these more general settings: it exists a real time OCA deciding the language $L_{\alpha,\beta,\gamma} = \{a_1^ia_2a_3^ja_4a_5^k : i,j,k \in \mathbb{N}, j = \alpha i + \beta k + \gamma\}$ for every non-negative rational numbers α, β, γ. Let us outline the modified construction. Be warned that we will use rational coordinates but the technique to revert to an OCA diagram is standard. Firstly observe that we may reduce the range of the synchronization in locating the two generals later: with the left general G_g at $(3+\gamma, 5+\gamma)$ and the right general G_d at $(2j-1-\gamma, 5+\gamma)$, the set of firing points becomes $\{(1+2k+\gamma, 1+2j-\gamma) : k \in \mathbb{N}, 0 < k < j-\gamma\}$. We explain now the construction in terms of geometric transformations. It will be achieved by the composition of two directional scalings (see Fig. 4).



(a) Scaling in the up-left to down-right direction with factor $1/\alpha = 4/3$

(b) Scaling in the up-right to down-left direction with factor $1/\beta = 2/3$

Fig. 4. Two scalings

Initiated from the unique symbol a_2, the first transformation starts at point $(0,0)$ and applies on its cone of consequences $\{(c,t) : |c| \leq t\}$, i.e., the future light cone of $(0,0)$ which encompasses the set of points impacted by $(0,0)$. It leaves the line $t = c$ stable and scales by the factor $1/\alpha$ in the up-left to down-right direction according to the map $M_g = \begin{pmatrix} (1+\alpha)/(2\alpha) & (\alpha-1)/(2\alpha) \\ (\alpha-1)/(2\alpha) & (1+\alpha)/(2\alpha) \end{pmatrix}$. Observe that the transformation is workable. Inside the cone, the neighborhood constraints are satisfied. The right side of the cone is stable. And the computation on the subword a_1^+ occurring below the left side can easily be scaled.

The second transformation is symmetric. Initiated from the unique symbol a_4, it starts at point $(2j+2, 0)$ and applies on its cone of consequences $\{(c,t)\colon |c-2j-2| \leq t\}$. It leaves the line $t+c=2j+2$ stable and scales by the factor $1/\beta$ in the up-right to down-left direction according to the map $M_d = \begin{pmatrix} (1+\beta)/(2\beta) & (1-\beta)/(2\beta) \\ (1-\beta)/(2\beta) & (1+\beta)/(2\beta) \end{pmatrix}$.

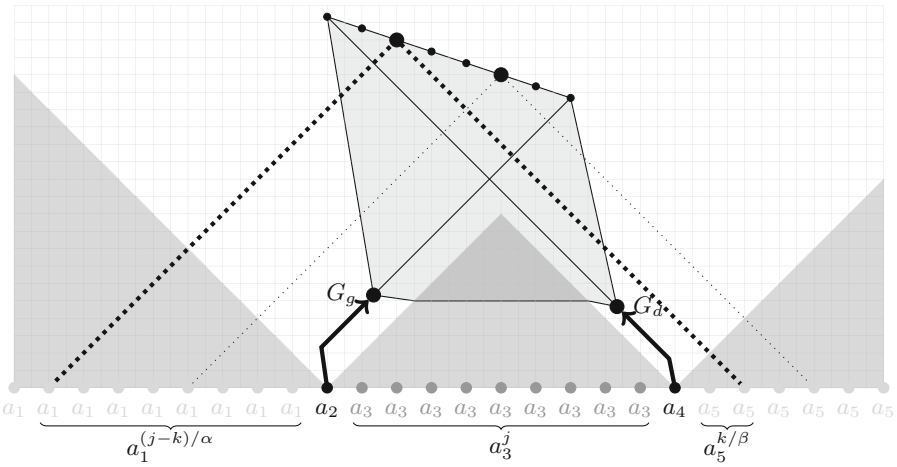


Fig. 5. Composition of two scalings

Now, the composition of these two scalings applies inside the cone of consequences of $(j+1, j+1)$: $\{c,t)\colon |c-j-1| \leq t-j-1\}$, intersection of the two cones issued from a_2 and a_4. See Fig. 5. It corresponds to the affine transformation with origin $(j+1, j+1)$ and matrix $M = M_g \times M_d = \begin{pmatrix} (\alpha+\beta)/(2\alpha\beta) & (\alpha-\beta)/(2\alpha\beta) \\ (\alpha-\beta)/(2\alpha\beta) & (\alpha+\beta)/(2\alpha\beta) \end{pmatrix}$. Thus the firing points $\{(1+2k+\gamma, 1+2j-\gamma)\colon k \in \mathbb{N}, 0 < k < j-\gamma\}$ being inside the cone of $(j+1, j+1)$ are mapped to the points $\{(1+j-(j-k-\gamma)/\alpha + k/\beta,$ $1+j+(j-k-\gamma)/\alpha+k/\beta)\colon k \in \mathbb{N}, 0 < k < j-\gamma\}$. Among these points, some of them match OCA's sites providing k/β and $(j-k-\gamma)/\alpha$ are integers. Moreover such a point, with $(j-k-\gamma)/\alpha, k/\beta \in \mathbb{N}$, corresponds to the output of the input subword that begins at $(2(j-k-\gamma)/\alpha, 0)$ and ends at $(2j+2+2k/\beta, 0)$, i.e., the subword $a_1^n a_2 a_3^j a_4 a_5^m$ with $n = (j-k-\gamma)/\alpha$, $m = k/\beta$. To conclude just note that $\alpha n + \beta m + \gamma = j$. □

Proposition 2. *For any homomorphism h on $\mathcal{A}$ such that neither* $\texttt{link}(h(a_1), h(a_2), h(a_3))$ *nor* $\texttt{link}(h(a_3), h(a_4), h(a_5))$ *holds, $h(L_{\alpha,\beta,\gamma})$ is a real time OCA language.*

Proof. One has to be careful because real time OCA is not closed under morphism: each recursively enumerable language can be written as the image of a

real time OCA language by a morphism [4]. But here we play with the image of languages with very simple structure.

First observe that $h(\{a_1^n a_2 a_3^{\alpha n+\beta m+\gamma} a_4 a_5^m : n < N_0 \text{ or } m < N_0\})$ is a linear CFL, so to show that $h(\{a_1^n a_2 a_3^{\alpha n+\beta m+\gamma} a_4 a_5^m : n, m \geq N_0\})$ is a real time OCA language will suffice. According to Proposition 1, there exists a real time OCA $(Q, \mathcal{A}, Q_{accept}, \delta)$ which recognizes the language $L = \{a_1^n a_2 a_3^{\alpha n+\beta m+\gamma} a_4 a_5^m : n, m \geq N_0\}$. Following Okhotin [10], L is defined by the linear conjunctive grammar $\mathcal{G} = (\mathcal{A}, \{S\} \cup \{A_q : q \in Q\}, S, \mathcal{R})$ where $\mathcal{R}$ contains the following rules:

$$
\begin{array}{l}
S \to A_q \text{ for all } q \in Q_{accept} \\
A_a \to a \text{ for all } a \in \mathcal{A} \\
A_{\delta(g,d)} \to A_g b \,\&\, c A_d \text{ for all } g, d \in Q \text{ and all } b, c \in \mathcal{A}
\end{array}
$$

Now let us make explicit, in the grammar rules, the bounded feature of the language. All words are of shape $a_1^+ a_2 a_3^+ a_4 a_5^+$. We denote by $\texttt{Follow}(i)$ the set of j such that a_j follows immediately a_i in the expression $a_1^+ a_2 a_3^+ a_4 a_5^+$: $\texttt{Follow}(1) = \{1, 2\}$, $\texttt{Follow}(2) = \{3\}$, $\texttt{Follow}(3) = \{3, 4\}$, $\texttt{Follow}(4) = \{5\}$, $\texttt{Follow}(5) = \{5\}$. Then we rewrite $\mathcal{G}$ to $\mathcal{G}' = (\mathcal{A}, \{S\} \cup \{(A_q, 1, 5) : q \in Q_{accept}\} \cup \{(A_q, i, j) : q \in Q \backslash Q_{accept}, 1 \leq i \leq j \leq 5\}, S, \mathcal{R}')$ where $\mathcal{R}'$ contains the following rules:

$$
\begin{array}{l}
S \to (A_q, 1, 5) \text{ for all } q \in Q_{accept} \\
(A_{a_i}, i, i) \to a_i \text{ for all } i = 1, \cdots, 5 \\
(A_{\delta(g,d)}, i, j) \to (A_g, i, s) a_j \,\&\, a_i (A_d, r, j) \text{ for all } g, d \in Q \text{ and all } i, j, r, s \\
\qquad\qquad \text{with } 1 \leq i \leq r \leq s \leq j \leq 5,\ r \in \texttt{Follow}(i),\ j \in \texttt{Follow}(s)
\end{array}
$$

To gain a better understanding of the last derivation rule, see Fig. 6, it depicts the corresponding OCA transition. With such refinements, we get that $L_{\mathcal{G}'}(A_q, i, j)$ is the subset of $L_{\mathcal{G}}(A_q)$ whose words begin with a_i, ends with a_j and which are factors of $a_1^+ a_2 a_3^+ a_4 a_5^+$.

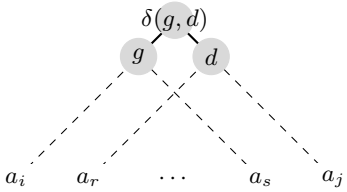


Fig. 6. Interpretation of the rule $(A_{\delta(g,d)}, i, j) \to (A_g, i, s) a_j \,\&\, a_i (A_d, r, j)$ with $i \leq r \leq s \leq j$, $r \in \texttt{Follow}(i)$, $j \in \texttt{Follow}(s)$

In addition, to meet later requirements, we replace all rules $(A_{a_i}, i, i) \to a_i$ with the rules $(A_{\delta(a_{i_1}, \cdots, a_{i_{N_0}})}, i_1, i_{N_0}) \to a_{i_1} \cdots a_{i_{N_0}}$ for all words $a_{i_1} \cdots a_{i_{N_0}}$

of length N_0 which are factors of $a_1^{N_0}a_2a_3^{N_0}a_4a_5^{N_0}$. It does not alter the expressiveness of the grammar.

Finally, in replacing each symbol a_i with its image $h(a_i)$, $\mathcal{G}$ is rewritten to $\mathcal{H} = (\Sigma, \{S\} \cup \{(A_q, i, j) \colon q \in Q, 1 \leq i \leq j \leq 5\}, S, \mathcal{R}'')$ where $\mathcal{R}''$ contains the following rules:

$$
\begin{aligned}
&S \to (A_q, 1, 5) \text{ for all } q \in Q_{accept}\\
&(A_{\delta(a_{i_1}, \cdots, a_{i_{N_0}})}, i_1, i_{N_0}) \to h(a_{i_1} \cdots a_{i_{N_0}}) \text{ for all words } a_{i_1} \cdots a_{i_{N_0}} \text{ of}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{length } N_0 \text{ which are factors of } a_1^{N_0}a_2a_3^{N_0}a_4a_5^{N_0}\\
&(A_{\delta(g,d)}, i, j) \to (A_g, i, s)h(a_j) \,\&\, h(a_i)(A_d, r, j) \text{ for all } g, d \in Q \text{ and all}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i, j, r, s \text{ with } 1 \leq i \leq r \leq s \leq j \leq 5,\ r \in \texttt{Follow}(i),\ j \in \texttt{Follow}(s)
\end{aligned}
$$

Clearly, $h(L_{\mathcal{G}}(S)) \subseteq L_{\mathcal{H}}(S)$. Let us ensure that they are equal and, more specifically, that $L_{\mathcal{H}}(A, i, j) \subseteq h(L_{\mathcal{G}}(A, i, j))$ for every variable (A, i, j). The proof is done by induction on the height of the parse trees. The inductive assumption is that all words generated within $\mathcal{H}$ by a tree of height h and root node (A, i, j) are images by h of words generated within $\mathcal{G}$ by a tree of height h and root node (A, i, j).

The base case. The trees of height 1 within $\mathcal{H}$ display the derivations $(A_q, i_1, i_{N_0}) \to h(a_{i_1} \cdots a_{i_{N_0}})$ whereas their counterparts within $\mathcal{G}$ display the derivations $(A_q, i_1, i_{N_0}) \to a_{i_1} \cdots a_{i_{N_0}}$.

The inductive step. We focus on the trees with root node $(A_q, 1, 5)$ and omit the ones with root (A_q, i, j) when $i > 1$ or $j < 5$ that are easier to handle. Given any tree T within $\mathcal{H}$ of height $h+1 > 1$ and root $(A_q, 1, 5)$. The root node expands into two subtrees of height at most h according to some rule $(A_q, 1, 5) \to (A_g, 1, s)h(a_5) \,\&\, h(a_1)(A_d, r, 5)$ where $\delta(g, d) = q$, s is 4 or 5 and r is 1 or 2. By assumption, $(A_g, 1, s)$ produces the image by h of a word $a_1^{n_1}a_2a_3^{n_2}a_4a_5^{n_3}$ with $n_1 > 0, n_2 > N_0, n_3 \geq 0$ and $(A_d, r, 5)$ produces the image by h of a word $a_1^{m_1}a_2a_3^{m_2}a_4a_5^{m_3}$ with $m_1 \geq 0, m_2 > N_0, m_3 > 0$. Hence the root node produces $h(a_1^{n_1}a_2a_3^{n_2}a_4a_5^{n_3+1}) = h(a_1^{m_1+1}a_2a_3^{m_2}a_4a_5^{m_3})$. Next we may notice that, in the proof of Lemma 1, the hypothesis that the first exponents n_1 and m_1 are greater than N_0 is not used, as well, (in handling the word backward) for the last exponents n_r and m_r. Bearing in mind that neither $\texttt{link}(h(a_1), h(a_2), h(a_3))$ nor $\texttt{link}(h(a_3), h(a_4), h(a_5))$ holds, all prerequisites to apply Lemma 1 are satisfied. So $n_1 = m_1 + 1, n_2 = m_2, n_3 + 1 = m_3$. It follows that $a_1^{n_1}a_2a_3^{n_2}a_4a_5^{n_3+1}$ is indeed represented by a tree within $\mathcal{G}$ structured as T. □

4 Conclusion

Based on the precise characterization of the poly-slender CFL in terms of Dyck loops given by Ilie, Rozenberg and Salomaa, we have shown that linear-slender CFL are real time OCA languages. More than the result in itself, the approach appears interesting. It mixes algorithmic constructions with grammar tools and combinatoric properties: Starting from a real time OCA recognizing a specific language, here a variant of Čulík's one, we make use of Okhotin's result to

translate it in terms of a linear conjunctive grammar. Then in modifying the grammar, we capture a wider set of real time OCA languages structured like the starting one.

Now the challenge would be to show that poly-slender CFL are also real time OCA languages, in other words, that all Dyck loops are real time OCA languages. For that purpose, a key step would be to exhibit a generalization of Čulík's construction to handle languages of shape $a_0 b_1^{n_1} a_1 b_2^{n_2} a_2 \cdots b_r^{n_r} a_r$ where the integers $n_1, \cdots, n_r$ are linear combinations of k integers and the symbols a_i, b_i are distinct.

References

1. Choffrut, C., Čulík II, K.: On real-time cellular automata and trellis automata. Acta Informatica **21**(4), 393–407 (1984)
2. Čulík II, K.: Variations of the firing squad problem and applications. Information Processing Letters **30**(3), 152–157 (1989)
3. Čulík II, K., Gruska, J., Salomaa, A.: Systolic trellis automata. I, II. International Journal Computer Mathematics **16**, 3–22 (1984)
4. Čulík II, K., Gruska, J., Salomaa, A.: Systolic trellis automata: Stability, decidability and complexity. Information and Control **71**(3), 218–230 (1986)
5. Dyer, C.R.: One-way bounded cellular automata. Information and Control **44**(3), 261–281 (1980)
6. Ginsburg, S., Spanier, E.H.: Bounded algol-like languages. Transactions of the American Mathematical Society **113**, 333–368 (1964)
7. Ilie, L., Rozenberg, G., Salomaa, A.: A characterization of poly-slender context-free languages. Theoretical Informatics and Applications **34**(1), 77–86 (2000)
8. Latteux, M., Thierrin, G.: On bounded context-free languages. Elektronische Informationsverarbeitung und Kybernetik **20**(1), 3–8 (1984)
9. Naji, M.: Ambiguity of context-free languages as a function of the word length. FB Informatik. Goethe Universität, Frankfurt am Main (1998)
10. Okhotin, A.: On the equivalence of linear conjunctive grammars and trellis automata. RAIRO Informatique Thorique et Applications **38**(1), 69–88 (2004)
11. Okhotin, A.: Comparing Linear Conjunctive Languages to Subfamilies of the Context-Free Languages. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 431–443. Springer, Heidelberg (2011)
12. Terrier, V.: On real time one-way cellular array. Theoretical Computer Science **141**(1–2), 331–335 (1995)
13. Terrier, V.: Language not recognizable in real time by one-way cellular automata. Theoretical Computer Science **156**(1–2), 281–287 (1996)
14. Terrier, V.: Closure properties of cellular automata. Theoretical Computer Science **352**(1–3), 97–107 (2006)

Author Index